2 Introduction to File System

2.1 FILE CONCEPT :

Definition 1 :

"A file is a named collection of related information that is recorded on secondary storage."

Definition 2 :

"A file is a collection of data that normally stored on a secondary storage device such as a hard disk or floppy diskette."

File System is a method for storing and organizing computer files and the data that contain, to make it easy to find and access the data.

It may use a data storage device such as hard disk or CD-ROM and involve maintaining the physical location of the files.

From a user's perspective, a file is the smallest allotment of logical secondary storage; that is, data can not be written to secondary storage unless they are within a file.

Mostly File System make use of an underlying data storage device that offers access to an array of fixed-size blocks called Sectors, generally of 512 bytes each.

They typically have directories which associates filename with files.

Commonly, files represents programs (both source and object forms) and data.

Data files may be numeric, alphabetic, alphanumeric, or binary. Files may be free form, such as text files, or may be formatted rigidly.

In general, a file is a sequence of bits, bytes, lines, or records, the meaning of which is defined by the files creator and user.

The information in a file is defined by its creator. *Many different types information may be stored in a file - source programs, object programs, executable programs, numeric data, text, payroll records, graphics images, sound recordings, and so on.*

A file has a certained defined **structure**, which depends 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 declaration 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.

FILE ATTRIBUTES :

A file's attributes vary from one operating system to another but typically consist of these:

- **Name**

  The symbolic file name is the only information kept in human readable.

- **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**

  This information is needed for systems that support different types of files.

- ***Location***

  This information pointer to a device and to the location of the file 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.

- **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.

2.2 OPERATIONS ON FILE :

A file can be considered as an **abstract** data type. An operating system must provide a number of operations associated with files so that users can safely store and retrieve data. The following **abstract operations** can be performed on files :

- **Creating a file :**

  First, a space in the file system must be found for the file. Second, an entry for the new file must be made in the directory. The directory entry records the name of the file and the location in the file system.

- **Writing a file :**

  A system call is used to write a file. We provide a file identifier and the information to be written. The operating system performs the write. Operating system must maintain the write pointer in the file.

- **Reading a file :**

  To read a file, a system call is made that specifies that specifies the name of the file and where (in memory) the next block of the file should be put. Again, the directory is searched for the associated directory entry, and the directory will need a pointer to the next block to be read. Once the block is read, the pointer is updated.

- **Repositioning within a file :**
  The directory is searched 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 files seek.
- **Deleting a file :**
  The operating system must provide the facility to delete a file. It frees the space allocated to the file and removes the file entry from the directory, so that it can be reused by other files.
- **Truncating A File :**
  Truncating a file means erasing the contents of the file but keeping the directory entry. The file will effectively be empty.
- **Appending A File :**
  This includes appending new information to the end of the existing file.
- **Renaming A File :**
  This operation includes renaming the existing file.

FILE TYPES:

File types refer to classifying the content of the file, such as a program, text file, executable program or data file.

In Windows operating systems, the file type is derived from the filename extension. Typical file types and their extensions are

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run 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, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

Windows associates applications (programs) with specific file types. For example, the default application that opens to process a file of type .txt is the Notepad editor.

2.3 FILE ACCESS METHODS :

- ✓ Files store information when it is use, this information must be access and read into computer memory.
- ✓ There are several ways that the information in the file can be access some system provide only one access method for file on other system uses many different access method.
- ✓ There are following access method.
  1. Sequential Access Method
  2. Direct Access / Random Access Method
  3. Other Access Method

1. Sequential Access Method :

- ✓ This is the simplest access method. The information in the file is process in order, one record after other.
- ✓ This model of access is by far the most common, for example Editor and Compiler usually access files in these fashion.

✓ The bulk of the operations on a file are reads and writes. A read operation reads the next portion of the file and automatically advances a file pointer, which tracks the I/O location. Similarly a write appends to the end of the files and advances to the end of the newly written material (the new end of file).

