Unit 2: Introduction to File System & File Management
2.1 File Concept
2.2 Operations on File
2.3 File Access Methods (Sequential Access & Direct Access)
2.4 Directory Systems File Management Functions
2.5 File System(2.1) & Directory Structure Organization
2.6 File Protection

2.1 File Concept

What is a File? Which type of information is stored in the file?

File

- A file is a named collection of related information recorded on secondary storage, usually as a sequence of bytes, with two views.
  - Logical (programmer) view, as the users see it (how they are used and what properties they have.)
  - Physical (os) view, as it resides on secondary storage.
- Data files may be numeric, alphanumeric, or binary. Files may be free from, such as text files, or maybe formatted rigidly.
- In general, a file is a sequence of bits, bytes, lines, or records the meaning of which is defined by the file's creator and user.
- Many different types of information may be stored in a file- source programs, object programs, executable programs, numeric data, text, payroll records, graphic images, sound recordings, and so on.
- A file has a defined structure, depending on its type.
  - A text file is a sequence of characters organized into lines (and possibly pages).
  - A source file is a sequence of subroutines and functions, each of which is further organized as declarations followed by executable statements.
  - An object file is a sequence of bytes organized into blocks understandable by the system's linker.
  - An executable file is a series of code sections that the loader can bring into memory and execute.

Record

- A record is a collection of related fields treated as a unit.
- A file consists of a sequence of records.
- In a text file a record corresponds to a line of text, in other cases, a record has no physical basis, it is just a convenient collection of values chosen to suit the application.
- For example, employee records, and student records.

File system

- A part of an operating system, that deals with files, is known as a file system.
- A file system is a collection of files and also a collection of directory structures.

- The **file system** consists of two distinct parts: **a collection of files**, each storing related data, and a **directory structure**, which organizes and provides information about all the files in the system.
- It is the responsibility of a file system to manage all the files.
- It provides facilities to create & delete files, read & write contents of files, etc.
- It specifies conventions for naming files. These conventions include the maximum number of characters in a name which characters can be used and, in some systems, how long the file name suffix can be.
- A file system also includes a format for specifying the path to a file through the structure of directories.

File Naming

- Files store information on disks, providing convenience for users by naming them when created and later referring to them, eliminating concerns about disk location and functionality.
- Current operating systems allow strings of one to eight letters, including digits and special characters, as legal file names, with many supporting names up to 255 characters.
- File systems like UNIX and MS-DOS differentiate between case sensitive file names, with UNIX having three distinct test1, Test1, and TEST1 files, while MS-DOS has the same file.
- Operating systems often use two-part file names separated by a period (.), with the extension following it indicating the file's content.
- MS-DOS file names range from 1-8 characters with optional extension of 1-3 characters, while UNIX file sizes depend on user and may include multiple extensions.
- Some of the more common file extensions are given in following figure.

Extension	Meaning
.txt	General text file
.c	C source program
.bak	Backup file
.obj	Object file (Compiler output, but yet not linked)
.exe	Executable file
.gif	Graphical Interchange Format file
.hlp	Help file
.html	World Wide Web Hyper Text Markup Language file
.jpg	Still picture encoded with JPEG standard
.mp3	Music encoded in MPEG layer 3 audio format
.mpg	Movie encoded with the MPEG standard
.pdf	Portable Document Format file
.ps	PostScript file
.zip	Compressed archive file

File attribute

- When a file is named, it becomes independent of the process, the user, and even the system that created it. For instance, one user might create the file, and another user might edit that file by specifying its name.
- A file's attributes vary from one OS to another but typically consist of these:
    - **Name:** A file is named for the convenience of the user and is referred to by its name. The symbolic file name is the only information kept in human-readable form.
    - **Identifier:** This unique tag, usually a number, identifies the file within the file system; it is the non–human-readable name for the file.
    - **Type:** Type information is needed for those systems that support different types. The types depend on the extensions of the file. For eg, .exe for the executable, and .obj for the object file.
    - **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 (in bytes, words, or blocks), and possibly the maximum allowed size are included in this attribute.
    - **Protection:** Access-control information determines who can do reading, writing, executing, and so on.
    - **Usage Count:** This value indicates the number of processes that are currently using this file.
    - **Time, date, and user identification:** This information may be kept for creation, last modification, and last use. These data can be useful for protection, security, and usage monitoring.
- The information about all files is kept in the directory structure, which also resides on secondary storage; typically, a directory entry consists of the file's name and its unique identifier.

