# Unit 5

## I/O Devices

### Types of I/O Devices

I/O devices can be broadly classified into:

1. **Human-Readable Devices**: Devices used for communication with users (e.g., monitors, keyboards, printers).
2. **Machine-Readable Devices**: Devices used for communication between systems (e.g., sensors, controllers, modems).
3. **Communication Devices**: Devices used for data transfer between computers (e.g., network interface cards).

---

### I/O Device Characteristics

1. **Speed**:
   - Devices vary widely in data transfer speed. For example:
     - Hard drives: 100 MB/s.
     - Keyboard: A few bytes per second.
2. **Data Transfer**:
   - Can occur in **blocks** (e.g., disks) or **streams** (e.g., keyboards).
3. **Access Method**:
   - **Sequential Access**: Data is accessed in order (e.g., tapes).
   - **Random Access**: Data can be accessed in any order (e.g., disks).
4. **I/O Modes**:
   - **Polling**: The CPU repeatedly checks the device's status.
   - **Interrupt-driven I/O**: The device sends an interrupt to the CPU when ready for data transfer.

- **DMA (Direct Memory Access)**: Data transfer occurs directly between the device and memory without CPU intervention.

## Organization of the I/O Function

1. **Programmed I/O:** It is due to the result of the I/O instructions that are written in the computer program. Each data item transfer is initiated by an instruction in the program. Usually the transfer is from a CPU register and memory. In this case it requires constant monitoring by the CPU of the peripheral devices. **Example of Programmed I/O:** In this case, the I/O device does not have direct access to the memory unit. A transfer from I/O device to memory requires the execution of several instructions by the CPU, including an input instruction to transfer the data from device to the CPU and store instruction to transfer the data from CPU to memory. In programmed I/O, the CPU stays in the program loop until the I/O unit indicates that it is ready for data transfer. This is a time consuming process since it needlessly keeps the CPU busy. This situation can be avoided by using an interrupt facility. This is discussed below.
2. **Interrupt- initiated I/O:** Since in the above case we saw the CPU is kept busy unnecessarily. This situation can very well be avoided by using an interrupt driven method for data transfer. By using interrupt facility and special commands to inform the interface to issue an interrupt request signal whenever data is available from any device. In the meantime the CPU can proceed for any other program execution. The interface meanwhile keeps monitoring the device. Whenever it is determined that the device is ready for data transfer it initiates an interrupt request signal to the computer. Upon detection of an external interrupt signal the CPU stops momentarily the task that it was already performing, branches to the service program to process the I/O transfer, and then return to the task it was originally performing.

    - The I/O transfer rate is limited by the speed with which the processor can test and service a device.
    - The processor is tied up in managing an I/O transfer; a number of instructions must be executed for each I/O transfer.
    - Terms:
        - Hardware Interrupts: Interrupts present in the hardware pins.

- Software Interrupts: These are the instructions used in the program whenever the required functionality is needed.
- Vectored interrupts: These interrupts are associated with the static vector address.
- Non-vectored interrupts: These interrupts are associated with the dynamic vector address.
- Maskable Interrupts: These interrupts can be enabled or disabled explicitly.
- Non-maskable interrupts: These are always in the enabled state. we cannot disable them.
- External interrupts: Generated by external devices such as I/O.
- Internal interrupts: These devices are generated by the internal components of the processor such as power failure, error instruction, temperature sensor, etc.
- Synchronous interrupts: These interrupts are controlled by the fixed time interval. All the interval interrupts are called as synchronous interrupts.
- Asynchronous interrupts: These are initiated based on the feedback of previous instructions. All the external interrupts are called as asynchronous interrupts.

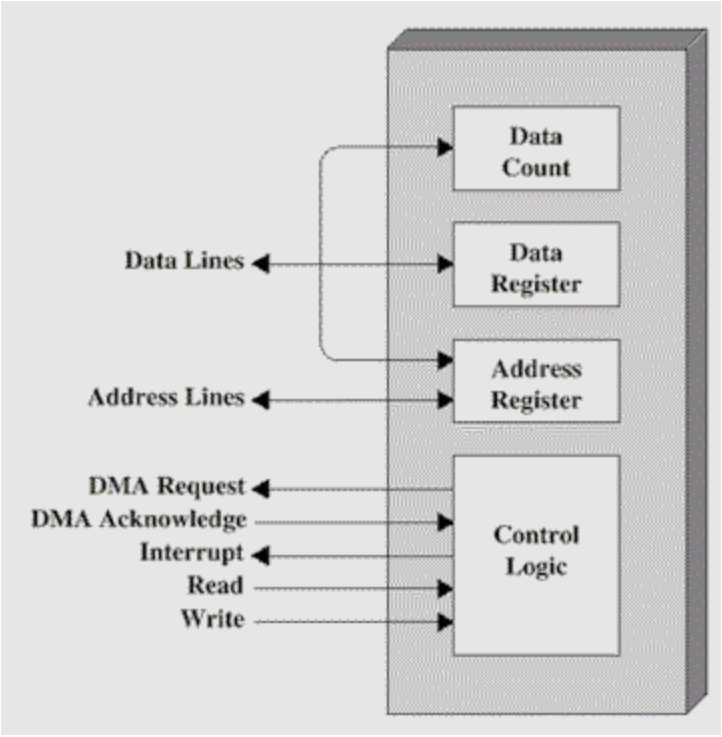
3. **Direct Memory Access** : The data transfer between a fast storage media such as magnetic disk and memory unit is limited by the speed of the CPU. Thus we can allow the peripherals directly communicate with each other using the memory buses, removing the intervention of the CPU. This type of data transfer technique is known as DMA or direct memory access. During DMA the CPU is idle and it has no control over the memory buses. The DMA controller takes over the buses to manage the transfer directly between the I/O devices and the memory unit.
    a. Bus grant request time.
    b. Transfer the entire block of data at transfer rate of device because the device is usually slow than the speed at which the data can be transferred

to CPU.

c. Release the control of the bus back to CPU So, total time taken to transfer the N bytes = Bus grant request time + (N) * (memory transfer rate) + Bus release control time.

