File System Implementation

- A file is a collection of related information. The file system resides on secondary storage and provides efficient and convenient access to the disk by allowing data to be stored, located, and retrieved.
- File system implementation in an operating system refers to how the file system manages the storage and retrieval of data on a physical storage device such as a hard drive, solid-state drive, or flash drive.
- The file system implementation includes several components, including:
    1. **File System Structure:** The file system structure refers to how the files and directories are organized and stored on the physical storage device. This includes the layout of file systems data structures such as the directory structure, file allocation table, and nodes.
    2. **File Allocation:** The file allocation mechanism determines how files are allocated on the storage device. This can include allocation techniques such as contiguous allocation, linked allocation, indexed allocation, or a combination of these techniques.
    3. **Data Retrieval:** The file system implementation determines how the data is read from and written to the physical storage device. This includes strategies such as buffering and caching to optimize file I/O performance.
    4. **Security and Permissions:** The file system implementation includes features for managing file security and permissions. This includes access control lists (ACLs), file permissions, and ownership management.
    5. **Recovery and Fault Tolerance:** The file system implementation includes features for recovering from system failures and maintaining data integrity. This includes techniques such as journaling and file system snapshots.

- File system implementation is a critical aspect of an operating system as it directly impacts the performance, reliability, and security of the system.
- Different operating systems use different file system implementations based on the specific needs of the system and the intended use cases.
- There are several types of file systems, each designed for specific purposes and compatible with different operating systems. Some common file system types include:
    1. **FAT32 (File Allocation Table 32):** Commonly used in older versions of Windows and compatible with various operating systems.
    2. **NTFS (New Technology File System):** Used in modern Windows operating systems, offering improved performance, reliability, and security features.
    3. **ext4 (Fourth Extended File System):** Used in Linux distributions, providing features such as journaling, large file support, and extended file attributes.
    4. **HFS+ (Hierarchical File System Plus):** Used in macOS systems prior to macOS High Sierra, offering support for journaling and case-insensitive file names.

5. **APFS (Apple File System):** Introduced in macOS High Sierra and the default file system for macOS and iOS devices, featuring enhanced performance, security, and snapshot capabilities.
6. **ZFS (Zettabyte File System):** A high-performance file system known for its advanced features, including data integrity, volume management, and efficient snapshots..

Key Steps Involved In File System Implementation

- File system implementation is a crucial component of an operating system, as it provides an interface between the user and the physical storage device. Here are the key steps involved in file system implementation:
  1. **Partitioning the storage device:** The first step in file system implementation is to partition the physical storage device into one or more logical partitions. Each partition is formatted with a specific file system that defines the way files and directories are organized and stored.
  2. **File system structures:** File system structures are the data structures used by the operating system to manage files and directories. Some of the key file system structures include the superblock, inode table, directory structure, and file allocation table.
  3. **Allocation of storage space:** The file system must allocate storage space for each file and directory on the storage device. There are several methods for allocating storage space, including contiguous, linked, and indexed allocation.
  4. **File operations:** The file system provides a set of operations that can be performed on files and directories, including create, delete, read, write, open, close, and seek. These operations are implemented using the file system structures and the storage allocation methods.
  5. **File system security:** The file system must provide security mechanisms to protect files and directories from unauthorized access or modification. This can be done by setting file permissions, access control lists, or encryption.
  6. **File system maintenance:** The file system must be maintained to ensure efficient and reliable operation. This includes tasks such as disk defragmentation, disk checking, and backup and recovery.
- Overall, file system implementation is a complex and critical component of an operating system. The efficiency and reliability of the file system have a significant impact on the performance and stability of the entire system.

File System Operations

- File creation and deletion

  File creation involves allocating space on the disk for a new file and setting up its attributes and permissions. File deletion involves removing the file from the disk and releasing the space it occupies. In some file systems, deleted files may be recoverable if they have not been overwritten.

- File open and close
  File open involves establishing a connection between the file and a process or application that wishes to access it. File close involves terminating that connection and freeing up any resources used by the process or application.

- File read and write
  File read involves retrieving data from a file and transferring it to a process or application. File write involves sending data from a process or application to a file. These operations can be performed at various levels of granularity, such as bytes, blocks, or sectors.

