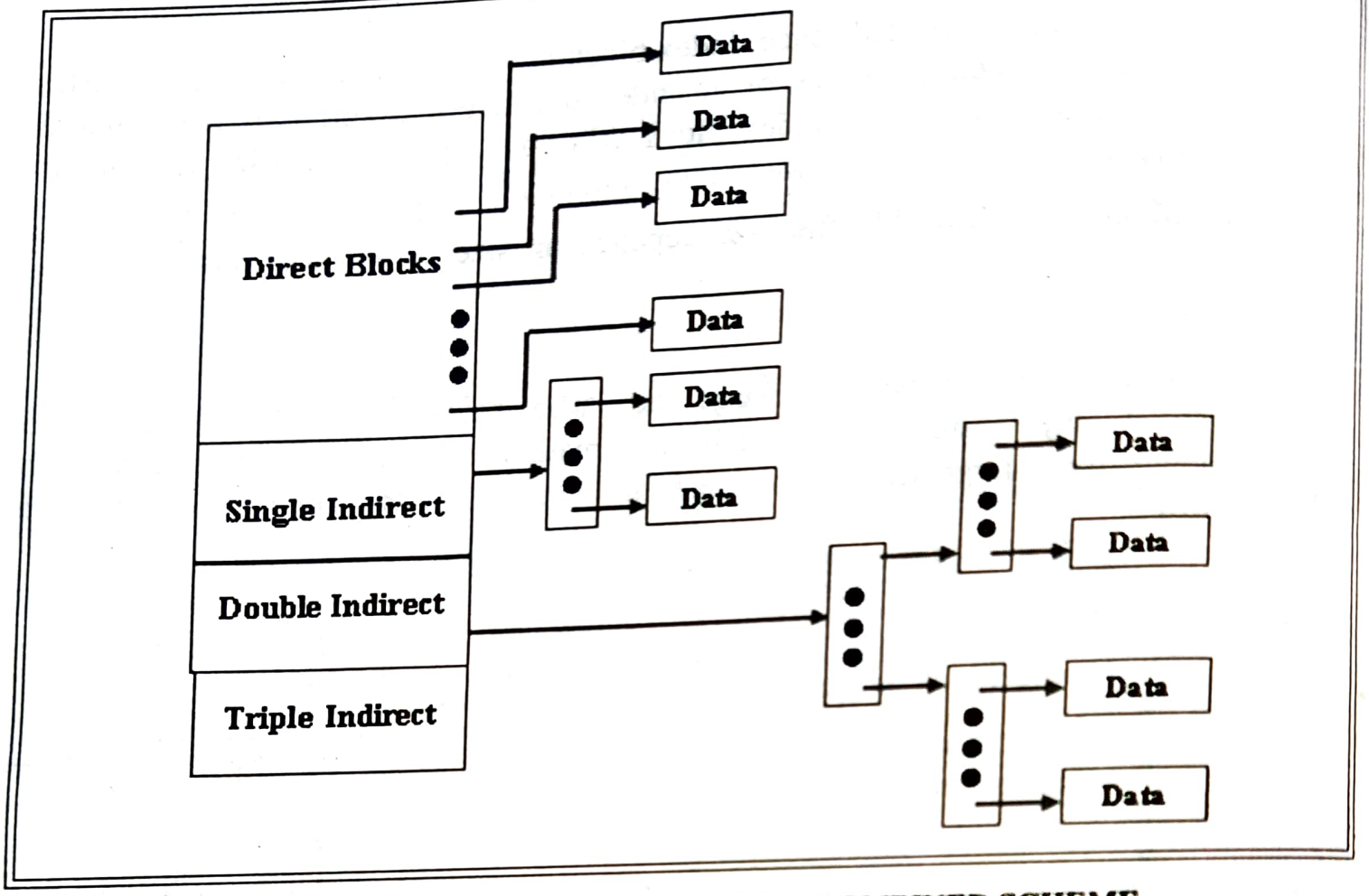

FIGURE-4.17 : INDEXED ALLOCATION: COMBINED SCHEME

4.7 File Protection

Files are used to store information. Depending upon the users and applications, this information can be very important. Consider the importance of various records stored in files in some bank. What will be the consequences if this information gets lost, or damaged or corrupted ?

