Introduction

A file is collection of information that is kept on secondary storage, usually as a sequence of bytes, with two views that is a logical view as the users see it and physical views as it actually resides on secondary storage.From the user‘s point of view, one of the most important parts of an operating system is the file system.The file system provides the resource abstractions typically associated with secondary storage. The file system permits users to create data collections, called files, with desirable properties, such as

Long-term existence: Files are stored on disk or other secondary storage and do not vanishwhen turn off the power.

- **Sharable between processes**: Files have names and can have associated access acceptance that allow controlled sharing.
- **Structure**: Depending on the file system, a file can have an internal structure that is convenient for particular applications. In addition, files can be organized into hierarchical or more complex structure to reflect the relationships among files.

The knowledge in a file is defined by its originator. Commonly a file represents programs and data. In general a file is a sequence of bits, bytes, lines or records the meaning of which is defined by file‘s creator and user.

- A file has a certain characterize structure which depends on its type.
- A text file is anarray of characters formulated into lines.
- A source file is a sequence of sub routines and functions.
- An object file is 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.

File Attributes

A file attribute is a metadata that is data about data which describes or is associated with a computer file. For example, an operating system often keeps track of the date a file was created and last modified as well as file‘s size and extension. Files attributes may vary from one operating system to another but typically consists of:

- Name of the file
- Identifier: This unique tag associated with the file, identifies the file within the file system.
- Type
- Location: This information is a director to a device and to the location of the file on that device.
- Size: The current size of the file (in bytes, words and blocks).
- Protection: Access control information determines who can read, who can write and who canexecute the file.
- Time, date and identification: This information may be kept for production, last alteration and lastusage.

File Operations

Any file system provides not only a means to store data organized as files, buta collection of functions that can be performed on files. Distinctive operations includethe following:

a) **Create:** A new file is defined and positioned within the structure of files.
b) **Delete:** A file is removed from the file structure and destroyed.
c) **Open:** An existing file is declared to be ―opened‖ by a process, allowing the process to perform functions on the file.
d) **Close:** The file is closed with respect to a process, so that the process no longer may perform
functions on the file, until the process opens the file again.
e) **Read**: A process reads all or a portion of the data in a file.
f) **Write**: A process updates a file, either by adding new data that expands the size of the file or bychanging the values of existing data items in the file.

File Structure

Files can be structuredas a stream of bytes; the operating system does not know or care what is in thefile. It sees only bytes. The basic things associated with the file are:

a) Field:A field is the primary element of data.An individual field contains a single value, such as aperson‛s last name, age, phone number,date.

- It is identified by its length and data type. Depending on the file design, fields may be fixed lengthor variable length.
- In some cases field may consists of two or three subfields: the actual value to be stored, thename of the field and the length of the field.

b) Record:

- A record is a collection of related fields that can be treated as a unit by some application program. Like a student record would contain such fields as student name, roll number, Date of birth, Class, and so on.
- Depending on design, records may be of fixed or variable length.
- A record will be of variable length if some of its fields are of variable length or if thenumber of fields may vary. In the second case, each field is usually accompanied by a field name.

c) File:

- A file is a collection of similar records. The file is treated as a specific entity by users andapplications and may be referenced by name.
- Files have file names and may be created and deleted. Access control restrictions areusually applied on the files.
- Depending upon the type of user access is being granted. In UNIX we have classified theusers into Owner, Group and Others.
- In some more refined systems, such controls are imposed at the record or even the fieldlevel. Some file systems are structured only in terms of fields, not records. In that case, a file is a collection of fields.

d) Database:

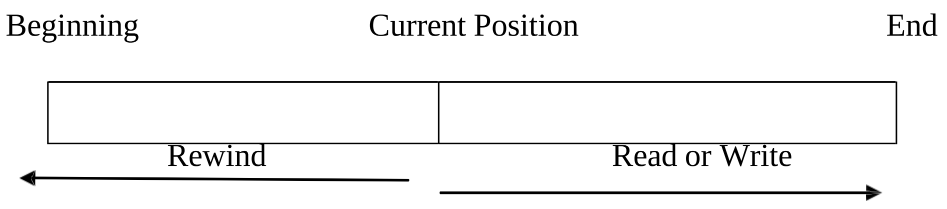
- A database is a collection of related data.
- The crucial aspects of a database are that the relationships that exist among elements of data are explicit and that the database is designed for use by a number of different applications.
- A database may consist of all of the information related to an organization or project, such as a business or a scientific study. The database itself consists of one or more kinds of files.

