Unit-04
File System Interface

By

Balaram Das

Department of EE & EEE, GIET University, Gunupur

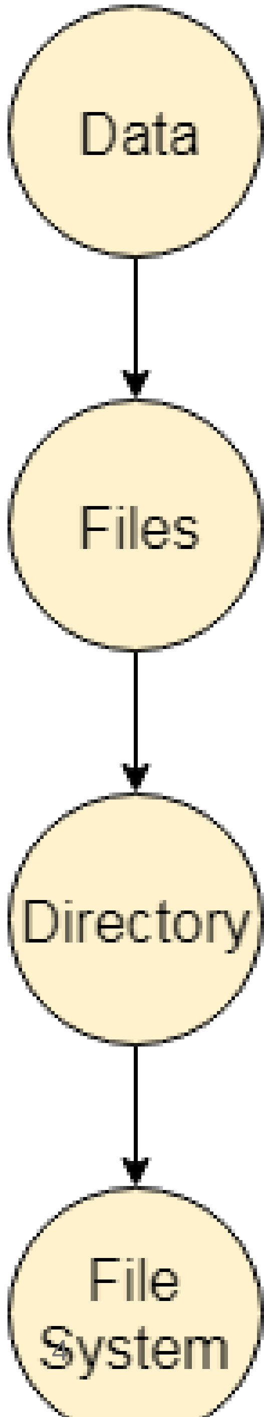
Introduction

- The memory manager is responsible for maintaining primary memory, while the file manager is responsible for maintaining secondary storage (e.g., hard disks).
- A file is a collection of related information recorded on secondary storage such as magnetic disks, tapes, and optical disks.

The Concept of a File

- From the user's perspective, a file is the smallest allotment of logical secondary storage.
- File System consists of
- (a) A collection of files
- (b) A directory structure
- (c) (possibly) partitions

- Files are stored in a file system, which may exist on a disk or in the main memory. Files can be simple (plain text) or complex (specially-formatted).
- The collection of files is known as Directory. The collection of directories at the different levels, is known as File System.



Attributes of the File

- **1.Name:** Every file carries a name by which the file is recognized in the file system. One directory cannot have two files with the same name.
- **2.Identifier:** Along with the name, Each File has its own extension which identifies the type of the file. For example, a text file has the extension **.txt,** A video file can have the extension **.mp4.**
- **3.Type:** In a File System, the Files are classified in different types such as video files, audio files, text files, executable files, etc.

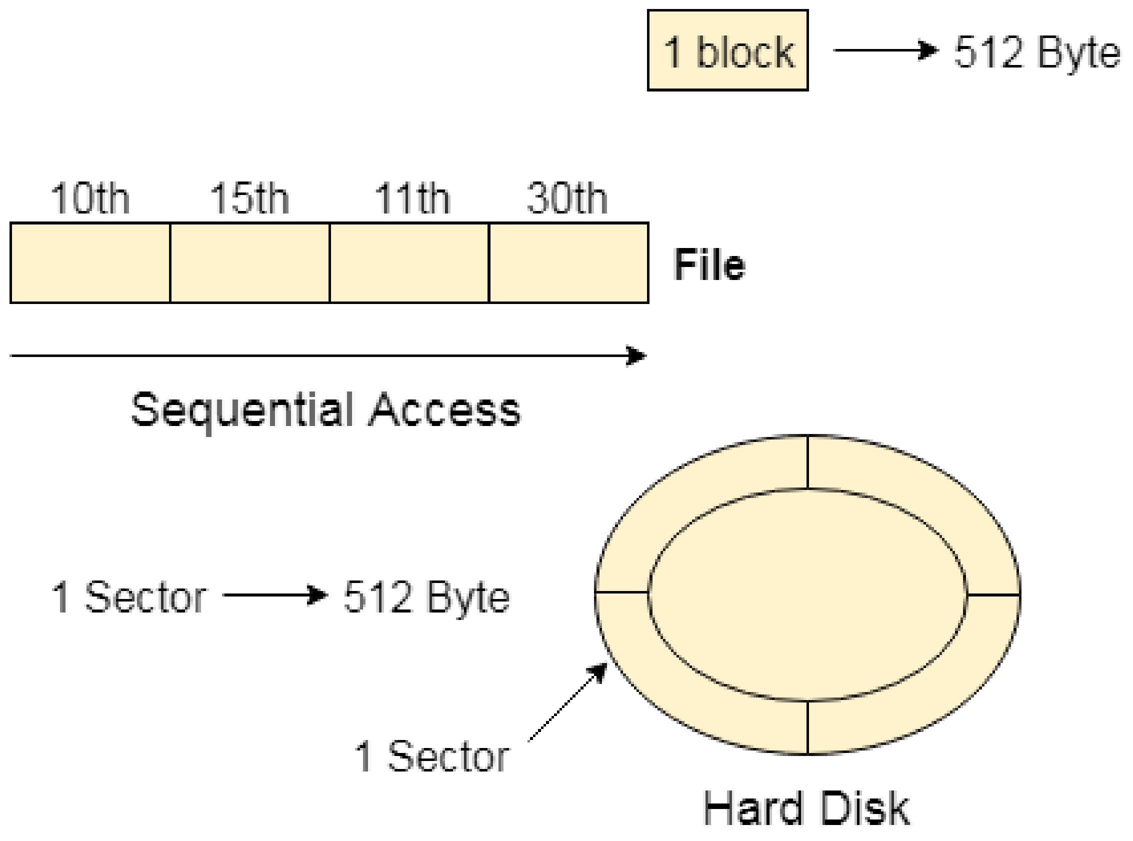
- **4.Location:** In the File System, there are several locations on which, the files can be stored. Each file carries its location as its attribute.
- **5.Size:** The Size of the File is one of its most important attribute. By size of the file, we mean the number of bytes acquired by the file in the memory.
- **6.Protection:** The Admin of the computer may want the different protections for the different files. Therefore, each file carries its own set of permissions to the different group of Users.
- **7.Time and Date:** Every file carries a time stamp which contains the time and date on which the file is last modified.

- **Advantages of File System**
- **Organization:** A file system allows files to be organized into directories and subdirectories, making it easier to manage and locate files.
- **Data protection:** File systems often include features such as file and folder permissions, backup and restore, and error detection and correction, to protect data from loss or corruption.
- **Improved performance:** A well-designed file system can improve the performance of reading and writing data by organizing it efficiently on disk.

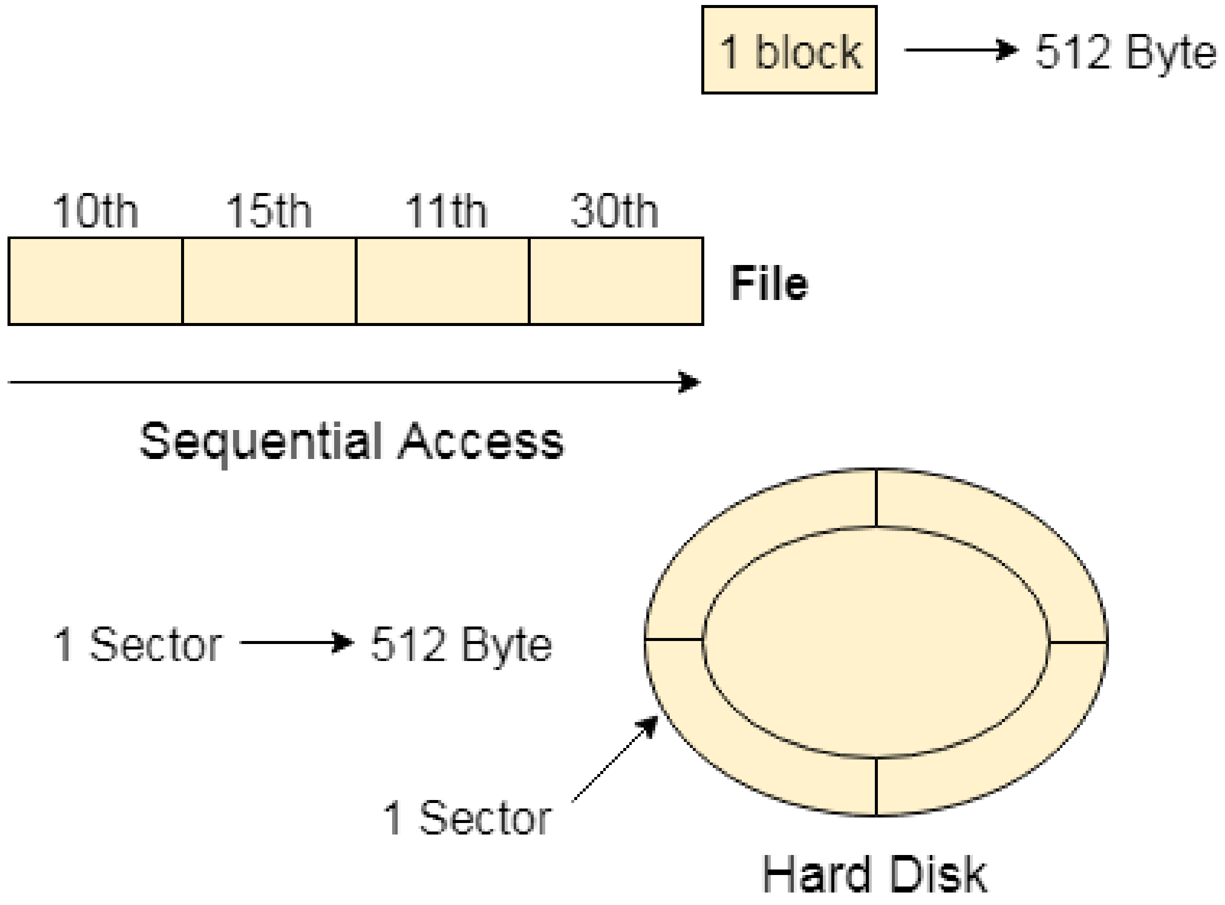
- **Disadvantages of File System**
- **Compatibility issues:** Different file systems may not be compatible with each other, making it difficult to transfer data between different operating systems.
- **Disk space overhead:** File systems may use some disk space to store metadata and other overhead information, reducing the amount of space available for user data.
- **Vulnerability:** File systems can be vulnerable to data corruption, malware, and other security threats, which can compromise the stability and security of the system.

File Access Methods

- Files stores information, this information must be accessed and read into computer memory.
- There are so many ways that the information in the file can be accessed.



- **1. Sequential file access:**
- Information in the file is processed in order i.e. one record after the other.
- Magnetic tapes are supporting this type of file accessing.
- Eg: A file consisting of 100 records, the current position of read/write head is 45th record, suppose we want to read the 75th record then, it access sequentially from 45, 46, 47 …….. 74, 75. So the read/write head traverse all the records between 45 to 75.



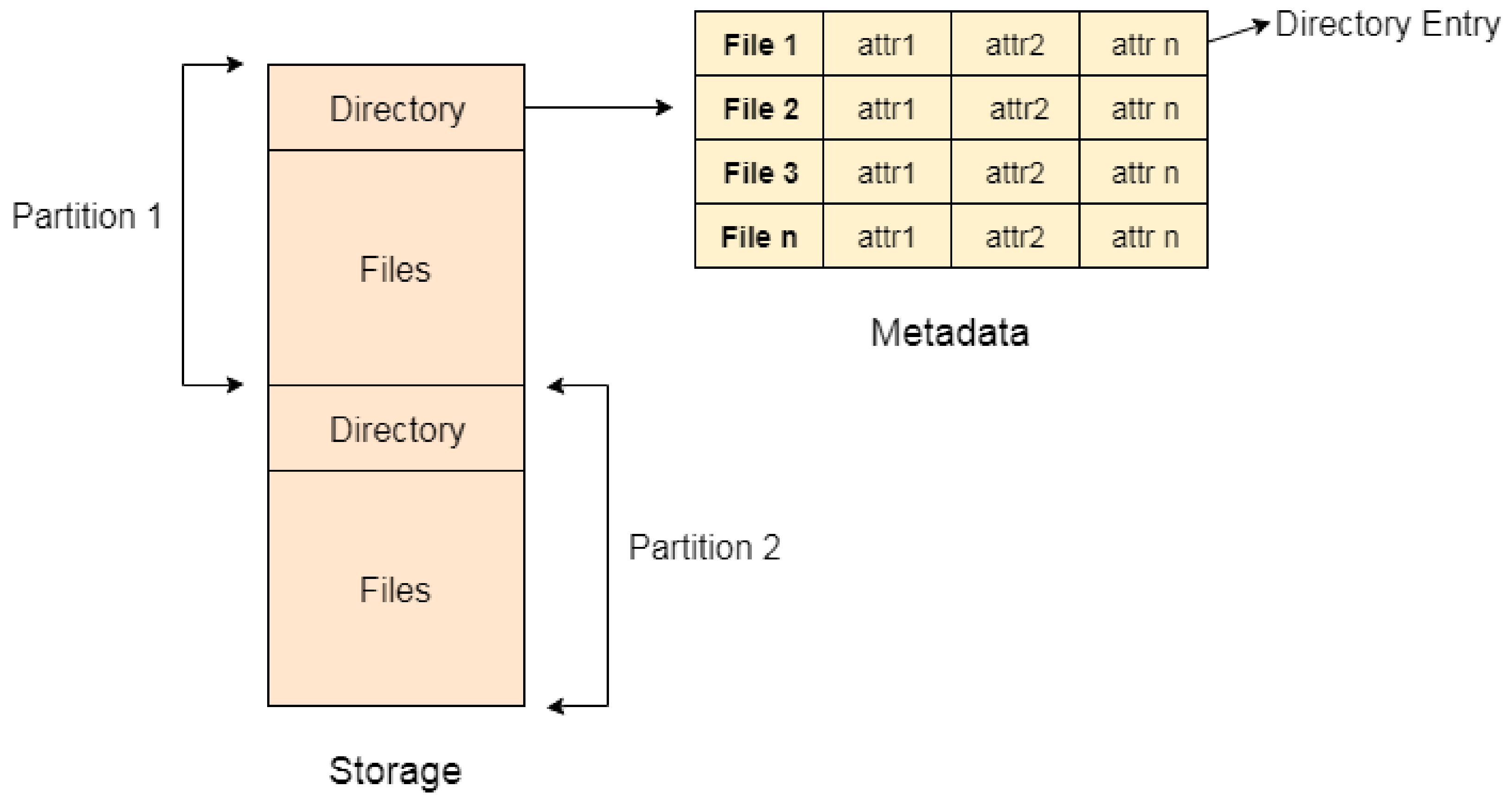
- **2. Direct access:**
- Direct access is also called relative access. Here records can read/write randomly without any order. The direct access method is based on a disk model of a file, because disks allow random access to any file block.
- Ex: CD consists of 10 songs, at present we are listening song 3, If we want to listen song 10, we can shift to 10.

