UNIT-VI

File system Interface

Objectives:

Students will be able

- To explain concepts of a file
- To discuss file access methods, file sharing, and directory structures
- To explore file-system protection

Syllabus: File system Interface

Concept of a file (File attributes, file operations), Access Methods (Sequential access, direct access), Directory structure (overview, single-level, two-level, tree structured, acyclic-graph), File system mounting, files sharing (multiple users, remote file systems) and protection

Outcomes:

Students will be able to

- Understand about file operations, file attributes.
- Know the file structure and directory structure.
- Learn about various types of directories and file sharing.

Learning material

6.1. Concept of File

- ➢ A file is a collection of related information that is recorded on secondary storage.
- ➢ A file is a smallest allotment of logical secondary storage that is data can't be written to secondary storage unless they are within a file.
- ➢ File represent programs (both source and object forms) and data.
- ➢ Data file may be numeric, alphabetic, alphanumerical, or binary.
- ➢ A file is a sequence of bits, bytes, lines, or records, the meaning of which is defined by file's creator and user.
- ➢ The information of a file is defined by its creator.
- ➢ Many different type of information may be stored in file.
- ➢ Examples: source programs, object programs, executable programs, numeric data, text, payroll records, graphic images, sound recording and so on.

6.1.1. File Attributes:

- **Name:** The symbolic file name is the only information kept in human readable code.
- **Identifier**: This is unique tag, usually a number, identifies the file within the file system. It is the non human readable name for the file.
- **Type**: This information is needed for those systems that support different type.
- **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 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.
- **Time & Date:** This information may be kept for creation, last modification, and last use. These data can be useful for protection, security and usage monitoring.

6.1.2. File Operations:

A file is an abstract data type. Operating system must do for each of the six basic file operations.

- **Creating a file:** Two steps are required to create a file. First, space in the file system must be found for the file. Second, 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.
- **Reading a file:** to read from a file, we use a system call that specifies the name of the file and where the next block of the file should be put. Again, the directory is searched for the associated directory entry, and the system needs to keep a read pointer to the location in the file where the next read is take place. Once the read has taken place, the read pointer is updated.
- **Repositioning with in a file:** The directory is searched for the appropriate entry, and the current file position is set to a given value.
- **Deleting a file:** whatever files want to delete from directory, those files have to found in directory. If file is found then it will be deleted. Deleted file space used for store the next file.
- **Truncating a file:** The file attributes are not changed but the content of file deleted.
- There are several issues are associated with an open file.

❖ **File pointer:** it is a unique pointer for each process operating on file.

❖ **File open count:** how many times has the current file has been opened and not yet closed. When this counter reaches zero the file can be removed from the table.

❖ **Disk location of the file:** most file operations required to modify data within a file.

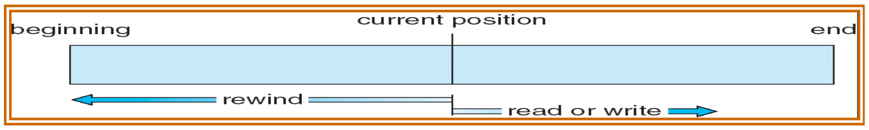
❖ **Access right:** Each process can open a file in access mode. This information stored on pre process table. So that the operating system can allow or deny subsequent I/O operations.

6.2. File Access Methods

Files contain information. When it is used, this information must be accessed and read into computer memory. The information of a file can be accessed in different ways.

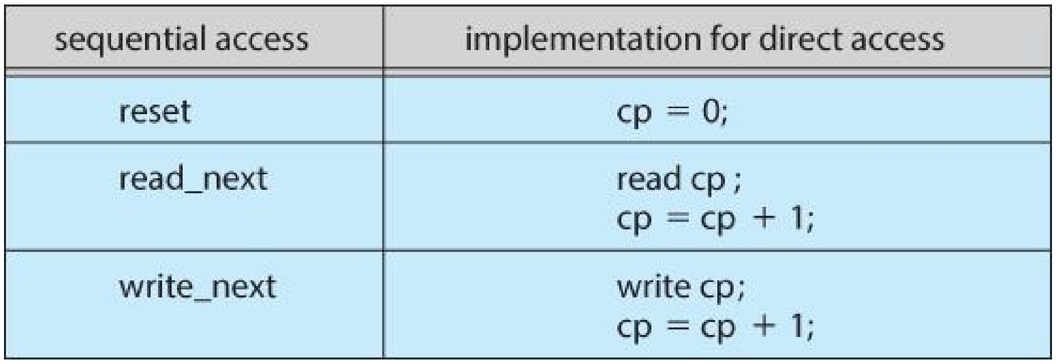
6.2.1. Sequential Access:

