

Unit-5: Information management

Information management in operating systems refers to the process of collecting, organizing, and storing data in a structured and efficient manner. It is an essential aspect of any operating system as it plays a crucial role in maintaining the stability, security, and performance of the system. The goal of information management in operating systems is to ensure that the data is easily accessible, retrievable, and secure.

The operating system is responsible for managing different types of data such as system configurations, user files, applications, and system logs. The information management system in an operating system includes various components such as storage management, file systems, and access control mechanisms.

Storage management is the process of managing the physical storage devices and memory in the system. This includes managing the allocation and de allocation of storage space, ensuring data consistency and integrity, and managing the storage resources to maximize efficiency and performance.

File systems are the structures used to store and organize data in the operating system. It determines the layout of files and directories on the storage devices and provides the mechanism for accessing and retrieving the data. Different file systems have different features, such as support for large files, security, and reliability.

Access control mechanisms are used to manage user access to the data and system resources. This includes setting permissions for users and groups, managing authentication and authorization processes, and enforcing access policies.

In conclusion, information management in operating systems is a critical aspect of any operating system as it affects the stability, security, and performance of the system. By managing the storage, file systems, and access control mechanisms, the operating system can ensure that data is easily accessible, retrievable, and secure.

A Simple File system

A Simple File System (SFS) is a basic file system that is used to store and manage data on a storage device. It is a hierarchical file system that organizes data in a tree-like structure of directories and files. The main purpose of SFS is to provide an efficient and reliable method for organizing and storing data.

In an SFS, the root directory is the top-level directory that contains all other directories and files. Directories within the root directory are called sub-directories and can contain further sub-directories or files. The files in the system can be any type of data, such as text files, images, or audio files.

SFS uses file attributes to store information about each file, such as the file name, creation date, size, and ownership. It also uses file descriptors to manage the access and retrieval of data. The file descriptors are unique numbers assigned to each file and used to reference the file in the system.

SFS uses a **file allocation table (FAT)** to manage the allocation and de-allocation of storage space. The FAT is a table that contains information about the location of each file on the storage device. When a file is created, the operating system allocates a portion of the storage space to the file and updates the FAT. When the file is deleted, the operating system frees up the storage space and updates the FAT accordingly.

In conclusion, SFS is a simple and efficient file system that provides a basic mechanism for organizing and storing data. It is suitable for small systems with limited storage requirements, and it is easy to implement and maintain. SFS is a good starting point for understanding file systems and can be used as a foundation for more advanced file systems.

General Model of a File System

A file system is a way of organizing and storing data on a storage medium (e.g. a hard disk, flash drive, etc.) so that it can be easily retrieved, updated and managed. The general model of a file system consists of the following components:

1. **Storage space:** This is the physical space on the storage medium where the data is stored.
2. **Files:** A file is a collection of data that is stored on the storage medium. Files can be of different types (e.g. text files, image files, audio files, etc.) and have different attributes (e.g. name, size, date of creation, etc.).
3. **Directories:** A directory is a special type of file that contains a list of other files and/or directories. Directories allow users to organize files into a hierarchical structure.
4. **File names:** A file name is a unique identifier that is assigned to each file in the file system. File names can be either simple (e.g. "file.txt") or hierarchical (e.g. "/dir1/dir2/file.txt").
5. **File metadata:** This is data that provides information about a file (e.g. its size, date of creation, etc.).
6. **File access methods:** These are the methods used to read and write data to and from files.
7. **Allocation methods:** These are the methods used to allocate storage space on the storage medium for files.

The general model of a file system provides a framework for organizing and managing data on a storage medium in a way that makes it easily accessible, retrievable and manageable.

Symbolic File System

A symbolic file system, also known as a symbolic link or symlink, is a special type of file in a file system that acts as a pointer to another file or directory. Unlike a hard link, which is a direct link to the actual file, a symbolic link is a separate file that contains a reference to the original file.

When a program accesses a symbolic link, the operating system redirects the program to the file or directory that the link is pointing to, as if the program were accessing the original file directly. This allows programs to access the linked file or directory without having to know its actual location.

