Chapter-(File Management)

What is file management?

Managing files and folders on the hard disk.

File management

Allocate space to file and folders.

File management

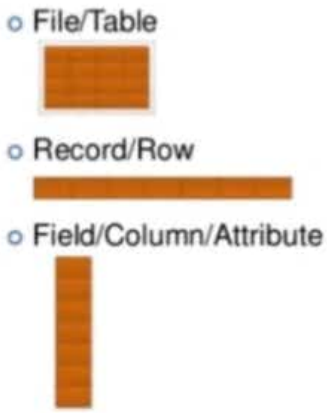
De-allocate space from
file and folders.

- A **file** is a collection of data stored on mass storage (e.g., disk or tape).
  - The data is subdivided into **records** (e.g., student information).
  - Each record contains a number of **fields** (e.g., roll number, name).
  - One (or more) field is the **key** field (e.g., roll number).

What is File? What is File Organization?

✓ **File:**

- File is a collection of records related to each other.
- The file size is limited by the size of memory and storage medium.



✓ **File Organization:**

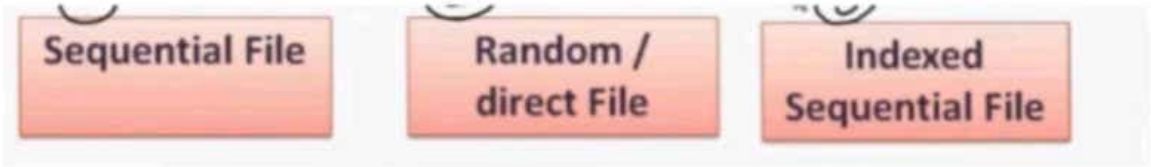
- File organization refers to the way data is stored in a file.

✓ **Objectives of File Organization:**

- It contains an optimal selection of records, i.e., records can be selected as fast as possible.
- To perform insert, delete or update transaction on the records should be quick and easy.
- The duplicate records cannot be induced as a result of insert, update or delete.
- For the minimal cost of storage, records should be stored efficiently.

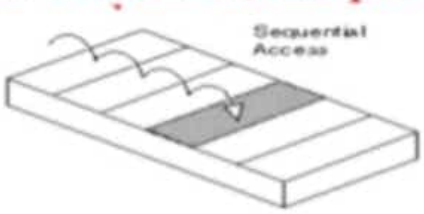
How Do We Organize Files on Storage Media?

- A technique for physically arranging the records of a file on secondary storage devices is called file organization.
- The files are organized on storage media in the following ways



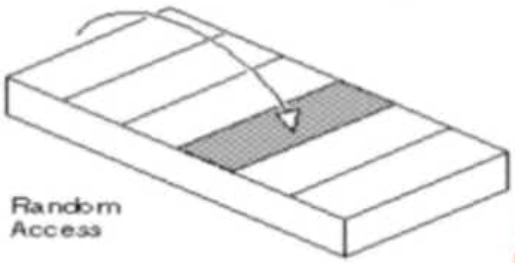
1.Sequential Files

- The records in sequential file organization are stored in sequence.
- A sequence means the records are stored one after the other.
- The records can be retrieved only in the sequence in which they were stored. The principal storage media for sequential files is magnetic tape.
- The **Major Disadvantage** of sequential access is that it is very slow. If the last record is to be retrieved, all preceding record are read before reaching the la



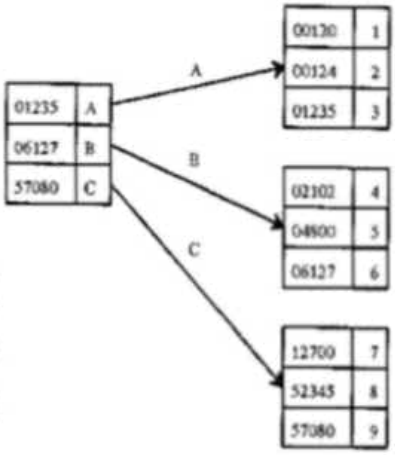
2. Direct or Random Files

- The records in direct file organization are not stored in a particular sequence. A key value of a record is used to determine the location to store the record. Each record is accessed directly without going through the preceding records.
- This file organization is suitable for storing data on disk.