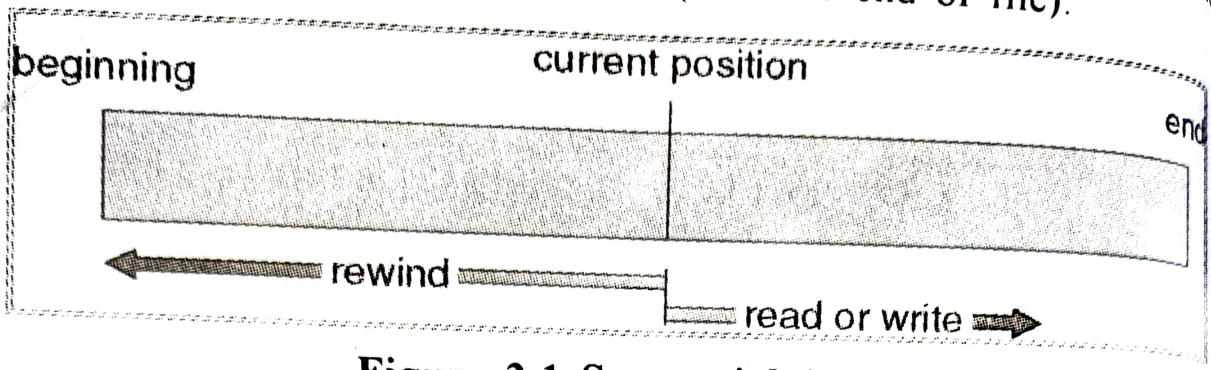

Figure 2.1 Sequential Access

✓ Sequential access is based on the TAPE MODEL of a file, and work as well as on sequential access devices.

2. Direct Access / Random Access Method :

✓ In this method, information in a file is read no particular order.

✓ The direct access method is based on a disk model of a file.

✓ For direct access, the file is viewed as a numbered sequence of blocks or records. A direct-access file allows arbitrary blocks to be read or written. After block 18 has been read, block 57 could be next and then block 3. There are no restrictions on the order of reading and writing for a direct access file. Direct access files are of great use for intermediate access to large amounts of information. There is no restriction on the order of reading or writing for a direct access file.

✓ Database applications are often of this type.

✓ The file operations must be modified to include the block number as a parameter. It works like "read n", where n is the block number rather than "read next". Similarly, it writes with "write n" rather that "write next".

✓ An alternative "approach retains "read next" and "write next". It adds an operation "position file to n" where n is the block number. Then we would issue the command "position to n" and then "read next".

Sequential Access	Implementation for Direct Access
Reset	cp = 0;
read next	read cp; cp = cp + 1;
write next	write cp; cp = cp + 1;

Figure 2.2

3. Other Access Method :

- Other access methods can be built on top of a direct-access method. These additional methods generally involve the construction of an index for a file. The index contains pointers to the various blocks. To find an entry in the file, the index is searched first and the pointer is then used to access the file directly to find the desired entry.
- With a large file, the index 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 that would point to the actual data items.
- For example, IBM's indexed sequential access method (ISAM) uses a small master index that points to disk blocks of a secondary index. The secondary index blocks point to the actual file blocks. The file is kept sorted on a defined key. We first make a binary search of the master index to find a particular item. It provides the block number of the secondary index. This block is read in. Binary search is used again 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.

