Definition of file system implementation

File system implementation is the process of designing, developing, and implementing the software components that manage the organization, allocation, and access to files on a storage device in an operating system.

Importance of file system implementation

The file system implementation plays a critical role in ensuring the reliability, performance, and security of an operating system's file storage capabilities. Without an effective file system implementation, an operating system cannot efficiently manage the storage of data on a storage device, resulting in data loss, corruption, and inefficiency.

File System Structure

* Disk layout and partitioning
* File system organization
* File allocation methods
* Directory structure

Disk layout and partitioning

The disk layout and partitioning refers to how a physical disk is divided into logical partitions, which can be accessed by the operating system as separate entities. The disk is divided into one or more partitions which can be formatted with a file system. Disk partitioning involves creating partitions on the disk, while disk formatting involves creating a file system on the partition. The partitioning process is typically done when the disk is first installed, and the formatting process is typically done when a partition is created.

File system organization

The file system organization refers to how files and directories are stored on the disk. A file system is responsible for managing files and directories, and providing a way for users and applications to access and modify them. Different file systems may organize files and directories in different ways, and may use different methods for storing and accessing them. For example, the FAT file system used by Windows organizes files in a simple directory hierarchy, while the HFS+ file system used by macOS organizes files in a more complex tree structure.

File allocation methods

The file allocation method refers to how file data is stored on the disk. There are several different file allocation methods, including contiguous allocation, linked allocation, and indexed allocation. Contiguous allocation stores files in contiguous blocks on the disk, while linked allocation uses pointers to link blocks of data together. Indexed allocation uses an index to keep track of where each file block is stored on the disk.

Directory structure

The directory structure refers to how directories are organized and managed on the disk. Directories are used to organize files and other directories into a hierarchy, which can be navigated by users and applications. Different file systems may use different directory structures, including single-level directories, two-level directories, and tree-structured directories. Directories can also have various attributes such as permissions and ownership, which can control who can access and modify files within them.

File System Operations

* File creation and deletion
* File open and close
* File read and write
* File seek and position
* File attributes and permissions

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
* Consistency checking and error recovery
* File locking and concurrency control
* Performance optimization

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, checksumming, 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**

There is the number of algorithms by using which, the directories can be implemented. However, the selection of an appropriate directory implementation algorithm may significantly affect the performance of the system.

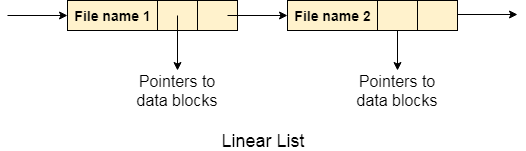
The directory implementation algorithms are classified according to the data structure they are using. There are mainly two algorithms which are used in these days.

### 1. Linear List

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

**Characteristics**

1. When a new file is created, then the entire list is checked whether the new file name is matching to a existing file name or not. In case, it doesn't exist, the file can be created at the beginning or at the end. Therefore, searching for a unique name is a big concern because traversing the whole list takes time.
2. The list needs to be traversed in case of every operation (creation, deletion, updating, etc) on the files therefore the systems become inefficient.

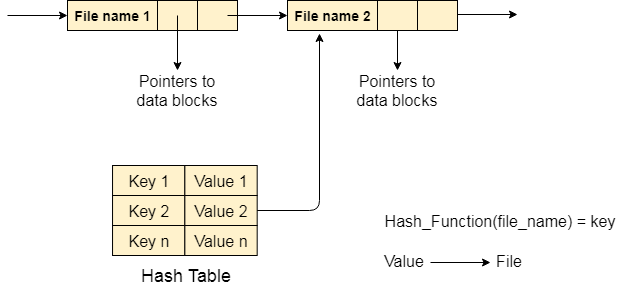


### 2. Hash Table

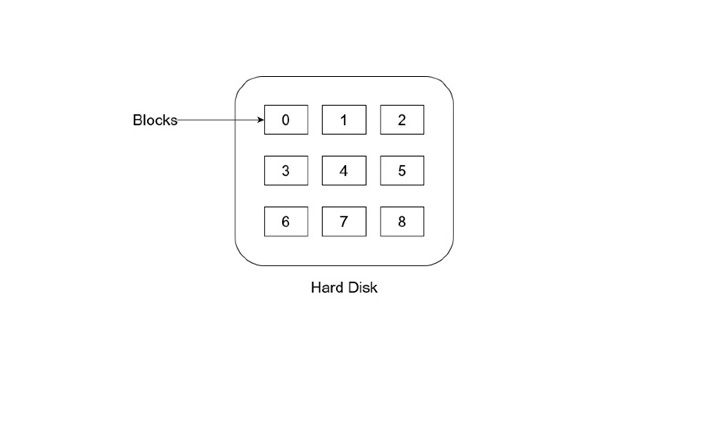
To overcome the drawbacks of singly linked list implementation of directories, there is an alternative approach that is hash table. This approach suggests to use hash table along with the linked lists.

