UNIT VI: File System Management

File Management: Concept of File, Access methods, File types, File operation, Directory structures, directory implementation, File System structure, Allocation methods, Free-space management, efficiency and performance.

Syllabus Unit-6

Disk Management: Disk structure, Disk scheduling - FCFS, SSTF, SCAN, C-SCAN, LOOK, C-LOOK, Disk formatting, Boot-block, Bad block, I/O devices, Device controllers and Device driver.

2.0 Introduction

This unit mainly deals with file concepts in OS such as files, their operations and attributes. It also involves different file access methods along with various space and file allocation methods. It describes file system structure along with its implementation. It also focuses on directory structure in computer. It gives idea about different issues in secondary file storage system. It describes various disks scheduling algorithm. It also explains concept of recovery and log structure in file system. It explains how protection can be provided in OS. It gives idea about free space management in OS. At end of this chapter numerical based on disk scheduling algorithm are solved.

2.1 File

File is collection of inter-related information defined by its creator. Files are mapped by operating system on the physical devices which are generally non volatile, means content remains as it is although there is power failure. Computers generally stores information on storage media as magnetic and optical disk whose logical view can be provided by OS. File is considered as a smallest allotment of logical secondary storage with user perspective. Computer cannot store data on secondary storage devices unless they are written on file. Files generally contain data and programs (source code, Object code, etc.).

Every file has specific structures according to its types. File may also contain numeric, alphanumeric and binary information. File is considered to be a sequence of bits, bytes, line and records created by users. File is also called as entry point into directory. File is considered to be a backbone of any computing. Language translator makes use of file system to store reloadable, absolute and executable programs. File can be recognized by its unique name along with set of attributes as type, size, permission, etc.

A file is an object on a computer that stores data, information, settings, or commands used with a computer program. In a graphical user interface (GUI) such as Microsoft Windows, files display as icons that relate to the program that opens the file. For example, the picture is an icon associated with Adobe Acrobat PDF files. If this file was on your computer, double-clicking the icon in Windows would open that file in Adobe Acrobat or the PDF reader installed on the computer.

In computing, a file system or filesystem is used to control how data is stored and retrieved. Without a file system, information placed in a storage medium would be one large body of data with no way to tell where one piece of information stops and the next begins.

2.1.1 Types of Files

Generally we observe different files created in computer which may vary as per OSdevices. The various types of files are as follows:

1. Text File: It generally stores textual data which is character based and it has extension .txt
2. Source File: It contains the program written in specific language such as C, C++, Java, .Net,etc. having extension .c, .cpp and .java and many more.
3. Object File: It contains the non executable code which is in machine understandable formatcreated by compiler of specific language. It has extension .obj

4. Batch File: It is similar to script file in Linux, use to run multiple commands in sequence. It gives commands to command interpreter. It has extension .sh, .bat.
5. Executable File: It contains executable code which is in machine understandable format generated after linking object file with different library files. It has extension .exe.
6. Library File: It contains the information about routines for programmers, which helps to decrease load of developers. It has extension .lib, .dll
7. Archive File: It is a compressed file, which has extensions .arc, .zip, .rar, etc.
8. Multimedia File: It contains the multimedia information as audio, video and having extension .mp3, .avi, .mpeg, etc.
9. Special File: It contains information about devices and other required information.
Note: File can have other types also.

2.1.2 Attributes of File System

The attribute of file is called as metadata i.e. data about data, which describes the behavior of file. Each attribute can have one of two states set and cleared. The file attributes may vary from one operating system to another but generally consist these. File attributes are settings associated with computer files that grant or deny certain rights to how a user or the operating system can access that file. For example, IBM compatible computers running MS-DOS or Microsoft Windows have capabilities of having read, archive, system, and hidden attributes.

- **Name:** This attribute represents name of file given by its creator for sake of differentiating it from other files. E.g. def, xyz. Name attribute is user perspective for identification to file.
- **Identifier:** The system can identify every file based on its identifier which is a unique tag or number in non human readable form. This attribute is system perspective to the file for identification.
- **Type:** This attribute tells us what kind of data is stored in the file. e.g. if type of file is .txt then we can say it contains only text data, if type of file is archive then we can say it contain compressed data.
- **Location:** This attribute is nothing but the address of file that means it tells us where the file is stored. It can be traced by using path of file.
- **Size:** This attribute represents how much amount of data hold by particular file which can be measured in terms of bytes, K bytes, words or number of blocks.
- **Time, data and user identification:** This attribute tells us that when the specific file was created and last modified for sake of security and protection.
- **Usage Count:** This attribute represents the number of processes which are currently using the specific file.

Following also are some special attribute of file system.

- Read-only - Allows a file to be read, but nothing can be written to the file or changed.
- Archive - Tells Windows Backup to backup the file.
- System - System file.
- Hidden - File will not be shown when doing a regular dir from DOS.

2.1.3 Operations on File System

The operation on the file represents the action performed on the file, which can create, read, write, delete, modify and truncate. Operation on the file can be performed by operating system by using system call. Various operations performed on file are as follows:

- **Creation of File:** This operation allows user to create file on secondary storage as Hard disk. Sufficient space is required to create file on disk.
- **Writing a File:** This operation allows user to write specific data on file. While writing data user must ensure that data should match with type of file, i.e. audio data cannot be written on text file. The system must keep write pointer to location in file where next write is to take place.
- **Reading a File:** This operation allow user to read content of file which require read system call. The read system call requires name of file and where the next block of file should be kept. System needs to keep read pointer to location in file where next read is to take place.
- **Deleting a File:** This operation allows to delete file, which is already created by user. While deleting file it should be searched in directory. The space which gets free after deletion of file can be reused by other file. During delete operation logical structure of file gets deleted.
- **Truncate a File:** This operation allows user to delete only content of file. But structure of file remains as it is. After truncating file if user wishes to reuse that file structure then he can do it. The logical structure of file remains as it is till deletion of file.

Apart from these above mentioned basic operations user can perform other operation on file also which are as below. 1. Copy, 2. Renaming, 3. Appending, 4. Repositioning, 5. Locking

Note:-For every file operation OS provides specific system call to carry out operation.

2.2 File system structure

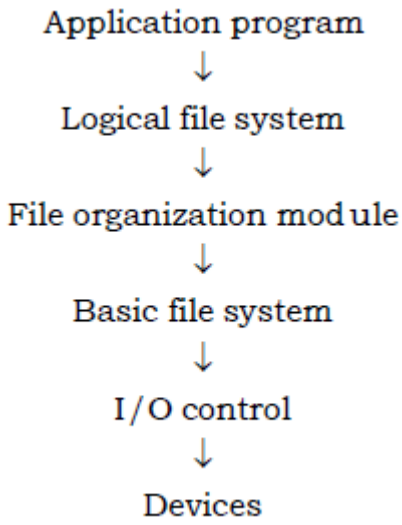
The file system structure is the most basic level of organization in an operating system. Almost all of the ways an operating system interacts with its users, applications & security model are depend upon the way it organizes files on storage devices. Providing common file system structure ensures users & programs to access & write files. File system break files into two logical categories Shareable vs. Non-sharable files Variable vs. Static files. Sharable files are those that can be accessed locally & by remote host & non-sharable files are only available locally. Variable files can be changed at any time but static files do not change. To provide efficient & convenient access to the disk, the OS imposes one or more file systems to allow data to be stared, located & retrieved easily. The file system itself is generally composed of many different levels as shown in figure 2.1.

Process:-

- When an application program asks for a file, the first request is directed to the logical file system. The logical file system contains the Meta data of the file and directory structure. If the application program doesn't have the required permissions of the file then this layer will throw an error. Logical file systems also verify the path to the file.
- Generally, files are divided into various logical blocks. Files are to be stored in the hard disk and to be retrieved from the hard disk. Hard disk is divided into various tracks and sectors. Therefore, in order to store and retrieve the files, the logical blocks need to be

mapped to physical blocks. This mapping is done by File organization module. It is also responsible for free space management.

- Once File organization module decided which physical block the application program needs, it passes this information to basic file system. The basic file system is responsible for issuing the commands to I/O control in order to fetch those blocks.
- I/O controls contain the codes by using which it can access hard disk. These codes are known as device drivers. I/O controls are also responsible for handling interrupts.



Fiure2.1. Layered file system

Each level in above design uses the features of lower levels to create new features for we by higher levels. Lowest level is I/O control which consists of device drivers & interrupts handlers the basic file system needs only to issue generic commands to appropriate device drives to read & write physical locks on disk. File organization module know about files & their logical blocks & physical blocks. File organization module can translate logical block address to physical block addresses for basic file system to transfer. Logical file system managers metadata information, which includes all of the file system structure except the actual data files.

2.3 File system Implementation:

File system stores several important data structures on the disk. A boot control block contains the information needed by system to boot an OS. A volume control block contains partition details, such as number of blocks in the partition, size of the blocks, free block count & free block pointers. A file control block contains details about ownership, size, permission, dates etc; **whose structure (figure 2.2) is shown below.**

File permission
File dates (create, access)
File owner, group, ACL
File size
File data blocks

Figure2.2. File Control Block

The **figure2.3** illustrates some of the interactions of file system components when files are created & used. When a new file is created, a new file control block is allocated & filled out with important information regarding the new file. When file is accessed during a program, the open () system call reads in the file control block information from disk & stores it in the system wide open file table.

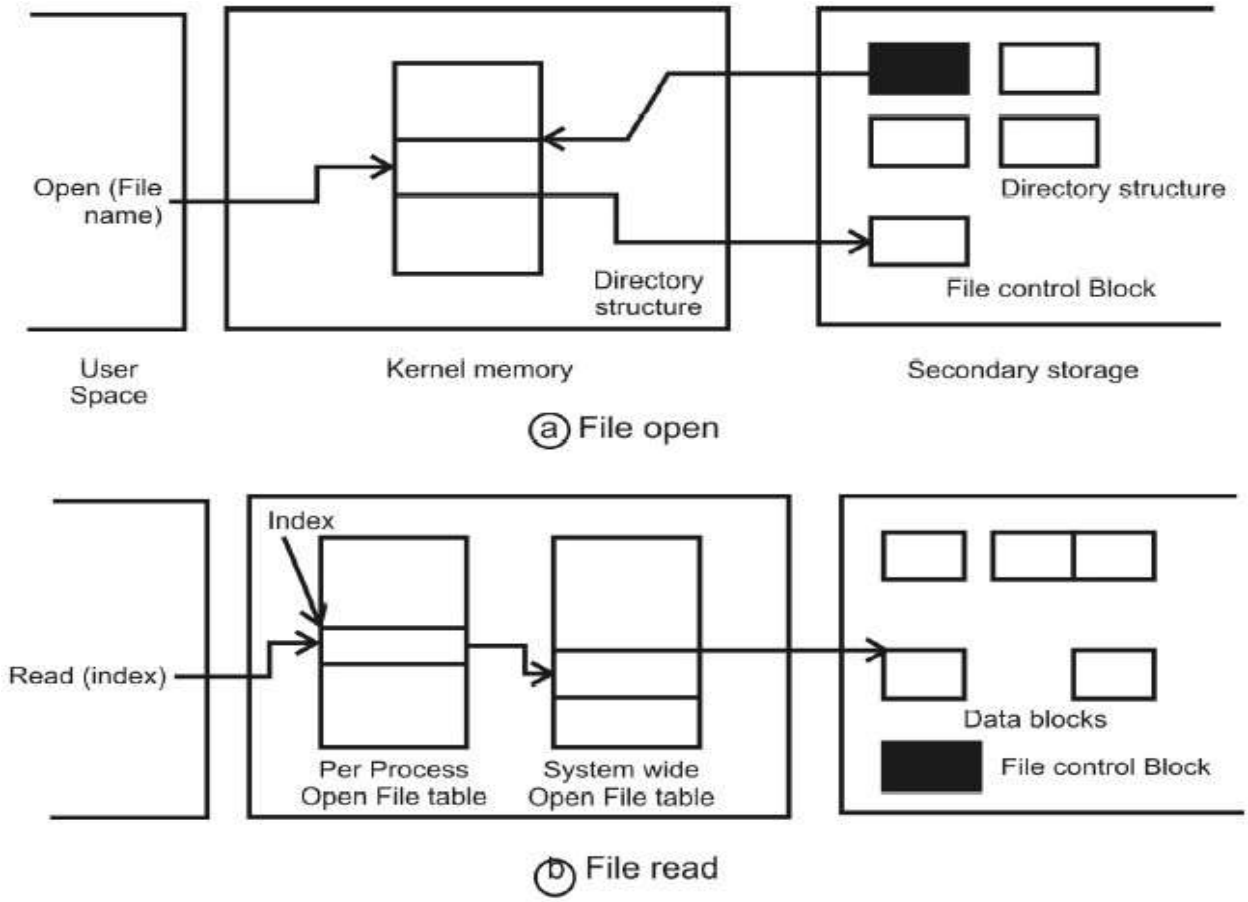


Figure2.3. In memory file system structure

If another process already has file open when a new request comes in for the same file & it is sharable, then counter in the system wide table is incremented. When a file is closed, per process table entry is release & counter in the system wide table is decremented.

2.3.1 Partitions & Mounting

Physical disks are commonly divided into smaller units called partitions. Partitions can either be used as raw devices or they can be formatted to hold file system. Partitions containing file system can only be accessed using file system structure by ordinary users. The boot block is accessed as part of raw partition, by the boot program prior to any OS being loaded. Modern

boot programs understand multiple OS & file system formats. The root partition contains the OS kernel & at least the key portion of OS needed to complete the boot process. At boot time the root partition is mounted & control is transferred from boot program to kernel.

2.3.2 Virtual file systems

A virtual file system provides a common interface to multiple different file system types. In addition, it provides for a unique identifier (*Vnode*) for files across the entire space, including file systems of different types. VFS in Linux is based upon 4 key object types.

- The inode object, representing an individual file.
- The file object, representing an open file.
- The superblock object, representing a file system.
- The entry object, representing directory entry.

Linux VFS provides a set of common functionalities for each file system using function pointers accessed through table. The figure 2.4 shows the schematic view of VFS.

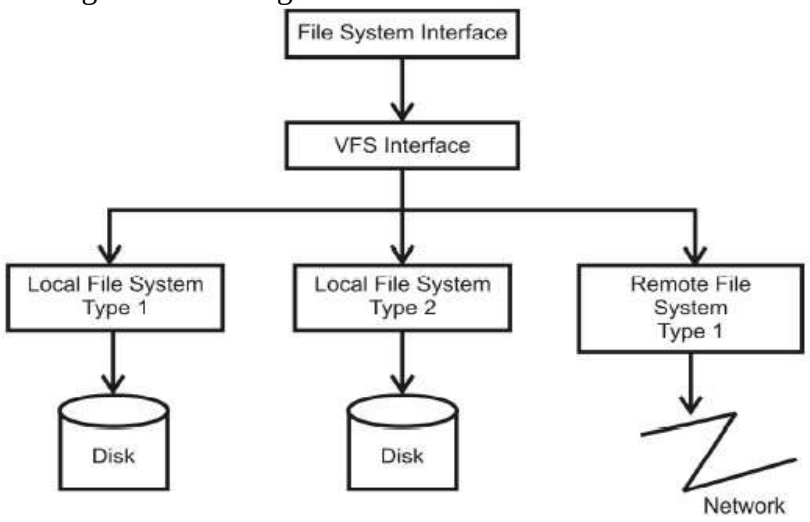


Figure2.4. Figure:-Schematic view of virtual file system

2.4 File Access Methods

The access method allows user to access data stored on disk and other external devices. An access method is also an application or hardware mechanism that moves data between computer and other external devices. Information in the file can be accessed in several ways which may vary according to system as sequential, random and other access methods.

2.4.1 Sequential Access

In this type of method data on disk can be accessed in predetermined order of sequence. Sequential access method is one of simplest access method. Information in file is processed one after another e.g. most common system software like compiler and editors. Read operation on file reads the next portion of file and automatically advances file pointer which track the I/O location. Similarly write operation appends to end of the file and advances to end of newly written data. Sequential access method is based on tape model. Sequential access is sometimes the only way of accessing data e.g. magnetic tape. The following model represents sequential access method.

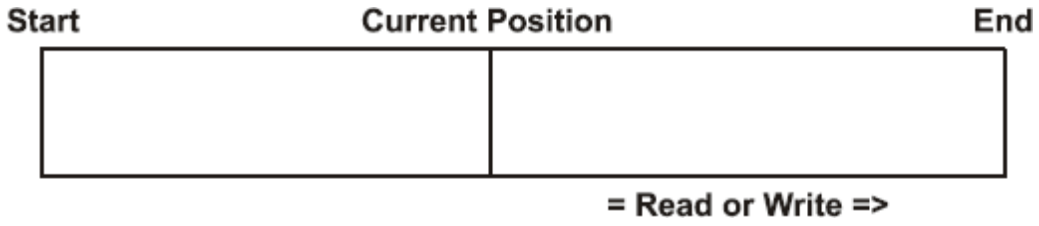


Figure 2.5 Sequential access method

Such a file can be reset to beginning and on some system program may be able to skip forward or backward of record. Basically it works on concept of read next and writes next. Figure 2.5 shows working.

2.4.2 Random Access or Direct Access or Relative Access

This method is based on disk model. This method allows reading file block in no particular (random) order. This method is mostly useful for immediate access to large amount of information e.g. in case of database. The block number provided by user to operating system is normally a relative block number. A relative block number is an index relative to the beginning of file. The use of relative block number allows OS to decide where the file should be placed and helps to prevent user from accessing portion of file system. In this method there is no restriction on reading and writing of file content. This method works like read n rather than read next where n is a block number. Likewise write n rather than write next. Following fig. shows working of random access method. Figure 2.6 shows working.

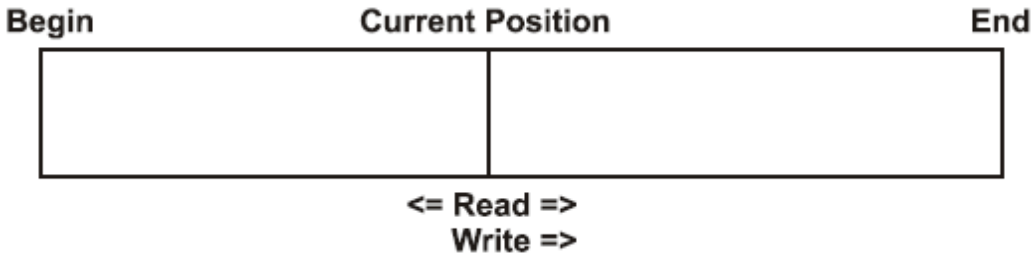


Figure 2.6. Direct Access Method

2.4.3 Other Access Method

Other than direct and sequential access method there can be other access methods which can be developed on top of direct access method. These methods generally involve the construction of an index for file. The index file contains pointer to various blocks. To find record in file we first search the index and then use pointer to access the file. (Figure 2.7 describe) This method is useful when we want to merge more files in index structure. This method allows direct access of file because pointer contains the reference to files. Index file can be varied in multiple levels. Most of the files on the internet have huge databases which are categorized in index fashion. We need to maintain the reference of relative file as relative file contains the actual content. It requires less amount of space as it can load run time. In this method searching of data can be done in faster way due to its file arrangement. This access method also called index sequential access method. The model of this method is shown in below fig.

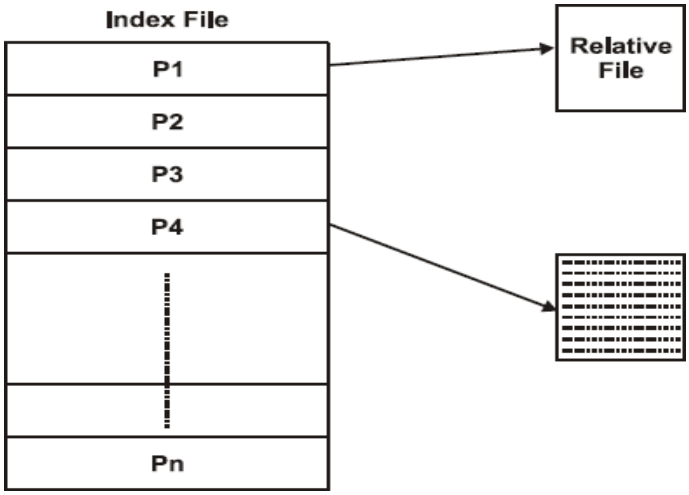


Figure 2.7 Other access method

2.5 Disk Space Management

Disk can be used in its whole for file system. Disk is a place where multiple files can be stored. Disk can be partitioned into number of parts. A file system can be created on each part of disk. These partitions also called volume. Each volume that contains file system must also contain information about files in system. This information is kept in entries of device directory. The device directory records information such as name, location, size and type of all files on that volume. Space on disk manages, to support fast access of file.

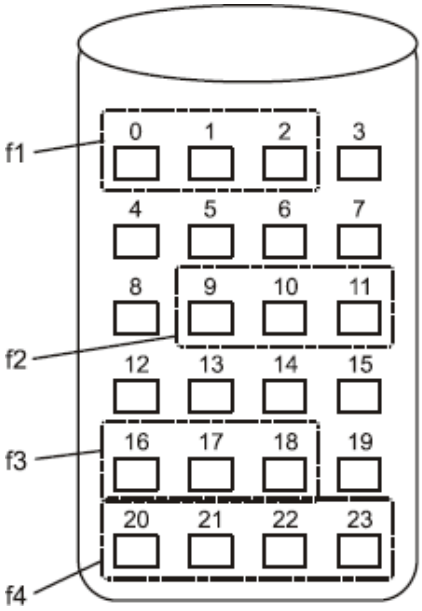
2.5.1 Disk Allocation Method

In order to utilize the disk space effectively three allocation methods are available which are as below.