- **3. Indexed Sequential File access**
- The main disadvantage in the sequential file is, it takes more time to access a Record .Records are organized in sequence based on a key field.
- Ex: A file consisting of 60000 records, the master index divide the total records into 6 blocks, each block consisting of a pointer to secondary index.
- The secondary index divide the 10,000 records into 10 indexes.
- Each index consisting of a pointer to its original location. Each record in the index file consisting of 2 field, A key field and a pointer field.

Directory Structure

- Directory can be defined as the listing of the related files on the disk.
- The directory may store some or the entire file attributes.
- To get the benefit of different file systems on the different operating systems, A hard disk can be divided into the number of partitions of different sizes.
- The partitions are also called volumes or mini disks.

- Each partition must have at least one directory in which, all the files of the partition can be listed.
- A directory entry is maintained for each file in the directory which stores all the information related to that file.

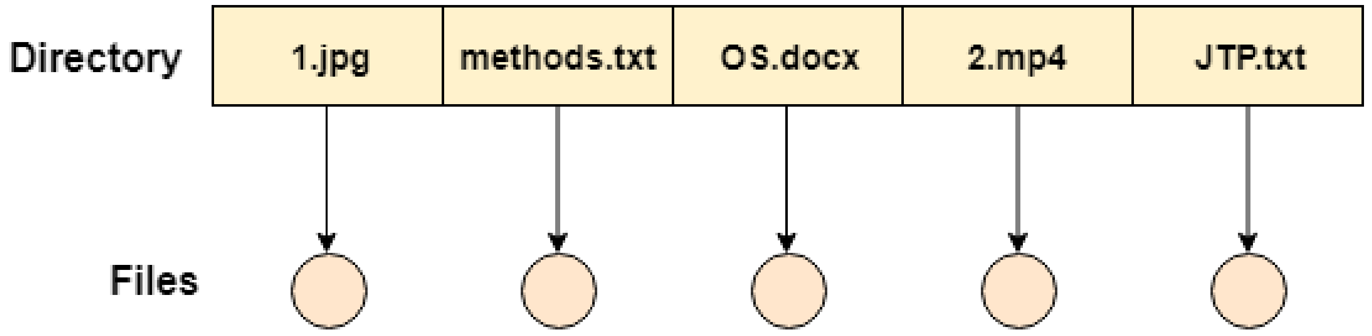

Partition 1
Directory
Files
Directory
Files
Partition 2
Storage
File 1 attr1 attr2 attr n
File 2 attr1 attr2 attr n
File 3 attr1 attr2 attr n
File n attr1 attr2 attr n
Directory Entry
Metadata

- A directory can be viewed as a file which contains the Meta data of the bunch of files.
- Every Directory supports a number of common operations on the file:

1. File Creation
2. Search for the file
3. File deletion
4. Renaming the file
5. Traversing Files
6. Listing of files

Single Level Directory

- The simplest method is to have one big list of all the files on the disk.
- The entire system will contain only one directory which is supposed to mention all the files present in the file system.
- The directory contains one entry per each file present on the file system.



Single Level Directory

- This type of directories can be used for a simple system.
- **Advantages**

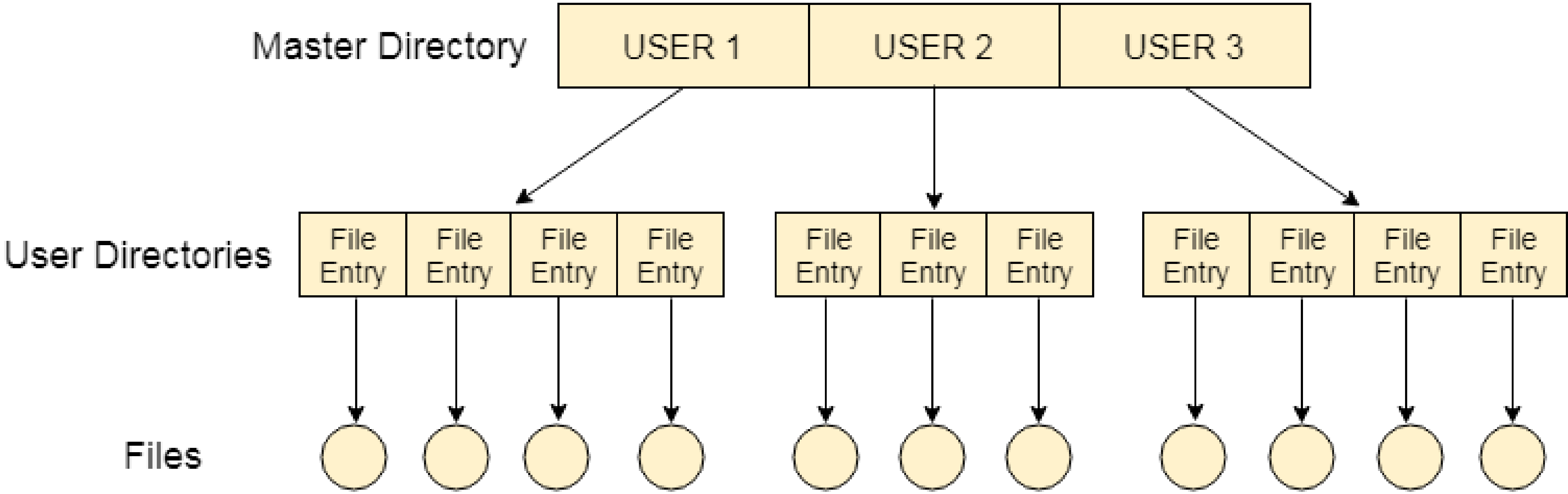
1. Implementation is very simple.
2. If the sizes of the files are very small, then the searching becomes faster.
3. File creation, searching, deletion is very simple since we have only one directory.

- **Disadvantages**

1. We cannot have two files with the same name.
2. The directory may be very big therefore searching for a file may take so much time.
3. Protection cannot be implemented for multiple users.
4. There are no ways to group same kind of files.
5. Choosing the unique name for every file is a bit complex.

Two Level Directory

- In two level directory systems, we can create a separate directory for each user.
- There is one master directory which contains separate directories dedicated to each user.
- For each user, there is a different directory present at the second level, containing group of user's file.
- The system doesn't let a user to enter in the other user's directory without permission.



Two Level Directory

Characteristics of two-level directory system

1. Each files has a path name as ***/User-name/directory-name/***

2. Different users can have the same file name.

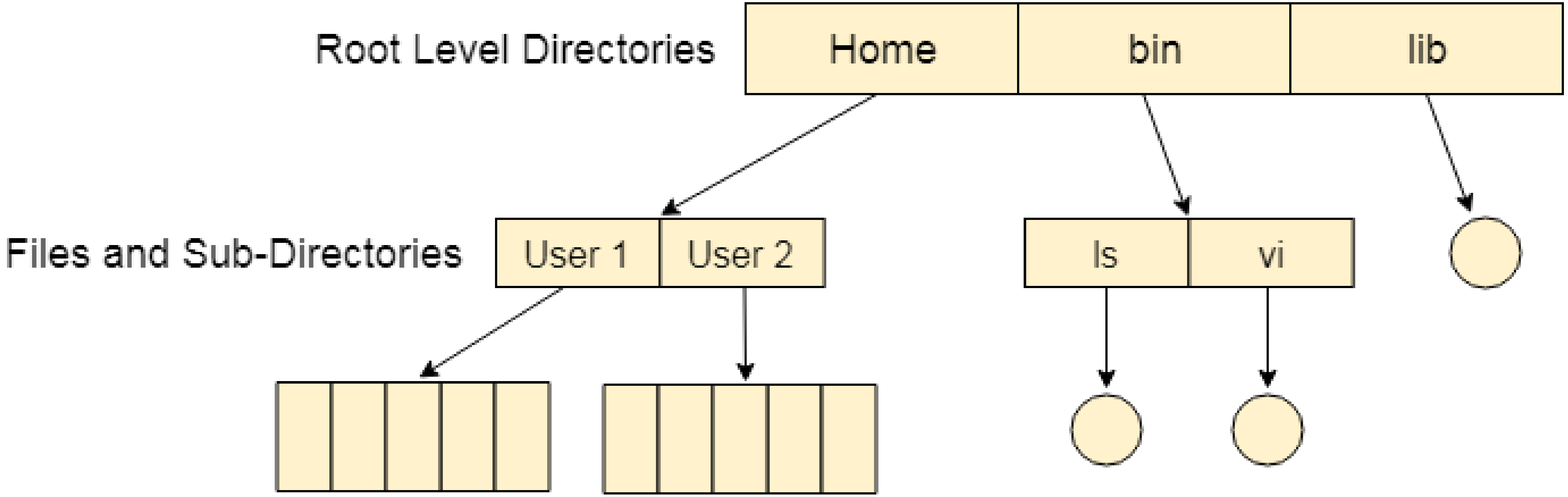
3. Searching becomes more efficient as only one user's list needs to be traversed.

4. The same kind of files cannot be grouped into a single directory for a particular user.

Tree Structured Directory

- In Tree structured directory system, any directory entry can either be a file or sub directory.
- Tree structured directory system overcomes the drawbacks of two-level directory system.
- The similar kind of files can now be grouped in one directory.
- Each user has its own directory, and it cannot enter in the other user's directory.
- However, the user has the permission to read the root's data but he cannot write or modify this.
- Only administrator of the system has the complete access of root directory.

- Searching is more efficient in this directory structure.
- The concept of current working directory is used.
- A file can be accessed by two types of path, either relative or absolute.
- Absolute path is the path of the file with respect to the root directory of the system while relative path is the path with respect to the current working directory of the system.
- In tree structured directory systems, the user is given the privilege to create the files as well as directories.



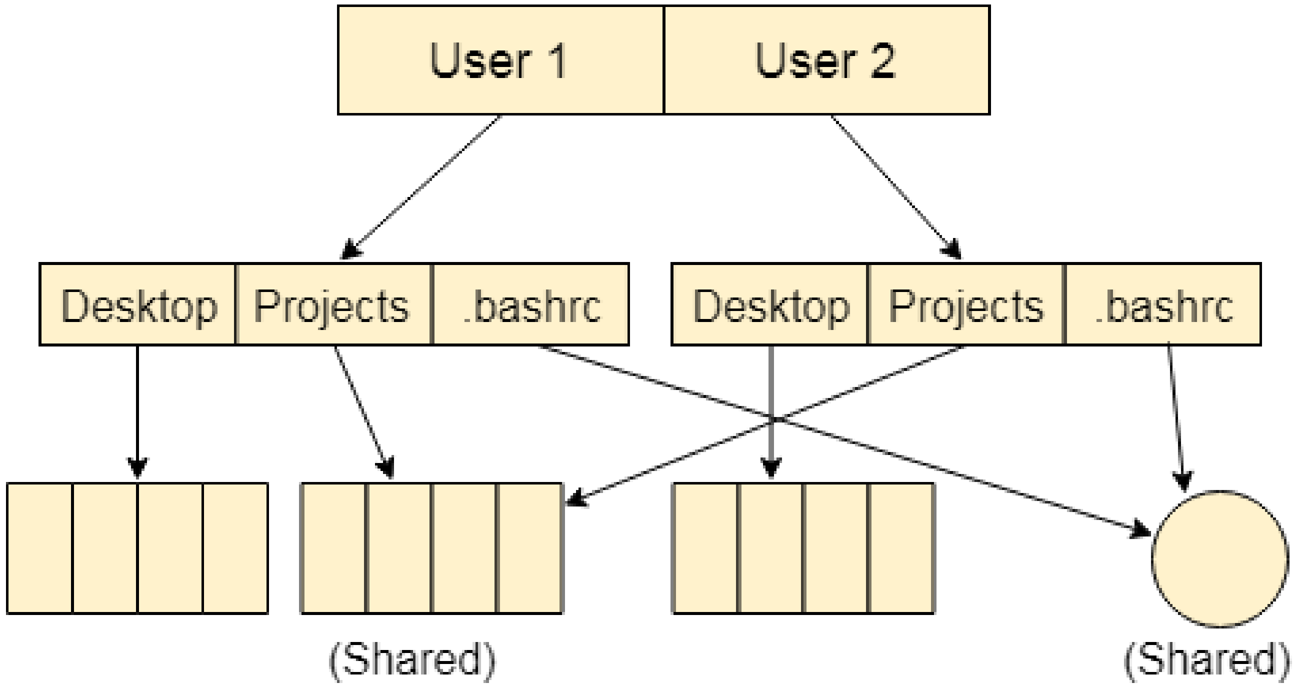
The Structured Directory System

Acyclic-Graph Structured Directories

- The tree structured directory system doesn't allow the same file to exist in multiple directories therefore sharing is major concern in tree structured directory system.
- We can provide sharing by making the directory an acyclic graph.
- In this system, two or more directory entry can point to the same file or sub directory.
- That file or sub directory is shared between the two directory entries.

- These kinds of directory graphs can be made using links.
- We can have multiple paths for a same file.
- Links can either be symbolic (logical) or hard link (physical).

- If a file gets deleted in acyclic graph structured directory system, then
- 1. In the case of soft link, the file just gets deleted and we are left with a dangling pointer.
- 2. In the case of hard link, the actual file will be deleted only if all the references to it gets deleted.



Acyclic-Graph Structured Directory System

File System Mounting

- It allows the users to organize and access files from different storage devices.
- If we compare it to a real-life scenario, it is closely related to “Connecting puzzles to get a complete picture of data”.

- Mounting is a process in which the operating system adds the directories and files from a storage device to the user's computer file system.
- The file system is attached to an empty directory, by adding so the system user can access the data that is available inside the storage device through the system file manager.
- Storage systems can be internal hard disks, external hard disks, USB flash drivers, SSD cards, memory cards, network-attached storage devices, CDs and DVDs, remote file systems, or anything else.

Terminologies used in File System Mounting

- **File System:** It is the method used by the operating system to manage data storage in a storage device. So, a user can access and organize the directories and files in an efficient manner.
- **Device name:** It is a name/identifier given to a storage partition. In windows, for example, “D:” in windows.
- **Mount point**: It is an empty directory in which we are adding the file system during the process of mounting.