Sequential access
1 2 3 4 5 6 7 8
Random access
1 3 7 2 8 6 4 5

3. Indexed Sequential Files



- In indexed sequential file organization, records are stored in ascending or descending order.
- The order is based on a value called key. Additionall indexed file organization maintains an index in a file
- An index consists of key values and the correspondi disk address for each record in the file.
- Index refers to the place on a disk where a record is stored. The index file is updated whenever a record is added or deleted from the file.

. Indexed Sequential Files (Cont)

- The records in indexed file organization can be accessed in sequential access as well as random access or direct access.
- The records in this file type require more space on storage media.
- **This method is slower than direct file organization as it requires to perform an index search.**

File Concept

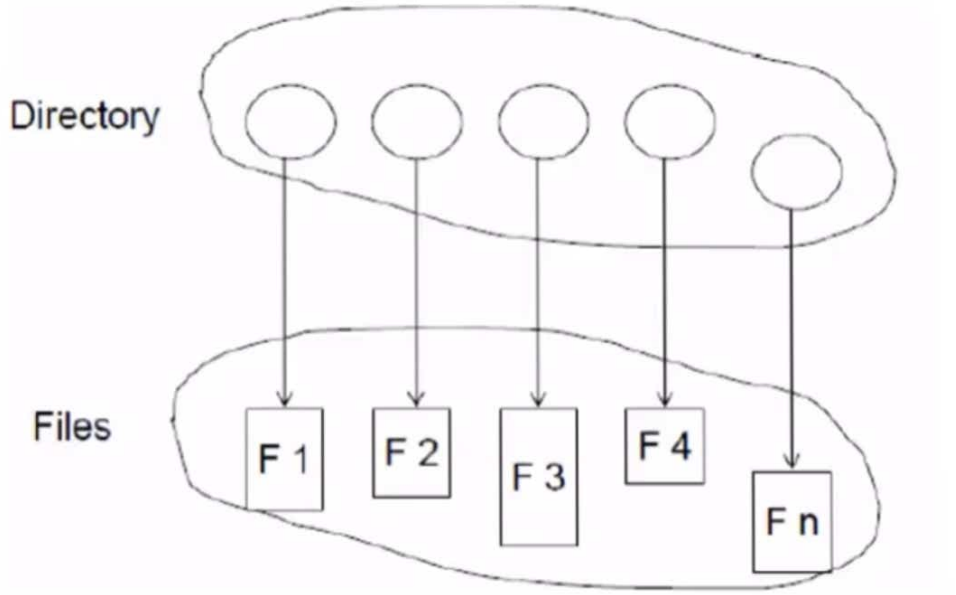
- File = a named collection of related info on secondary storage
- Data can't be written to secondary storage unless its in a file
- A file has a certain defined **structure** according to its type
    - **Text file**: sequence of characters organized into lines
    - **Source file**: sequence of subroutines & functions
    - **Object file**: sequence of bytes organized into blocks
    - **Executable file**: series of code sections

File Attributes

- **Name**: The only info kept in human-readable form
- **Identifier**: Unique tag
- **Type**: Info needed for those systems that support different types
- **Location**: Pointer to a device and to the location of the file
- **Size**: Current size (in bytes, words, or blocks)
- **Protection**: Access-control info
- **Time, date, & user id**: Useful for protection & usage monitoring
- Information about files are kept in the directory structure, which is maintained on the dis