4. Buffer the byte into the buffer
5. Inform the CPU that the device has 1 byte to transfer (i.e. bus grant request)
6. Transfer the byte (at system bus speed)
7. Release the control of the bus back to CPU.

## Direct Memory Access




An I/O device that can perform DMA has a small CPU to carry out the transfer of data and access memory.

The diagram to the right shows the control, data and address lines that are needed in the system

The next figure shows that an I/O operation can break into the instruction cycle at a number of different points.

## DMA Breakpoints

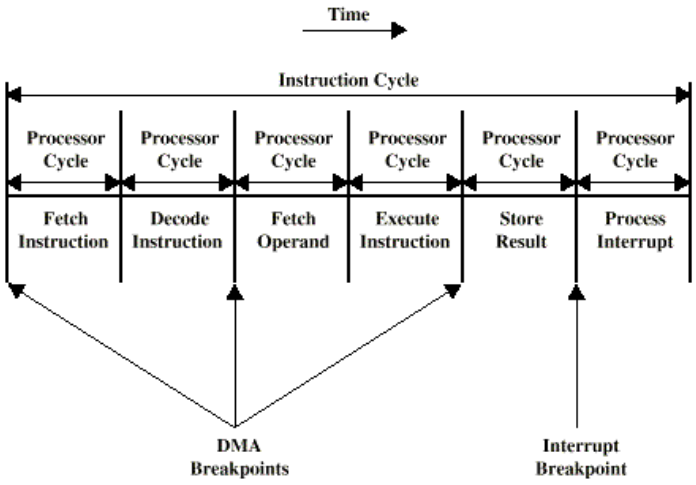
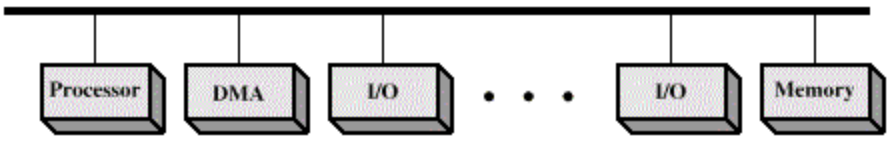
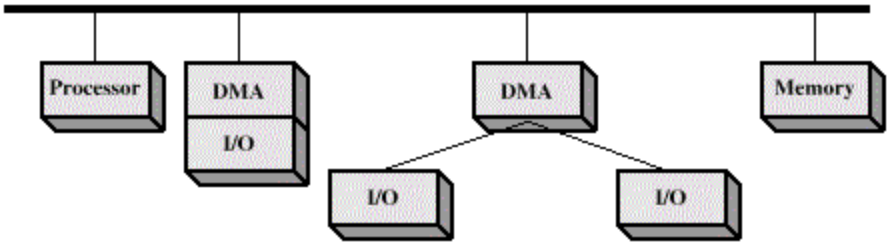



**Figure 11.2 DMA and Interrupt Breakpoints During an Instruction Cycle**

# DMA Design Alternatives




**(a) Single-bus, detached DMA**




**(b) Single-bus, Integrated DMA-I/O**

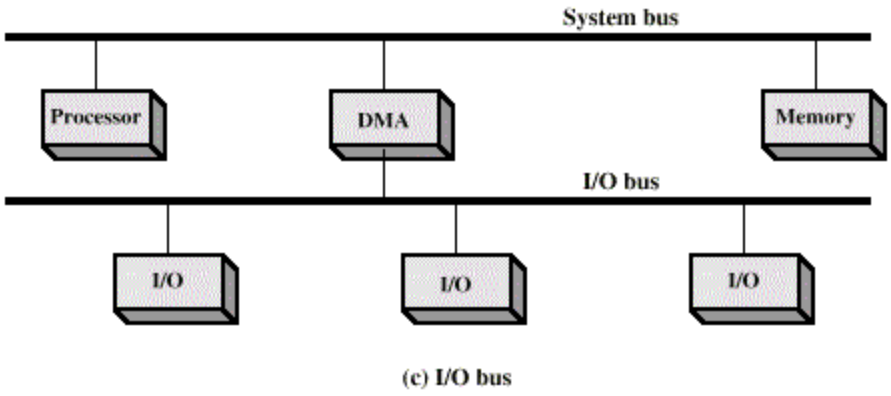
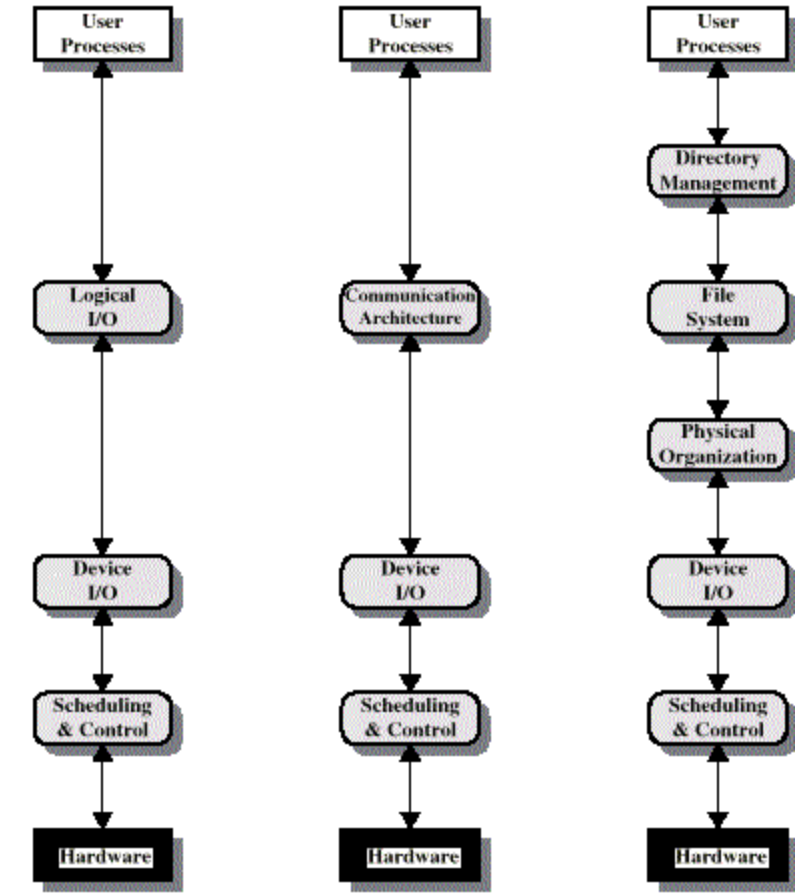