Symbolic links are useful for several purposes, including:

1. **Creating aliases for files or directories:** A symbolic link can be used to create an alias for a file or directory, allowing it to be accessed from multiple locations without having to copy the file or directory to each location.
2. **Redirecting a file or directory to a new location:** If a file or directory is moved to a new location, a symbolic link can be created in its original location that points to its new location, allowing programs that access the file or directory through the original location to continue working without modification.
3. **Implementing file or directory hierarchies:** A symbolic link can be used to create a file or directory hierarchy, where a link at one location points to a file or directory at another location.

Basic File System

A file system is a way of organizing and storing digital information on a storage device, such as a hard drive or solid-state drive. The file system provides a way to name, store, retrieve, and manipulate the stored information.

There are several types of file systems, including:

1. **FAT** (File Allocation Table): This is an old file system that was used on floppy disks and older versions of Windows. It has limited capabilities and is not widely used today.
2. **NTFS** (New Technology File System): This is the file system used by recent versions of Windows. It provides advanced features such as file and folder permissions, encryption, and support for large storage devices.
3. **exFAT** (Extended File Allocation Table): This is a newer file system that is used for removable storage devices, such as memory cards and USB drives. It provides similar features to NTFS but is optimized for use on smaller, removable storage devices.
4. **HFS** (Hierarchical File System): This is the file system used by Apple's Mac OS X. It provides features similar to those found in NTFS, but is optimized for use with Mac OS X.
5. **Ext2/Ext3/Ext4**: These are file systems used by Linux. Ext4 is the latest and most widely used file system for Linux.

A file system is responsible for several key tasks, including:

1. Allocating space on the storage device to new files and directories.
2. Keeping track of which areas of the storage device are being used and which are free.
3. Ensuring that information is stored in a way that is easily recoverable in the event of a crash or other failure.

4. Providing a way for the operating system to access and manipulate the stored information.

Access control verification

Access control verification is the process of verifying that a user has the proper authorization to access a particular resource or system. This process is essential in ensuring the security and confidentiality of sensitive information.

Access control verification is typically performed using some combination of user authentication and authorization. User authentication is the process of verifying the identity of the user, such as by requiring a password or smart card. Authorization is the process of determining if the authenticated user is allowed to access the requested resource or system.

There are several different methods for implementing access control verification, including:

1. **Role-based access control:** Users are assigned to roles, and access to resources is granted based on the roles.
2. **Rule-based access control:** Access to resources is determined by a set of rules or conditions, such as the time of day or the user's location.
3. **Discretionary access control:** Access to resources is controlled by the owner of the resource, who grants access to specific users.
4. **Mandatory access control:** Access to resources is determined by security labels assigned to both the user and the resource, and access is only granted if the security labels match.

In summary, access control verification is an important aspect of security that helps to ensure that only authorized users can access sensitive information and systems. The method used for access control verification will depend on the specific security requirements and the resources being protected.

Logical File System

A Logical File System (LFS) is an abstract way of organizing and accessing data in a computer file system. It's a way of viewing and manipulating the underlying physical storage, such as hard drives, as a collection of files and directories, rather than just a block of raw storage.

In an LFS, files are organized into a hierarchical structure of directories and sub-directories, allowing for easy navigation and organization of data. The LFS provides a higher-level interface for managing files and directories, including operations like creating, reading, writing, and deleting files, as well as creating and removing directories.

One of the key features of an LFS is the ability to use file names and paths, which makes it easier for users to locate and access their files. The LFS also provides a mechanism for tracking changes to files and directories, such as updates or deletions, and for managing access control to prevent unauthorized access.

LFS are implemented in operating systems, and are a fundamental component of modern computing. Some examples of LFS include the file system used by Windows (NTFS), the file system used by Unix-based systems (such as ext4), and the file system used by Apple's MacOS (HFS+).

Physical File system- File System Interface

A Physical File System (PFS) is a low-level interface to the actual physical storage devices, such as hard drives, solid-state drives (SSDs), and flash memory. The PFS is responsible for reading and writing data to the physical storage media and for handling issues such as storage allocation and fragmentation.

The Physical File System Interface is the set of operations and data structures that a file system uses to communicate with the physical storage media. This interface provides a way for the file system to interact with the physical storage devices, such as reading and writing data blocks, as well as performing maintenance operations such as error checking and correction, and wear leveling.

