# Operating System: File System Management

**Prepared By:**

**Dr. Sanjeev Gangwar**

**Assistant Professor,**

**Department of Computer Applications,**

**VBS Purvanchal University, Jaunpur**

# File System

**File Concept:** A file is a collection of similar records. The file is treated as a single entity by users and applications and may be referred by name. Files have unique file names and may be created and deleted. Restrictions on access control usually apply at the file level.

A file is a container for a collection of information. The file϶ manager provides a protection mechanism to allow users administrator how processes executing on behalf of different users can access the information in a file. File protection is a fundamental property of files because it allows different people to store their information on a shared computer.

File represents programs and data. Data files may be numeric, alphabetic, binary or alpha numeric. Files may be free form, such as text files. In general, file is sequence of bits, bytes, lines or records.

A common technique for implementing file types is to include the type as part of the file name. the name is split into two parts a name and an extension. Various file types are shown in the following table.

| File Type | Usual extension | Function |
|---|---|---|
| executable | exe, com, bin | read to run machine language program |
| object | obj, O | compiled, machine language, not linked |
| source Code | c, cc, java | source code in various languages |
| batch | bat, sh | commands to the command interpreter |
| text | txt, doc | textual data, documents |
| word processor | wp, tex, rtf, doc | various word-processor formats |
| library | lib, dll, mpeg, mov | libraries of routines for programmers |
| print or view | arc, zip, tar | ASCII or binary file in a format for printing or viewing |
| archieve | arc, zip, tar | related files grouped into one file, sometimes compressed for archiving or storage |
| multimedia | mpeg, mov, rm | binary file containing audio or A/V information |

**File Structure:** Four terms are commonly used for files.

- **Field:** A field is the basic element of data. An individual field contains a single value such as employee name, date.
- **Record:** A record is a collection of related fields that can be treated as single unit by application programs. A record will be variable length.

- **File:** A file is a collection of records, a file treat as a single entity by user and application programs.
- **Database:** A database is a collection of related data which consist of relationship the elements of the data

For example, a telephone book is analogous to a file. It contains a list of records, each of which consists of three fields: name, address, and telephone number.

**File Attributes:** A file has certain attributes which vary from one operating system to another.

- **Name:** Every file has a name by which it is referred.
- **Identifier:** It is unique number that identifies the file within the file system.
- **Type:** This information is needed for those systems that support different types of files.
- **Location:** It is a pointer to a device & to the location of the file on that device
- **Size:** It is the current size of a file in bytes, words or blocks.
- **Protection:** It is the access control information that determines who can read, write & execute a file.
- **Time, date & user identification:** It gives information about time of creation or last modification & last use.

**File Operations:** Any file system provides not only a means to store data organized as files, but a collection of functions that can be performed on files. Typical operations include the following:

- **Creating files:** Two steps are necessary to create a file. First, space must be found for the file in the file system. Secondly, an entry must be made in the directory for the new file.
- **Reading a file:** Data & read from the file at the current position. The system must keep a read pointer to know the location in the file from where the next read is to take place. Once the read has been taken place, the read pointer is updated.
- **Writing a file:** Data are written to the file at the current position. The system must keep a write pointer to know the location in the file where the next write is to take place. The write pointer must be updated whenever a write occurs.
- **Repositioning within a file:** The directory is searched for the appropriate entry & the current file position is set to a given value. After repositioning data can be read from or written into that position. Repositioning within a file does not need to involve any actual I/O. this file operation is also known as a file seek.
- **Deleting a file:** To delete a file, we search the directory for the required file. After deletion, the space is released so that it can be reused by other files.
- **Truncating a file:** the user may want to erase the contents of a file but keep its attribute. Rather than forcing the user to delete the file and then recreate it, this function allows all attributes to remain unchanged- except for file length- but lets the file be reset to length zero and its file space released.