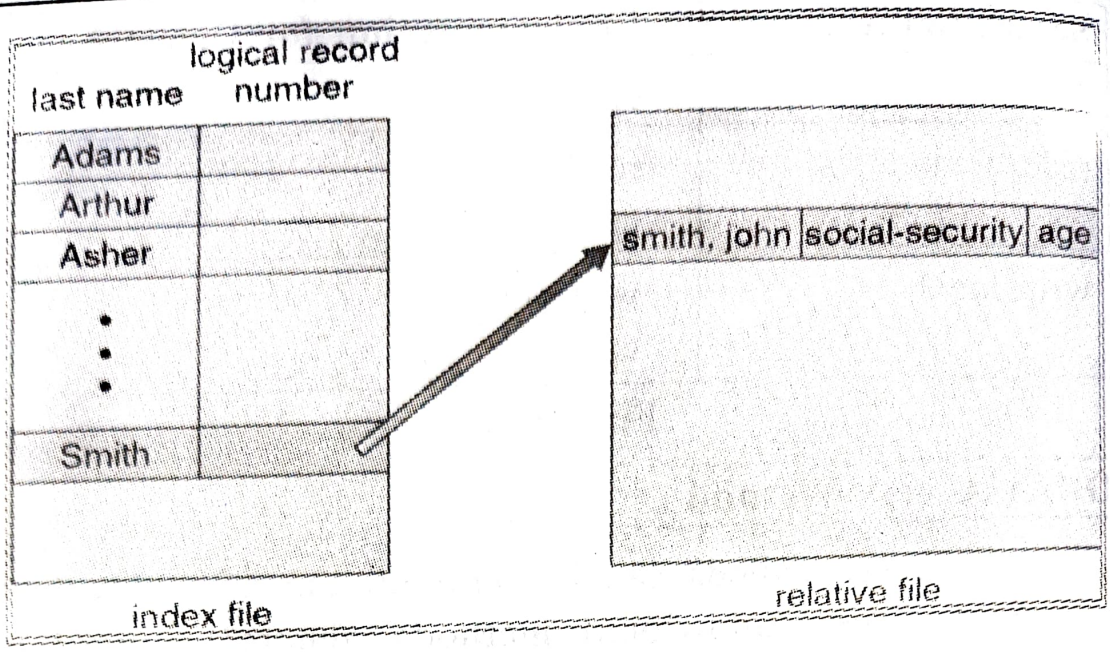

Figure 2.3 Index Access

2.4 FILE DIRECTORY SYSTEMS :

In order to store many files, we need some organization in the way we store and reference them. Commonly, there are several levels of abstraction.

First is that disks are broken-up into partitions. Each partition is logically treated as a separate hard drive. A disk can have many partitions. There is usually at least one partition on every hard drive.

Each partition (being a logical hard drive) has a directory that maintains a list of files in the directory.

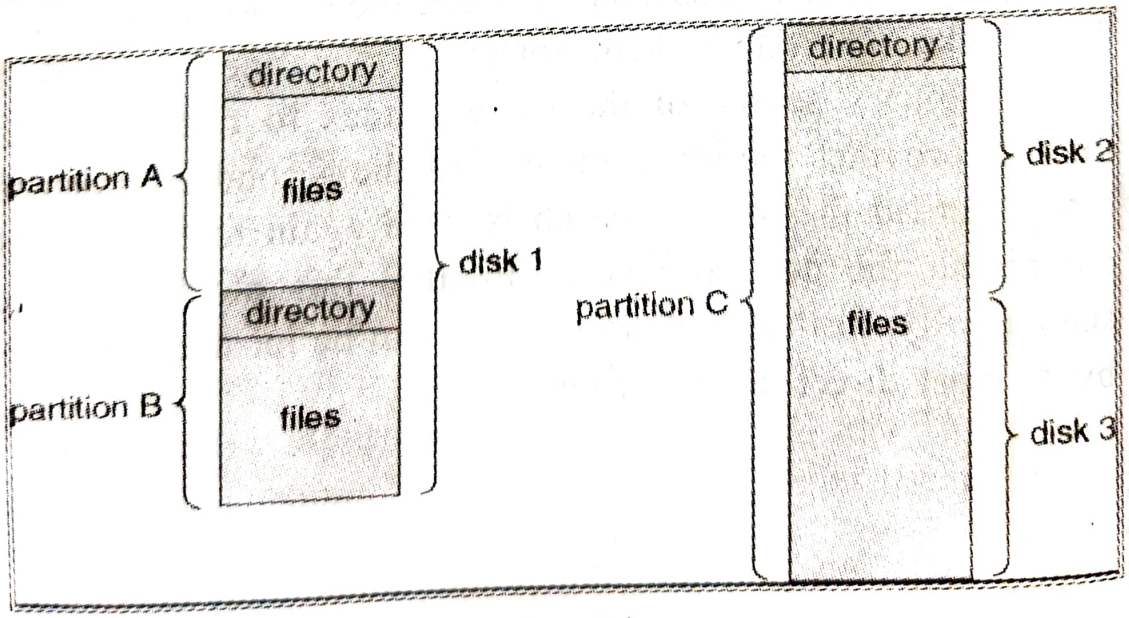

Figure 2.4 Directory

A collection of nodes containing information about all files. Both the directory structure and the files reside on disk. Backups of these two structures are kept on tapes