The Physical File System Interface is typically implemented by the storage controller, which is responsible for managing the physical storage media. This interface may include operations such as reading and writing blocks of data, as well as management operations such as formatting the storage media, configuring partitions, and checking the health of the storage devices.

The Physical File System Interface is an important component of a computer system, as it enables the file system to access the physical storage devices and manage the data stored on them. A well-designed Physical File System Interface can improve performance, reliability, and overall system stability by providing a low-level and efficient way of accessing the physical storage media

File Concept

A file is a collection of data that is stored in a computer's memory or on a storage device such as a hard drive, flash drive, or cloud storage service. Files can contain text, images, audio, video, or any other type of digital data.

The file concept is fundamental to modern computing and provides a convenient way to organize and access data. Files are usually given a name, which makes it easier for

users to locate and access them. They can also be organized into directories and subdirectories, allowing for easy navigation and management of data.

Each file has a type or format that determines how the data it contains is organized and how it can be used. For example, a text file might contain plain text, while an image file might contain binary data that represents an image. Different types of files can be opened and edited with different software applications, such as a text editor for text files, or an image editor for image files.

In a computer's file system, files are stored as a series of bits and bytes, which represent the data they contain. The file system is responsible for managing the files, including allocating space for new files, updating existing files, and deleting files that are no longer needed.

Overall, the file concept is an important part of modern computing, providing a convenient and efficient way to store and manage data.

Access Methods

Access Methods are the techniques and algorithms used by computer systems to retrieve data stored in a database, file system, or other type of data storage. Access methods determine the way in which data is stored and organized, as well as the way in which it can be retrieved.

There are several different types of access methods, each with its own advantages and disadvantages, and each suited to different types of data and usage patterns. Some common access methods include:

1. **Sequential Access:** Data is stored and retrieved in a linear, sequential manner, similar to reading a book from front to back. This method is typically used for large data files where the data must be read in a specific order, such as log files.

2. **Direct Access:** Data is stored and retrieved based on its physical location on the storage device, allowing for fast access to specific data. This method is used for systems where data needs to be retrieved quickly, such as real-time systems.
3. **Indexed Access:** Data is stored and retrieved using an index, which acts as a lookup table to find the location of specific data in the storage system. This method is commonly used for databases, where data is organized into tables and indexes are used to find specific records quickly.
4. **Hash-Based Access:** Data is stored and retrieved using a hash function, which converts the data into a unique value that can be used to index and retrieve the data. This method is used for systems where data must be retrieved quickly, and where data is frequently searched for.

Access methods play a critical role in information management, as they determine the performance, reliability, and scalability of a system. The choice of access method depends on the type of data being stored, the usage patterns, and the performance requirements of the system.

Directory Structure

A directory structure is the way files and folders are organized and stored on a computer's file system. It provides a way to organize and categorize files and folders into a hierarchical structure, making it easier to find and access data.

Here's an example of a directory structure:

```
/
home/
user1/
documents/
resume.txt
cover_letter.txt
```

```
pictures/  
vacation/  
hawaii.jpg  
grand_canyon.jpg  
profile_pic.jpg  
user2/  
documents/  
budget.xlsx  
presentation.ppt  
music/  
pop/  
justin_bieber.mp3  
ariana_grande.mp3  
rock/  
metallica.mp3  
acdc.mp3  
etc/  
config_files/  
apache2.conf  
ssh_config  
tmp/
```

In this example, the root directory (/) contains several subdirectories, including home, etc, and tmp. The home directory contains user-specific data, with separate directories for user1 and user2. Each user has a documents directory and a pictures directory, which contain their personal files. The user1 directory contains a pictures directory with two subdirectories: vacation and profile_pic.

The etc directory contains system-wide configuration files, and the tmp directory is used for temporary storage.

This example demonstrates how a directory structure can be used to organize data into a hierarchical structure, making it easier to locate and access data. By categorizing and

organizing data into meaningful directories, users can quickly find the files they need, even if they have a large number of files and folders stored on their computer.

SECURITY

Protection/ Security in information management refers to the measures taken to ensure the confidentiality, integrity, and availability of data stored in a computer system or network. The goal of protection is to prevent unauthorized access, modification, or destruction of data, and to ensure that data is available when needed.

