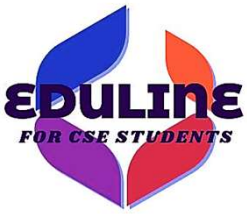
MODULE 5

CHAPTER 1 – FILE SYSTEM

CO – Students will be able to explain the security aspects and algorithms for file and storage management in Operating Systems



Prepared By Mr. EBIN PM, AP

FILE CONCEPT

- A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.
- A file can be defined as a data structure which stores the sequence of records. Files are stored in a file system, which may exist on a disk or in the main memory. Files can be simple (plain text) or complex (specially-formatted).
- The collection of files is known as Directory. The collection of directories at the different levels, is known as File System.

FILE ATTRIBUTES

- **Name :** Every file carries a name by which the file is recognized in the file system. One directory cannot have two files with the same name.
- **Identifier :** Along with the name, Each File has its own extension which identifies the type of the file. For example, a text file has the extension .txt, A video file can have the extension .mp4
- **Type :** In a File System, the Files are classified in different types such as video files, audio files, text files, executable files, etc.
- **Location :** In the File System, there are several locations on which, the files can be stored. Each file carries its location as its attribute.

- **Size:** The Size of the File is one of its most important attribute. By size of the file, we mean the number of bytes acquired by the file in the memory.
- **Protection :** The Admin of the computer may want the different protections for the different files. Therefore each file carries its own set of permissions to the different group of Users.
- **Time and Date :** Every file carries a time stamp which contains the time and date on which the file is last modified.

FILE OPERATIONS

- A file is a collection of logically related data that is recorded on the secondary storage in the form of sequence of operations.
- The content of the files are defined by its creator who is creating the file.
- The various operations which can be implemented on a file such as read, write, open and close etc. are called file operations.
- These operations are performed by the user by using the commands provided by the operating system.
- Some common operations are as follows:

❖**Create operation:** This operation is used to create a file in the file system. To create a new file of a particular type the associated application program calls the file system. This file system allocates space to the file. As the file system knows the format of directory structure, so entry of this new file is made into the appropriate directory.

❖**Open operation:** Once the file is created, it must be opened before performing the file processing operations. When the user wants to open a file, it provides a file name to open the particular file in the file system. It tells the operating system to invoke the open system call and passes the file name to the file system.

- **Write operation:** A system call write is issued that specifies the name of the file and the length of the data has to be written to the file. Whenever the file length is increased by specified value and the file pointer is repositioned after the last byte written.
- **Read operation:** This operation reads the contents from a file. A Read pointer is maintained by the OS, pointing to the position up to which the data has been read.
- **Re-position or Seek operation:** The seek system call re-positions the file pointers from the current position to a specific place in the file i.e. forward or backward depending upon the user's requirement. This operation is generally performed with those file management systems that support direct access files.

- **Delete operation:** Deleting the file will not only delete all the data stored inside the file it is also used so that disk space occupied by it is freed. In order to delete the specified file the directory is searched. When the directory entry is located, all the associated file space and the directory entry is released.
- **Truncate operation:** Truncating is simply deleting the file except deleting attributes. The file is not completely deleted although the information stored inside the file gets replaced.
- **Close operation:** When the processing of the file is complete, it should be closed so that all the changes made permanent and all the resources occupied should be released.

❖**Append operation:** This operation adds data to the end of the file.

❖**Rename operation:** This operation is used to rename the existing file.

FILE TYPES

- There are numerous file types that an operating system uses internally and are not generally used or required by the system user.
- These files could be application software files, kernel files, configuration files, metadata files, etc. Windows supports the following file types:

➢**Regular Files :** Regular files consist of information related to the user. The files are usually either ASCII or binary. ASCII files contain lines of text. The major benefit of an ASCII file is that it can be displayed or printed as it is, and it can be edited using a text editor. Regular files are supported by both Windows as well as UNIX-based operating systems.

- **Directories :** A directory in the file system is a structure that contains references to other files and possibly other directories. Files could be arranged by storing related files in the same directory. Directories are supported by both Windows as well as UNIX-based operating systems.
- **Character Special Files :** A character special file provides access to an I/O device. Examples of character special files include a terminal file, a system console file, a NULL file, a file descriptor file, etc.
- **Block Special Files :** Block special files enable buffered access to hardware devices They also provide some abstraction from their specifics. Unlike character special files, block special files always allow the programmer to read and write a block of any size or alignment. Block special files are supported by UNIX-based operating systems.

File types- name, extension

File Type	Usual extension	Function
Executable	exe, com, bin or none	ready-to-run machine- language program
Object	obj, o	complied, machine language, not linked
Source code	c. p, pas, 177, asm, a	source code in various languages
Batch	bat, sh	Series of commands to be executed
Text	txt, doc	textual data documents
Word processor	doc,docs, tex, rrf, etc.	various word-processor formats
Library	lib, h	libraries of routines
Archive	arc, zip, tar	related files grouped into one file, sometimes compressed.