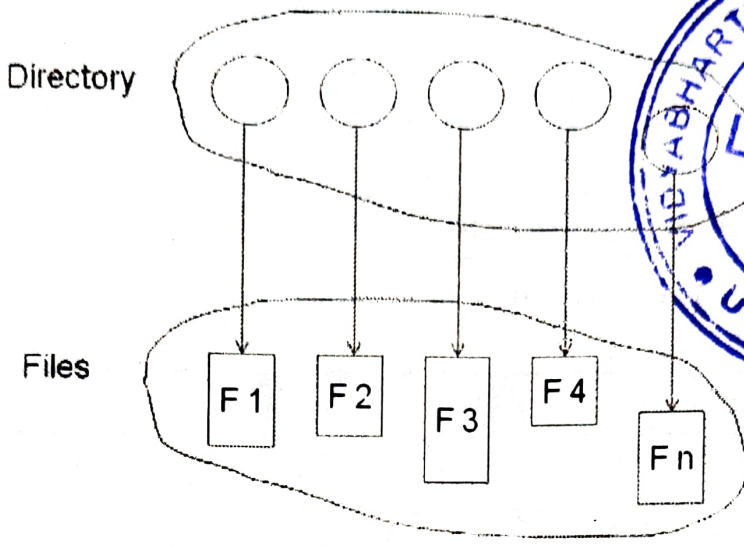

Figure 2.5

OPERATION ON DIRECTORY :

1. **Search for Files :**

   We need to be able to search directory structure to find the entry for a particular file. Since file have symbolic name and similar name may include a relationship between a file, we may want be able to find all files whose name match a particular patterns.

2. **Create a File :**

   We need to new file to be created and added to the directory.

3. **Delete a File :**

   When a file is no longer needed we want to remove it from the directory.

4. **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 the list.

5. **Renaming A Directory :**

   Because the name of the file represents it contain to its user. The name must be changeable when the content or use of the file in this. Renaming the file may also allow its position within the directory structure to be change.

6. Traversing A File System :

It is useful to be able to access every directory, and every file within a directory structure. For reliability; it is a good idea to save the content and structure of an entry file system at a regular entry. This saving often consists of copying all files in magnetic tape. This technique provides back up copy in case of system failure or if the file is simply no longer in use.

LEVELS OF DIRECTORY STRUCTURE :

1. Single Level Directory :

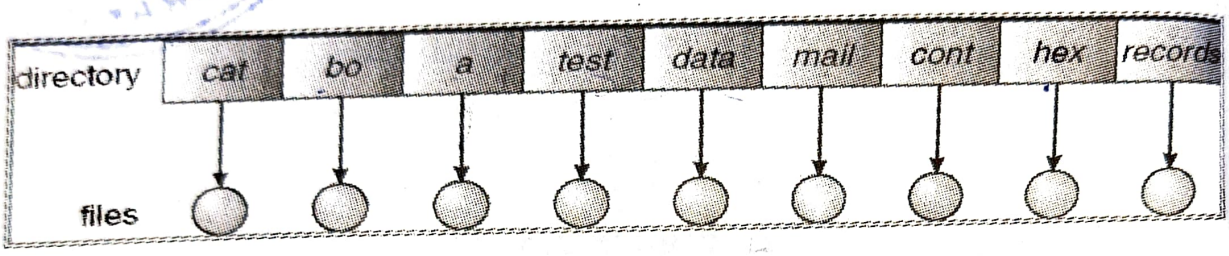

Figure 2.6 Single Level Directories

- ✓ In a single-level directory system, all the files are placed in one directory. This is very common on single-user OS. So, the simplest directory structure is the single level.
- ✓ A single-level directory has significant limitations when the number of files increases or when there is more than one user. Since all files are in the same directory, they must have unique names. If there are two users who call their data file "test", then the unique-name rule is violated. Although file names are generally selected to reflect the content of the file, they are often quite limited in length.
- ✓ Even with a single-user, as the number of files increases, it becomes difficult to remember the names of all the files to create files with unique names.
- ✓ It is not uncommon for a user to have hundreds of file on one system on equal number of addition file on another system. In such an environment, keeping tacks of so many files is difficult task.

