Figure 10.3 Indexed sequential file

10.4 File directories

The directory contains information about the files , including attributes , locations and ownership. Some times the directories consisting of subdirectories also. The directory is it self a file, owned by the operating system and accessible by various file management routines.

Directory structure

Some times the file system consisting of millions of files, at that situation it is very hard to manage the files. To manage these files grouped these files and load one group in to one partition. Each partition is called a directory. The directory can be viewed as a symbol table that translate filenames in to their directory entries. A directory structure provides a mechanism for organizing many files in the file system.

Operations on the file directories:

The operations that can be performed on a directory are as follows.

Search for a file:	Search the directory structure for required file.
Create a file:	Whenever we create a file , should make an entry in the directory.
Delete a file:	When a file is no longer needed , we want to remove it from the directory.
List a directory:	We can know the list of files in the directory.
Rename a file:	whenever we need to change the name of the file we can change the name.
Traverse the file system:	We need to access every directory , and every file with in a directory structure we can traverse the file system.

Single level directory system

It is simplest of all directory structures, in this the directory system having only one directory, it consisting of the all files. Some times it is said to be ' Root directory'. For example consider the figure 10.6. here is directory contains 4 files (A,B,C,D)

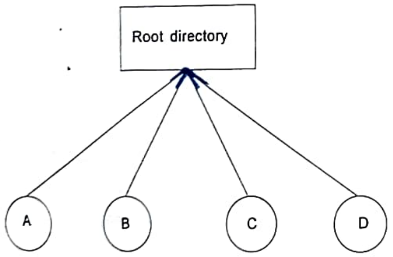


Figure 10.6 Single level directory

The advantage of this scheme are its simplicity and the ability to locate files quickly. The problem with single level directory is different users may accidentally use the same names for their files., for example , if user1 creates a file called sample, and then later user2 creates a file called sample, then user2's file will overwrite A's file. That 's why it is not used in the multi-user system , it is used on small embedded system.

Two-level directory structure

The problem in single level directory is different users may be accidentally use the same names for their files. To avoid this problem each user need a private directory. In this way names chosen by one user don't interfere with names chosen by a different user and there is no problem caused by the same occurring in two or more directories. Consider the figure 10.7 for better understanding.

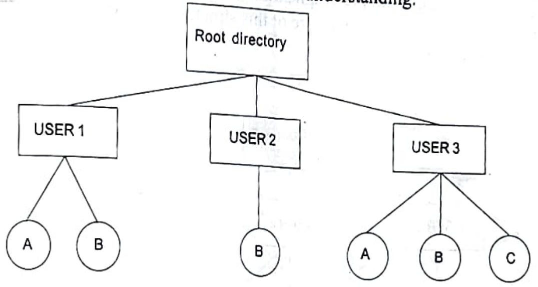


Figure 10.7 A two level directory system

Here root directory is the first level directory . it consisting of entries of ' user directory. User1, user2,user3 are the user levels of directories. A,B,C are the files.

Hierarchical directory system

The two-level directory eliminates name conflicts among users but it is not satisfactory for users with a large number of files. To avoid this creates the subdirectory and load the same type of files in to the subdirectory. So in this method each can have as many directories are needed. Consider the figure 10.8 for better understanding.

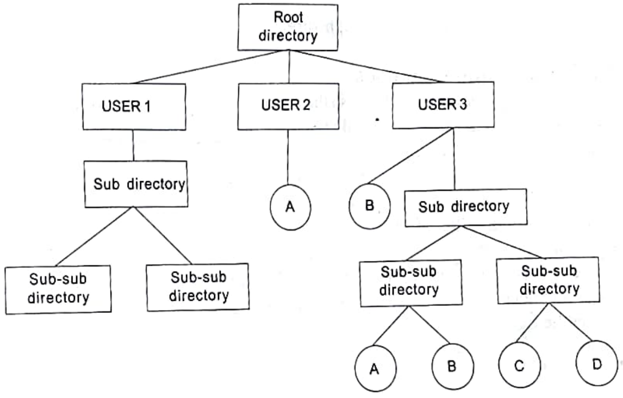


Figure 10.8 A hierarchical directory system

This directory structure looks like tree, that's why it is also said to be tree-level directory structure'.

General graph directory structure

When we add links to an existing tree structured directory , the tree structure is destroyed, resulting in a simple graph structure. Consider the figure 10.9 for better understanding. The primary advantage of this structure is traversing is easy and file sharing also possible.

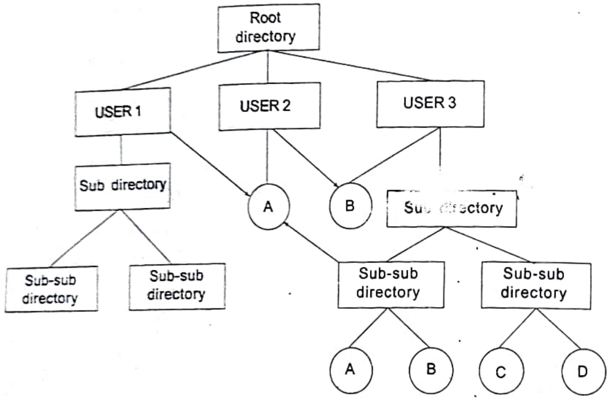


Figure 10.9 Graph directory structure.

