File

A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.

File Structure

A File Structure should be according to a required format that the operating system can understand.

- A file has a certain defined structure according to its type.
- A text file is a sequence of characters organized into lines.
- A source file is a sequence of procedures and functions.
- An object file is a sequence of bytes organized into blocks that are understandable by the machine.
- When operating system defines different file structures, it also contains the code to support these file structure. Unix, MS-DOS support minimum number of file structure.

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

- Collection of files is a file directory. 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.

  Information contained in a device directory are:

  Name, Type, Address, Current length, Maximum length, Date last accessed

  Date last updated, Owner id, Protection information

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.

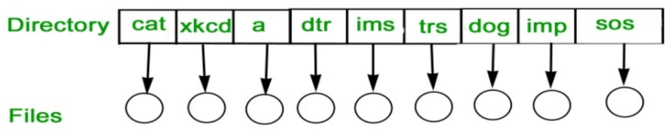
Advantages of maintaining directories are:

- **Efficiency:** A file can be located more quickly.
- **Naming:** It becomes convenient for users as two users can have same name for different files or may have different name for same file.
- **Grouping:** Logical grouping of files can be done by properties e.g. all java programs, all games etc.

SINGLE-LEVEL DIRECTORY

In this a single directory is maintained for all the users.

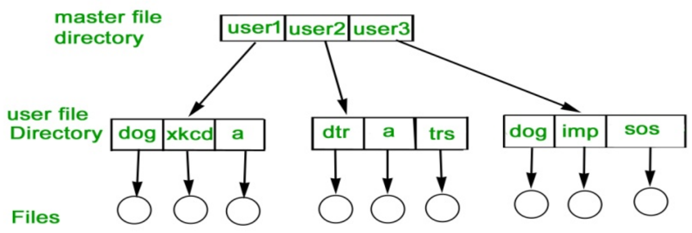
- **Naming problem:** Users cannot have same name for two files.
- **Grouping problem:** Users cannot group files according to their need.
-



TWO-LEVEL DIRECTORY

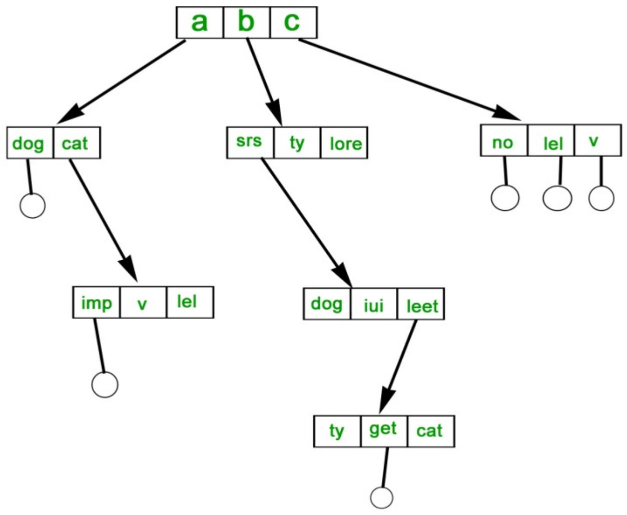
In this separate directories for each user is maintained.

- Path name:Due to two levels there is a path name for every file to locate that file.
- Now,we can have same file name for different user.
- Searching is efficient in this method.



TREE-STRUCTURED DIRECTORY :

Directory is maintained in the form of a tree. Searching is efficient and also there is grouping capability. We have absolute or relative path name for a file.