2. Two Level Directory :

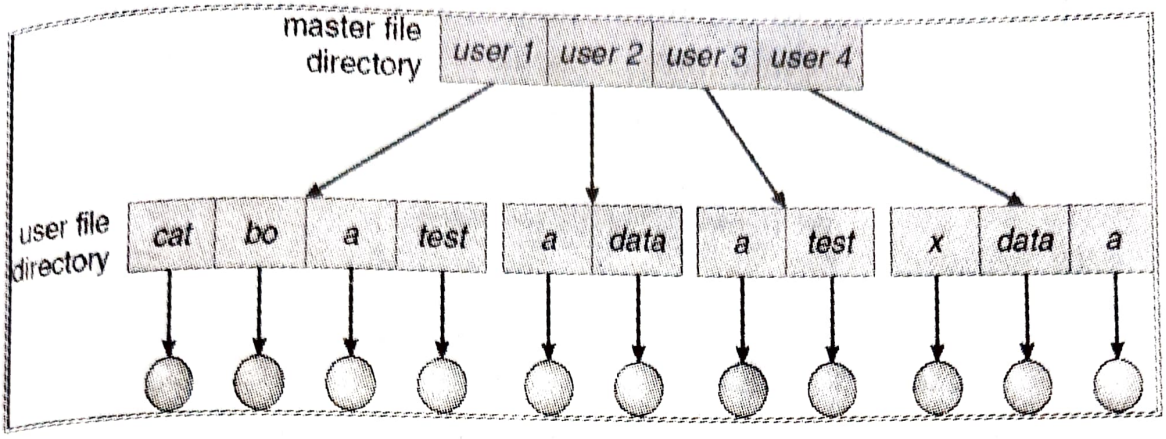

Figure 2.7 Two Level Directories

- ✓ It solves some of the naming issues and perhaps has higher capacity.
- ✓ It can separate multiple users but cannot allow them shared access to files. It is not extendible in a general sense.
- ✓ In the two level directory structures, each user has his/her own UFD (User File Directory). Each UFD has a similar structure, but list only the files of single users.
- ✓ When a user refers to particular files, only his UFD is search. So different user may have files with the same name, as long as all the file names within each UFD are unique.
- ✓ To create a file for a user, the Operating System searches only that user UFD to ascertain whether another file of that name exists. To delete a file, the Operating System confines its search to the local UFD, so it cannot accidentally delete another user file that has the same name.
- ✓ The user directory themselves must be created and delete as necessary.
- ✓ Two level directories are not much better but they provide a bit more abstraction. For example, we can have several users having their own set of single-level directories.

3. Tree Level Directory :

- ✓ This is also called a hierarchical file system. More than two levels of directories are possible.

- ✓ In the tree-structured directory, the directories themselves files. This leads to the possibility of having sub-directories can contain files and sub-subdirectories.
- ✓ Each file contains a unique file path.

