UNIT 4

File management

File

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

Attributes of the File

1.Name

2.Identifier

3.Type

4.Location

5.Size

6.Time and Date

File Operations:

Creating Files

Opening Files

Reading Files

Writing Files

Renaming Files

Deleting Files

File Types

There are a large number of file types. Each has a particular purpose. The type of a file indicates its use cases, contents, etc. Some common types are:

1. **Media:**

Media files store media data such as images, audio, icons, video, etc. Common extensions: img, mp3, mp4, jpg, png, flac, etc.

2. Programs:

These files store code, markup, commands, scripts, and are usually executable. Common extensions: c, cpp, java, xml, html, css, js, ts, py, sql, etc.

4. Operating System Level:

These files are present with the OS for its internal use. Common extensions: bin, sh, bat, dl, etc.

5. Document:

These files are used for managing office programs such as documents, spreadsheets, etc. Common extensions: xl, doc, docx, pdf, ppt, etc.

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 accessFI
- 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 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

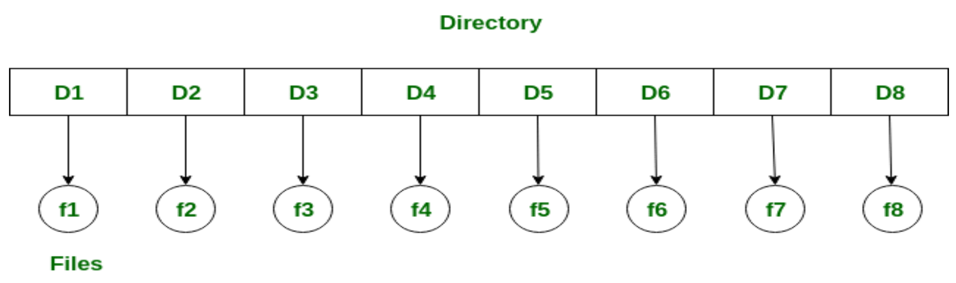
- This mechanism is built up on the basis of sequential access.
- An index is created for each file which contains pointers to various blocks.
- Index is searched sequentially and its pointer is used to access the file directly.

Directory structure

Single-level directory

The single-level directory is the simplest directory structure. In it, all files are contained in the same directory which makes it easy to support and understand.

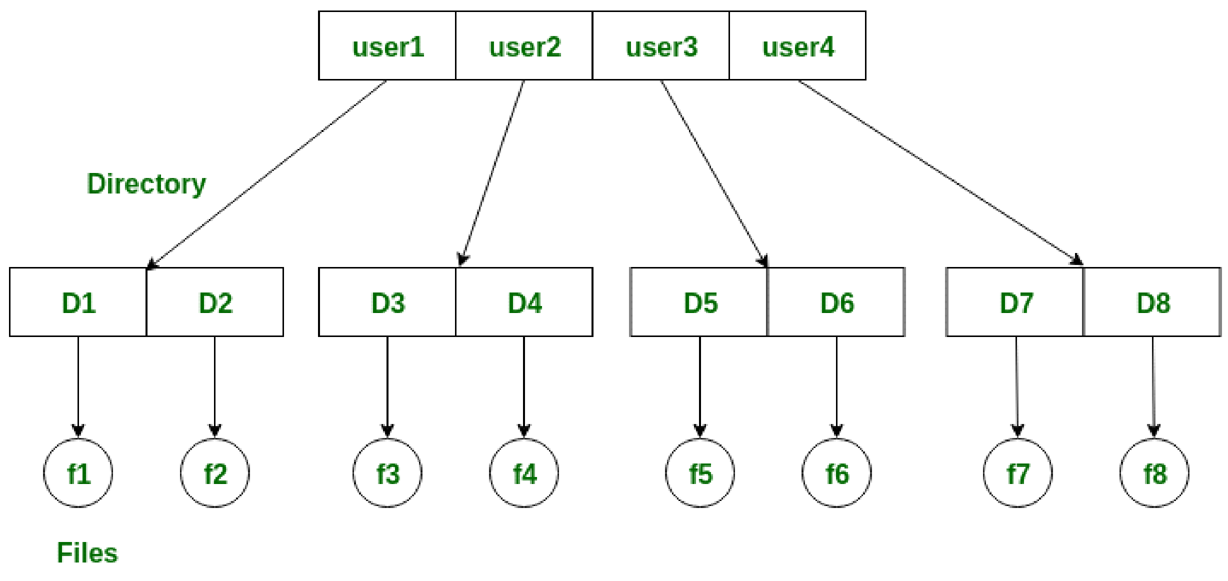
A single level directory has a significant limitation, however, when the number of files increases or when the system has more than one user. Since all the files are in the same directory, they must have a unique name. if two users call their dataset test, then the unique name rule violated.