**Figure 11.3 Alternative DMA Configurations**

## Logical Structure of the I/O function

- **Logical I/O**- open, close, read, write
- **Device I/O**- buffering, controller commands
- **Scheduling and Control**- queuing and scheduling, interrupts handled here
- **Directory management**- symbolic names converted to pointers, create, delete, rename
- **File system**- the logical operations on the files, open, close, read, write, locate, access rights.
- **Physical organization**- mapping of the file blocks to the device blocks




(a) Local peripheral device (b) Communications port (c) File system

## What is I/O Buffering?

I/O buffering is a technique used in computer systems to improve the efficiency of input and output (I/O) operations. It involves the temporary storage of data in a buffer, which is a reserved area of memory, to reduce the number of I/O operations and manage the flow of data between fast and slow devices or processes

### Types of I/O Buffering Techniques

### 1. Single Buffer

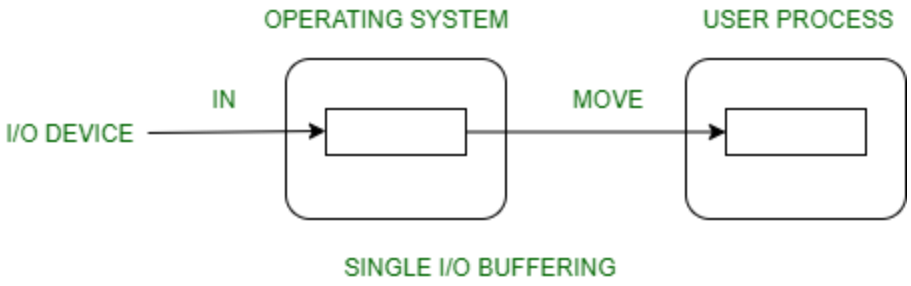
Using one buffer to store data temporarily. A buffer is provided by the operating system to the system portion of the main memory.

**Block Oriented Device**

- The system buffer takes the input.
- After taking the input, the block gets transferred to the user space by the process and then the process requests for another block.
- Two blocks work simultaneously, when one block of data is processed by the user process, the next block is being read in.
- OS can swap the processes.
- OS can record the data of the system buffer to user processes.

**Stream Oriented Device**

- Line- at a time operation is used for scroll-made terminals. The user inputs one line at a time, with a carriage return signaling at the end of a line.
- Byte-at-a-time operation is used on forms mode, terminals when each keystroke is significant.



SINGLE I/O BUFFERING

## 2. Double Buffer

In this technique the operating system Uses two buffers to allow continuous data transfer between two process or two devices.

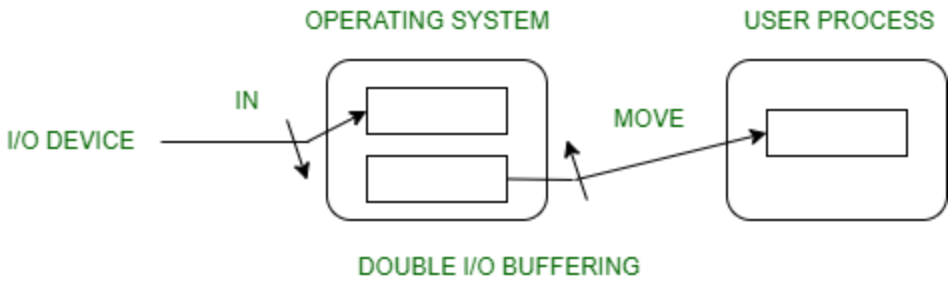
**Block Oriented**

- There are two buffers in the system.
- One buffer is used by the driver or controller to store data while waiting for it to be taken by higher level of the hierarchy.
- Other buffer is used to store data from the lower level module.
- Double buffering is also known as buffer swapping.

- A major disadvantage of double buffering is that the complexity of the process get increased.
- If the process performs rapid bursts of I/O, then using double buffering may be deficient.

**Stream Oriented**

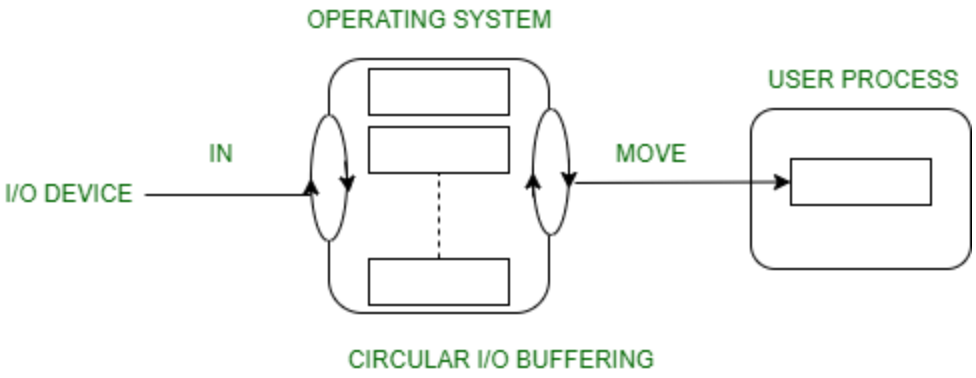
- Line- at a time I/O, the user process need not be suspended for input or output, unless process runs ahead of the double buffer.
- Byte- at a time operations, double buffer offers no advantage over a single buffer of twice the length.




## 3. Circular Buffer

In this technique the **operating system** Uses a circular buffer to manage continuous data streams efficiently.

- When more than two buffers are used, the collection of buffers is itself referred to as a circular buffer.
- In this, the data do not directly passed from the producer to the consumer because the data would change due to overwriting of buffers before they had been consumed.
- The producer can only fill up to buffer i-1 while data in buffer i is waiting to be consumed.