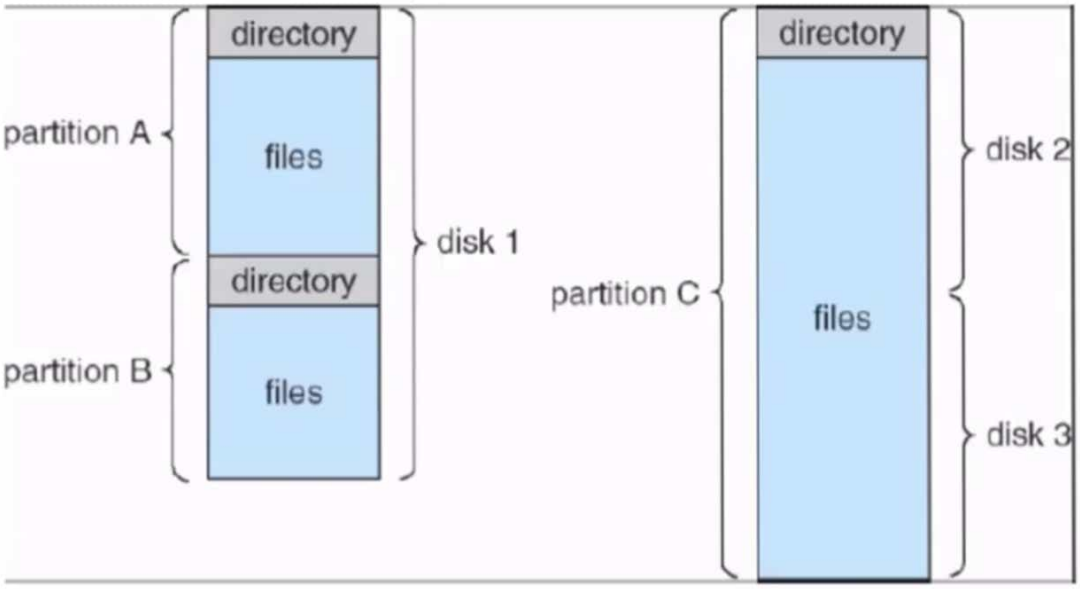
Directory and Disk Structure

- A collection of nodes containing information about all files



- Both the directory structure and the files reside on disk
- Backups of these two structures are kept on tapes

- **Typical File-System Organization:**



Storage Structure

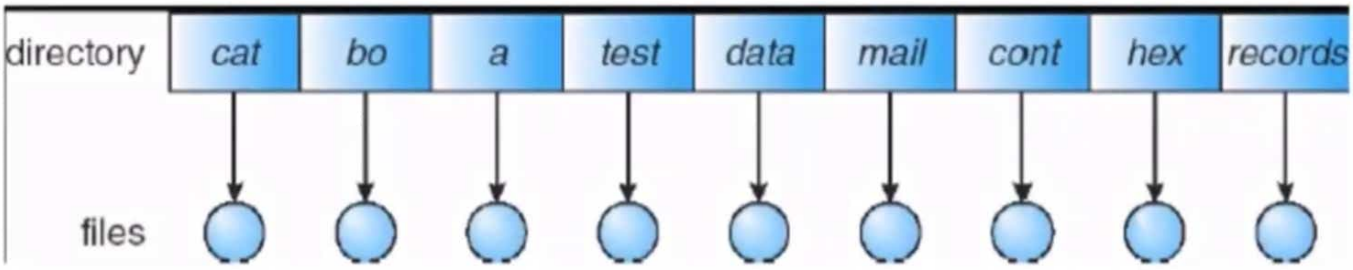
- Organizing a lot of data can be done in two parts:
  - Disks are split into one or more partitions
  - Each partition contains info about files within it (e.g. name, location, size, type…) in a device directory

Directory Overview

- Operations that can be performed on a directory:
  - Search for a file
  - Create a file
  - Delete a file
  - List a directory
  - Rename a file
  - Traverse the file system