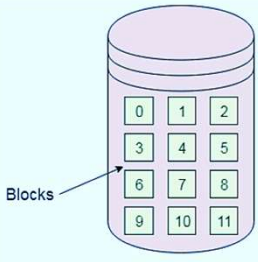
❖Functions of a File

- They are used for storing data in a computer.
- They enable the separation of data according to some criteria.
- They enable efficient, simple, organized access to data.
- They help in isolating sensitive or important data from the rest of the data.
- They enable locating particular data items in the storage medium.

FILE ALLOCATION METHODS

1. Contiguous Allocation

- Each file occupies a contiguous address space on disk.
- Assigned disk address is in linear order.
- Easy to implement. External fragmentation is a major issue with this type of allocation technique.

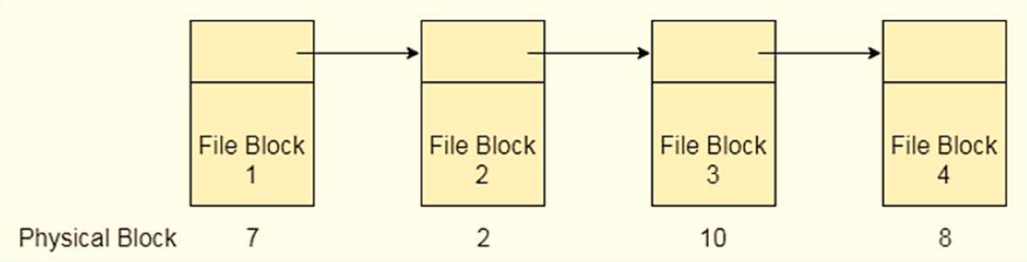


File Name	Start	Length	Allocated Blocks
abc.text	0	3	0,1,2
video.mp4	4	2	4,5
jtp.docx	9	3	9,10,11

Directory

2. Linked List Allocation

- Linked List allocation solves all problems of contiguous allocation. In linked list allocation, each file is considered as the linked list of disk blocks. However, the disks blocks allocated to a particular file need not to be contiguous on the disk. Each disk block allocated to a file contains a pointer which points to the next disk block allocated to the same file.



Advantages

- There is no external fragmentation with linked allocation.
- Any free block can be utilized in order to satisfy the file block requests.
- File can continue to grow as long as the free blocks are available.
- Directory entry will only contain the starting block address.

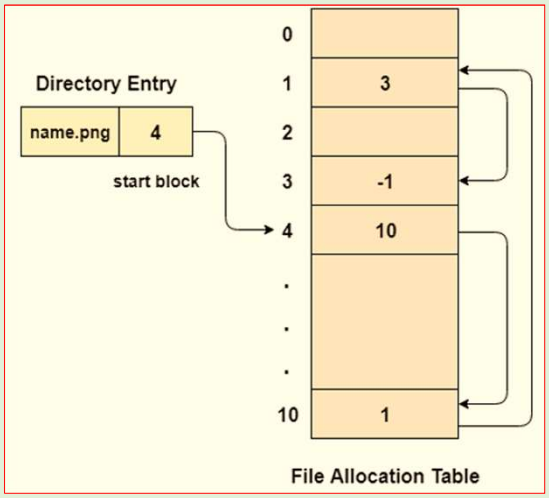
Disadvantages

- Random Access is not provided.
- Pointers require some space in the disk blocks.
- Any of the pointers in the linked list must not be broken otherwise the file will get corrupted.
- Need to traverse each block.

3. File Allocation Table

- The main disadvantage of linked list allocation is that the Random access to a particular block is not provided. In order to access a block, we need to access all its previous blocks.
- File Allocation Table overcomes this drawback of linked list allocation. In this scheme, a file allocation table is maintained, which gathers all the disk block links. The table has one entry for each disk block and is indexed by block number.
- File allocation table needs to be cached in order to reduce the number of head seeks. Now the head doesn't need to traverse all the disk blocks in order to access one successive block.

- It simply accesses the file allocation table, read the desired block entry from there and access that block. This is the way by which the random access is accomplished by using FAT. It is used by MS-DOS and pre-NT Windows versions.



Advantages

- Uses the whole disk block for data.
- A bad disk block doesn't cause all successive blocks lost.
- Random access is provided although its not too fast.
- Only FAT needs to be traversed in each file operation.

Disadvantages

- Each Disk block needs a FAT entry.
- FAT size may be very big depending upon the number of FAT entries.
- Number of FAT entries can be reduced by increasing the block size but it will also increase Internal Fragmentation.