- File seek and position
  File seek involves moving the current position of the file pointer to a specific byte or block within the file. File position refers to the current location of the file pointer within the file. These operations are useful for random access and manipulation of specific portions of a file.

- File attributes and permissions
  File attributes refer to metadata associated with a file, such as its name, size, and creation/modification dates. File permissions refer to the access control settings that determine who can read, write, execute, or modify a file. These settings can be set for individual users or groups, and can be used to restrict access to sensitive data or programs.

- Each of these file system operations is essential for managing files and directories on a computer or network. The implementation of these operations may vary depending on the type of file system and the operating system being used.

Implementation Issues

- Disk space management
  File systems need to manage disk space efficiently to avoid wasting space and to ensure that files can be stored in contiguous blocks whenever possible. Techniques for disk space management include free space management, fragmentation prevention, and garbage collection.

- Consistency checking and error recovery

File systems need to ensure that files and directories remain consistent and error-free. Techniques for consistency checking and error recovery include journaling, check summing, and redundancy. If errors occur, file systems may need to perform recovery operations to restore lost or damaged data.

- File locking and concurrency control
  File systems need to manage access to files by multiple processes or users to avoid conflicts and ensure data integrity. Techniques for file locking and concurrency control include file locking, semaphore, and transaction management.

- Performance optimization
  File systems need to optimize performance by reducing file access times, increasing throughput, and minimizing system overhead. Techniques for performance optimization include caching, buffering, prefetching, and parallel processing.

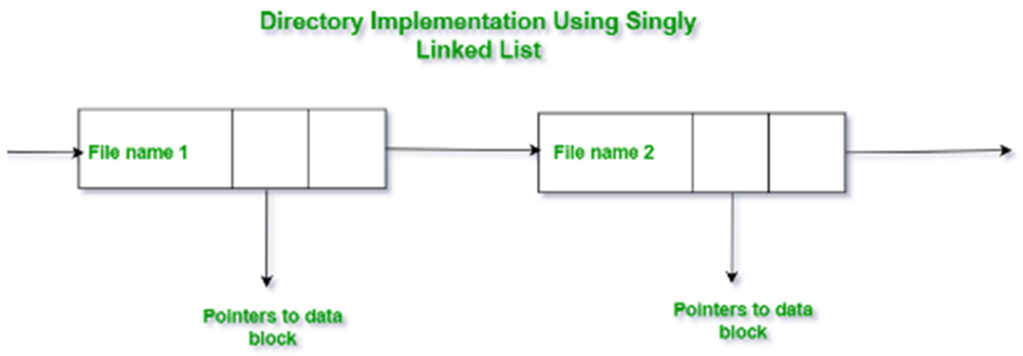
- These implementation issues are critical for ensuring that file systems operate efficiently, reliably, and securely. File system designers must carefully balance these factors to create a system that meets the needs of its users and the applications that use it.

Directory Implementation

- Directory implementation in the operating system can be done using Singly Linked List and Hash table.
- The efficiency, reliability, and performance of a file system are greatly affected by the selection of directory-allocation and directory-management algorithms.
- There are numerous ways in which the directories can be implemented. But we need to choose an appropriate directory implementation algorithm that enhances the performance of the system.

Directory Implementation using Singly Linked List

- The implementation of directories using a singly linked list is easy to program but is time-consuming to execute. Here we implement a directory by using a linear list of filenames with pointers to the data blocks.



