What is a File ?

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.



Attributes of the File

**1.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.

**2.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.**

**3.Type**

In a File System, the Files are classified in different types such as video files, audio files, text files, executable files, etc.

**4.Location**

In the File System, there are several locations on which, the files can be stored. Each file carries its location as its attribute.

**5.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.

**6.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.

**7.Time and Date**

Every file carries a time stamp which contains the time and date on which the file is last modified.

# Operations on the File

There are various operations which can be implemented on a file. We will see all of them in detail.

**1.Create**

Creation of the file is the most important operation on the file. Different types of files are created by different methods for example text editors are used to create a text file, word processors are used to create a word file and Image editors are used to create the image files.

**2.Write**

Writing the file is different from creating the file. The OS maintains a write pointer for every file which points to the position in the file from which, the data needs to be written.

**3.Read**

Every file is opened in three different modes : Read, Write and append. A Read pointer is maintained by the OS, pointing to the position up to which, the data has been read.

**4.Re-position**

Re-positioning is simply moving the file pointers forward or backward depending upon the user's requirement. It is also called as seeking.

**5.Delete**

Deleting the file will not only delete all the data stored inside the file, It also deletes all the attributes of the file. The space which is allocated to the file will now become available and can be allocated to the other files.

**6.Truncate**

Truncating is simply deleting the file except deleting attributes. The file is not completely deleted although the information stored inside the file get replaced.

## File Type

File type refers to the ability of the operating system to distinguish different types of file such as text files source files and binary files etc. Many operating systems support many types of files. Operating system like MS-DOS and UNIX have the following types of files −

### Ordinary files

* These are the files that contain user information.
* These may have text, databases or executable program.
* The user can apply various operations on such files like add, modify, delete or even remove the entire file.

### Directory files

* These files contain list of file names and other information related to these files.

### Special files

* These files are also known as device files.
* These files represent physical device like disks, terminals, printers, networks, tape drive etc.

These files are of two types −

* **Character special files** − data is handled character by character as in case of terminals or printers.
* **Block special files** − data is handled in blocks as in the case of disks and tapes.

## File Access Mechanisms

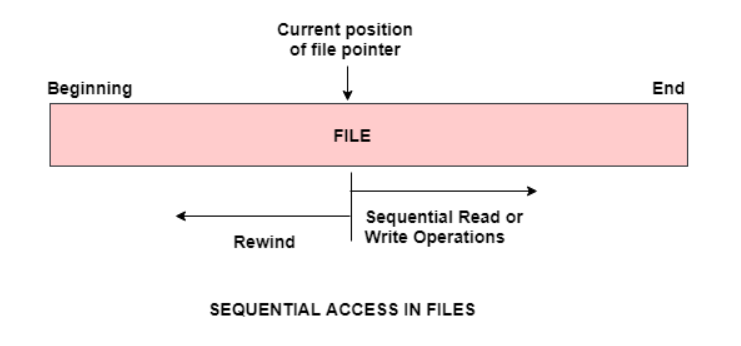
File access mechanism refers to the manner in which the records of a file may be accessed. There are several ways to access files −

* Sequential access
* Direct/Random access
* Indexed sequential access

### Sequential access

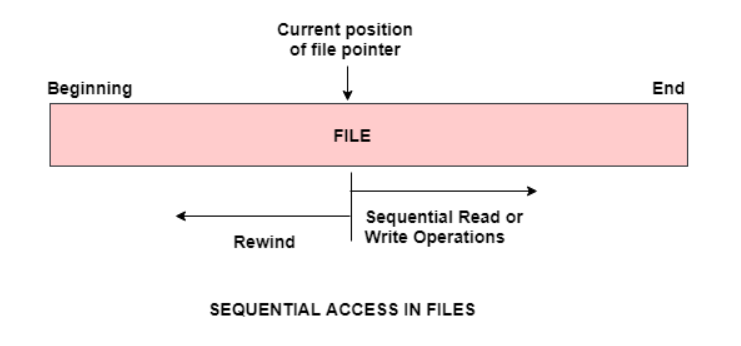
A sequential access is that in which the records are accessed in some sequence, i.e., the information in the file is processed in order, one record after the other. This access method is the most primitive one. Example: Compilers usually access files in this fashion.

A diagram to illustrate sequential access is as follows:



### Direct/Random access

* Random access file organization provides, accessing the records directly.
* Each record has its own address on the file with by the help of which it can be directly accessed for reading or writing.
* The records need not be in any sequence within the file and they need not be in adjacent locations on the storage medium.



### Indexed sequential access

* This mechanism is built up on base of sequential access.
* An index is created for each file which contains pointers to various blocks.
* Index is searched sequentially and its pointer is used to access the file directly.

## Space Allocation

Files are allocated disk spaces by operating system. Operating systems deploy following three main ways to allocate disk space to files.

* Contiguous Allocation
* Linked Allocation
* Indexed Allocation

### 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.

### Linked Allocation

* Each file carries a list of links to disk blocks.
* Directory contains link / pointer to first block of a file.
* No external fragmentation
* Effectively used in sequential access file.
* Inefficient in case of direct access file.