File types

- The types of files recognized by the system are **regular**, **directory**, or **special**. However, the operating system uses many variations of these basic types.

Regular files

- Regular files are the most common files and are used to contain data. Regular files are in the form of text files or binary files:

✔ **Text files**

- Text files are regular files that contain information stored in ASCII format text and are readable by the user. You can display and print these files.
- The lines of a text file must not contain NUL characters, and none can exceed bytes in length, including the newline character.

✔ **Binary files**

- Binary files are regular files that contain information readable by the computer.
- Binary files might be executable files that instruct the system to accomplish a job. Commands and programs are stored in executable, binary files.

- Special compiling programs translate ASCII text into binary code.

Directory files

- A directory file contains information that the system needs to access all types of files, but directory files do not contain the actual file data.
- As a result, directories occupy less space than a regular file and give the file system structure flexibility and depth.
- Each directory entry represents either a file or a subdirectory.

Special files

- Special files define devices for the system or are temporary files created by processes.
- The basic types of special files are FIFO (first-in, first-out), block, and character
- Windows (and some other systems) use special file extensions to indicate the type of each file**:**

File type	**Usual extension**	**Function**
Executable	Exe, com, bin or none	Ready-to-machine-language program
Object	Obj, o	Compiled, machine language not linked
Source code	C, cc, java, pas, asm, a	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, a, so, dll	Libraries of routines for programmers
Print or view	Ps, pdf, tar	ASCII or binary file in a format for printing or viewing
Archive	Arc, zip, tar	Related files are grouped into one file, sometimes compressed, for archiving or storage

Multimedia	Mpeg, mov, rmm, mp3, avi	Binary file containing audio or A/V information

2.2 Operations on File

- A file is an abstract data type. To define a file properly, we need to consider the operations that can be performed on files. The basic file operations are given below.
- The OS can provide system calls to create, write, read, reposition, delete, and truncate files.

Creating file:

A new file can be created by a system call embedded in a program or by an OS command issued by an interactive user.

Two steps are necessary to create a file.

1. Space in the file system must be found for the file.
2. An entry for the new file must be made in the directory.

Writing a file:

To write a file, we make 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. The write pointer must be updated whenever a write occurs.

Reading a file:

To read from a file, we use a system call that specifies the file name and where (in memory) 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.

Because a process usually reads from or writes to a file, the current operation location can be kept as a pre-process **current-file-position** pointer.

Both the read and write operations use this same pointer, saving space and reducing system complexity.

Repositioning within a file

The directory searches for the appropriate entry, and the current-file-position pointer is repositioned to a given value. Repositioning within a file need not 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 named file.

Having found the associated directory entry, we release all file space, so that it can be reused by other files, and erase the directory entry.

Truncating a file

The user may want to erase the contents of a file but keep its attributes.

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.

Open a file

Before using a file, it must be opened. File attributes and data contents are fetched in main memory for rapid access on later calls.

Memory space in the main memory is allocated to store fetched information.

Close a file

When the use of the file is finished, it should be closed to free up main memory space.

File attributes and data contents are stored back on disk. This may contain modified information if the file is updated.

Append

This is a restricted form of write operation. Here, data are only added to the end of a file.

Get Attributes

This operation is used to retrieve file attributes.

Set Attributes

This operation is used to write file attributes. Generally, it is used to change file attributes such as file protection information.

