UNIT -5

File Management

File systems :

The file system provides the mechanism for on –line storage of and access to both data and programs of the OS and all the users of the computer system.

The file system consists of two distinct parts:

1. Files
2. Directory structure.

A collection of files , each storing related data , and a directory structure , which organizes and provides information about all the files in the system.

Basic concepts of file

1. **FILE**

The file system is collection of related information that is programs and data.

2. **FILE ATTRIBUTES**

A file has certain other attributes : They are

- NAME

  The symbolic file name is the only information kept in human – readable form.

- TYPES

  This information is needed for those system that support different types.

- LOCATION

  This information is a pointer to a device and to the location of the file on that device.

- SIZE

  The current size of the file, and possibly the maximum allowed size are included in this attribute.

- PROTECTION

  Access-Control information controls who can do reading, writing, executing, and so on.
- TIME ,DATE AND USER IDENTIFICATION

  This information may be kept for
  - Creation
  - Last modification
  - Last use.

These data can be useful for protection, security, and usage monitoring.

File Operation

A file is an abstract data type. There are six basic file operations.

- Creating a file
  - Two steps are necessary to create a file.
    - First, space in the file system must be found for the file.
    - Second, an entry for the new file must be made in the directory.
- Writing a file
  - To write a file, we use a system call specifying both the name of the file and the information to be written to the file.
  - The system must keep a write pointer to the location in the file where the next write is to take place.
- Reading a file
  - To write a file, we use a system call that specifies the name of the file and where the next block of the file should be put.
  - The system needs to keep a read pointer to the location in the file where the next read is to take place.
- Repositioning within a file
  - The directory is searched for the appropriate entry and the current-file position is set to given value.
  - This file operation is also known as a file seeks.
- Deleting a file
  - To delete a file we search the directory for the file. Having found the associated directory entry, we release all file space and erase the directory entry.
- Truncating a file
  - This operation makes the attributes of a file to remain the same, but erases the contents of the file.

- This function allows all attributes to remain unchanged but for the file to be reset to length zero.

❖ Appending a file
- This operation appends new information to the end of an existing file.

❖ Renaming a file
- This operation changes the file name.

❖ Coping a file
- This operation copy one file to another file.

Open file table

To avoid constant searching, Os keeps a small table containing information about all open files called open file table.

- **Multiuser Environment**
  - The implementation of the open and close operations in a multiuser environment, such as UNIX, is more complicated.
  - The OS uses two levels of internal tables.

  i) Per-Process table

  ii) System wide table.

i. **Per-Process table**

The Per-Process table contains information regarding the use of the file by the process.

ii. **System wide table.**

The System wide table contains information that is process independent, such as the location of the file on disk, access dates, and file size.

Additional Attributes

Additional Attributes included in open file table.

- **File Pointer**

  The system must track the last read/write location as a current file position pointer. This pointer is unique to each process operating on the file.

- **File open count**

  This counter tracks the number of opens and closes, and reaches zero on the last close. The system can then remove the entry.

- **Disk location of the file**

  The information needs to locate the file on disk is kept in memory to avoid having to read it from disk for each operation.

File Types

A common technique for implementing file types is to include the types as part of the file name. The name is split into two parts:---a name and an extension.

File Type	Usual extension	Function
Executable	exe, com, bin or none	Ready to run machine-language program
Object	obj,.O	Compiled, machine-language, not linked
Source Code	c, p, pas,f77, asm, a	Source code in various languages.
Batch	bat, sh	Commands to the commannd nterpreter
Text	txt, doc	Textual data, documents
Word processor	wp, tex, rrf, etc	Various word-processor formats
Library	lib, a	Libraries of routines for programmers
Print or view	ps, dvi, gif	ASCII or binary file in a format for printing or viewing
Archive	arc, zip, tar	Related files gropued into one file, sometimes compressed for archiving of storage

****

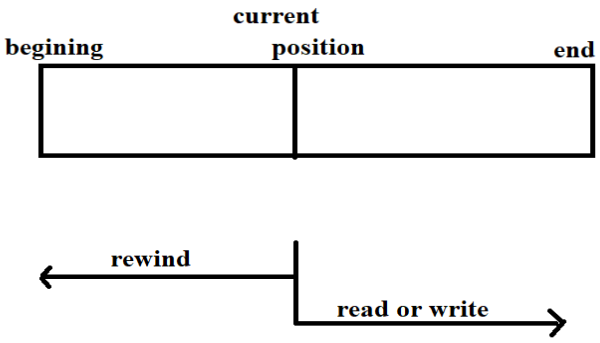
Access Methods

Files store information. There are several ways that the information in the file can be accessed.

Sequential Access

- The simplest access method is sequential access.
- Information in the file is processed in order, one record after the order.
- A read operation reads the next portion of the file and automatically advances a file pointer.
- A write operation appends to the end of the file and advances the file pointer to the end of the newly written material.
- A program may be able to skip forward or backward **n** records.

- Sequential access is based on a tape model of a file.