Single-Level Directory

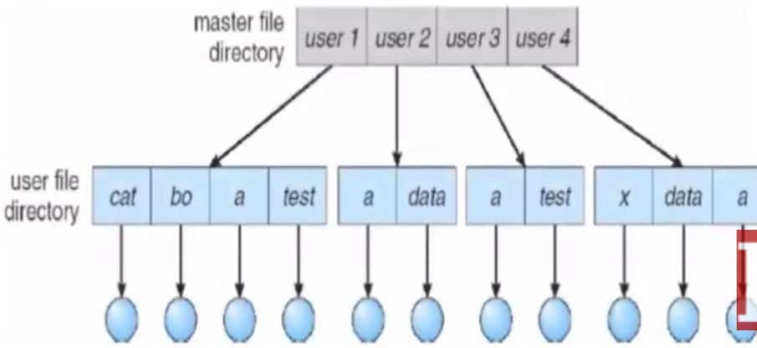
- All files are contained in the same directory
- Limitations because all files must have unique names
- A single directory for all users



- Naming problem
- Grouping problem

Two-Level Directory

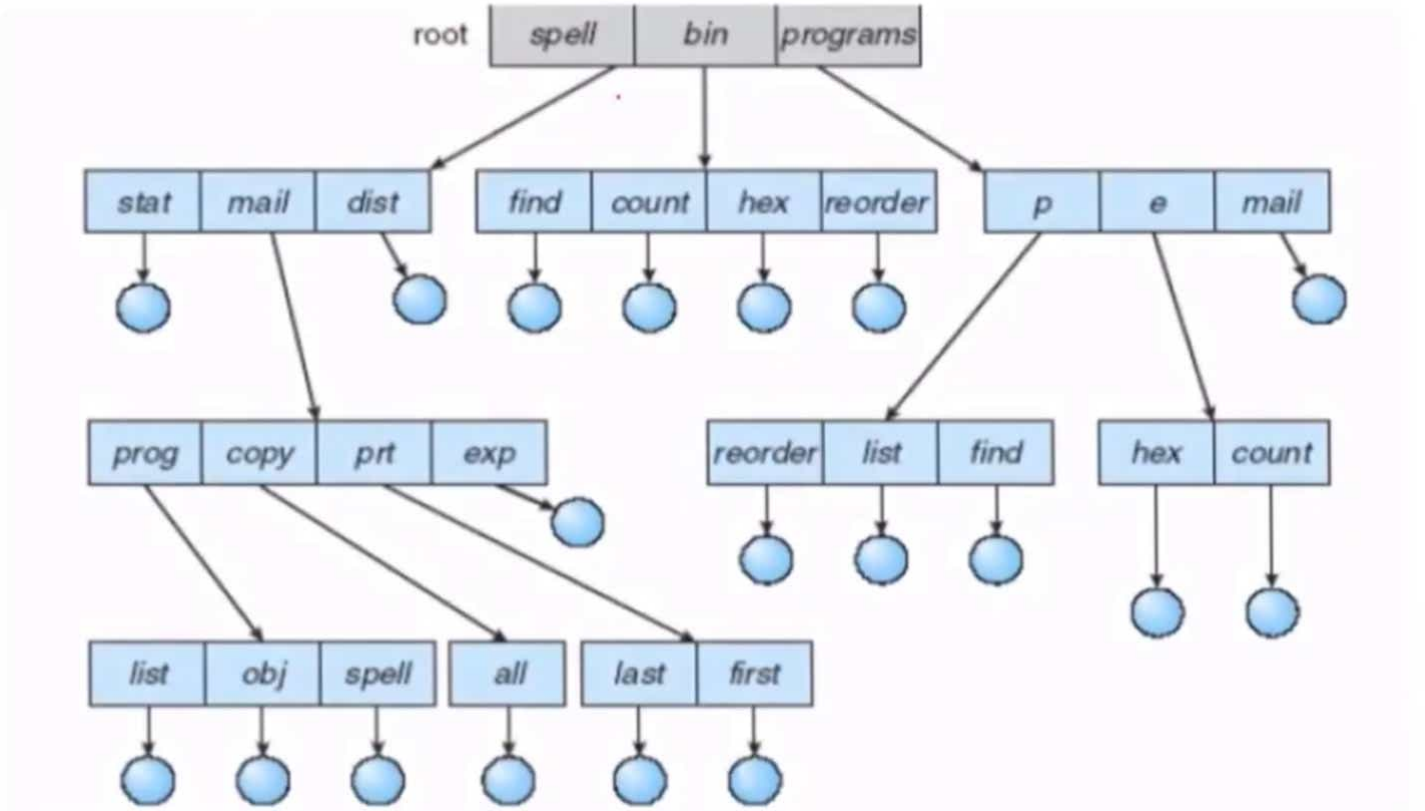
- Separate directory for each user (UFD = user file directory)
- Each entry in the MFD (master file directory) points to a UFD
- Advantage: No filename-collision among different users
- Disadvantage: Users are isolated from one another and can't cooperate on the same task
- System files (e.g. compilers, loaders, libraries...) are contained in a special user directory (e.g. user 0) that is searched if the OS doesn't find the file in the local UFD
- Search path = directory sequence searched when a file is named