1. **Contiguous allocation.**
2. **Linked allocation.**
3. **Index Allocation.**

1. Contiguous Allocation (Dynamic Allocation)

In this method each file occupies a set of contiguous blocks on the disk where disk address defines linear ordering on disk. Linear ordering access block b+1 after block b normally do not require head movement. In this case head needs only to move from one track to next so number of disk seeks required for accessing continuously allocated files is minimal.



Directory

File Name	Start Block	Length
f1	0	3
f2	9	3
f3	16	3
f4	20	4

Figure 2.8: Contiguous allocation method

The working of contiguous allocation method is given in figure 2.8 where contiguous allocation of file is defined by disk address and length of first block. If file is n blocks long and starts at location b then it occupies b, b+1, b+2. b+n-1. Here directory contain 4 file i.e. f1, f2, f3, f4 where file f1 starts from block 0 and has size 3. Similarly file f2, f3 and f4 have starting blocks and their respective lengths.

Advantages:

1. **Sequential Access:** Contiguous allocation supports sequential access of file blocks, where file system needs to remember disk address of last referred block and then, when necessary reads the next block.
2. **Easy Direct Access:** It is also possible to read nth block from the disk if block b is starting of file then b+1 can be immediately accessed.

Disadvantages

1. **External Fragmentation:** Contiguous allocation may suffer from the problem of external fragmentation as it requires only contiguous blocks to store file. E.g. suppose we want to store file f5 in above fig of length 7 then we cannot store it in above structure shown in fig. as no 7 contiguous blocks are available.
2. **Time Consuming:** As contiguous allocation based on sequential access it takes more time to access nth block. E.g. suppose we want to visit 5th block then we have to visit 4th block first and then 5th block.
3. **Need of compaction:** It requires re-collecting all free blocks together called as compaction which takes more time.
4. It's difficult to find free space as it requires only contiguous block.

2. Linked Allocation:

This method overcomes all the disadvantages of contiguous allocation. In this method every file is linked with a list of disk blocks where all disk blocks may be scattered all over disk and directory contains the pointer pointing to first and last block of file.

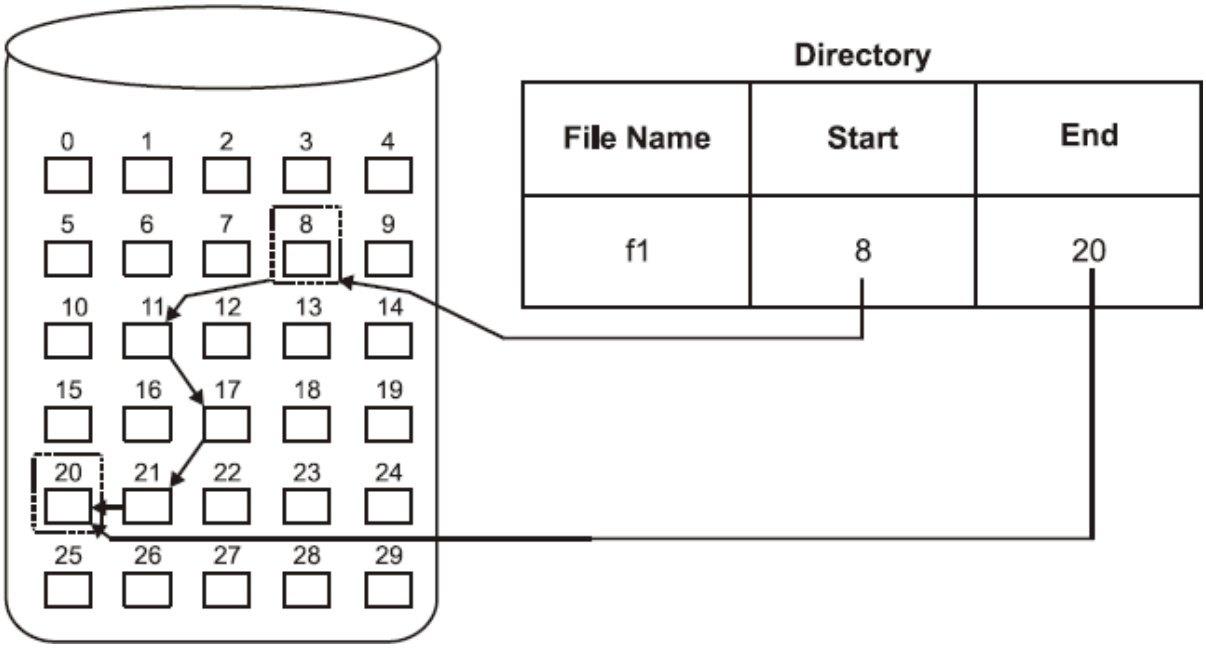


Figure 2.9 Link Allocation Method

The above fig. shows working of link allocation method where file f1 is of size 5 starts at block 8 and ends at block 20. The directory contains the starting and ending block numbers but it does not contain record of middle block numbers. Here every block contains pointer to next block i.e. in figure 2.9, block 8 contains pointer to block 11 and similar will be the case for other blocks. To create new file we simply create new entry in the directory. With link allocation each directory entry has a pointer to first block of file.

Advantages:

1. Linked allocation does not suffer from external fragmentation as it can occupy any free block available on disk unlike contiguous allocation.
2. No need to declare size of file when it is created.
3. There is no need of compaction technique as it does not face the problem of fragmentation.
4. This method allows logical to physical block mapping to remain simple but improve disk throughput and decrease space needed for block allocation.

Disadvantages:

1. **No Direct Access:** In this method we cannot access any block directly but we need to start from block specified in directory.
2. **More Storage Space Required:** Linked allocation requires more space as it stores both pointer to starting block and end block for every file.
3. **Reliability Problem:** As this method is totally based on pointer to access file, if any pointer is lost then next blocks cannot be accessed.

3. Index Allocation

In linked allocation direct access to any block of file is not possible as it visits all blocks in one file in sequential order, so this problem can be solved in index allocation by bringing all pointers together at one location called index block. Each file contains its own index block which

is an array of disk block addresses. Here directory for every file contains the address of index block. When file is created, all pointers in index block are set to null, when its block is written, a block is obtained from free space manager and it's address is put in the zth index block entry.

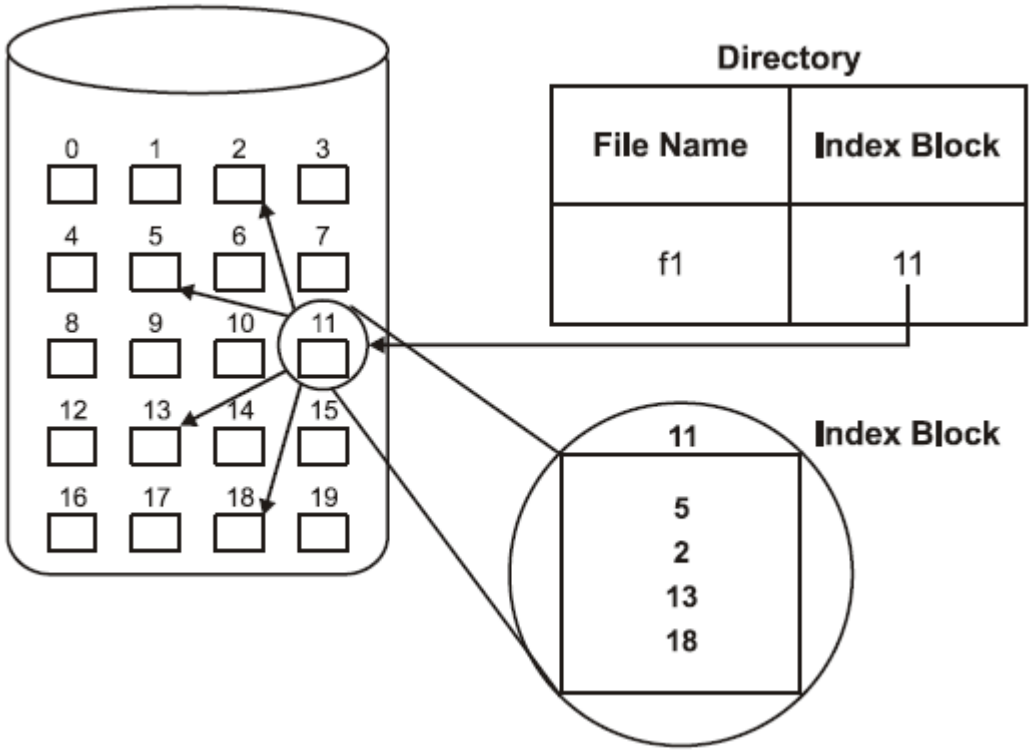


Figure 2.10: Index Allocation Method

The above fig. shows working of index allocation method where directory maintains only pointer to index block which ultimately contains the all pointers to blocks of file say F1 here. The size of file F1 is 4.

Advantages:

1. Index allocation supports direct access without suffering from external fragmentation.
2. This method does not require file allocation table (FAT).
3. No need to declare size of file in the beginning.

Disadvantages:

1. **Pointer Overhead:** Pointer overhead of index block is greater than pointer overhead of linked allocation.
2. **Index Block Overhead:** In this method every file must have separate index block.
3. **Reliability:** If pointer to index block is lost, then no block of file can be accessed.

Note:-Explain non contiguous allocation strategies in disk space management system. As a size of file either grows or shrinks over the time and also user rarely knows in advance about how large their file will be. But contiguous allocation method requires that the size of file should be estimated at the time of its creation. So at that situation non contiguous allocation strategies are useful as:

1. Linked Allocation
2. Index Allocation

Note: Same explanation given in previous question for linked and index allocation can be written here.

2.6 Directory Structure

Directory acts as a container which organizes the files into hierarchical structure. Directory is also called minidisk, if it has at least one partition. It can store number of files called as volume. Most important directory is root "/". Root directory is mounted at boot time and it contains base system necessary to prepare OS for multiuser operation. Root directory is called as mount point to other files. In file system top level directory is called root while directory below to another is called subdirectory and directory above subdirectory is called parent directory. In DOS and Windows root directory is indicated by backslash (\).

2.6.1 Different Directory Structures

There are various types of directory structures which are shown below.

1. Single level directory
2. Two level directory
3. Tree structure directory
4. Acyclic graph directory

1. Single Level Directory

It's a one of the simplest directory structure in which all files are contained in same directory, which is easy to understand and maintain.

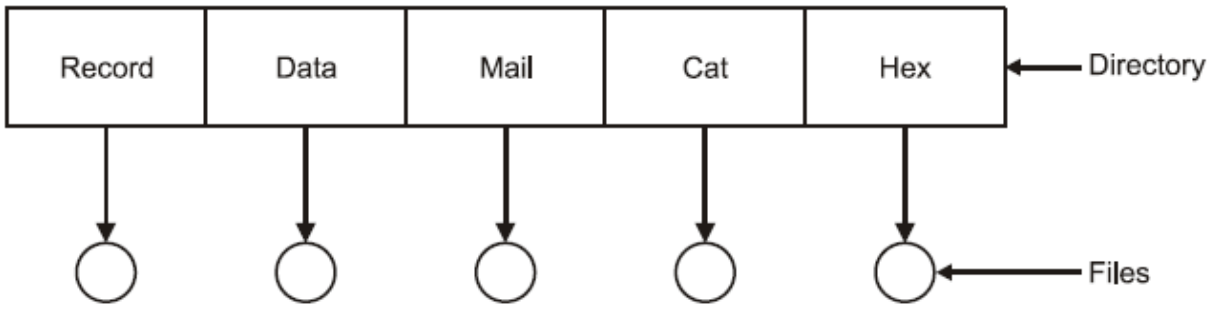


Figure 2.11: Single Level Directory

The figure 2.11 shows structure of single level directory, in which there is only one level of directory. The directory called record can contain number of files, but we cannot create more subdirectories. Users can create directory as they require, but all files in directory must have unique names, two users can not give same name to their files.

Disadvantages:

1. Naming Problem: User may find problem to remember the file in directory as different users refer same directory.

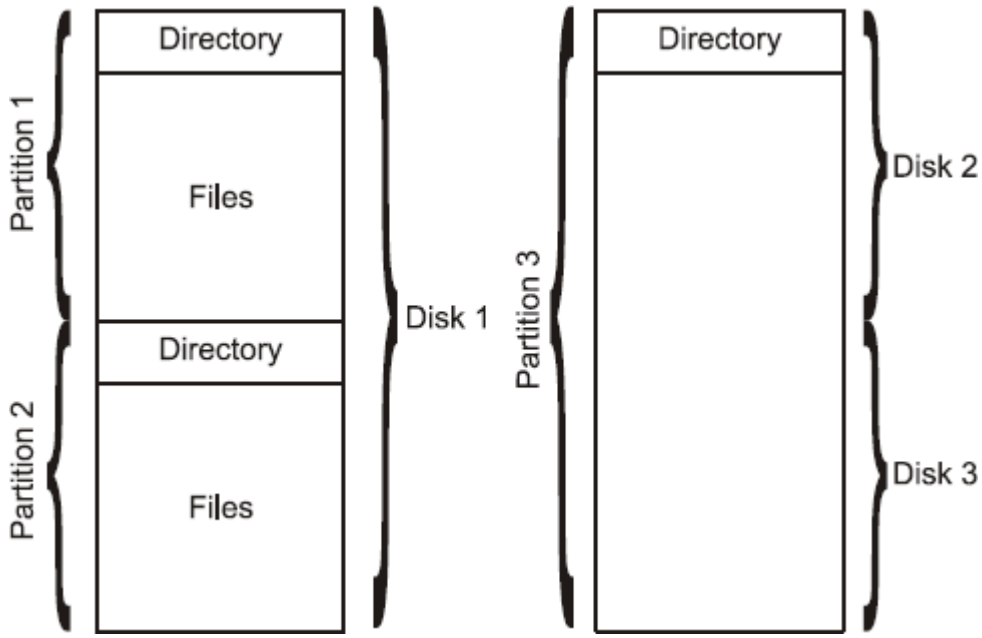


Figure 2.12: Directory Structure

The figure 2.12 shows directory structure where partitions are used to provide several separate areas within one disk, each treated as a separate storage device while other system allows partition to be larger than disk. So user needs to be concerned with logical directory and file structure and he ignores the problem of physically allocating space for file. Various operations to be performed on a directory are as follows, Search for file, Creation of file, Deletion of file, Renaming of file, Traversing file system, Listing of Directory.

2. High Overhead: It leads to high overhead on single directory due to increase in the size of directory.

3. Grouping Problem: As single level directory does not support grouping facility it leads to some kind of problem when hundreds of files are residing in single directory.

4. Scalability Problem: Due to single level it cannot accommodate more files in same directory.

2. Two Level Directory

A single level directory often leads to confusion of file names for different users, for that effective solution is to create separate directory. This directory structure supports to create separate directories to the users. Separate level directory are called upper level directory or user file directory (UFD).

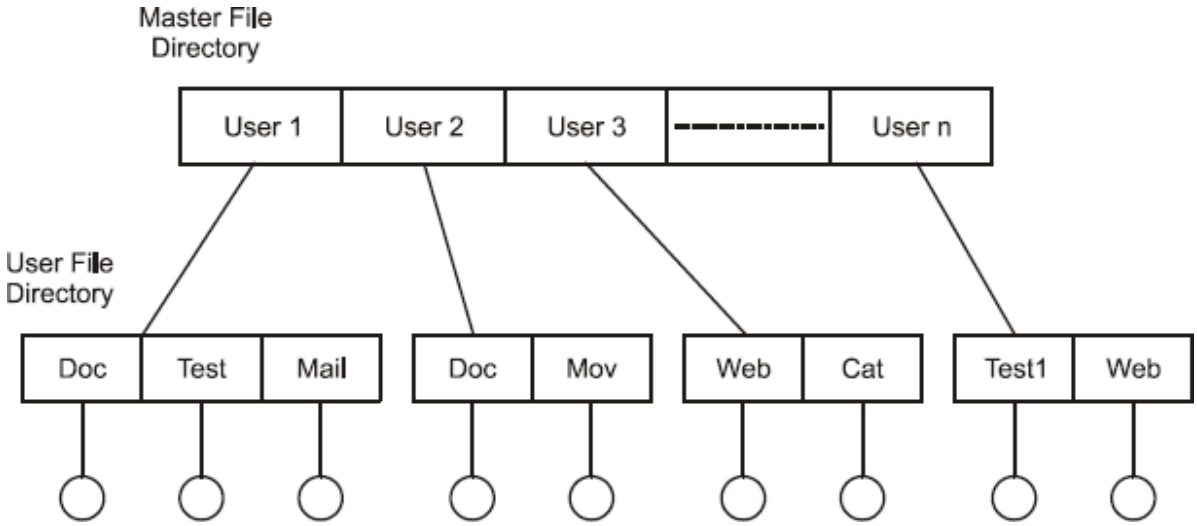


Figure 2.13: Two level directory structure

The figure 2.13 shows two level directory structures which consist of master file directory and user file directory. A user has total control and access to its own directory. User file directory must be created and deleted as necessary. It solves naming problem raised in single level directory structure. This system is effective for searching. It supports isolation of users and sharing of other user files. This structure effectively isolates one user from another but this isolation is advantages when users are completely independent to each other.

3. Tree Structure Directory

It's an extension of two level directory structures. It allows users to create subdirectories with arbitrary height and organize their files accordingly. It's a most common directory structure. This structure provides efficient searching of files. Tree structure directory provides grouping capability. In this structure naming problem of files is totally solved. It supports both directories i.e. current and working directory as well it supports both paths i.e. absolute and relative. Absolute path begins from root to end of file and relative path defines a path from current directory. This structure has root directory and every file in system has unique path name.

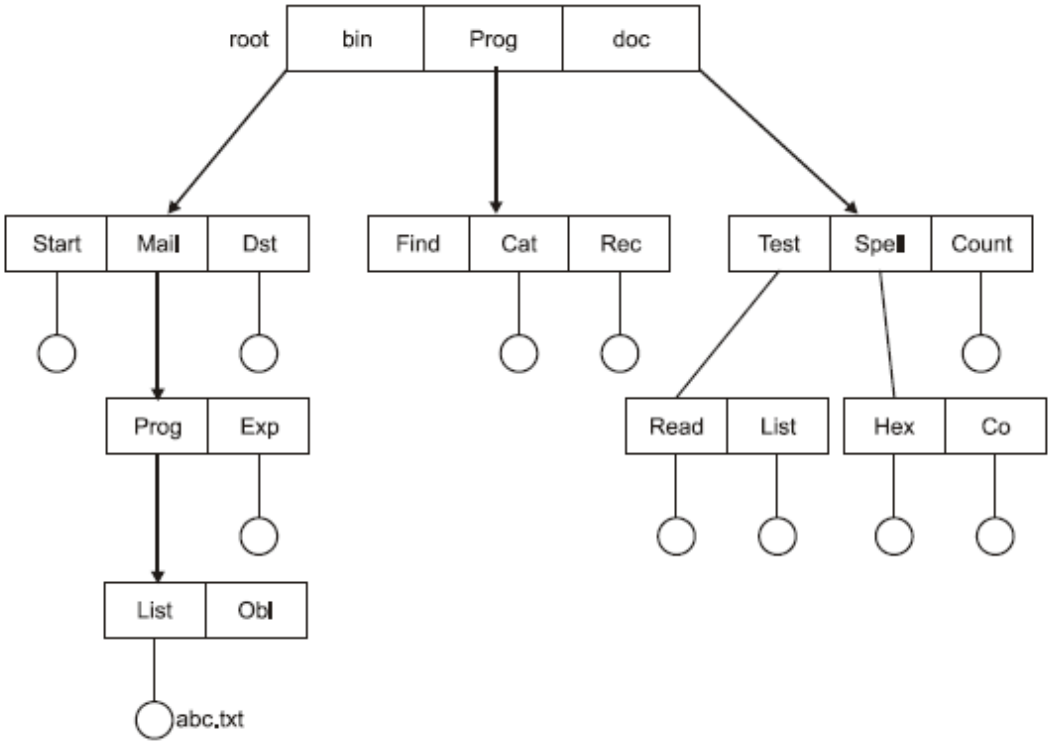


Figure 2.14: Tree Structure Directory

Above figure 2.14 shows tree structure directory in which if user wants to access abc.txt file for that absolute path which is root/bin/mail/prg/list/abc.txt i.e. from root to specify file and relative path is Prog/List/abc.txt if current directory is Prog. It provides more flexibility to create directory level with grouping mechanism. An interesting policy in tree structure directory concerns how to handle deletion of directory if it is empty then it simply can be deleted, but if directory to be deleted is not empty then to delete a directory, user must first delete all files in that directory. With tree structure directory, users can be allowed to access in addition to their files, the files of other user. A path to a file in tree structure directory can be longer than path in two level directories.

4. Acyclic Graph Directory

The tree structure and two level directory structures face problem while sharing of files among various users. This problem has been overcome by acyclic graph directory structure. Acyclic graph is a graph with no cycle which allows sharing subdirectories and files. Consider e.g. two developers working on same project and they often require access to same subdirectory with file as that file is dependable to each developers work. Since both developers are equally responsible for project, both want Subdirectory to be in their own directories, so common subdirectory should be shared. Sharing is important for subdirectories as new file created by one person will automatically appear in all shared subdirectories. When people working in team, all files they want to share can be put into one directory.