Acyclic-Graph Directories

- When the same files need to be accessed in more than one place in the directory structure (e.g. because they are being shared by more than one user / process), it can be useful to provide an acyclic-graph structure. (Note the ***directed*** arcs from parent to child.)
- UNIX provides two types of ***links*** for implementing the acyclic-graph structure. (See "man ln" for more details.)
    - A ***hard link*** (usually just called a link) involves multiple directory entries that both refer to the same file. Hard links are only valid for ordinary files in the same filesystem.
    - A ***symbolic link***, that involves a special file, containing information about where to find the linked file. Symbolic links may be used to link directories and/or files in other filesystems, as well as ordinary files in the current filesystem.
- Windows only supports symbolic links, termed ***shortcuts.***
- Hard links require a ***reference count***, or ***link count*** for each file, keeping track of how many directory entries are currently referring to this file. Whenever one of the references is removed the link count is reduced, and when it reaches zero, the disk space can be reclaimed.
- For symbolic links there is some question as to what to do with the symbolic links when the original file is moved or deleted:
    - One option is to find all the symbolic links and adjust them also.
    - Another is to leave the symbolic links dangling, and discover that they are no longer valid the next time they are used.

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.

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.

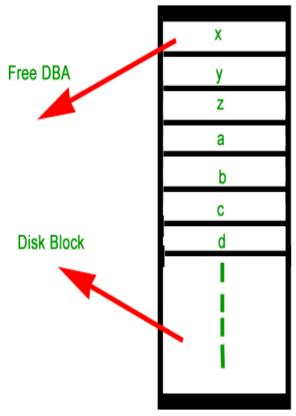
Indexed sequential access

- 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.

Disk Free Space Management:

Just as the space that is allocated to files must be managed ,so the space that is not currently allocated to any file must be managed. To perform any of the file allocation techniques,it is necessary to know what blocks on the disk are available. Thus we need a disk allocation table in addition to a file allocation table.The following are the approaches used for free space management.

1. **Bit Tables** : This method uses a vector containing one bit for each block on the disk. Each entry for a 0 corresponds to a free block and each 1 corresponds to a block in use.
   For example: 00011010111100110001
   In this vector every bit correspond to a particular vector and 0 implies that, that particular block is free and 1 implies that the block is already occupied. A bit table has the advantage that it is relatively easy to find one or a contiguous group of free blocks. Thus, a bit table works well with any of the file allocation methods. Another advantage is that it is as small as possible.
2. **Free Block List** : In this method, each block is assigned a number sequentially and the list of the numbers of all free blocks is maintained in a reserved block of the disk.



File Sharing

10.5.1 Multiple Users

- On a multi-user system, more information needs to be stored for each file:
    - The owner (user) who owns the file, and who can control its access.
    - The group of other user IDs that may have some special access to the file.
    - What access rights are afforded to the owner (**U**ser), the **G**roup, and to the rest of the world (the universe, a.k.a. **O**thers.)
    - Some systems have more complicated access control, allowing or denying specific accesses to specifically named users or groups.

10.5.2 Remote File Systems

- The advent of the Internet introduces issues for accessing files stored on remote computers
    - The original method was ftp, allowing individual files to be transported across systems as needed. Ftp can be either account and password controlled, or ***anonymous***, not requiring any user name or password.
    - Various forms of ***distributed file systems*** allow remote file systems to be mounted onto a local directory structure, and accessed using normal file access commands. (The actual files are still transported across the network as needed, possibly using ftp as the underlying transport mechanism.)
    - The WWW has made it easy once again to access files on remote systems without mounting their filesystems, generally using (anonymous) ftp as the underlying file transport mechanism.

10.5.2.1 The Client-Server Model

- When one computer system remotely mounts a filesystem that is physically located on another system, the system which physically owns the files acts as a ***server***, and the system which mounts them is the ***client.***
- User IDs and group IDs must be consistent across both systems for the system to work properly. (I.e. this is most applicable across multiple computers managed by the same organization, shared by a common group of users.)
- The same computer can be both a client and a server. (E.g. cross-linked file systems.)
- There are a number of security concerns involved in this model:
    - Servers commonly restrict mount permission to certain trusted systems only. Spoofing (a computer pretending to be a different computer) is a potential security risk.
    - Servers may restrict remote access to read-only.
    - Servers restrict which filesystems may be remotely mounted. Generally the information within those subsystems is limited, relatively public, and protected by frequent backups.
- The NFS (Network File System) is a classic example of such a system.