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.

Now, searching becomes efficient due to the fact that now, entire list will not be searched on every operating. Only hash table entries are checked using the key and if an entry found then the corresponding file will be fetched using the value.



File allocation methods refer to the strategies employed by computer operating systems for the efficient distribution of storage space on disks or other storage media. Their main objective is to optimize the utilization of available space and minimize fragmentation, which can impede file access and decrease the overall performance of the system. There are several different file allocation methods that are commonly used, each with its own strengths and weaknesses.



**Contiguous File Allocation**

In this method, files are stored in a continuous block of free space on the disk meaning that all the data for a particular file is stored in one continuous section of the disk. When a file is created, the operating system searches for a contiguous block of free space large enough to accommodate the file. If such a block is found, the file is stored in that block, and the operating system keeps track of the starting address and the size of the block.

The advantage of contiguous file allocation is that it provides fast access to files, as the operating system only needs to remember the starting address of the file. When a user requests access to a file, the operating system can quickly locate the file's starting address and read the entire file sequentially. This method is particularly useful for large files, such as video or audio files, which can be accessed more quickly when stored in contiguous blocks.

However, contiguous file allocation has some limitations. One significant disadvantage is that it can lead to fragmentation when files are deleted or when new files are created. If a file is deleted, the space it occupied becomes free, but that space may not be contiguous with the remaining free space on the disk. This can result in gaps or fragments of free space scattered throughout the disk, making it difficult for the operating system to find contiguous blocks of free space for new files.

**Linked File Allocation**

In this method, files are stored in non-contiguous blocks of free space on the disk, and each block is linked to the next block using a pointer. When a file is created, the operating system searches for a series of free blocks that are large enough to store the file, and it links them together using pointers. Each block contains the address of the next block in the file, allowing the operating system to access the entire file by following the chain of pointers.

The advantage of linked file allocation is that it can accommodate files of any size, as the file can be stored in multiple non-contiguous blocks. This method also avoids fragmentation, as files can be stored in any available free space on the disk, without the need to find a contiguous block of free space.

However, linked file allocation has some limitations. One significant disadvantage is that it can result in slower access times to files, as the operating system needs to follow the chain of pointers to access the entire file. This method may also require more disk space, as each block contains a pointer to the next block in the file. Additionally, if a pointer becomes damaged or lost, it can result in the loss of the entire file, as the operating system cannot access the entire chain of blocks.

**Indexed File Allocation**

To address some of the limitations, operating systems can use a variation of linked file allocation called indexed file allocation. In indexed file allocation, files are stored in non-contiguous blocks, but instead of linking each block together, the operating system creates an index block that contains a list of pointers to each block in the file. When a file is created, the operating system searches for a series of free blocks that are large enough to store the file and creates an index block that contains pointers to each of those blocks. Each block of the file is then stored in a separate block on the disk.

The advantage of indexed file allocation is that it provides fast access to files, as the operating system only needs to read the index block to locate the file's blocks. This method also avoids fragmentation, as files can be stored in any available free space on the disk, without the need to find a contiguous block of free space. Indexed file allocation also reduces the risk of data loss, as the index block can be duplicated to provide redundancy.

However, indexed file allocation has some limitations. One significant disadvantage is that it can result in wasted disk space, as the index block can take up a significant amount of space on the disk. This method also requires more disk space than linked file allocation, as each block of the file is stored separately on the disk.