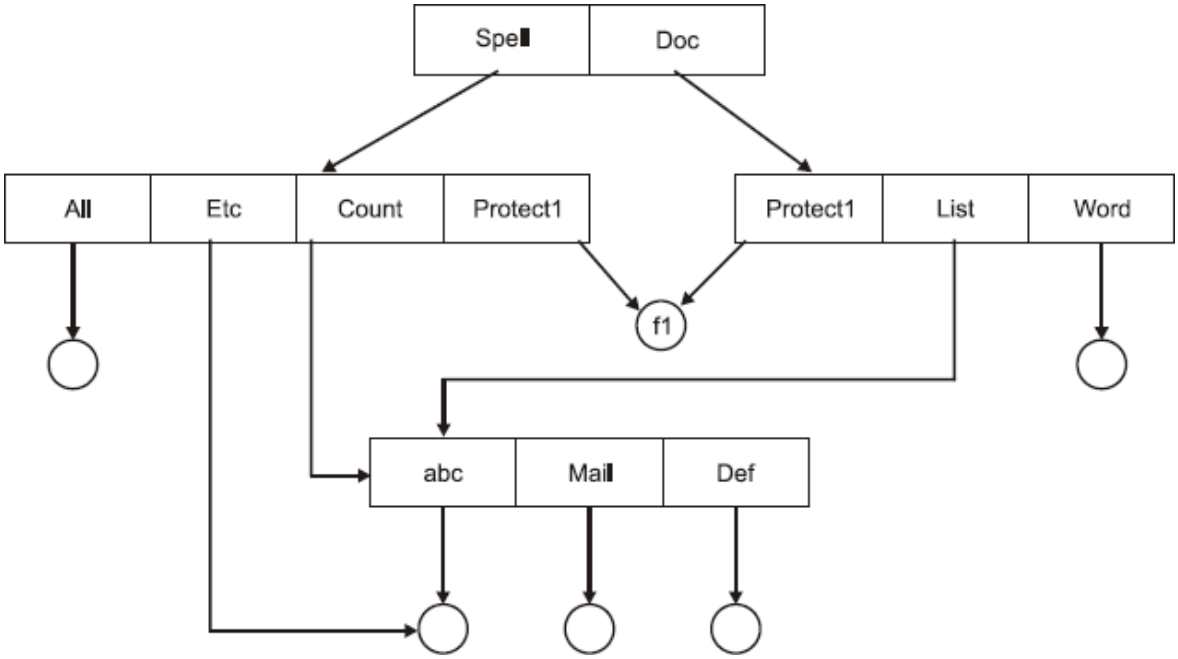


Figure 2.15: Acyclic Graph Directory

The figure 2.15 shows acyclic graph directory structure where file f1 is shared by two developers working on same project. With shared file only one actual file exists so any changes made by one person are immediately visible to other user. The primary advantage of this structure is, it is more flexible than simple tree structure. This implementation is possible by using symbolic link duplicate directory entries

Drawbacks:-

1. **Existence of Multiple Absolute Paths:** For shared file there may exist multiple paths, which may create problem when user wants to traverse that file.
2. Problem in deletion of file.

   if a file gets deleted in acyclic graph structured directory system, then

   a. In the case of soft link, the file just gets deleted and we are left with a dangling pointer.
   b. In the case of hard link, the actual file will be deleted only if all the references to it gets deleted.
   Example of soft link is creation of shortcut where size of shortcut is small and in case of hard link both file have same size.
3. **Dangling of Pointer:** It may face problem when deletion of shared file occurs.
4. **Prevention of Cycle Formation:** In this approach user must ensure that, there is no formation of cycle in graph.

2.6.2 Directory Implementation

There are many ways to implement directory but some of very popular ways to implement directories are:

1. Linear List

2. Hash Table

1. Linear List: It's a one of simplest ways of implementing directory which uses linear list and file names with pointer to data blocks. Linear list of directory entries require a linear search to find particular entry but it takes more time. Here all operations of file like create, delete needs to search directory to ensure that no other file is existing in same directory. A linked list can be used to decrease time to delete file. A real disadvantage of linear list is linear search.

2. Hash Table:

It's another way to implement directory. Hash table takes value computed from file name and return a pointer to file name in linear list. So use of pointer saves time of searching file. Some problem with Hash table is its fixed structure, size and dependency on hash function. It is much faster than linear search in linear list. It is more secured than linear list.

2.6.3 Device Directory

A device or special file is an interface for a device driver that appears in a file system as if it were an ordinary file. This device directory structure contains all required information for devices. The oracle Solaris OS includes both \dev and \devices directory for device drivers. Almost all drivers in the \dev directory are like to \devices directory. \dev directory is UNIX standard. Device files often provide simple interface to peripheral devices such as printer and serial port, but they can also be used to access specific resource on these devices. Each device directory contains collection of information and statistics

As

- **Cache** - The device cache
- **Capacity** - The capacity of device is 512 byte blocks
- **Driver** - The driver and version used to control device
- **Media** - The type of device such as disk
- **Model** - The model name or number of devices
- **Setting** - Collection of current device parameter

The content of device directory varies according to type of device connected.

2.7 Log Structured File System

A log structured file system is a file system in which data and metadata are written sequentially to a circular buffer called Log. The design was implemented in 1992. Conventional file system tends to log out files with great care for spatial locality and makes in place changes to their data structure in order to perform well on optical and magnetic disk. Design of log-structure system is based on hypothesis that will no longer be effective because of increasing memory size of modern computer. Log structure treats it's storage as a circular log and write sequentially to head of log.

Advantages:

- It helps to improve disk seek time.
- It reduces disk memory trip for fetching data from disk and loading into memory.

Disadvantages:

- It is costlier due to metadata storage
- More rotational latency.

The following are the data structures used in the LFS implementation.

- **Inodes:**
  As in Unix, inodes contain physical block pointers to files.

- **Inode Map:**
  This table indicates the location of each inode on the disk. The inode map is written in the segment itself.
- **Segment Summary:**
  This maintains information about each block in the segment.
- **Segment Usage Table:**
  This tells us the amount of data on a block.

2.8 Recovery

In data management, recovery is process that involves copying backup files from secondary storage to another disk. Recovery is performed in order to return data to its original condition if file is damaged. The most common data recovery scenario involves an OS failure in which case the goal is simply to copy all wanted files to another disk. This can be easily accomplished using file manager. Another scenario is disk level failure such as disk partition or hard disk failure, in any of these cases data cannot be easily read. Hardware and software based recovery of damage area. To check data damage consistency checking technique is used in which consistency checker checks data in directory structure. Backup and restore is required for failure of magnetic disk.

1. **Full backup** - It will copy all file to a secondary storage to another disk.
2. **Regular backup** - It will take daily backup of data file.
3. **Permanent Backup** - It will take total backup for permanently.
4. **Incremental Backup** - It will take backup periodically.
5. Various ways of taking backup may vary as per user's requirement.

2.9 Disk Scheduling

Disk scheduling is the process of arranging set of requests waiting/pending in request queue to access disk. Disk is a secondary storage on which data is stored in permanent manner. One of the responsibilities of OS is to use hardware efficiently. To use the disk by OS efficiently, the access time should be less and disk bandwidth should be large. In operating systems, seek time is very important. Since all device requests are linked in queues, the seek time is increased causing the system to slow down. Disk Scheduling Algorithms are used to reduce the total seek time of any request

Disk Scheduling: As we know that on a single Computer we can Perform Many Operations at a Time so that Management is also required on all the Running Processes those are running on the System at a Time. With the help or Advent of the Multi-programming we can Execute Many Programs at a Time. So to Control and provide the Memory to all the Processes Operating System uses the Concept of Disk Scheduling.

Objectives of Disk Scheduling are:-

- To minimize the access time.
- To maximize the throughput.
- To provide fairness to all requests.

Issues in Disk Scheduling

- Selecting disk request from queue of I/O requests.
- Some disk requests may have to wait for longer time before being served.
- Decide when to process the I/O request.

Disk:-

Generally, a disk is a round plate on which data can be encoded /stored. Disk is a collection of plates called platters. Platters are also divided into different tracks and track is also divided into different sectors.

Track, The (circular) area on a disk platter which can be accessed without moving the access arm of the drive is called track. Information is stored on a disk surface in concentric circles of small width, for each having a distinct diameter. Each circle is called a track.

Sectors**,** each track is further broken down into smaller units called sectors. As sector is the basic unit of data storage on a hard disk. A single track typically can have thousands of sectors and each sector can hold more than 512 bytes of data. A few additional bytes are required for control structures and error detection and correction.

Heads, Every hard drive consists of platters and read-write heads. If a drive has four platters, it usually has eight read-write heads, one on the top and bottom of each platter. The head value is the number of read-write heads in the drive.

A cylinder, each platter is divided into tracks. The cylinder value is the number of tracks on one side of each platter. There is the same number of cylinders on each side of each platter. The sector value is the number of sectors in each cylinder (or track), each sector consisting of (normally) 512 bytes. **Figure 2.16 shows the pictorial structure of disk.**

There are two basic types of disks: magnetic disks and optical disks. On magnetic disks, data is encoded as microscopic magnetized needles on the disk's surface. We can record and erase data on a magnetic disk any number of times, just as we can with a cassette tape. Whereas, optical disk records data by burning microscopic holes in the surface of the disk with a laser. To read the disk, another laser beam shines on the disk and detects the holes by changes in the reflection pattern. Figure 2.16 shows the structure of disk where, arm assembly, disk, and other scheme are described.

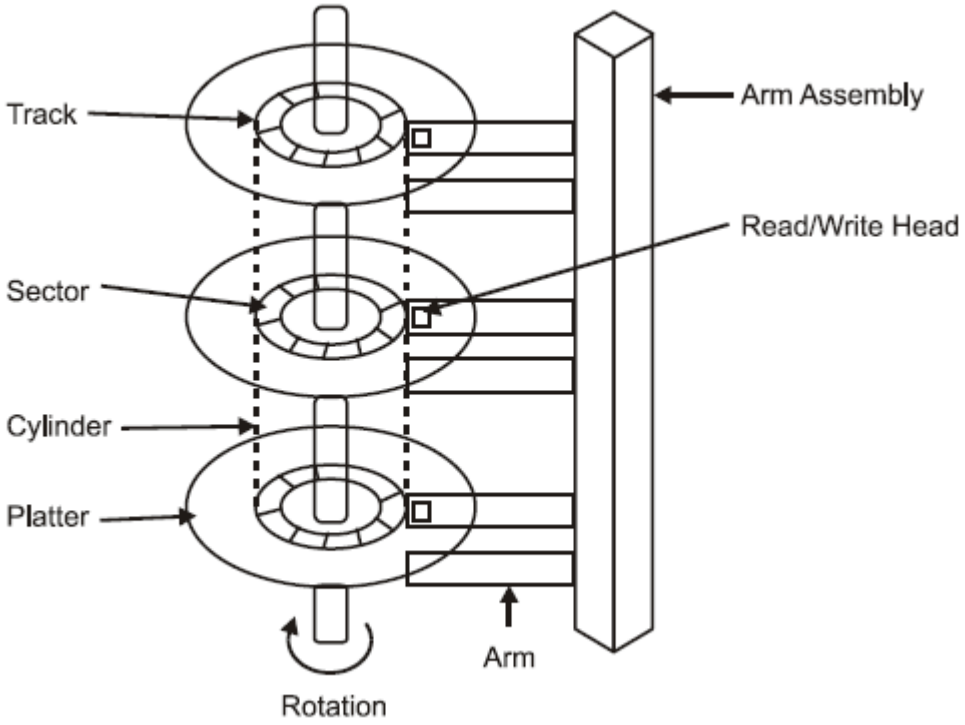


Figure 2.16: Structure of Hard Disk

Basic terms related to Hard Disk

- **Seek Time -** It's a time for disk arm to move the head to cylinder containing the desired sector.
- **Rotational Latency -** It's an additional time for disk to rotate the desired sector to disk head.
- **Disk Bandwidth -** It's a total number of bytes transferred to total time between the first request for service and completion of last transfer.

2.9.1 Disk Caching

A disk cache is a mechanism for improve the time it takes to read from or write to a hard disk. Today, the disk cache is usually included as part of the hard disk. The disk cache holds data that has recently been read and, in some cases, adjacent data areas that are likely to be accessed next. Write caching is also provided with some disk caches.

Disk cache is portion of RAM used to speed up access to data on disk. It also acts as a general purpose RAM in computer that is reserved for use by disk drive. Hard disk caches are more effective but they are more expensive and small. Disk cache stores copies of frequently used data on disk so that it can read data without referring to disk. We can say disk cache is a mechanism which saves time it takes to read from or write to hard disk. Today disk cache is included as part of hard disk. The main advantage of disk cache is that it improves disk access time and response time to application.

2.9.2 Disk Scheduling Algorithms

In order to provide disk access for set of requests waiting/pending in request queue in sequential manner, various algorithms are proposed which consider different criteria while providing disk access to set off request. Different disk scheduling algorithms are defined below.

- FCFS (First Come First Serve) scheduling
- SSTF (Shortest Seek Time First) scheduling
- SCAN Scheduling
- C-SCAN Scheduling
- LOOK Scheduling
- C-LOOK Scheduling

1. FCFS Scheduling:

This is one of the simplest forms of disk scheduling, which serves the request in first come first serve manner. This algorithm is intrinsically fair but does not provide fastest service. Consider example, disk queue with request for I/O to blocks on cylinders, in order 98, 183, 37, 122, 14, 124, 65, 67 if disk head is initially at cylinder 53. Calculate required seek time.

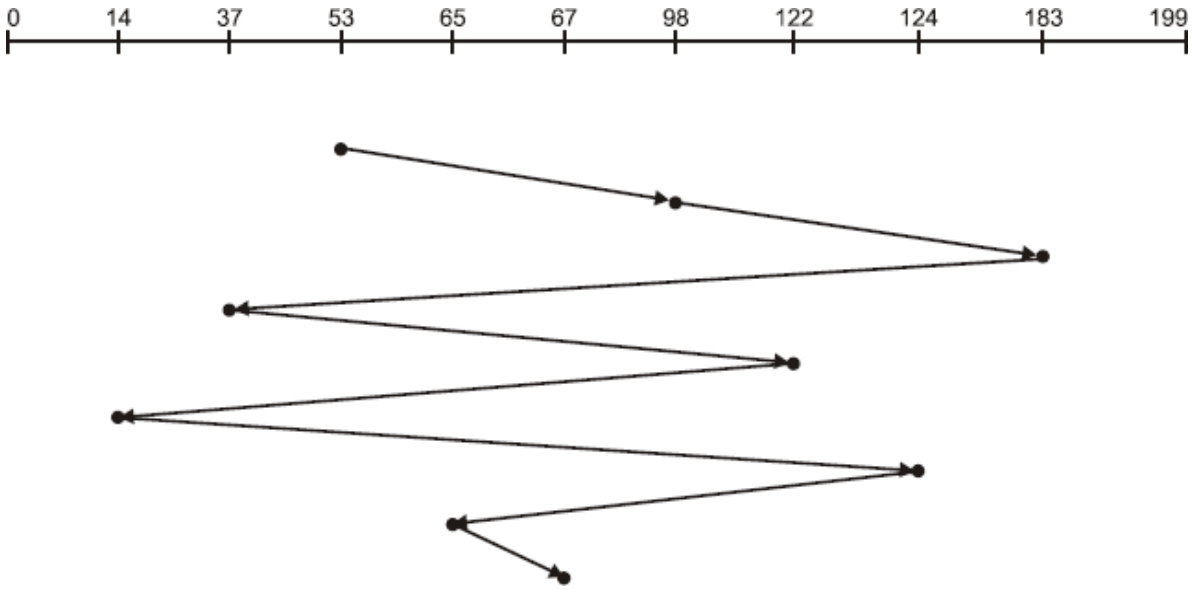


Figure 2.17: FCFS Disk Scheduling Algorithm

This algorithm mechanism is, it will first move from 53 to 98 because 98 is first request, then to 183, 37, 122, 14, 124, 65 and finally to 67. That means it serves request in same order in that has arrived. Figure 2.17 shows the movement of read write head over cylinder.
Total seek time or head movement is calculated as below,

Calculations:-

= [98-53]+[183-98]+[183-37]+[122-37]+[122-14]+[124-14]+[124-65]+[67-65]
= 45+85+146+85+108+110+59+2
= 640

Advantages-

- It is simple, easy to understand and implement.
- It does not cause starvation to any request.

Disadvantages-

- It results in increased total seek time.
- It is inefficient.

2. SSTF Scheduling:

This algorithm provides service to the request which is close to current head position, before moving head away to service other request, that means this algorithm gives preference to

request having shortest seek time from current head to target head. Since seek time increases with number of cylinders traverse by head, this algorithm chooses pending request closest to current head position. This algorithm overcomes the problem of FCFS of more seek time. But by the same time this algorithm may suffer from the problem of starvation. For some of the requests as they have more sought time compared to other. This algorithm is a form of SJF scheduling Consider same problem, following will be the scheduling graph obtained by using SSTF algorithm.

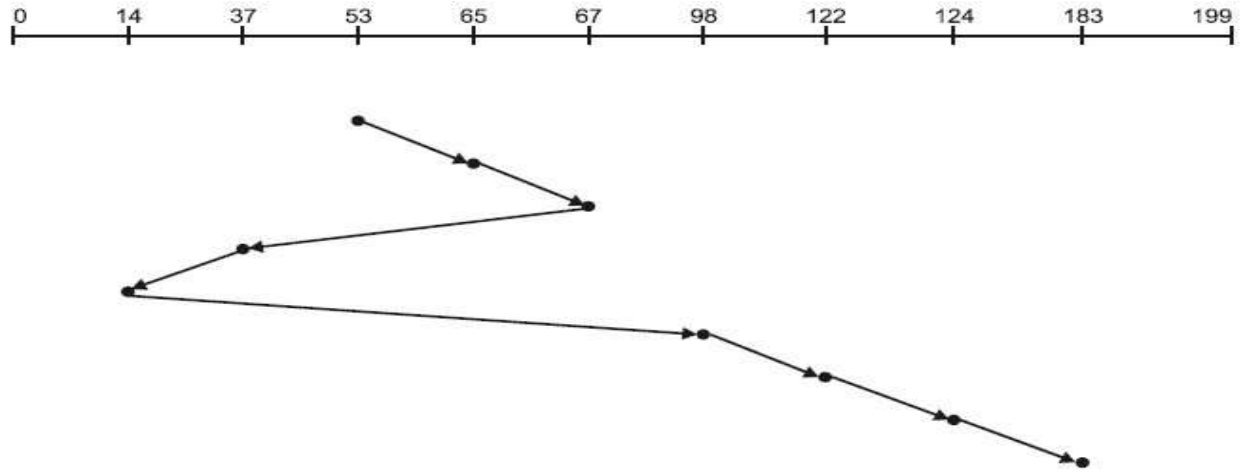


Figure 2.18: SSTF Disk Scheduling

As current head position is 53 then it will check for request closest to current head position 53 to both sides. Among set of the requests it is found to be 65 is closure to 53. Then it will search for request closest to 65 and i.e. 67, so in this way it will traverse all requests in order 37, 14, 98, 122, 124, and finally 183. Figure 2.18 shows the movement of read write head over cylinder.

Total head movements for SSTF is

Calculations:-
= [65-53]+[67-65]+[67-37]+[37-14]+[98-14]+[122-98]+[124-122]+[183-124]
= 12+2+30+23+84+24+2+59
= 236

Advantages-

- It reduces the total seek time as compared to FCFS.
- It provides increased throughput.
- It provides less average response time and waiting time.

Disadvantages-

- There is an overhead of finding out the closest request.
- The requests which are far from the head might starve for the CPU.
- It provides high variance in response time and waiting time.
- Switching the direction of head frequently slows down the algorithm.

3. SCAN Scheduling:

In this algorithm disk arm starts at one end of disk and moves towards other end, serving request as it reaches each cylinder, until it gets to other end of disk. Once it reaches to other end, direction of head movement is reversed and serving continues. Head continuously scan back and forth across disk. This algorithm is also called elevator (lift) algorithm as it behaves in same way as that of elevator in building.

Note:-

- ***In this algorithm it is not given in which direction head should move initially from current position, either to left or right so head should move towards that direction from where that end is nearest to current position.***
- ***If direction is given then follow the direction and if not choose any direction***
- ***Move the direction where the density of request is high***

Consider same example.

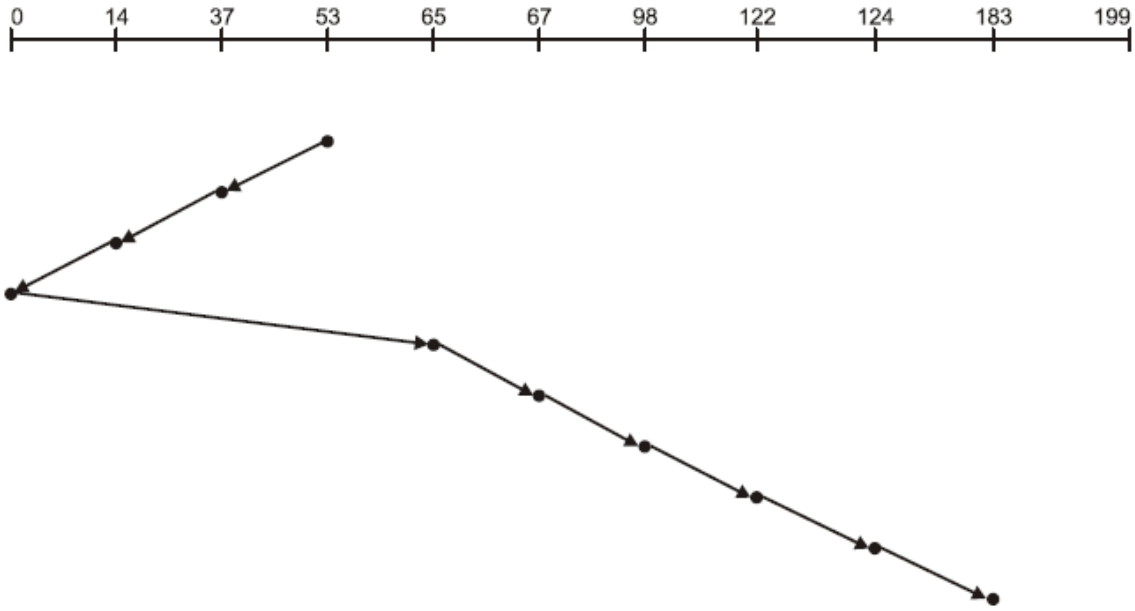


Figure 2.19: SCAN Scheduling

The above fig. shows graph for SCAN scheduling in which head it is currently at position 53 then moves towards left end as this end is nearest to current position and serves 37, 14 and finally it goes to position 0. Then It will reverse back and move towards other end of disk serving requests at 65, 67, 98, 122, 124 and finally 183 and stops there. Figure 2.19 shows the movement of read write head over cylinder.