Tree-Structured Directories

- Users can create their own subdirectories and organize files
- **Absolute path names**: begin at the root
- **Relative path names**: define a path from the current directory
- To delete an empty directory:
  - Just delete it
- To delete a non-empty directory:
  - First delete all files in the directory, or
  - Delete all that directory's files and subdirectories
- Efficient searching
- Grouping Capability

- Current directory (working directory)
  - cd /spell/mail/prog
  - type list



Storage media

Hard disk

CD

DVD

Magnetic Tape

Optical Disk Drive

File System

Organization of file and folder to make operation easy

- **File Management:**

  ⟫ File Management is a process of maintaining and organizing files in the computer.

  ⟫ Files are the organized by their extension in Window.

  ⟫ Window checks the file extension when the user opens a file.

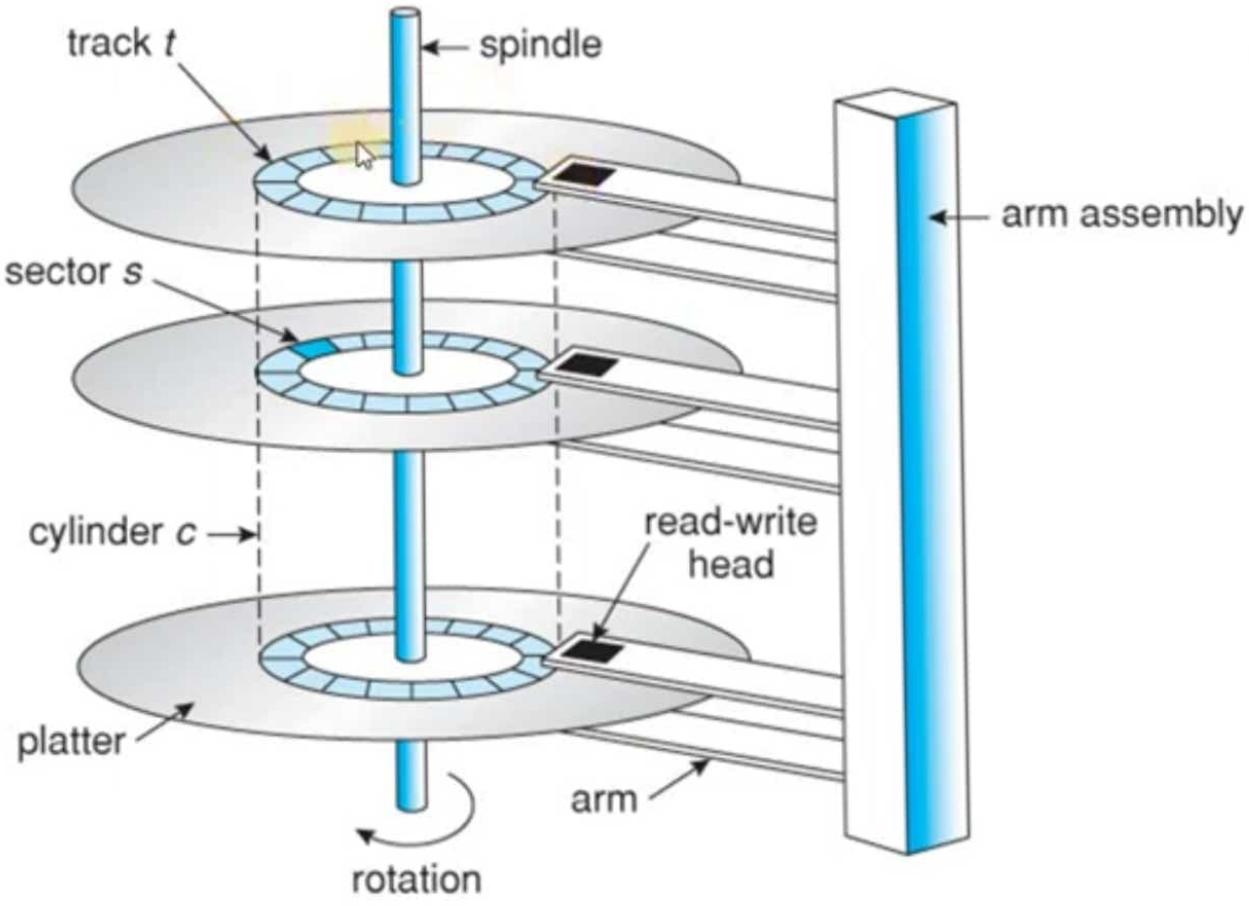
  ⟫ File extension is checked to determine what action should be taken.