- It is the simplest access method and it is sequential.
- Information is processed one after another in sequential.
- This mode of access is common and used in editors and compilers.
- The general operations on files are read and write.
- A read operation reads the next portion of the file and automatically advances the file pointer, which tracks the I/O location.
- A write operation appends to end of the file and advances to the end of newly written material.



6.2.2. Direct Access:

- Direct access method is also called as relative access.
- A file is made up to a fixed size logical records that allow programs read and write records in any order.
- Databases and airline reservations are example for this mode.
- The direct access is based on the disk model of a file since disk allows random access to any file block.
- For direct access, the file is viewed as a numbered sequence of block or record.
- Thus, we may read block 14 then block 59 and then we can write block 17.
- There is no restriction on the order of reading and writing for direct access file.
- A block number provided by the user to the operating system is normally a relative block number, the first relative block of the file is 0 and then 1 and so on.



sequential access	implementation for direct access
reset	cp = 0;
read_next	read cp ; cp = cp + 1;

write_next	write cp;
	cp = cp + 1;

6.2.3. Other Access Methods:

- Other access methods can be built on top of a direct access method.
- These methods constructs index for files.
- This index containing pointer to the various blocks.
- With large files, the index file itself may become too large to be kept in memory.
- There is a solution to create an index for the index file.
- The primary index file would contain pointer to secondary index files, which would point to the actual data items.
- Example: Indexed sequential access method(ISAM)

6.3. Directory Structure

- The file systems are extensive in computers.
- Some systems store some millions of files on disks.
- To manage all these data, we need to organize them in two parts.
- **First,** disks are split into one or more partitions also known as mini disks.
    - Each disk on system contains at least one partition, which is low level structure in which files and directories reside.
- **Second**, each partition contains information about files within it.
    - This information is kept in entries in a device directory or volume table of contents.
    - The device directory records information such as name, location, size and type for al files on this partition.

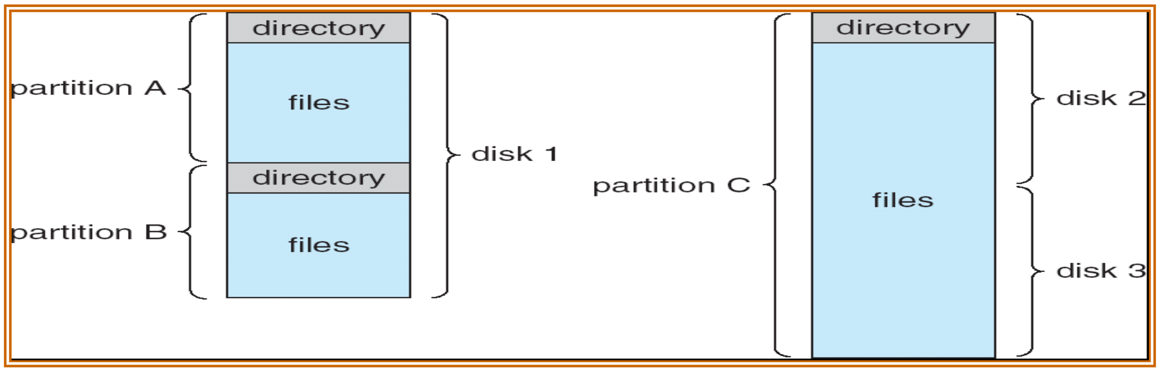


Fig: A typical file system organization

When considering a particular directory structure, we need to keep in mind the operations that are to be performed on a directory.