Usually, there is an isolated database management system that is independent of the operating system, although that system may make use of some file management programs.

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. Some systems provide only one access method for files.

i. **Sequential Access -** The simplest access method is sequential access. Information in the file is handled in order, one record after the other. This mode of access is by far themost common. Like editors and compilers usually approach files in this fashion.Reads and writes make up the bulk of the operations on a file. A readoperation—read next—reads the next portion of the file and automatically advances a file pointer, which observes the I/O location. Similarly, the write operation—write next—appends to the end of the file and advances to the end of the newly written material.



ii. **Direct Access -** Another method is direct access. A file is created of fixed length logical records that allow programs to read and write records rapidly in no particular order. The direct-access method is depending on a disk model of a file, since disks grant irregular access to any file block. For direct access, the file is viewed as a numbered sequence of blocks or records. Thus, we may read block 49, then read block 89, and then write block 6. There are no limitationson the order of reading or writing for a direct-access file.Direct-access files are of enormous use for immediate access to hugevolumeof information. Databases are often of this type. When a query concerning a particular subject lands, we figure out which block contains the answer and then read that block directly to provide the desired information.

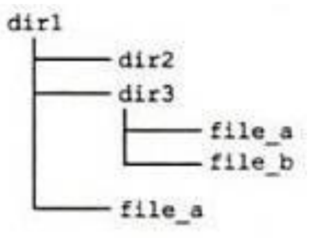
Sequential Access	**Implementation for direct access**
reset	cp=0
Read next	Read cp; Cp=cp+1 ;
Write next	Write cp; Cp=cp+1 ;

Simulation of sequential access on direct access

We can simply simulate sequential access on a direct access file by simply keeping a variable cp that defines our current position as shown in figure. Simulating a direct access file on a sequential-access file, however is extremely inefficient.

Directories and Names

File system allows users to organize files and other file system objects through the use of directories. A directory or a folder has been traditionally defined to be a file system object that contains other file system objects. However, that definition is misleading. In most cases, it is more accurate to define a directory as an object that contains the names of file system objects. On systems where there is a one to one mapping between names and objects, the distinction between name and object may not be important. But some file systems allow objects to have multiple names or even no name; under some circumstances, the distinction becomes important. Entries in directories determine the full pathname associated with a file system object. Starting with the root directory, the full path name is constructed by concatenating the sequence of names traversed.For example consider the file system in below figure. In the directory dir1 contains three names: file_a, dir2 and dir3.The directory dir2 contains no names. The directory dir3 contains the names file_a and file_b. This organization defines six full path names. On UNIX, those names would be:



File system

/dir1/dir2

/dir1/dir3

/dir1/dir3/file_a

/dir1/dir3/file_b

Partitions

Files are typically stored on secondary storage devices (although system Ram, random access memory, may also be used to store files, such as temporary files, for which quick access is desired). a file system may incorporate the notion of a partition which determines on which device a file will be stored.On some systems, such as DOS and windows, the partition is specified as a part of path name. The DOS/Window name c:\rules\section.1 specifies the file system object named \rules\section. 1 in partition c: On other systems like UNIX, partitions are mounted into single unified file system name space.

When partitions are mounted, one partition serves the "root" partition whose name space serves as the initial unified file system name space.Other file systems can then be spliced into the unified file system name space, typically at any existing directory node. The root directory in the mounted file system then replaces the mount point directory in the unified file system name space. The partitions name space exists in the unified file system name space underneath the mount point.

The operating system must decide what becomes of the contents of the directory where a partition is to be mounted. On some systems, the directory must be empty before a partition may be mounted. On others, its contents become hidden. The designer of the operating system must also decide whether a partition may be mounted more than once. Multiple mounting gives each file in the partition multiple names in the unified name space.

Pre-Process Root Directory

On some systems, it is possible to designate a directory that will serve as the root directory for a process. Once such a designation occurs, only the sub tree headed by that directory is visible to the process. On a system where no file sharing is desirable, each user's home directory could be made the root directory for all processes the user executes. To those processes, the root of the file system would be the user's home directory. This capability may also be used by applications to provide additional security. A web or FTP server may be so configured, limiting its access to a particular sub tree reserved for web or FTP data.