Two-level directory –

As we have seen, a single level directory often leads to confusion of files names among different users. the solution to this problem is to create a separate directory for each user.

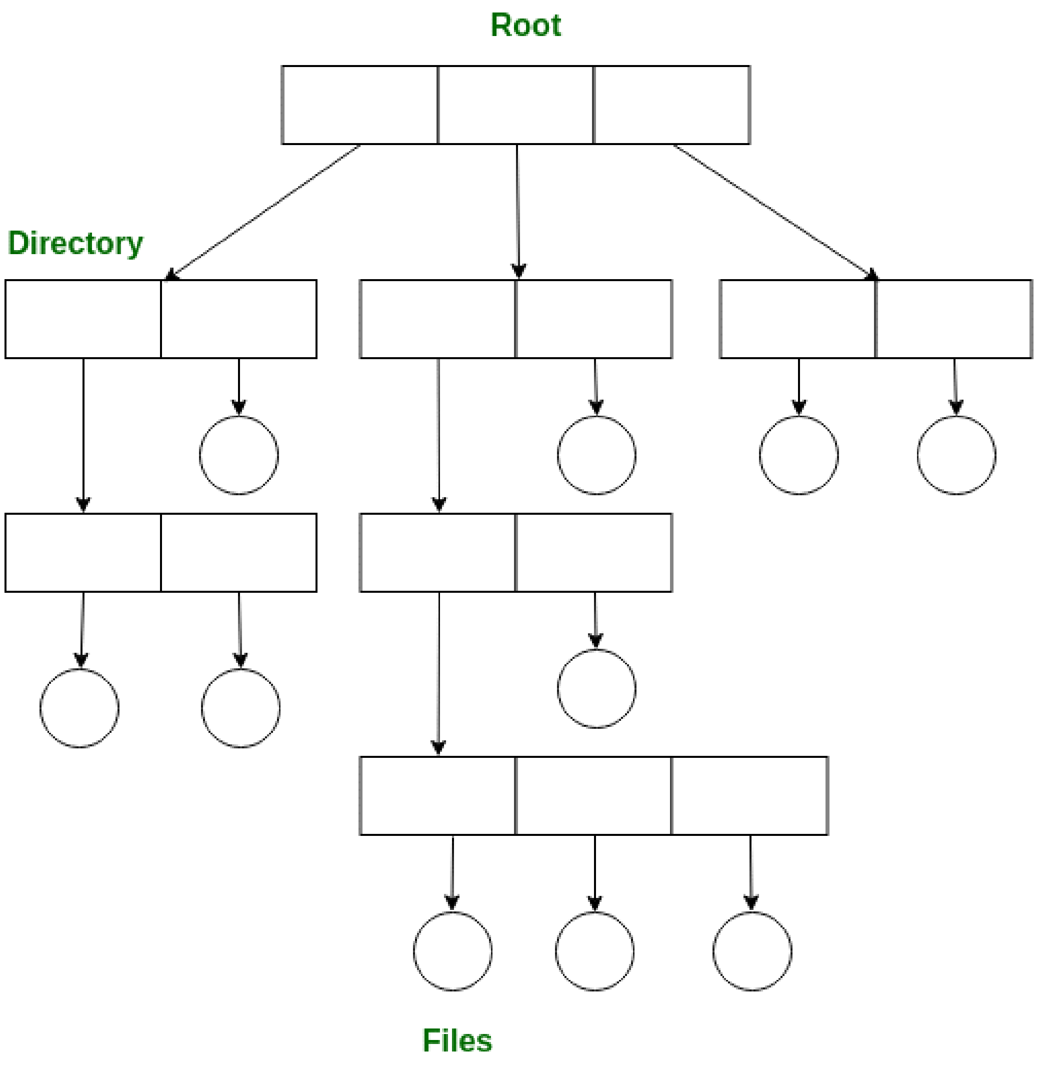
In the two-level directory structure, each user has their own *user files directory (UFD)*. The UFDs have similar structures, but each lists only the files of a single user. system's *master file directory (MFD)* is searches whenever a new user id=s logged in. The MFD is indexed by username or account number, and each entry points to the UFD for that user.