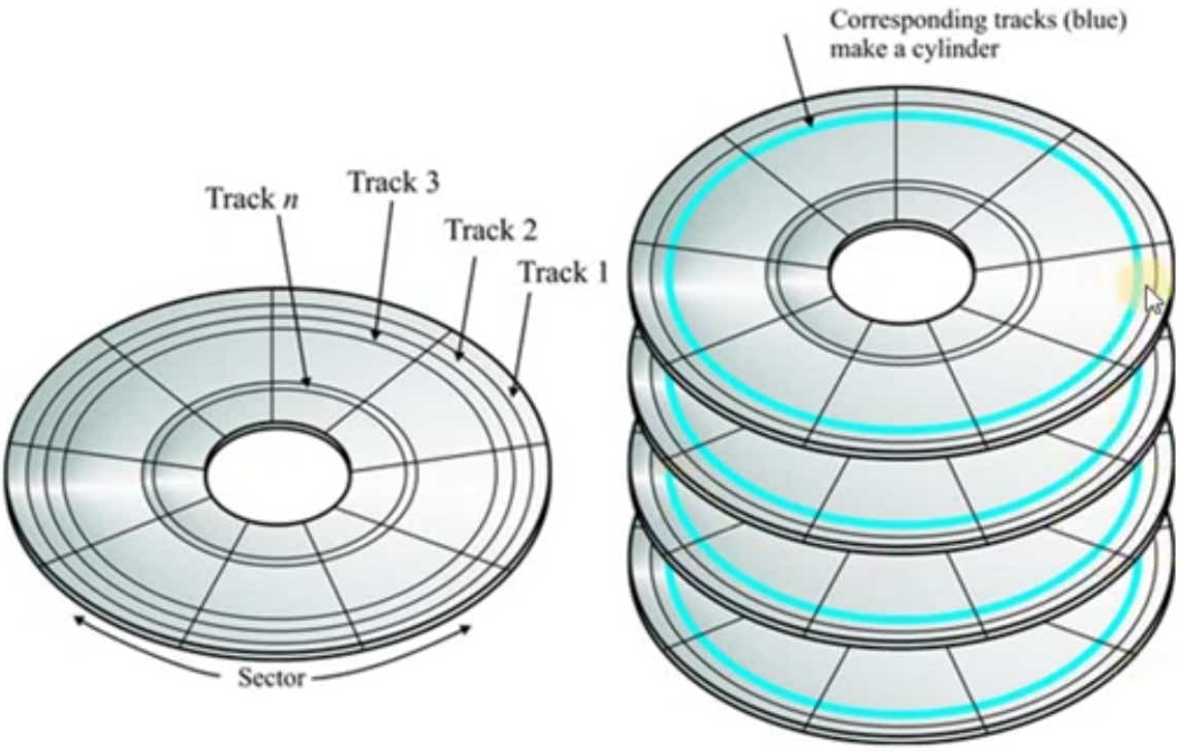
File Operations

- **Creating a file**:
  - First, space in the file system must be found for the file
  - Then, an entry for the file must be made in the directory
- **Writing a file**:
  - Make a system call specifying both the name of the file and the info 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**:
  - Use a system call that specifies the name of the file and where in memory the next block of the file should be put
  - Once the read has taken place, the read pointer is updated

- **Repositioning within a file**:
  - The directory is searched for the appropriate entry and the current-file-position is set to a given value
- **Deleting a file**:
  - Search the directory for the named file and release all file space and erase the directory entry
- **Truncating a file**:
  - The contents of a file are erased but its attributes stay
- Most of these file operations involve searching the directory for the entry associated with the named file
- To avoid this constant searching, many systems require that an 'open' system call be used before that file is first used
- The OS keeps a small table containing info about all open files

- When a file operation is requested, the file is specified via an index into the open-file table, so no searching is required
- When the file is no longer actively used, it is closed by the process and the OS removes its entry in the open-file table
- Some systems implicitly open a file when the first reference is made to it, and close it automatically when the program ends
- Most systems require that the programmer open a file explicitly with the 'open' system call before that file can be used