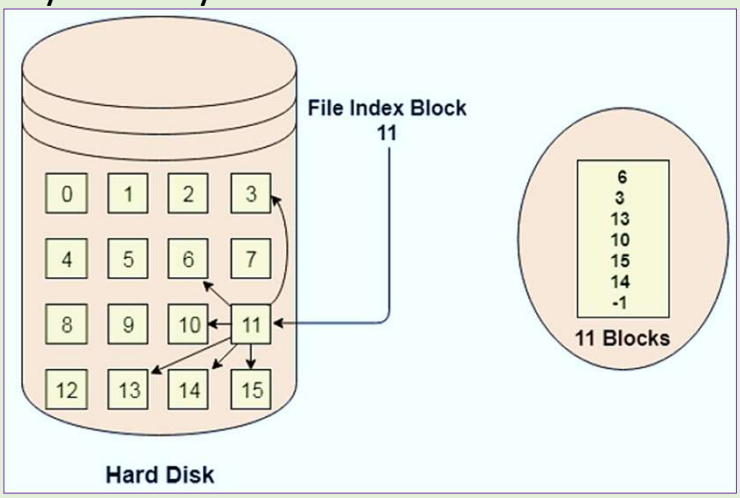
Prepared By Mr. EBIN PM, AP EDULINE 19

4. Indexed Allocation

- File allocation table tries to solve as many problems as possible but leads to a drawback.
- The more the number of blocks, the more will be the size of FAT. Here we need a new technology which can solve such problems.
- Therefore, we need to allocate more space to a file allocation table. Since, file allocation table needs to be cached therefore it is impossible to have as many space in cache.
- Instead of maintaining a file allocation table of all the disk pointers, Indexed allocation scheme stores all the disk pointers in one of the blocks called as indexed block.

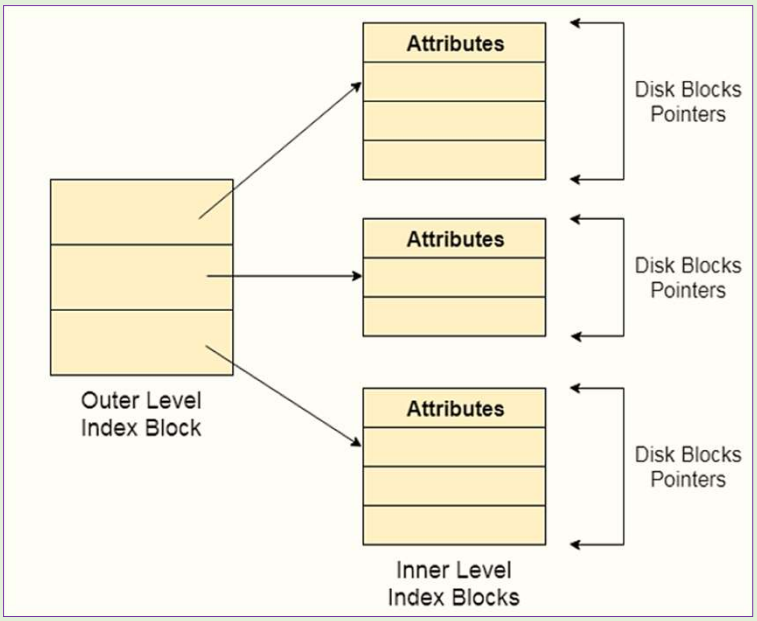
Prepared By Mr. EBIN PM, AP EDULINE 20

- Indexed block doesn't hold the file data, but it holds the pointers to all the disk blocks allocated to that particular file.
- Directory entry will only contain the index block address.



5. Multilevel Index Allocation

- In Multilevel index allocation, we have various levels of indices.
- There are outer level index blocks which contain the pointers to the inner level index blocks and the inner level index blocks contain the pointers to the file data.
- The outer level index is used to find the inner level index.
- The inner level index is used to find the desired data block.
- **Advantage:** Random Access becomes better and efficient.
- **Disadvantage:** Access time for a file will be higher.



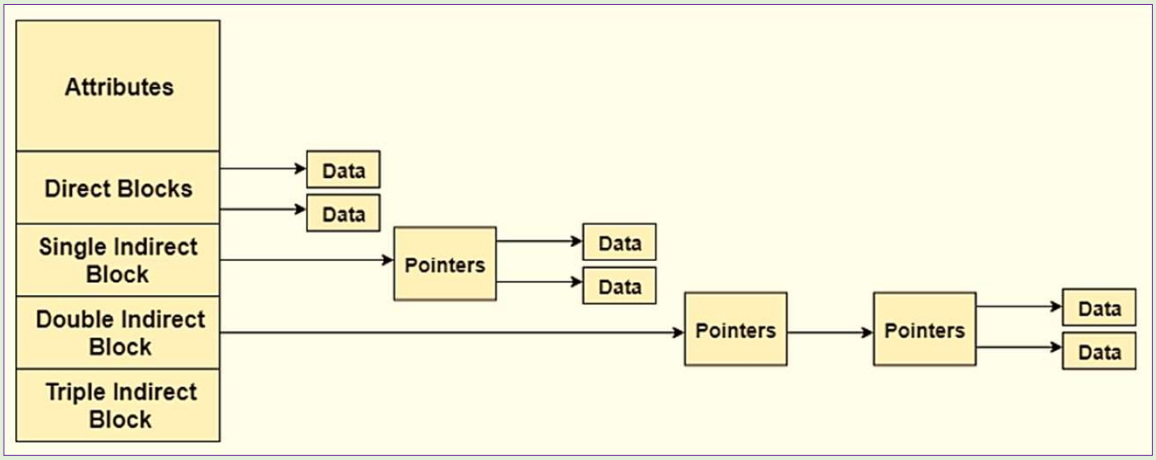
6. I-node (Index node)