10.5.2.2 Distributed Information Systems

- The ***Domain Name System, DNS,*** provides for a unique naming system across all of the Internet.
- Domain names are maintained by the ***Network Information System, NIS***, which unfortunately has several security issues. NIS+ is a more secure version, but has not yet gained the same widespread acceptance as NIS.
- Microsoft's ***Common Internet File System, CIFS***, establishes a ***network login*** for each user on a networked system with shared file access. Older Windows systems used ***domains***, and newer systems (XP, 2000), use ***active directories.*** User names must match across the network for this system to be valid.
- A newer approach is the ***Lightweight Directory-Access Protocol, LDAP,*** which provides a ***secure single sign-on*** for all users to access all resources on a network. This is a secure system which is gaining in popularity, and which has the maintenance advantage of combining authorization information in one central location.

10.5.2.3 Failure Modes

- When a local disk file is unavailable, the result is generally known immediately, and is generally non-recoverable. The only reasonable response is for the response to fail.
- However when a remote file is unavailable, there are many possible reasons, and whether or not it is unrecoverable is not readily apparent. Hence most remote access systems allow for blocking or delayed response, in the hopes that the remote system (or the network) will come back up eventually.

10.5.3 Consistency Semantics

- ***Consistency Semantics*** deals with the consistency between the views of shared files on a networked system. When one user changes the file, when do other users see the changes?

- At first glance this appears to have all of the synchronization issues discussed in Chapter 6. Unfortunately the long delays involved in network operations prohibit the use of atomic operations as discussed in that chapter.

10.5.3.1 UNIX Semantics

- The UNIX file system uses the following semantics:
    - Writes to an open file are immediately visible to any other user who has the file open.
    - One implementation uses a shared location pointer, which is adjusted for all sharing users.
- The file is associated with a single exclusive physical resource, which may delay some accesses.

10.5.3.2 Session Semantics

- The Andrew File System, AFS uses the following semantics:
    - Writes to an open file are not immediately visible to other users.
    - When a file is closed, any changes made become available only to users who open the file at a later time.
- According to these semantics, a file can be associated with multiple (possibly different) views. Almost no constraints are imposed on scheduling accesses. No user is delayed in reading or writing their personal copy of the file.
- AFS file systems may be accessible by systems around the world. Access control is maintained through (somewhat) complicated access control lists, which may grant access to the entire world (literally) or to specifically named users accessing the files from specifically named remote environments.

10.5.3.3 Immutable-Shared-Files Semantics

- Under this system, when a file is declared as ***shared*** by its creator, it becomes immutable and the name cannot be re-used for any other resource. Hence it becomes read-only, and shared access is simple.

Protection

- Files must be kept safe for reliability (against accidental damage), and protection (against deliberate malicious access.) The former is usually managed with backup copies. This section discusses the latter.
- One simple protection scheme is to remove all access to a file. However this makes the file unusable, so some sort of controlled access must be arranged.

10.6.1 Types of Access

- The following low-level operations are often controlled:
    - Read - View the contents of the file

    - Write - Change the contents of the file.
    - Execute - Load the file onto the CPU and follow the instructions contained therein.
    - Append - Add to the end of an existing file.
    - Delete - Remove a file from the system.
    - List -View the name and other attributes of files on the system.
- Higher-level operations, such as copy, can generally be performed through combinations of the above.

10.6.2 Access Control

- One approach is to have complicated ***Access Control Lists, ACL,*** which specify exactly what access is allowed or denied for specific users or groups.
    - The AFS uses this system for distributed access.
    - Control is very finely adjustable, but may be complicated, particularly when the specific users involved are unknown. (AFS allows some wild cards, so for example all users on a certain remote system may be trusted, or a given username may be trusted when accessing from any remote system.)