- **Search for a file:** we need to be able to search a directory structure to find the entry for a particular file. Since files have symbolic names and similar names may indicate a relationship between files, we may want to be able to find all files whose names match a particular pattern.
- **Create a file:** new files needed to be crated and added to the directory.
- **Delete a file:** when a file is not needed then it removed from directory.
- **List a directory:** we need to be able to list the files in a directory, and the contents of the directory entry for each file in a list.
- **Rename a file:** the name of a file represents its content to its user, the name must be changeable when the contents or use of the file changes.
- **Traverse the file system:** In file system, every file and every directory accessed by the user. Save files content and its structure at each regular interval time. if we are saving like regular interval times then it is easy to backup in case of system failure.

6.3.1. Single-Level Directory

- ➢ All files are contained in single directory.
- ➢ It has a limitation when number of users is more than one.

- In single level directory all file names are unique.
- If two users call their data file name as test, then unique-name rule is violated.

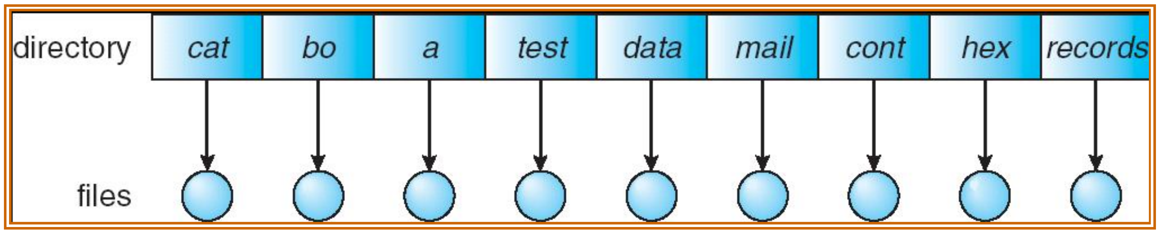


Fig: Single-level directory.

6.3.2. Two-Level Directory

- In single-level directory has a problem if two files are same name.
- To overcome the above problem use Two-level directory for each user.
- In Two-level directory structure, each user has own directory called as user file directory (UFD).
- Each UFD has a similar structure but list of files are different from one UFD to other UFD.
- When a user job is started, then the system searches in Master File Directory (MFD).
- In MFD each user has their name and account number.
- When a user wants to refer a particular file then he must search in his own directory or UFD.
- If a user create or delete files then those operations are done from their local UFD's.