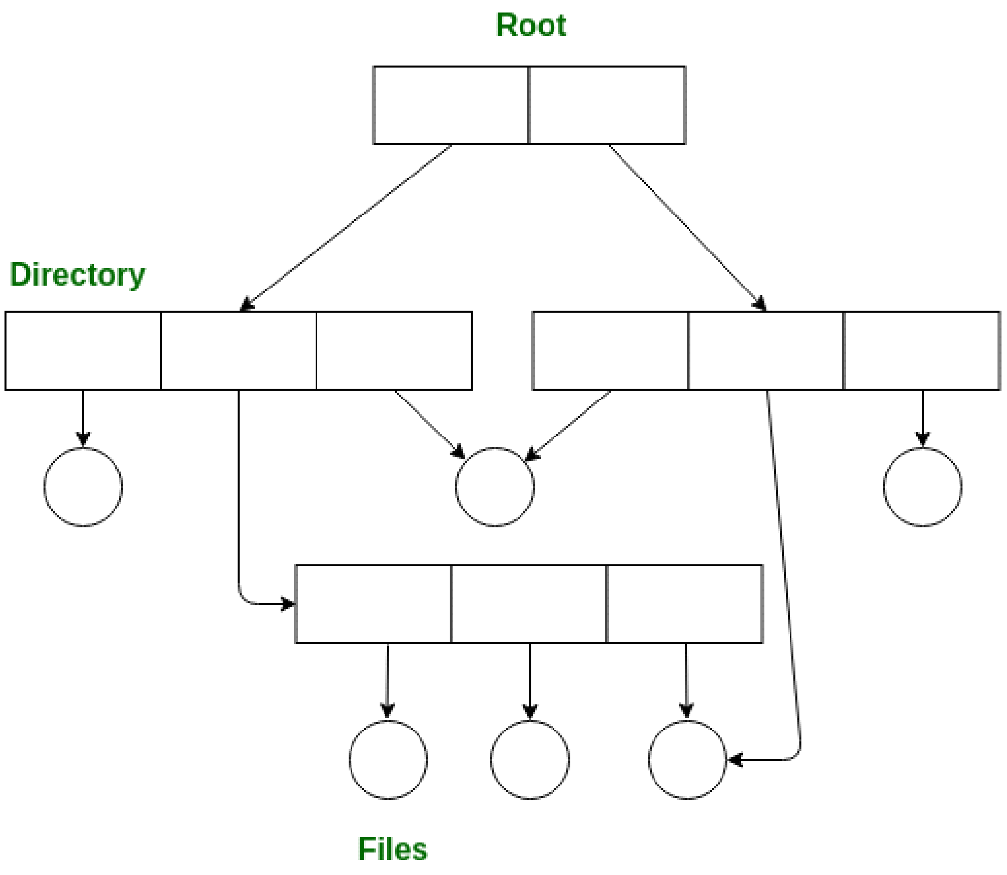
Tree-structured directory –

Once we have seen a two-level directory as a tree of height 2, the natural generalization is to extend the directory structure to a tree of arbitrary height.

This generalization allows the user to create their own subdirectories and to organize their files accordingly.