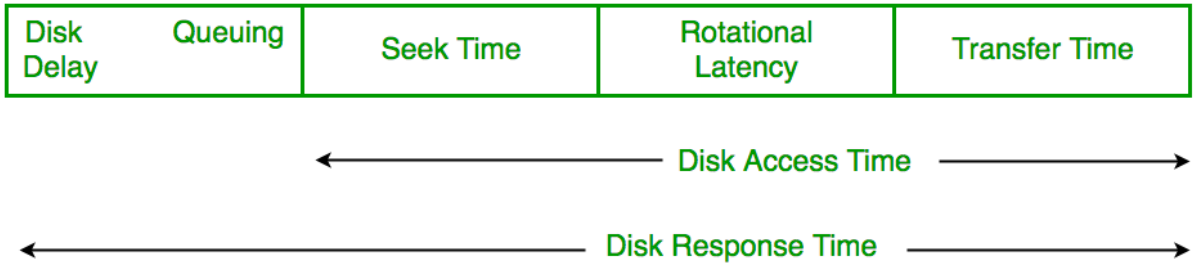

## What are Disk Scheduling Algorithms?

Disk scheduling algorithms are crucial in managing how data is read from and written to a computer's hard disk. These algorithms help determine the order in which disk read and write requests are processed, significantly impacting the speed and efficiency of data access. Common disk scheduling methods include First-Come, First-Served (FCFS), Shortest Seek Time First (SSTF), SCAN, C-SCAN, LOOK, and C-LOOK. By understanding and implementing these algorithms, we can optimize system performance and ensure faster data retrieval.

### Key Terms Associated with Disk Scheduling

- **Seek Time:** Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or written. So the disk scheduling algorithm that gives a minimum average seek time is better.
- **Rotational Latency:** Rotational Latency is the time taken by the desired sector of the disk to rotate into a position so that it can access the read/write heads. So the disk scheduling algorithm that gives minimum rotational latency is better.
- **Transfer Time:** Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and the number of bytes to be transferred.
- **Disk Access Time:**

> Disk Access Time = Seek Time + Rotational Latency + Transfer Time  Total Seek Time = Total head Movement * Seek Time




*Disk Access Time and Disk Response Time*

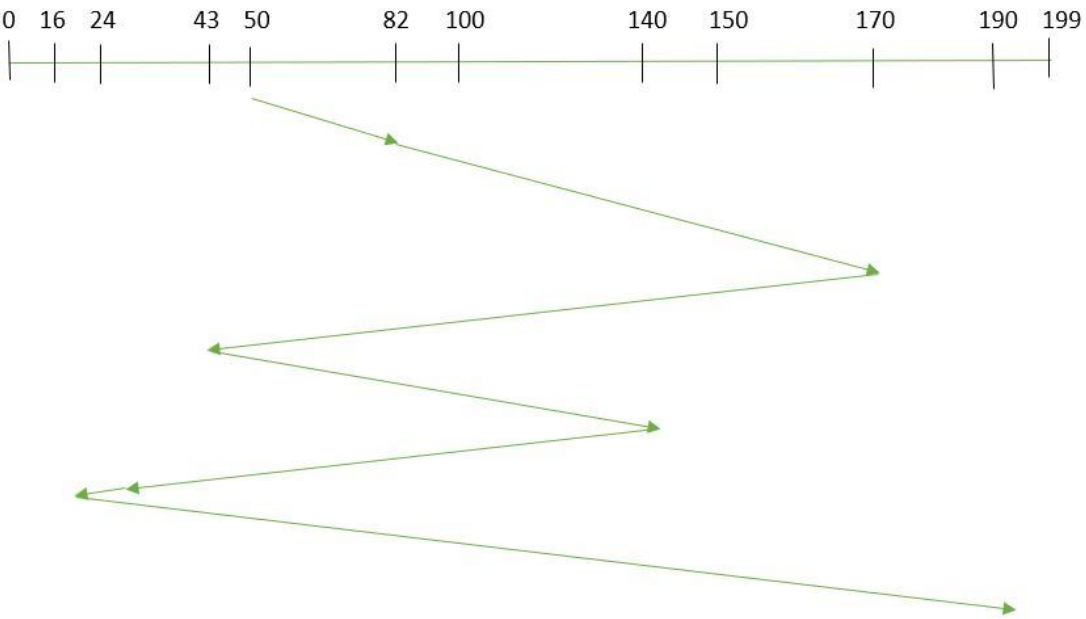
- **Disk Response Time:** Response Time is the average time spent by a request waiting to perform its I/O operation. The average *Response time* is the response time of all requests. *Variance Response Time* is the measure of how individual requests are serviced with respect to average response time. So the disk scheduling algorithm that gives minimum variance response time is better.

## Goal of Disk Scheduling Algorithms

- Minimize Seek Time
- Maximize Throughput
- Minimize Latency
- Fairness
- Efficiency in Resource Utilization

## 1. FCFS (First Come First Serve)

**FCFS** is the simplest of all Disk Scheduling Algorithms. In FCFS, the requests are addressed in the order they arrive in the disk queue. Let us understand this with the help of an example.