10.5 File allocation methods

Files are normally stored on the disks, so the main problem is how to allocate space to these files so that disk space is utilized effectively and files can be accessed quickly. Three major methods of allocating disk space are in wide use: contiguous, linked and indexed. Each method has its advantages and disadvantages.

10.5.1 Contiguous allocation

In this allocation method each file occupies a set of contiguous blocks on the disk. For example a disk consisting of 1kb blocks. A100 kb file would be allocated 100 consecutive blocks. With 2kb blocks, it would be allocated 50 consecutive blocks. Consider the figure 10.10 for better understanding.

File allocation table

File name	Start name	length
A	21	5
B	3	10
C	35	7

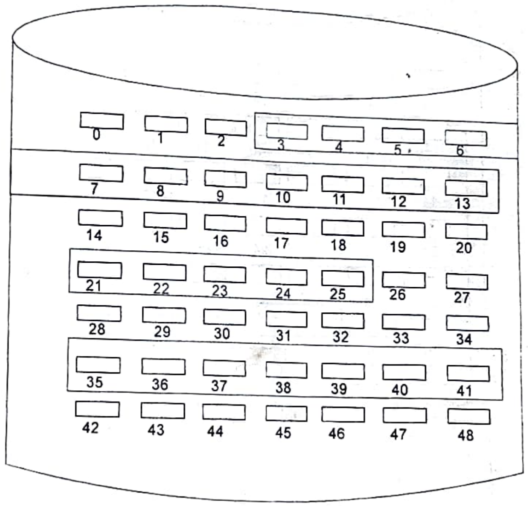
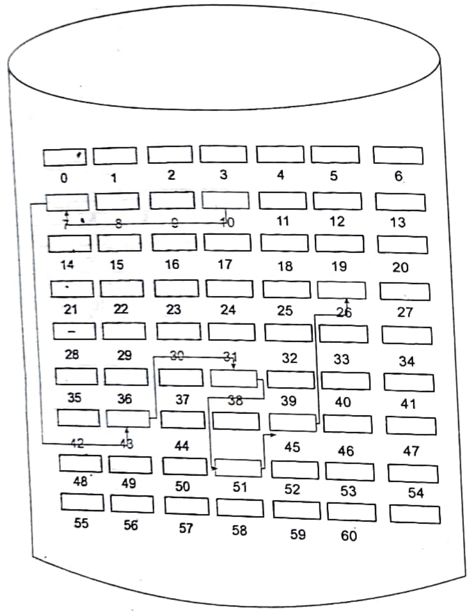


Figure 10.10 : Contigous File allocation

In the figure the right hand side part is the file allocating table' it consisting of a single entry for each file. It shows the file names starting block of the file and size of the file. For example file B size is 10 blocks., the file B stored from 3 to 13 blocks continuously. This method is best suited for sequential files. The main problem in this is it is difficult to find the contiguous free blocks in the disk. Another problem is external fragmentation, it means some free blocks could happen between two files.

10.5.2 Linked allocation

We can avoid the external fragmentation in this scheme. And also it is easy to locate the files , because allocation is on an individual block basis. Each block contains a pointer to the next free block in the chain. Here also the file allocation table consisting of a single entry for each file. Using this method any free block can be added to a chain very easily. There is a link between one block to another block, that's why it is said to be linked allocation. Consider the figure 10.11 for better understand.



File allocation table

File name	Start block	length
Bubble sort	10	7

Figure 10.11 Linked allocation

Advantages:

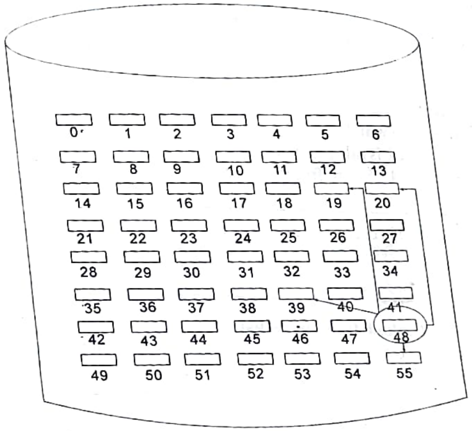
1. Avoid the external fragmentation and compaction.
2. Suited for sequential files.

Disadvantages:

1. The pointer itself occupies some memory with in the block.
2. It takes much accessing time.

10.5.3 Grouped allocation (or) indexed allocation

We can overcome the problems using this method which were faced in the previous methods. In this method the file allocation table contains a single entry for each file. The entry consisting of one index block, the index block having the pointers to the other blocks. Which were occupied by the particular file. Consider the figure 10.12 for better understanding.



File allocation table

File name	Index block
.	.
.	.
Bubble sort	42
...	.

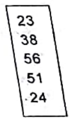


Figure 10.12 : Index allocation

For example bubble sort .c is a file, the entry for the file in the file allocation table points to be indexed block. The index block consisting of pointers to the other blocks. Which were occupied by the particular file. In the figure 23,38,56,51,24 these are the blocks occupied by the bubble sort .c .

Advantages

1. Indexed allocation supports both sequential and direct access files, that's why it is the most popular method.
2. The file indexes are not physically stored as part of the file allocation table.