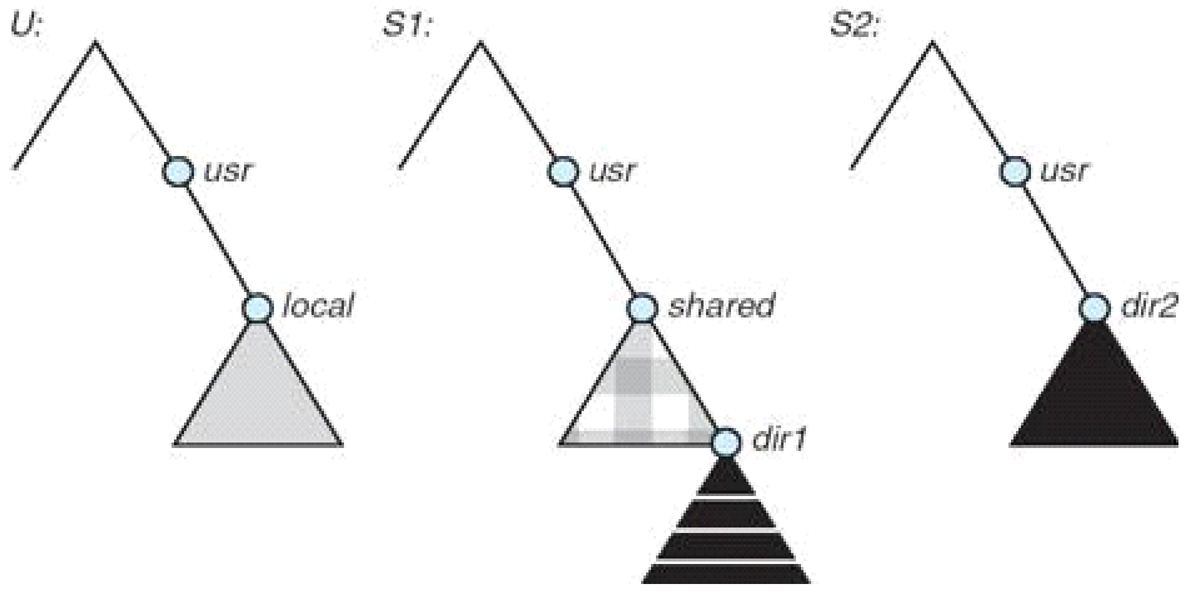


Acyclic graph directory – An acyclic graph is a graph with no cycle and allows us to share subdirectories and files. The same file or subdirectories may be in two different directories. It is a natural generalization of the tree-structured directory.



Mounting in File System:

Mounting refers attaching a root directory from one file system to that of another.



File sharing

File sharing is the practice of sharing or offering access to digital information or resources, including documents, multimedia (audio/video), graphics, computer programs, images and e-books. It is the private or public distribution of data or resources in a network with different levels of sharing privileges.

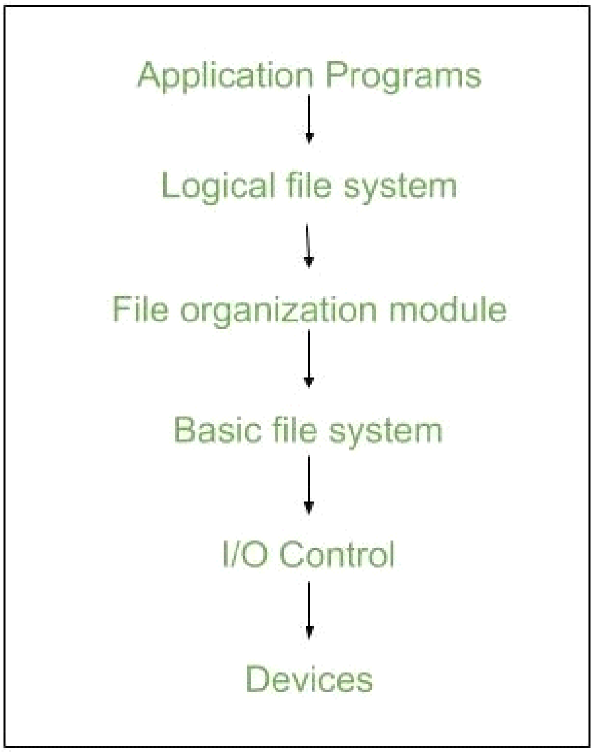
File sharing can be done using several methods. The most common techniques for file storage, distribution and transmission include the following:

- Removable storage devices
- Centralized file hosting server installations on networks
- World Wide Web-oriented hyperlinked documents
- Distributed peer-to-peer networks

- **protection**

  Security refers to providing a protection system to computer system resources such as CPU, memory, disk, software programs and most importantly data/information stored in the computer system. If a computer program is run by an unauthorized user, then he/she may cause severe damage to computer or data stored in it. So a computer system must be protected against unauthorized access

File system structure:



Application programs:program developed by user on file**.**

Logical file system :it checks whether the file is present in directory structure or not.if the file is found it reads the logical block number of file and location**.**

File organization Module :it finds the physical block number of the file.