Rename

This operation is used to change the name of an existing file. This is not strictly necessary because a file can be copied to a new file with a new name and then the old file can be deleted.

2.3 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.

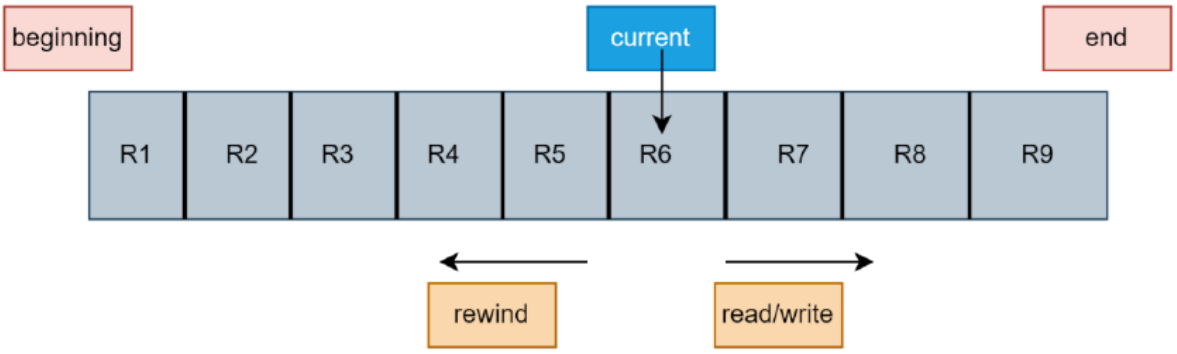
we will learn about the three types of file access methods. They are:

1. Sequential access
2. Direct access
3. Indexed sequential access

1. **Sequential access**

- It is one of the simplest access methods and is widely used by editors and compilers.
- Most of the operating systems access the file sequentially.
- You must have seen the audio cassettes. Even they use the same access method.

- The files are a collection of records. Accessing the file is equivalent to accessing the records. In the sequential access method, each record is accessed sequentially, one after the other. Consider the below image for more clarity.



The figure represents a file. The current pointer is pointing to the record currently being accessed. In the sequential access method, the current pointer cannot directly jump to any record. It has to "cross" every record that comes in its path. Suppose there are nine records in the file from R1 to R9. The current pointer is at record R6. If we want to access record R8, we have to first access record R6 and record R7. This is one of the major disadvantages of the sequential access method.

Advantages of Sequential Access:

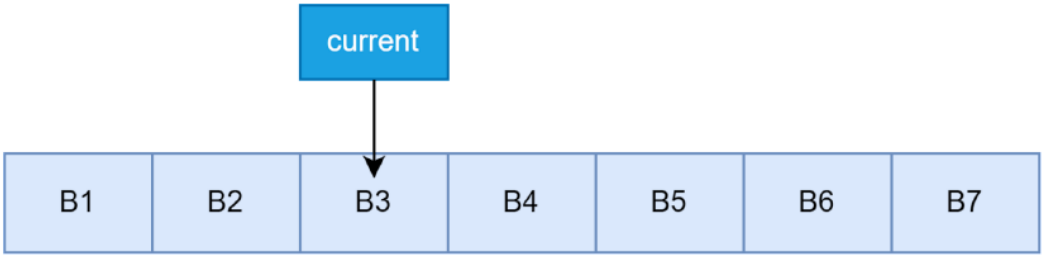
- The sequential access mechanism is very easy to implement.
- It uses lexicographic order to enable quick access to the next entry.

Disadvantages of Sequential Access:

- Sequential access will become slow if the next file record to be retrieved is not present next to the currently pointed record.
- Adding a new record may need relocating a significant number of records of the file.

2. Direct/Random/Relative Access

- Direct-access files are of great use for immediate access to large amounts of information like database systems.
- 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 need is stored in the 10th block. In that case, sequential access will not be implemented because it will cross all the blocks to access the needed record.
- The direct-access method requires file operations to include the block number as a parameter, which is typically a relative block number.
- This allows the OS to decide where the file should be placed and prevents users from accessing unrelated parts of the file system.