Total head movements for above problem is

Calculations:-
= [53 - 37] + [37 - 14] + [14 - 0] + [65 - 0] + [67 - 65] + [98 - 67] + [122 - 98] + [124-122]+[183-124]
= 16+33+14+65+2+31+24+2+59
= 236

Advantages-

- It is simple, easy to understand and implement.
- It does not lead to starvation.
- It provides low variance in response time and waiting time.

Disadvantages-

- It causes long waiting time for the cylinders just visited by the head.
- It causes the head to move till the end of the disk even if there are no requests to be serviced.

4. C-SCAN Scheduling

Circular scan is variation of SCAN designed to provide more uniform wait time. Like SCAN, C-SCAN moves head from one end of disk to other serving request along the way. Once head reaches to one end it immediately returns to beginning of disk without serving a request on return trip. Then again it reverses direction and serves the remaining request. In this algorithm head should move towards that end initially, which has heaviest density of request. Consider same example. (Example special/private vehicle)

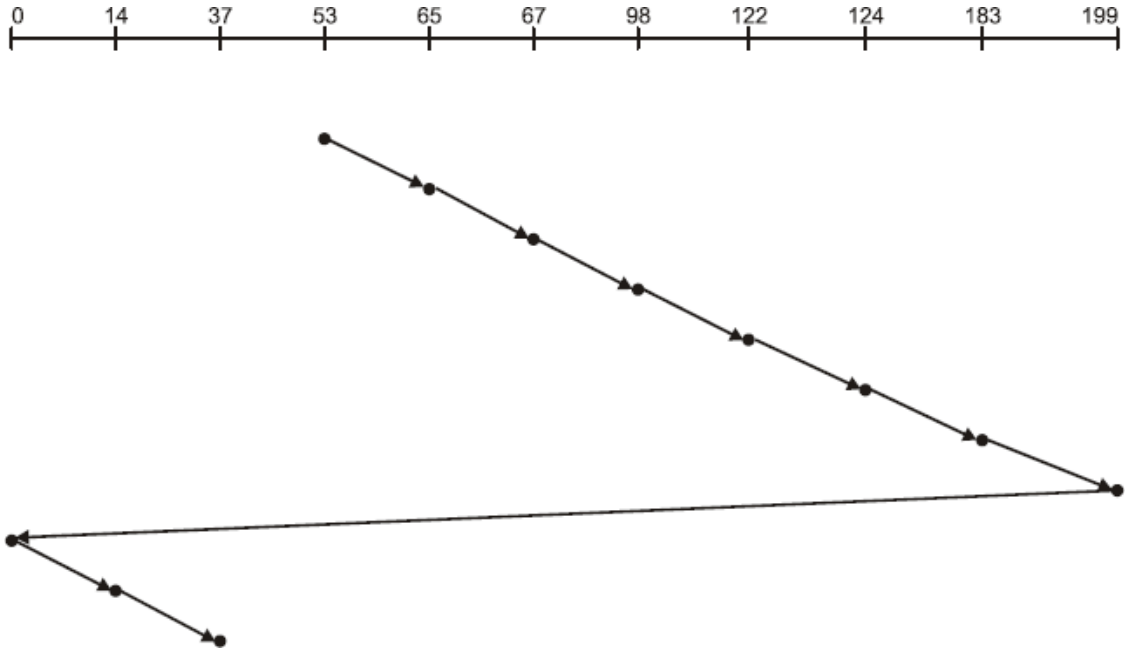


Figure 2.20: C-SCAN Scheduling

The above fig. shows graph for C-SCAN scheduling in which head first moves towards right end as it is having heaviest density of request serving requests as 65, 67,98, 122, 124, 183 then it reaches to end and moves back to beginning without serving any request and again it reverses direction serves requests 14 and 37. Figure 2.20 shows the movement of read write head over cylinder.

Total head movement is given below.

Calculations:-
= [65 - 53] + [67 - 65] + [98 - 67] + [122 - 98] + [124 - 122] + [183 - 124] + [199 - 183] + [199 - 0] + [14 – 0] + [37 - 14]
= 12+2+31+24+2+59+16+199+14+23
= 382

Advantages-

- The waiting time for the cylinders just visited by the head is reduced as compared to the SCAN Algorithm.
- It provides uniform waiting time.
- It provides better response time.

Disadvantages-

- It causes more seek movements as compared to SCAN Algorithm.
- It causes the head to move till the end of the disk even if there are no requests to be serviced.

5. LOOK Scheduling

This algorithm is very much similar to SCAN algorithm but only difference is that head does not move to the end of cylinder but it goes towards that end only up to last request and then it turns back to move towards another end to serve remaining request.

Consider same example.

The fig. shows graph for LOOK Scheduling in which head moves towards left end as it is nearer to current position and serves requests 37, 14 and then reverses and serves remaining request 65, 67, 98, 122, 124 and 183. Figure 2.21 shows the movement of read write head over cylinder.

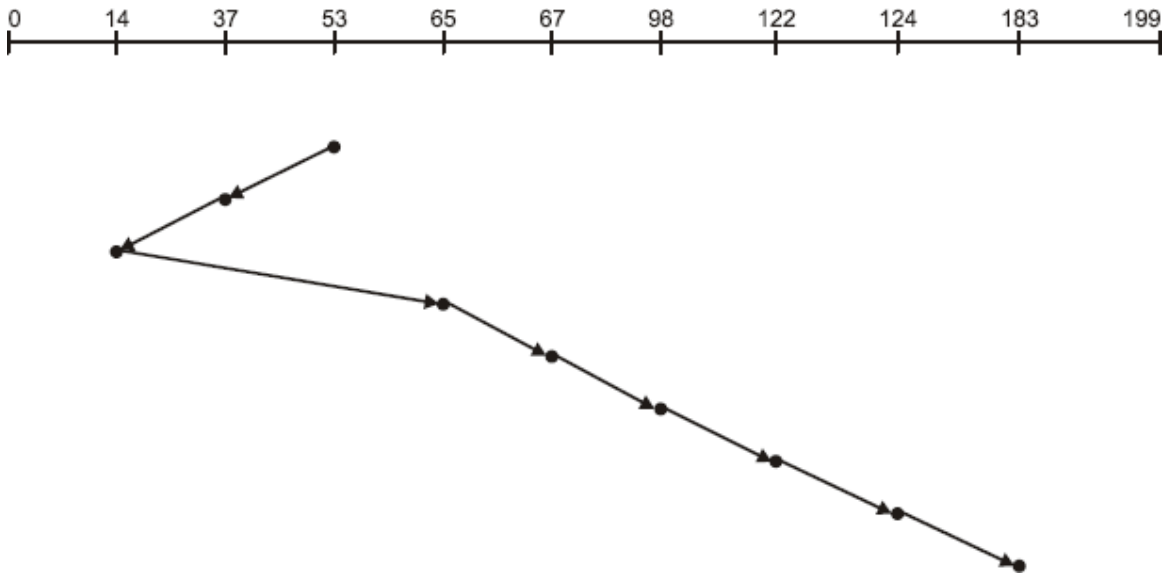


Figure 2.21: LOOK Scheduling

So total head movement for LOOK scheduling is,

Calculations:-
= [53-37]+[37-14]+[65-14]+[67-65]+[98-67]+[122-98]+[124-122]+[183-124]
= 16+23+51+2+31+24+2+59
= 208

Advantages-

- It does not causes the head to move till the ends of the disk when there are no requests to be serviced.
- It provides better performance as compared to SCAN Algorithm.
- It does not lead to starvation.
- It provides low variance in response time and waiting time.

Disadvantages-

- There is an overhead of finding the end requests.
- It causes long waiting time for the cylinders just visited by the head

Note:- *The main difference between SCAN Algorithm and LOOK Algorithm is-*
SCAN Algorithm scans all the cylinders of the disk starting from one end to the other end even if there are no requests at the ends.
LOOK Algorithm scans all the cylinders of the disk starting from the first request at one end to the last request at the other end.

6. C-LOOK Scheduling

This algorithm is very much similar to C-SCAN, only difference is that head does not go to end of cylinder in either directions but head goes up to last request in both directions. Consider the same example.

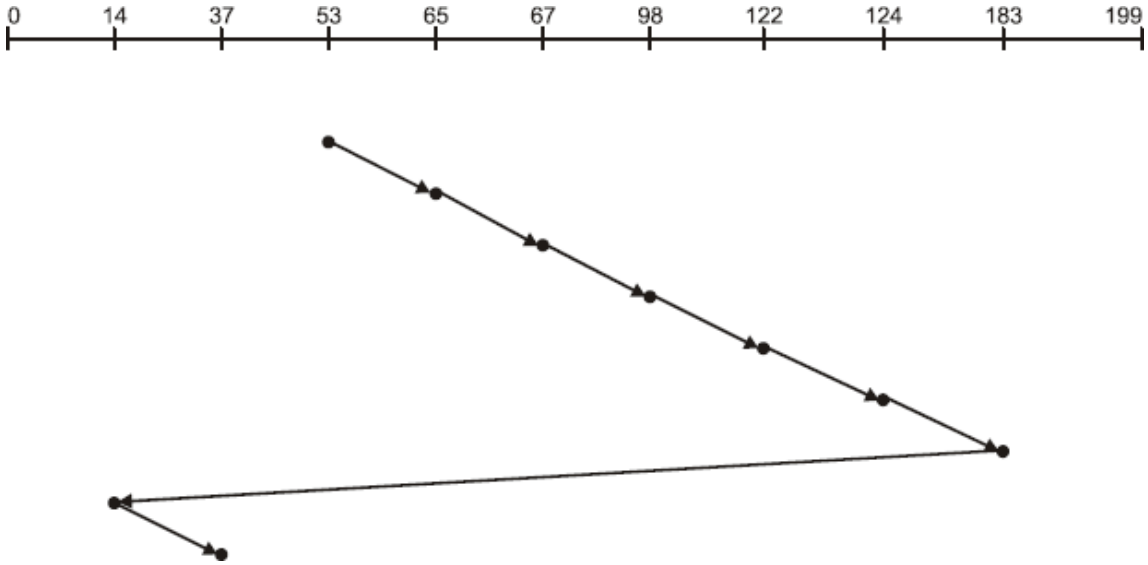


Figure 2.22: C-LOOK Scheduling

The above fig. shows graph for C-LOOK scheduling in which head moves towards right direction first, as it is having heaviest density of requests and serves requests 65, 67, 98, 122, 124 and 183 then it move towards other end and serves requests 14 and 37. Figure 2.22 shows the movement of read write head over cylinder.

Total head movement for CLOOK scheduling is given below.

Calculations:- = [65-53]+[67-65]+[98-67]+[122-98]+[124-122]+[183-0]+[37-14] = 12+2+31+24+2+59+169+23 = 322

Advantages-

- It does not causes the head to move till the ends of the disk when there are no requests to be serviced.
- It reduces the waiting time for the cylinders just visited by the head.
- It provides better performance as compared to LOOK Algorithm.
- It does not lead to starvation.
- It provides low variance in response time and waiting time.

Disadvantages-

- There is an overhead of finding the end requests.

2.10 Protection

As we keep huge amount of information in computer system, so we required to keep it safe from physical damage, Illegal and improper access. A computer system must be protected against unauthorized access, if computer is accessed by unauthorized user then it may cause several damage to data stored in system. The main goal of protection is to prevent malicious access & ensure that each resource is used only in accordance with system policies. Protection can be provided in many ways. A protection mechanism has to deal with following issues

- Loss of Availability
- Loss of data Integrity
- Loss of secrecy
- Loss of privacy
- Theft & Fraud
- Accidental losses

A systems that do not permit access to the files of other users, do not need protection. We can provide protection by controlling the access. More about protection and security we have discussed in chapter 6.

Types of Access

Following will be the types of access provided by OS.

- Read-Allow to perform only read operation.
- Write-Allow to perform both & Read & Write.
- Execute-Allow to load file into memory & execute it.
- Append-Allow to write new information at the end of file.
- Delete-Allow to delete the file.
- List-Allow to list the name & attributes of file.

Other operations such as rename, copy, may also be controlled. Many protection mechanisms have been proposed; each has advantages & disadvantages & must be appropriate for its intended application.

2.10.1Access Control:

The most common approach to protection problem is to make access dependent on the identity of the user. Different users may need different types of access to a file or directory. The most general scheme to implement identity dependent access is to associate with each file & directory, an access control list (ACL) specifying user names & types of access allowed for each user. When user request access to file, the OS checks ACL of the file & if that user is listed for requested access, then access is allowed. To reduce the length of ACL, many system recognize 3 classification of users as

- Owner - User who created file
- Group - Set of users who are sharing the file & need similar access is a group.
- Universe - All other users in the system constitute the universe.

The most common approach is to combine access control list with the more general owner, group & universe access control scheme. For e.g. consider that Mr. John is writing an article & he has 3 fellow Members assisting him then protection associated with this article is Mr. John can perform all operations on article. While fellow members can only read & write. All other users can only read but not write. Another mechanism to provide protection is to provide password with each file. The use of password has some drawbacks as number of password that

user need to remember & if only one password is used for all files then if one password is back then all files are accessible.

2.11 Free Space Management

Since disk space is limited, we need to reuse the space from deleted files for new files, if possible. To keep track of free disk space, the system maintains a free space list, which is a list of all free disk blocks. To create a file, we search a free space list for the required amount of space & allocate that space to the new file. When a file is deleted, its disk space is added to the free space list. There are four techniques used to manage free space as

- Bit vector
- Linked list
- Grouping
- Counting

(1) Bit vector

Mostly free space list is implemented as bit map or bit vector. Each block is represented by single bit. If block is free, the bit is 1. If block is allocated bit is 0. For example consider a disk where blocks 2, 3, 4, 5, 8, 9 are free & rest of block is allocated. The free space bitmap would be 00111100110 the main advantage of this approach is its relative simplicity & its efficiency in finding free blocks. But it is inefficient unless entire vector kept in main memory.

(2) Linked list

In this technique all free blocks are link together by keeping pointer to the first free block in a special location on the disk & caching it in memory. The first block contains pointer to the next free disk block. For example consider that blocks 1, 3, 7, 10 & 12 are free then it can be maintain using linked list as, figure 2.23 shows.

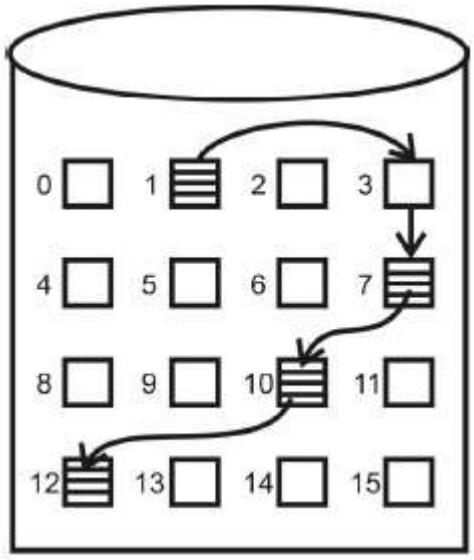


Figure 2.23: Linked free space list

This technique is not efficient as we cannot directly access any free block as they are linked to each other by linked list.

(3) Grouping

To overcome the problem of linked free space list new technique evolved called grouping. This technique is synonymous with index block method. In this technique address of

all free blocks are kept in the first free block. The address of a large number of free blocks can now be found quickly.

(4) Counting

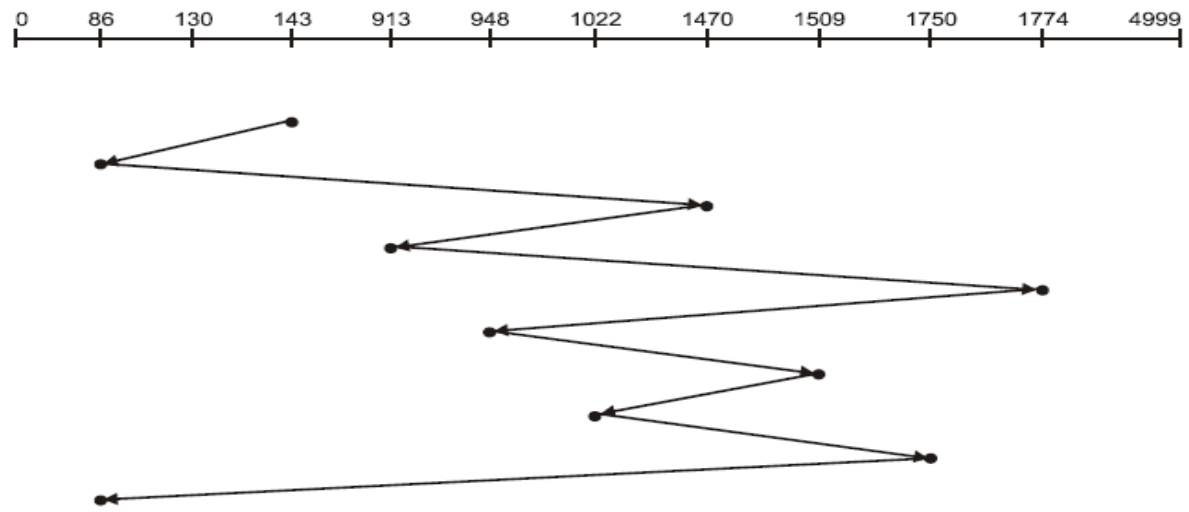
Generally several contiguous blocks may be allocated or freed simultaneously when space is allocated with contiguous allocation. Instead of keeping address of n free blocks, in this technique we keep the address of first free block & count of free contiguous blocks that follow the first block. Each entry in the free space list then consists of a disk afresh & a count. For example consider 1, 2, 3, 4 & 6, 7, 8, 9 block are free as shown in below fig so here we keep address block 1 & block 6 & then count of free contiguous blocks.

Numerical Based on Disk Scheduling

Problem 1:- Suppose that disk drive has 5000 cylinders numbered from 0 to 4999. The drive is currently serving a request at cylinder 143. The queue of pending request in FIFO is 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130. Starting from current head position what is total distance that disk arm moves to satisfy all the pending requests for each of following disk scheduling algorithms? 1. FCFS, 2. SSTF, 3. SCAN, 4. LOOK, 5. C-SCAN, 6. C-LOOK.