There are several types of indexed file allocation methods used in computer operating systems, each with its own strengths and weaknesses −

* **Single-Level Index** − This method is the simplest form of indexed file allocation. In this method, a single index block is created for each file, and it contains pointers to the blocks that make up the file. This method is useful for small files but can become inefficient for larger files as the index block can take up a significant amount of space.
* **Multi-Level Index** − This method is an improvement over the single-level index method. In this method, multiple index blocks are used to store the pointers to the blocks that make up the file. The first level index block contains pointers to the second level index blocks, and so on. This method is useful for large files as it reduces the size of each index block and allows for faster access to the file.
* **Combined Index** − This method combines the benefits of both contiguous and indexed file allocation methods. In this method, a portion of the file is stored contiguously, and the rest is stored using indexed file allocation. The contiguous portion of the file is accessed quickly, while the indexed portion can accommodate files of any size.
* **Linked Index** − This method is similar to linked file allocation, but instead of linking blocks of the file together, an index block is created that contains pointers to the next index block. Each index block contains pointers to the data blocks that make up the file. This method is useful for large files, but it can result in slower access times to the file.
* **Inverted Index** − This method is used in databases to store indexes of records. In this method, a separate index block is created for each record type, and each block contains pointers to the data blocks that contain records of that type. This method is useful for fast access to specific types of records.

**File Allocation Table**

File Allocation Table (FAT) is a file system that uses a table to store information about the allocation of files on a disk or other storage media. In a FAT file system, the file allocation table is a data structure that contains a list of entries, each of which represents a block of storage space on the disk. The entries in the file allocation table indicate whether a block of storage space is free or allocated, and if it is allocated, they indicate which file or directory the block is associated with. When a file is created, the operating system searches for a series of free blocks of storage space on the disk and records the allocation of these blocks in the file allocation table. As the file is modified or expanded, the operating system updates the entries in the file allocation table to reflect the new allocation of blocks.

The FAT file system has several advantages. It is a simple and efficient file system that is well-suited to small disks and low-powered devices. It is also widely supported by many operating systems and can be used on a variety of storage media, including hard disks, floppy disks, and flash drives.

However, the FAT file system also has some limitations. It can be susceptible to file fragmentation, where files become fragmented across multiple non-contiguous blocks of storage space on the disk. This can slow down file access times and reduce overall system performance. Additionally, the file allocation table can become corrupted, leading to data loss or disk errors.

**Free Space Management**

A file system is responsible to allocate the free blocks to the file therefore it has to keep track of all the free blocks present in the disk. There are mainly two approaches by using which, the free blocks in the disk are managed.

## 1. Bit Vector

In this approach, the free space list is implemented as a bit map vector. It contains the number of bits where each bit represents each block.

If the block is empty then the bit is 1 otherwise it is 0. Initially all the blocks are empty therefore each bit in the bit map vector contains 1.

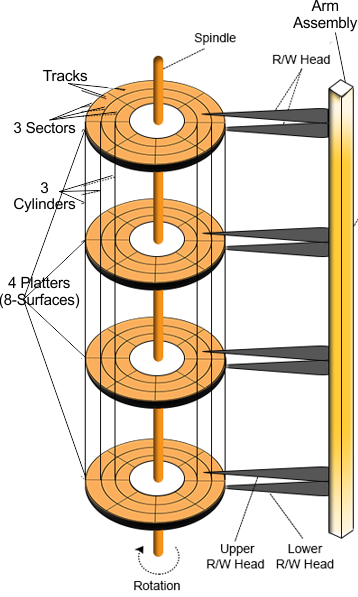
LAs the space allocation proceeds, the file system starts allocating blocks to the files and setting the respective bit to 0.

## 2. Linked List

It is another approach for free space management. This approach suggests linking together all the free blocks and keeping a pointer in the cache which points to the first free block.

Therefore, all the free blocks on the disks will be linked together with a pointer. Whenever a block gets allocated, its previous free block will be linked to its next free block.

A hard disk is a secondary storage that stores a large amount of data. The hard disk drive contains dozens of disks. These disks are also known as platters. These platters are mounted over the spindle, which rotates in any direction, i.e., clockwise or anti-clockwise. Let’s look at the hard disk structure in [OS](https://cstaleem.com/operating-system).



ckly and Easily Check Storage on Your Hard Drive

ay Video

**Platter**

The manufacturer constructs the Platter from aluminum or iron oxide. The platter diameter range is 1.8 inches to 5.25 inches.

Ezoic

* One surface of the Platter requires one Read/Write head, and a second R/W head is used for the other surface to store information.
* Every Platter holds the same no of tracks.
* Multiple platters increase the storage capacity.

Below is a descriptive diagram of a single platter.

**R/W Head**

R/W Heads move forth and back over the Platter surfaces to Read or Write the data on sectors. Read/Write heads do not touch the platter surface.