Direct Access

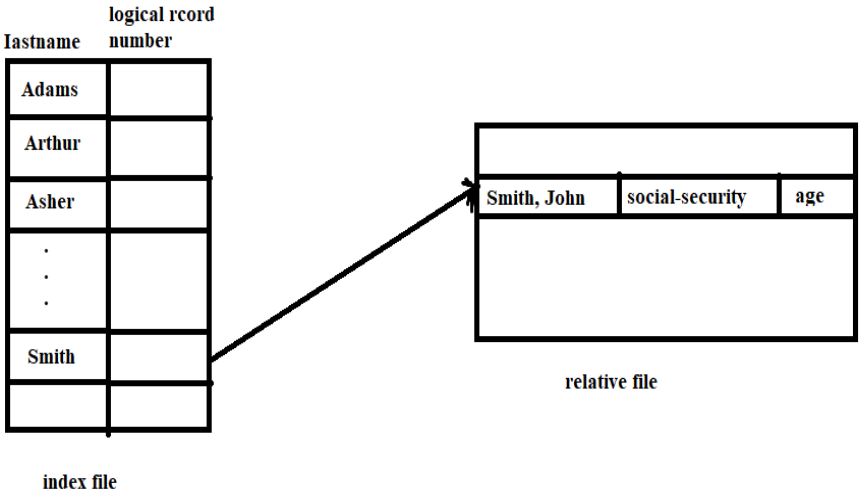
- A direct access is also called as relative access.
- A file is made up of fixed-length logical records that allow programs to read and write records rapidly in no particular order.
- The direct access method is based on a disk model of a file
- Direct access file are of great use for immediate access to large amounts of information.
- Here it is possible to read block directly to provide the desired information.
- The two file operations performed are read **n**, write **n**, and where **n** is the block number.
- The block number provided by the user to the operating system is normally a relative block number.
- It is easy to simulate sequential access on a direct-access file. It we simply keep a variable **CP,** which define our current position, then we can simulate sequential file operations, as shown below.

Sequential Access	Implementation for direct access

reset	CP : = 0
read next	read CP CP : = CP+1
write next	write CP CP : = CP+1

Index Method

- Index method can be built on top of a direct access method.
- The index contains pointer to the various blocks.
- To find an entry in the file, one first search the index and then use the pointer to access the file directly and to find desired entry.
- With large files the index file itself may become too large to be kept in memory.
- One solution is to create an index for the index file.
- The primary index file would contain pointers to secondary index files, which would point to the actual data items.
- The secondary index blocks pointers to the actual file blocks.



****

Directory Structure

- Directory are basically symbol tables of files. A single flat directory can contain a list of all files in a system. A directory contains information about the files, including attributes, location and ownership. Operating system manages this types of information.
- Operations that may be performed on the directory as follows.
    1. Search
    2. Create a file
    3. Delete file
    4. Rename a file
    5. List directory

1. **Search:**
   Directory structure is searched for finding particular file in the directory. Files have symbolic names and similar names may indicate a relationship between files.
2. **Create a file:**
   When a new file is created, an entry must be added to the directory.

3. **Delete file:**

   When a file is deleted, an entry must be added to the directory.

4. **Rename a file:**

   Name of the files must be changeable when the content or use of the file changes. Renaming a file may also allow its position within the directory structure to be changed.

5. **List directory:**

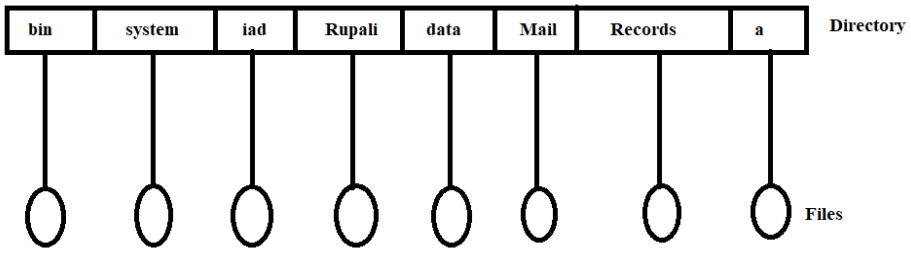
   All or portion of the directory may be requested. Request is made by a user and result in a listing of all files owned by that user plus some of the attributes of each file.\

Different types of directory structures are given below.

1. Single level directory
2. Two level directory
3. Tree structured directory
4. Acyclic graph directories
5. General graph directory

Single Level directory

Single level directory structure is simple directory structure. All files are contained in the same directory. Single level directory structure. Easy to implement and maintain.



Single level directory

Disadvantage of single level directory are as follow:

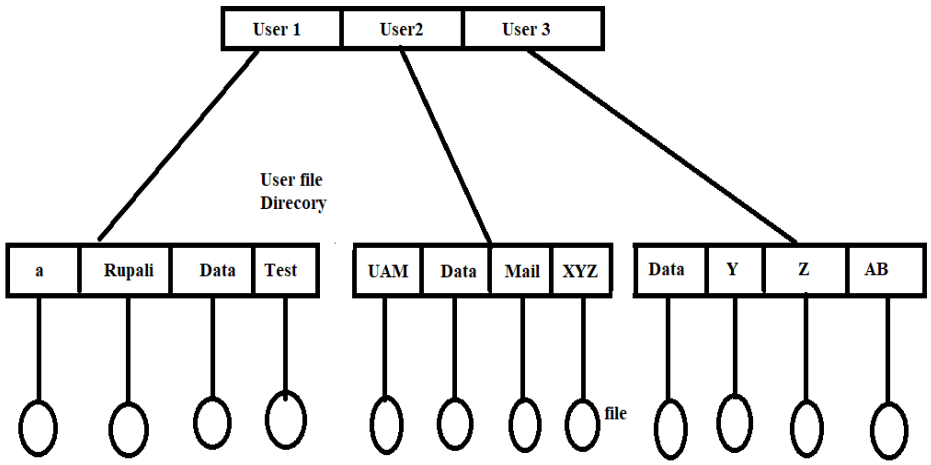
1) Not suitable for a large number or files and more than one user.

2) Because of single directory, files require unique file name.
3) It is difficult to remember the names of all the files as the number of files increases.

MS-DOS operating system allows only 11 character file names whereas Unix allows 225 characters.

Two Level Directory

In two level directory, each user has his own directory. It is called user file directory(UFD). Each user file directory has a similar structure. When a user refers to a particular file, only his own UFD is searched. Different users may have files with the same name, as long as all the file names within each UFD are unique.