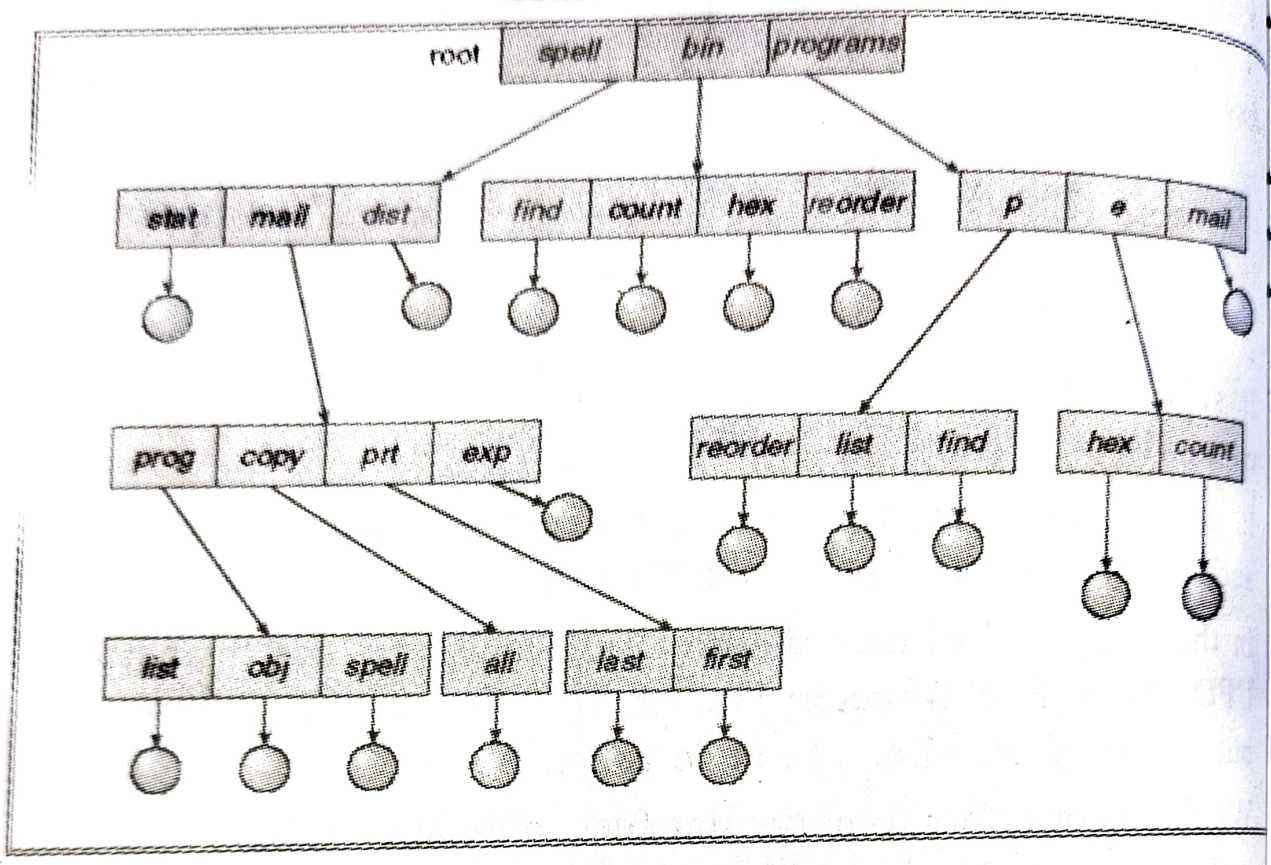

Figure 2.8 Tree Level Directories

- ✓ An interesting policy decision in a tree-structured directo structure is how to handle the deletion of a directory. If directory is empty, its entry in its containing directory c simply be deleted. Suppose the directory to be deleted is n empty, but contains several files, or possibly sub-directories.
- ✓ Some systems will not delete a directory unless it is empty. T user must first delete all the files in a directory to delete th directory. If there are any subdirectories, this procedure must applied recursively to them so that, they can be deleted als This approach may result in a heavy amount of work.
- ✓ An alternative approach is just to assume that, when a request made to delete a directory, all of that directory's files a sub-directories are also to be deleted.

4. Acyclic-Graph Directory :

- ✓ The acyclic directory structure is an extension of the tree-structured directory structure.
- ✓ It permits users to create shared files and directories.
- ✓ Shared files and directories can be created using 'link' operation. This operation allows a file or a directory to appear in more than one directory.
- ✓ Directories can contain files as well as sub-directories also.
- ✓ A file may contain more than one file path.
- ✓ This directory structure does not contain cycles.