Basic file system :it issues commands to the device drivers to read/write physical block.

I/O Control level :it accept command and perform operations by interacting with devices.

We can **implement** file system by using two types data structures

- **1. On-disk Structures –**

  Generally they contain information about total number of disk blocks, free disk blocks, location of them, etc. Below given are different on-disk structures :

Boot Control Block –
It is usually the first block of volume and it contains information needed to boot an operating system

Volume Control Block –
It has information about a particular partition

Directory Structure –Used to organize files**.**it contains information about list of files.

Per-File FCB – information about the file,creation,date,time,permissinons,name,type.

- **2. In-Memory Structure :**

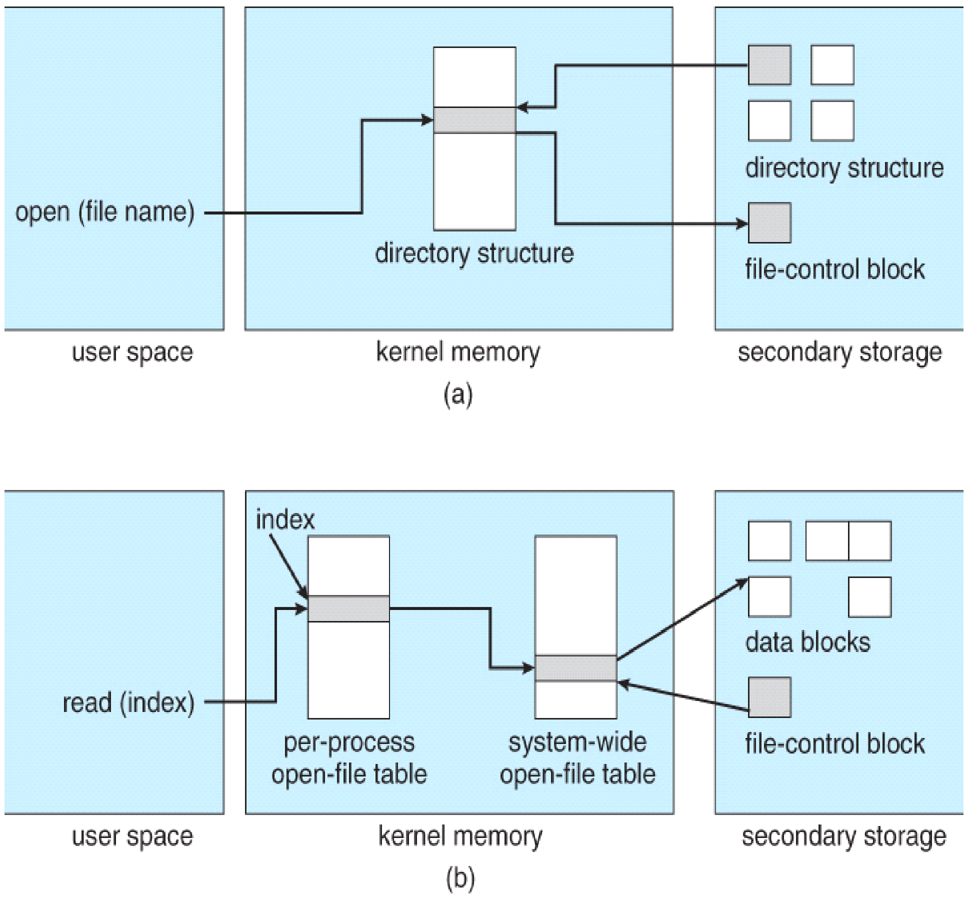
  Mount Table –

  It contains information about each mounted volume.
- **Directory-Structure cache –**

  This cache holds the directory information of recently accessed directories.
- **System wide open-file table –**

  It contains the copy of FCB of each open file.
- **Per-process open-file table –**

  It contains information opened by that particular process and it maps with appropriate system wide open-file



Directory Implementation:

There are mainly two algorithms which are used these days.