1. FCFS

Scenario:-

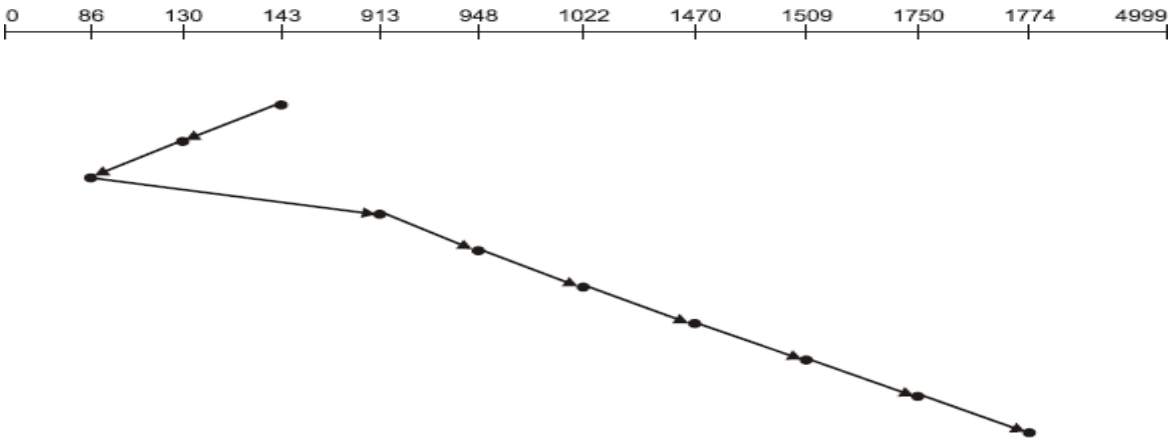


Total head movements for FCFS

Calculations:-
= [143 - 86] + [1470 - 86] + [1470 - 913] + [1774 - 913] + [1774 - 948] + [1509 - 948] + [1509 - 1022] + [1750 - 1022] + [1750 - 130]
= 57+1384+557+861+826+561+487+728+1620
= **7081**

2. SSTF

Scenario:-



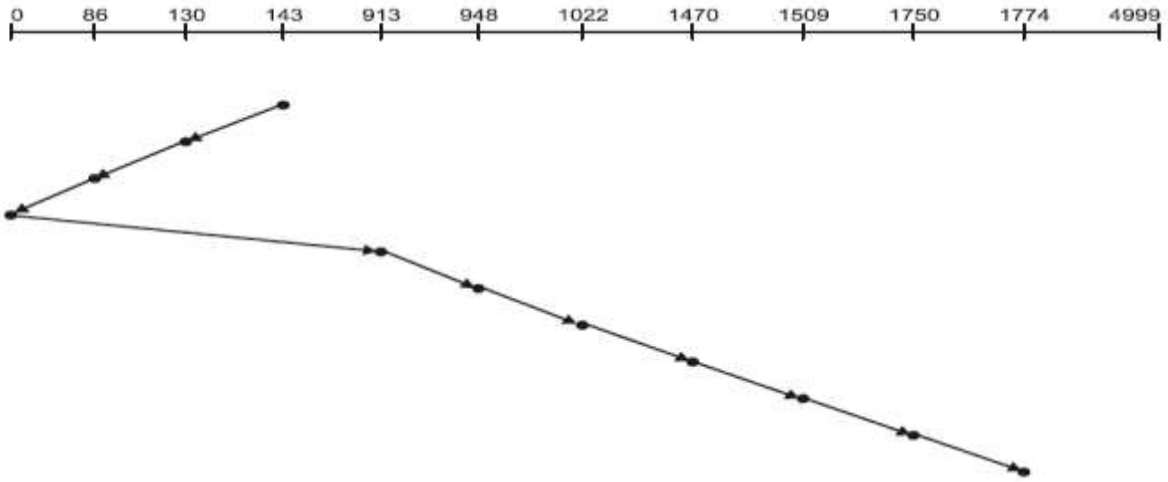
Total head movements for SSTF

Calculations:-
= [143 - 130] + [130 - 86] + [913 - 86] + [948 - 913] + [1022 - 948] + [1470 - 1022] + [1509 - 1470] + [1750 - 1509] + [1774 - 1750]
= 13+44+827+35+74+448+39+241+24
= 1745

3. SCAN

Note: ***In this algorithm it is not given in which direction head should move initially from current position, either to left or right so we have considered towards the nearest end from current position.***

Scenario:-

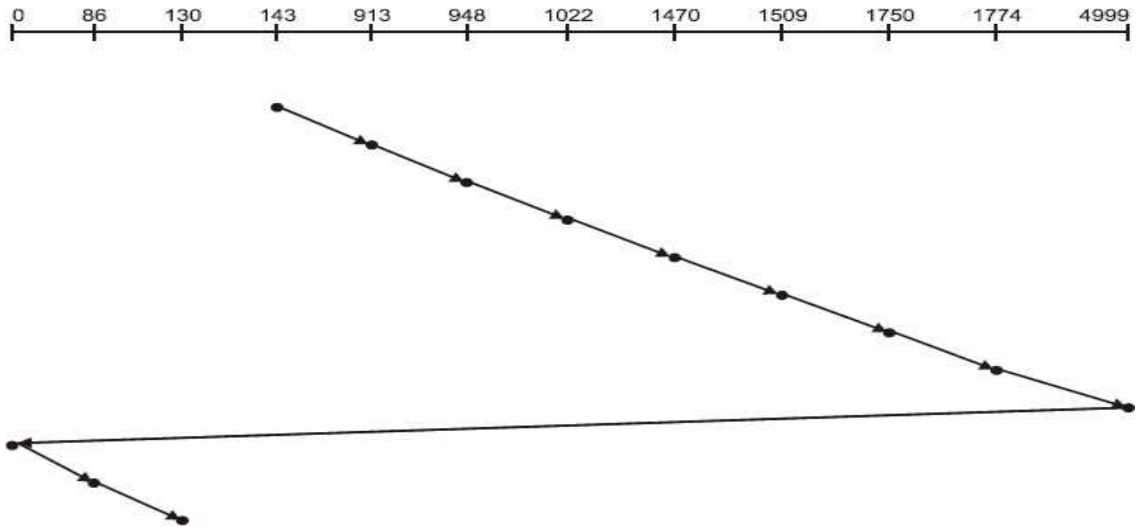


Total head movements for SCAN

Calculations:-
= [143 - 130] + [130 - 86] + [86 - 0] + [913 - 0] + [948 - 913] + [1022 - 948] + [1470 - 1022] + [1509 - 1470] + [1750 - 1509] + [1774 - 1750]
= 13+44+86+913+35+74+448+39+241+24

= 1917

4. CSCAN
Scenario:-

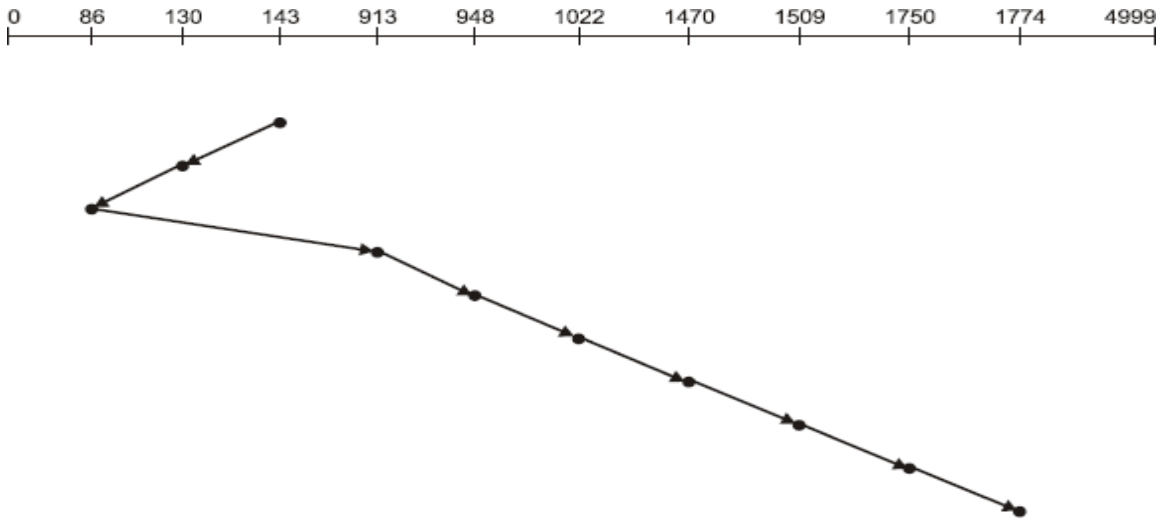


Total head movements for C-SCAN

Calculations:-
= [913 - 143] + [948 - 913] + [1022 - 948] + [1470 - 1022] + [1509 - 1470] + [1750 - 1509] + [1774 - 1750] + [4999 - 1774] + [86 - 0] + [130 - 86]
= 770+35+74+448+39+241+24+3225+4999+86+44
= 9985

Note: ***If both sides from current head position have same density of requests, then head should move towards that end which is nearer from current head position.***

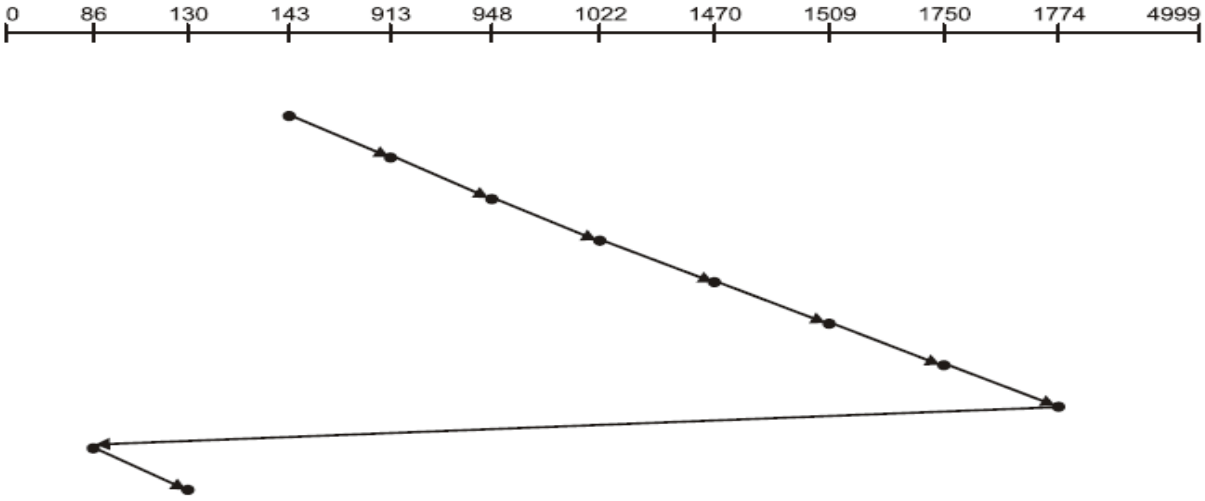
5.LOOK:-
Scenario:-



Total head movement for LOOK

Calculations:-
= [143 - 130] + [130 - 86] + [913 - 86] + [948 - 913] + [1022 - 948] + [1470 - 1022] + [1509 - 1470] + [1750 - 1509] + [1774 - 1750]
= 13+44+827+35+74+448+39+24+24
= 1745

6. C-LOOK
Scenario:-

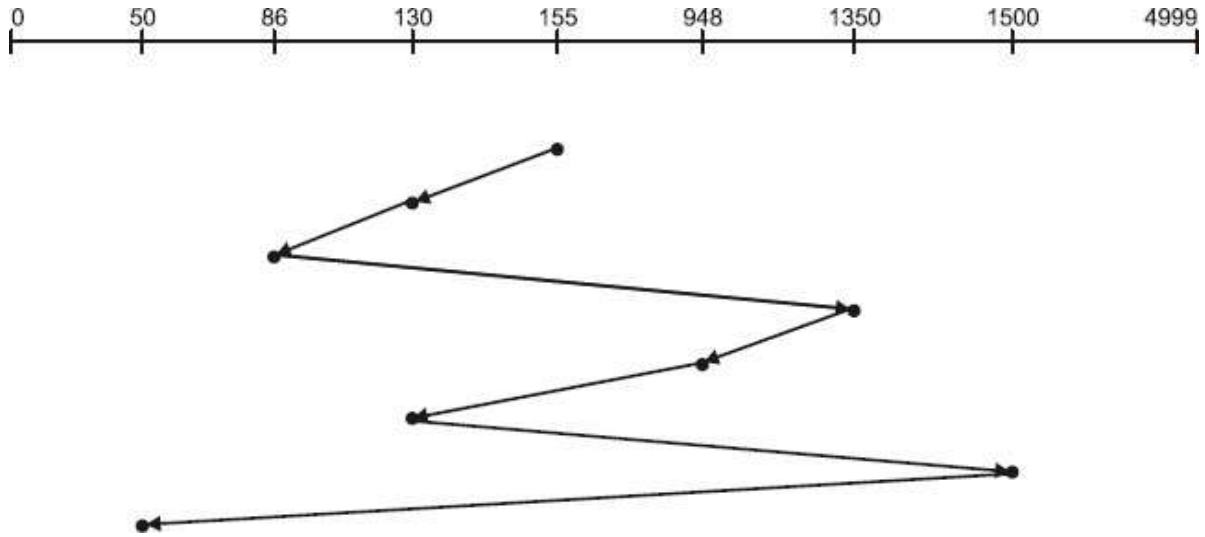


Total head movement for C-LOOK

Calculations:-
= (913-143) + (948-913) + (1022-948) + (1470-1022) + (1509-1470) + (1750-1509)+(1774-1750)+(1774-86)+(130-86)
= 770+35+74+448+39+241+24+1688+44
= 3363

Problem 2: Suppose that disk drive has 5000 cylinders numbered from 0 to 4999. The drive is currently serving at cylinder 155. The queue of pending request is 86, 1350, 948, 130, 1500, 50. Starting at current head position what is total distance that disk arm moves to satisfy all pending requests for each of following disk scheduling algorithm? 1. FCFS, 2. SSTF, 3. SCAN, 4. C-LOOK, 5. C-SCAN, 6. LOOK.

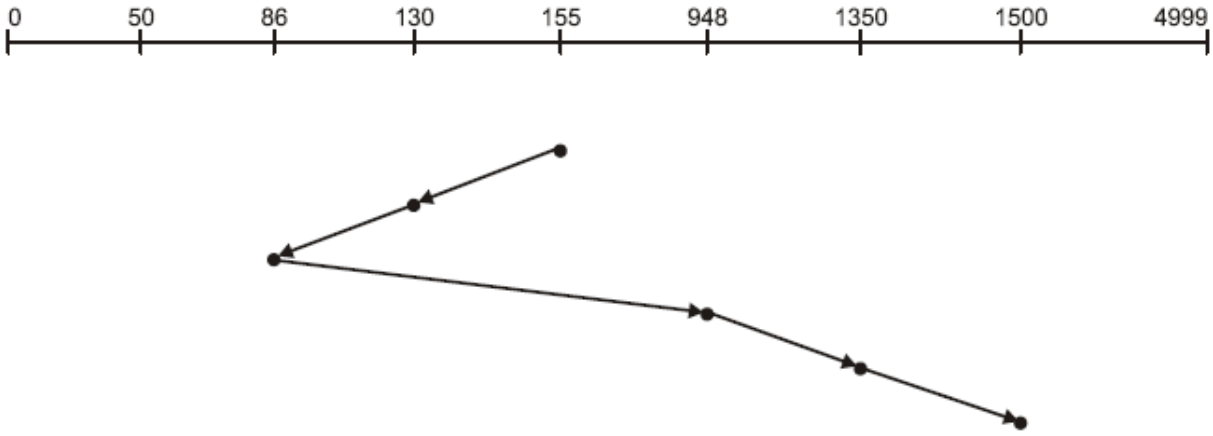
1. FCFS
Scenario:-



Total head movements for FCFS

Calculations:-
= [155-86]+[1350-86]+[1350-948]+[948-130]+[1500-130]+[1500-50]
= 69+1264+402+818+1370+1450
= 5373

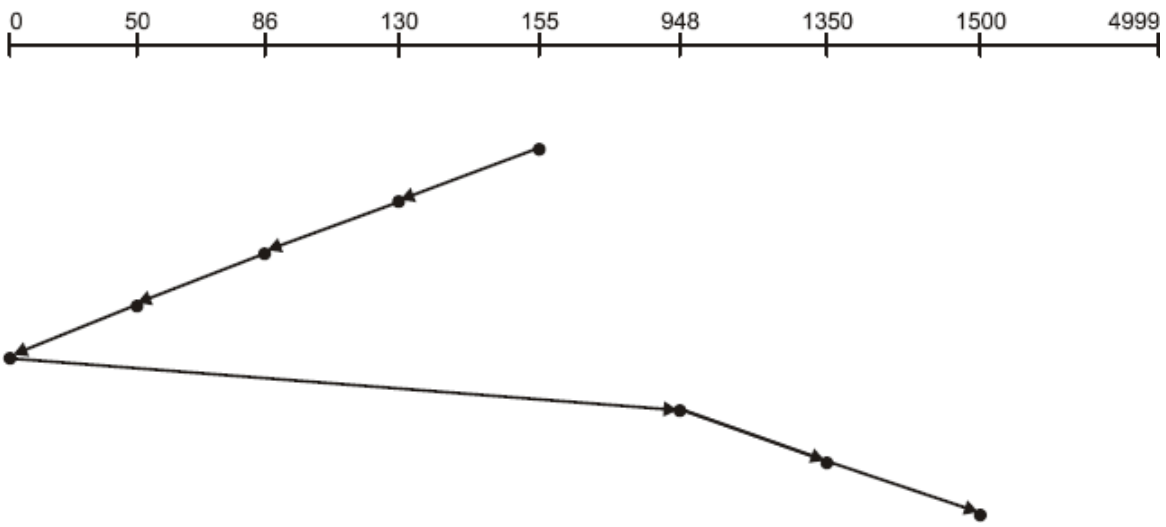
2. SSTF
Scenario:-



Total head movements for SSTF

Calculations:-
= [155-130]+[130-86]+[86-50]+[948-50]+[1350-948]+[1500-1350]
= 25+44+36+898+402+150
= 1555

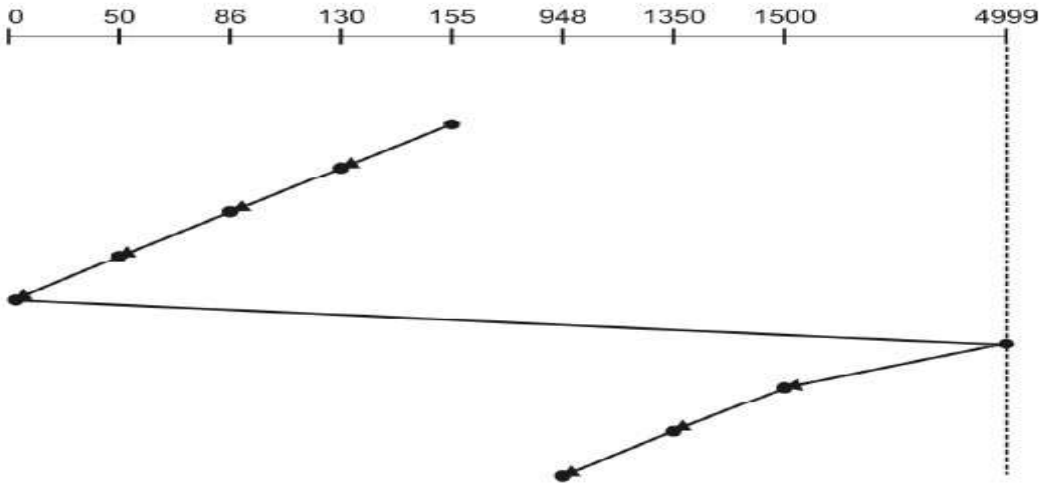
3. SCAN
Scenario:-



Total head movements for SCAN

Calculation:-
= [155-130]+[130-86]+[86-50]+[50-0]+[948-0]+[1350-948]+[1500-1350]
= 25+44+36+50+948+402+150
= 1655

4. CSCAN
Scenario:-

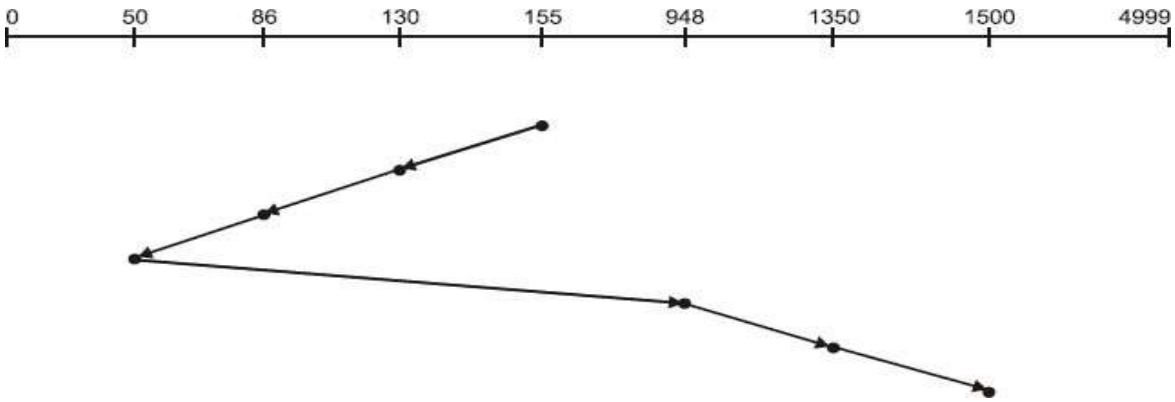


Total head movement is

Calculation:-
= (155 - 130) + (130 - 86) + (86 - 50) + (50 - 0) + (4999 - 0) + (4999 - 1500) + (1500 - 1350) + (1350 - 948)
= 25 + 44 + 36 + 50 + 4999 + 3499 + 150 + 402
= 9205

5. LOOK

Scenario:-

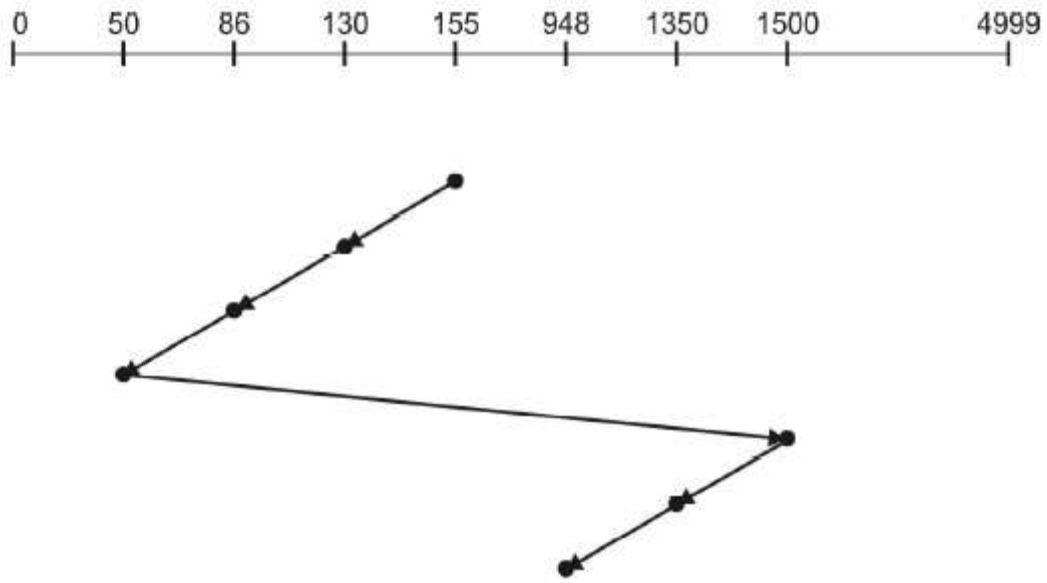


Total head movements for LOOK

Calculations:-
= [155-130]+[130-86]+[86-50]+[948-50]+[1350-948]+[1500-1350]
= 25+44+36+898+402+150
= 1555

6. C-LOOK

Scenario:-

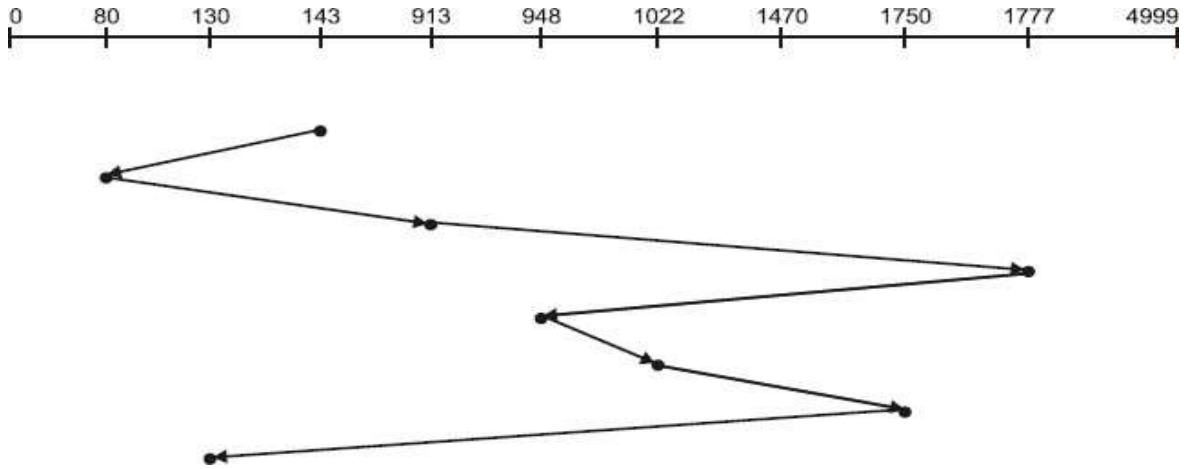


Total head movement is

Calculations:-
= (155-130)+(130-86)+(86-50)+(50-1500)+(1500-1350)+(1350-948)
= 25 + 44 + 36 + 1450 + 150 + 402
= 2107

Problem 3: Suppose that disk has 5000 cylinders. The drive is currently serving request at cylinder 143. The queue of pending requests in FIFO ordered as 80, 1470, 913, 1777, 948, 1022, 1750, 130. What is total distance that disk arm moves for following algorithm?