So, in short, no one wants to loose information stored in files in any way. Every one wants information stored in files to be safe.

Here, safety comes in two ways: 1. Reliability, and 2. Protection.

1. **Reliability :**
   - Reliability means safety from physical damage.
   - File systems can be damaged by hardware problems, fire, dirt, power failures, head crashes and so on. Files may be deleted accidentally (or even intentionally too...!!!).

- Reliability is generally provided by keeping more than one copies of files, means duplicating files.
- Files are duplicated mostly on some different locations, or in different devices. Magnetic tapes are most widely used for backups of large files.
- Operation of duplicating files (copying files) is generally done at regular time intervals. This can be done ones per day or week or even month depending upon the importance of the information kept in files. This can be done automatically using system software or manually.
- If original files are damage (unfortunately), then they can be retrieved back from the backups.

2. Protection :

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

These two ways are described below :

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

2. **Access Control :**
   - Here, a list, called an access control list (ACL), is associated with each file. This list specifies the user name and type of access allowed for that user. This information is stored for all the possible users of the system.
   - This list is generally kept in directory entry along with file name and other information.
   - When user requests access to a file, the operating system checks the ACL associated with that file. If that user is listed for the requested access, only then access is allowed. Else, access is denied.
   - This method also suffers from some problems.
   - First problem with this method is: Constructing an ACL is tedious task. Also, all users of the system are not known in advance.
   - Second problem with this method is: ACL is kept in directory entry. So, now directory entry should be of varying length instead of fixed length. This results in more complicated space management.
   - Solution to this problem is to divide all users in various categories, and then to allow access on such categories instead of individual users.
   - UNIX uses such type of scheme.
   - It divides users in three categories: **Owner**, **Group** and **Others**. Access is given on such category. All the users of that category will be allowed that access. (This scheme is described in detail in chapter 6).

4.4.3 Directory Structures

"Directory structures refer to the way how directories and files are organized." There are five different types of directory structures available. All of them have their own advantages and disadvantages. These directory structures are given as below:

1. Single-Level Directory
2. Two-Level Directory
3. Tree structured Directories
4. Acyclic Graph Directory
5. General Graph Directory

The description about each of these directory structures is given as below:

1. Single-Level Directory :

- This is the simplest directory structure.
- All files are contained in the same directory. Sometimes it is called the root directory.
- Directory can contain only files. (It cannot have other sub-directories.)
- This type of structure was used in early personal computers. Even first supercomputer, the CDC 6600, used this type of directory structure for simplicity.
- The following figure 4.6 represents this directory structure. In this figure, directory is represented using rectangle, while files are represented using circles. Here, single directory contains nine different files named cat, bo, a, test, data, mail, cont, hex and records. (Remember: names written in rectangle are file names.)

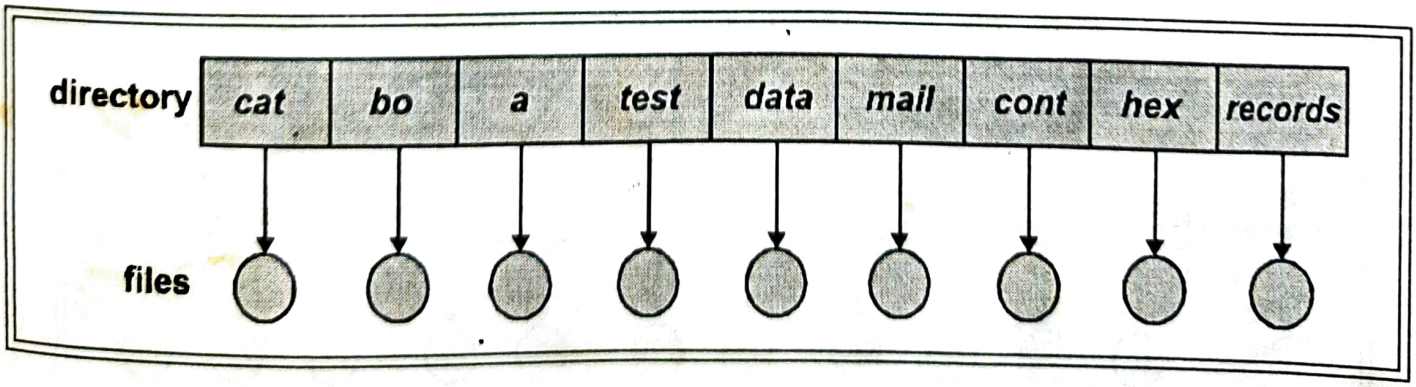

FIGURE-4.6 : SINGLE-LEVEL DIRECTORY

- **Advantages :**
  - (i) This type of structure is very simple.
  - (ii) As there is only one directory, searching a file is very quick.