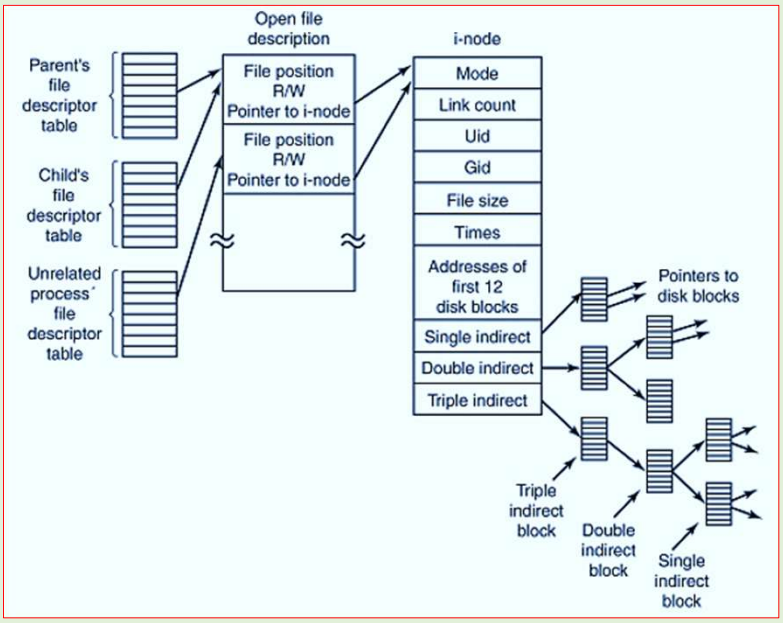
- In UNIX based operating systems, each file is indexed by an Inode. Inode are the special disk block which is created with the creation of the file system. The number of files or directories in a file system depends on the number of Inodes in the file system.

➢ An Inode includes the following information

1. Attributes (permissions, time stamp, ownership details, etc) of the file
2. A number of direct blocks which contains the pointers to first 12 blocks of the file.
3. A single indirect pointer which points to an index block. If the file cannot be indexed entirely by the direct blocks then the single indirect pointer is used.

4. A double indirect pointer which points to a disk block that is a collection of the pointers to the disk blocks which are index blocks. Double index pointer is used if the file is too big to be indexed entirely by the direct blocks as well as the single indirect pointer.
5. A triple indirect pointer that points to a disk block that is a collection of pointers. Each of the pointers is separately pointing to a disk block which also contains a collection of pointers which are separately pointing to an index block that contains the pointers to the file blocks.





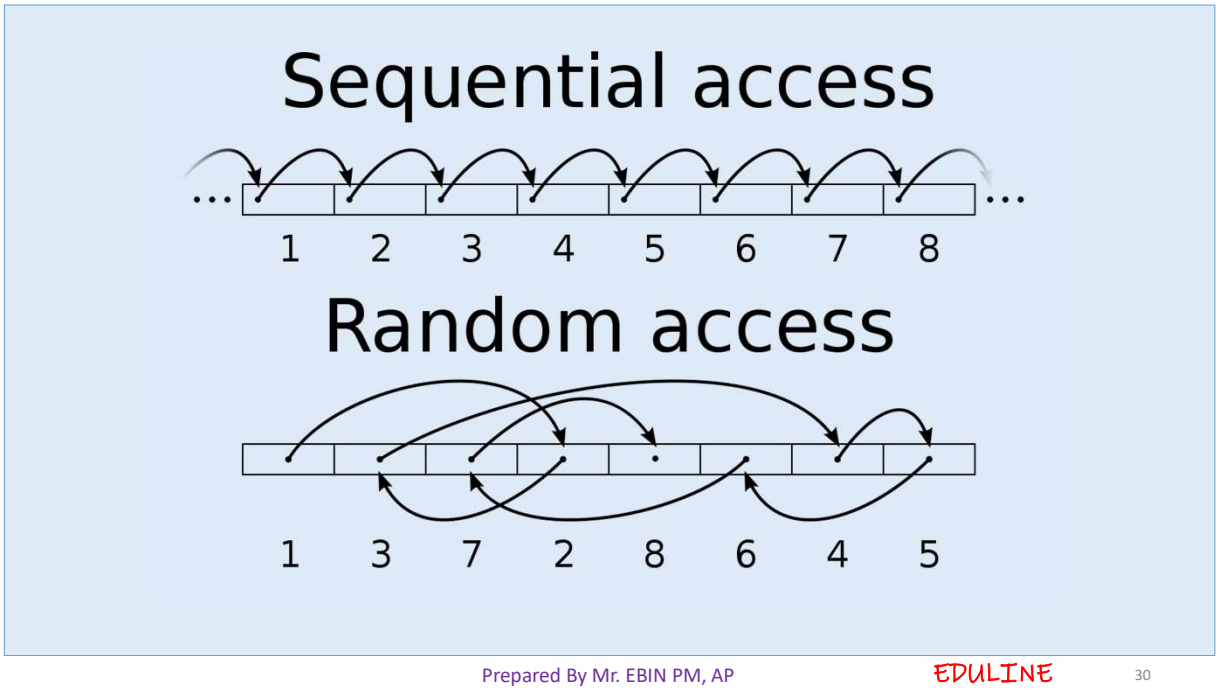
FILE ACCESS METHODS

❖Sequential access

- The records are accessed in some sequence, i.e., the information in the file is processed in order, one record after the other. This access method is the most primitive one. Example: Compilers usually access files in this fashion.
- In sequential access, the OS read the file word by word. A pointer is maintained which initially points to the base address of the file. If the user wants to read first word of the file then the pointer provides that word to the user and increases its value by 1 word. This process continues till the end of the file