There are several ways to protect information:

1. **Access Control:** Access control refers to the measures taken to ensure that only authorized users can access specific data. This can include setting permissions on files and folders, using passwords or biometric authentication, and implementing firewalls to control access to a network.
2. **Encryption:** Encryption is the process of converting data into a form that can only be deciphered by authorized users. Encryption can be used to protect data while it is stored, transmitted, or processed.
3. **Backup and Recovery:** Backup and recovery refers to the process of creating and storing copies of data to ensure that it can be recovered in the event of a disaster or data loss. This includes regular backups of data to an external storage device, as well as disaster recovery plans to ensure that critical systems and data can be restored in the event of an outage.
4. **Physical Security:** Physical security refers to the measures taken to protect computer systems and data from theft, damage, or unauthorized access. This can include secure server rooms, locks and security cameras, and other physical measures to prevent unauthorized access to computer systems.
5. **Network Security:** Network security refers to the measures taken to protect a network from unauthorized access, malicious attacks, and data theft. This can

include firewalls, intrusion detection systems, and encryption to protect data as it is transmitted over the network.

Consistency Semantics In File System

Implementation

Consistency semantics in a file system refers to the guarantees about the state of the file system and the order of operations. There are several consistency models that a file system can implement, including:

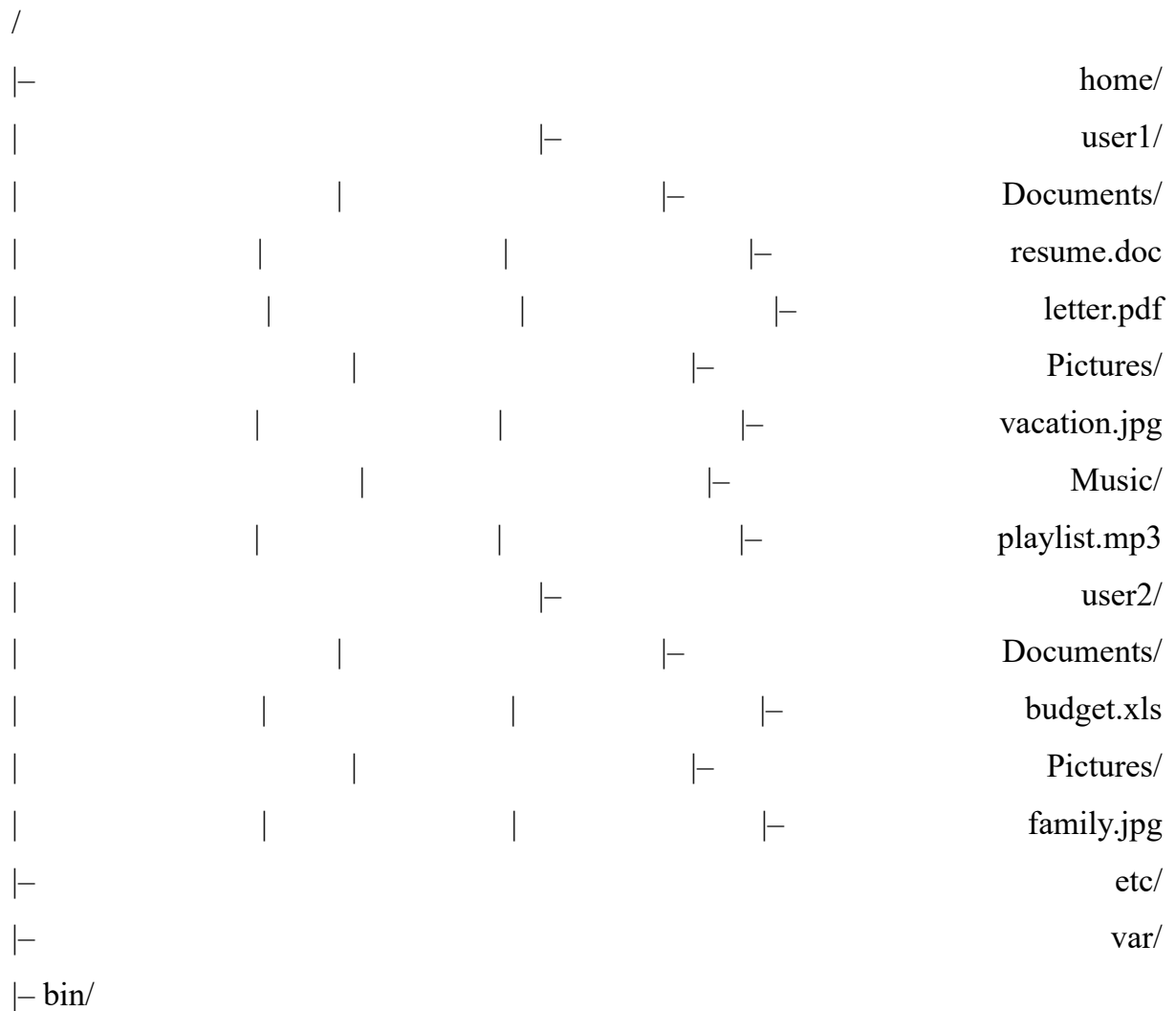
1. Strong consistency: All operations on the file system appear to occur atomically and in the order they were issued.
2. Eventual consistency: After a sufficient amount of time, all nodes in the file system will eventually agree on the state of the data.
3. Casual consistency: Operations that are casually related are guaranteed to be seen in the same order by all nodes in the file system.