- **Disadvantages :**
  - (i) It is not suitable for multi-user systems. Different users may provide same file name for different files, possibly overwriting other's files. This is called a file name collision.
  - (ii) It is even not suitable for single user system when number of files becomes too large. User cannot remember the names of all files.
  - (iii) Different files cannot be grouped.

2. Two-Level Directory :

- It gives each user a private directory.
- This is to avoid file name collisions among different users.
- Two levels of directories are used here: First one is a root directory or master file directory (MFD); and second are user file directories (UFD) or simply, user directories.
- User directories can contain only files. They cannot have sub-directories.
- The following figure 4.7 represents this directory structure. In this figure, directories are represented using rectangles, while files are represented using circles.
- Here, root directory contains four subdirectories related to four different users named user 1, user 2, user 3 and user 4. 'user 1' directory contains four different files named cat, bo, a and test. In a similar way, directories 'user 2', 'user 3' and 'user 4' contain two, two, and three files respectively.

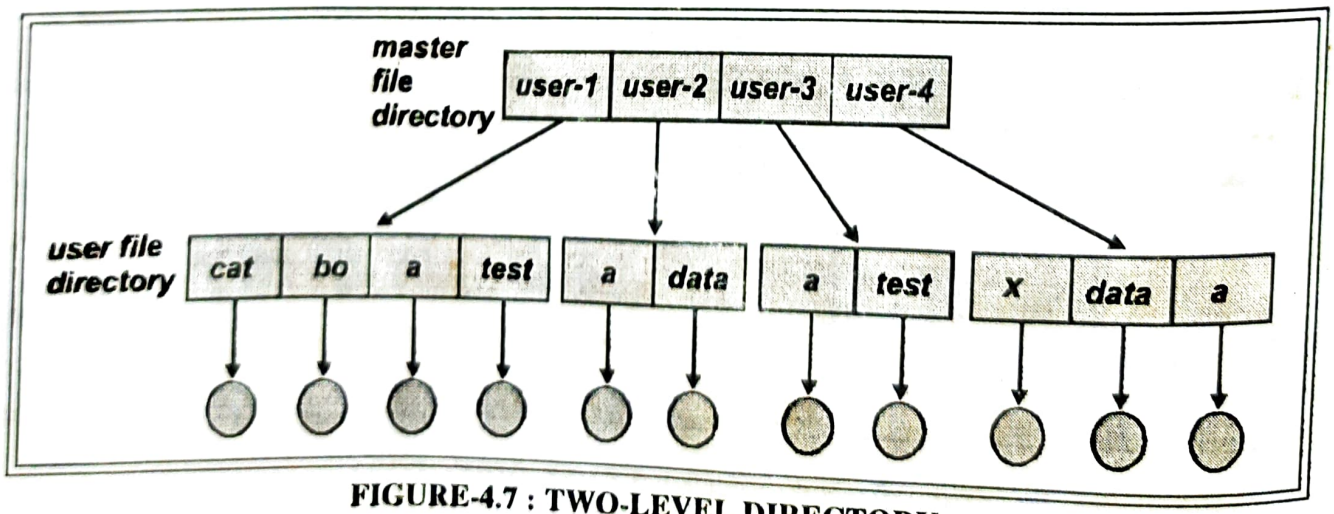

FIGURE-4.7 : TWO-LEVEL DIRECTORY

- **Advantages :**
  - (i) It avoids file name collisions.
  - (ii) It can be used in multi-user systems.
  - (iii) Searching is efficient here; as user has to search in its own single directory.
- **Disadvantages :**
  - (i) It is not suitable for users having large number of files.
  - (ii) Different files cannot be grouped.
  - (iii) One extra system directory is required to store system files.