File Sharing

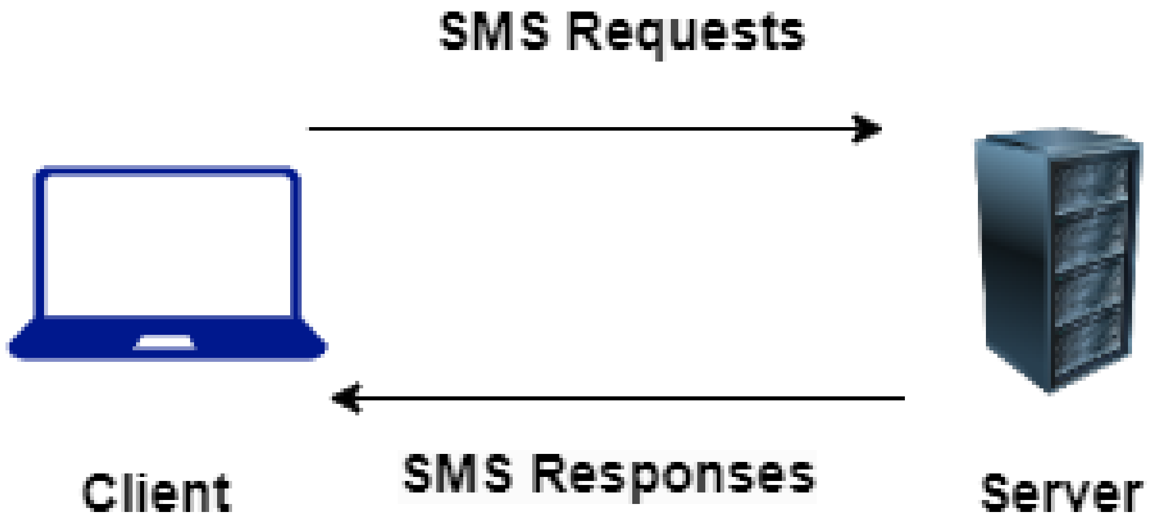
- File Sharing in an Operating System(OS) denotes how information and files are shared between different users, computers, or devices on a network.
- Files are units of data that are stored in a computer in the form of documents/images/videos or any other types of information needed.
- File sharing can be done using email attachments, cloud services, etc.

Primary Terminology Related to File Sharing

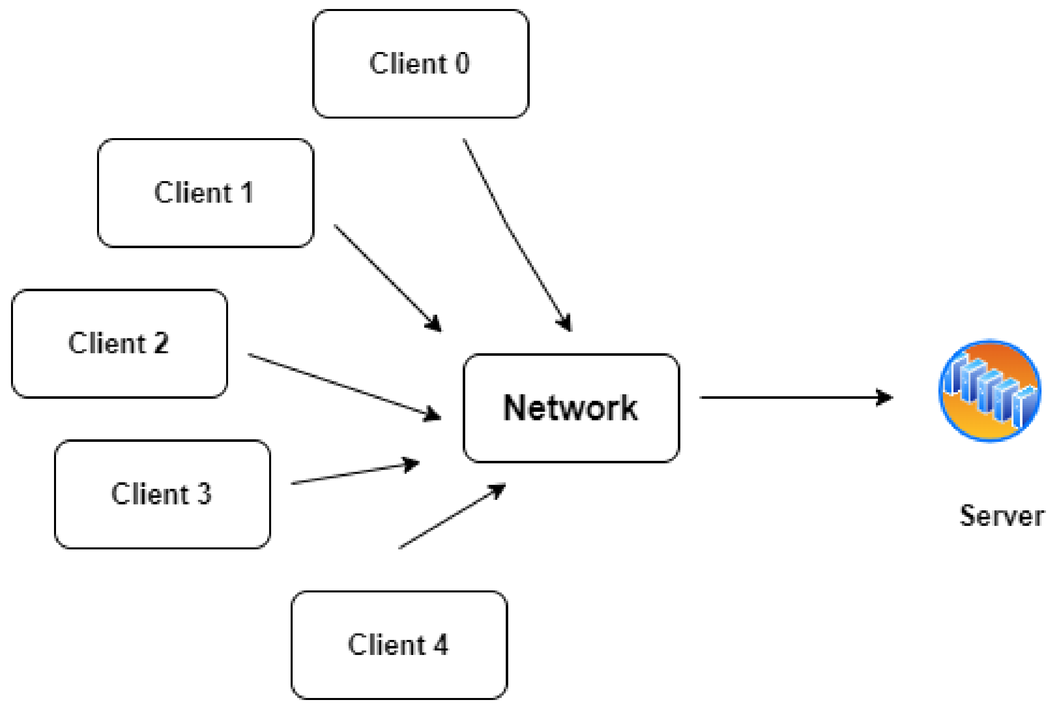
- **Folder/Directory:** It is basically like a container for all our files on a computer.
- The folder can contain files and even other folders maintaining like hierarchical structure for organizing data.

- **Networking:** It is involved in connecting computers or devices where we need to share the resources. Networks can be local (LAN) or global (Internet).
- **IP Address:** It is numerical data given to every connected device on the network
- **Protocol:** It is given as the set of rules which drives the communication between devices on a network. In the context of file sharing, protocols define how files are transferred between computers.
- **File Transfer Protocol (FTP):** FTP is a standard network protocol used to transfer files between a client and a server on a computer network.

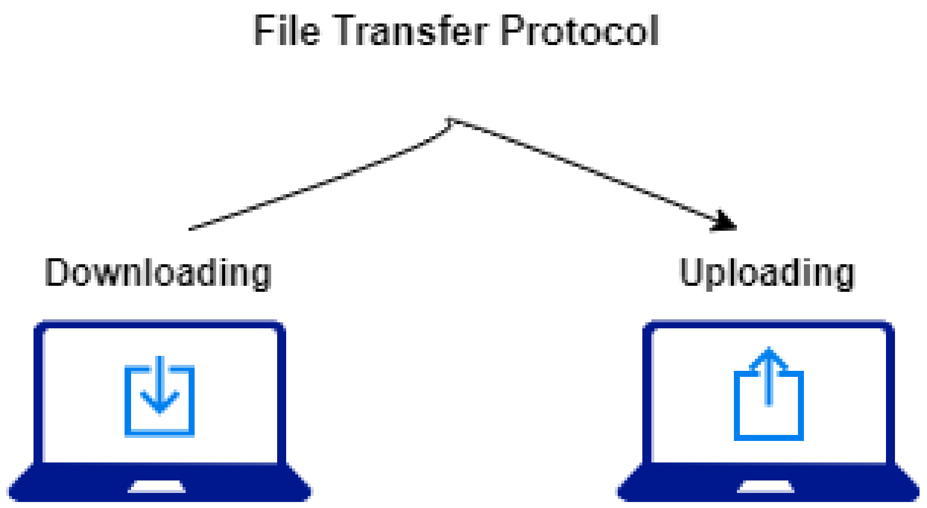
- **Various Ways to Achieve File Sharing**
- **1. Server Message Block (SMB)**
- SMB is like a network-based file-sharing protocol mainly used in Windows operating systems.
- It allows our computer to share files/printers on a network. SMB is now the standard way for seamless file transfer methods and printer sharing.



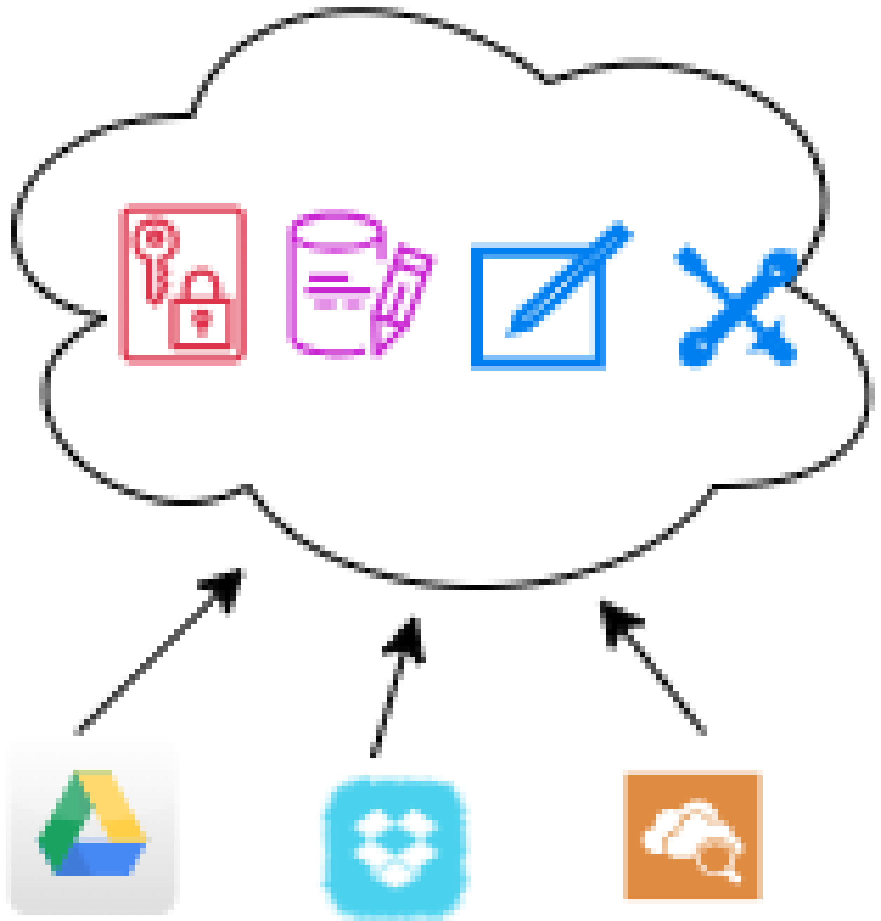
- **2. Network File System (NFS)**
- NFS is a distributed-based file-sharing protocol mainly used in Linux/Unix-based operating Systems.
- It allows a computer to share files over a network as if they were based on local.
- It provides an efficient way of transferring files between servers and clients.



- **3. File Transfer Protocol (FTP)**
- It is the most common standard protocol for transferring files between a client and a server on a computer network.
- FTPs support both uploading and downloading of files, here we can download, upload, and transfer files from Computer A to Computer B over the internet or between computer systems.



- **4. Cloud-Based File Sharing**
- It involves the famous ways of using online services like Google Drive, DropBox, One Drive, etc.
- Any user can store files over these cloud services, and they can share them with others, providing access from many users.
- It includes collaboration in real-time file sharing and version control access.



Protection

- File protection in an operating system is the process of securing files from unauthorized access, alteration, or deletion.
- It is critical for data security and ensures that sensitive information remains confidential and secure.
- Operating systems provide various mechanisms and techniques such as file permissions, encryption, access control lists, auditing, and physical file security to protect files.
- Proper file protection involves user authentication, authorization, access control, encryption, and auditing.
- File protection in an operating system is essential to maintain data security and minimize the risk of data breaches and other security incidents.

Type of File protection

- **File Permissions** – File permissions are a basic form of file protection that controls access to files by setting permissions for users and groups.
- File permissions allow the system administrator to assign specific access rights to users and groups, which can include read, write, and execute privileges.
- File permissions can be modified by the system administrator at any time to adjust access privileges, which helps to prevent unauthorized access.

- **Encryption** – Encryption is the process of converting plain text into ciphertext to protect files from unauthorized access.
- Encrypted files can only be accessed by authorized users who have the correct encryption key to decrypt them.
- Encryption is widely used to secure sensitive data such as financial information, personal data, and other confidential information.
- In an operating system, encryption can be applied to individual files or entire directories, providing an extra layer of protection against unauthorized access.

- **Access Control Lists (ACLs)** – Access control lists (ACLs) are lists of permissions attached to files and directories that define which users or groups have access to them and what actions they can perform on them.
- ACLs can be more granular than file permissions, allowing the system administrator to specify exactly which users or groups can access specific files or directories.
- ACLs can also be used to grant or deny specific permissions, such as read, write, or execute privileges, to individual users or groups.

- **Auditing and Logging** – Auditing and logging are mechanisms used to track and monitor file access, changes, and deletions.
- It involves creating a record of all file access and changes, including who accessed the file, what actions were performed, and when they were performed.
- Auditing and logging can help to detect and prevent unauthorized access and can also provide an audit trail for compliance purposes.

- **Physical File Security** – Physical file security involves protecting files from physical damage or theft.
- It includes measures such as file storage and access control, backup and recovery, and physical security best practices.
- Physical file security is essential for ensuring the integrity and availability of critical data, as well as compliance with regulatory requirements.

Advantages of File protection

- **Data Security** – File protection mechanisms such as encryption, access control lists, and file permissions provide robust data security by preventing unauthorized access to files.
- **Compliance** – File protection mechanisms are essential for compliance with regulatory requirements such as GDPR, HIPAA, and PCI-DSS. These regulations require organizations to implement appropriate security measures to protect sensitive data from unauthorized access, alteration, or deletion.
- **Business Continuity** – File protection mechanisms are essential for ensuring business continuity by preventing data loss due to accidental or malicious deletion, corruption, or other types of damage.

- **Increased Productivity** – File protection mechanisms can help to increase productivity by ensuring that files are available to authorized users when they need them.
- **Enhanced Collaboration** – File protection mechanisms can help to enhance collaboration by allowing authorized users to access and share files securely.
- **Reputation** – File protection mechanisms can enhance an organization's reputation by demonstrating a commitment to data security and compliance.

Disadvantages of File protection

- **Overhead** – Some file protection mechanisms such as encryption, access control lists, and auditing can add overhead to system performance. This can impact system resources and slow down file access and processing times.
- **Complexity** – File protection mechanisms can be complex and require specialized knowledge to implement and manage. This can lead to errors and misconfigurations that compromise data security.

- **Compatibility Issues** – Some file protection mechanisms may not be compatible with all types of files or applications, leading to compatibility issues and limitations in file usage.
- **Cost** – Implementing robust file protection mechanisms can be expensive, especially for small organizations with limited budgets. This can make it difficult to achieve full data protection.
- **User Frustration** – Stringent file protection mechanisms such as complex passwords, frequent authentication requirements, and restricted access can frustrate users and impact productivity.

File System Implementation

- 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 inodes.

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.
- Some common file systems used in operating systems include NTFS and FAT in Windows, and ext4 and XFS in Linux.

Advantages

1. **Efficient Data Storage:** File system implementation ensures efficient data storage on a physical storage device. It provides a structured way of organizing files and directories, which makes it easy to find and access files.
2. **Data Security:** File system implementation includes features for managing file security and permissions. This ensures that sensitive data is protected from unauthorized access.
3. **Data Recovery:** The file system implementation includes features for recovering from system failures and maintaining data integrity. This helps to prevent data loss and ensures that data can be recovered in the event of a system failure.

4. Improved Performance: File system implementation includes techniques such as buffering and caching to optimize file I/O performance. This results in faster access to data and improved overall system performance.

5. Scalability: File system implementation can be designed to be scalable, making it possible to store and retrieve large amounts of data efficiently.

6. Flexibility: Different file system implementations can be designed to meet specific needs and use cases. This allows developers to choose the best file system implementation for their specific requirements.

7. Cross-Platform Compatibility: Many file system implementations are cross-platform compatible, which means they can be used on different operating systems. This makes it easy to transfer files between different systems.

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 Structure

- File types also can be used to indicate the internal structure of the file.
- The operating system requires that an executable file have a specific structure so that it can determine where in memory to load the file and what the location of the first instruction is.
- If the OS supports multiple file structures, the resulting size of the OS is large.
- If the OS defines 5 different file structures, it needs to contain the code to support these file structures.
- All OS must support at least one structure of an executable file so that the system can load and run programs.