1. Linear List

In this algorithm, all the files in a directory are maintained as a singly linked list. Each file contains the pointers to the data blocks which are assigned to it and the next file in the directory.

2. Hash Table

To overcome the drawbacks of singly linked list implementation of directories, there is an alternative approach that is hash table. This approach suggests to use hash table along with the linked lists.

A key-value pair for each file in the directory gets generated and stored in the hash table. The key can be determined by applying the hash function on the file name while the key points to the corresponding file stored in the directory.

Space Allocation

The allocation methods define how the files are stored in the disk blocks. There are three main disk space or file allocation methods.

- Contiguous Allocation
- Linked Allocation
- Indexed Allocation

The main idea behind these methods is to provide:

- Efficient disk space utilization.
- Fast access to the file blocks.

All the three methods have their own advantages and disadvantages as discussed below:

1. Contiguous Allocation

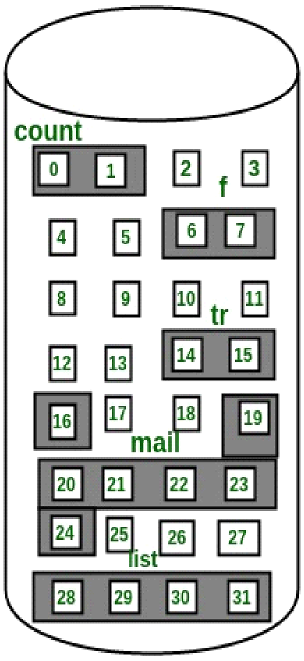
In this scheme, each file occupies a contiguous set of blocks on the disk. For example, if a file requires n blocks and is given a block b as the starting location, then the blocks assigned to the file will be: *b, b+1, b+2,……b+n-1.* This means that given the starting block address and the length of the file (in terms of blocks required), we can determine the blocks occupied by the file.

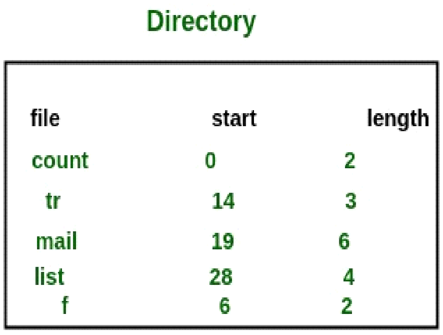
The directory entry for a file with contiguous allocation contains

- Address of starting block
- Length of the allocated portion.

The *file 'mail'* in the following figure starts from the block 19 with length = 6 blocks.

Therefore, it occupies *19, 20, 21, 22, 23, 24* blocks.





Directory

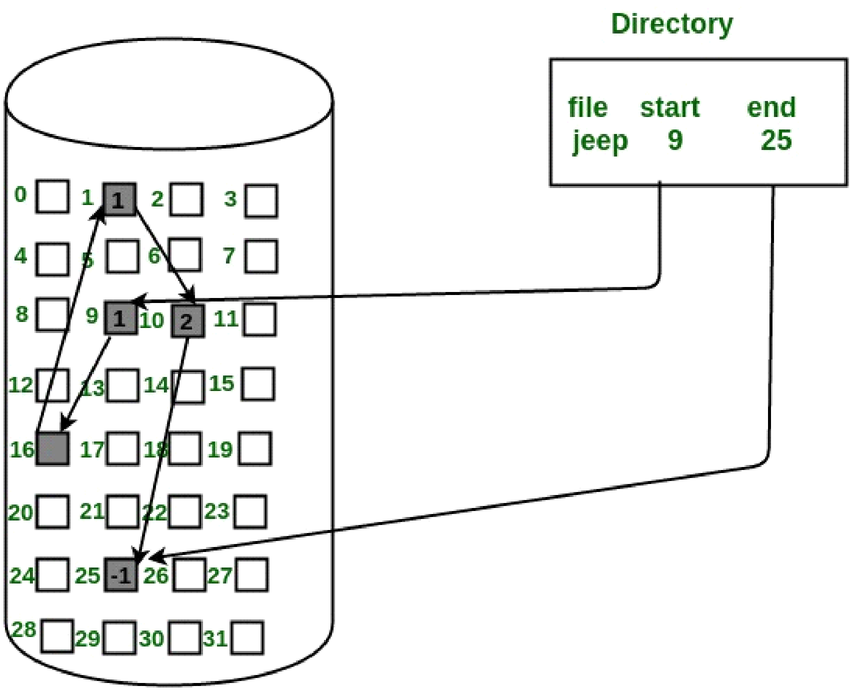
file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