* The magnetic field writes data onto the platter surface.
* When the R/W head contacts the platter surface, it may create bad sectors.
* Had disk may damage due to these bad sectors.

Ezoic

**Tracks**

Circular areas of the disk are known as tracks.

* There may be more than 1000 tracks on a 3.5-inch hard disk and sector size.
* Track Numbering starts with zero at the outermost track.

**Sectors**

We further divide tracks into several small units, and these units are known as sectors.

Ezoic

* Sectors are the most minor physical storage units on disk.
* The size of each sector is almost always 512 Bytes.
* Sector Numbering starts with the number 1 at the outermost tracks.

**Cylinder**

All Corresponding tracks with the same radius of all platters in the Hard disk are known as cylinders. In simple words, we say

“Each track of all platters with the same radius is called a cylinder”.

So, the number of tracks on the Platter always equals the number of cylinders. For example, in a hard disk, where each Platter contains 600 tracks, the number of cylinders will also be 600 in the hard disk.

Cylinder Numbering starts with zero at the outermost cylinder.

**Cluster**

Cluster is also known as blocks. A group of sectors makes a cluster. There may be 64 or more sectors in a cluster. OS uses these clusters to Read/Write the data.

**Disk Capacity**

As we know, there are several platters in the hard disk. Each Platter contains two R/W heads. There are several cylinders/tracks in the hard disk. Multiple sectors divide each track. Each sector has some size, but most sectors are 512 Bytes.

Disk scheduling and management are essential components of a computer's operating system that handle the organization and access of data on a disk. Disk scheduling algorithms determine the order in which the read/write head of the disk moves to access data, which impacts the efficiency and speed of accessing data. Some commonly used disk scheduling algorithms include First-Come-First-Serve, Shortest Seek Time First, and SCAN. Disk management, on the other hand, involves tasks such as disk partitioning, formatting, and file system creation. It ensures that the disk is properly utilized, optimized for performance, and maintained to prevent data loss or corruption. Efficient disk scheduling and management are crucial for ensuring the smooth and effective operation of a computer system.

## Types of Disk Scheduling Algorithms

* **First-Come, First-Served (FCFS)** − The FCFS algorithm is the simplest disk scheduling algorithm, in which the requests are processed in the order they arrive. It is a nonpreemptive algorithm that does not take into account the distance between the current request and the next request.
* **Shortest Seek Time First (SSTF)** − The SSTF algorithm processes the request with the shortest seek time, i.e., the request that requires the least amount of movement by the disk's head. This algorithm can minimize the head movement, but it may result in starvation of some requests.
* **SCAN** − The SCAN algorithm processes the requests in a specific direction, either from the innermost track to the outermost track or vice versa. After reaching the end of the disk, the head reverses direction and processes the requests in the opposite direction. This algorithm provides a balance between the number of requests processed and the time each request waits.
* **C-SCAN** − The C-SCAN algorithm is similar to the SCAN algorithm, but the head moves only in one direction and processes requests only in that direction. When it reaches the end of the disk, it immediately returns to the beginning of the disk and starts processing requests again. This algorithm can eliminate the possibility of starvation.
* **LOOK** − The LOOK algorithm is similar to the SCAN algorithm, but the head reverses direction when it reaches the last request in the current direction. This algorithm can reduce the waiting time for requests that are close to the head's current position.
* **C-LOOK** − The C-LOOK algorithm is similar to the C-SCAN algorithm, but the head reverses direction when it reaches the last request in the current direction. This algorithm can reduce the waiting time for requests that are close to the head's current position and eliminate the possibility of starvation.

## Evaluation Criteria for Disk Scheduling Algorithms

* **Throughput** − Throughput is the number of input/output operations that can be processed per unit time. A disk scheduling algorithm with high throughput can process more requests in a shorter time, resulting in higher system performance.
* **Turnaround time** − Turnaround time is the time taken to process a request from the time it is submitted to the time it is completed. A disk scheduling algorithm with a low turnaround time can provide faster service to the users.
* **Waiting time** − Waiting time is the time that a request spends waiting in the queue before it is processed. A disk scheduling algorithm with a low waiting time can provide faster service to the users.
* **Response time** − Response time is the time taken to provide the first response to a user's request. A disk scheduling algorithm with a low response time can provide faster service to the users.
* **Fairness** − Fairness refers to the equal treatment of all requests in the queue. A disk scheduling algorithm that provides fair treatment to all requests can ensure that no request is starved of resources.