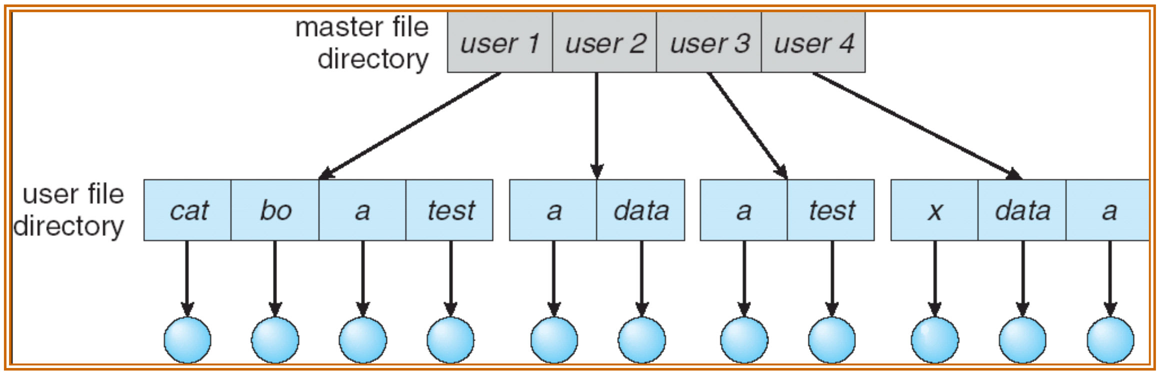


Fig: Two-level Directory Structure

6.3.3. Tree-Structured Directories

- The directory structure is a tree with arbitrary heights.
- The tree has a root directory and every file path is unique.
- A directory contains set of files or subdirectories.
- The internal format of each directory is same.
- We are using bits to represent files and directories.
- ‘1’ represents for directories and ‘0’ represents file.
- Tree structure directories are very efficient in search operation.
- An interesting policy decision in a tree-structured directory structure is how to handle the deletion of a directory.
- Each file has specific path. These paths are divided into two parts. First one is absolute path, second is relative path.
- 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 directory.
- An interesting policy decision in a tree-structured directory structure is how to handle the deletion of a directory.
- If a directory is empty, its entry in its containing directory can simply be deleted.
- Suppose the directory to be deleted is not empty, but contains several files or subdirectories then one of two approaches can be taken.
- Some systems, such as MS-DOS, will not delete a directory unless it is empty.
- Thus, to delete a directory, the user must delete all the files in that directory.
- if any subdirectories exist , this procedure must be applied recursively to them, so that they can be deleted.
- The other approach taken by UNIX rm command is to provide the option that, when a request is made to delete a directory, that entire directory’s files and subdirectories are also to be deleted.

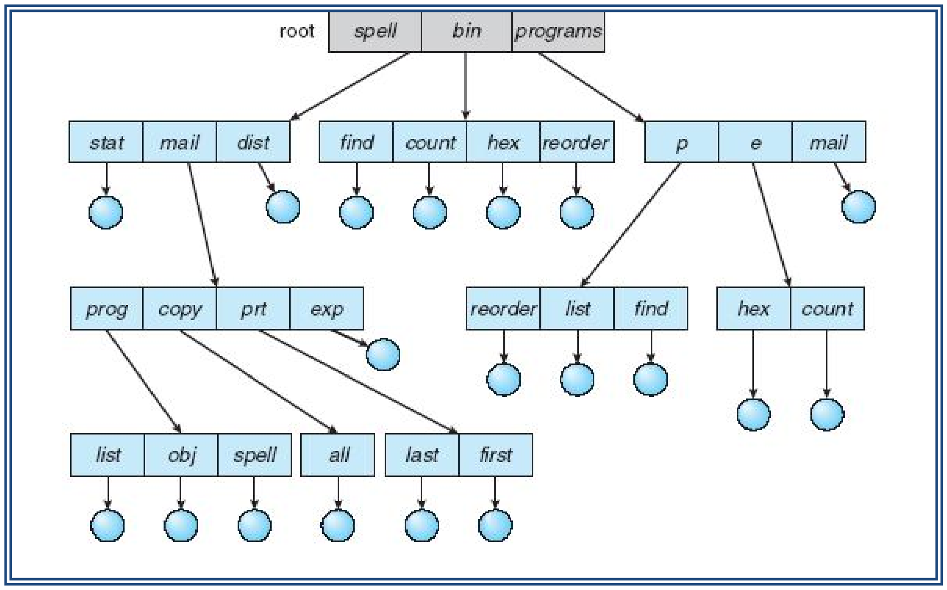


Fig: Tree-structured directory structure

6.3.4. Acyclic-Graph Directories

- An acyclic graph is, a graph with no cycles that allows directories to share subdirectories and files.
- The *same* file or subdirectory may be in two different directories. The acyclic graph is a natural generalization of the tree-structured directory scheme.
- An acyclic-graph directory structure is more flexible than is a simple tree
- Structure, but it is also more complex.
- Several problems are there in acyclic graph:
- First one is a file may have multiple absolute path names. Consequently, distinct file names may refer to the same file.
- This situation is similar to the aliasing problem for programming languages.
- If we are trying to traverse the entire file system to find a file, to accumulate statistics on all files, or to copy all files to backup storage.
- This problem becomes significant, since we do not want to traverse shared structures more than once.

- Another problem involves deletion. When can the space allocated to a shared file be de allocated and reused?
- One possibility is to remove the file whenever anyone deletes it, but this action may leave dangling pointers to the now-nonexistent file.
- If the remaining file pointers contain actual disk addresses, and the space is subsequently reused for other files, these dangling pointers may point into the middle of other files.

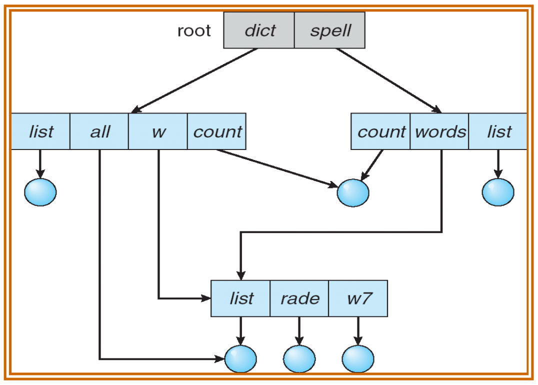


Fig: Acyclic-graph directory structure

6.3.5. General Graph Directory

If cycles are allowed in the graphs, then several problems can arise:

- Search algorithms can go into infinite loops. One solution is to not follow links in search algorithms.
- Sub-trees can become disconnected from the rest of the tree and still not have their reference counts reduced to zero.
- Periodic garbage collection is required to detect and resolve this problem.