**File Structure:** File types also can be used to indicate the internal structure of the file. The operating system requires that an executable file have a specific structure so that it can determine where in memory to load the file and what the location of the first instruction is. If

OS supports multiple file structures; the resulting size of OS is large. If the OS defines 5 different file structures, it needs to contain the code to support these file structures. All OS must support at least one structure that of an executable file so that the system is able to load and run programs.

**Internal File Structure:** In UNIX OS, defines all files to be simply stream of bytes. Each byte is individually addressable by its offset from the beginning or end of the file. In this case, the logical record size is 1 byte. The file system automatically packs and unpacks bytes into physical disk blocks, say 512 bytes per block.

The logical record size, physical block size, packing determines how many logical records are in each physical block. The packing can be done by the user's application program or OS. A file may be considered a sequence of blocks. If each block were 512 bytes, a file of 1949 bytes would be allocated 4 blocks (2048 bytes). The last 99 bytes would be wasted. It is called internal fragmentation all file systems suffer from internal fragmentation, the larger the block size, the greater the internal fragmentation.

**File Access Methods:** Files store information. When it is used, this information must be accessed and read into computer memory. The information in the file can be accessed in several ways.

**Sequential Access:** It is the simplest access method. Information in the file is processed in order i.e. one record after another. A process can read all the data in a file in order starting from beginning but can't skip & read arbitrarily from any location. Sequential files can be rewound. It is convenient when storage medium was magnetic tape rather than disk.

Eg : A file consisting of 100 records, the current position of read/write head is 45th record, suppose we want to read the 75th record then, it access sequentially from 45, 46, 47 …….. 74, 75. So the read/write head traverse all the records between 45 to 75.

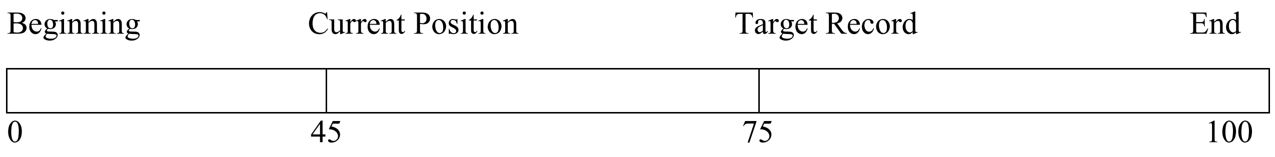

Fig 1: sequential access file

**Direct Access:** A file is made up of fixed length-logical records that allow programs to read & write records rapidly in no particular order. This method can be used when disk are used for storing files. This method is used in many applications e.g. database systems. If an airline customer wants to reserve a seat on a particular flight, the reservation program must be able to access the record for that flight directly without reading the records before it. In a direct access file, there is no restriction in the order of reading or writing. For example, we can read block 14, then read block 50 & then write block 7 etc. Direct access files are very useful for immediate access to large amount of information.

**Indexed Access:** In this method an index is created which contains a key field and pointers to the various blocks. To find an entry in the file for a key value, we first search the index and then use the pointer to directly access a file and find the desired entry.

With large files, the index file itself may become too large to be keep 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. Figure 2 shows a situation as implemented by VMS (Virtual Memory Storage) index and relative files.