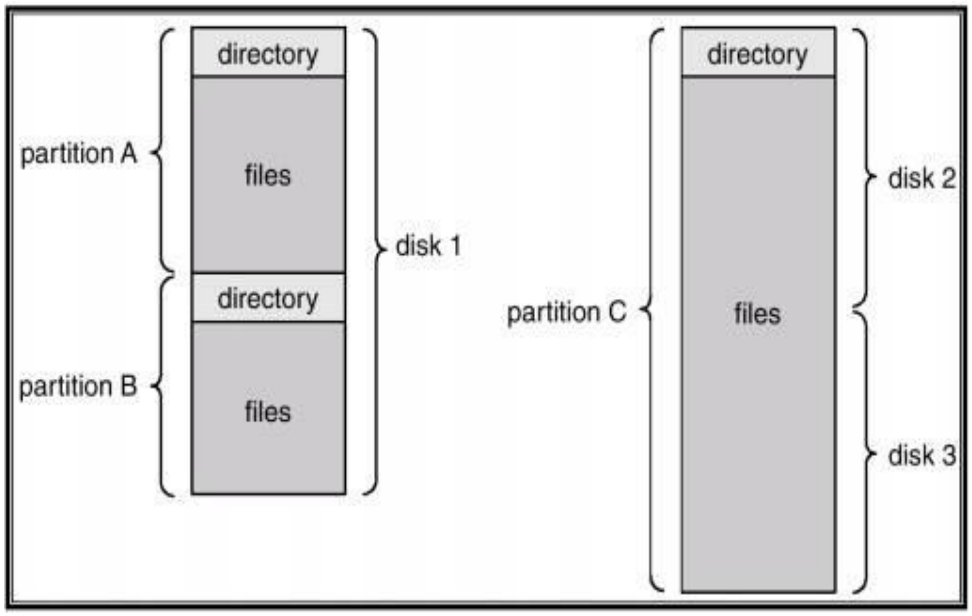
Directory structure

The file systems of computers can be vast. Some systems store millions of files on terabytes of disk. To manage all these data, we need to organize them. This organization engages the use of directories. In many systems, directories are treated as files which keep track of all other files.

A single flat directory can contain a list of all files in a system. Flat directories are those directories in which the root directory contains all system files and there is no other sub directory. When hierarchical directories are used the collection of all directory and sub directory entries defines the totality of system files.

Storage Structure

- Sometimes it is desirable to place numerous file systems on a disk or to use parts of the disk for a file system and other parts for other things, such as swap space or unformatted raw disk space. These parts are known differently as partitions, slices or minidisks
- A file system can be created on each of these parts of the disk. We simple refer to a chunk of a storage that holds a file system as a volume.



File system organization

- Each volume that contains a file system must also contain information about the files in the system. This information is conserved in entries in a device directory or volume table of contents.
- The device directory records information such as name, location, size and type for all files on that volume figure below shows a typical file-system organization.

Directory Overview

To keep track of files, file systems normally have directories or folders. Usually, a directory is itself a file. The directory can be viewed as a symbol table that translates file names into their directory entries. A typical directory entry contains information (attributes, location and ownership) about a file. We want to be able to include entries, to omit entries, to search for a named entry, to list all the entries in the directory.

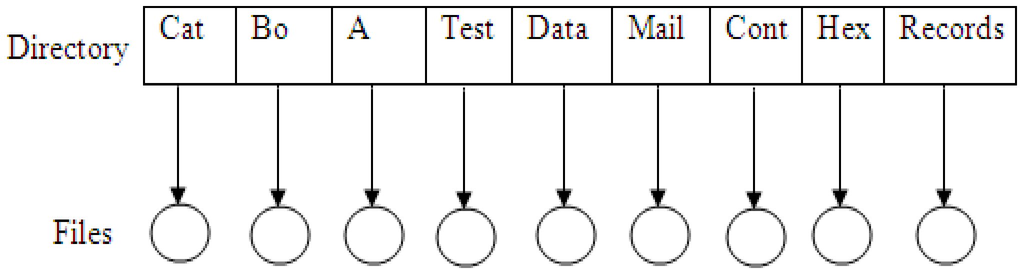
When considering a particular directory structure, there are different operations that are to be performedon a directory:

1. **Search for a file**: We need to be able to search a directory structure to find the entry for a particular file.