Two Level Directory Structure

To create a file for a user, the operating system searches only that users directory to ascertain whether another file of that name exists. To delete a file, the operating system confines the search to the local UFD.

Operating system cannot accidently delete another user's file that has same name.

Advantages:

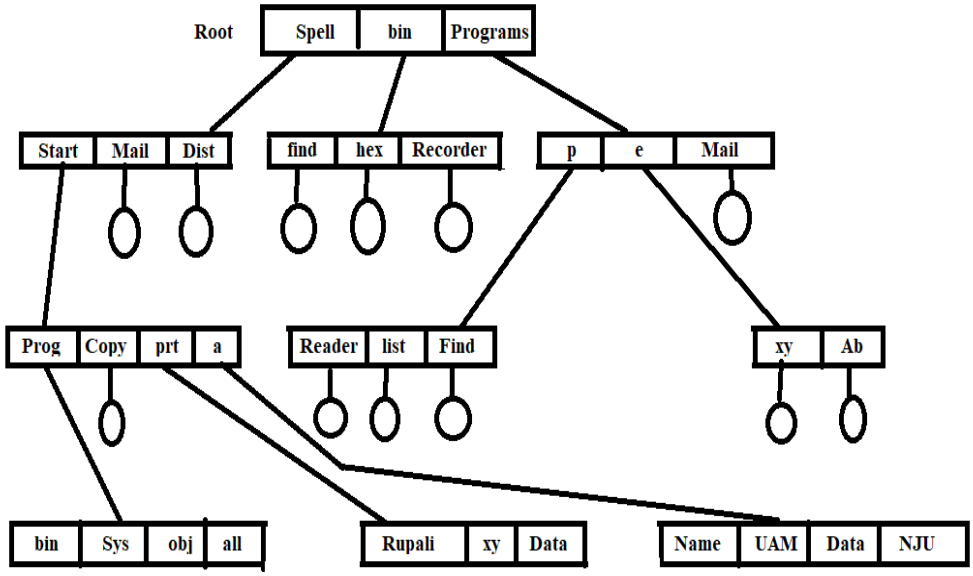
Solves name collision problem.

Disadvantage:

Users are independent and so cannot cooperate.

Tree Structured Directories:

- MS-DOS system is tree structure directory. It allows users to create their own subdirectory and to organize their files accordingly. A subdirectory contains a set of files or subdirectories. A directory is simple another file but it is treated in a special way.
- All the directories have the same internal format. Once bit in each directory entry defines the entry as a file(0) or as a subdirectory(1). Special system calls are used to create and delete directories.



Tree Structure directory

- In normal use, each has a current directory. Current directory should contain most of the files that are of current interest to the user. When a reference is made to a file, the current directory is searched. Path name is used to search or for any operation on file with another directory.
- Path names can be of two types:
  a) Absolute path name.
  b) Relative path name.
    - **Absolute Path name:** begins at the root and follows a path down to the specified file, giving the directory names in the path.

- **Relative path name:** defines a path from the current directory. MS-DOS will not delete a directory unless it is empty, for deleting a directory two approaches can be taken
  1. User must delete all the files from the directory. Make it empty directory.
  2. In Unix, rm command is used with some option for deleting directory.

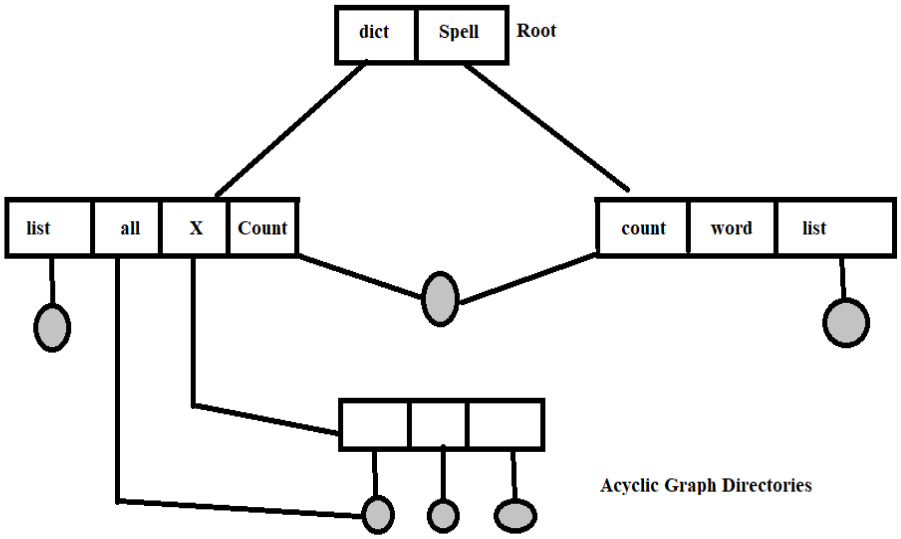
❋ When a request is make to delete a directory, all that directory files and subdirectories are also to be deleted. A path to a file in a tree structured directory can be longer than that in a two level directory.

Advantage:

User can access files of other users using path name.

Acyclic Graph Directories

- It allows directories to have shared subdirectories and files.
- Same file or directory may be in two different directories.
- Graph with no cycles is a generalization of the tree structured scheme.
- Shared files and subdirectories can be implemented by using links or we can duplicate all information about them in both shared directories.
- A link is effectively a pointer to another file or subdirectory.
- A links is implemented as an absolute is more flexible then a simple tree structure, but sometimes it is more complex.
- Figure shows a acyclic graph directory structure.