Allocation methods

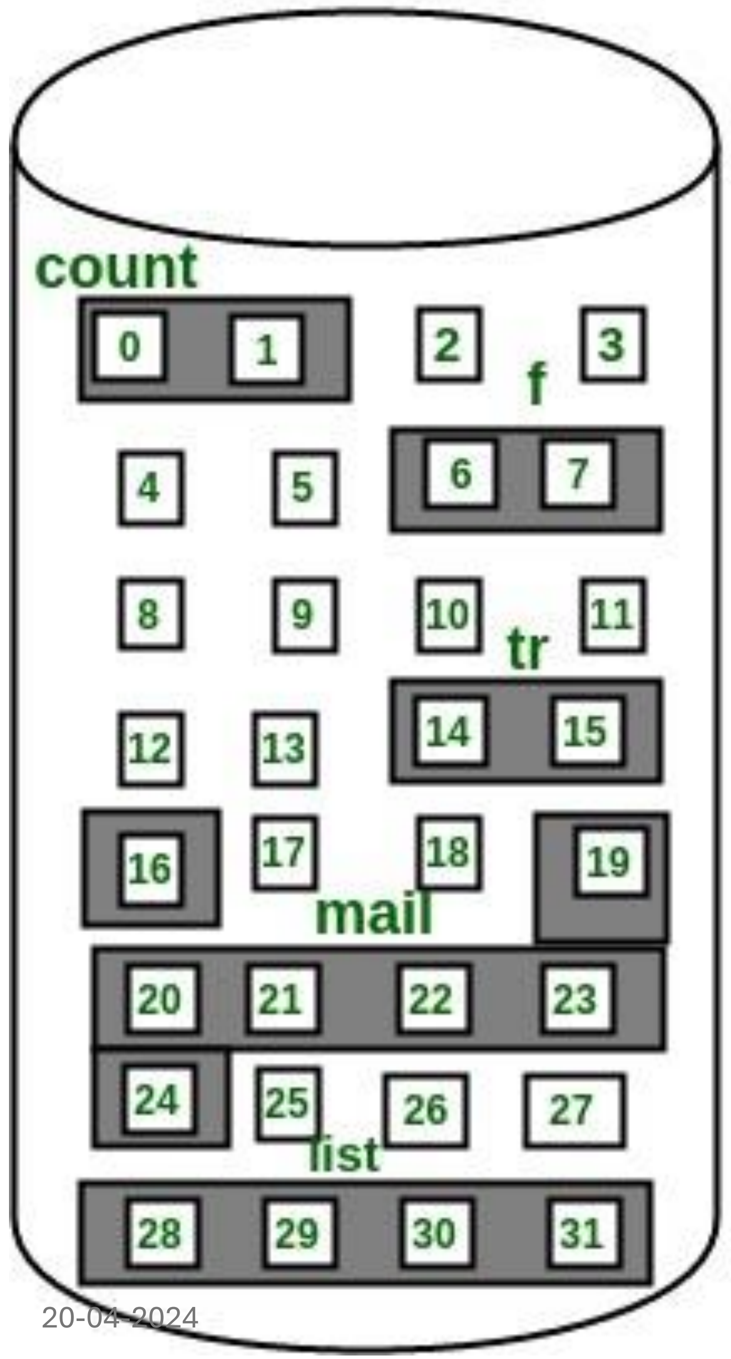
- The allocation methods define how the files are stored in the disk blocks.
- There are three main disk space or file allocation methods.
- Contiguous Allocation
- Linked Allocation
- Indexed Allocation

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

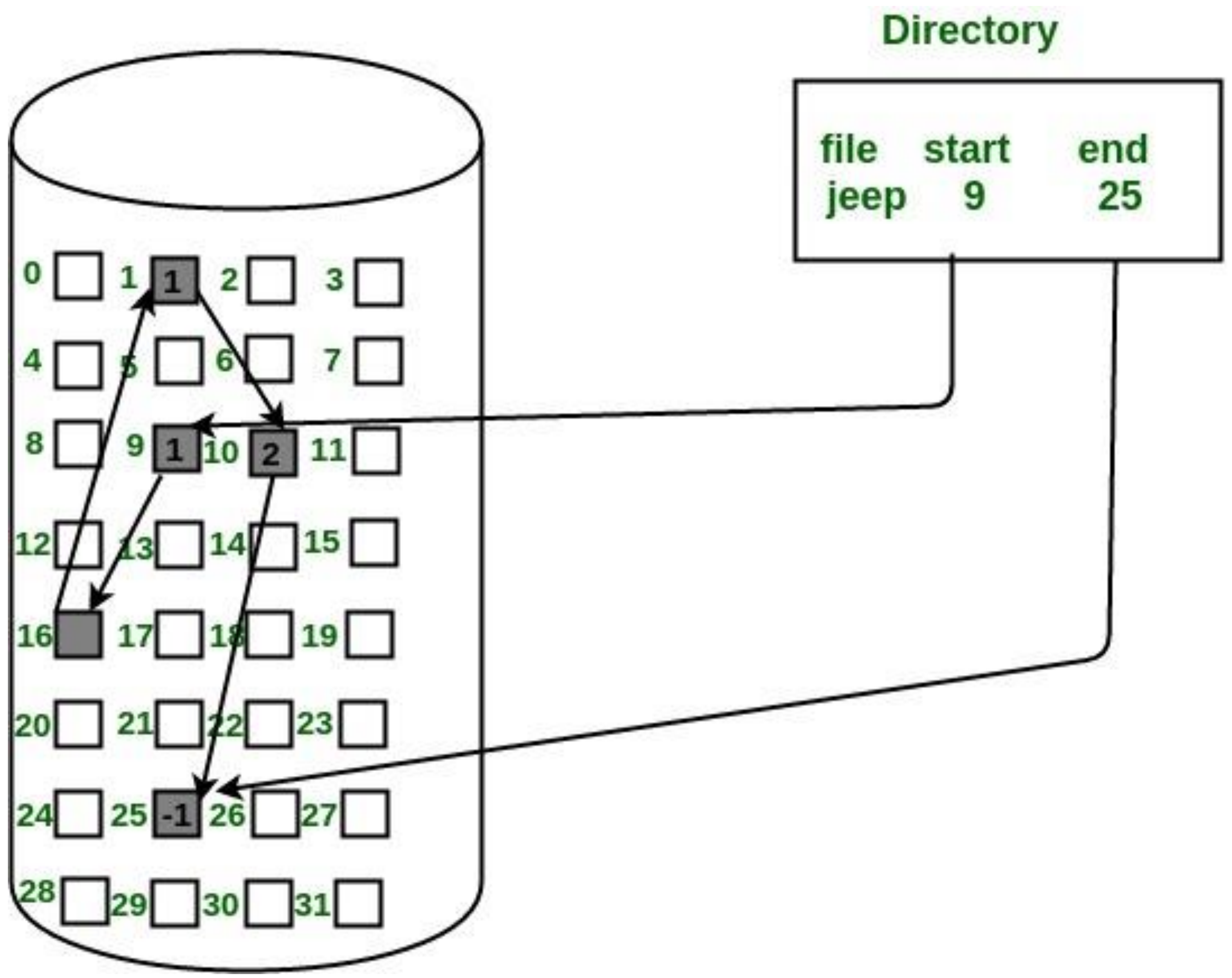
- **Advantages:**
- Both the Sequential and Direct Accesses are supported by this. For direct access, the address of the kth block of the file which starts at block b can easily be obtained as (b+k).
- This is extremely fast since the number of seeks are minimal because of contiguous allocation of file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

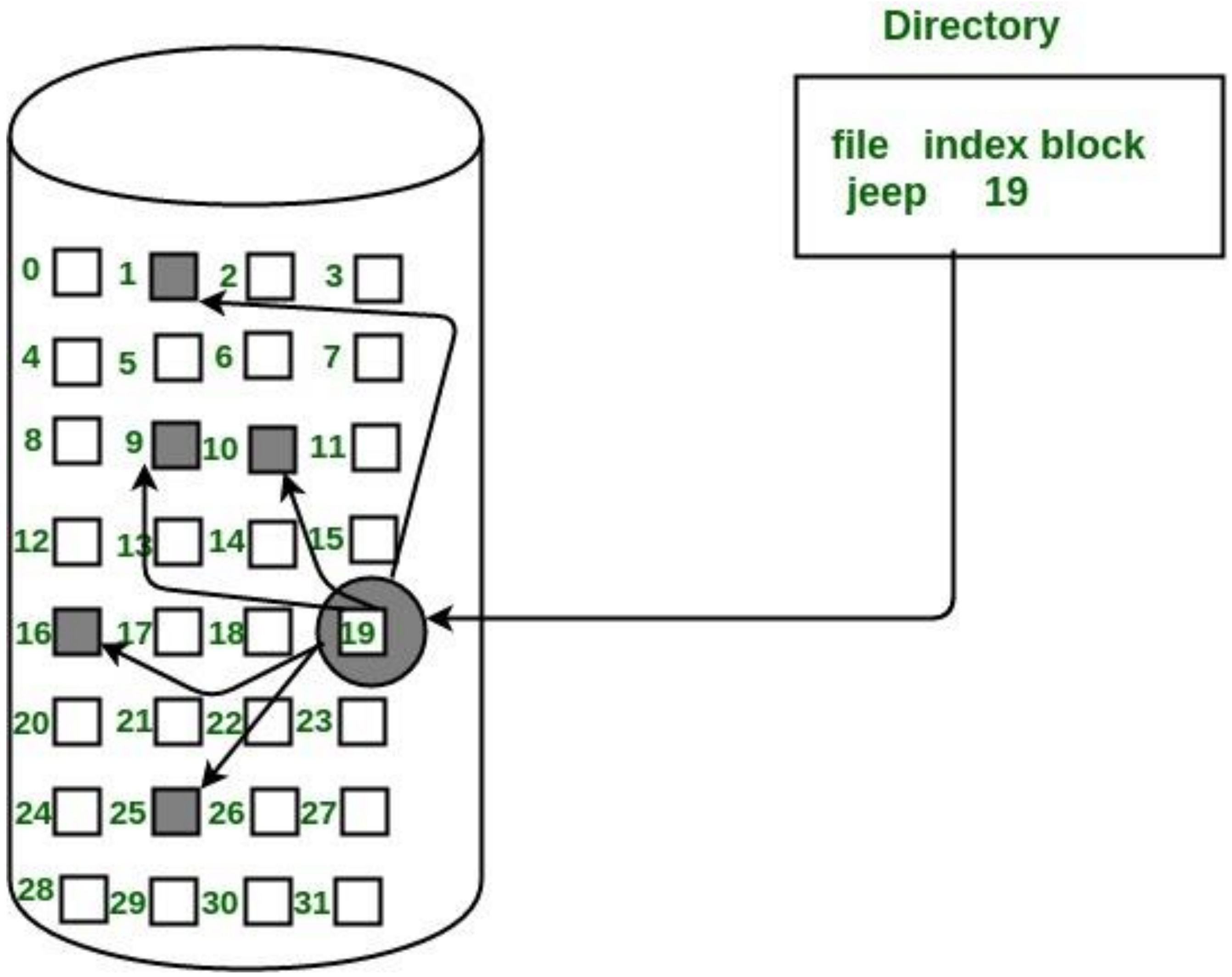


- **Advantages:**
- This is very flexible in terms of file size. File size can be increased easily since the system does not have to look for a contiguous chunk of memory.
- This method does not suffer from external fragmentation. This makes it relatively better in terms of memory utilization.
- **Disadvantages:**
- 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. Here are some of the commonly used free space management techniques:

1. **Linked Allocation:** In this technique, each file is represented by a linked list of disk blocks. When a file is created, the operating system finds enough free space on the disk and links the blocks of the file to form a chain. This method is simple to implement but can lead to fragmentation and waste of space.

1. **Contiguous Allocation:** In this technique, each file is stored as a contiguous block of disk space. When a file is created, the operating system finds a contiguous block of free space and assigns it to the file. This method is efficient as it minimizes fragmentation but suffers from the problem of external fragmentation.
2. **Indexed Allocation:** In this technique, a separate index block is used to store the addresses of all the disk blocks that make up a file. When a file is created, the operating system creates an index block and stores the addresses of all the blocks in the file. This method is efficient in terms of storage space and minimizes fragmentation.

1. **File Allocation Table (FAT):** In this technique, the operating system uses a file allocation table to keep track of the location of each file on the disk. When a file is created, the operating system updates the file allocation table with the address of the disk blocks that make up the file. This method is widely used in Microsoft Windows operating systems.
2. **Volume Shadow Copy:** This is a technology used in Microsoft Windows operating systems to create backup copies of files or entire volumes. When a file is modified, the operating system creates a shadow copy of the file and stores it in a separate location. This method is useful for data recovery and protection against accidental file deletion.

- Overall, free space management is a crucial function of operating systems, as it ensures that storage devices are utilized efficiently and effectively.
- The system keeps track 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 that 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 a 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**.