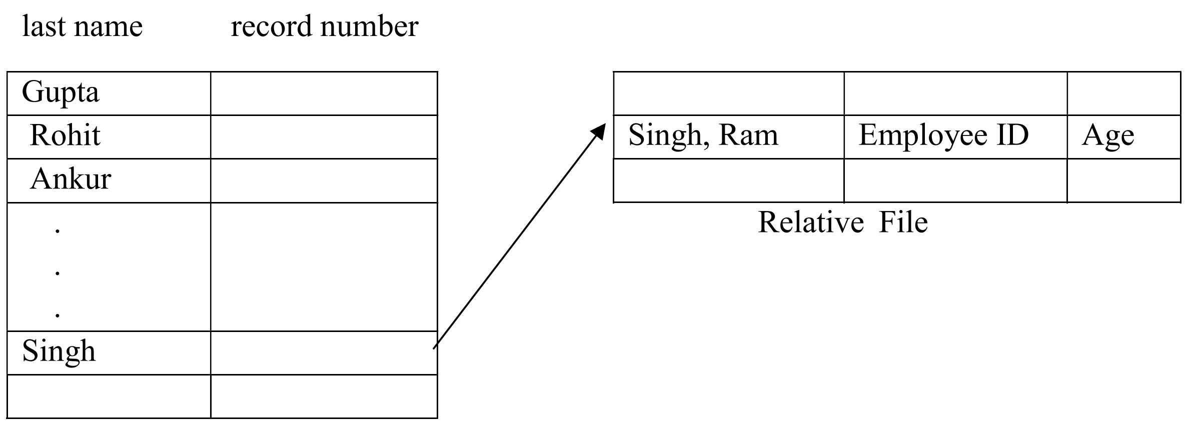

Fig 2: Example of index and relative files

**File Directories:** Sometimes the file system consisting of millions of files, at that situation it is very hard to manage the files. To manage these files grouped these files and load one group into one partition. Each partition is called a directory. A directory structure provides a mechanism for organizing many files in the file system.

**Operation on the directories**: the various operations that can be performed on directory are:

- Searching for a file: we need to search a directory structure to find the entry for a particular file. Since files have symbolic names in a user-readable form and similar names may indicate a relationship between files. We may want to able to find all files whose names match a particular pattern.
- Create a file: new files are created and added to the directory.
- Delete a file: when we do not require to use a particular file, it is removed from the directory.

- List a directory: when we want to list all the files in a particular directory, and the contents of directory entry for each file in the list.
- Rename a file: Whenever we need to change the name of the file, we can change the name.
- Traverse the file system: in a directory structure, we may wish to access every directory, and every file within a directory structure.

**Directory Structure:** The most common schemes for defining the structure of the directory are:

(i) **Single-level Directory:** it is the simplest directory structure. All files are contained in the same directory which is easy to support and understand (figure 3)

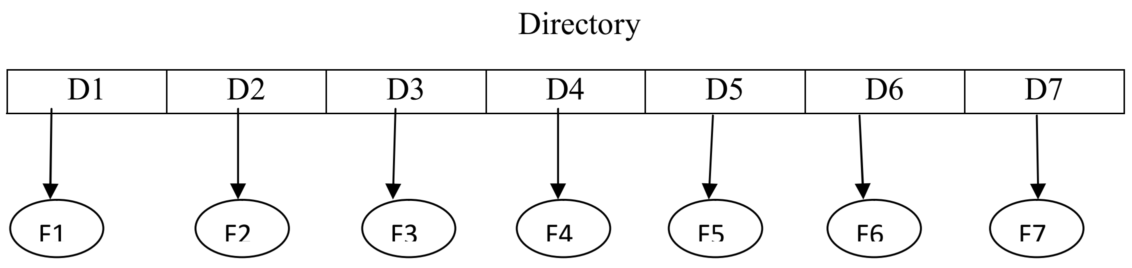

Fig 3: Single-level directory structure

### Advantages:

- Since it is a single directory, so its implementation is very easy.
- If the files are smaller in size, searching will become faster.
- The operations like file creation, searching, deletion, updating is very easy in such a directory structure.

### Disadvantages:

- There may chance of name collision because two files can not have the same name.
- Searching will become time taking if the directory is large.
- The same type of files cannot be grouped together.

(ii) **Two-level Directory:** the disadvantage of single level directory is the confusion of files names between different users. The solution for this problem is to create a directory for each user as shown in figure 4.

In the two-level directory structure, each user has his own user files directory (UFD). Each user has similar structure but lists only the files of a single user. When user login, the system's master file directory (MFD) is searched. The master file directory is indexed by user name or account number and each entry points to the user directory for that user.