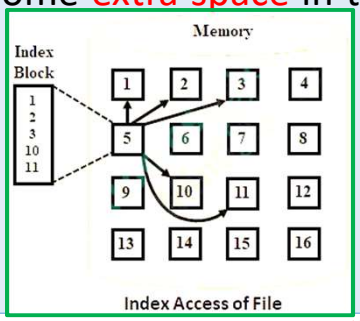
❖Direct / Random Access

- The Direct Access is mostly required in the case of database systems. In most of the cases, we need filtered information from the database. The sequential access can be very slow and inefficient in such cases.
- Suppose every block of the storage stores 4 records and we know that the record we needed is stored in 10th block. In that case, the sequential access will not be implemented because it will traverse all the blocks in order to access the needed record.
- Direct access will give the required result despite of the fact that the operating system has to perform some complex tasks such as determining the desired block number. However, that is generally implemented in database applications.



❖Indexed Access

- If a file can be sorted on any place, then an index can be assigned to a group of certain records. However, A particular record can be accessed by its index. The index is nothing but the address of a record in the file.
- In index accessing, searching in a large database became very quick and easy but we need to have some extra space in the memory to store the index value.
- Single Level index
- Multi Level index



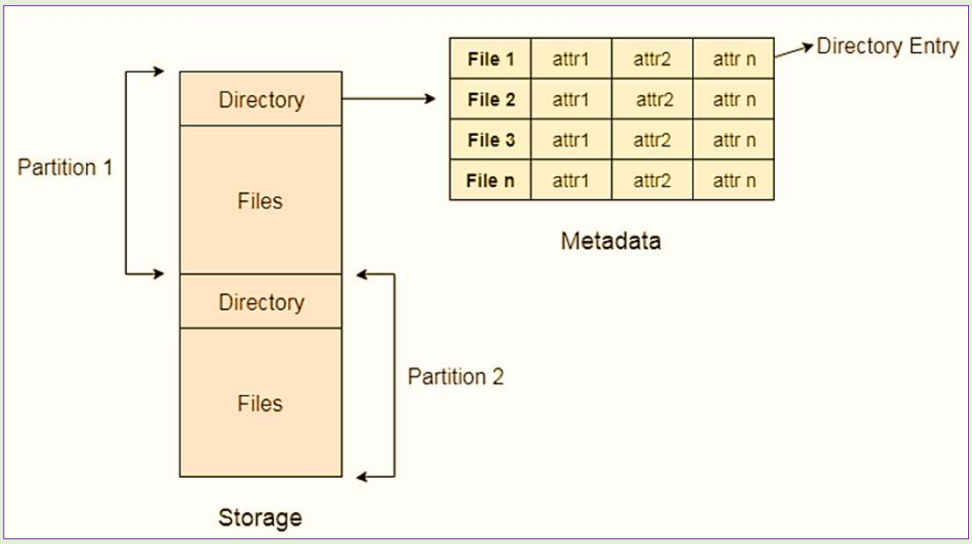
Prepared By Mr. EBIN PM, AP EDULINE 31

DIRECTORY STRUCTURE

- Directory can be defined as the listing of the related files on the disk. The directory may store some or the entire file attributes.
- To get the benefit of different file systems on the different operating systems, A hard disk can be divided into the number of partitions of different sizes. The partitions are also called volumes or mini disks.
- Each partition must have at least one directory in which, all the files of the partition can be listed.
- A directory entry is maintained for each file in the directory which stores all the information related to that file.

Prepared By Mr. EBIN PM, AP EDULINE 32

- A directory can be viewed as a file which contains the Meta data of the bunch of files.



Prepared By Mr. EBIN PM, AP EDULINE 33

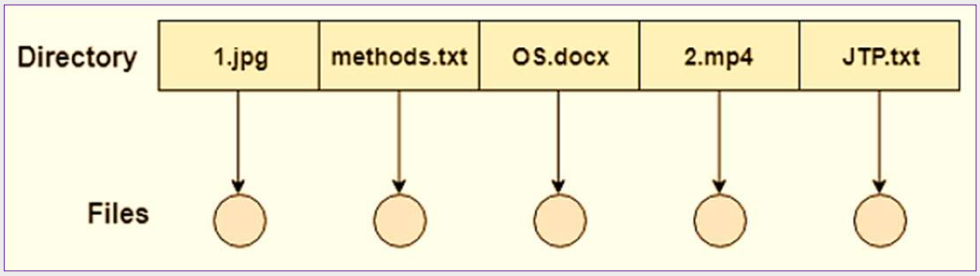
- Every Directory supports a number of common operations on the file:

- File Creation
- Search for the file
- File deletion
- Renaming the file
- Traversing Files
- Listing of files

Prepared By Mr. EBIN PM, AP EDULINE 34

❖Single Level Directory

- The simplest method is to have one big list of all the files on the disk. The entire system will contain only one directory which is supposed to mention all the files present in the file system.
- All the files are contained in the same directory.