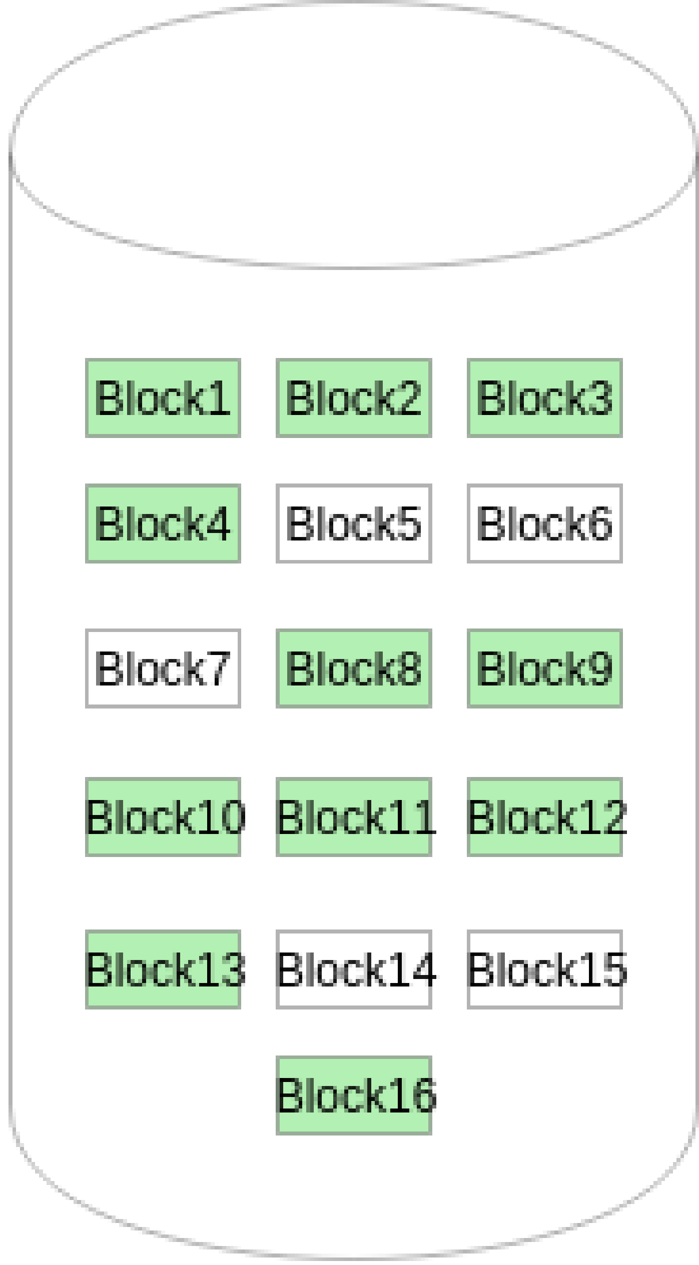

Block1
Block2
Block3
Block4
Block5
Block6
Block7
Block8
Block9
Block10
Block11
Block12
Block13
Block14
Block15
Block16

Figure - 1

1.Advantages –

1. Simple to understand.
2. 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.

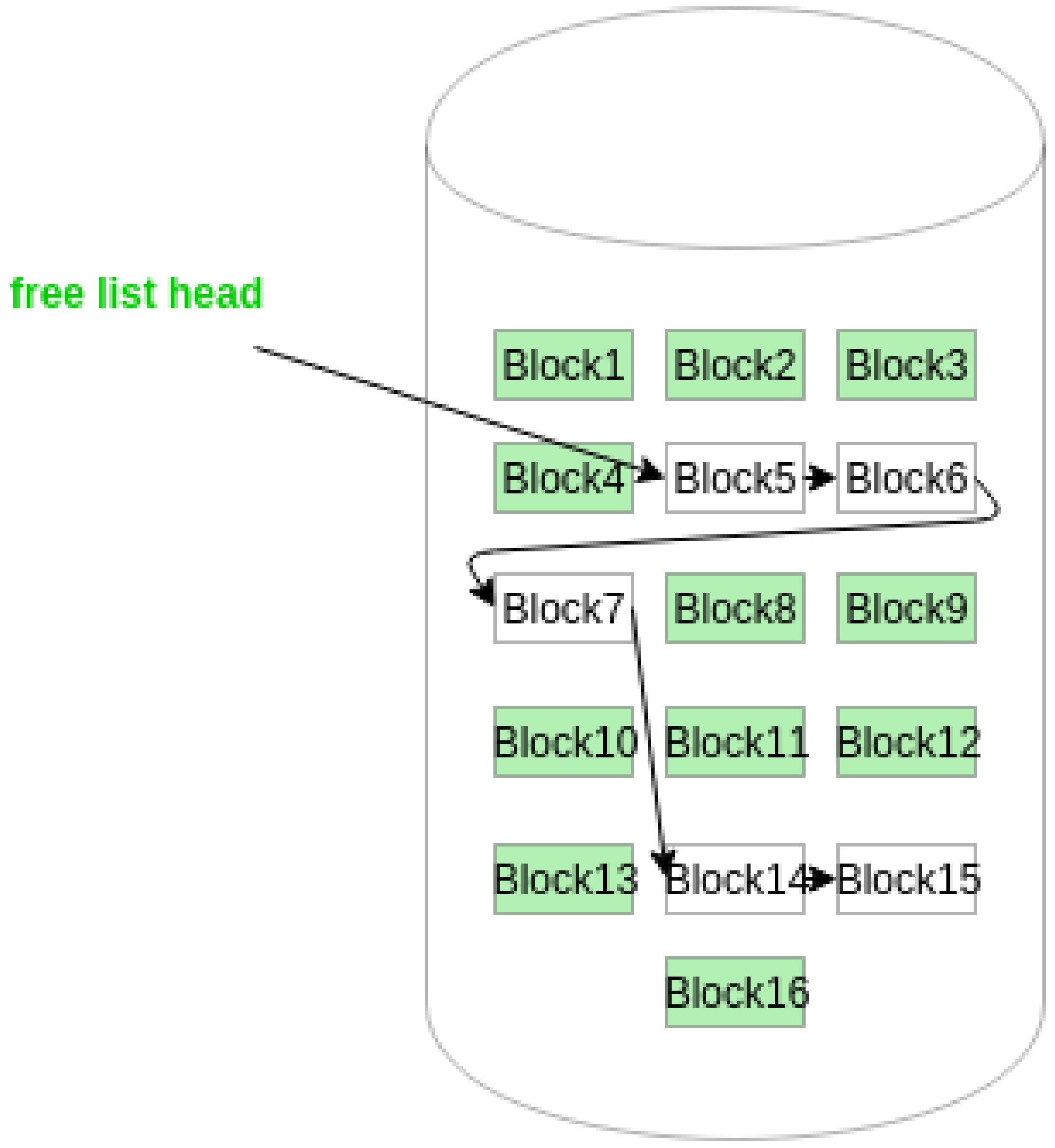


Figure - 2

1. **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.
2. **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:
    1. Address of first free disk block
    2. A number n

- **Here are some 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.

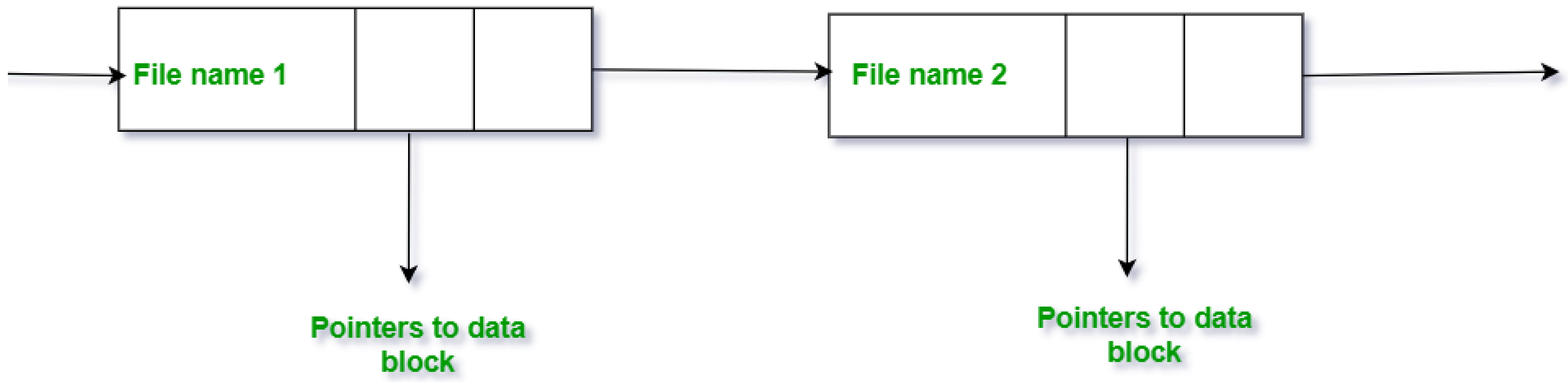
- **Disadvantages:**

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.

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. However, 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.

Directory Implementation Using Singly Linked List



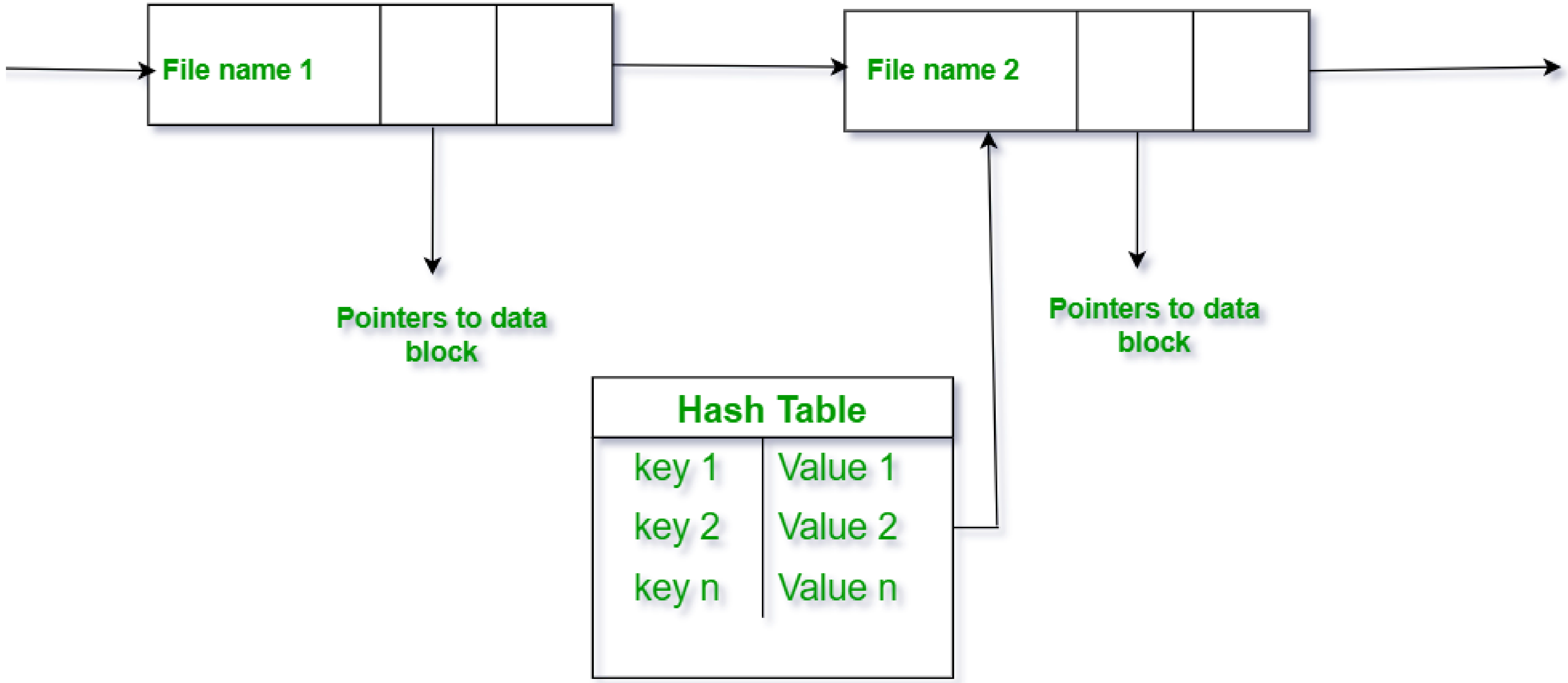
- To create a new file the entire list has to be checked such that the new directory does not exist previously.
- The new directory then can be added to the end of the list or at the beginning of the list.
- In order to delete a file, we first search the directory with the name of the file to be deleted. 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.

Directory Implementation Using Hash Table



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

Efficiency and Performance

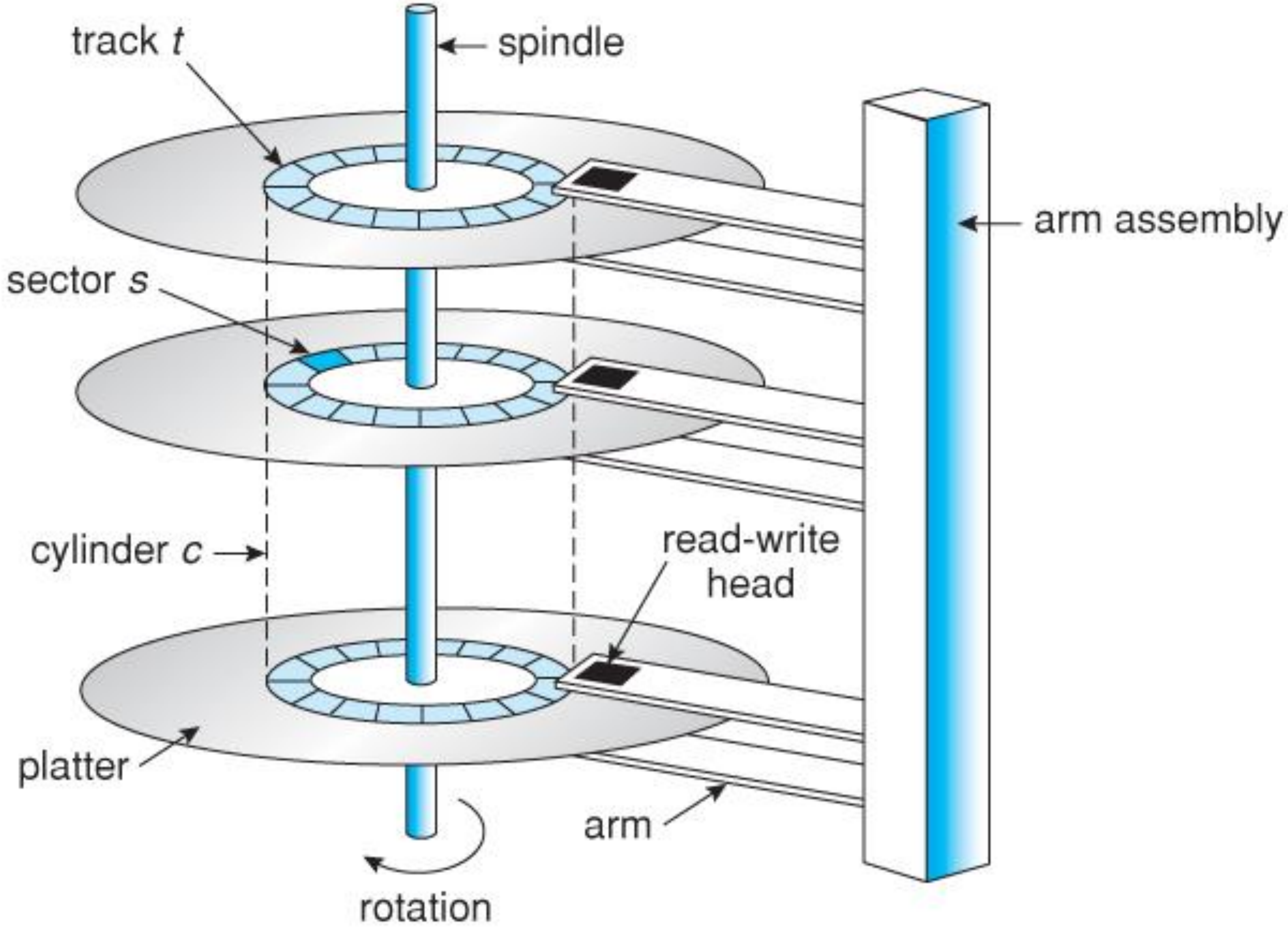
- **Efficiency and Performance**
- **Efficiency dependent on:**
- ● Disk allocation and directory algorithms
- ● Types of data kept in file’s directory entry
- **Performance**
- ● Disk cache – a separate section of main memory for frequently used blocks
- ● free-behind and read-ahead – techniques to optimize sequential access
- ● improve PC performance by dedicating a section of memory as a virtual disk, or RAM disk

Mass Storage Structure

Overview of Mass Storage Structure

- **Magnetic Disks**
- Traditional magnetic disks have the following basic structure:
  - One or more ***platters*** in the form of disks covered with magnetic media. ***Hard disk*** platters are made of rigid metal, while "***floppy***" disks are made of more flexible plastic.
  - Each platter has two working ***surfaces.*** Older hard disk drives would sometimes not use the very top or bottom surface of a stack of platters, as these surfaces were more susceptible to potential damage.
  - Each working surface is divided into a number of concentric rings called ***tracks.*** The collection of all tracks that are the same distance from the edge of the platter, (i.e. all tracks immediately above one another in the following diagram) is called a ***cylinder***.