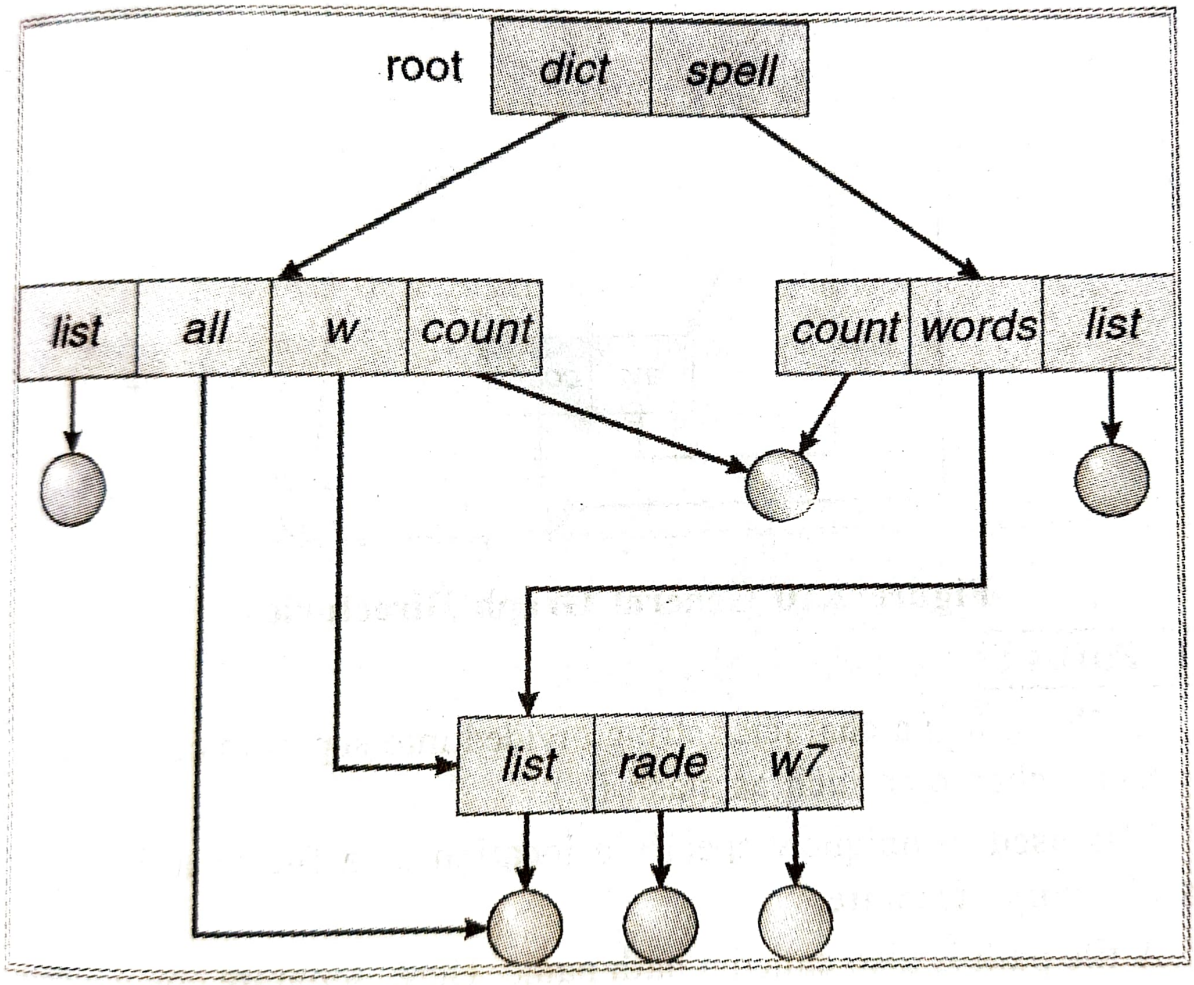

Figure 2.9 Acyclic Graph Directory

5. General Graph Directory :

- ✓ Acyclic directories suffer from the fact that they cannot have any cycles. In acyclic directories we are intentionally restricting the structure not to have any cycles. This means that every time

we create a link file, we need to check and ensure that no cycle has been created. This is not an easy task.

✓ General Graph directories allow cycles in the graph. The algorithms that traverse graphs and acyclic graphs are quite different. For a graph with possible cycles in it, we need to maintain a list of nodes we have already visited in order not to visit them again.