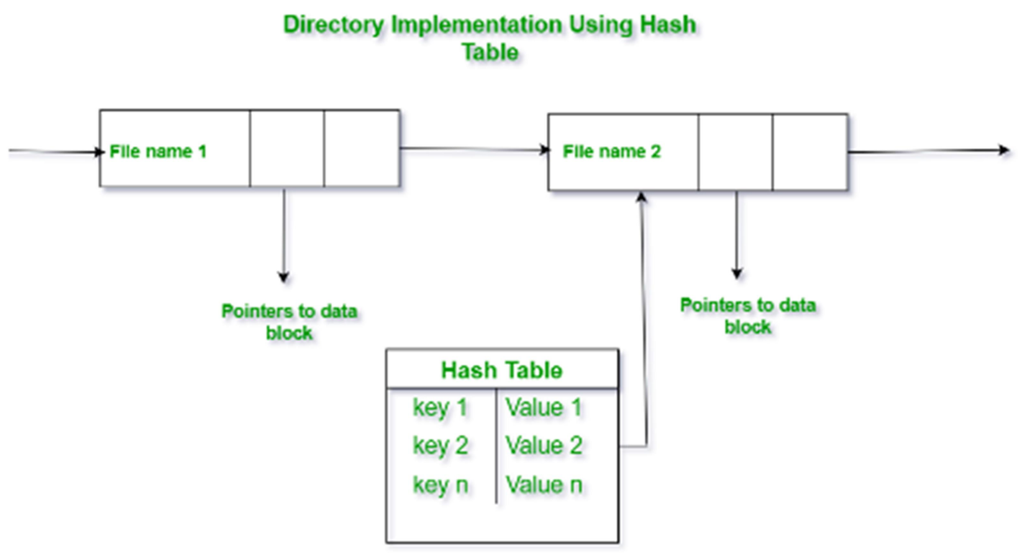
- To create a new file the entire list has to be checked such that the new directory does not exist previously.
- The new directory then can be added to the end of the list or at the beginning of the list.
- In order to delete a file, we first search the directory with the name of the file to be deleted. After searching we can delete that file by releasing the space allocated to it.
- To reuse the directory entry we can mark that entry as unused or we can append it to the list of free directories.
- To delete a file linked list is the best choice as it takes less time.

Disadvantage

- The main disadvantage of using a linked list is that when the user needs to find a file the user has to do a linear search. In today's world directory information is used quite frequently and linked list implementation results in slow access to a file. So the operating system maintains a cache to store the most recently used directory information.

Directory Implementation using Hash Table

- An alternative data structure that can be used for directory implementation is a hash table. It overcomes the major drawbacks of directory implementation using a linked list. In this method, we use a hash table along with the linked list.
- Here the linked list stores the directory entries, but a hash data structure is used in combination with the linked list.
- In the hash table for each pair in the directory key-value pair is generated. The hash function on the file name determines the key and this key points to the corresponding file stored in the directory.
- This method efficiently decreases the directory search time as the entire list will not be searched on every operation. Using the keys the hash table entries are checked and when the file is found it is fetched.



Disadvantage:

- The major drawback of using the hash table is that generally, it has a fixed size and its dependency on size. But this method is usually faster than linear search through an entire directory using a linked list.

File Allocation Methods

- The allocation methods define how the files are stored in the disk blocks. There are three main disk space or file allocation methods.
  1. Contiguous Allocation
  2. Linked Allocation
  3. Indexed Allocation

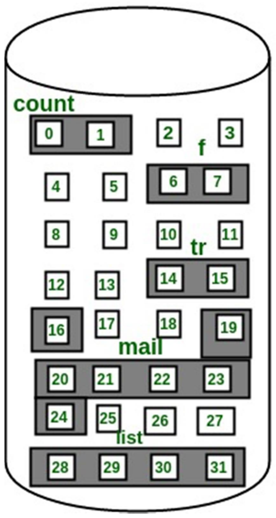
- The main idea behind these methods is to provide:
  - Efficient disk space utilization.
  - Fast access to the file blocks.

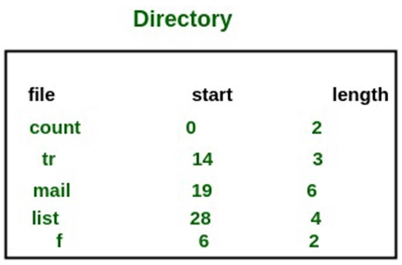
All the three methods have their own advantages and disadvantages as discussed below:

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

Advantages:

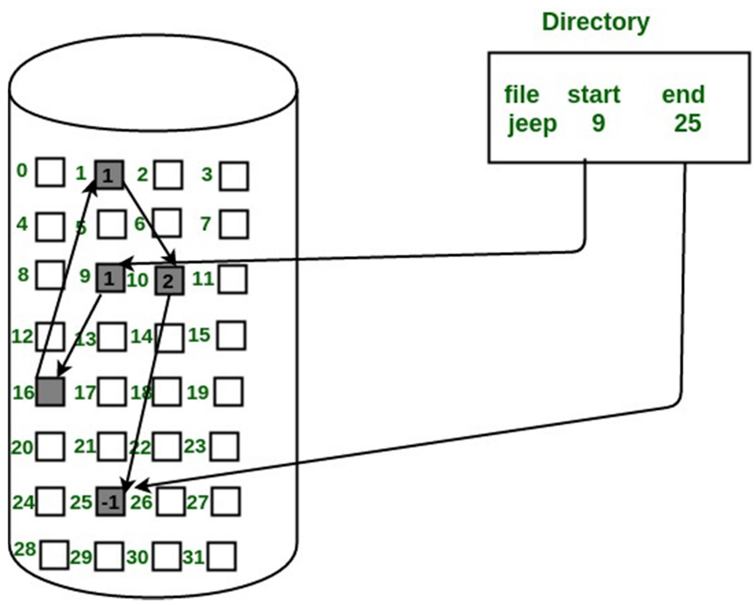
- Both the Sequential and Direct Accesses are supported by this. For direct access, the address of the kth block of the file which starts at block b can easily be obtained as (b+k).
- This is extremely fast since the number of seeks are minimal because of contiguous allocation of file blocks.

Disadvantages:

- This method suffers from both internal and external fragmentation. This makes it inefficient in terms of memory utilization.
- Increasing file size is difficult because it depends on the availability of contiguous memory at a particular instance.

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.



Advantages:

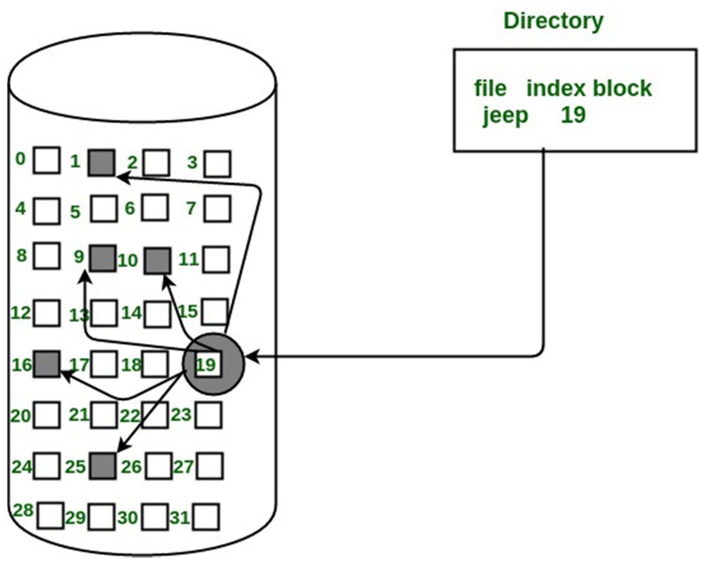
- This is very flexible in terms of file size. File size can be increased easily since the system does not have to look for a contiguous chunk of memory.
- This method does not suffer from external fragmentation. This makes it relatively better in terms of memory utilization.

Disadvantages:

- Because the file blocks are distributed randomly on the disk, a large number of seeks are needed to access every block individually. This makes linked allocation slower.
- It does not support random or direct access. We can not directly access the blocks of a file. A block k of a file can be accessed by traversing k blocks sequentially (sequential access) from the starting block of the file via block pointers.
- Pointers required in the linked allocation incur some extra overhead.

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:



Advantages:

- This supports direct access to the blocks occupied by the file and therefore provides fast access to the file blocks.
- It overcomes the problem of external fragmentation.

Disadvantages:

- The pointer overhead for indexed allocation is greater than linked allocation.
- For very small files, say files that expand only 2-3 blocks, the indexed allocation would keep one entire block (index block) for the pointers which is inefficient in terms of memory utilization. However, in linked allocation we lose the space of only 1 pointer per block.