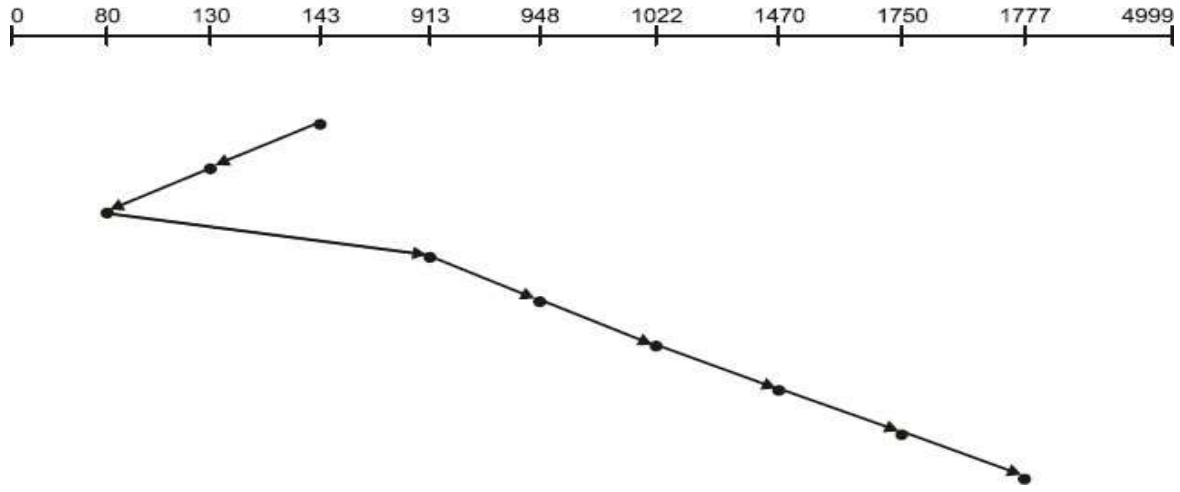
1. FCFS
Scenario:-



Total head movements for FCFS

Calculations:-
= [143-80] + [1470-80] + [1470-913] + [913-1777] + [1777-948] + [948-1022] + [1022-1750] + [1750-130]
= 63+1390+557+864+829+74+728+1620
= 6125

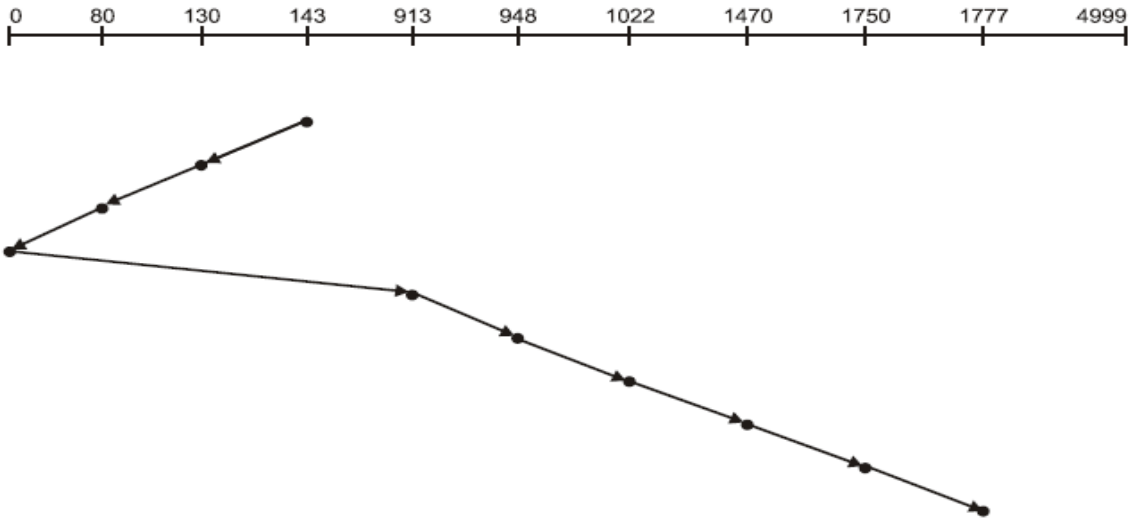
2. SSTF
Scenario:-



Total head movements for SSTF

Calculations:-
= [143-130] + [130-80] + [913-80] + [948-913] + [1022-948] + [1470-1022] + [1750-1470] + [1777-1750]
= 13+50+833+35+74+448+280+27
= 1760

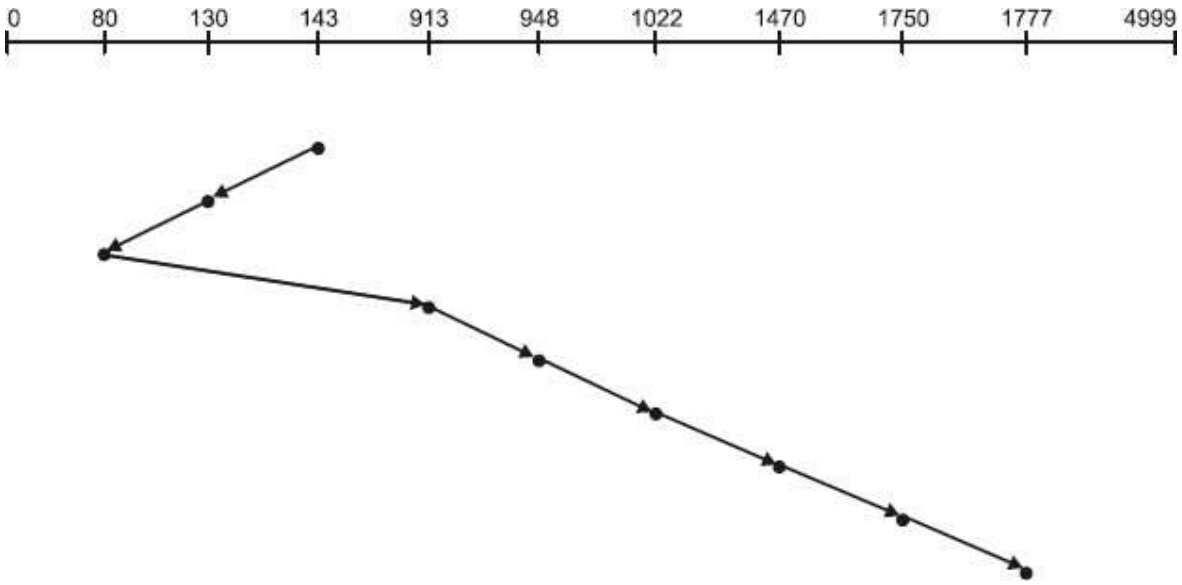
3. SCAN
Scenario:-



Total head movements for SCAN

Calculations:-
= [143-130] + [130-80] + [80-0] + [913-0] + [948-913] + [1022-948] + [1470- 1022] + [1750-1470] + [1777-1750]
= 13+50+80+913+35+74+448+280+27
= 1920

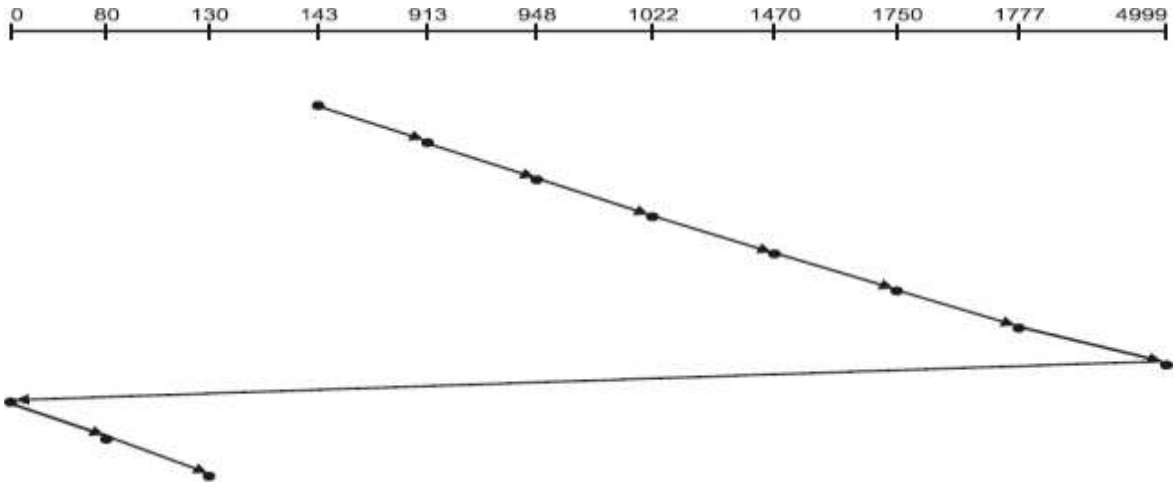
4. LOOK
Scenario:-



Total head movements for LOOK

Calculations:-
= [143-130] + [130-80] + [913-80] + [948-913] + [1022-948] + [1470-1022] + [1750-1470] + [1777-1750]
= 13+50+833+35+74+448+280+27
= 1760

5. C-SCAN
Scenario:-



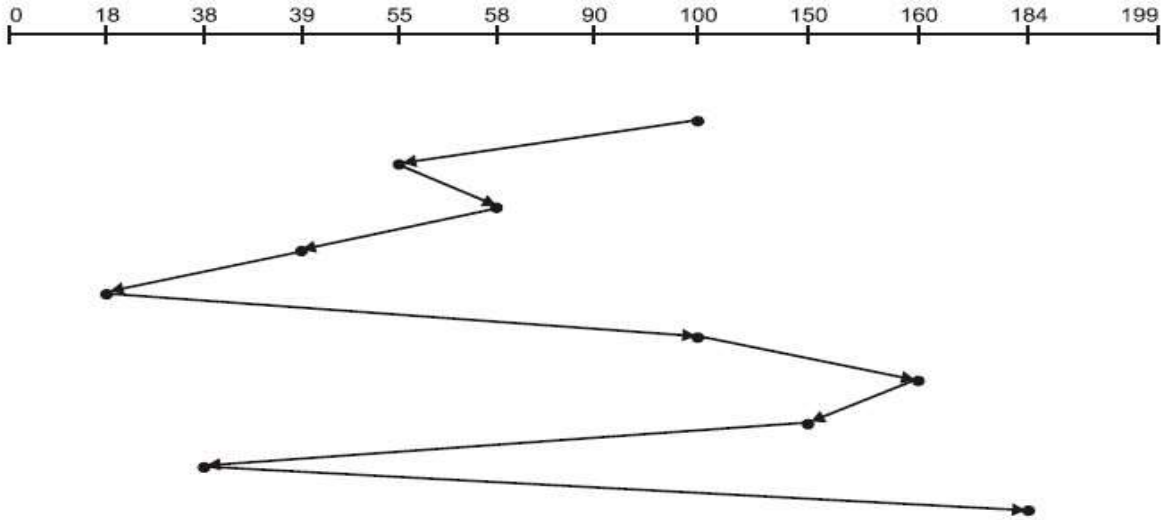
Total head movements for C-SCAN

Calculations:-
= [913-143]+[948-913]+[1470-948]+[1750-1470]+[1777-1750]+[4999-0] + [80-0]+[130-80]
= 770+35+74+448+280+27+3222+4999+80+50
= 9985

Problem 4: Suppose disk drive has 200cylinders numbered from 0 to 199. The initial head position is at 100 tracks. The queue of pending request is FIFO is 55, 58, 39, 18, 90, 160, 150, 38, and 184. Calculate total seek time for each of following.1. FCFS, 2. SSTF, 3. SCAN, 4. LOOK, 5. C-SCAN, 6. C-LOOK.

1. FCFS

Scenario:-



Total head movements for FCFS

Calculations:-

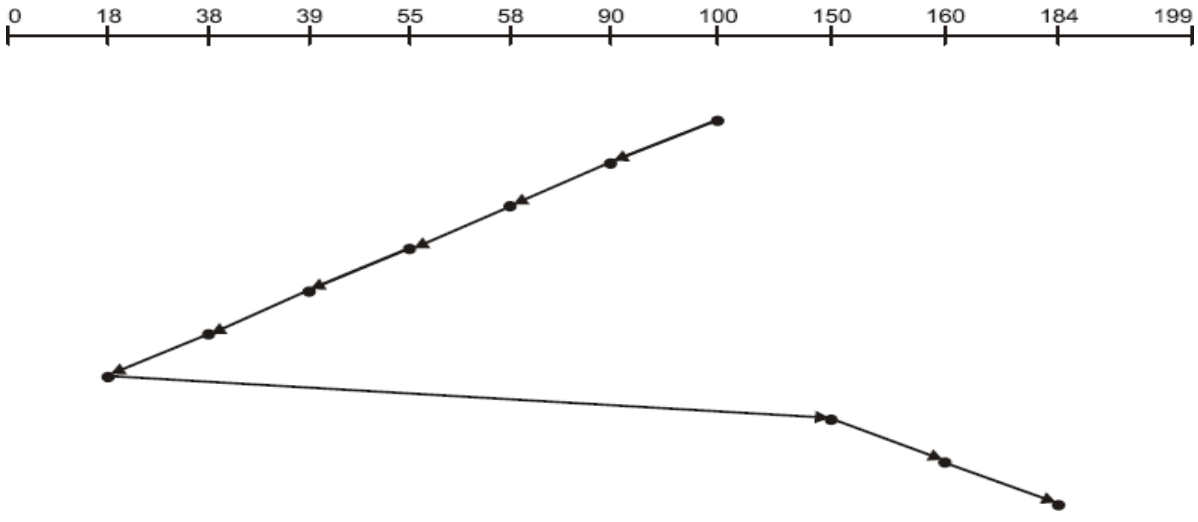
= [100-55]+[58-55]+[58-39]+[39-18]+[90-18]+[160-90]+[160-150]+[150-38] + [184-38]

= 45+3+19+21+72+70+10+112+146

= 498

2. SSTF

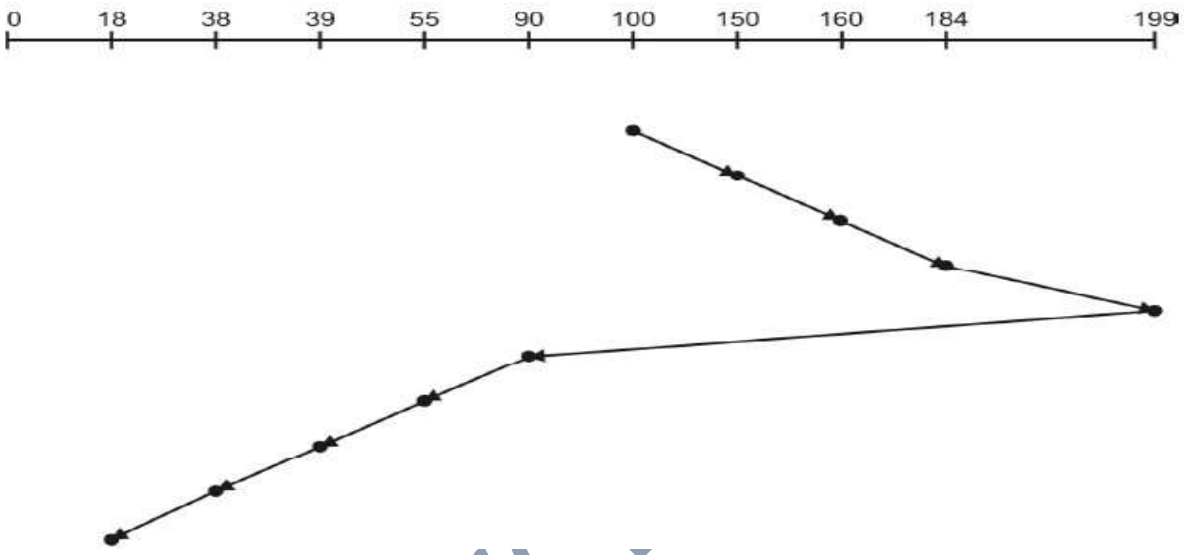
Scenario:-



Total head movements for SSTF

Calculations:-
= [100-90]+[90-58]+[58-55]+[55-39]+[39-38]+[38-18]+[150-18]+[160-150] + [184-160]
= 10+32+3+16+1+20+132+10+24
= 248

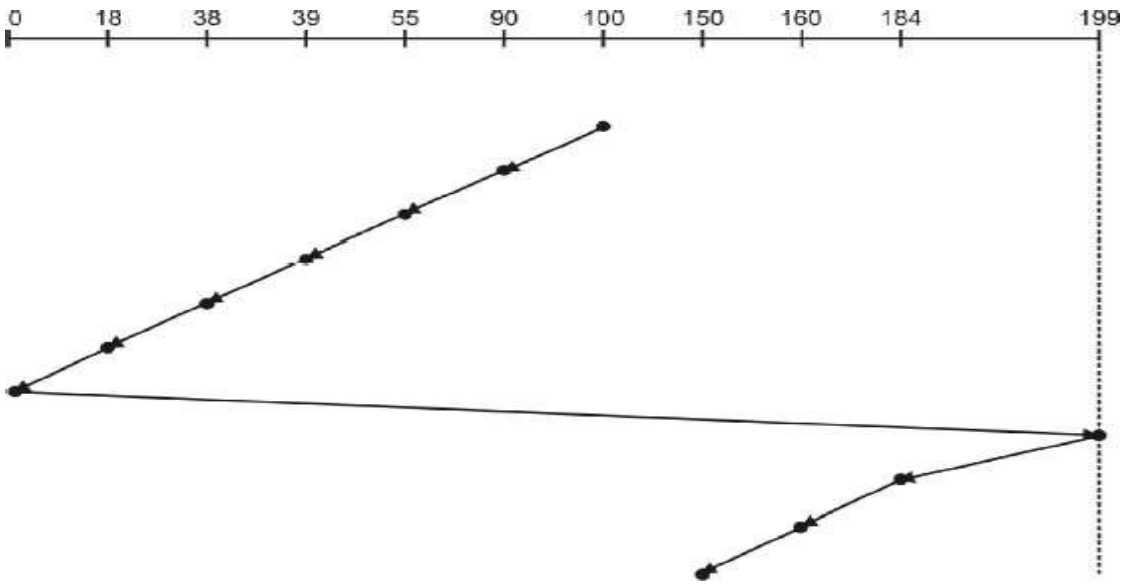
3. SCAN
Scenario:-



Total head movement is

Calculations:-
= (150-100) + (160 - 150) + (184 - 160) + (199 - 184) + (199 - 90) + (90 - 55) + (55 - 39) + (39 - 38) + (38 - 18)
= 50 + 10 + 24 + 15 + 109 + 35 + 16 + 1 + 20
= 280

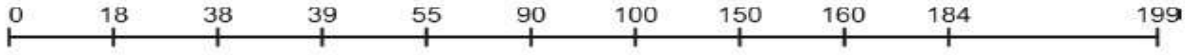
4. CSCAN
Scenario:-

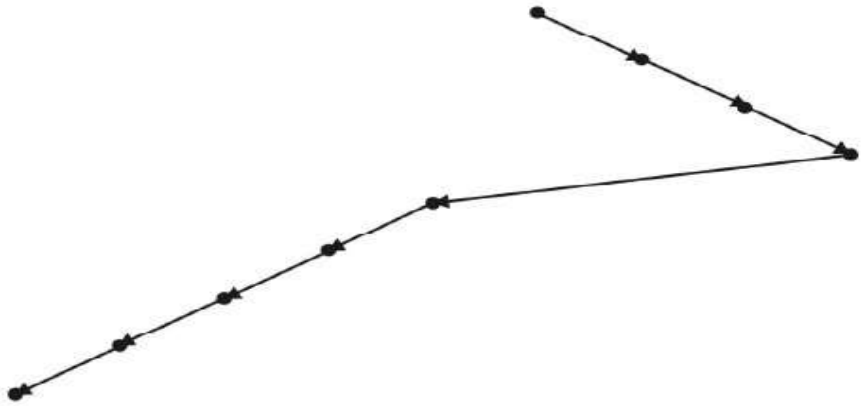


Total head movement is

Calculations:-
= (100 - 90)+(90 - 55) + (55 - 39) + (39 - 38) + (38 - 18) + (18 - 0) + (199 - 0) + (199 - 184) + (184 - 160) + (160 - 150)
= 10 + 35 + 16 + 01 + 20 + 199 + 15 + 24 + 10 + 18
= 348