2. Linked List Allocation

In this scheme, each file is a linked list of disk blocks which need not be contiguous. The disk blocks can be scattered anywhere on the disk.

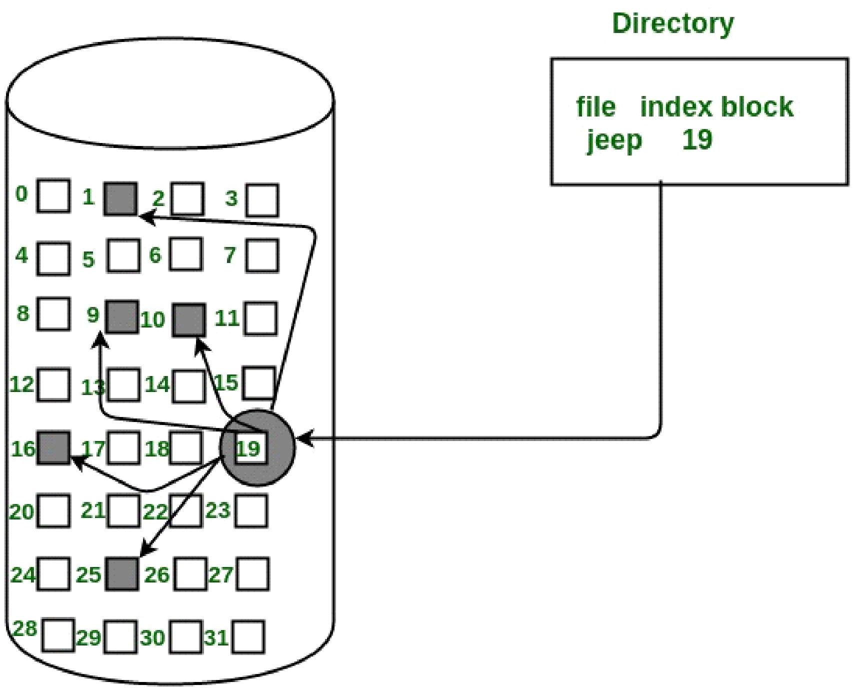
The directory entry contains a pointer to the starting and the ending file block. Each block contains a pointer to the next block occupied by the file.

The file 'jeep' in following image shows how the blocks are randomly distributed. The last block (25) contains -1 indicating a null pointer and does not point to any other block.



3. Indexed Allocation

In this scheme, a special block known as the **Index block** contains the pointers to all the blocks occupied by a file. Each file has its own index block. The ith entry in the index block contains the disk address of the ith file block. The directory entry contains the address of the index block as shown in the image:



Free space management:

Bitmap or Bit vector –

A Bitmap or Bit Vector is series or collection of bits where each bit corresponds to a disk block. The bit can take two values: 0 and 1: *0 indicates that the block is allocated* and 1 indicates a free block.

The given instance of disk blocks on the disk in *Figure 1* (where green blocks are allocated) can be represented by a bitmap of 16 bits as: **0000111000000110**.

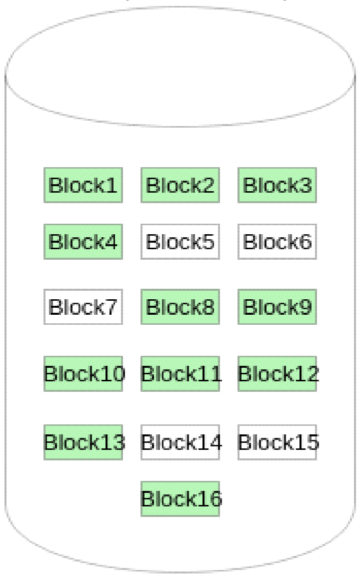


Figure - 1

Linked List –

In this approach, the free disk blocks are linked together i.e. a free block contains a pointer to the next free block. The block number of the very first disk block is stored at a separate location on disk and is also cached in memory.