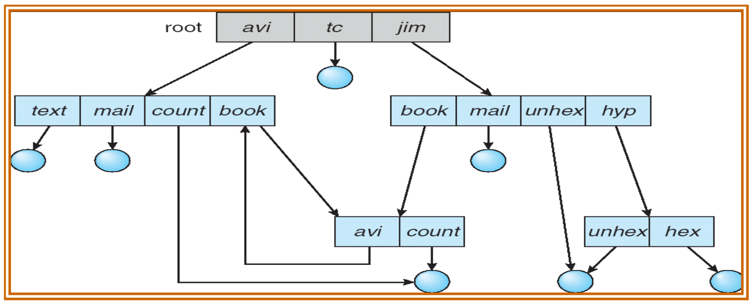
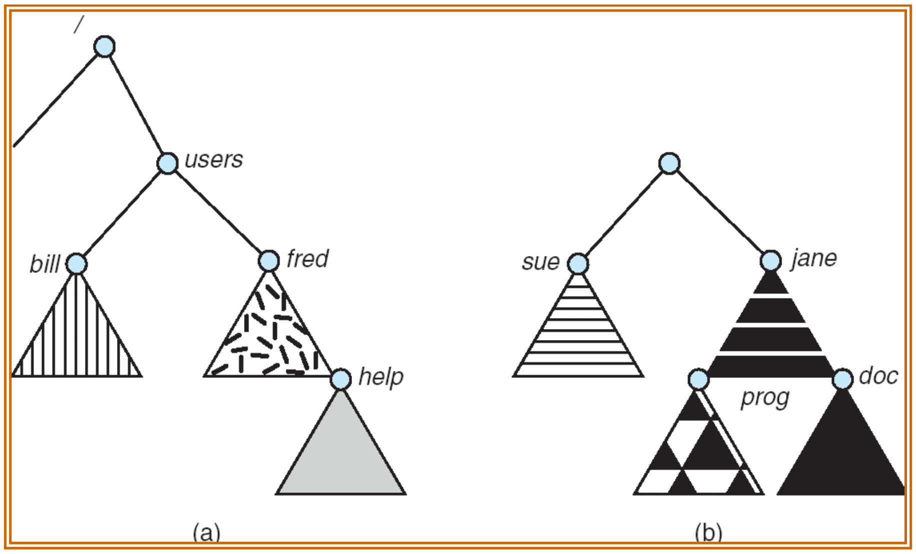