Implementing consistency semantics in a file system typically involves techniques such as locking, versioning, and distributed consensus protocols. The specific implementation will depend on the desired consistency guarantees and the underlying architecture of the file system

File – System Structure

A file system structure is the way that files and directories are organized on a storage device.

Here is an example of a file system structure



In this example, the root directory / contains three subdirectories: home/, etc/, var/, and bin/. The home/ directory contains two subdirectories, user1/ and user2/, each of which represents a different user's home directory. Each user's home directory contains a Documents/ directory, a Pictures/ directory, and a Music/ directory. The Documents/ directory contains various documents, such as resume.doc and letter.pdf, while the Pictures/ directory contains pictures such as vacation.jpg and family.jpg. The Music/ directory contains audio files such as playlist.mp3.

File Allocation Methods

There are different kinds of methods that are used to allocate disk space. We must select the best method for the file allocation because it will directly affect the system performance and system efficiency. With the help of the allocation method, we can utilize the disk, and also files can be accessed.

There are various types of file allocations method:

1. Contiguous allocation
2. Extents
3. Linked allocation
4. Clustering
5. FAT
6. Indexed allocation
7. Linked Indexed allocation
8. Multilevel Indexed allocation
9. Inode

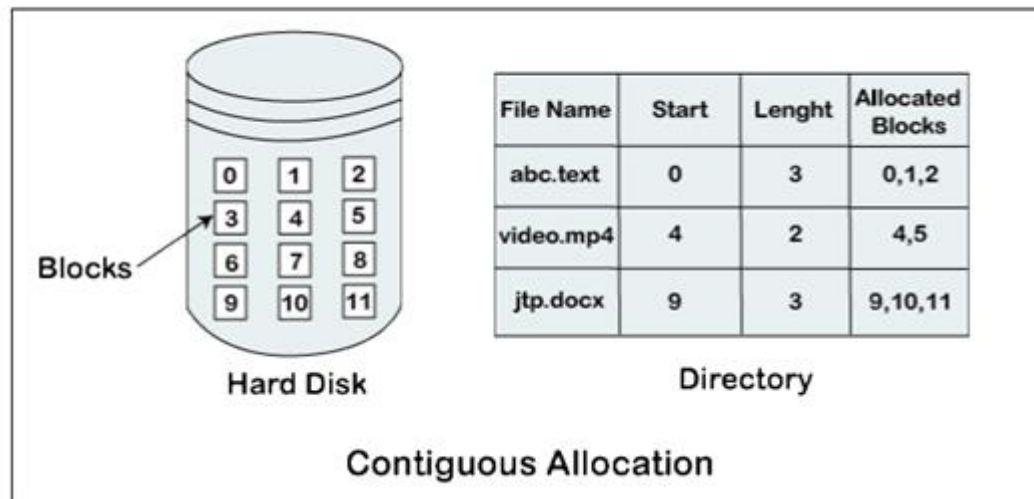
There are different types of file allocation methods, but we mainly use three types of file allocation methods:

1. Contiguous allocation
2. Linked list allocation
3. Indexed allocation

These methods provide quick access to the file blocks and also the utilization of disk space in an efficient manner.