3. Tree Structured Directories :

- It permits users to create own sub-directories.
- This is also called a hierarchical file system. More than two levels of directories are possible.
- Directories can contain files as well as sub-directories also.
- Each file contains a unique file path. (A file path is described in section 4.4.4).
- The following figure 4.8 represents this directory structure. In this figure, directories are represented using rectangles, while files are represented using circles. The file path for a file named 'spell' is also given in figure.

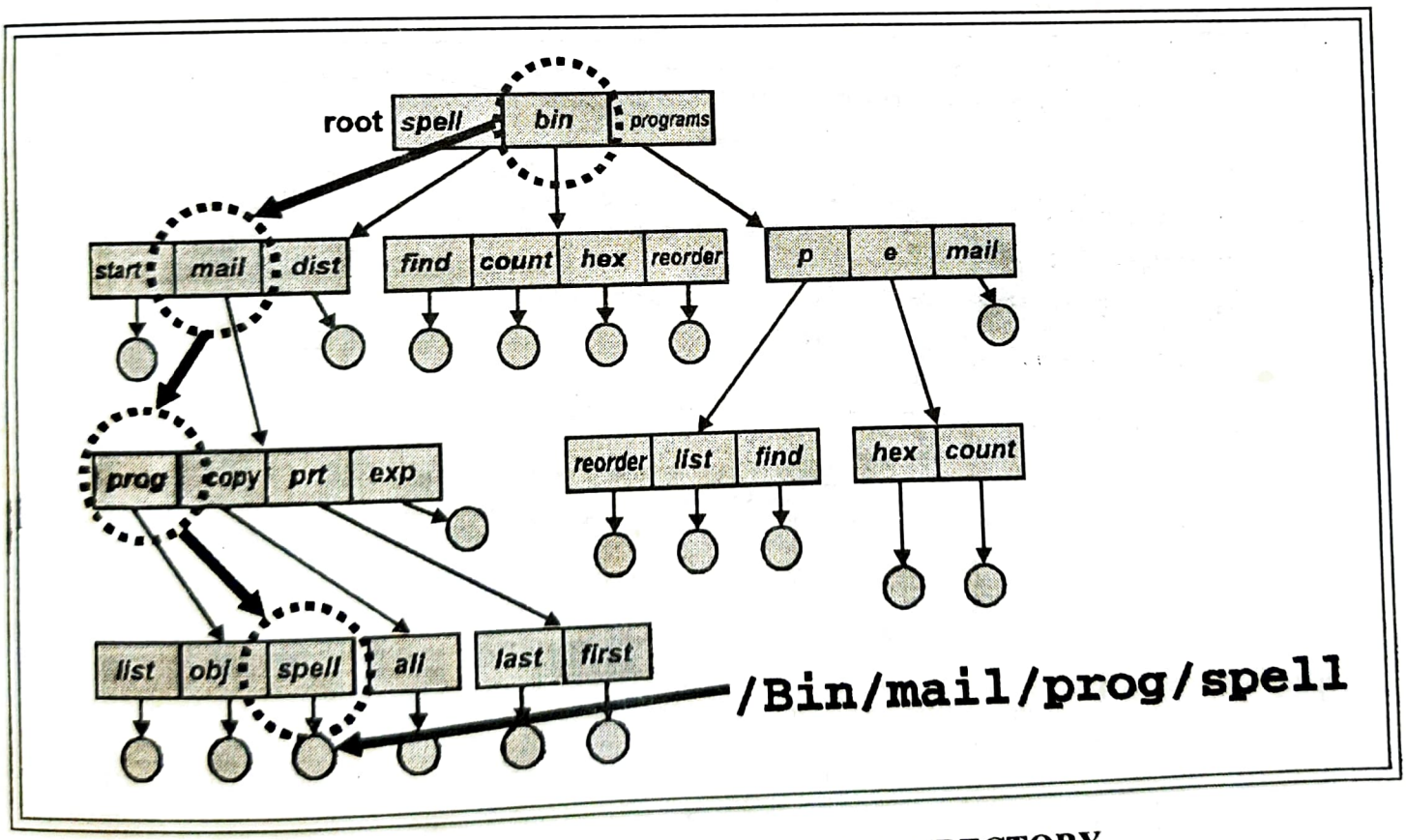

FIGURE-4.8 : TREE STRUCTURED DIRECTORY

- **Advantages :**
  - **(i)** It avoids file name collisions.
  - **(ii)** It can be used in multi-user systems.
  - **(iii)** Different files can be grouped here.
- **Disadvantage :**
  - **(i)** Sharing of files and directories is not possible here.