Advantages

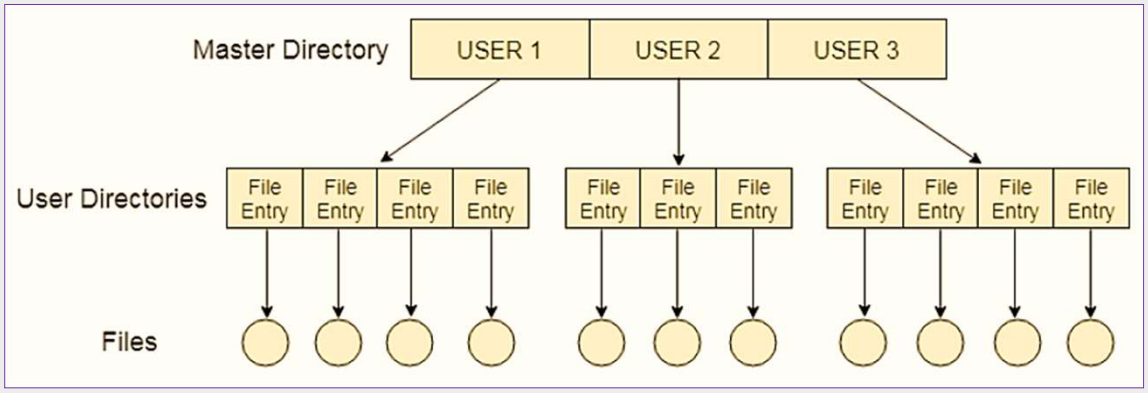
- Implementation is very simple.
- If the sizes of the files are very small then the searching becomes faster.
- File creation, searching, deletion is very simple since we have only one directory.

Disadvantages

- We cannot have two files with the same name.
- The directory may be very big therefore searching for a file may take so much time.
- Protection cannot be implemented for multiple users.
- There are no ways to group same kind of files.

❖Two Level Directory

- In two level directory systems, we can create a separate directory for each user.
- There is one master directory which contains separate directories dedicated to each user.
- For each user, there is a different directory present at the second level, containing group of user's file.
- The system doesn't let a user to enter in the other user's directory without permission.



➢Characteristics of two level directory system

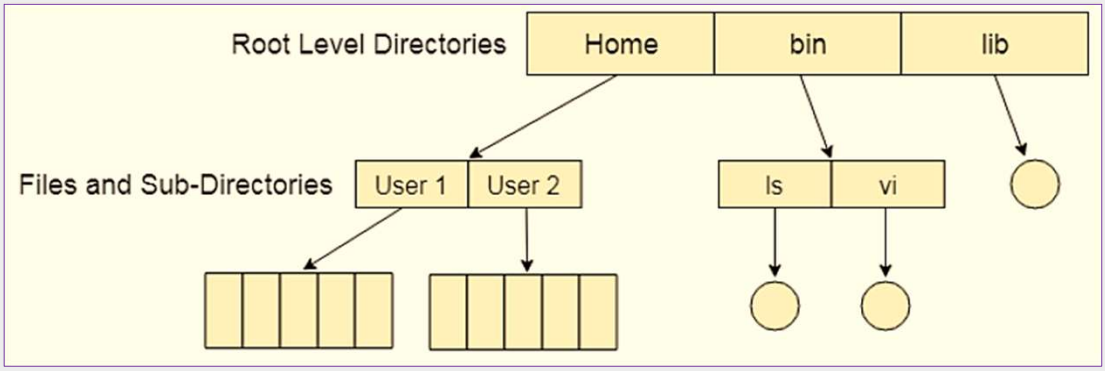
- Each files has a path name as /User-name/directory-name/
- Different users can have the same file name.
- Searching becomes more efficient as only one user's list needs to be traversed.
- The same kind of files cannot be grouped into a single directory for a particular user.

Every Operating System maintains a variable as PWD which contains the present directory name (present user name) so that the searching can be done appropriately.

❖Tree Structured Directory

- In Tree structured directory system, any directory entry can either be a file or sub directory. Tree structured directory system overcomes the drawbacks of two level directory system. The similar kind of files can now be grouped in one directory.
- Each user has its own directory and it cannot enter in the other user's directory. However, the user has the permission to read the root's data but he cannot write or modify this. Only administrator of the system has the complete access of root directory.
- Searching is more efficient in this directory structure. The concept of current working directory is used. A file can be accessed by two types of path, either relative or absolute.

- Absolute path is the path of the file with respect to the root directory of the system while relative path is the path with respect to the current working directory of the system. In tree structured directory systems, the user is given the privilege to create the files as well as directories.