- A **per-process** table tracks all files that a process has open and includes access rights to the file & accounting info
- Each entry in the per-process table in turn points to a **system-wide** open-file table, which contains process- independent info, such as the file's disk location, access dates, and file size


track t
spindle
arm assembly
sector s
cylinder c
read-write
head
platter
arm
rotation


Corresponding tracks (blue)
make a cylinder
Track n
Track 3
Track 2
Track 1
Sector

Seek Time :

- It is the time that is taken by the head of a disc to move from one track to another track on a disk.
- Depends on:
  - Speed of read/write head
  - Distance between current and final position

Rotation Latency :

- The time required by the read/write head to rotate to the requested sector from the current position is called Rotational Latency.
- Depends on:
  - Rotational Speed of a disk, faster is better
  - Track and Sector Density, more is better
  - Amount to data to be Transferred

Transfer Time:

- Transfer time is the time taken to transfer the data from the disk.
- Depends on:
  - Rotational Speed of a disk, faster is better
  - Track and Sector Density, more is better
  - Amount to data to be Transferred

File Allocation Methods

- File is divided into logical “Blocks”
- Secondary memory (Disk) is divided into “Sectors”
- Allocation methods can be mainly into 2 types:
  - **Contiguous Allocation**
  - **Non-contiguous Allocation**
    - Linked List Allocation
    - Indexed Allocation

Purpose of File Allocation

- Efficient Disk Utilization
- Access should be faster

Contiguous File Allocation

- Storing data in the disk in a continuous fashion

- Files will be divided into blocks and the blocks will be written into the disk in a sequential manner - block1, block2, block3, etc...