5. LOOK
Scenario:-



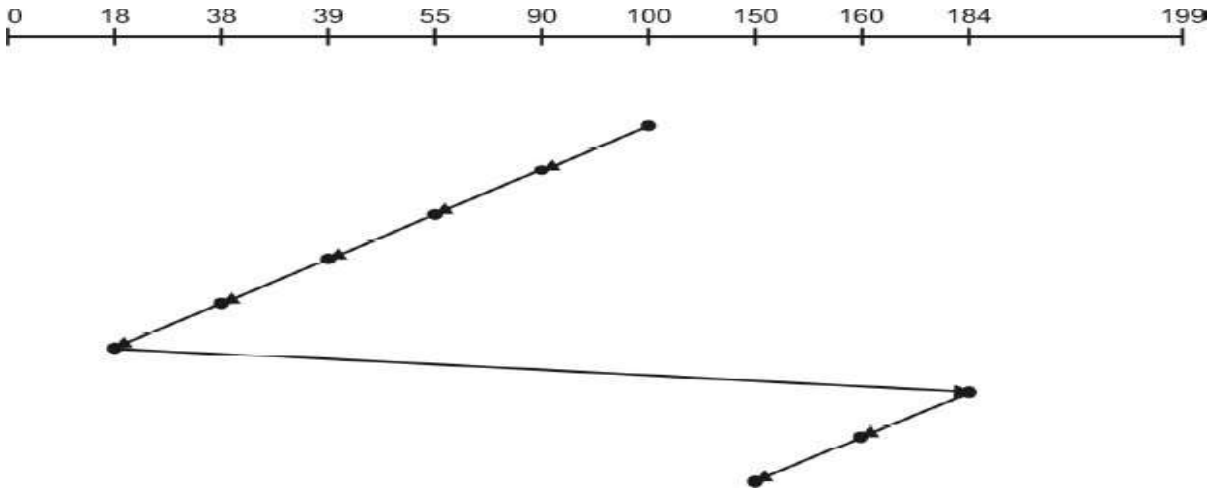


Total head movement is

Calculations:-
= (150 - 100) + (160 - 150) + (184 - 160) + (184 - 90) + (90 - 55) + (55 - 39) + (39 - 38) + (38 - 18)
= 50 + 10 + 24 + 94 + 35 + 16 + 01 + 20
= 250

6. CLOOK

Scenario:-



Total head movement is

Calculations:-

= (100 - 90)+(90 - 55)+(55 - 39)+(39 - 38)+(38 - 18)+(184 - 18)+ (184 - 160) + (160 - 150)

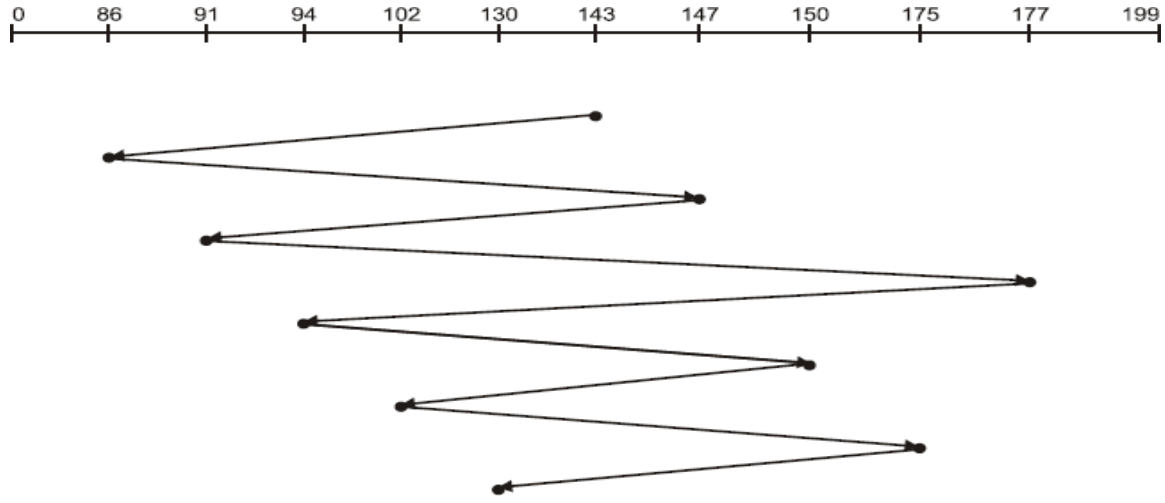
= 10 + 35 + 16 + 01 + 20 + 166 + 24 + 10

= 282

Problem 5: Suppose head of moving disk with 200 track numbered from 0 to 199 is currently serving request at 143. The queue of request is kept in FIFO order 86, 147, 91, 177, 94, 150, 102, 175, 130. What is total number of head movements needed to satisfy these requests for following disk scheduling algorithm?

1. FCFS

Scenario:-



Total head movements for FCFS

Calculations:-

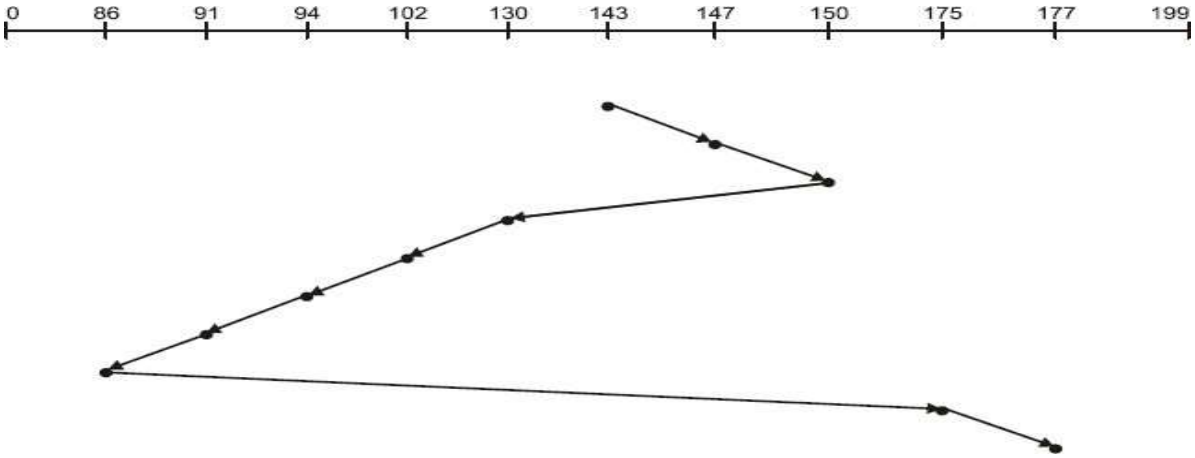
= [143-86]+[147-86]+[147-91]+[177-91]+[177-94]+[150-94]+[150-102] + [175-102]+[175-130]

= 57+61+56+86+83+56+48+73+45

= 565

2. SSTF

Scenario:-



Total head movements for SSTF

Calculations:-

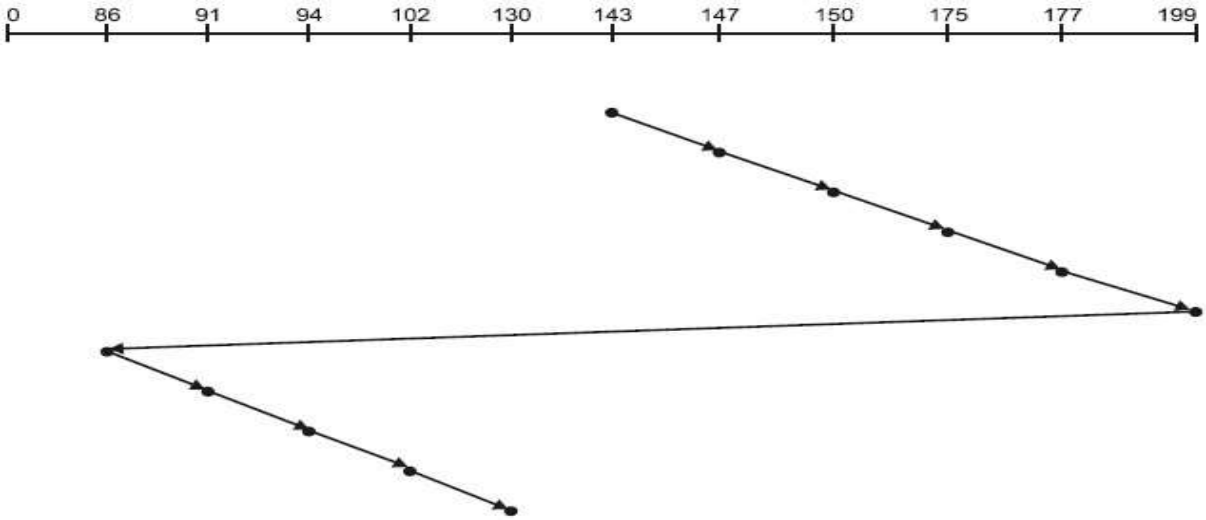
= [143-147]+[150-147]+[150-130]+[130-102]+[102-94]+[94-91]+[91-86] + [175-86]+[177-175]

= 4 + 3 + 20 + 28 + 8 + 3 + 5 + 89 + 2

= 162

3. SCAN

Scenario:-



Total head movement for SCAN

Calculations:-

= [147-143]+[150-147]+[175-150]+[177-175]+[199-177]+[199-86]+[91-86] + [94-91]+[102-94]+[130-102]

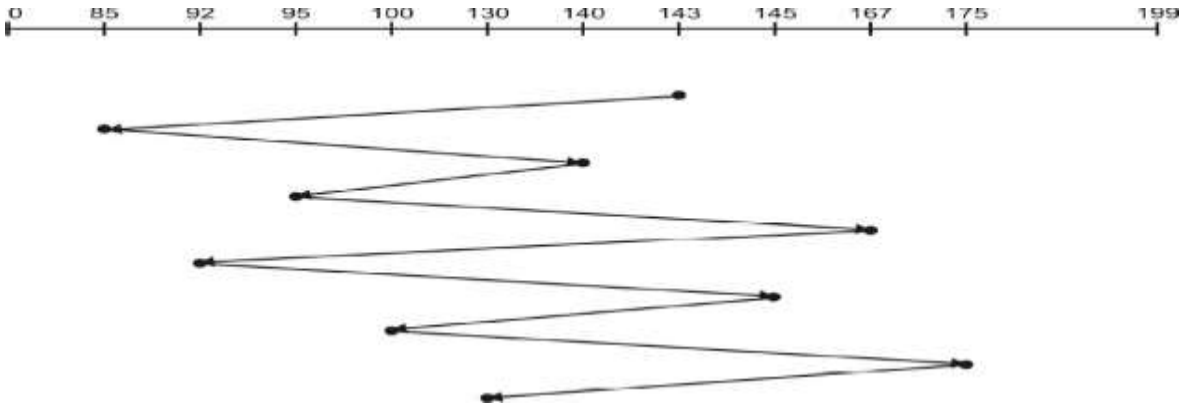
= 4+3+25+2+22+113+5+3+8+28

= 213

Problem 6: Suppose that head of a moving hard disk with 200 tracks numbered 0 to 199 is currently serving a request at track 143. If queue of request is kept in the FIFO order 85, 140, 95, 167, 92, 145, 100, 175, 130 what is total head movement to satisfy these requests for following disk scheduling algorithm? FCFS, SSIF, SCAN.

1. FCFS

Scenario:-

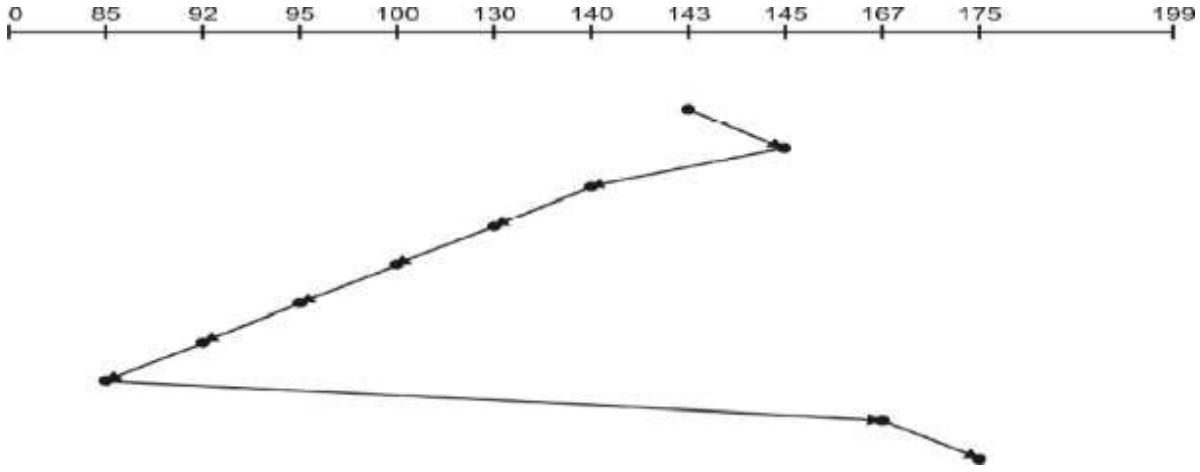


Total head movement for above problem is

Calculations:-
= (143 - 85) + (140 - 85) + (140 -95) + (167 - 95) + (167 - 92) + (145 - 92) + (145 - 100) + (175 - 100) + (175 - 130)
= 58 + 55 + 45 + 72 + 75 + 53 + 45 + 75 + 45
= 523

2. SSTF

Scenario:-

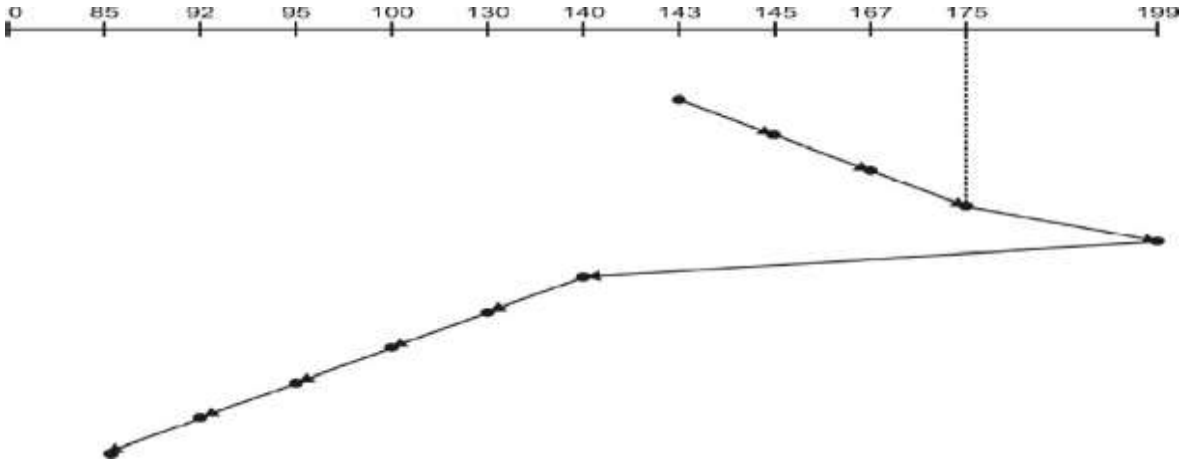


Total head movement for above problem is

Calculations:-
= (145 - 143) + (145 - 140) + (140 - 130) + (130 - 100)+(100 - 95)+(95 - 92) + (92 - 85) + (167 - 85) + (175 - 167)
= 2 + 5 + 10 + 30 + 5 + 3 + 7+ 82 + 8
= 152

3. SCAN

Scenario:-



Total head movement for above problem is

Calculations:-

= (145 - 143)+(167 - 145)+(175 - 167)+(199 - 175)+(199 - 140)+(140 - 130) +(130 - 100)+(100 - 95)+(95 - 92)+(92 - 85)

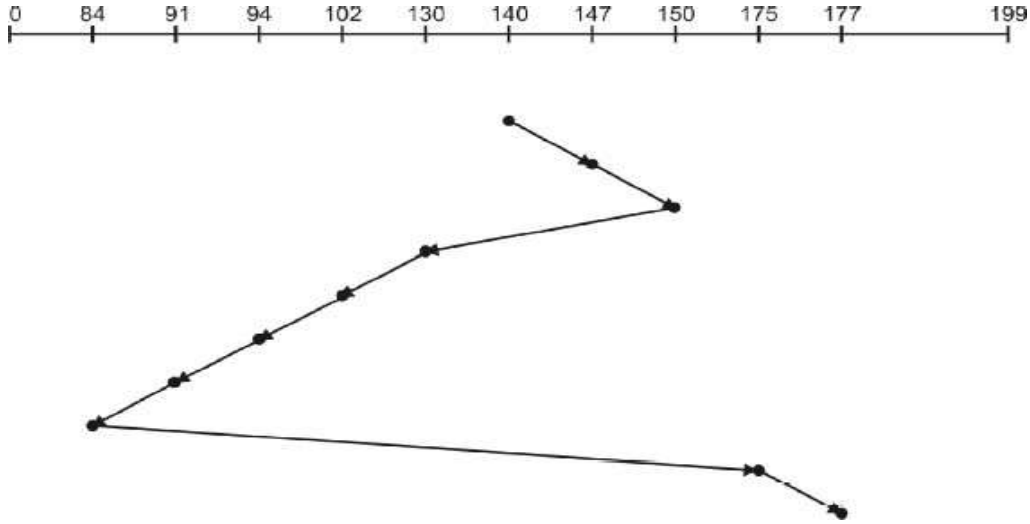
= 2 + 22 + 8 + 24 + 59 + 10 + 30 + 5 + 3 + 7

= 170

Problem 7: Suppose head of moving hard disk with 200 tracks numbering 0 to 199 is currently serving request at track 140. The arrival of request is kept in FIFO order. The requests at the track are 84, 147, 91, 177, 94, 150, 102, 175, 130. Assume earlier direction to be towards zero. Calculate total head movement for following disk scheduling also. (a) SSTF (b) SCAN (c) LOOK (d) FCFS.

1. SSTF

Scenario:-

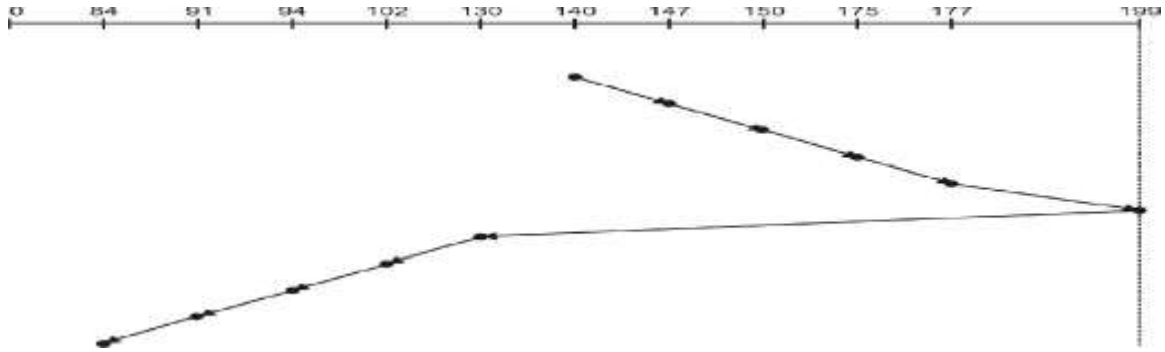


Total head movement is

Calculations:-
= (147 - 140)+(150 - 147)+(150 - 130)+(130 - 102)+(102 - 94)+(94 - 91) +(91 - 84)+(175 - 84)+(177 - 175)
= 7 + 3 + 20 + 28 + 8 + 3 + 7 + 91 + 2
= 169

2. SCAN

Scenario:-

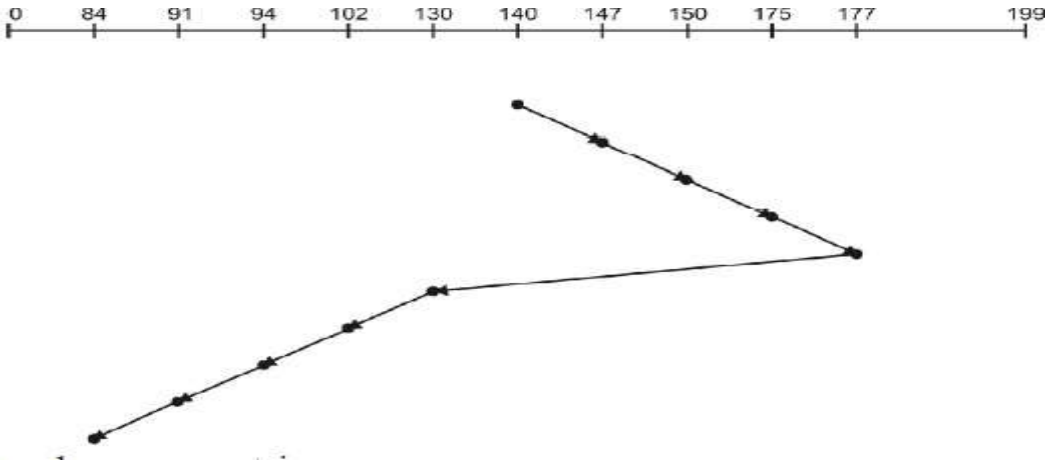


Total head movement is

Calculations:-
= (147 - 140)+(150 - 147)+(175 - 150)+(177 - 175)+(199 - 177)+(199 - 130) +(130 - 102)+(102 - 94)+(94 - 91)+(91 - 84)
= 7 + 3 + 25 + 2 + 22 + 69 + 28 + 8 + 3 + 7
= 174

3. LOOK

Scenario:-

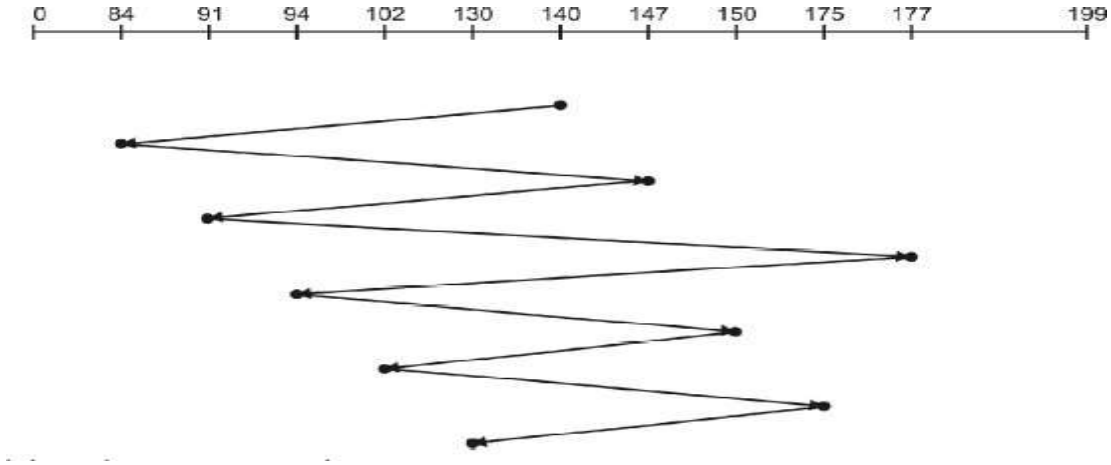


Total head movement is

Calculations:-
= (147 - 140) + (150 - 147) + (175 - 150) + (177 - 175) + (177 - 130) + (130 - 102) + (102 - 94) + (94 - 91) + (91 - 84)
= 7 + 3 + 25 + 2 + 47 + 28 + 8 + 3 + 7
= 130

4. F

CFS



Scenar
io:-

Calculations:-

Total head movements is
= (140 - 84) + (147 - 84) + (147 - 91) + (177 - 91) + (177 - 94) + (150 - 94)
+ (150 - 102) + (175 - 102) + (175 - 130)
= 56 + 63 + 56 + 86 + 83 + 56 + 48 + 73 + 45
= 566

Disk formatting

Disk formatting is a process to configure the data-storage devices such as hard-drive, floppy disk and flash drive when we are going to use them for the very first time or we can say initial usage. Disk formatting is usually required when new operating system is going to be used by the user. It is also done when there is space issue and we require additional space for the storage of more data in the drives. When we format the disk then the existing files within the disk is also erased.
We can perform disk formatting on both magnetic platter hard-drives and solid-state drives.