### Indexed Allocation

* Provides solutions to problems of contiguous and linked allocation.
* A index block is created having all pointers to files.
* Each file has its own index block which stores the addresses of disk space occupied by the file.
* Directory contains the addresses of index blocks of files.

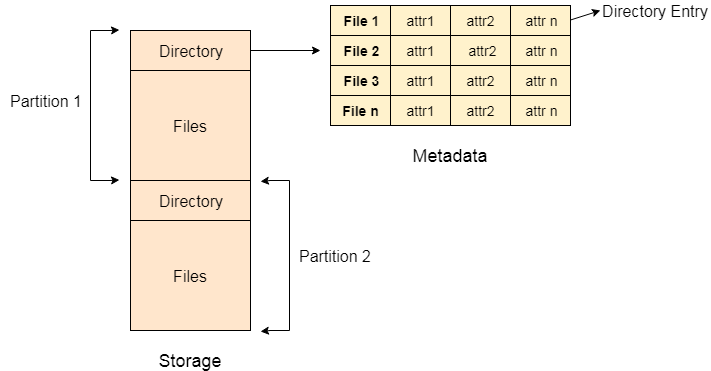
Directory Structure

What is a directory?

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.



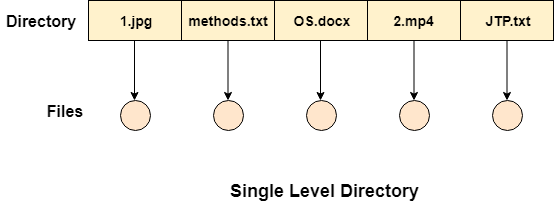
A directory can be viewed as a file which contains the Meta data of the bunch of files.

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

1. File Creation
2. Search for the file
3. File deletion
4. Renaming the file
5. Traversing Files
6. Listing of files

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. The directory contains one entry per each file present on the file system.



This type of directories can be used for a simple system.

Advantages

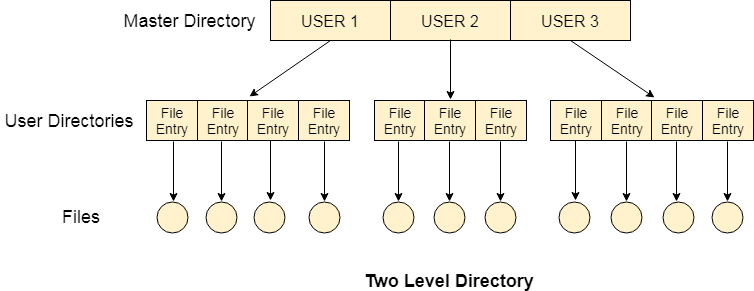
1. Implementation is very simple.
2. If the sizes of the files are very small then the searching becomes faster.
3. File creation, searching, deletion is very simple since we have only one directory.

Disadvantages

1. We cannot have two files with the same name.
2. The directory may be very big therefore searching for a file may take so much time.
3. Protection cannot be implemented for multiple users.
4. There are no ways to group same kind of files.
5. Choosing the unique name for every file is a bit complex and limits the number of files in the system because most of the Operating System limits the number of characters used to construct the file name.

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

1. Each files has a path name as ***/User-name/directory-name/***
2. Different users can have the same file name.
3. Searching becomes more efficient as only one user's list needs to be traversed.
4. 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.

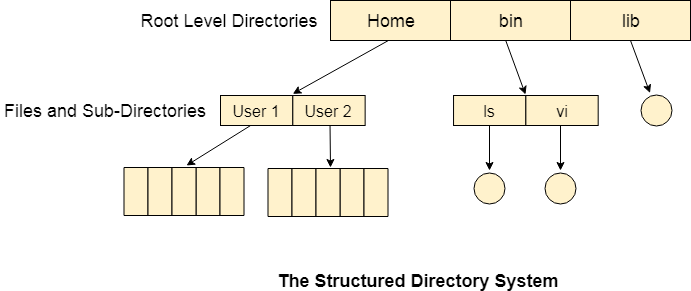
Tree Structured Directory

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

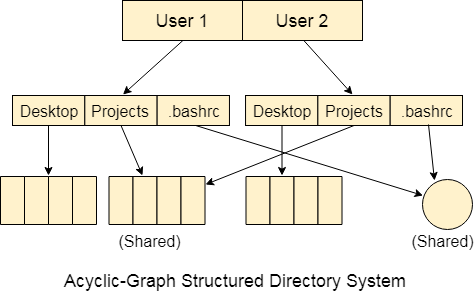
A tree structured directory system may consist of various levels therefore there is a set of permissions assigned to each file and directory.

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

1. In the case of soft link, the file just gets deleted and we are left with a dangling pointer.

2. In the case of hard link, the actual file will be deleted only if all the references to it gets deleted.

# File Systems

File system is the part of the operating system which is responsible for file management. It provides a mechanism to store the data and access to the file contents including data and programs. Some Operating systems treats everything as a file for example Ubuntu.