## Selection of Disk Scheduling Algorithm

* **Deterministic** − In deterministic disk scheduling, the algorithm is selected based on the characteristics of the workload. For example, if the workload consists of short requests, the SSTF algorithm may be the best choice.
* **Dynamic** − In dynamic disk scheduling, the algorithm is selected based on the current state of the system. For example, if the disk is heavily loaded, the operating system may switch to a more efficient algorithm to handle the increased workload. Dynamic disk scheduling can adapt to changes in the workload and improve system performance.

The selection of the disk scheduling algorithm depends on several factors, including the characteristics of the workload, system performance requirements, and the available resources. The operating system must evaluate the different disk scheduling algorithms based on the evaluation criteria and select the algorithm that best meets the system's requirements.

### Advantages

* **Improved performance** − Disk scheduling algorithms ensure that data is accessed in the most efficient manner possible, which improves the performance of the system.
* **Fairness** − Disk scheduling algorithms ensure that all requests for data access are treated fairly and given a chance to be processed.
* **Reduced disk fragmentation** − Disk scheduling algorithms can help to reduce disk fragmentation by accessing data in a more organized manner.

### Disadvantages

* **Overhead** − Disk scheduling algorithms can create overhead and delay in processing data access requests, which can reduce overall system performance.
* **Complexity** − Some disk scheduling algorithms can be complex and difficult to understand, which may make it difficult to optimize system performance.
* **Risk of starvation** − Disk scheduling algorithms can result in starvation of certain requests, which can lead to inefficiencies and reduced system performance.

## Disk Management

## Introduction

Disk management is the process of organizing, optimizing, and maintaining data on a disk within a computer system. It involves tasks such as disk partitioning, formatting, and file system creation to ensure efficient use of disk space, prevent data loss, and improve system performance. Proper disk management is crucial for preventing issues such as disk failure, fragmentation, and data corruption. By allocating space to files and creating a file system, disk management helps to organize data and improve the efficiency and speed of accessing it.

## Techniques for Disk Management

* **File Allocation Table (FAT)** − FAT is a file system used by many operating systems, including Windows and some versions of Linux. It uses a table to keep track of the location of files on the disk, and it supports long file names and basic file security features.
* **New Technology File System (NTFS)** − NTFS is a file system used by Windows operating systems that offers advanced features, including file compression, encryption, and disk quotas. It also provides better performance and reliability compared to FAT.
* **Unix File System (UFS)** − UFS is a file system used by Unix and Unix-like operating systems, including Linux and macOS. It supports features such as file permissions, symbolic links, and journaling, which provides faster recovery from disk errors.
* **ZFS** − ZFS is a file system used by some operating systems, including FreeBSD and Open Solaris. It provides advanced features such as data compression, snapshotting, and automatic repair of data errors.
* **Disk Quotas** − Disk quotas are used to limit the amount of disk space that a user or group can use. This helps to prevent one user from monopolizing disk space and can improve system performance and reliability.

## Disk Optimization Techniques

* **Defragmentation** − Defragmentation is the process of rearranging files on the disk to reduce fragmentation, which can slow down disk access times. Defragmentation can improve disk performance and extend the life of the disk.
* **Compression** − Compression is the process of reducing the size of files to save disk space. It can be used to compress files that are not frequently accessed, such as archives or backups. Compression can improve disk utilization and reduce storage costs.
* **Encryption** − Encryption is the process of encoding data to prevent unauthorized access. It can be used to protect sensitive data, such as financial or personal information. Encryption can improve data security and prevent data breaches.

### Advantages

* **Data organization** − Disk management helps to organize data on a disk by allocating space to files and creating a file system. This improves the efficiency and speed of accessing data.
* **Prevent data loss** − Disk management includes features such as disk partitioning, formatting, and backup. This helps to prevent data loss and recover lost data in the event of disk failure or other issues.
* **Optimized performance** − Proper disk management ensures that the disk is optimized for performance, which improves the speed and efficiency of data access.

### Disadvantages

* **Complexity** − Disk management can be complex and difficult to understand, especially for novice users. This can result in errors or improper disk configurations that may cause data loss or corruption.
* **Time-consuming** − Disk management tasks such as disk partitioning and formatting can be time-consuming, especially for large disks or complex configurations.
* **Risk of data loss** − Improper disk management can result in data loss or corruption, which can be difficult or impossible to recover.