2. **Create a file:** New files need to be created and added to directory.
3. **Delete a file:** When a file is no longer required, we want to be able to omit it from the directory.
4. **List a directory**: We need to be able to list the files in directory and the contents of the directoryentry for each file in the list.
5. **Rename a file:** Because the name of a file represents its contents to its users, we must be ableto alter the name when the contents or use of the file changes.
6. **Traverse the file system:** We may wish to approach every directory and every file within the
directory structure. For accuracy it is a good idea to save the contents and structure of the entirefile system at regular intervals.

I. Single – Level directory

The smooth directory structure is the single level directory. All files are enclosed in the same directory, which is easy to support and figure out.



File system organization

On early personal computers this system was trivial, in part because there was only one user. The world's first supercomputer CDC 6600 also had only a single directory for all files, even though it was used by many users at once.

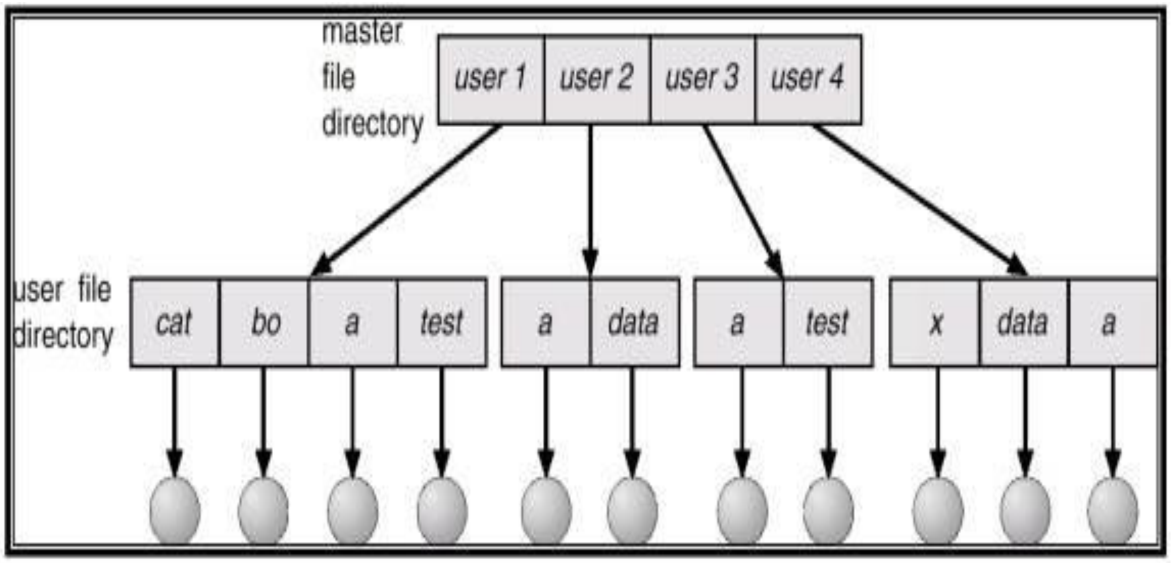
A single-level directory has significant drawbacks, when the number of files increases or when the system has more than one user. Since all files are in the same directory, they must have unique names. If two users call their data file test then the unique name rule is opposed. Even a single user on a single level directory may find it difficult to remember the names of all the files as the number of files increases.

Advantage:
- Easy to support and understand

Disadvantage:
- Requires unique file names
- No natural system for keeping track of filenames.

II. Two level Directory

The standard solution to drawbacks of a single-level directory is to create a separate directory for each user. In the two level directory structures, each user has its own user file directory (UFD). The UFDs have identical structures but each lists only the files of a single user.



Two level directories

A two level directory can be understood of as a tree, or an inverted tree of height 2. The root of the tree is the master file directory (MFD) and its direct descendants are the UFDs. The descendants of UFDs are the files themselves and the files are the leaves of the trees.

When a user job starts or a user logs in, the systems master file directory (MFD) is searched. The MFD is ordered by the user name or account number and each entry points to the UFD.

Advantages:

- Solves the name collision problem
- Isolates users from one another, a form of protection.
- Efficient searching.

Disadvantages:

- This structure effectively isolates one user from another.
- Restricts user cooperation
- No logical grouping capability.

III. Tree Structured Directories

In the past we have seen how to look a two-level directory as a two-level tree, the natural generalization is to extend the directory structure to a tree of arbitrary height. This generalization allows users to construct theirown subdirectories and to organize their files accordingly. A tree is the most trivial directory structure. The tree has a root directory, and every file in the system has a unique path name.A directory (or subdirectory) consists of a set of files or subdirectories. Adirectory is simply another file, but it is treated in a specific way. All directories have the identical internal format. One bit in each directory entry describes the entry as a file (0) or as a subdirectory (1).