Disadvantage:

- Multiple absolute path names, traverse shared files more than once.
- Sometimes deletion leads to have dangling pointer to non-existent file. Deletion problem can be overcome in two ways.
  1. Remove link.
  2. Maintain file reference list which keep the count of the number of references.

General Graph Directory

Cycles are allowed to exist in general graph directory.

FILE PROTECTION

- The protection of a file is mostly needed in multi-user environment where a file is shared among several users.
- On system which does not permit access to the files of other users, protection is not required.

- Protection mechanism must provide controlled access by restricting the types of files which can be made.
- Access is permitted or denied depending upon several factors one of which is the type of access requested.

Different types of operations may be controlled. These operations are:

- Reading from the file
- Writing on the file
- Executing the file by loading into main memory.
- Deleting the file and releasing the space.

- Many different protection mechanisms have been proposed. Each one has got advantages and disadvantages and must be selected which is suitable for its application.
- A large computer system requires different types of protection mechanisms compared to a small computer system used by a small group of people.
- The protection can be supported with file itself but the more common scheme is to provide protection with path.
- Thus if a path name refers to a directory, the user must be allowed access to both the directory and the file.
- In systems where files may have numerous path names, a given user may then have different access rights to a file depending upon the path name used.

Password:

In this approach, password is associate with each file. Just as a password is required to access a computer system, access to each file will be also controlled by a password.

There are, however, several disadvantages to this scheme:

- If we associate a separate password with each file, the number of passwords that need to be remembered are quite large, making the scheme impractical.

- If only one password is used for all the files, then once it is discovered all files are accessible.
- Some systems allow a used to associate a password with a subdirectory rather than an individual file, to deal with this problem.
- Commonly, only one password is associated with each file.
- Thus protection is on an all-or-nothing basis.
- To provide protection on a more detailed level multiple passwords are needed.

Access lists:

- Another approach is to make access dependent on the identity of the user.
- Various users may need different types of access to a file or directory.
- An access list can be associated with each file and directory, specifying the user name and the types of access allowed-for each user.
- When a user requests access to a particular file, the operating system checks the access list associated with each file.
- If that user is listed for the requested access, the access is allowed.
- Otherwise, a protection violation occurs and the user job is denied access to the file.

Access Groups:

The main problem with access lists can be their length. If we want to allow everyone to read a file, we must list all users with read access. This has two undesirable consequence.

- Constructing such a list may be a tedious and unrewarding task, especially if we not know in advance the list of users in the system.
- The directory entry which previously was of fixed size needs now to be of variable size, resulting in space management being more complicated.

These problems can be resolved by using a condensed version of the access list.To condense the length of the access list, many systems recognize three classifications of users in connection with each file

- **Owner-**The user who created the file.
- **Group-** A set of users who are sharing the file and need similar access.
- **Universe-** All other users in the system.

****

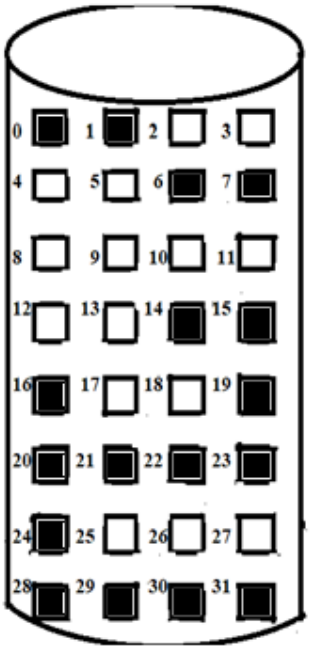
FILE ALLOCATION METHOD

Three major methods of allocating disk space are in wide use:

1. Contiguous
2. Linked
3. Indexed

Contiguous Allocation

The contiguous allocation method requires each file to occupy a set of contiguous blocks on the disk.



First	Start	Length
Count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

The directory entry for each file indicates the address of the starting block and the length of the area allocated for this file.

Accessing a file that has been allocated contiguously is easy.

Disadvantage

- **External Fragmentation:**
  External fragmentation exists whenever free space is broken into chunks.
- **Space Allocation:**
  A major problem is determining how much space is needed for a file. When the file is created, the total amount of space it will need must be found and allocated.
  - ✓ If too little space is allocated to a file, then the file cannot be extended in the future.
  - ✓ If more space is allocated and only less space is occupied, then much of the space may remain unused for a long time. This is called as internal fragmentation.