4. Acyclic Graph Directory :

- 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. (A file path is described in section 4.4.4).
- This directory structure does not contain cycles.
- The following figure 4.9 represents this directory structure. In this figure, directories are represented using rectangles, while files are represented using circles. Also figure shows that a file named 'count' is shared from two different locations (directories) 'dict' and 'spell'.

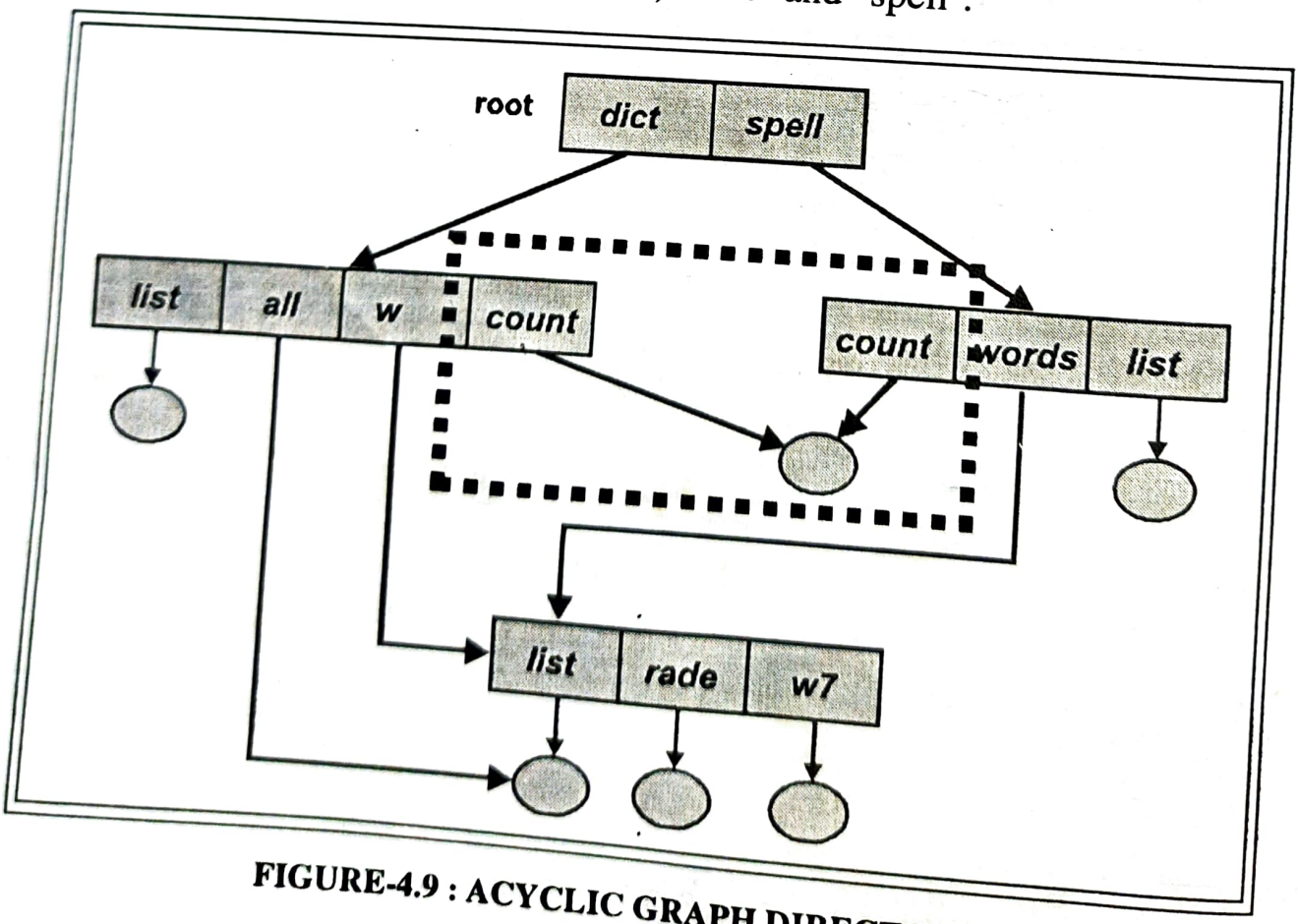

FIGURE-4.9 : ACYCLIC GRAPH DIRECTORY

- **Advantages :**
  - (i) It allows sharing of files and directories.
- **Disadvantage :**
  - (i) Deletion of shared file or directory is complex. When such file/directory is deleted, directory entry from all related directories should be deleted.