Path names can be of two types: absolute and relative

- An **absolute path** name commences at the root and follows a path down to the specified file,giving the directory names on the path.
- A **relative path** name characterizes a path from the current directory

With a tree structured directory system, user can be granted access in addition to their files, the files of other users. For example, user B can access a file of user A by specifying its path names. User B can specify either absolute or relative path name. Alternatively, user B can change her current directory to be user A‘s directory and access the file by its file names.

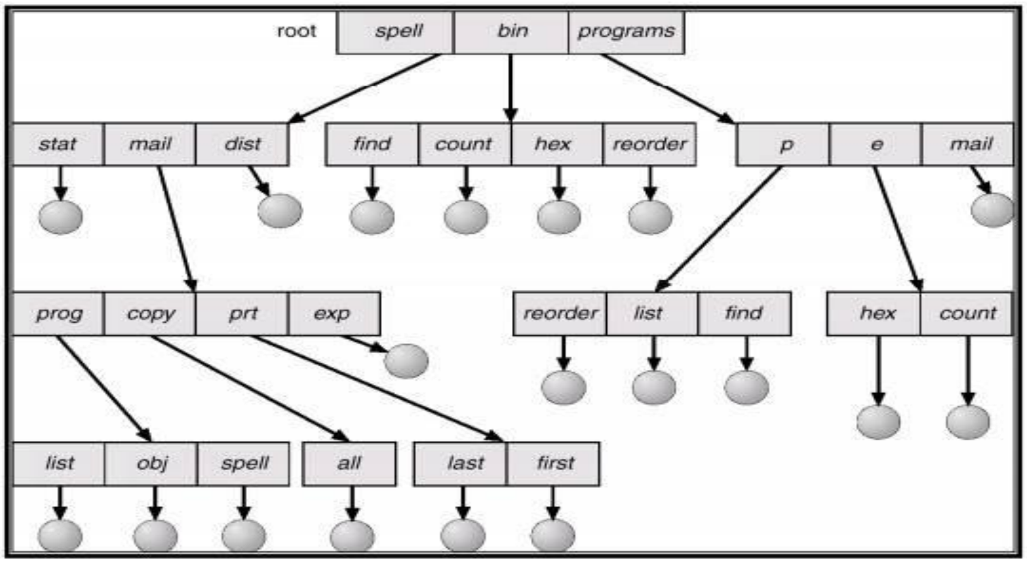


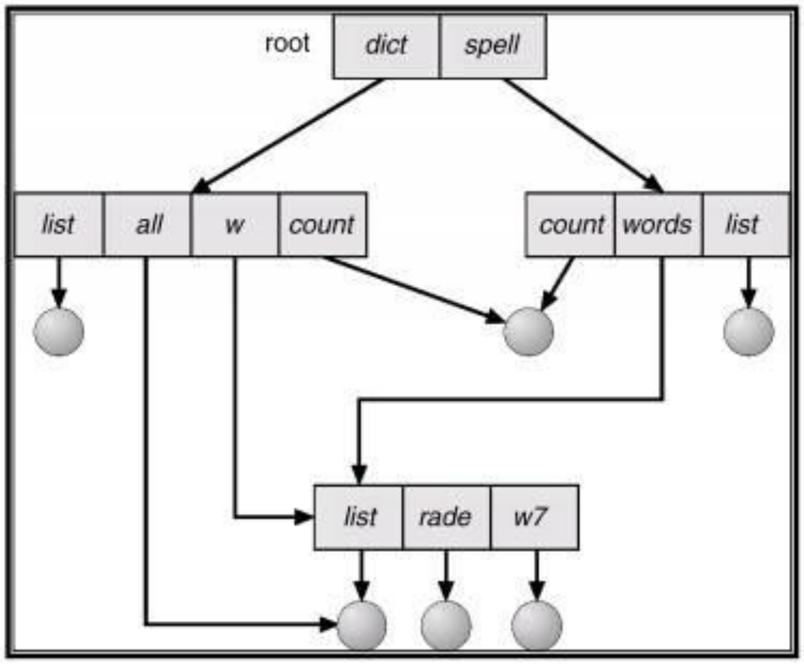
Fig 10.7 Tree structured directories

Special system calls are used to generate and omit directories. In normal use, each process has a current directory. The current directory should contain most of the files that are of current interest to the process. When reference is made to a file, the current directory is searched. If a file is needed that is not in the current directory, then the user usually must either specify a path name or switch the current directory to be the directory holding that file.