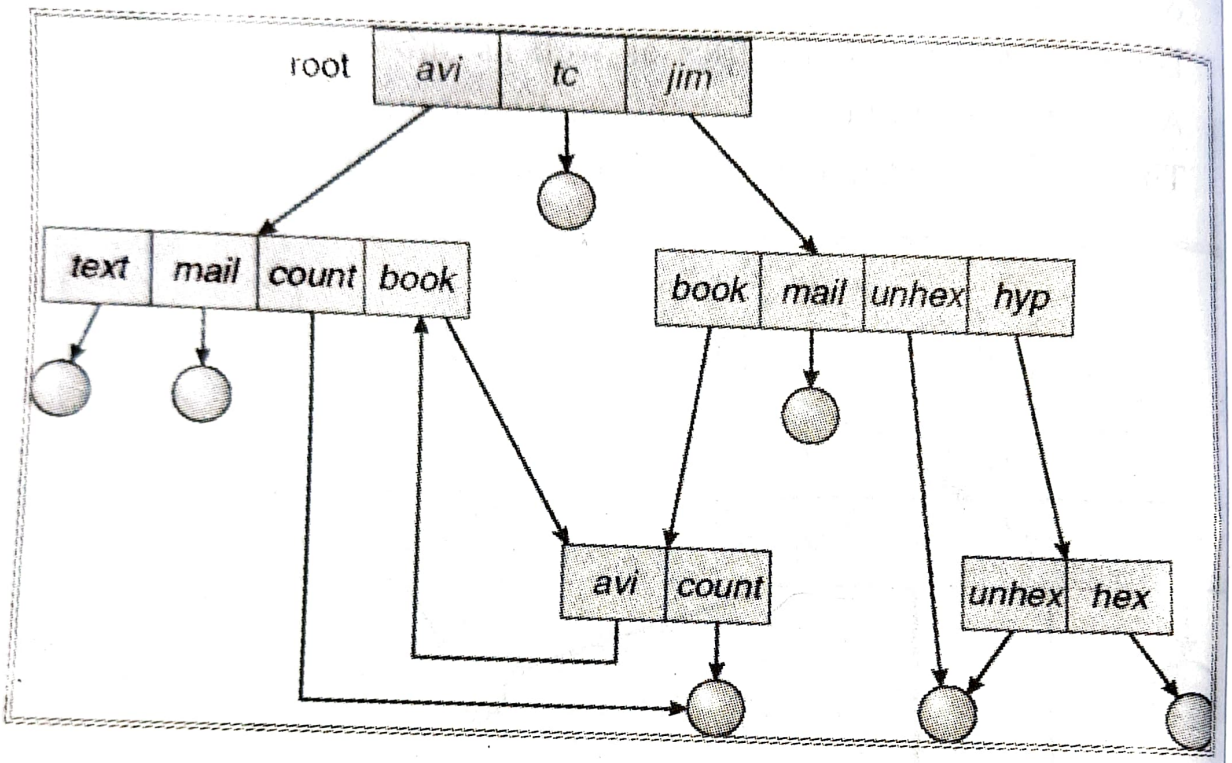

Figure 2.10 General Graph Directories

File Paths :

A file path is a character string divided into separate components by special character such as slash ('/').

It is used to uniquely specify a location of a file or a directory in a directory structure.

A file path is also called a path name. Here, separate components are directories; except that the last component can be a file also.

There are two different methods to specify file paths :

1. **Absolute Path :**

✓ It starts with the root directory of a file system. Here, file paths are given with respect to the root directory.

Relative Path :

It starts with the current directory designated by a user. Here, file paths are given with respect to such current directory.

EXERCISE

Q-1. Short Questions :

1. What is File ?
2. List out File attributes.
3. What is Source File ?
4. What is Object File ?
5. What is Executable File ? Give example of two binary executable file.
6. What is File Path ? Give different types of file path.

Q-2. Long Questions :

1. Explain Operation on File.
2. Write a note on File Access Method.
3. Explain File Directory Structure Level.

■ ■ ■