➢Permissions on the file and directory

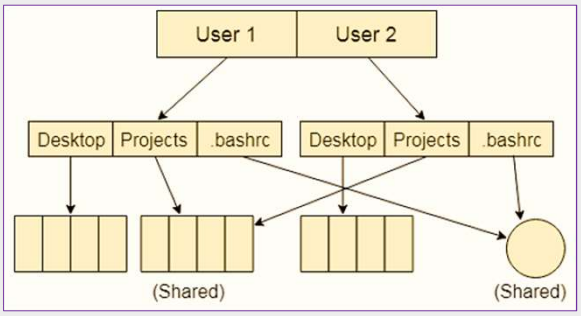
- A tree structured directory system may consist of various levels therefore there is a set of permissions assigned to each file and directory.
- The permissions are **R W X** which are regarding reading, writing and the execution of the files or directory. The permissions are assigned to three types of users: owner, group and others.
- There is a identification bit which differentiate between directory and file. For a directory, it is d and for a file, it is dot (.)

❖Acyclic-Graph Structured Directories

- The tree structured directory system doesn't allow the same file to exist in multiple directories therefore sharing is major concern in tree structured directory system.
- We can provide sharing by making the directory an acyclic graph. In this system, two or more directory entry can point to the same file or sub directory. That file or sub directory is shared between the two directory entries.
- These kinds of directory graphs can be made using links or aliases. We can have multiple paths for a same file. Links can either be symbolic (logical) or hard link (physical).

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

- ➢In the case of soft link, the file just gets deleted and we are left with a dangling pointer.
- ➢In the case of hard link, the actual file will be deleted only if all the references to it gets deleted.



DISK SCHEDULING

- A process needs two type of time, CPU time and IO time.
- For I/O, it requests the Operating system to access the disk.
- However, the operating system must be fare enough to satisfy each request and at the same time, operating system must maintain the efficiency and speed of process execution.
- The technique that operating system uses to determine the request which is to be satisfied next is called disk scheduling.

❖Important terms related to disk scheduling

- **Seek Time :** Seek time is the time taken in locating the disk arm to a specified track where the read/write request will be satisfied.
- **Rotational Latency :** It is the time taken by the desired sector to rotate itself to the position from where it can access the R/W heads.
- **Transfer Time :** It is the time taken to transfer the data.
- **Disk Access Time :** Disk access time is given as,

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

- **Disk Response Time :** It is the average of time spent by each request waiting for the IO operation.

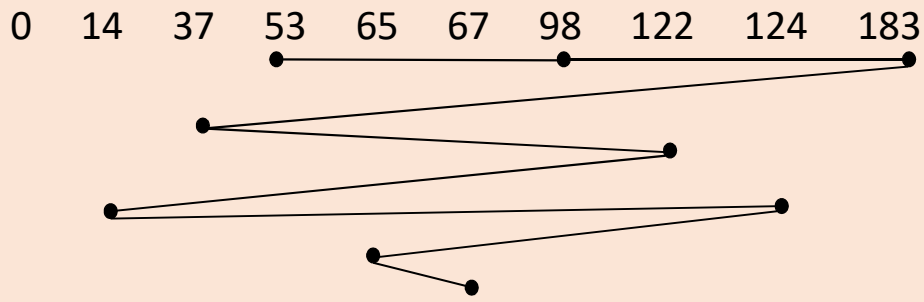
DISK SCHEDULING ALGORITHMS

- The main purpose of disk scheduling algorithm is to select a disk request from the queue of IO requests and decide the schedule when this request will be processed.

➢ **FCFS SCHEDULING**

- It is the simplest Disk Scheduling algorithm.
- It services the IO requests in the order in which they arrive.
- There is no starvation in this algorithm, every request is serviced.

Eg: Consider a disk queue with requests for I/O to blocks on cylinders 98, 183,37,122,14,124,65,67 in the order. The disk head is initially at the cylinder 53.



No. of Head movements = (183-53)+(183-37)+(122-37)+(122-14)+(124-14)+(124-65)+(67-65) = 640

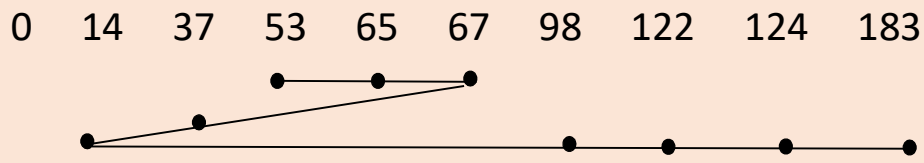
➢SSTF SCHEDULING

- Shortest seek time first (SSTF) algorithm selects the disk I/O request which requires the least disk arm movement from its current position regardless of the direction. It reduces the total seek time as compared to FCFS.

▪ **Disadvantages**

- It may cause starvation for some requests.
- Switching direction on the frequent basis slows the working of algorithm.
- It is not the most optimal algorithm

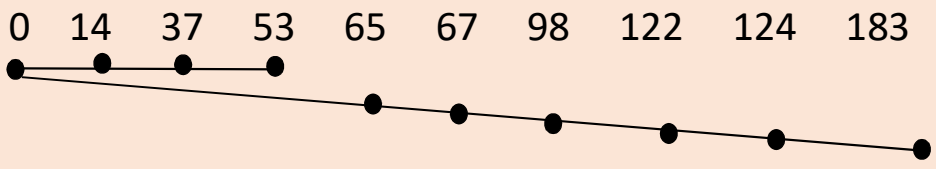
Eg: queue = 98,183,37,122,14,124,65,67. Head starts at 53.