Advantages of Direct/Relative Access:

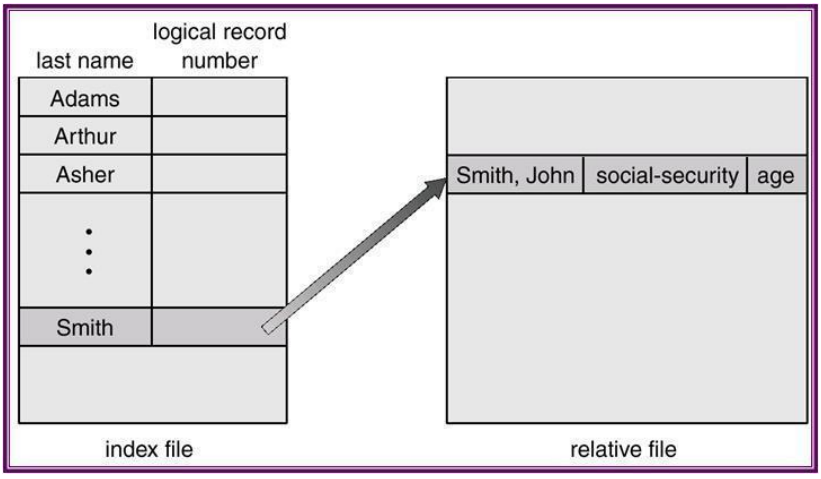
- The files can be retrieved right away with a direct access mechanism(structure), reducing the average access time of a file.
- There is no need to traverse all of the blocks that come before the required block to access the record.

Disadvantages of Direct/Relative Access:

- The direct access mechanism is typically difficult to implement due to its complexity.
- Organizations can face security issues as a result of direct access as the users may access/modify the sensitive information. As a result, additional security processes must be put in place.

3. Indexed Sequential Access

- These methods generally involve the construction of an index for the file.
    - To find a record in the file, we first search the index to learn exactly which block contains the desired record.
    - And then use the pointer to access the file directly and to find the desired record.
- This s tructure allows us to search a large file doing little I/O.
- 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.



- To find a particular item, we first make a binary search of the master index, which provides the block number of the secondary index.
- The secondary index blocks point to the actual file blocks.
- This block is read in, and again a binary search is used to find the block containing the desired record. Finally, this block is searched sequentially.
- In this way any record can be located from its key by at most direct-access reads.

Advantages of Indexed Sequential Access:

- If the index table is appropriately arranged, it accesses the records very quickly.
- Records can be added at any position in the file quickly.

Disadvantages of Indexed Sequential Access:

- When compared to other file access methods, it is costly and less efficient.
- It needs additional storage space.

2.4**Directory systems File Management Functions**

Directory systems in file management provide a structure for organizing, storing, and retrieving files on a storage device. Key functions of directory systems include:

1. **File Organization and Storage:**

- Directories group related files and subdirectories, creating an organized hierarchy that enables users to quickly locate and manage files.

2. **File Naming and Identification:**

- Assigns unique names to files within each directory, preventing conflicts and making files easily identifiable.

3. **Path Management:**

- Supports path structures (absolute and relative paths) that allow users and programs to navigate the file system and access files across various directories.

4. **Access Control and Security:**

    - **Enables the setting of permissions on files and directories, allowing controlled access for different users and preventing unauthorized modifications.**

5. **File Retrieval and Search:**

- Provides tools to search and retrieve files based on names, extensions, or other attributes, improving accessibility.

6. **File Metadata Storage:**

- Maintains metadata (such as file size, type, creation/modification dates) to manage file information, aiding in file sorting and tracking.

7. **Space Management and Quotas:**

- Allocates and tracks storage space for directories and files, and can enforce storage limits to prevent individual users from overusing space.