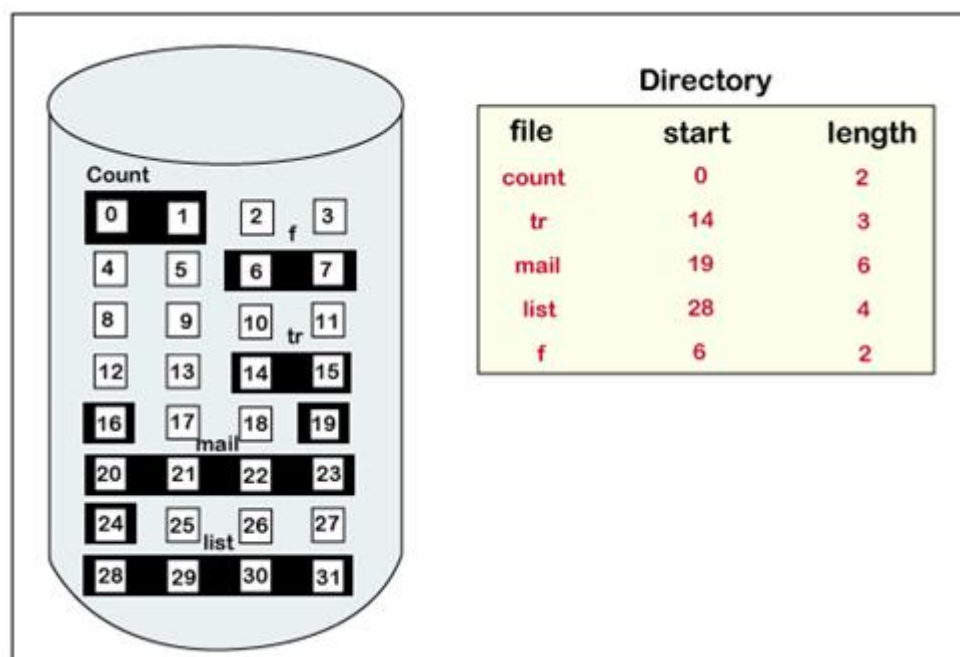
Contiguous Allocation: – Contiguous allocation is one of the most used methods for allocation. Contiguous allocation means we allocate the block in such a manner, so that in the hard disk, all the blocks get the contiguous physical block.

We can see in the below figure that in the directory, we have three files. In the table, we have mentioned the starting block and the length of all the files. We can see in the table that for each file, we allocate a contiguous block.



Example of contiguous allocation

We can see in the given diagram, that there is a file. The name of the file is 'mail.' The file starts from the 19th block and the length of the file is 6. So, the file occupies 6 blocks in a contiguous manner. Thus, it will hold blocks 19, 20, 21, 22, 23, 24.



Advantages of Contiguous Allocation

The advantages of contiguous allocation are:

1. The contiguous allocation method gives excellent read performance.
2. Contiguous allocation is easy to implement.
3. The contiguous allocation method supports both types of file access methods that are sequential access and direct access.
4. The Contiguous allocation method is fast because, in this method number of seeks is less due to the contiguous allocation of file blocks.

Disadvantages of Contiguous allocation

The disadvantages of contiguous allocation method are:

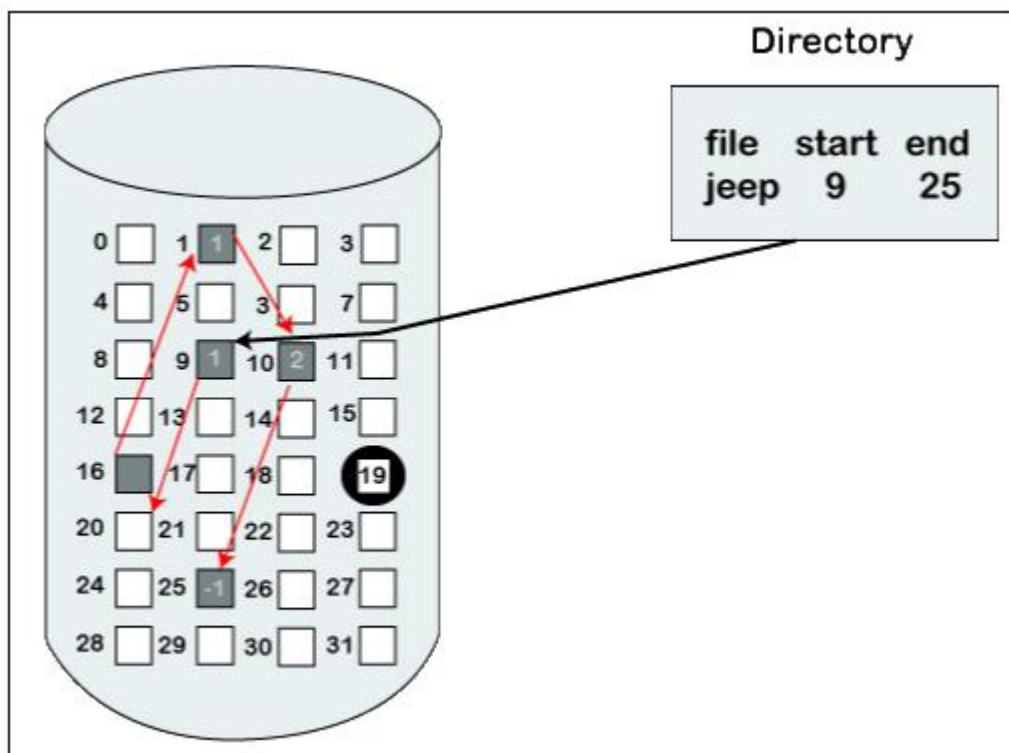
1. In the contiguous allocation method, sometimes disk can be fragmented.
2. In this method, it is difficult to increase the size of the file due to the availability of the contiguous memory block.

Linked List Allocation

The linked list allocation method overcomes the drawbacks of the contiguous allocation method. In this file allocation method, each file is treated as a linked list of disks blocks. In the linked list allocation method, it is not required that disk blocks assigned to a specific file are in the contiguous order on the disk. The directory entry comprises of a pointer for starting file block and also for the ending file block. Each disk block that is allocated or assigned to a file consists of a pointer, and that pointer point the next block of the disk, which is allocated to the same file.

Example of linked list allocation

We can see in the below figure that we have a file named 'jeep.' The value of the start is 9. So, we have to start the allocation from the 9th block, and blocks are allocated in a random manner. The value of the end is 25. It means the allocation is finished on the 25th block. We can see in the below figure that the block (25) comprised of -1, which means a null pointer, and it will not point to another block.



Advantages of Linked list allocation

There are various advantages of linked list allocation:

1. In linked list allocation, there is no external fragmentation. Due to this, we can utilize the memory better.
2. In linked list allocation, a directory entry only comprises of the starting block address.
3. The linked allocation method is flexible because we can quickly increase the size of the file because, in this to allocate a file, we do not require a chunk of memory in a contiguous form.