*First Come First Serve*

**Example:**

Suppose the order of request is- (82,170,43,140,24,16,190)

And current position of Read/Write head is: 50

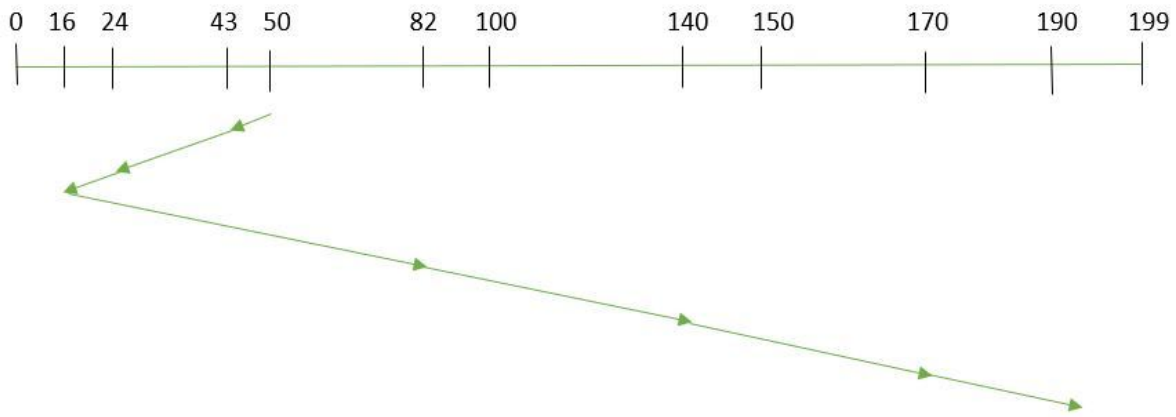
```
So, total overhead movement  (total distance covered by the d
isk arm) =
(82-50)+(170-82)+(170-43)+(140-43)+(140-24)+(24-16)+(190-16)
=642
```

## 2. SSTF (Shortest Seek Time First)

In **SSTF (Shortest Seek Time First)**, requests having the shortest seek time are executed first. So, the seek time of every request is calculated in advance in the queue and then they are scheduled according to their calculated seek time. As a result, the request near the disk arm will get executed first. SSTF is certainly an

improvement over FCFS as it decreases the average response time and increases the throughput of the system. Let us understand this with the help of an example.

## Example:




*Shortest Seek Time First*

Suppose the order of request is- (82,170,43,140,24,16,190)

And current position of Read/Write head is: 50

So,

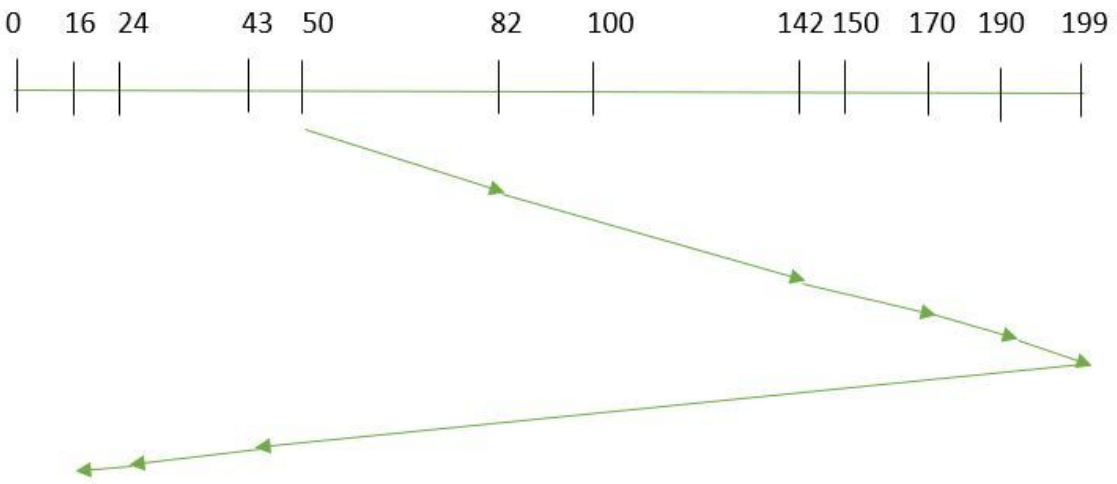
```
total overhead movement  (total distance covered by the disk
arm) =
(50-43)+(43-24)+(24-16)+(82-16)+(140-82)+(170-140)+(190-170)
=208
```

## 3. SCAN

In the **SCAN algorithm** the disk arm moves in a particular direction and services the requests coming in its path and after reaching the end of the disk, it reverses its direction and again services the request arriving in its path. So, this algorithm works as an elevator and is hence also known as an **elevator algorithm.** As a

result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

**Example:**




*SCAN Algorithm*

Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move **“towards the larger value”.**

Therefore, the total overhead movement (total distance covered by the disk arm) is calculated as

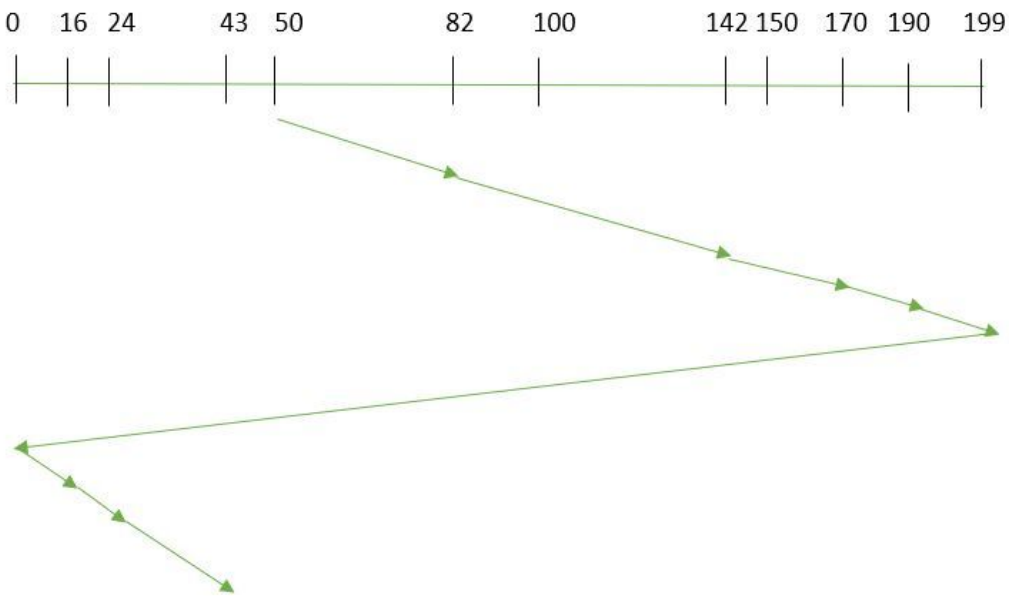
```
= (199-50) + (199-16) = 332
```

## 4. C-SCAN

In the **SCAN algorithm**, the disk arm again scans the path that has been scanned, after reversing its direction. So, it may be possible that too many requests are waiting at the other end or there may be zero or few requests pending at the scanned area.