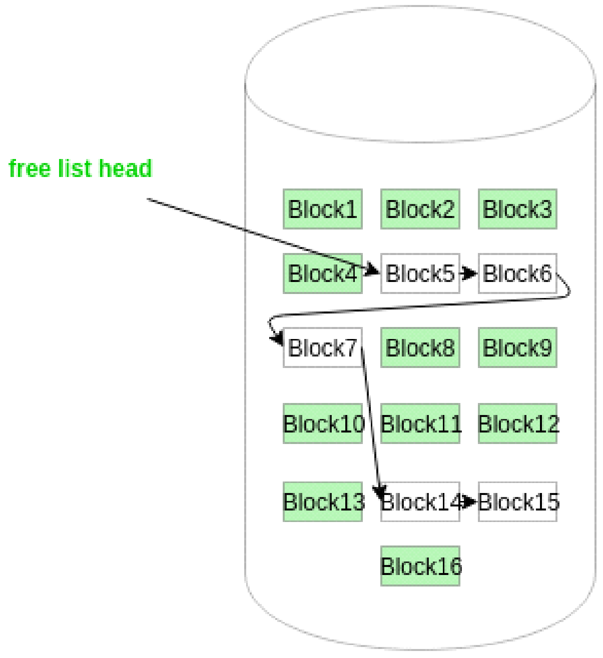


Figure - 2

- **Grouping –**

  This approach stores the address of the free blocks in the first free block. The first free block stores the address of some, say n free blocks. Out of these n blocks, the first n-1 blocks are actually free and the last block contains the address of next free n blocks.

  Counting –

  This approach stores the address of the first free disk block and a number n of free contiguous disk blocks that follow the first block.

  Every entry in the list would contain:

  - Address of first free disk block
  - A number n

Performance & Efficiency

Performance

Even after the basic file-system algorithms have been selected, we can still improve performance in several ways. As will be discussed in Chapter 13, most disk controllers include local memory to form an on-board cache that is large enough to store entire tracks at a time. Once a seek is performed, the track is read into the disk cache starting at the sector under the disk head (reducing latency time). The disk controller then transfers any sector requests to the operating system. Once blocks make it from the disk controller into main memory, the operating system may cache the blocks there. Some systems maintain a separate section of main memory for a buffer cache, where blocks are kept under the assumption that they will be used again shortly. Other systems cache file data using a page cache.

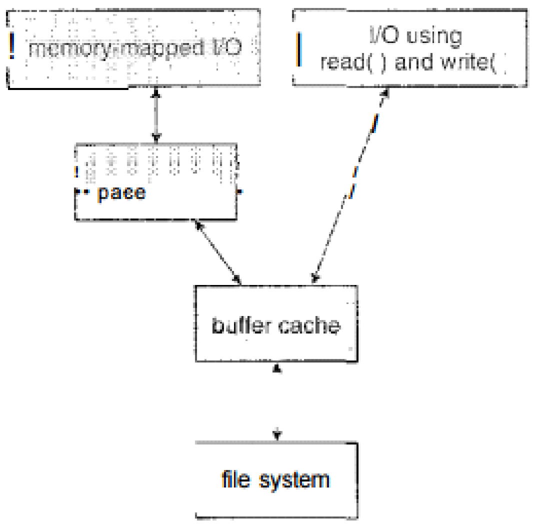


Figure 11.11 I/O without a unified buffer cache.

Efficiency

The efficient use of disk space depends heavily on the disk allocation and directory algorithms in use. For instance, UNIX inodes are preallocated on a volume. Even an "empty" disk has a percentage of its space lost to inodes. However, by preallocating the inodes and. spreading them across the volume, we improve the file system's performance. This improved performance results from the UNIX allocation and free-space algorithms, which try to keep a file's data blocks near that file's inode block to reduce seek time. As another example, performance at the cost of internal fragmentation.