8. **Directory Operations:**

- Supports basic directory operations, such as creating, renaming, moving, and deleting directories, allowing flexibility in file organization.

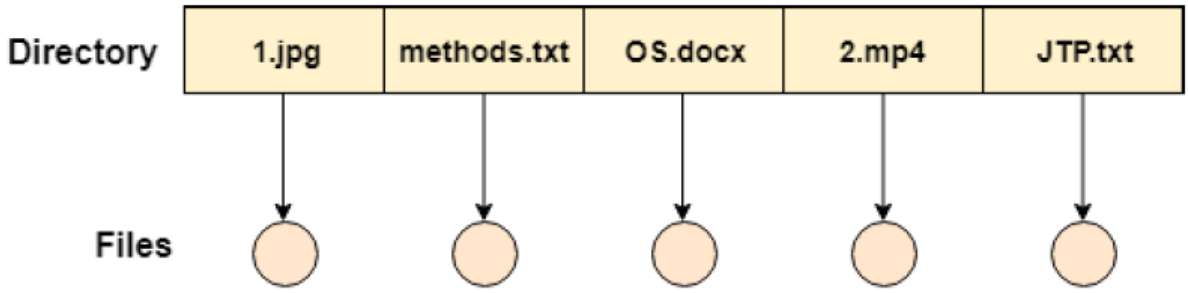
These functions enable efficient file handling, organization, and control, contributing to a structured and secure file system.

2.5 Directory Structure Organization

- A directory is a container that is used to contain related folders and files on the disk.
- It organizes files and folders into a hierarchical manner.
- The directory contains information about the files, including attributes, location and ownership. Much of this information especially that is concerned with storage is managed by the operating system.
- The directory is itself a file, accessible by various file management routines.
- 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:
    - File Creation
    - To insert entries
    - Search for the file
    - File deletion
    - Renaming the file
    - Traversing Files
    - Listing of files

Single Level Directory

- The simplest directory structure is the single – level directory or root directory.
- All files are contained in the same directory, which is easy to support and understand.
- The users are not allowed to create subdirectories under the root directory.
- single – level directory has significant limitations, when the number of files increases or when the system has more than one user.



Single Level Directory

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. It is not suitable for multi-user systems. Different users may provide same file name for different files, possibly overwriting other's files. This is called a file name collision.
2. It is even not suitable for single user system when number of file becomes too large.
3. The directory may be very big therefore searching for a file may take so much time.
4. Different files can't be grouped.

Two Level Directory

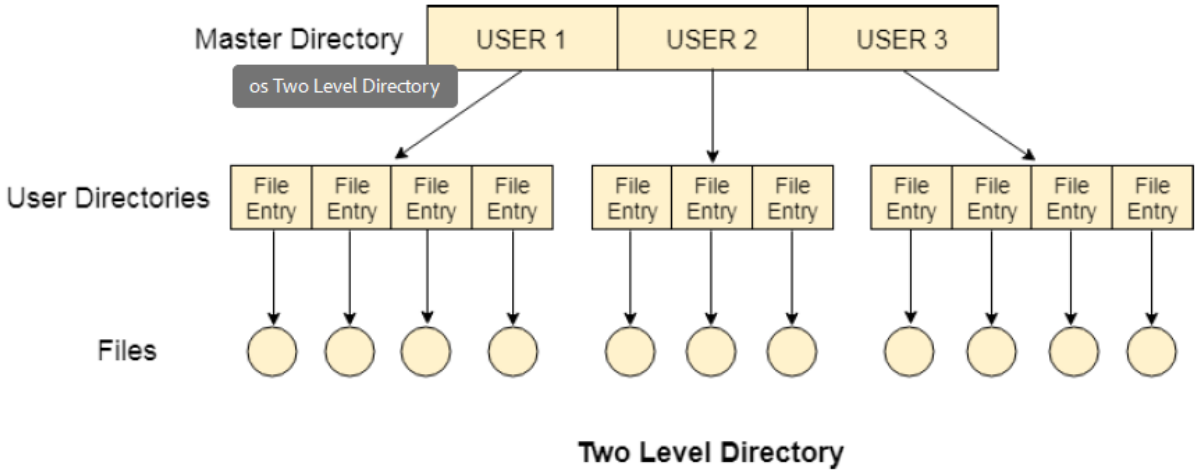
There is one master directory which contains separate directories dedicated to each user. For each user, there is a different directory present at the second level, containing group of user's file. The system doesn't let a user to enter in the other user's directory without permission.