These situations are avoided in the *CSCAN* algorithm in which the disk arm instead of reversing its direction goes to the other end of the disk and starts servicing the requests from there. So, the disk arm moves in a circular fashion and this algorithm is also similar to the SCAN algorithm hence it is known as C-SCAN (Circular SCAN).

## Example:




*Circular SCAN*

Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move **“towards the larger value”.**

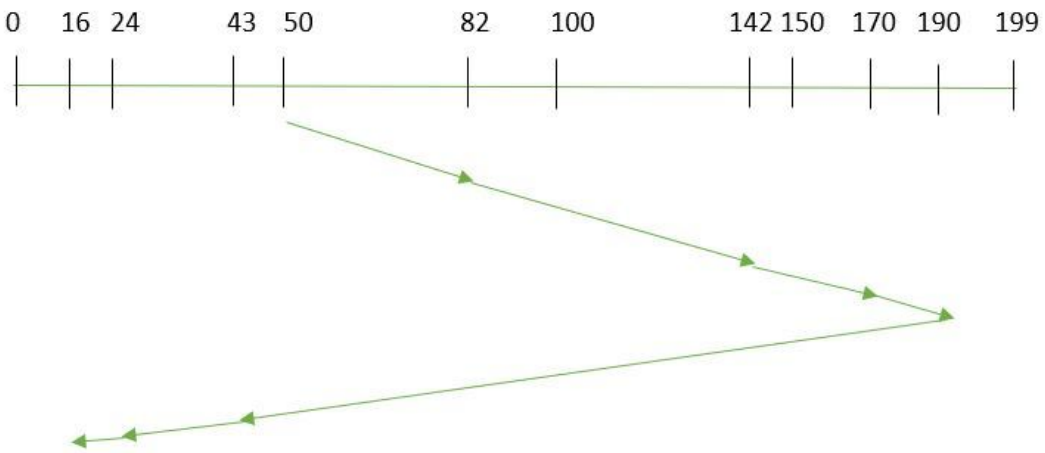
So, the total overhead movement  (total distance covered by the disk arm) is calculated as:

```
=(199-50) + (199-0) + (43-0) = 391
```

## 5. LOOK

**LOOK Algorithm** is similar to the SCAN disk scheduling algorithm except for the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only. Thus it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

## Example:




*LOOK Algorithm*

Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move **“towards the larger value”.**

So, the total overhead movement  (total distance covered by the disk arm) is calculated as:

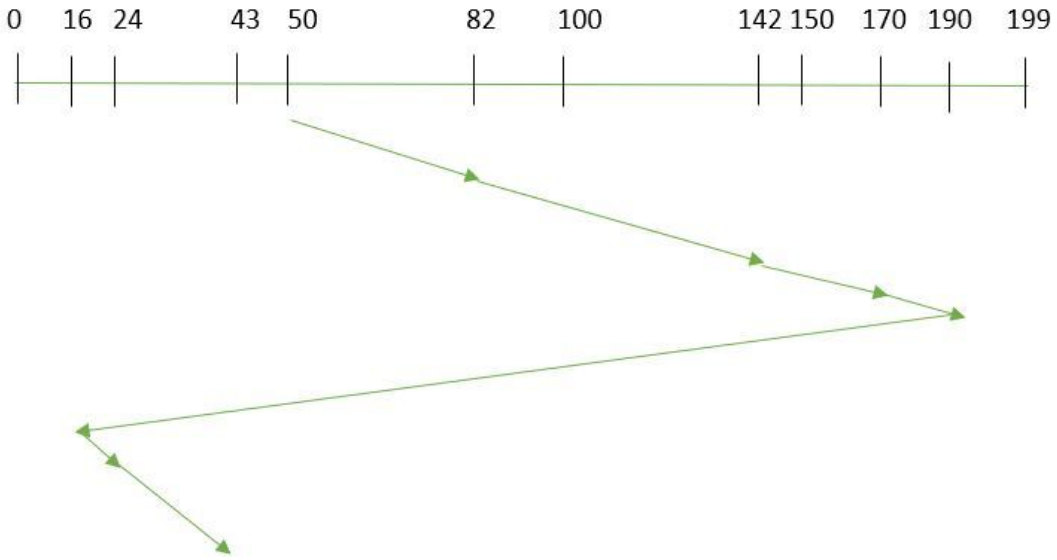
```
= (190-50) + (190-16) = 314
```

## 6. C-LOOK

As LOOK is similar to the SCAN algorithm, in a similar way, **C-LOOK** is similar to the CSCAN disk scheduling algorithm. In CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

## Example:

1. Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move **"towards the larger value"**




*C-LOOK*

So, the total overhead movement (total distance covered by the disk arm) is calculated as

```
= (190-50) + (190-16) + (43-16) = 341
```

# File Systems in Operating System

A computer file is defined as a medium used for saving and managing data in the computer system. The data stored in the computer system is completely in digital format, although there can be various types of files that help us to store the data.

File systems are a crucial part of any operating system, providing a structured way to store, organize, and manage data on storage devices such as hard drives, SSDs, and USB drives. Essentially, a file system acts as a bridge between the operating system and the physical storage hardware, allowing users and applications to create, read, update, and delete files in an organized and efficient manner.

## What is a File System?

A file system is a method an operating system uses to store, organize, and manage files and directories on a storage device. Some common types of file systems include:

- **FAT (File Allocation Table):** An older file system used by older versions of Windows and other operating systems.
- **NTFS (New Technology File System):** A modern file system used by Windows. It supports features such as file and folder permissions, compression, and encryption.
- **ext (Extended File System):** A file system commonly used on **Linux** and **Unix**based operating systems.
- **HFS (Hierarchical File System):** A file system used by macOS.
- **APFS (Apple File System):** A new file system introduced by Apple for their Macs and iOS devices.

A file is a collection of related information that is recorded on secondary storage. Or file is a collection of logically related entities. From the user's perspective, a file is the smallest allotment of logical secondary storage.