  - (ii) A file may have multiple file paths. This may create problem in some operations. For example, if user copies all files to backup storage, then such files will be copied multiple times.
  - (iii) It is difficult to ensure that there are no any cycles in a directory structure.

5. General Graph Directory :

- It is same as acyclic graph directory, but, it also allows cycles in a directory structure.
- The following figure 4.10 represents this directory structure. In this figure, directories are represented using squares, while files are represented using circles.

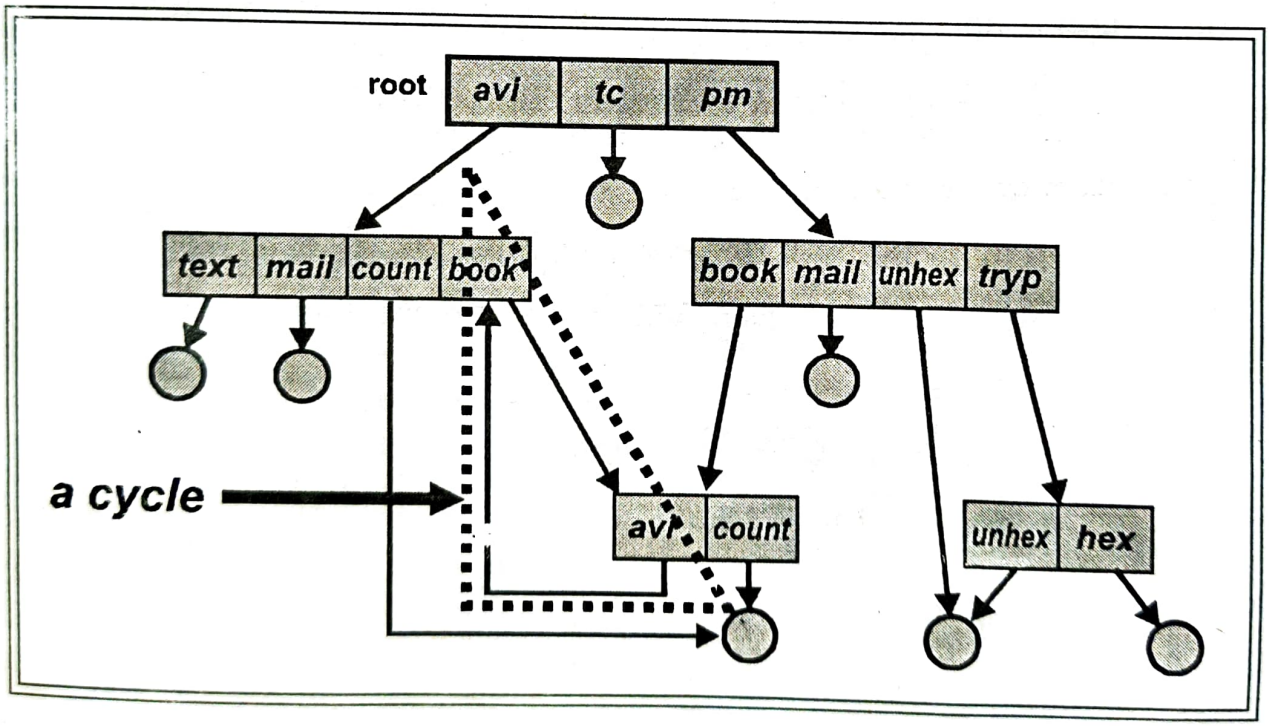

FIGURE-4.10 : GENERAL GRAPH DIRECTORY

- **Advantages :**
  - (i) It allows sharing of files and directories.
- **Disadvantage :**
  - (i) Searching should be done carefully. Else, it may get into infinite loops.