- Each track is further divided into ***sectors,*** traditionally containing 512 bytes of data each, although some modern disks occasionally use larger sector sizes. (Sectors also include a header and a trailer, including checksum information among other things. Larger sector sizes reduce the fraction of the disk consumed by headers and trailers, but increase internal fragmentation and the amount of disk that must be marked bad in the case of errors.)
- The data on a hard drive is read by read-write ***heads.*** The standard configuration (shown below) uses one head per surface, each on a separate ***arm***, and controlled by a common ***arm assembly*** which moves all heads simultaneously from one cylinder to another. (Other configurations, including independent read-write heads, may speed up disk access, but involve serious technical difficulties.)
- The storage capacity of a traditional disk drive is equal to the number of heads (i.e. the number of working surfaces), times the number of tracks per surface, times the number of sectors per track, times the number of bytes per sector. A particular physical block of data is specified by providing the head-sector-cylinder number at which it is located.


track t
spindle
arm assembly
sector s
cylinder c
read-write head
platter
arm
rotation

- In operation the disk rotates at high speed, such as 7200 rpm (120 revolutions per second.) The rate at which data can be transferred from the disk to the computer is composed of several steps:
  - The ***positioning time***, a.k.a. the ***seek time*** or ***random access time*** is the time required to move the heads from one cylinder to another, and for the heads to settle down after the move. This is typically the slowest step in the process and the predominant bottleneck to overall transfer rates.
  - The ***rotational latency*** is the amount of time required for the desired sector to rotate around and come under the read-write head.This can range anywhere from zero to one full revolution, and on the average will equal one-half revolution. This is another physical step and is usually the second slowest step behind seek time. (For a disk rotating at 7200 rpm, the average rotational latency would be 1/2 revolution / 120 revolutions per second, or just over 4 milliseconds, a long time by computer standards.
  - The ***transfer rate***, which is the time required to move the data electronically from the disk to the computer. (Some authors may also use the term transfer rate to refer to the overall transfer rate, including seek time and rotational latency as well as the electronic data transfer rate.)

- Disk heads "fly" over the surface on a very thin cushion of air. If they should accidentally contact the disk, then a ***head crash*** occurs, which may or may not permanently damage the disk or even destroy it completely. For this reason it is normal to ***park*** the disk heads when turning a computer off, which means to move the heads off the disk or to an area of the disk where there is no data stored.
- Floppy disks are normally ***removable***. Hard drives can also be removable, and some are even ***hot-swappable***, meaning they can be removed while the computer is running, and a new hard drive inserted in their place.
- Disk drives are connected to the computer via a cable known as the ***I/O Bus.*** Some of the common interface formats include Enhanced Integrated Drive Electronics, EIDE; Advanced Technology Attachment, ATA; Serial ATA, SATA, Universal Serial Bus, USB; Fiber Channel, FC, and Small Computer Systems Interface, SCSI.
- The ***host controller*** is at the computer end of the I/O bus, and the ***disk controller*** is built into the disk itself. The CPU issues commands to the host controller via I/O ports. Data is transferred between the magnetic surface and onboard ***cache*** by the disk controller, and then the data is transferred from that cache to the host controller and the motherboard memory at electronic speeds.

- **Solid-State Disks - New**
- As technologies improve and economics change, old technologies are often used in different ways. One example of this is the increasing used of ***solid state disks, or SSDs.***
- SSDs use memory technology as a small fast hard disk. Specific implementations may use either flash memory or DRAM chips protected by a battery to sustain the information through power cycles.
- Because SSDs have no moving parts they are much faster than traditional hard drives, and certain problems such as the scheduling of disk accesses simply do not apply.
- However SSDs also have their weaknesses: They are more expensive than hard drives, generally not as large, and may have shorter life spans.
- SSDs are especially useful as a high-speed cache of hard-disk information that must be accessed quickly. One example is to store filesystem meta-data, e.g. directory and inode information, that must be accessed quickly and often. Another variation is a boot disk containing the OS and some application executables, but no vital user data. SSDs are also used in laptops to make them smaller, faster, and lighter.
- Because SSDs are so much faster than traditional hard disks, the throughput of the bus can become a limiting factor, causing some SSDs to be connected directly to the system PCI bus for example.

- **Magnetic Tapes - was 12.1.2**
- Magnetic tapes were once used for common secondary storage before the days of hard disk drives, but today are used primarily for backups.
- Accessing a particular spot on a magnetic tape can be slow, but once reading or writing commences, access speeds are comparable to disk drives.
- Capacities of tape drives can range from 20 to 200 GB, and compression can double that capacity.

Disk Structure

- The traditional head-sector-cylinder, HSC numbers are mapped to linear block addresses by numbering the first sector on the first head on the outermost track as sector 0. Numbering proceeds with the rest of the sectors on that same track, and then the rest of the tracks on the same cylinder before proceeding through the rest of the cylinders to the center of the disk. In modern practice these linear block addresses are used in place of the HSC numbers for a variety of reasons:
    1. The linear length of tracks near the outer edge of the disk is much longer than for those tracks located near the center, and therefore it is possible to squeeze many more sectors onto outer tracks than onto inner ones.
    2. All disks have some bad sectors, and therefore disks maintain a few spare sectors that can be used in place of the bad ones. The mapping of spare sectors to bad sectors in managed internally to the disk controller.
    3. Modern hard drives can have thousands of cylinders, and hundreds of sectors per track on their outermost tracks. These numbers exceed the range of HSC numbers for many (older) operating systems, and therefore disks can be configured for any convenient combination of HSC values that falls within the total number of sectors physically on the drive.

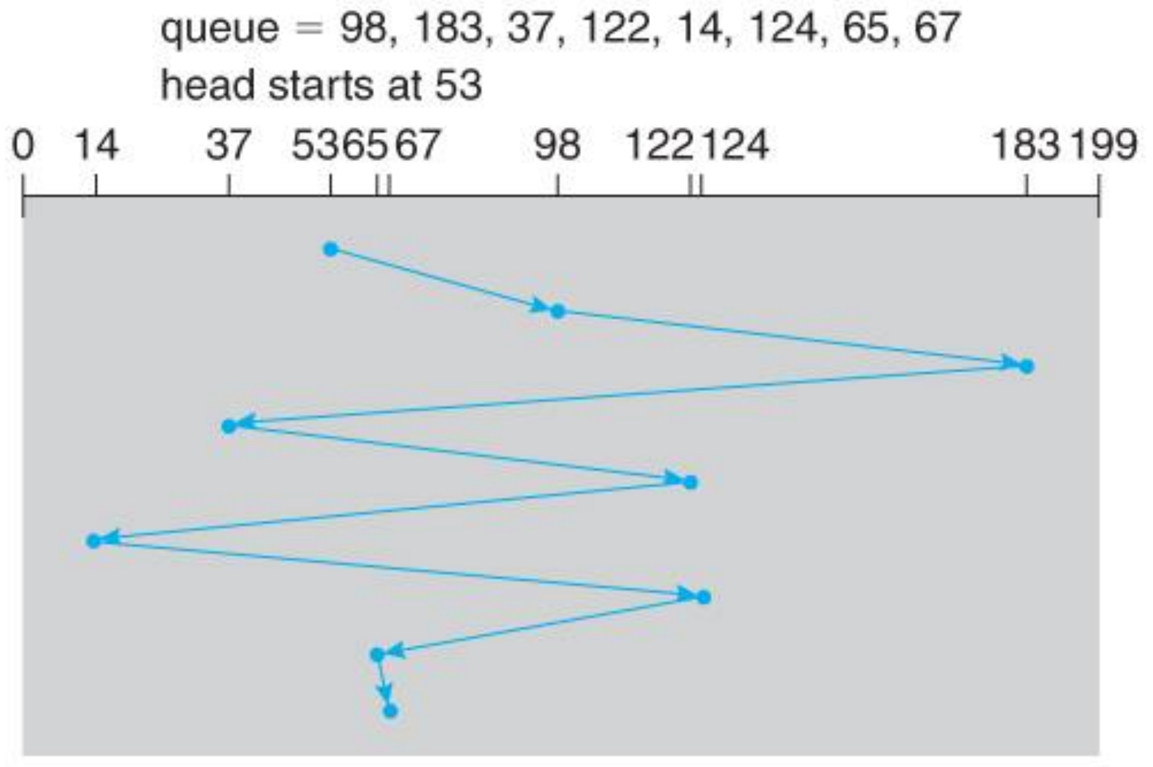
- There is a limit to how closely packed individual bits can be placed on a physical media, but that limit is growing increasingly more packed as technological advances are made.
- Modern disks pack many more sectors into outer cylinders than inner ones, using one of two approaches:
  - With ***Constant Linear Velocity, CLV,*** the density of bits is uniform from cylinder to cylinder. Because there are more sectors in outer cylinders, the disk spins slower when reading those cylinders, causing the rate of bits passing under the read-write head to remain constant. This is the approach used by modern CDs and DVDs.
  - With ***Constant Angular Velocity, CAV,*** the disk rotates at a constant angular speed, with the bit density decreasing on outer cylinders. (These disks would have a constant number of sectors per track on all cylinders.)

Disk Scheduling

- As mentioned earlier, disk transfer speeds are limited primarily by ***seek times*** and ***rotational latency.*** When multiple requests are to be processed there is also some inherent delay in waiting for other requests to be processed.
- ***Bandwidth*** is measured by the amount of data transferred divided by the total amount of time from the first request being made to the last transfer being completed, (for a series of disk requests.)
- Both bandwidth and access time can be improved by processing requests in a good order.
- Disk requests include the disk address, memory address, number of sectors to transfer, and whether the request is for reading or writing.

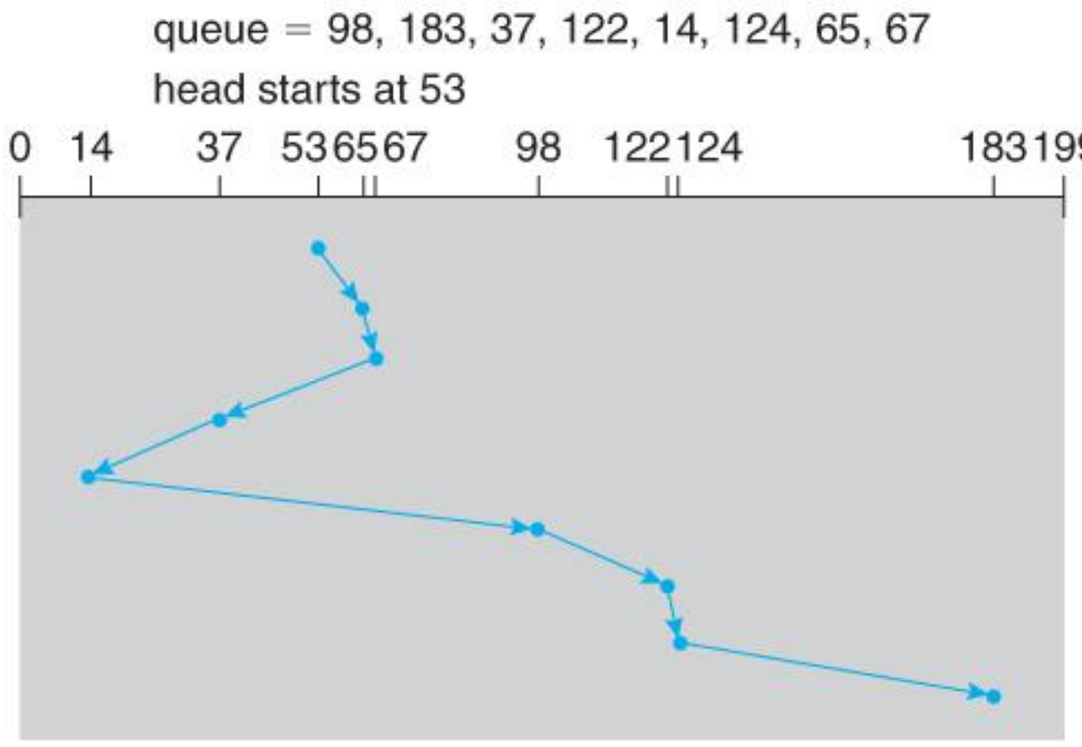
FCFS Scheduling

- The simplest form of disk scheduling is, of course, the first-come, first-served (FCFS) algorithm. This algorithm is intrinsically fair, but it generally does not provide the fastest service. Consider, for example, a disk queue with requests for I/O to blocks on cylinders 98, 183, 37, 122, 14, 124, 65, 67,

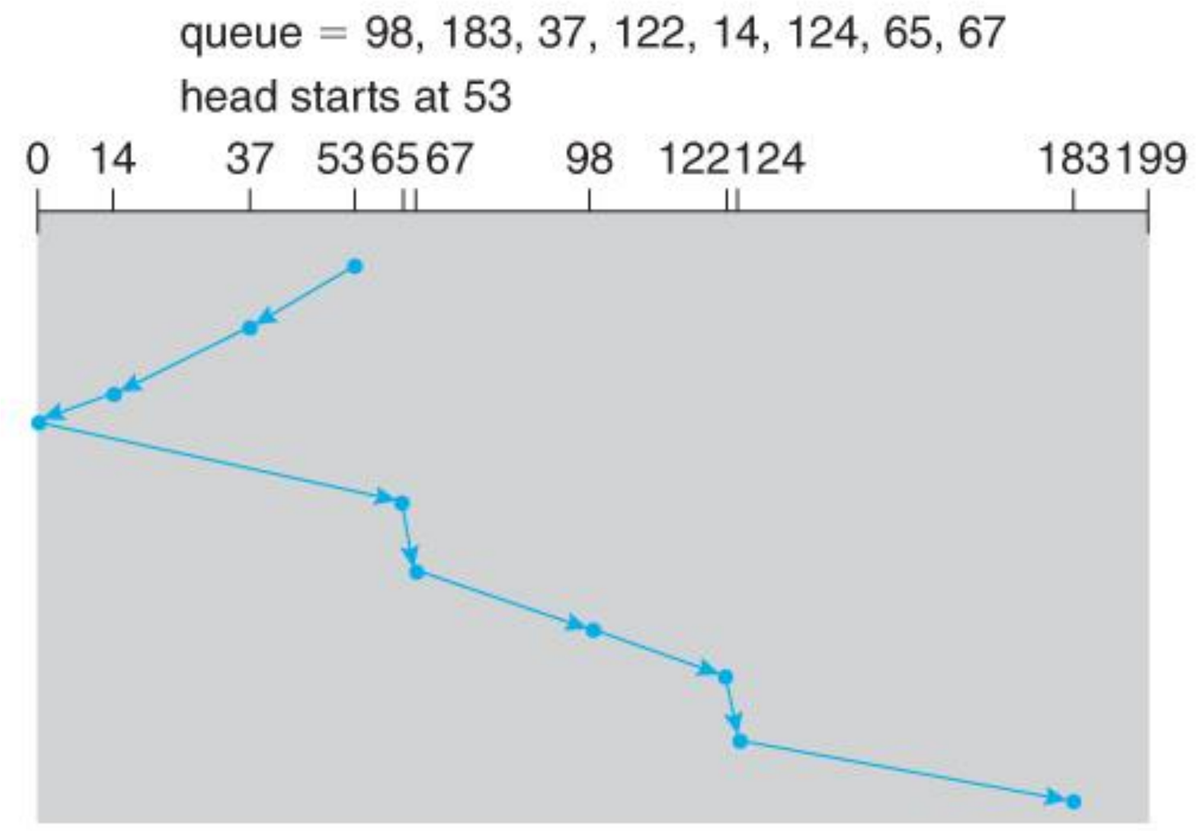


SSTF Scheduling

- ***hortest Seek Time First*** scheduling is more efficient, but may lead to starvation if a constant stream of requests arrives for the same general area of the disk.
- SSTF reduces the total head movement to 236 cylinders, down from 640 required for the same set of requests under FCFS. Note, however that the distance could be reduced still further to 208 by starting with 37 and then 14 first before processing the rest of the requests.