**The name  of the file is divided into two parts as shown below:**

- Name
- Extension, separated by a period.

## Issues Handled By File System

We've seen a variety of data structures where the file could be kept. The file system's job is to keep the files organized in the best way possible.

A free space is created on the hard drive whenever a file is deleted from it. To reallocate them to other files, many of these spaces may need to be recovered. Choosing where to store the files on the **hard disc** is the main issue with files one block may or may not be used to store a file. It may be kept in the disk's non-contiguous blocks. We must keep track of all the blocks where the files are partially located.

# Record Blocking

**Record Blocking** is a technique used in file systems to organize and transfer data efficiently between memory and storage devices. Since data on storage devices (like disks) is transferred in fixed-sized blocks, **record blocking** involves grouping smaller logical records into blocks for efficient storage and retrieval.

---

## Why Record Blocking is Needed

1. **Disk I/O Efficiency**:
    - Disks are designed to read and write data in large, fixed-size blocks. Operating on smaller records would be inefficient due to frequent I/O operations.
    - Grouping records into a block reduces the number of I/O operations.
2. **Storage Efficiency**:
    - If each logical record were stored separately, unused portions of the storage block (called internal fragmentation) would waste space.

---

## Types of Record Blocking

1. **Fixed-Length Blocking**:
    - Fixed-sized logical records are packed into fixed-sized physical blocks.
    - **Example**: A block of size 1024 bytes can contain 4 records of 256 bytes each.

- **Advantages**: Simple implementation; predictable performance.
- **Disadvantages**: May result in **internal fragmentation** if the record size does not evenly divide the block size.

2. **Variable-Length Spanned Blocking**:
    - Logical records of variable sizes are packed into blocks, and records can span across multiple blocks if needed.
    - **Example**: A record larger than the block size (e.g., 1200 bytes in a 1024-byte block) is split between two blocks.
    - **Advantages**: Maximizes storage efficiency.
    - **Disadvantages**: Complicates retrieval as multiple blocks may need to be accessed for a single record.
3. **Variable-Length Unspanned Blocking**:
    - Logical records of variable sizes are packed into blocks, but no record spans across blocks. If a record doesn't fit, the remaining space in the block is left unused.
    - **Advantages**: Simplifies retrieval since each record resides entirely within a single block.
    - **Disadvantages**: May lead to **external fragmentation** (wasted space within blocks).

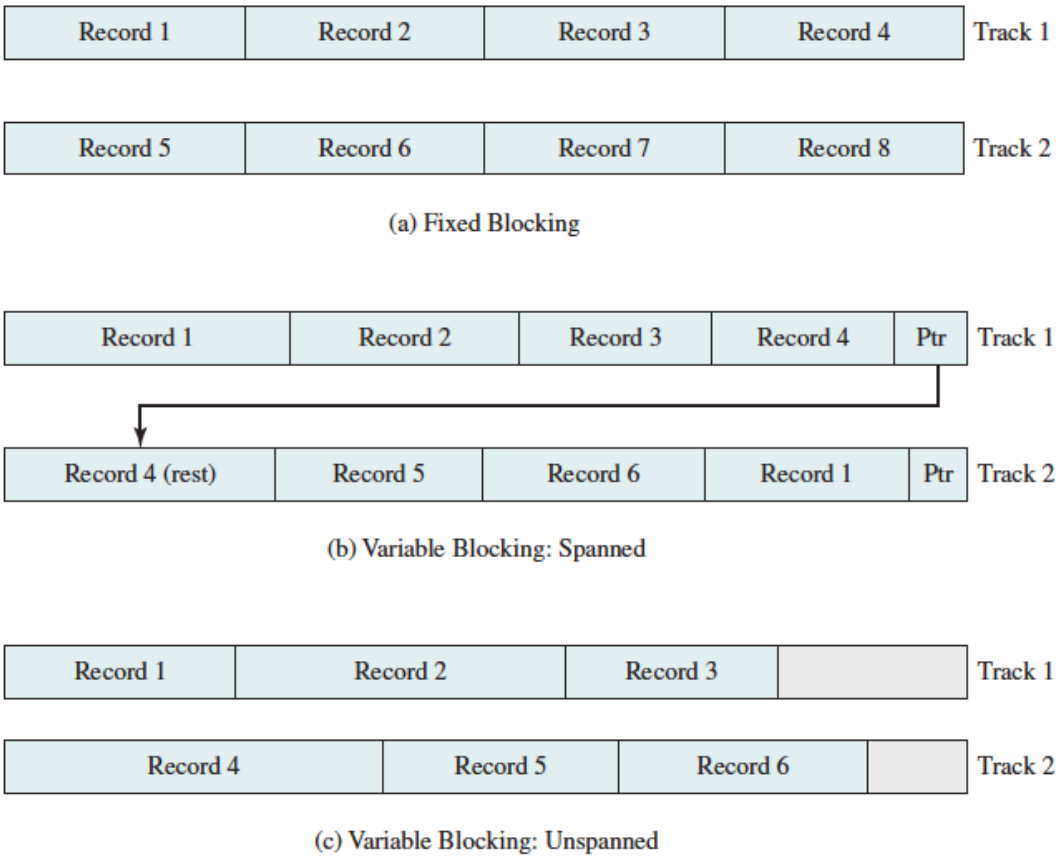


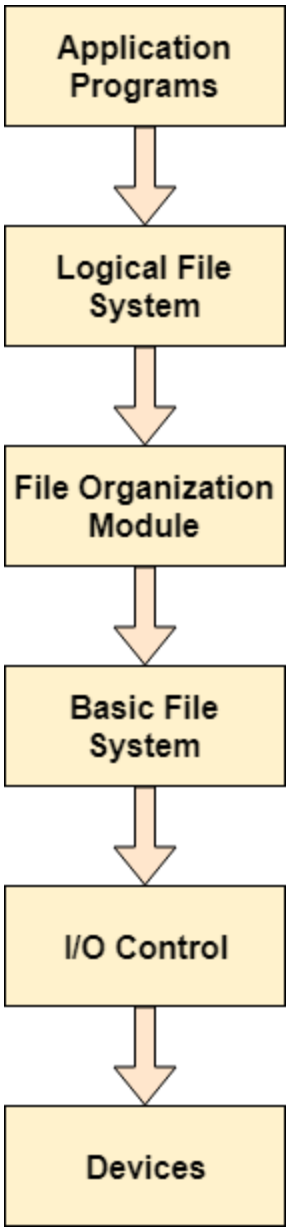

**Figure 12.8** **Record-Blocking Methods**

## File System Structure

File System provide efficient access to the disk by allowing data to be stored, located and retrieved in a convenient way. A file System must be able to store the file, locate the file and retrieve the file.