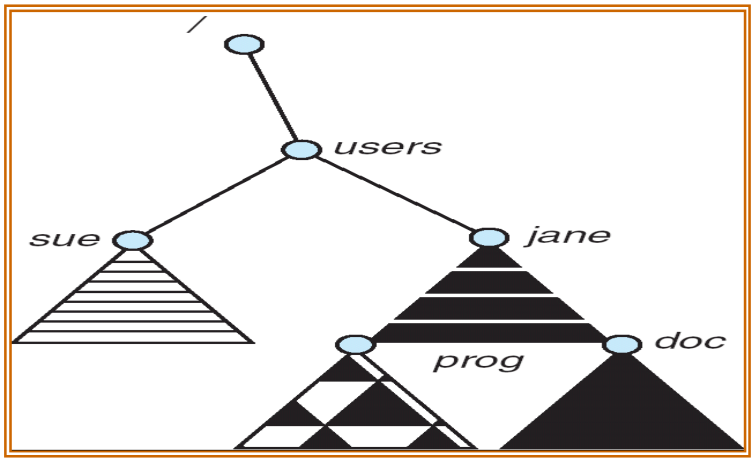
Fig: General graph directory

6.4. File System Mounting

- Combining two or more number of files into a large tree structure is the basic idea of file system mounting.
- In the file system we are using mount command for the purpose of at which point file has to mount.
- That means it provide a mount point (directory) on which to attach it.
- Once a file system is mounted onto a mount point, any further references to that directory actually refer to the root of the mounted file system.
- Any files (or sub-directories) that had been stored in the mount point directory prior to mounting the new file system are now hidden by the mounted file system, and are no longer available.
- For this reason some systems only allow mounting onto empty directories.
- File systems can only be mounted by root, unless root has previously configured certain file system to be mountable onto certain pre-determined mount points.
- Anyone can run the mount command to see what file systems is currently mounted. File systems may be mounted read-only, or have other restrictions imposed.



(a) Existing. (b) Unmounted Partition



Mount Point

- The traditional Windows OS runs an extended two-tier directory structure, where the first tier of the structure separates volumes by drive letters, and a tree structure is implemented below that level.
- Macintosh runs a similar system, where each new volume that is found is automatically mounted and added to the desktop when it is found.
- More recent Windows systems allow file systems to be mounted to any directory in the file system, much like UNIX.

6.5. File Sharing

6.5.1. 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 (User), the Group, and to the rest of the world.
- Some systems have more complicated access control, allowing or denying specific accesses to specifically named users or groups.

6.5.2. 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 or 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 WWW has made it easy once again to access files on remote systems without mounting their file systems, generally using (anonymous) ftp as the underlying file transport mechanism.

6.5.2.1. The Client-Server Model

- When one computer system remotely mounts a file system 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
- The same computer can be both a client and a server.
- 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 file systems 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.

6.5.2.2. Distributed Information Systems

- The Domain Name System, DNS, provides for a unique naming system across the entire 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.

6.6. Protection

- Files must be kept safe for reliability and protection.
- The former is usually managed with backup copies.
- 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.

6.6.1. Types of Access

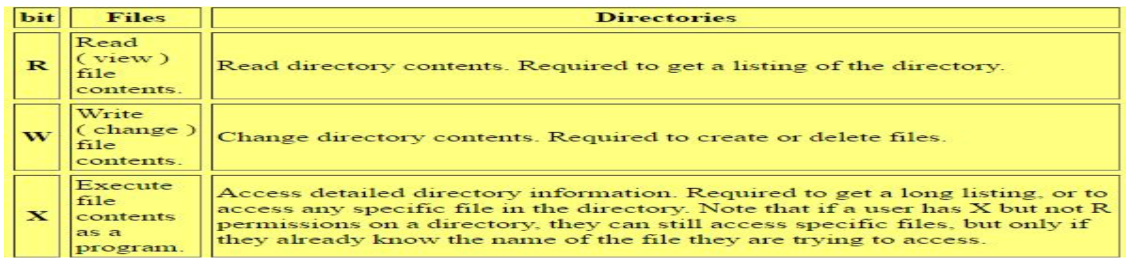
The following low-level operations are often controlled:

- 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.

6.6.2. 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.
- UNIX uses a set of 9 access control bits, in three groups of three.
- These correspond to R, W, and X permissions for each of the Owner, Group, and Others. The RWX bits control the following privileges for ordinary files and directories:



bit	Files	Directories
R	Read (view) file contents.	Read directory contents. Required to get a listing of the directory.
W	Write (change) file contents.	Change directory contents. Required to create or delete files.
X	Execute file contents as a program.	Access detailed directory information. Required to get a long listing, or to access any specific file in the directory. Note that if a user has X but not R permissions on a directory, they can still access specific files, but only if they already know the name of the file they are trying to access.

In addition there are some special bits that can also be applied:

- The set **user ID (SUID) bit** and/or the set group ID (SGID) bits applied to executable files temporarily change the identity of whoever runs the program to match that of the owner / group of the executable program.
- The **sticky bit** on a directory modifies write permission, allowing users to only delete files for which they are the owner.
    - This allows everyone to create files in /tmp, for example, but to only delete files which they have created, and not anyone else's.

- The SUID, SGID, and sticky bits are indicated with an S, S, and T in the positions for executes permission for the user, group, and others, respectively.
    - If the letter is lower case, (s, s, t), then the corresponding execute permission is not also given.
    - If it is upper case, (S, S, T), then the corresponding execute permission is given.