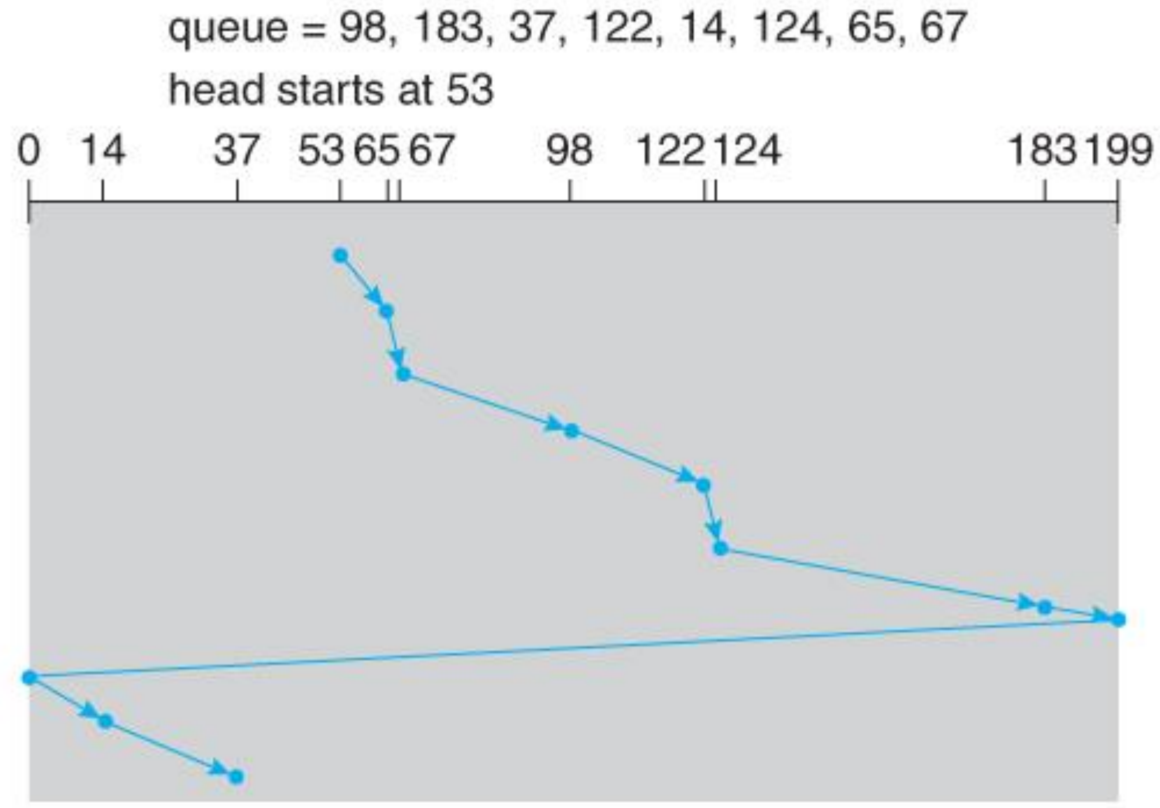


- **SCAN Scheduling**
- The ***SCAN*** algorithm, a.k.a. the ***elevator*** algorithm moves back and forth from one end of the disk to the other, similarly to an elevator processing requests in a tall building.

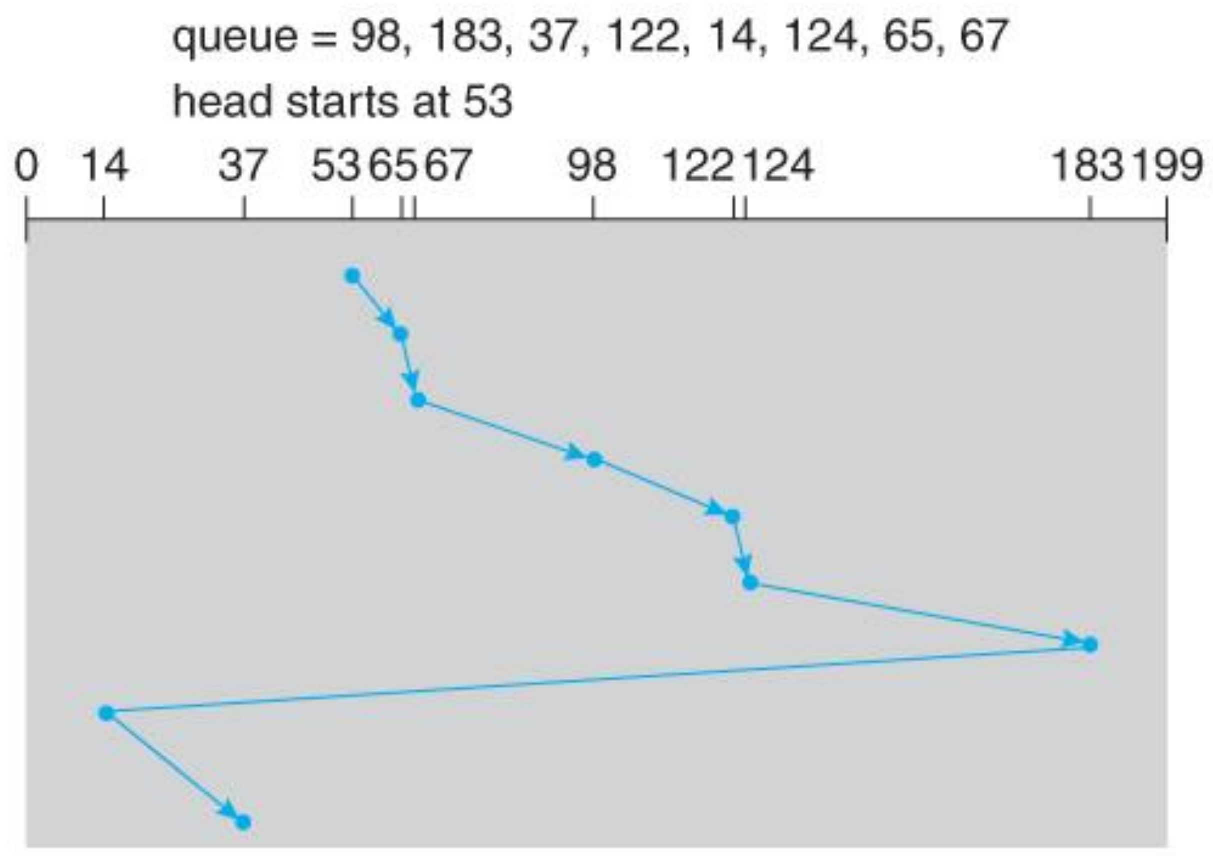


- Under the SCAN algorithm, If a request arrives just ahead of the moving head then it will be processed right away, but if it arrives just after the head has passed, then it will have to wait for the head to pass going the other way on the return trip. This leads to a fairly wide variation in access times which can be improved upon.
- Consider, for example, when the head reaches the high end of the disk: Requests with high cylinder numbers just missed the passing head, which means they are all fairly recent requests, whereas requests with low numbers may have been waiting for a much longer time. Making the return scan from high to low then ends up accessing recent requests first and making older requests wait that much longer.

- **C-SCAN Scheduling**
- The ***Circular-SCAN*** algorithm improves upon SCAN by treating all requests in a circular queue fashion - Once the head reaches the end of the disk, it returns to the other end without processing any requests, and then starts again from the beginning of the disk:



- **LOOK Scheduling**
- ***LOOK*** scheduling improves upon SCAN by looking ahead at the queue of pending requests, and not moving the heads any farther towards the end of the disk than is necessary. The following diagram illustrates the circular form of LOOK:



- **Selection of a Disk-Scheduling Algorithm**
- With very low loads all algorithms are equal, since there will normally only be one request to process at a time.
- For slightly larger loads, SSTF offers better performance than FCFS, but may lead to starvation when loads become heavy enough.
- For busier systems, SCAN and LOOK algorithms eliminate starvation problems.
- The actual optimal algorithm may be something even more complex than those discussed here, but the incremental improvements are generally not worth the additional overhead.

- Some improvement to overall filesystem access times can be made by intelligent placement of directory and/or inode information. If those structures are placed in the middle of the disk instead of at the beginning of the disk, then the maximum distance from those structures to data blocks is reduced to only one-half of the disk size. If those structures can be further distributed and furthermore have their data blocks stored as close as possible to the corresponding directory structures, then that reduces still further the overall time to find the disk block numbers and then access the corresponding data blocks.

- On modern disks the rotational latency can be almost as significant as the seek time, however it is not within the OSes control to account for that, because modern disks do not reveal their internal sector mapping schemes, (particularly when bad blocks have been remapped to spare sectors.)
    - Some disk manufacturers provide for disk scheduling algorithms directly on their disk controllers, (which do know the actual geometry of the disk as well as any remapping), so that if a series of requests are sent from the computer to the controller then those requests can be processed in an optimal order.
    - Unfortunately there are some considerations that the OS must take into account that are beyond the abilities of the on-board disk-scheduling algorithms, such as priorities of some requests over others, or the need to process certain requests in a particular order. For this reason OSes may elect to spoon-feed requests to the disk controller one at a time in certain situations.

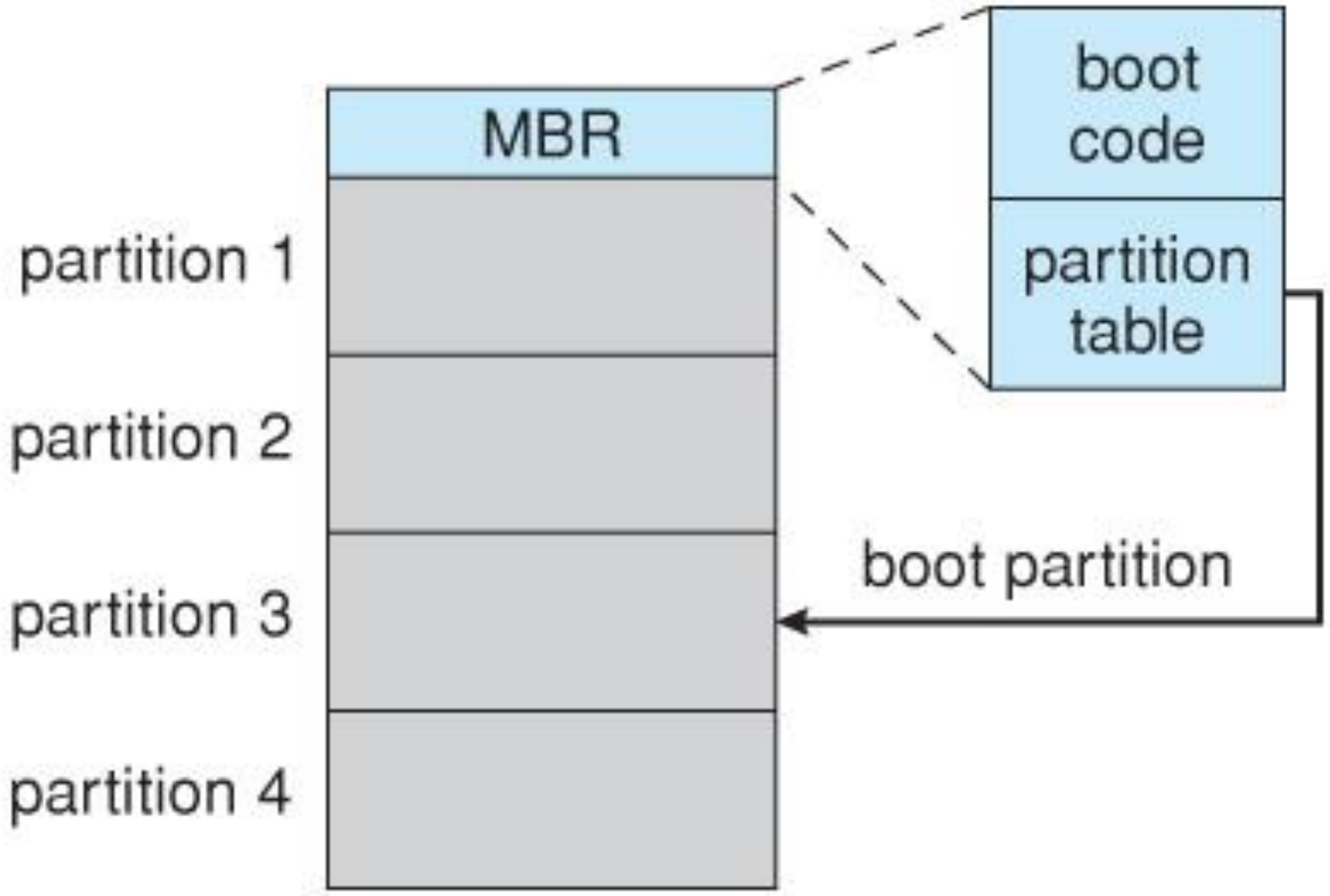
Disk Management

- **Disk Formatting**
- Before a disk can be used, it has to be ***low-level formatted***, which means laying down all of the headers and trailers marking the beginning and ends of each sector. Included in the header and trailer are the linear sector numbers, and ***error-correcting codes, ECC,*** which allow damaged sectors to not only be detected, but in many cases for the damaged data to be recovered (depending on the extent of the damage.) Sector sizes are traditionally 512 bytes, but may be larger, particularly in larger drives.
- ECC calculation is performed with every disk read or write, and if damage is detected but the data is recoverable, then a ***soft error*** has occurred. Soft errors are generally handled by the on-board disk controller, and never seen by the OS. (See below.)