When we are going to use hard-drive for initial use it is going to search for virus. It can scan for virus and repair the bad sectors within the drive. Disk formatting has also the capability to erase the bad applications and various sophisticated viruses.

As we know that disk formatting deletes data and removes all the programs installed with in the drive. So it can be done with caution. We must have the backup of all the data and applications which we require. No-doubt disk formatting requires time. But the frequent formatting of the disk

decreases the life of the hard-drive.

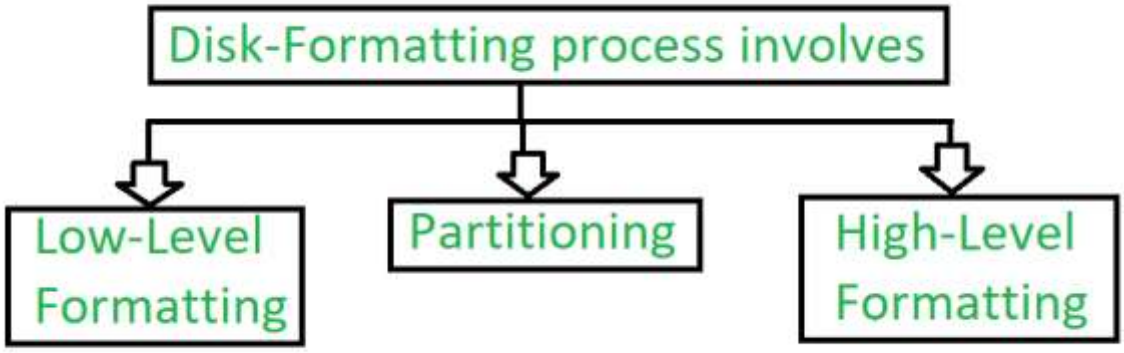


Figure – Formatting process of disk

1. Low-level Formatting :
Low level formatting is a type of physical formatting. It is the process of marking of cylinders and tracks of the blank hard-disk. After this there is the division of tracks into sectors with the sector markers. Now-a-days low-level formatting is performed by the hard-disk manufactures themselves.

We have data in our hard-disk and when we perform low-level formatting in the presence of data in the hard-disk all the data have been erased and it is impossible to recover that data. Some users make such a format that they can avoid their privacy leakage. Otherwise low-level will cause damage to hard-disk shortens the service-life.

Therefore, this formatting is not suggested to users.

2. Partitioning :
As suggesting from the name, partitioning means divisions. Partitioning is the process of dividing the hard-disk into one or more regions. The regions are called as partitions.

It can be performed by the users and it will affect the disk performance.

3. High-level Formatting :
High-level formatting is the process of writing. Writing on a file system, cluster size, partition label, and so on for a newly created partition or volume. It is done to erase the hard-disk and again installing the operating system on the disk-drive.

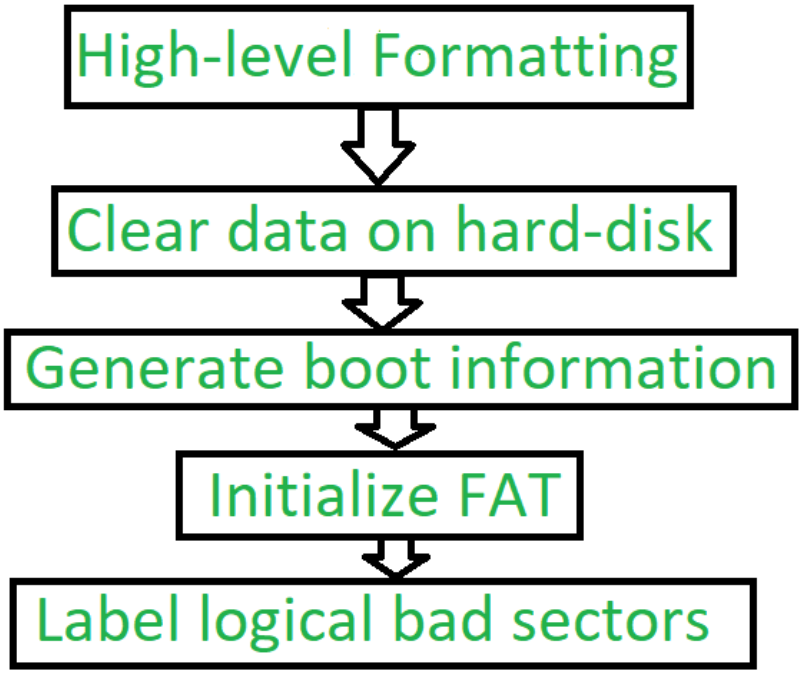


Figure – Steps of High-level Formatting

Firstly High-level formatting clears the data on hard-disk, then it will generate boot information, then it will initialise FAT after this it will go for label logical bad sectors when partition has existed.

Formatting done by the user is the high-level formatting.

Generally, It does not harm the hard-disk.It can be done easily with the Administrator, Windows snap-in Disk Management tool, diskpart, etc.

We can use such a format to fix some problems like errors in the file system, corrupted hard-drive and develop bad sectors

Boot Block and Bad Block in Operating System

The operating system is responsible for several other features of disk management, such as disk initialization, boot block or booting from disk, and bad block. A boot block is a region of a hard disk, floppy disk, optical disc, or other data storage device that contains machine code to be loaded into random-access memory (RAM) by a computer system's built-in firmware.

And a bad block is a sector on a computer's disk drive or a flash memory that cannot be used due to permanent damage, such as physical damage to the disk surface or failed flash memory transistors.

What is Boot Block in Operating System?

When a computer starts running or reboots to get an instance, it needs an initial program to run. This initial program is known as **the *bootstrap*** program, and it must initialize all aspects of the system, such as:

- First, initializes the CPU registers, device controllers, main memory, and then starts the operating system.
- The bootstrap program finds the operating system kernel on disk to do its job and then loads that kernel into memory.
- And last jumps to the initial address to begin the operating-system execution.

The bootstrap is stored in read-only memory (ROM). This location is convenient because ROM needs no initialization, and it is at a fixed location that the processor can start executing when powered up or reset. Since ROM is read-only memory, it cannot be infected by a computer virus. The problem is changing this bootstrap code requires changing the ROM and hardware chips. That's why systems store a tiny bootstrap loader program in the boot ROM whose job is to bring in a full bootstrap program from disk.

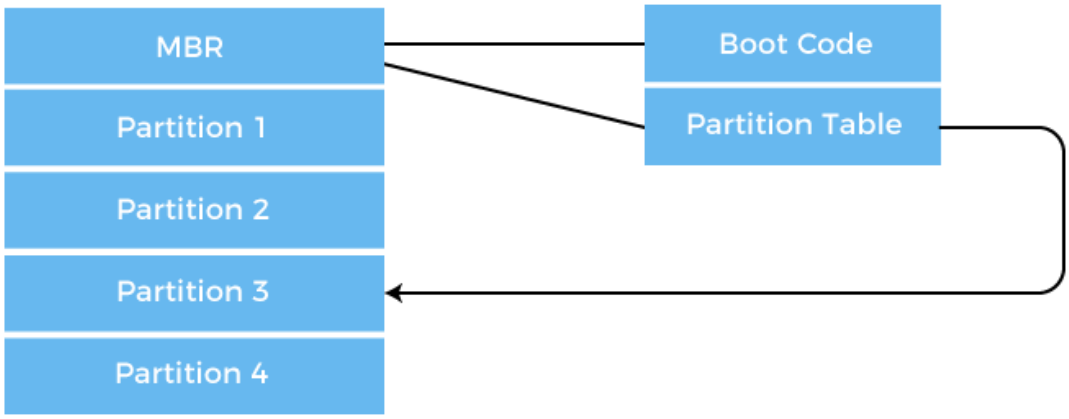
The full bootstrap program can change easily, and a new version is written onto the disk. The full bootstrap program is stored in **"*the boot blocks*"** at a fixed location on the disk. A disk that has a boot partition is called a ***boot disk*** or ***system disk***.

In the boot ROM, the code instructs the disk controller to read the boot blocks into memory (no device drivers are loaded at this point) and then starts executing that code. The full bootstrap program is more sophisticated than the bootstrap loader in the boot ROM because it can load the entire operating system from a non-fixed location on a disk and start the operating system running.

How Boot Block Works?

Let's try to understand this using an example of the boot process in Windows 2000.

The Windows 2000 stores its boot code in the first sector on the hard disk. The following image shows the booting from disk in Windows 2000.



- Moreover, Windows 2000 allows the hard disk to be divided into one or more partitions. This one partition is identified as the ***boot partition***, containing the operating system and the device drivers.
- In Windows 2000, booting starts by running the code placed in the system's ROM memory.
- This code allows the system to read code directly from the master boot record or MBR.
- The MBR also contains the table that lists the partition for the hard disk and a flag indicating which partition is to be boot from the system.
- Once the system identifies the boot partition, it reads the first sector from memory, known as a ***boot sector***. It continues the process with the remainder of the boot process, which includes loading various system services.

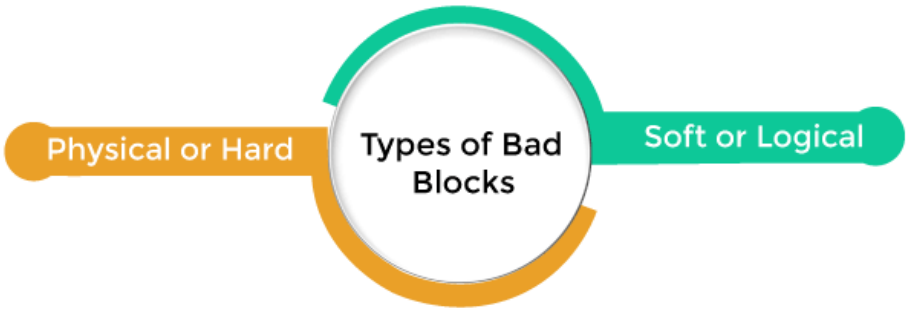
What is Bad Block in Operating System?

A bad block is an area of storage media that is no longer reliable for storing and retrieving data because it has been completely damaged or corrupted. Bad blocks are also referred to as ***bad sectors***.

We know disks have moving parts and have small tolerances. They are prone to failure. When the failure is complete, the disk needs to be replaced and its contents restored from backup media to the new disk. More frequently, one or more sectors become defective.

Types of Bad Blocks

There are two types of bad blocks in the operating system, such as:



1. **Physical** or **Hard Bad Block:** It comes from damage to the storage medium.
2. **Logical** or **Soft Bad Block:** It occurs when the operating system cannot read data from a sector.

For example, it occurs when the cyclic redundancy check (CRC) or error correction code (ECC) for a particular storage block and then does not match the data read by the disk.

Causes of Bad Block

Storage drives can ship from the factory with defective blocks that originated in the manufacturing process. The device with bad blocks is marked as defective before leaving the factory. These are remapped with the available extra memory cells.

Physical damage of a device also makes a bad block device because then the operating system is not able to access the data from the damaged device. Dropping a laptop, dust, and natural wear will also cause damage to the platter of the HDDs.

When the memory transistor fails, it will cause damage to the solid-state drive. Storage cells can also become unreliable over time, as NAND flash substrate in a cell becomes unusable after a certain number of program-erase cycles.

The erase process on the solid-state drive (SSD) requires many electrical charges through the flashcards. This degrades the oxide layer that separates the floating gate transistors from the flash memory silicon substrate and increases bit error rates. The drive's controller can use error detection and correction mechanisms to fix these errors. However, the errors can outstrip the controller's ability to correct them at some point, and the cell can become unreliable.

Software problems cause soft bad sectors. For example, if a computer unexpectedly shuts down, the hard drive also turns off in the middle of writing to a block. Due to this, the block could contain data that doesn't match the CRC detection error code, and then it would be identified as a bad sector.

How Bad Block Works

These blocks are handled in many ways, but it depends upon the ***disk*** and ***controller***. Bad blocks are handled manually for some disks with IDE controllers or simple disks.

1. The first strategy is to ***scan the disk*** to find bad blocks while the disk is being formatted. Any bad block discovered is flagged as unusable so that the file system does not allocate them. If blocks go bad during normal operation, a special program (Linux bad blocks command) must be run manually to search for the bad blocks and stop them away.
2. More sophisticated disks are smarter about bad-block recovery. The work of the controller is to ***maintain the list*** of bad blocks. The list formed by the controller is initialized during the low-level formatting at the factory and updated over the disk's life.

Low-level formatting holds the spare sectors which are not visible to the operating system. In the last, a controller replaces each bad sector logically with the spare sectors. This process is also known as ***sector sparing and forwarding***.

Example

In the operating system, a typical bad block transaction follows the following steps:

- Suppose the Operating system wants to read logical block 80.
- Now, the controller will calculate EEC and suppose it found the block as bad, then it reports to the operating system that the requested block is bad.
- Whenever the system is rebooted next time, a special command is used, and it will tell the controller that this sector is to be replaced with the spare sector.
- In future, whenever there is a request for block 80, the request is translated to the replacement sector's address by the controller.

Replacement of Bad Block

The redirection by the controller could invalidate any optimization by the operating system's disk-scheduling algorithm. For this reason, most disks are formatted to provide a few spare sectors in each cylinder and spare cylinder. Whenever the bad block will remap, the controller will use a spare sector from the same cylinder, if possible. Otherwise, a spare cylinder is also used.

Some controllers use the spare sector to replace the bad block. There is also another technique to replace the bad block, which is ***sector slipping***.

For example, suppose that logical block 20 becomes defective and the first available spare sector follows sector 200. then sector slipping starts remapping. All the sectors from 20 to 200, moving all down one spot. That sector 200 is copied into the spare, then sector 199 into 200, then 198 into 199, and so on, until sector 21 is copied into sector 22. In this way, slipping the sectors frees up the space of sector 21 so that sector 20 can be mapped to it.

The replacement of the bad block is not automatic because data in the bad block are usually lost. A process is trigger by the soft errors in which a copy of the block data is made, and the block is ***spared or slipped***. A hard error that is unrecoverable will lost all your data. Whatever file was using that block must be repaired, and that requires manual intervention.

Management of Bad Block

The best way to fix an HDD file that has been affected by a bad block is to write over the original file. This will cause the hard disk to remap the bad block or fix the data.

Bad block management is critical to improving NAND flash drive reliability and endurance. All changes must write to a new block, and the data in the original block must be marked for deletion.

- Once a flash drive fills up, the controller must start clearing out blocks marked for deletion before writing new data. After that, it consolidates good data by copying it to a new block. This process requires extra writes to consolidate the good data and results in write amplification where the number of actual writes exceeds the number requested. Write amplification can decrease the performance and life span of a flash drive.
- Flash vendors use many techniques to control write amplification. One, known as ***garbage collection***, involves proactively consolidating data by freeing up previously written blocks. These reallocated sectors can reduce the need to erase entire blocks of data for every write operation.
- Vendors also use data reduction technologies, such as ***compression*** and ***duplication***, to minimize the amount of data written and erased on a drive. In addition, an SSD's interface can help decrease write amplification. Serial ATA's TRIM and SAS's UNMAP commands identify data blocks no longer in use that can wipe out. This approach minimizes garbage collection and frees up space on the drive, resulting in better performance.

Difference between Device Driver and Device Controller in Operating System

In the world of software, device controllers and drivers are two commonly used software. Anyone with programming knowledge will be familiar with the words ***"device driver"*** and ***"device controller"***. A device driver is an operating system-specific and hardware-dependent program, and it offers to interrupt handling, which is required for the asynchronous time-dependent hardware interface. On the other hand, a device controller is a circuit board that interfaces between the device and the OS.

In this article, you will learn about the difference between a Device Driver and a Device Controller in the operating system. But before discussing the differences, you must know about the ***Device Driver*** and ***Device Controller*** in the operating system.

What is a Device Driver?

It is a software program that is utilized in computers to execute and operate systems that communicate with a component of a device. It is a code that is assigned to operating system users to enable the empowering of certain commands connected with a device.

It assists in the control and management of computer-connected devices. It is accomplished by providing the required number of functions for managing various portions of the device via programs generated by various types of software. Each new device comes with a built-in device driver.

These device drivers are essentially low-level programming software. It enables the computer system to perform functions for communication via many types of hardware devices. It is accomplished without needing to be concerned with the specifics of how hardware works. It aids in offering sufficient knowledge for carrying out these jobs.

What is a Device Controller?

It is a hardware program mainly utilized to connect a computer's operating system and functions in the phase by connecting the device driver. It is an electronic component that handles the link between incoming and outgoing signals in a processor by using chips.

It serves as a link between a device and any program that can receive commands from the operating system. These functions include buttons like reading, writing, etc. Every button and controller of various types of controllers differs from one another, with differences based on how they are utilized.

The device controller gets data from a connected system device and temporarily saves such data in a special purpose register inside the controller known as a local buffer. There is a device driver for every device controller. The memory is linked with the memory controller. The monitor is linked with the video controller, and the keyword is linked with the keyboard controller. The disk drive and USB drive are each attached to their respective disk controllers. These controllers are linked to the processor through the common bus.

Key differences between Device Driver and Device Controller in Operating System

Here, you will learn about the various key differences between ***Device Driver*** and ***Device Controller*** in operating systems. Some main differences between Device Driver and Device Controller in operating systems are as follows:

1. A device driver is a software method that is mainly utilized in computers to execute and operate systems that interact with a component of a device. On the other hand, a device controller is a hardware method that is mainly utilized to connect a computer's OS and functions in the phase by connecting the device driver.
2. The two types of device drivers are user and kernel device drivers. In contrast, the SCSI is a serial portal that is sufficient for the operation of a device controller.

3. A device driver is a type of software programming that assists in connecting with various types of operating systems. In contrast, a device controller is a type of hardware programming that acts as a bridge between OS in a computer system.
4. A device driver aids in interacting with the OS of various computer systems. In contrast, a device controller aids in understanding the links between the running and incoming signals from a computer's OS.
5. A device driver is a wider concept. In contrast, a device controller is a smaller concept.

Head-to-head comparison between the Device Driver and Device Controller in Operating System

The operating system has various head-to-head comparisons between the Device Driver and Device Controller in the Operating Systems. Some comparisons between Device Driver and Device Controller in Operating Systems are as follows:

Features	Device Driver	Device Controller
Definition	It is a software program that is mainly utilized in computers to execute and operate systems that interact with a device component.	It is a hardware program that is mainly utilized to connect a computer's OS and functions in the phase by connecting the device and the device driver.
Characteristic	It is a type of software programming that assists in connecting with various types of operating systems.	It is a type of hardware programming that acts as a bridge between OS in a computer system.
Types	The two types of device drivers are user and kernel device drivers.	The SCSI is a serial portal that is sufficient to operate a device controller.
Function	It aids in interacting with the OS of various computer systems.	It aids in understanding the links between the running and incoming signals from a computer's OS.
Concept	It has a wider concept.	It has a small concept.