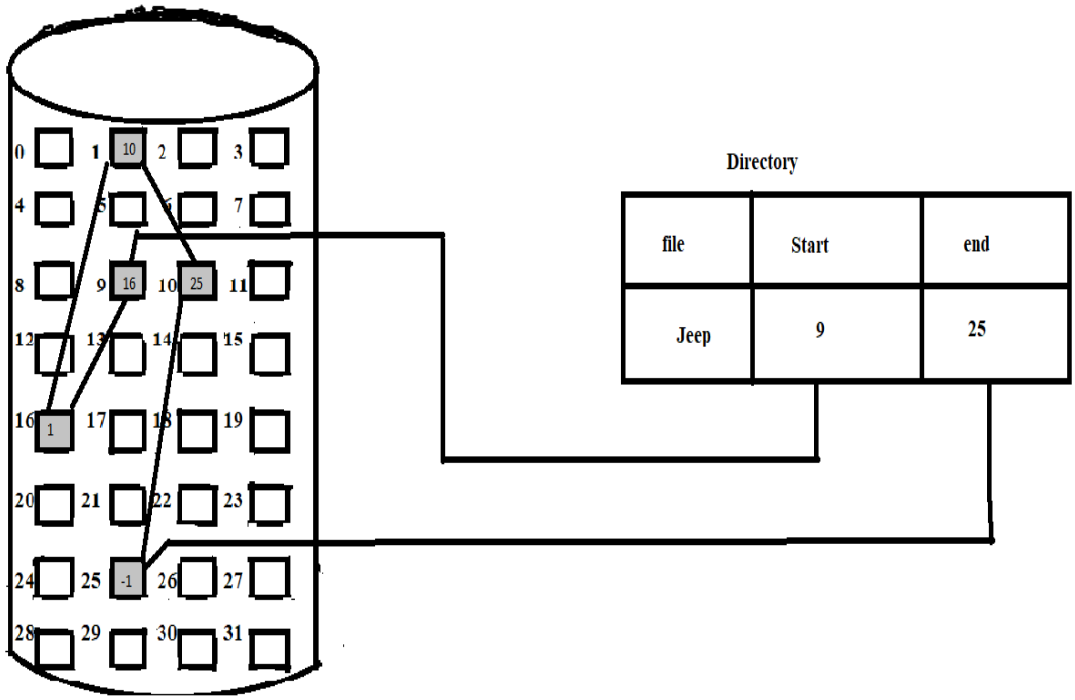
Linked Allocation

* Linked allocation solves all problems of contiguous allocation.

* With linked allocation, each file is a linked list of disk blocks.

*The disk blocks may be scattered anywhere on the disk.

* The directory contains a pointer to the first and last blocks of the file. Each block contains a pointer to the next block.



Advantages:

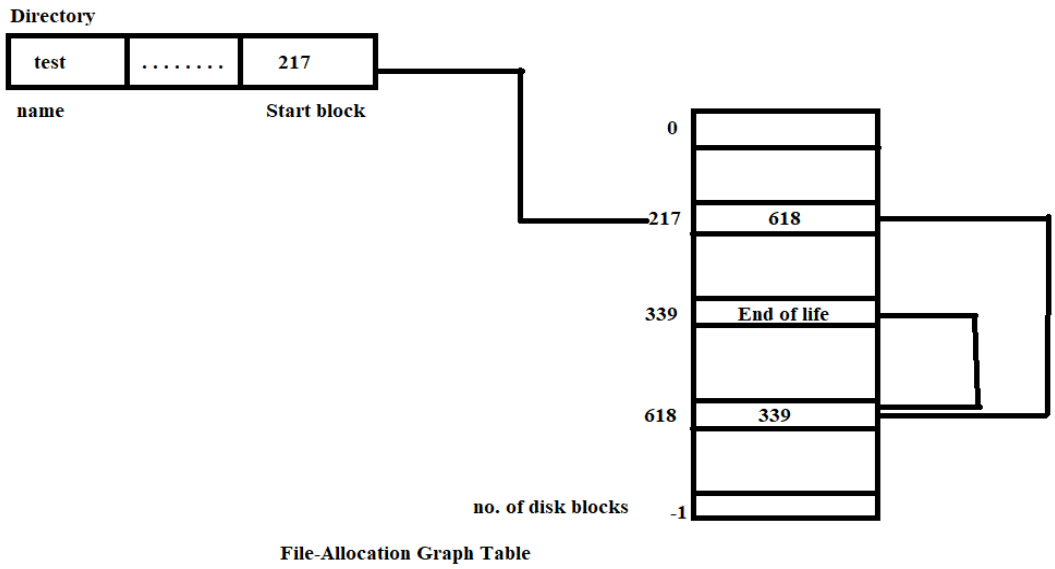
There is no external fragmentation with linked allocation and any free block on the free space list can be used to satisfy a request.

Disadvantage

- **Sequential Access:** The major problem is that it can be used effectively for only sequential access files and in efficient for direct access.
- **Space for Pointer:** Another disadvantage to linked allocation is the space required for the pointer.
- **Reliability:** Since the files are linked together by pointers scattered all over the disk, if a pointer is lost or damaged then entire rest of content will be lost.

File Allocation Table:

An important variation of the linked allocation method is the use of a file allocation table(FAT).

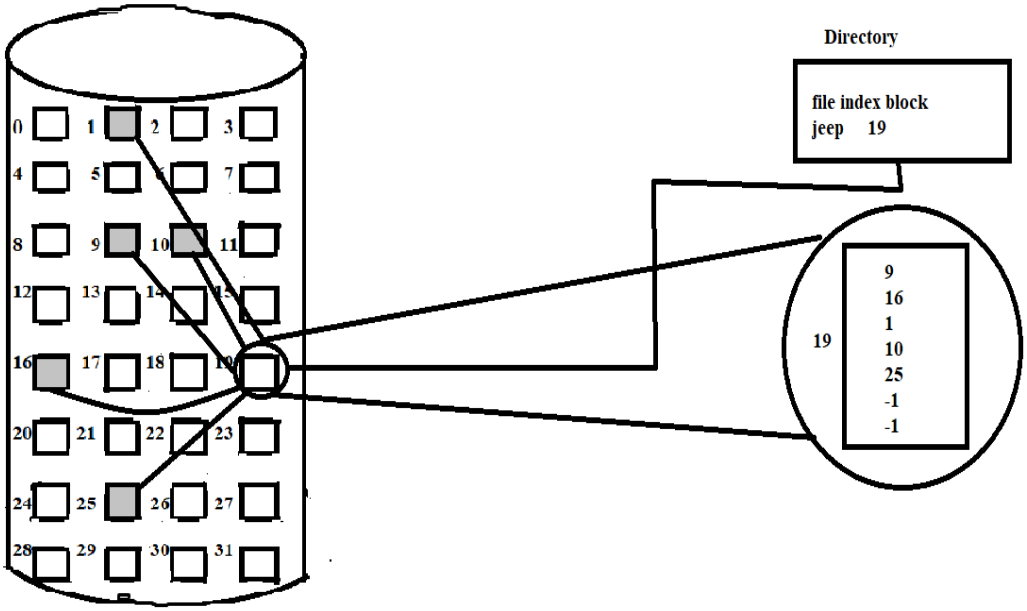


File-Allocation Graph Table

- The table has one entry for each disk block and is indexed by block number.
- The fat is used much as the block number of the first block of the file.
- The table entry indexed by that block number then contains the block number of the next block in the file.
- This chain continues until the last block which has special end of file value as the table entry.