When users refer to a particular file, only their own user file directory is searched. Thus different users may have files with the same name, as long as all file names within each user file directory are unique.

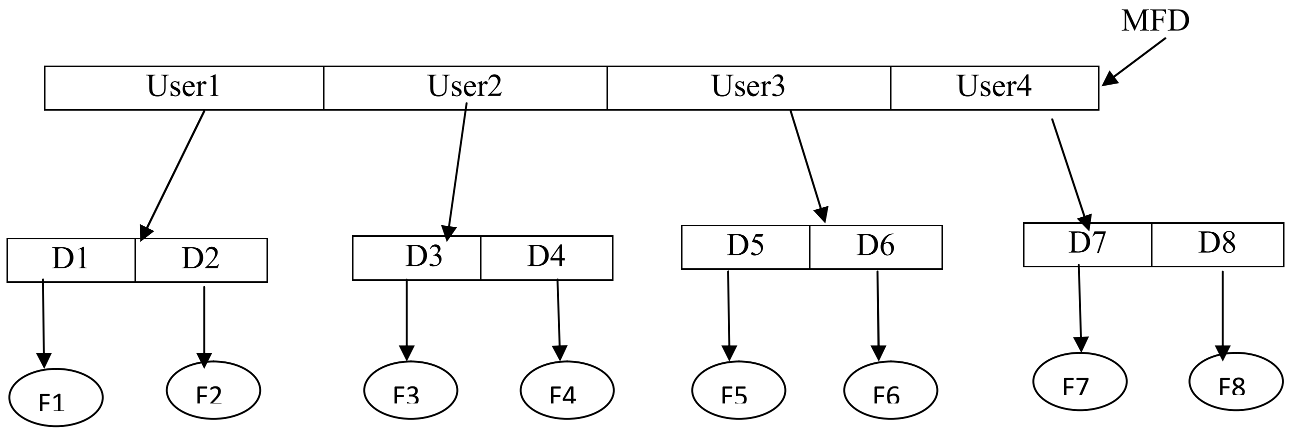

Fig 4: Two- level directory structure

**Advantages:**

- We can give full path like /User-name/directory-name.
- Different users can have same directory as well as file name.
- Searching of files become more easy due to path name and user-grouping.

**Disadvantages:**

- A user is not allowed to share files with other users.
- Searching will become time taking if the directory is large.
- Two files of the same type cannot be grouped together in the same user.

(iii) **Tree- Structured Directory:** the tree-structure allows user to create their own sub-directories and organize their files accordingly. The tree has a root directory. Every file in the systems has a unique path name. A path is the path from the root through all the sub-directories to a specified file. A directory contains a set of files and or sub-directories.