The standard solution to limitations of single – level directory is to create a separate directory for each user knows as master directory.

 In Two level directory structure, the users create directory directly inside the root directory.

 But once a user creates such directory, further he cannot create any subdirectory inside that directory. The system doesn't let a user to enter in the other user's directory without permission.

 In a two-level directory structure, each user has their own UFD, which forms a tree with the MFD as the root, UFDs as its direct descendants, and files as leaves.



Two Level Directory

Advantages

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

Disadvantages

1. It is not suitable for users having large number of files.
2. Different files cannot be grouped.

Tree Level Directory Structure

- A tree is the most common directory structure. The tree has a root directory, and every file in the system has a unique path name.
- 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.
- In a tree directory structure, except root directory, every directory or file has only one parent directory.
- This generalization allows users to create their own subdirectories and to organize their files accordingly.
- 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 paths, either relative or absolute.**
    - An absolute path name begins at the root and follows a path down to the specified file, giving the directory names on the path.
    - A relative path name defines a path from 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.

Advantages

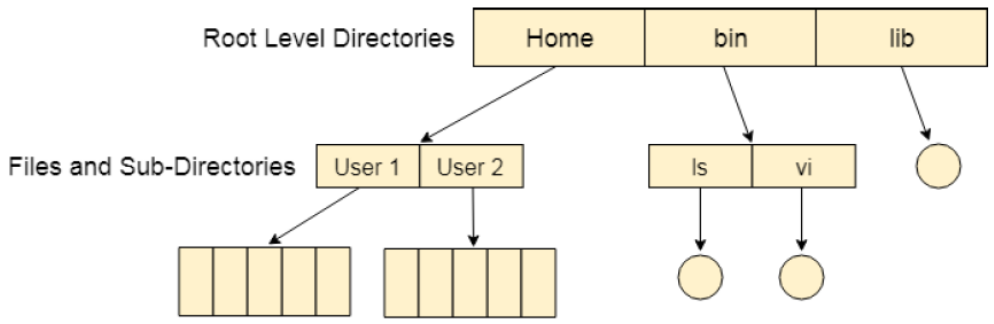
- Organizes files hierarchically, making management easier.
- Allows logical grouping and categorization of files.
- Facilitates easy navigation and access control.
- Supports efficient disk space utilization by minimizing duplicates.
- Scalable for adding new files or directories as needed.

Disadvantages

- Can become complex and hard to navigate with deep structures.
- Long path names may be inconvenient and error-prone.
- Risk of unintentional file duplication across directories.
- Requires ongoing maintenance and organization.

Acyclic Graph Directory

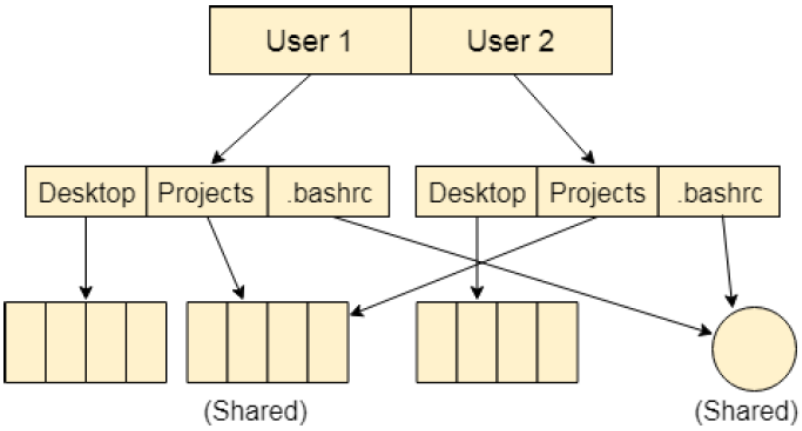
- The tree structured directory system doesn't allow the same file to exist in multiple directories.
- We can provide sharing by making the directory an acyclic graph.
- It permits users to create shared files and directory.
- When the same files need to be accessed in more than one place in the directory structure.
- Directories can contain files as well as sub-directories also.
- A file may contain more than one file path.
- Directory structure does not contain cycles.
- 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