Indexed Allocation:

- Each file has its own index block which is an array of disk block addresses. The i^{th} Entry in the index block points to the i^{th} block of the file.
- The directory contains the address of the index block.



Advantages:

No external fragmentation and equal support for sequential and random access.

Problems:

- Wastes space on index block. Possible big internal fragmentation.
- Needs special mechanism for treating big files because all block numbers do not fit in single index block.

Possible Mechanisms:

- Linked Scheme
- Multilevel Index
- Combined Scheme.

Linked Scheme

*An index block is normally one disk block.

*Thus, it can be read and written directly by itself.

* To allow for large files, we may link together several index blocks.

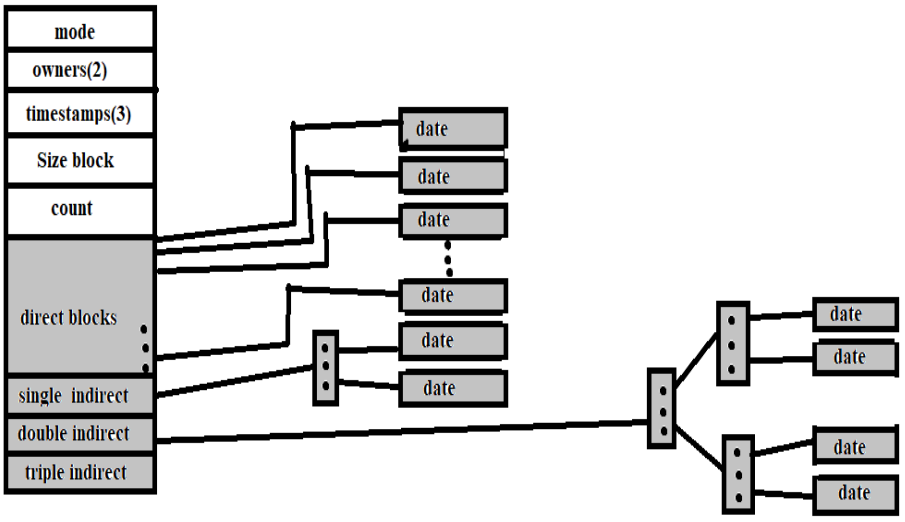
Multilevel Index

*A variant of the linked representation is to use a first-level index block to point to a set of second-level index blocks, which in turn point to the file blocks.

* To access a block, the operating system uses the first-level index to find a second-level index block, and that block to find the desired data block. This approach could be continued to a third or fourth level, depending on the desired maximum file size.

Combined Scheme

Another alternative is to use part of the block as one level index block, use next i entries of the block to store the number of first level index block for 2-level, 3-level,…..i+1 level indexes.



The UNIX incode

****

Free Space Management

*Since there is only a limited amount of disk space; it is necessary to reuse the space from deleted files for new files, if possible.

*To keep track of disk space, the system maintains a free-space list.

* The free space list records all disk blocks that are free.

*To create a file, we search the free-space list for the required amount of space, and allocate that space to the new file.

*This space is then removed from the free-space list, when a file is deleted, its space is added to the free-space list.

*There are several different structures that are used by different OS to keep track of free memory.

I. Bit vector
II. Linked list
III. Grouping
IV. Counting.

Bit vector

OS maintain the vector of bits. Each block is represented by one bit.

Example

Consider a disk where blocks 2,3,4,5,8,9,10,11,12,13,17,18,25,26,27 are free, and the rest of the blocks are allocated. The free-space bit map would be

12345…… 27 .

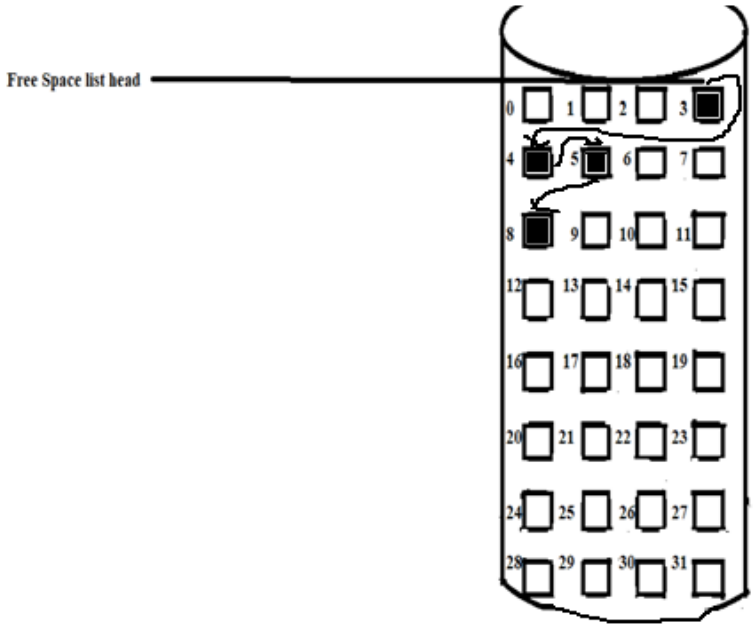
01111100111111000110000001110000001……

Advantage:

Relatively simple & efficient to fine first **n** consecutive free blocks one the disk.

Linked List

- Create linked list of free blocks.
- First free block number is stored by OS in special location.



linked free-space list on disk …….