Free space management

- Free space management is a critical aspect of operating systems as it involves managing the available storage space on the hard disk or other secondary storage devices. The operating system uses various techniques to manage free space and optimize the use of storage devices.
- The system keeps tracks of the free disk blocks for allocating space to files when they are created. Also, to reuse the space released from deleting the files, free space management becomes crucial.
- The system maintains a free space list which keeps track of the disk blocks that are not allocated to some file or directory. The free space list can be implemented mainly as:

1. 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.

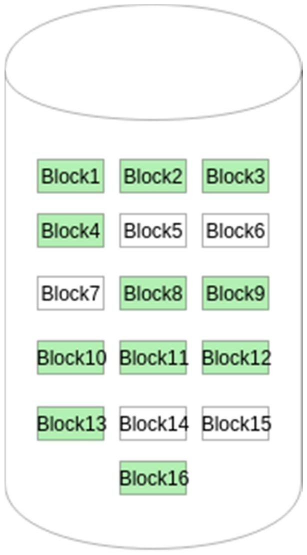


Figure - 1

Advantages –

- Simple to understand.
- Finding the first free block is efficient. It requires scanning the words (a group of 8 bits) in a bitmap for a non-zero word. (A 0-valued word has all bits 0). The first free block is then found by scanning for the first 1 bit in the non-zero word.

2. 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.
- In Figure-2, the free space list head points to Block 5 which points to Block 6, the next free block and so on. The last free block would contain a null pointer indicating the end of free list. A drawback of this method is the I/O required for free space list traversal.

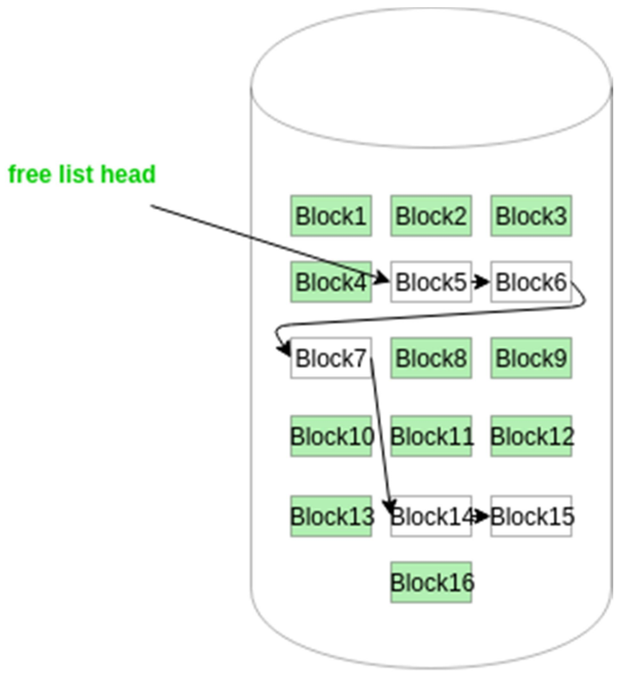


Figure - 2

3. 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.
- An advantage of this approach is that the addresses of a group of free disk blocks can be found easily.

4. 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

Advantages and disadvantages of free space management techniques in operating systems:

Advantages:

1. Efficient use of storage space: Free space management techniques help to optimize the use of storage space on the hard disk or other secondary storage devices.
2. Easy to implement: Some techniques, such as linked allocation, are simple to implement and require less overhead in terms of processing and memory resources.

3. Faster access to files: Techniques such as contiguous allocation can help to reduce disk fragmentation and improve access time to files.

Disadvantages:

1. Fragmentation: Techniques such as linked allocation can lead to fragmentation of disk space, which can decrease the efficiency of storage devices.
2. Overhead: Some techniques, such as indexed allocation, require additional overhead in terms of memory and processing resources to maintain index blocks.
3. Limited scalability: Some techniques, such as FAT, have limited scalability in terms of the number of files that can be stored on the disk.
4. Risk of data loss: In some cases, such as with contiguous allocation, if a file becomes corrupted or damaged, it may be difficult to recover the data.
5. Overall, the choice of free space management technique depends on the specific requirements of the operating system and the storage devices being used. While some techniques may offer advantages in terms of efficiency and speed, they may also have limitations and drawbacks that need to be considered.