No. of head movements = (67-53)+(67-14)+(183-14) = **236**

➢SCAN SCHEDULING

- It is also called as Elevator Algorithm.
- In this algorithm, the disk arm moves into a particular direction till the end, satisfying all the requests coming in its path, and then it turns back and moves in the reverse direction satisfying requests coming in its path.
- It works in the way an elevator works, elevator moves in a direction completely till the last floor of that direction and then turns back.
- Eg: queue = 98,183,37,122,14,124,65,67. Head starts at 53. Assume that the disk arm is moving towards left(to zero)

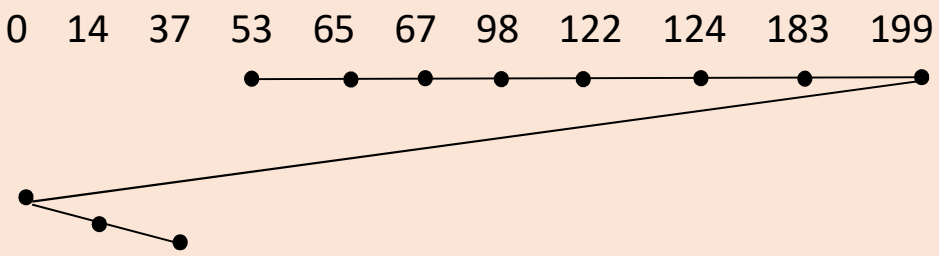


No. of head movements = 53+183 = **236**

➢C – SCAN SCHEDULING

- In C-SCAN algorithm, the arm of the disk moves in a particular direction servicing requests until it reaches the last cylinder, then it jumps to the last cylinder of the opposite direction without servicing any request then it turns back and start moving in that direction servicing the remaining requests.

- Eg: Queue = 98,183,37,122,14,124,65,67 . Head starts at 53.

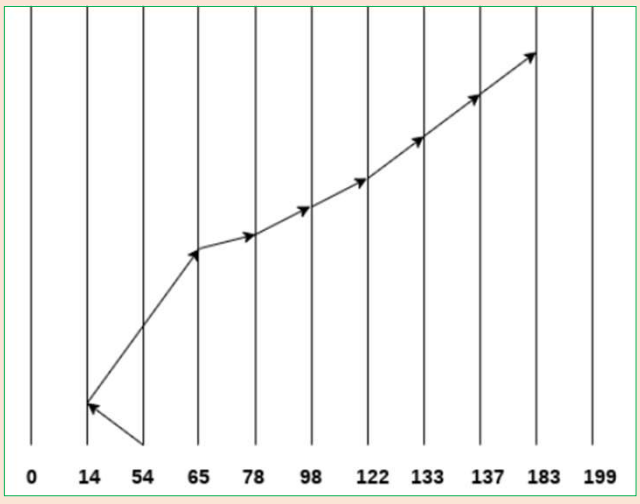


No. of head movements = (199-53)+37 = **183**

➢LOOK SCHEDULING

- It is like SCAN scheduling Algorithm to some extant except the difference that, in this scheduling algorithm, the arm of the disk stops moving inwards (or outwards) when no more request in that direction exists.
- This algorithm tries to overcome the overhead of SCAN algorithm which forces disk arm to move in one direction till the end regardless of knowing if any request exists in the direction or not.

Eg: Consider the following disk request sequence for a disk with 100 tracks 98, 137, 122, 183, 14, 133, 65, 78 Head pointer starting at 54 and moving in left direction. Find the number of head movements in cylinders using LOOK scheduling.

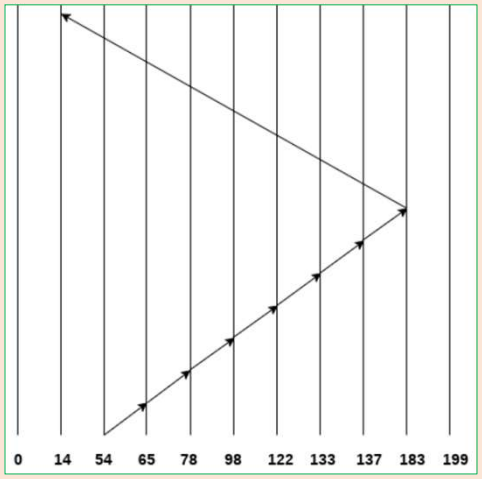


Number of head movement = 40 + 51 + 13 + +20 + 24 + 11 + 4 + 46 = **209**

➢C Look Scheduling

- C Look Algorithm is similar to C-SCAN algorithm to some extent.
- In this algorithm, the arm of the disk moves outwards servicing requests until it reaches the highest request cylinder, then it jumps to the lowest request cylinder without servicing any request then it again start moving outwards servicing the remaining requests.
- It is different from C SCAN algorithm in the sense that, C SCAN force the disk arm to move till the last cylinder regardless of knowing whether any request is to be serviced on that cylinder or not.

Eg: Consider the following disk request sequence for a disk with 100 tracks 98, 137, 122, 183, 14, 133, 65, 78 . Head pointer starting at 54 and moving in left direction. Find the number of head movements in cylinders using C LOOK scheduling.



Number of head movements = 11 + 13 + 20 + 24 + 11 + 4 + 46 + 169 = **298**