- Once the disk is low-level formatted, the next step is to partition the drive into one or more separate partitions. This step must be completed even if the disk is to be used as a single large partition, so that the partition table can be written to the beginning of the disk.
- After partitioning, then the filesystems must be ***logically formatted,*** which involves laying down the master directory information (FAT table or inode structure), initializing free lists, and creating at least the root directory of the filesystem. (Disk partitions which are to be used as raw devices are not logically formatted. This saves the overhead and disk space of the filesystem structure, but requires that the application program manage its own disk storage requirements.)

- **Boot Block**
- Computer ROM contains a ***bootstrap*** program (OS independent) with just enough code to find the first sector on the first hard drive on the first controller, load that sector into memory, and transfer control over to it. (The ROM bootstrap program may look in floppy and/or CD drives before accessing the hard drive, and is smart enough to recognize whether it has found valid boot code or not.)
- The first sector on the hard drive is known as the ***Master Boot Record, MBR,*** and contains a very small amount of code in addition to the ***partition table.*** The partition table documents how the disk is partitioned into logical disks, and indicates specifically which partition is the ***active*** or ***boot*** partition.

- The boot program then looks to the active partition to find an operating system, possibly loading up a slightly larger / more advanced boot program along the way.
- In a ***dual-boot*** (or larger multi-boot) system, the user may be given a choice of which operating system to boot, with a default action to be taken in the event of no response within some time frame.
- Once the kernel is found by the boot program, it is loaded into memory and then control is transferred over to the OS. The kernel will normally continue the boot process by initializing all important kernel data structures, launching important system services (e.g. network daemons, sched, init, etc.), and finally providing one or more login prompts. Boot options at this stage may include ***single-user*** a.k.a. ***maintenance*** or ***safe*** modes, in which very few system services are started - These modes are designed for system administrators to repair problems or otherwise maintain the system.


MBR
boot code
partition table
partition 1
partition 2
partition 3
partition 4
boot partition

- **Bad Blocks**
- No disk can be manufactured to 100% perfection, and all physical objects wear out over time. For these reasons all disks are shipped with a few bad blocks, and additional blocks can be expected to go bad slowly over time. If a large number of blocks go bad then the entire disk will need to be replaced, but a few here and there can be handled through other means.
- In the old days, bad blocks had to be checked for manually. Formatting of the disk or running certain disk-analysis tools would identify bad blocks, and attempt to read the data off of them one last time through repeated tries. Then the bad blocks would be mapped out and taken out of future service. Sometimes the data could be recovered, and sometimes it was lost forever. (Disk analysis tools could be either destructive or non-destructive.)

- Modern disk controllers make much better use of the error-correcting codes, so that bad blocks can be detected earlier and the data usually recovered. (Recall that blocks are tested with every write as well as with every read, so often errors can be detected before the write operation is complete, and the data simply written to a different sector instead.)
- Note that re-mapping of sectors from their normal linear progression can throw off the disk scheduling optimization of the OS, especially if the replacement sector is physically far away from the sector it is replacing. For this reason most disks normally keep a few spare sectors on each cylinder, as well as at least one spare cylinder. Whenever possible a bad sector will be mapped to another sector on the same cylinder, or at least a cylinder as close as possible. ***Sector slipping*** may also be performed, in which all sectors between the bad sector and the replacement sector are moved down by one, so that the linear progression of sector numbers can be maintained.
- If the data on a bad block cannot be recovered, then a ***hard error*** has occurred., which requires replacing the file(s) from backups, or rebuilding them from scratch.

Swap space Management

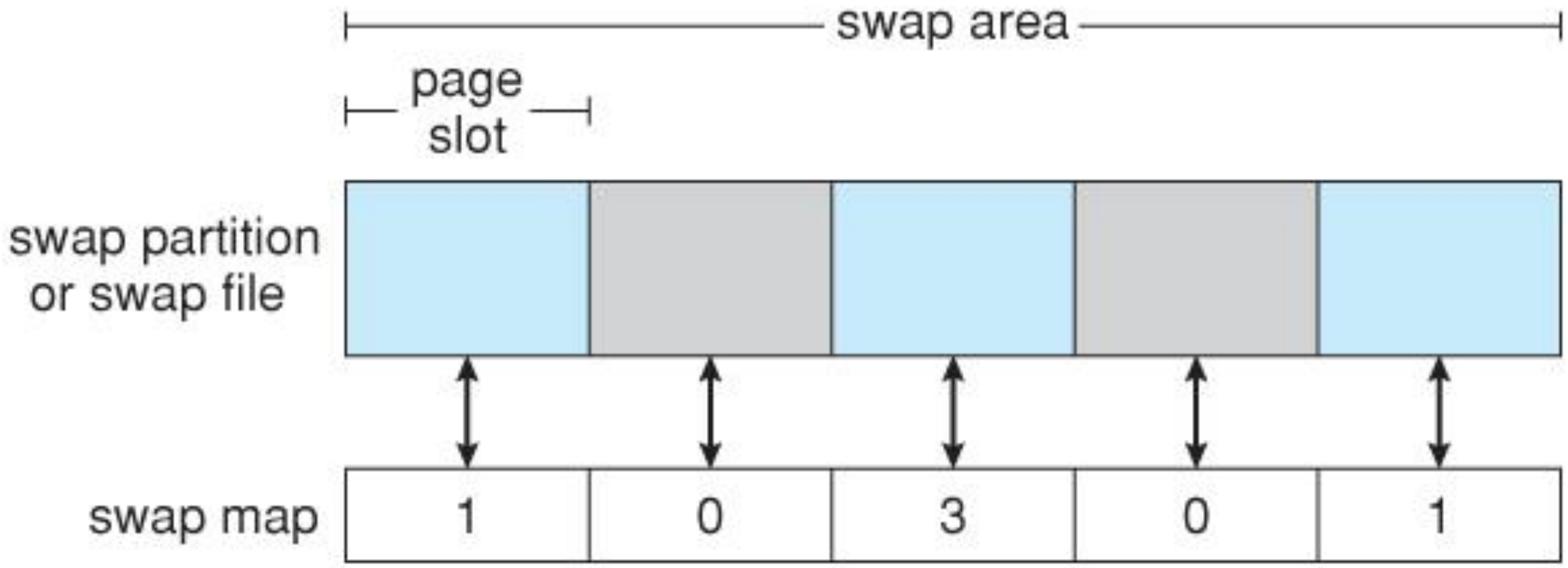
- Modern systems typically swap out pages as needed, rather than swapping out entire processes. Hence the swapping system is part of the virtual memory management system.
- Managing swap space is obviously an important task for modern OSes.

- **Swap-Space Use**
- The amount of swap space needed by an OS varies greatly according to how it is used. Some systems require an amount equal to physical RAM; some want a multiple of that; some want an amount equal to the amount by which virtual memory exceeds physical RAM, and some systems use little or none at all!
- Some systems support multiple swap spaces on separate disks in order to speed up the virtual memory system

- *Swap-Space Location*
- Swap space can be physically located in one of two locations:
- As a large file which is part of the regular filesystem. This is easy to implement, but inefficient. Not only must the swap space be accessed through the directory system, the file is also subject to fragmentation issues. Caching the block location helps in finding the physical blocks, but that is not a complete fix.
- As a raw partition, possibly on a separate or little-used disk. This allows the OS more control over swap space management, which is usually faster and more efficient. Fragmentation of swap space is generally not a big issue, as the space is re-initialized every time the system is rebooted. The downside of keeping swap space on a raw partition

- **Swap-Space Management: An Example**
- Historically OSes swapped out entire processes as needed. Modern systems swap out only individual pages, and only as needed. (For example process code blocks and other blocks that have not been changed since they were originally loaded are normally just freed from the virtual memory system rather than copying them to swap space, because it is faster to go find them again in the filesystem and read them back in from there than to write them out to swap space and then read them back.)
- In the mapping system shown below for Linux systems, a map of swap space is kept in memory, where each entry corresponds to a 4K block in the swap space. Zeros indicate free slots and non-zeros refer to how many processes have a mapping to that particular block (>1 for shared pages only.)
-

- **Figure 10.10 - The data structures for swapping on Linux systems.**



Protection - System Protection

- **Introduction:**
- **System protection** in an operating system refers to the mechanisms implemented by the operating system to ensure the security and integrity of the system. System protection involves various techniques to prevent unauthorized access, misuse, or modification of the operating system and its resources.
- There are several ways in which an operating system can provide system protection:
- **User authentication:** The operating system requires users to authenticate themselves before accessing the system. Usernames and passwords are commonly used for this purpose.
- **Access control:** The operating system uses access control lists (ACLs) to determine which users or processes have permission to access specific resources or perform specific actions.

- **Encryption:** The operating system can use encryption to protect sensitive data and prevent unauthorized access.
- **Firewall:** A firewall is a software program that monitors and controls incoming and outgoing network traffic based on predefined security rules.
- **Antivirus software:** Antivirus software is used to protect the system from viruses, malware, and other malicious software.
- **System updates and patches:** The operating system must be kept up-to-date with the latest security patches and updates to prevent known vulnerabilities from being exploited.
- By implementing these protection mechanisms, the operating system can prevent unauthorized access to the system, protect sensitive data, and ensure the overall security and integrity of the system.

- **What is Protection?**
- **Protection** refers to a mechanism which controls the access of programs, processes, or users to the resources defined by a computer system. We can take protection as a helper to multi programming operating system, so that many users might safely share a common logical name space such as directory or files.
- **Need for Protection:**
- To prevent the access of unauthorized users
- To ensure that each active programs or processes in the system uses resources only as the stated policy
- To improve reliability by detecting latent errors
- **Role of Protection:**
- The role of protection is to provide a mechanism that implement policies which defines the uses of resources in the computer system. Some policies are defined at the time of design of the system, some are designed by management of the system and some are defined by the users of the system to protect their own files and programs. Every application has different policies for use of the resources and they may change over time so protection of the system is not only concern of the designer of the operating system. Application programmer should also design the protection mechanism to protect their system against misuse. Policy is different from mechanism. Mechanisms determine how something will be done and policies determine what will be done. Policies are changed over time and place to place. Separation of mechanism and policy is important for the flexibility of the system.

- **Advantages of system protection in an operating system:**

1. Ensures the security and integrity of the system
2. Prevents unauthorized access, misuse, or modification of the operating system and its resources
3. Protects sensitive data
4. Provides a secure environment for users and applications
5. Prevents malware and other security threats from infecting the system
6. Allows for safe sharing of resources and data among users and applications
7. Helps maintain compliance with security regulations and standards

- **Disadvantages of system protection in an operating system:**

1. Can be complex and difficult to implement and manage
2. May slow down system performance due to increased security measures
3. Can cause compatibility issues with some applications or hardware
4. Can create a false sense of security if users are not properly educated on safe computing practices
5. Can create additional costs for implementing and maintaining security measures.

Goals of Protection

- **Goals of Protection :**
- Therefore, protection is a method of safeguarding data and processes against malicious and intentional intrusion. For that purpose, we have protection policies that are either designed by the system itself or specified by the management itself or are imposed by the programmers individually to protect their programs with extra safety.
- It also gives a multiprogramming OS the sense of safety that is required by its users to share common space like files or directories.
- The policies bind how the processes are to access the resources present in the computer system, resources like CPU, memory, software and even the OS. Both the OS designer and the application programmer are responsible for this. However, these policies always change from time to time.

Domain of Protection

- The protection policies limit the access of each process with respect to their resource handling. A process is bound to use only those resources which it requires to complete its task, in the time limit that it requires and also the mode in which it is required. That is the protected domain of a process.
- A computer system has processes and objects, which are treated as abstract data types, and these objects have operations specific to them. A domain element is described as <object, {set of operations on object}>.
- Each domain consists of a set of objects and the operations that can be performed on them. A domain can consist of either only a process or a procedure or a user. Then, if a domain corresponds to a procedure, then changing domain would mean changing procedure ID. Objects may share a common operation or two. Then the domains overlap.
-

Case studies: Linux

- **Linux:** Linux could be a free and open supply OS supported operating system standards. It provides programming interface still as programme compatible with operating system primarily based systems and provides giant selection applications. A UNIX operating system additionally contains several severally developed parts, leading to UNIX operating system that is totally compatible and free from proprietary code.

Case studies: Windows

- **Windows:** Windows may be a commissioned OS within which ASCII text file is inaccessible. it's designed for the people with the angle of getting no programming information and for business and alternative industrial users. it's terribly straightforward and simple to use. The distinction between Linux and Windows package is that Linux is completely freed from price whereas windows is marketable package and is expensive. Associate operating system could be a program meant to regulate the pc or computer hardware Associate behave as an treater between user and hardware. Linux is a open supply package wherever users will access the ASCII text file and might improve the code victimisation the system. On the opposite hand, in windows, users can't access ASCII text file, and it's a authorized OS. Let's see that the difference between Linux and windows:

S.NO	Linux	Windows
1.	Linux is a open source operating system.	While windows are the not the open source operating system.
2.	Linux is free of cost.	While it is costly.
3.	It’s file name case-sensitive.	While it’s file name is case-insensitive.
4.	In linux, monolithic kernel is used.	While in this, hybrid kernel is used.
5.	Linux is more efficient in comparison of windows.	While windows are less efficient.
6.	There is forward slash is used for Separating the directories.	While there is back slash is used for Separating the directories.

S.NO	Linux	Windows
7.	Linux provides more security than windows.	While it provides less security than linux.
8.	Linux is widely used in hacking purpose based systems.	While windows does not provide much efficiency in hacking.
9.	There are 3 types of user account – (1) Regular , (2) Root , (3) Service account	There are 4 types of user account – (1) Administrator , (2) Standard , (3) Child , (4) Guest
10.	Root user is the super user and has all administrative privileges.	Administrator user has all administrative privileges of computers.
11.	Linux file naming convention in case sensitive. Thus, sample and SAMPLE are 2 different files in Linux/Unix operating system.	In Windows, you cannot have 2 files with the same name in the same folder.