To switch directories, a system call is granted that takes a directory name as a parameter and uses it to reconsider the current directory. Thus, the user can switch his current directory whenever he desires. From one switchdirectory system call to the next, all open system calls search the current directory for the specified file. Note that the search path may or may not contain a special entry that stands for "the current directory."

IV. A cyclic graph directories

Examine two programmers who are functioning on a joint project. The files correlatedwith that project can be stored in a subdirectory, detaching them from other projects and files of the two programmers. But since both programmers are equally answerable for the project, bothneed the subdirectory to be in their own directories. The trivial subdirectory should be shared. A shared directory or file will exist in the file system in two (or more) places at once. A tree structure halts the sharing of files or directories. An acyclic graph —that is, a graph with no cycles—grants directories to share subdirectories and files. The same file or subdirectory may be in two distinctdirectories. The acyclic graph is a natural abstractionof the tree- structured directory scheme.



Acyclic graph directory

It is essential to note that a shared file (or directory) is not the same as twocopies of the file. With two copies, each programmer can view the copy ratherthan the original, but if one programmer modifies the file, the changes will notappear in the other's copy. With a shared file, only one actual file exists, so anychanges made by one person are immediately visible to the other. Sharing is particularly essential for subdirectories; a new file generated by one person will automatically appear in all the shared subdirectories.

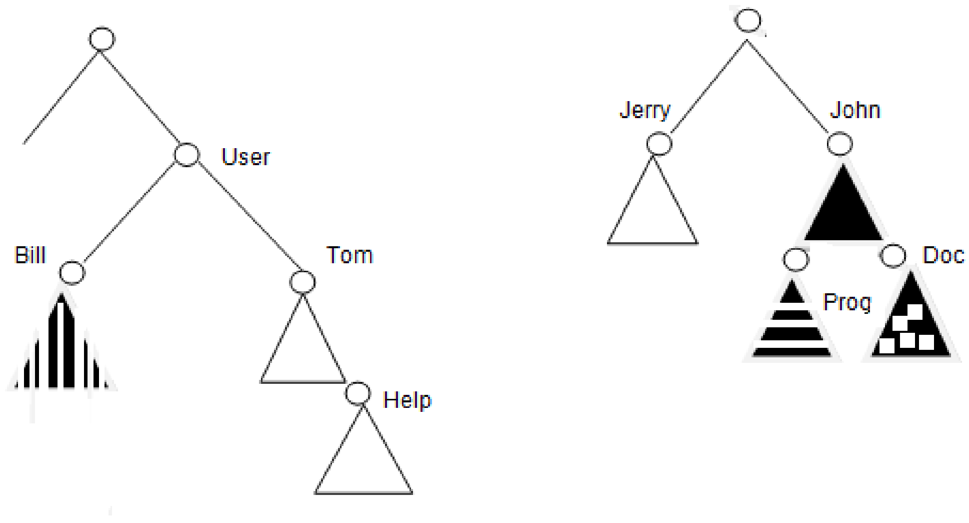
When people are functioning as a team, all the files they need to share can beput into one directory. The UFD of each team member will enclose this directory of shared files as a subdirectory. Even in the case of a single user, the user's file organization may require that some file be placed in different subdirectories. For example, a program written for a distinct project should be both in the directory of all programs and in the directory for that project.

Directory entries

Information about files is saved in a directory entry. This information saved depends on the operating system.Typically a directory entry encloses informationabout files ownership, location, size, access rights, times of different kinds of events (last alteration, creation).Most operating systems require file to be opened before it can be accessed. The open operation contains a pointer to the files directorylocation so all successive reference to the file can be via the pointer rather than the files name.On some systems, the directory entry points to a separate file structure where information about the file is stored. For example, on a UNIX system, File information is stored in a structure called an inode. The directory entry contains only the files name and its inode number.