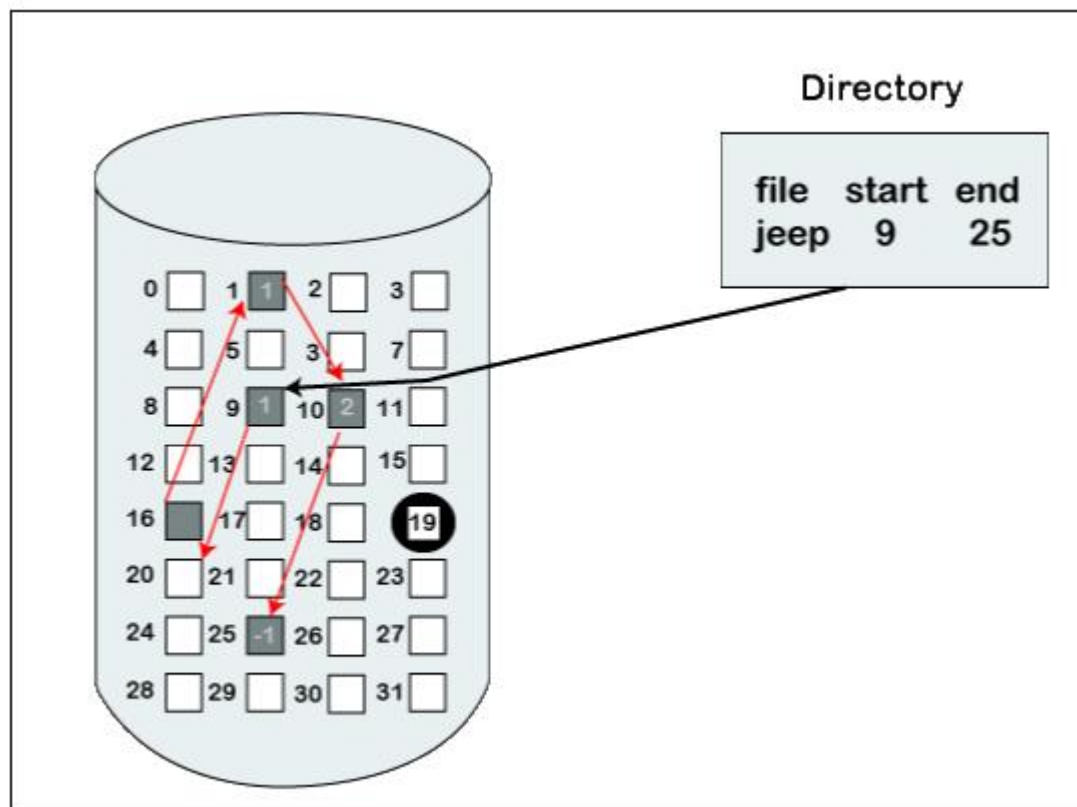
Disadvantages of Linked list Allocation

There are various disadvantages of linked list allocation:

1. Linked list allocation does not support direct access or random access.
2. In linked list allocation, we need to traverse each block.
3. If the pointer in the linked list break in linked list allocation, then the file gets corrupted.
4. In the disk block for the pointer, it needs some extra space.

Indexed Allocation

The Indexed allocation method is another method that is used for file allocation. In the index allocation method, we have an additional block, and that block is known as the index block. For each file, there is an individual index block. In the index block, the i th entry holds the disk address of the i th file block. We can see in the below figure that the directory entry comprises of the address of the index block.



Advantages of Indexed Allocation

The advantages of index allocation are:

1. The index allocation method solves the problem of external fragmentation.
2. Index allocation provides direct access.

Disadvantages of Indexed Allocation

The disadvantages of index allocation are:

1. In index allocation, pointer overhead is more.
2. We can lose the entire file if an index block is not correct.
3. It is totally a wastage to create an index for a small file.

A single index block cannot hold all the pointer for files with large sizes.

Free space management

Free space management is the process by which a file system manages unused disk space to ensure that it is available for future file storage. Here is an example of how free space management works:

1. **Initial allocation:** When a file system is first created, it allocates a portion of the disk space to be used for file storage. The remaining space is considered free space.
2. **File creation:** When a new file is created, the file system allocates a portion of the free space to store the file's data. The allocated space is now considered used space, and the remaining free space is updated.

3. **File deletion:** When a file is deleted, the space that was previously used to store the file's data becomes free space again. The file system updates the free space information to reflect this change.
4. **File expansion:** If a file grows in size, the file system may need to allocate additional disk space to store the file's new data. This additional space is taken from the free space pool.
5. **File contraction:** If a file decreases in size, the file system may choose to release the unused space back to the free space pool.
6. **De-fragmentation:** Over time, as files are created, deleted, and modified, the file system may become fragmented, with free space scattered across the disk. To ensure that the free space is available for future file storage, the file system may perform a de-fragmentation operation, which reorganizes the disk to consolidate the free space into larger contiguous blocks.

The specific methods used for free space management will depend on the file system being used and its requirements for disk space usage and efficiency. Some file systems use a bitmap to keep track of the used and free disk blocks, while others use a linked list or a tree structure.