The Structured Directory System

pointer.

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



Acyclic-Graph Structured Directory System

Advantages

- It allows sharing of files and directories.

Disadvantage

- Deletion of shared file or directory is complex. When such file/directory is deleted, directory entry from all related directories should be deleted.
- A file may have multiple file paths. This may create problem in some operations. for example, if user copies al files to backup storage, then such files will be copied multiple times.
- It is difficult to ensure that there are no any cycles in a directory structure.

2.6 File Protection

- Files are used to store information.
- Depending upon the users and applications, this information can be very important. Consider the importance of various records stored in files in some bank. What will be the consequences if this information gets lost, or damaged or corrupted? So, in short, no one wants to lose information stored in files in any way. Everyone wants information stored in files to be safe.
- Here, safety comes in two ways:
  i. Reliability, and
  ii. Protection.
- Both of these ways are described below.
  - **i. Reliability:**
    - Reliability means safety from physical damage.
    - File systems can be damaged by hardware problems, first, dirt, power failures, head crashes and so on. Files may be deleted accidentally (or even intentionally too…!!!).
    - Reliability is generally provided by keeping more than one copies of files, means duplicating files.

- Files are duplicated mostly on some different locations, or in different devices. Magnetic tapes are most widely used for backups of large files.
- Operation of duplicating files can be done either automatically using software or manually.
- If original files are damaged (unfortunately), then they can be retrieved back from the backups.

ii. Protection

- Protection means safety from improper access.
- Protection is required where files can be Shared among various users (or even among processes and machines on network).
- Some files need to be shared among all users, while some need to be shared among limited users.
- For example, a user may want to read, write and execute its own Programs. He also wants to share such programs with his partners. But, he doesn't want that other classmates can access these programs.
- The solution to this is to allow limited sharing. This can be done by limiting the type of file access. Here, accessing a file means to perform some operation on that file. Several operations on files can be controlled.
- Some of these operations are:
  - Read: Read a file.
  - Write: Write a file.
  - Execute: Load and execute a file.
  - Append: Add information at the end of a file.
  - Delete: Free the space allocated to a file.

- There are two most common Ways to limit the type of file access:
  i. Password
  ii. Access Control.
- These two ways are described below:

i. Password:

- Here, a password is associated with each file. Users can access a file, only if he knows the password.
- This scheme looks good, but it suffers from several disadvantages.
- There may be large number f files in a computer system. So, user need to remember lots of password.
- To avoid this problem, if single password is used for all files, then once it is discovered all files are accessible.

ii. Access Control:

- Here a list called an access control list (ACL), is associated with each file. This list specifies the user's name and type of access allowed for that user. This information is stored for all the possible users of the system.
- This list is generally kept in directory entry along with file name and other information.
- This method also suffers from some problems.

- First problem with this method is: Constructing an ACL is tedious task. Also all users of the system are not known in advance.
- Second problem with this method is: ACL Is kept in directory entry. So, now directory entry should be of varying length instead f fixed length. This results in more complicated space management.
- Solution to this problem is to divide all users in various categories and then to allow access on such categories instead of individual users.
- UNIX uses such type of scheme.
- It divides users in three categories:
  - Owner,
  - Group and
  - Others
- Access is given on such category. All the users of that category will be allowed that access.