Disk attachment is the process of connecting a storage device, such as a hard disk drive or solid-state drive, to a computer system. This process is essential for the proper functioning of an operating system as it allows the system to read and write data to the storage device. Disk attachment can be either internal or external, and there are several methods of attachment, including SATA, SCSI, and SAS. In this article, we will explore the different types of disk attachments, methods of attachment, disk formatting and partitioning, and disk management.

Definition

Disk attachment refers to the process of physically connecting a storage device, such as a hard disk drive or solid-state drive, to a computer system. The purpose of disk attachment is to enable the computer system to read and write data to the storage device.

Types of disk attachment

* Internal disk attachment
* External disk attachment
* Network Attached Storage
* Storage Area Network

**Internal disk attachment**

Definition

Internal disk attachment refers to the process of connecting a storage device directly to the motherboard of a computer system. This type of attachment is typically used for storage devices that are intended to be permanent components of the computer system, such as the primary hard disk drive.

Advantages

* **Faster data transfer speeds** − Internal disk attachment provides faster data transfer speeds compared to external attachment methods, such as USB or FireWire.
* **Better power management** − Internal storage devices can be more easily managed by the operating system's power management features, allowing for more efficient power usage.
* **More secure** − Since internal storage devices are physically connected to the motherboard, they are less likely to be accidentally disconnected or removed.

Disadvantages

* **Limited expansion** − Internal disk attachment limits the number of storage devices that can be connected to a computer system. This can be problematic for users who require a large amount of storage space.
* **The difficulty of access** − Since internal storage devices are located inside the computer system, accessing them for upgrades or repairs can be more difficult and time-consuming.
* **Higher cost** − Internal storage devices can be more expensive than external devices due to their higher performance and reliability requirements.

**External disk attachment**

Definition

External disk attachment refers to the process of connecting a storage device to a computer system via an external port, such as USB, Thunderbolt, or FireWire. This type of attachment is typically used for storage devices that are intended to be portable, such as external hard drives or USB flash drives.

**Advantages**

* **Portability** − External storage devices can be easily transported and used on multiple computer systems, making them ideal for users who require access to their data on the go.
* **Ease of access** − External storage devices are located outside the computer system, making them easy to access for upgrades or repairs.
* **Expandability** − External storage devices can be easily added or removed from a computer system, allowing for more storage space as needed.

**Disadvantages**

* **Slower data transfer speeds** − External disk attachments typically provide slower data transfer speeds compared to internal attachment methods, such as SATA.
* **Limited power management** − External storage devices may not be as easily managed by the operating system's power management features, leading to less efficient power usage.
* **Less secure** − External storage devices can be accidentally disconnected or removed, leading to potential data loss or corruption.

**Network Attached Storage**

Definition

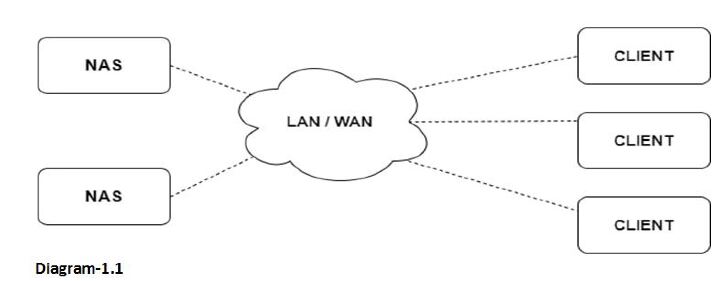
Network-attached storage (NAS) is a type of storage architecture where storage devices are connected to a network and provide file-level access to multiple clients or users. NAS devices are typically dedicated devices that contain one or more hard drives or solid-state drives, and they are connected to the network using standard Ethernet or Wi-Fi connections.

Advantages

* **Easy to set up and manage** − NAS devices are designed to be user-friendly, and they can be easily configured and managed using a web-based interface.
* **Cost-effective** − NAS devices are typically less expensive than other storage architectures, such as Storage Area Networks (SANs), and they can offer high-capacity storage for a relatively low cost.
* **Centralized storage** − NAS devices provide a centralized storage location that can be accessed by multiple users or devices on the network, which can be useful for sharing files and backing up data.

Disadvantages

* **Limited performance** − NAS devices may not offer the same level of performance as other storage architectures, such as SANs, especially for high-performance applications.
* **Limited scalability** − NAS devices may be limited in terms of scalability, especially for larger enterprise environments.
* **Network dependency** − NAS devices rely on network connectivity, which can be a potential point of failure or a bottleneck for storage access.