Advantage:

If linked allocation is used for files, same mechanism can be reused for free space management.

Disadvantage:

Inefficient and Unreliable.

Grouping:

- ✓ Use one free block as index, that stores addresses of other free blocks.
- ✓ First n-1 blocks stored in index block are actually free.
- ✓ Last block stored in free index is the next free index number.

Advantage:

Improves Efficiency and reliability.

Counting

Another approach is several contiguous blocks, may be allocated or freed simultaneously, particularly when space is allocated with the contiguous allocation algorithm.

Thus, rather than keeping a list of n free disk addresses, we can keep the address of the first free block and the number n of free contiguous blocks that follow the first block. Each entry in the free-space list then consists of a disk address and a count.

Advantage:

Works well for contiguous allocation mechanism and for cluster allocation.

****

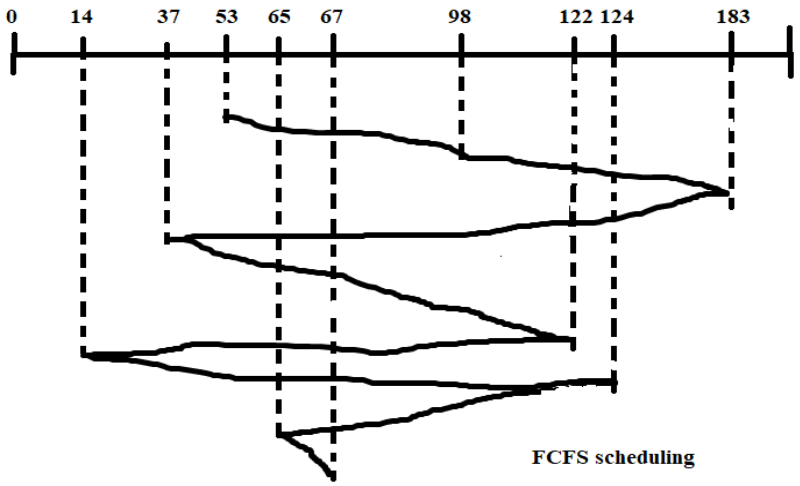
DISK SCHEDULING

- ✓ The performance of a computer system is to large extent dependent upon how fast a disk request is serviced.
- ✓ To service a request, a disk system requires that the head be moved to the desired track, then a wait for latency and finally the transfer of data.
- ✓ Fast access of disk depends on
  1.seek time
  2.rotational latency
  Seek Time:
  Time taken to move the head to the cylinder.
  Rotational latency:
  Time taken to rotate the desired sector to disk head.
  In the following section we will examine several scheduling algorithms.

First-Come-First-Served(FCFS) Scheduling

- ➢ This is the simplest form of disk scheduling in which the first request to arrive is the first one serviced.
- ➢ FCFS is easy to program, however, it may not provide the best service. Consider, for example, disk queue with requests involving tracks to read 98,183,37,122,14,124,65,and 67 in that order.

- If the disk head is initially at cylinder 53, it will first move from 53 to 98, then to 183, 37,122,14,124,65 and finally to 67 for a total head movement of 640 cylinders.
- Figure shows FCFS disk scheduling.



- FCFS is easy to implement but it does not provide fastest service. It cannot takes special action to minimize the overall seek time.
- FCFS is acceptable when the load on a disk is light but as the load grows, response time becomes longer.

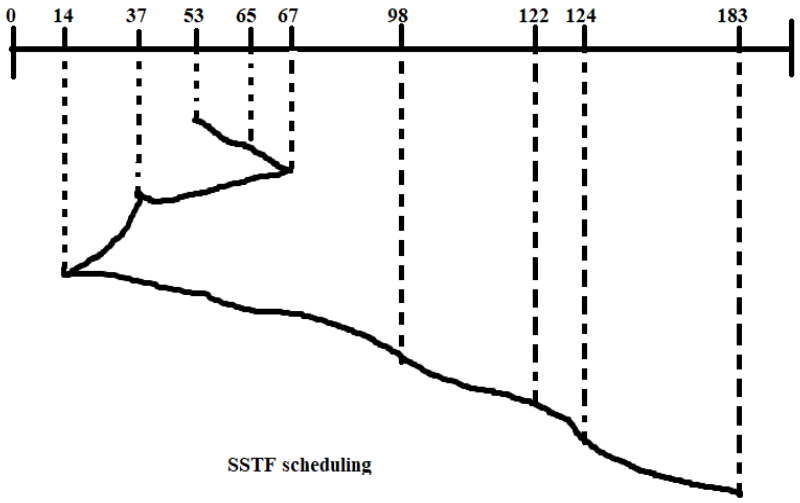
Band width:

Band width=total no of bytes transfered by total time between first request and last transfer.

Shortest-Seek-Time-First-Scheduling

- In shortest-seek-time(SSTF) scheduling priority is given to those processes which need the shortest seek, even if these requests are not the first ones in the queue.
- It means that all requests nearer to the current head positions are serviced together before moving head to distant tracks.

- This approach is implemented by moving the head to the closest track in the request queue from current position.
- For example, the disk request queue contains a set of references for blocks on tracks, 98,183,37,122,14,124,65 and 67. Figure shows SSTF scheduling.