6.6.3. Other Protection Approaches and Issues

- Some systems can apply passwords, either to individual files, or to specific sub-directories, or to the entire system.
- There is a trade-off between the number of passwords that must be maintained and the amount of information that is vulnerable to a lost or forgotten password.
- Access to a file requires access to all the files along its path as well.
- In a cyclic directory structure, users may have different access to the same file accessed through different paths.

UNIT-VI
Assignment-Cum-Tutorial Questions
SECTION-A

Objective Questions

1. ______ is a unique tag, usually a number, identifies the file within the file system. [ ]
   a) File identifier b) File name c) File type d)None of the mentioned
2. Reliability of files can be increased by : [ ]
   a) keeping the files safely in the memory
   b) making a different partition for the files
   c) by keeping them in external storage
   d) by keeping duplicate copies of the file
3. The main problem with access control lists is : [ ]
   a) their maintenance
   b) their length
   c) their permissions
   d) all of the mentioned
4. Many systems recognize three classifications of users in connection with each file (to condense the access control list) : [ ]
   a) Owner b) Group c) Universe d) All of the mentioned
5. To create a file [ ]
   a) allocate the space in file system
   b) make an entry for new file in director
   c) allocate the space in file system & make an entry for new file in directory
   d) none of the mentioned
6. File type can be represented by [ ]
   b) file name c) file extension
   c) file identifier d) none of the mentioned
7. What is the mounting of file system? [ ]
   a) crating of a file system
   b) deleting a file system

c) attaching portion of the file system into a directory structure

d) removing portion of the file system into a directory structure

8. Which one of the following explains the sequential file access method?

a) random access according to the given byte number [ ]

b) read bytes one at a time, in order

c) read/write sequentially by record

d) read/write randomly by record

9. Sequential access method ______ on random access devices.

a) works well [ ]

b) doesnt work well

c) maybe works well and doesnt work well

d) none of the mentioned

10. The direct access method is based on a ______ model of a file, as _____ allow random access to any file block. [ ]

a) magnetic tape, magnetic tapes c) tape, tapes

b) disk, disks d) all of the mentioned

11. For a direct access file : [ ]

a) there are restrictions on the order of reading and writing

b) there are no restrictions on the order of reading and writing

c) access is restricted permission wise

d) access is not restricted permission wise

12. A relative block number is an index relative to :

a) the beginning of the file [ ]

b) the end of the file

c) the last written position in file

d) none of the mentioned

13. For large files, when the index itself becomes too large to be kept in memory : [ ]

a) index is called

b) an index is created for the index file

c) secondary index files are created

d) all of the mentioned

14. The directory can be viewed as a _________ that translates file names into their directory entries. [ ]

    a) symbol table b) partition c) swap space d) cache

15. In the single level directory : [ ]

    a) All files are contained in different directories all at the same level

    b) All files are contained in the same directory

    c) Depends on the operating system

    d) None of the mentioned

16. In the two level directory structure : [ ]

    a) each user has his/her own user file directory

    b) the system doesn't its own master file directory

    c) all of the mentioned

    d) none of the mentioned

17. The disadvantage of the two level directory structure is that :

    a) it does not solve the name collision problem [ ]

    b) it solves the name collision problem

    c) it does not isolate users from one another

    d) it isolates users from one another

18. In the tree structured directories: [ ]

    a) the tree has the stem directory

    b) the tree has the leaf directory

    c) the tree has the root directory

    d) all of the mentioned

19. Path names can be of two types : [ ]

    a) absolute & relative c) local & global

    b) global & relative d) relative & local

20. When keeping a list of all the links/references to a file, and the list is empty, implies that : [ ]

    a) the file has no copies c) the file is deleted

    b) the file is hidden d) none of the mentioned

SECTION-B

Descriptive Questions

1. Explain different directory structures.
2. What are the operations that can be performed on a file?
3. How Access to files is controlled?
4. What is direct access method for files?
5. Explain various file accessing methods.
6. Write about single level and two level directory Structures.
7. What is a File? Explain about Files Sharing and Protection.
8. Discuss about the Single level directory structure.
9. Discuss about the two level directory structure.
10. Explain about different file attributes?
11. Briefly explain about file system mounting?
12. Explain about file system protection?