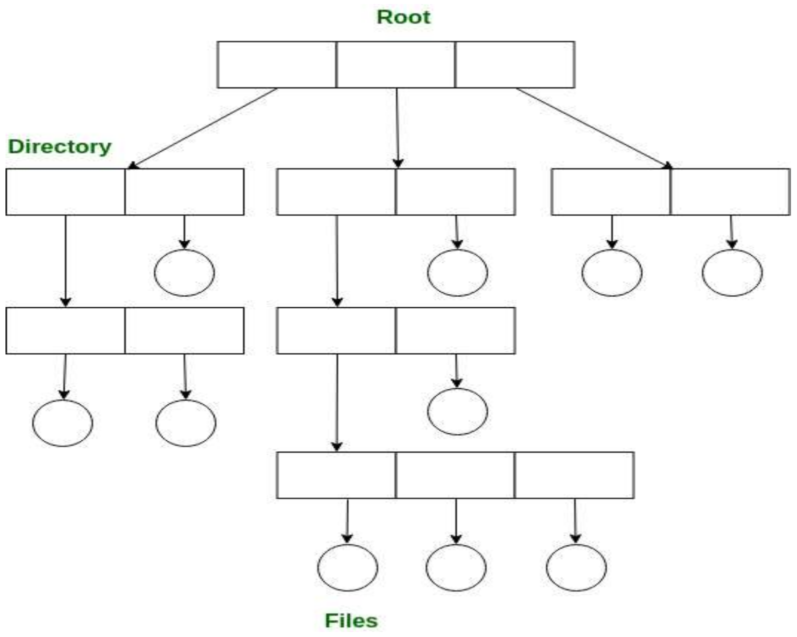

Fig 5: Tree-structured Directory

**Advantages:**

- User can access other user's files by specifying the path name.
- User can create his own sub-directories.
- Searching becomes very easy; we can use both absolute path as well as relative.

**Disadvantages:**

- Every file does not fit into the hierarchical model; files may be saved into multiple directories.
- We cannot share files.
- It is inefficient, because accessing a file may go under multiple directories.

(iv) **Acyclic Graph Directories:** A shared directory or file will exist in the file system in two or more places at once. A shared directory or file is not the same as two copies of the file. With a shared file there is only one actual field and any changes made by one person would be immediately visible to the other.

An acyclic graph allows directories to have shared sub-directories and files. The same file or sub-directory may be in two or more process exists in the file system at a time. An acyclic graph directory structure is more flexible than a simple structure but it is also more complex.

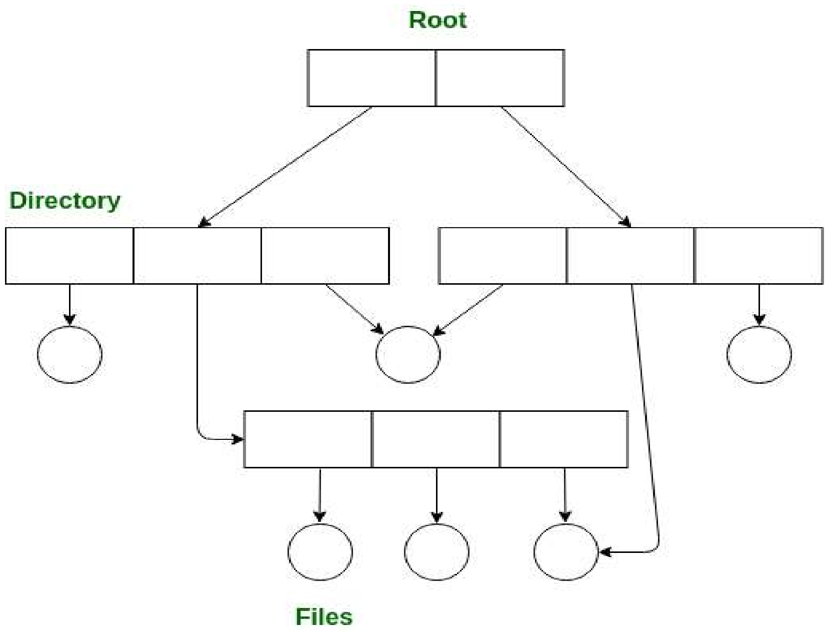

Fig 6: Acyclic graph structured directory

## Advantages:

- We can share files.
- Searching is easy due to different-different paths.
- Allow multiple directories to contain same file.

## Disadvantages:

- We share the files via linking; in case of deleting it may create the problem.
- Need to be cautions of dangling pointers when files are deleted.

**……………………………THE END…………………………………………….**

## References:

(1) Abraham Silberschatz, Galvin & Gagne, Operating System Concepts, John Wiley & Sons, INC.

(2) Harvay M.Deital, Introduction to Operating System, Addition Wesley Publication Company.

(3) Vijay Shukla, Operating System, S.K. Kataria & Sons.

(4) Naresh Chauhan, Principles of Operating System, Oxford University Press