**Storage Area Network**

Definition

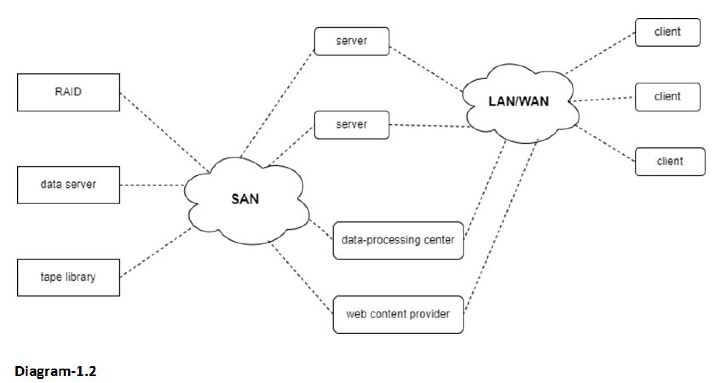
A Storage Area Network (SAN) is a specialized network that provides block-level access to storage devices, such as hard disk drives (HDDs), solid-state drives (SSDs), or tape libraries. SANs are designed to provide high-speed, low-latency storage access for servers or hosts, and they can be used to build complex storage infrastructures for enterprise data centers.

Advantage

SANs offer several advantages over other storage architectures. They can provide high-speed, lowlatency access to storage devices, which can be critical for high-performance applications such as databases or virtualized environments.

Disadvantage

SANs can also be complex and expensive to implement and maintain, and they may require specialized skills and expertise to configure and manage. They also require a dedicated network infrastructure, which can add to the overall cost and complexity of the storage infrastructure.



Disk attachment methods

SATA

Definition

Serial ATA (SATA) is a standard for connecting storage devices to a computer system. SATA uses a serial connection and is commonly used for connecting internal hard disk drives and solid-state drives.

**Advantages**

* **Faster data transfer speeds** − SATA provides faster data transfer speeds compared to older parallel ATA (PATA) standards.
* **Higher storage capacity** − SATA supports larger storage devices than PATA, allowing for more data to be stored on a single device.

**Disadvantages**

* **Limited cable length** − SATA cables are limited in length, which can be problematic for larger computer systems.
* **The limited number of devices** − SATA only supports a limited number of devices per controller, which can be problematic for users who require a large amount of storage space.

**SCSI**

Definition

Small Computer System Interface (SCSI) is a standard for connecting storage devices to a computer system. SCSI uses a parallel connection and is commonly used for connecting highperformance storage devices, such as hard disk drives and solid-state drives.

Advantages

* **High data transfer speeds** − SCSI provides high data transfer speeds compared to older standards, such as PATA.
* **Support for multiple devices** − SCSI supports a large number of devices per controller, making it ideal for users who require a large amount of storage space.

Disadvantages

* **Higher cost** − SCSI devices can be more expensive than other attachment methods due to their higher performance and reliability requirements.
* **Limited compatibility** − SCSI devices may not be compatible with all computer systems, which can be problematic for users who require a high-performance storage solution.

**SAS**

Definition

Serial Attached SCSI (SAS) is a standard for connecting storage devices to a computer system. SAS uses a serial connection and is commonly used for connecting high-performance storage devices, such as hard disk drives and solid-state drives.

**Advantages**

* **High data transfer speeds** − provides high data transfer speeds compared to older standards, such as PATA.
* **Support for multiple devices** − SAS supports a large number of devices per controller, making it ideal for users who require a large amount of storage space.

**Disadvantages**

* **Higher cost** − SAS devices can be more expensive than other attachment methods due to their higher performance and reliability requirements.
* **Limited compatibility** − SAS devices may not be compatible with all computer systems, which can be problematic for users who require a high-performance storage solution.