The File system takes care of the following issues

* **File Structure**

We have seen various data structures in which the file can be stored. The task of the file system is to maintain an optimal file structure.

* **Recovering Free space**

Whenever a file gets deleted from the hard disk, there is a free space created in the disk. There can be many such spaces which need to be recovered in order to reallocate them to other files.

* **disk space assignment to the files**

The major concern about the file is deciding where to store the files on the hard disk. There are various disks scheduling algorithm which will be covered later in this tutorial.

* **tracking data location**

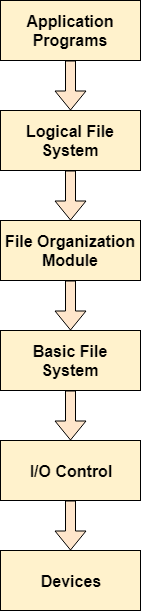
A File may or may not be stored within only one block. It can be stored in the non contiguous blocks on the disk. We need to keep track of all the blocks on which the part of the files reside.

File System Structure

File System provide efficient access to the disk by allowing data to be stored, located and retrieved in a convenient way. A file System must be able to store the file, locate the file and retrieve the file.

Most of the Operating Systems use layering approach for every task including file systems. Every layer of the file system is responsible for some activities.

The image shown below, elaborates how the file system is divided in different layers, and also the functionality of each layer.



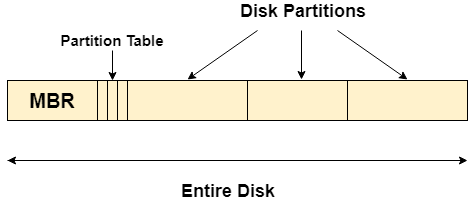
* When an application program asks for a file, the first request is directed to the logical file system. The logical file system contains the Meta data of the file and directory structure. If the application program doesn't have the required permissions of the file then this layer will throw an error. 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.

Master Boot Record (MBR)

Master boot record is the information present in the first sector of any hard disk. It contains the information regarding how and where the Operating system is located in the hard disk so that it can be booted in the RAM.

MBR is sometimes called master partition table because it includes a partition table which locates every partition in the hard disk.

Master boot record (MBR) also includes a program which reads the boot sector record of the partition that contains operating system.



What happens when you turn on your computer?

Due to the fact that the main memory is volatile, when we turn on our computer, CPU cannot access the main memory directly. However, there is a special program called as BIOS stored in ROM is accessed for the first time by the CPU.

BIOS contains the code, by executing which, the CPU access the very first partition of hard disk that is MBR. It contains a partition table for all the partitions of the hard disk.

Since, MBR contains the information about where the operating system is being stored and it also contains a program which can read the boot sector record of the partition, hence the CPU fetches all this information and load the operating system into the main memory.

On Disk Data Structures

There are various on disk data structures that are used to implement a file system. This structure may vary depending upon the operating system.

1. **Boot Control Block**

Boot Control Block contains all the information which is needed to boot an operating system from that volume. It is called boot block in UNIX file system. In NTFS, it is called the partition boot sector.

1. **Volume Control Block**

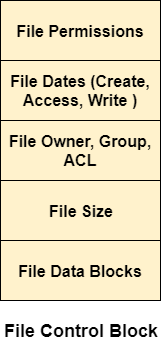
Volume control block all the information regarding that volume such as number of blocks, size of each block, partition table, pointers to free blocks and free FCB blocks. In UNIX file system, it is known as super block. In NTFS, this information is stored inside master file table.

1. **Directory Structure (per file system)**

A directory structure (per file system) contains file names and pointers to corresponding FCBs. In UNIX, it includes inode numbers associated to file names.

1. **File Control Block**

File Control block contains all the details about the file such as ownership details, permission details, file size,etc. In UFS, this detail is stored in inode. In NTFS, this information is stored inside master file table as a relational database structure. A typical file control block is shown in the image below.



In Memory Data Structure

Till now, we have discussed the data structures that are required to be present on the hard disk in order to implement file systems. Here, we will discuss the data structures required to be present in memory in order to implement the file system.

The in-memory data structures are used for file system management as well as performance improvement via caching. This information is loaded on the mount time and discarded on ejection.

1. **In-memory Mount Table**

In-memory mount table contains the list of all the devices which are being mounted to the system. Whenever the connection is maintained to a device, its entry will be done in the mount table.

1. **In-memory Directory structure cache**

This is the list of directory which is recently accessed by the CPU. The directories present in the list can also be accessed in the near future so it will be better to store them temporally in cache.

1. **System-wide open file table**

This is the list of all the open files in the system at a particular time. Whenever the user open any file for reading or writing, the entry will be made in this open file table.

1. **Per process Open file table**

It is the list of open files subjected to every process. Since there is already a list which is there for every open file in the system thereforeIt only contains Pointers to the appropriate entry in the system wide table.