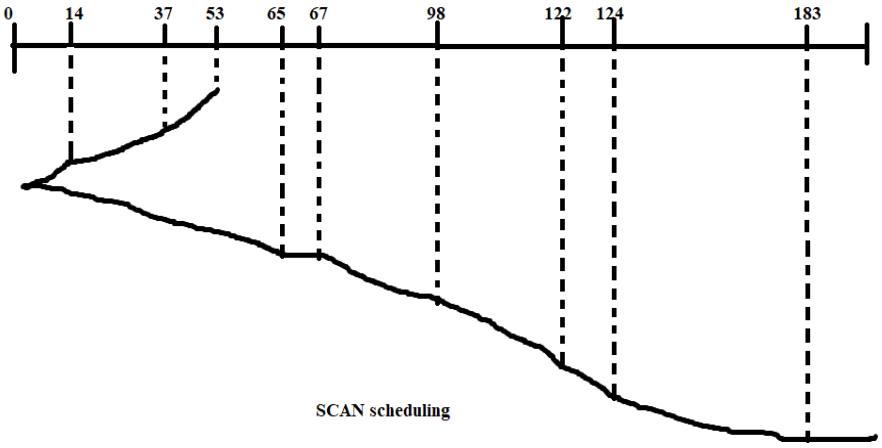
SSTF scheduling

- This scheduling method results in a total head movement of only 236 cylinders.
- Under heavy load SSTF can prevent distant requests from ever being serviced. This phenomenon is known as starvation. SSTF scheduling is essentially a form of shortest job first scheduling. SSTF scheduling algorithms are not very popular because of two reasons.
    1. Starvation possibly exists.
    2. It increases higher overheads.

Scan Scheduling

- In SCAN scheduling strategy the read/write head of a disk start from one end and move towards the other end, services requests as it reaches each track until-it reaches to another end of the disk. After reaching another end of disk, disk head reverse its path-direction while continuing with services whichever comes on the way. This way disk head continuously oscillates from end to end.

- If any request comes on its way it will be serviced immediately, while request arriving just behind the head will have to wait until disk head moves to the end of the disk, reverses direction and returns before being serviced. Scan has been the basis of most disk scheduling strategy actually implemented. The Scan algorithm is sometimes called the "**Elevator"** algorithm, since it is similar to the behaviour of elevators algorithm, as they service requests to move from floor to floor in building.
- Before applying SCAN to schedule the requests on cylinders 98,183,37,122,14,124,65 and 67,we need to know the direction of head movement, in addition to the heads current position(53).
- If the disk arm is moving toward 0, the head will service 37 and then 14.
- At cylinder 0, the arm will reverse and will move toward the other end of the disk, servicing the requests at 65,67,98,122,124 and 183.
- Figure shows the SCAN scheduling.



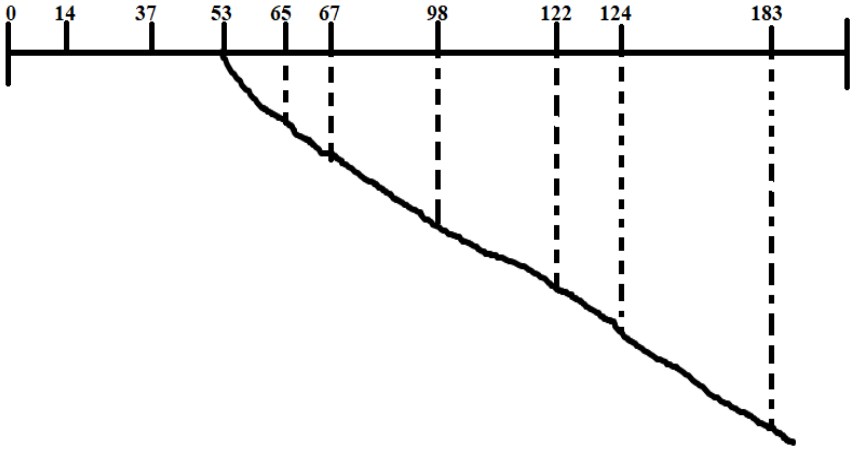
SCAN scheduling

- SCAN improve throughput because of the continuous scanning of disk from end to end, the outer tracks are visited less often than mid-range tracks, but this is not serious.

C-SCAN Scheduling

- Circular SCAN(C-SCAN) scheduling restricts scanning to one direction only. It is a variant of SCAN designed to provide a more uniform wait time.
- C-SCAN reduces the maximum delay experienced by new requests.

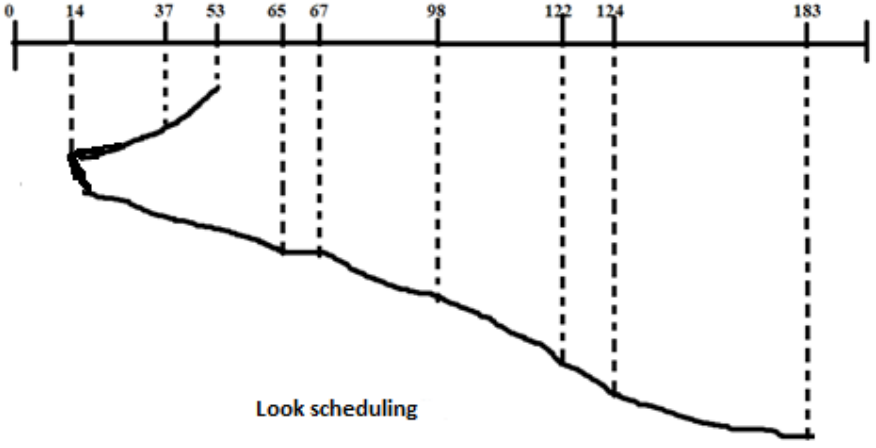
- C-SCAN moves the head from one end of the disk to the other, servicing requests along the way. When the head reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the returns trip. Figure shows the C-SCAN scheduling.



C-SCAN scheduling

Look Scheduling Algorithm

- Both scan and C-Scan move the disk arm across the full width of the disk but in look scheduling the arm goes only as far as the final request in each direction. Then it reverses direction immediately without going further.
- Like Scan and C-Scan versions we also have Look and C-Look.



Look scheduling