**Importance of Disk Attachment in OS**

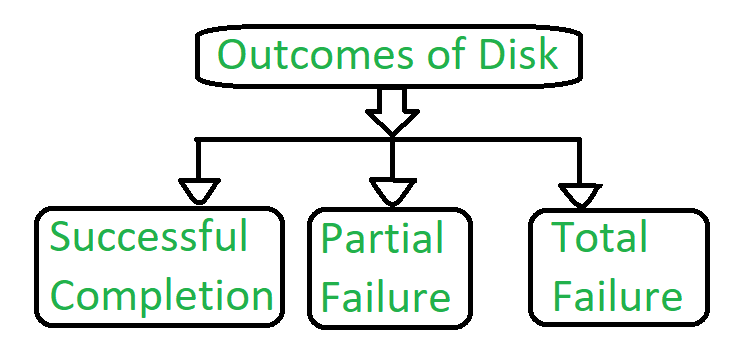
* **Data storage** − Disk attachment is necessary for storing data on a computer system. Without disk attachments, it would be impossible to save files or install software on the computer.
* **Performance** − Disk attachment plays a critical role in system performance. Faster and more efficient disk attachment technologies, such as Serial Attached SCSI (SAS), can improve the speed and responsiveness of the system.
* **Scalability** − As data storage needs increase, disk attachment technologies provide scalability by allowing additional disks to be added to the system. This can be particularly important for businesses and organizations that need to store large amounts of data.
* **Redundancy** − Disk attachment technologies can provide redundancy and failover capabilities to ensure that data remains accessible even in the event of disk failure.
* **Data protection** − Disk attachment technologies can provide data protection features such as RAID (Redundant Array of Independent Disks) to protect against data loss due to disk failure.

**Stable-Storage Implementation:** 

By definition, information residing in the **Stable-Storage** is never lost. Even, if the disk and CPU have some errors, it will never lose any data.

To achieve such storage, we need to replicate the required information on multiple storage devices with independent failure modes. The writing of an update should be coordinate in such a way that it would not delete all the copies of the state and when we are recovering from a failure we can force all the copies to a consistent and correct valued even if another failure occurs during the recovery. In this article we will discuss how to cover these needs.

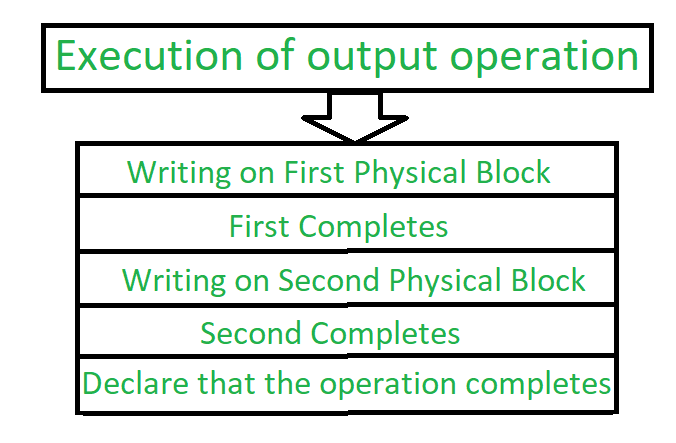
The disk write operation results to one of the following outcome:

  
**Figure –** Outcomes of Disk

1. **Successful completion –**   
   The data will be written correctly on the disk.
2. **Partial Failure –**   
   In this case, failure has occurred in the middle of the data transfer, such that only some sectors were written with the new data, and the sectors which were written during the failure may have been corrupted.
3. **Total Failure –**   
   The failure occurred before the disk write started, so the previous data values on the disk remains intact.

During writing a block somehow if failure occurs, the system’s first work is to detect the failure and then invoke a recovery process to restore the consistent state. To do that, the system must contain two physical block for each logical block.

An output operation is executed as follows:

  
**Figure –** Process of execution of output operation

1. Write the information onto the first physical block.
2. When the first write completes successfully, perform the same operation onto the second physical block.
3. When both the operations are successful, declare the operation as complete.

During the recovery from a failure each of the physical block is examined. If both are the same and no detectable error exists, then no further action is necessary. If one block contains detectable errors then we replace its content with the value of the other block. If neither block contains the detectable error, but the block differ in content, then we replace the content of first block with the content of the second block. This procedure of the recovery give us an conclusion that either the write to stable content succeeds successfully or it results in no change.

This procedure will be extended if we want arbitrarily large number of copies of each block of the stable storage. With the usage of large number of copies, the chances of the failure reduces. Generally, it is reasonable to simulate stable storage with only two copies. The data present in the stable storage is safe unless a failure destroys all the copies. The data that is present in the stable storage is guaranteed to be safe unless a failure destroys all the copies.

Because waiting for disk writes to complete is time consuming, many storage arrays add NVRAM as a cache. Since the memory is non-volatile it can be trusted to store the data in route to the disks. In this way it is considered as a part of the stable storage. Writing to the stable storage is much faster than to the disk, so performance is greatly improved.