File Mounting

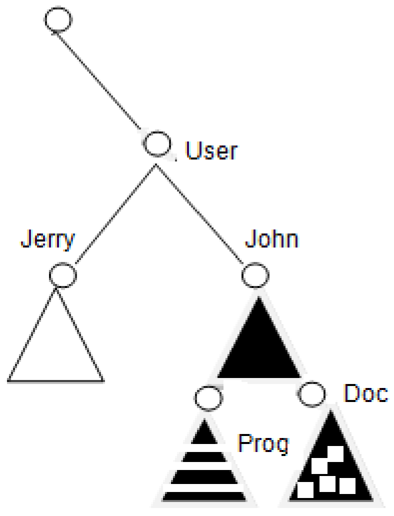
Just as a file must be opened before it is used, a file must be mounted before it can be available to processes on the system. The mount procedure is straight forward. The operating system is given the name of the device and the mount point the location within the file structure where the file system is to be attached.



(a) (b)

File System (a) Existing System (b) Unmounted Volume

Typically, a mount point is an empty directory. Next the operating system verifies that the device contains a valid file system. Finally the operating systems notes in its directory structure that a file system is mounted at the specified mount point. This scheme enables the operating system to traverse its directory structure, switching among the file systems as appropriate . To illustrate the file mounting consider the file system depicted in the figure, where the triangles represent sub-directories, At this point only the files on existing file system can be accessed.



Effects of unmounting the File

File Sharing

We explore the motivation for file sharing and some of the difficulties involved in allowing users to share files. Such file sharing is very desirable for users who want to collaborate and to reduce the effort required to achieve a computing goal.

Protection

- The protection of the 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. When information is stored in a computer system, we want to keep it safe from physical damage and improper access.
- Reliability is generally provided by duplicate copies of files.
- File systems can be damaged by hardware problems, power surges and failures. Files may be deleted accidently. Bugs in the file system can cause file contents to be lost.

Multiple Users

- When an operating system accommodates numerous users, the issues of file sharing, file naming, and file protection become notable.
- Given a directory structure that grants files to be shared by users, the system must negotiate thefile sharing. The system can either grant a user to access the files of other users by default or require that a user specifically grant access to the files.
- To implement sharing and protection the system must manage more file and directory attributes that are essential on a single-user system. Although abundant approaches have been taken to this requirement historically.
- Most systems have emerged to use the concepts of file (or directory) owner (or user) and group. The owner is the user who can modify attributes and grant access and who has the most control over the file.

- The group attribute specify a subset of users who can share access to the file. For example, the owner of a file on a UNIX system can issue all operations on a file, while members of the file'sgroup can execute one subset of those operations, and all other users can execute another subset of operations.Exactly which operations can be executed by group members and other users is definable by the file's owner.
  More details on permission attributes are included in the next section.

Remote File Systems

- With the advent of networks, communication among remote computers became possible.Networking grants the sharing of resources spread across a campus or even around the world.
- One accessible resource to share data is in the form of files.Through the evolution of network andfile technology, remote file-sharingmethods have changed.
- The first implemented method involves manually transferring files between machines viaprograms like ftp.
- The second leading method uses a distributed file system (DFS) in which remote directories isvisible from a local machine.
- The third method, the World Wide Web, is a reversion to the first. A browser is needed to gainaccess to the remote files, and separate operations (essentially a wrapper for ftp) are used to transfer files. Ftp is used for both anonymous and authenticated access. Anonymous accessgrants a user to transfer files without having an account on the remote system. The World Wide Web uses anonymous file exchange almost exclusively.

The Client- Server Model

- Remote file systems allow a computer to mount one or more file systems from one or more remote machines. The machine containing the files is the server, and the machine looking for access to the files is the client.
- The client-server link is common with networked machines. Generally, the server declares that a resource is available to clients and specifies exactly which resource and exactly which clients.
- A server can serve numerous clients, and a client can use numerous servers, depending on the implementation details of a given client-server facility.
- The server usually specifies the available files on a volume or directory level. Client identificationis more difficult.
- A client can be specified by a network name or other identifier, such as an IP address, but these can be spoofed, or imitated. As a result of spoofing, an unauthorized client could be allowed access to the server.
- More secure solutions include secure authentication of the client via encrypted keys.

File Protection and security

When information is kept in computer system, we want to keep it safe from the physical damage and improper access. Reliability is generally provided by duplicate copies of files. Many computers have systems programs that automatically copy disk disk files to tape at regular intervals to maintain a copy for use in future if the computer system accidently gets destroyed. File systems can be damaged by hardware problems , power surges etc.

Goals of Protection

a. To prevent malicious misuse of the system by users
b. To ensure that each shared resources is used only in accordance with system policies.
c. To ensure that errant programs cause the minimal amount of damage possible.