- Advantages
  - Easy to implement
  - Excellent read performance - supports sequential as well direct access

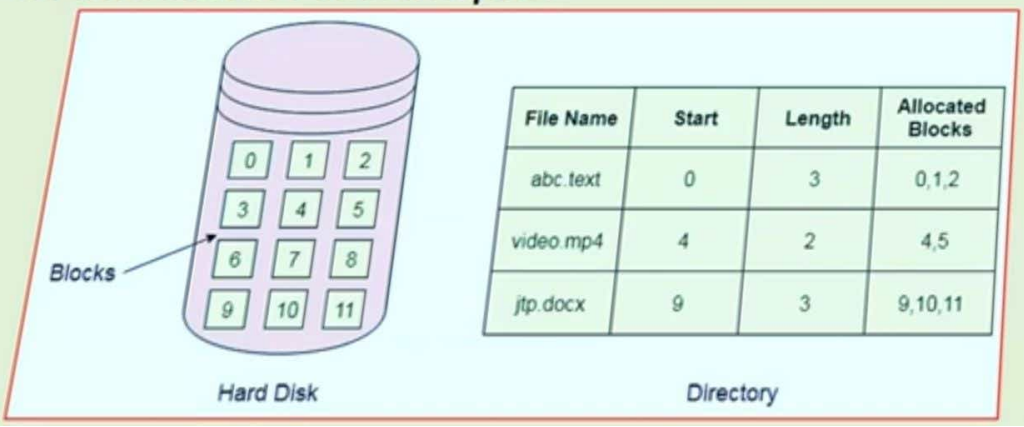
- Disadvantages
  - Disk will become fragmented
  - Difficult to grow the size of the file

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

Non-Contiguous Methods

- It is used in non-contiguous allocation
- In this each file is a linked list of disk blocks
- Disk blocks may be scattered any where on the disk
- The directory contains a pointer to the first and last block of the file
- Creating a file in chained allocation is easy
- There is no external fragmentation
- There is no need to declare size of the file when it is created
- Each file require space for pointer
- If pointer is lost then can't open file

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**.
- The free-space list records all *free* disk blocks—those not allocated to some file or directory.
- 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 disk space is added to the free-space list.

BIT VECTOR

- Here, the free-space list is implemented as a **bit map** or **bit vector**. Each block is represented by 1 bit. If the block is free, the bit is 1; if the block is allocated, the bit is 0.
- For example, consider a disk where blocks 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26, and 27 are free and the rest of the blocks are allocated. The free-space bit map would be

  001111001111110001100000011100000 ...

Advantage

- Simplicity
- Efficient

How to get to know the location of first free block

=(number of bits per word) × (number of 0-value words) + offset of first 1 bit.

LINKED LIST

- Other approach to free-space management is to link together all the free disk blocks, keeping a pointer to the first free block in a special location on the disk and caching it in memory.
- This first block contains a pointer to the next free disk block, and so on.
- This scheme is not efficient; to traverse the list, we must read each block, which requires substantial I/O time.
- But, traversing the free list is not a frequent action.

GROUPING AND COUNTING

GROUPING

- A modification of the free-list approach is to store the addresses of *n* free blocks in the first free block.
- The first *n*−1 of these blocks are actually free. The last block contains the addresses of another *n* free blocks, and so on.
- The addresses of a large number of free blocks can now be found quickly, unlike the situation when the standard linked-list approach is used.

COUNTING

- Generally, several contiguous blocks may be allocated or freed simultaneously, particularly when space is allocated with the contiguous-allocation algorithm or through clustering.
- 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.

Mount /Umount Filesystem

- **Command : Mount**

- All the devices are needed to mount(place) in location from where it can be access manually by specifying the filesystem type.
  Syntax : mount -t <typeofFilesystem> <SourceDevicePath> <destinationDir Path>

- **Command : umount**

- To unmount any Filesystem.
  Syntax : umount <source/destination path>