Most of the Operating Systems use layering approach for every task including file systems. Every layer of the file system is responsible for some activities.

The image shown below, elaborates how the file system is divided in different layers, and also the functionality of each layer.




- When an application program asks for a file, the first request is directed to the logical file system. The logical file system contains the Meta data of the file and directory structure. If the application program doesn't have the required permissions of the file then this layer will throw an error. Logical file systems also verify the path to the file.
- Generally, files are divided into various logical blocks. Files are to be stored in the hard disk and to be retrieved from the hard disk. Hard disk is divided into various tracks and sectors. Therefore, in order to store and retrieve the files,

the logical blocks need to be mapped to physical blocks. This mapping is done by File organization module. It is also responsible for free space management.

- Once File organization module decided which physical block the application program needs, it passes this information to basic file system. The basic file system is responsible for issuing the commands to I/O control in order to fetch those blocks.
- I/O controls contain the codes by using which it can access hard disk. These codes are known as device drivers. I/O controls are also responsible for handling interrupts.

# File Access Methods in Operating System

File access methods in an operating system are the techniques and processes used to read from and write to files stored on a computer's storage devices. There are several ways to access this information in the file. Some systems provide only one access method for files. Other systems, such as those of IBM, support many access methods, and choosing the right one for a particular application is a major design problem.

## File Access Methods

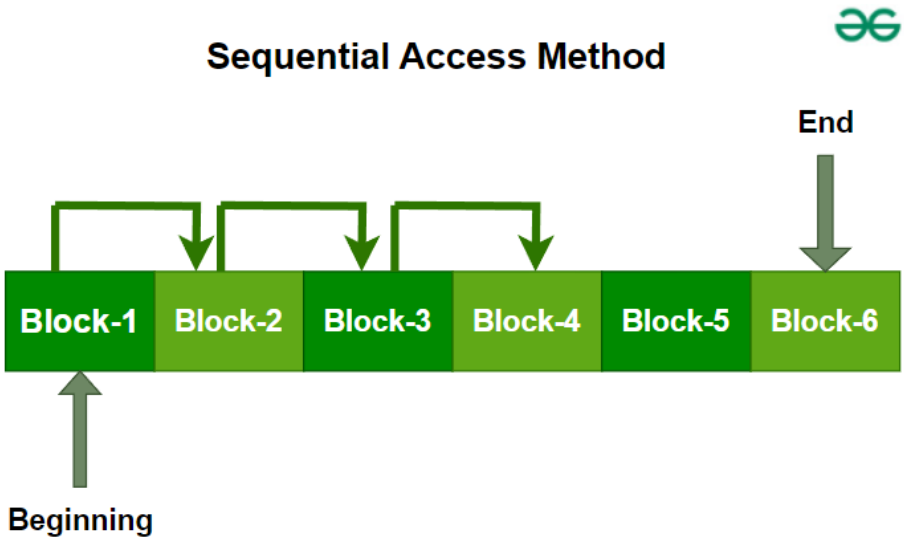
There are three ways to access a file in a computer system:

- Sequential-Access
- Direct Access
- Index sequential Method

## Sequential Access

It is the simplest access method. Information in the file is processed in order, one record after the other. This mode of access is by far the most common; for example, the editor and compiler usually access the file in this fashion.

Read and write make up the bulk of the operation on a file. A read operation -*read next*- reads the next position of the file and automatically advances a file pointer, which keeps track of the I/O location. Similarly, for the -write *next*- append to the end of the file and advance to the newly written material.




*Sequential Access Method*

## Key Points related to Sequential Access

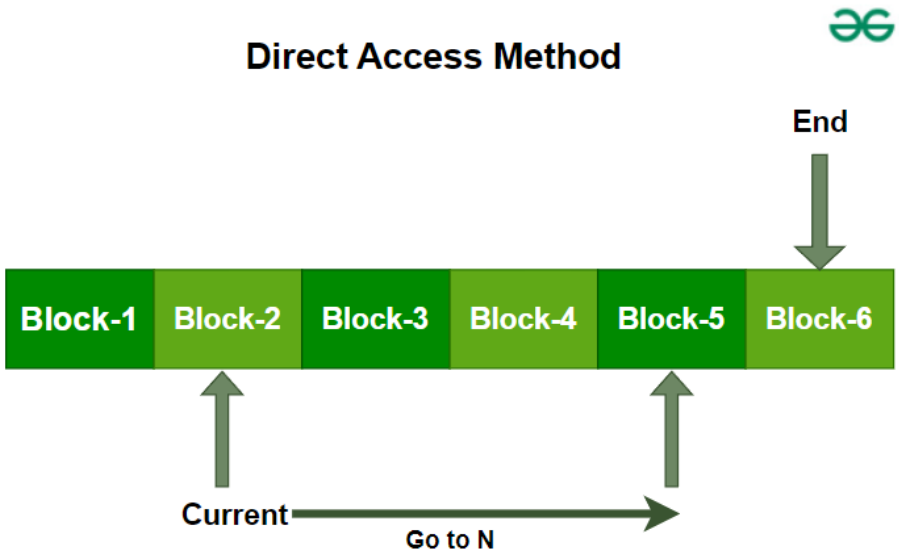
- Data is accessed from one record right after another record in an order.
- When we use the read command, it moves ahead pointer by one.
- When we use the write command, it will allocate memory and move the pointer to the end of the file.
- Such a method is reasonable for tape.

## Direct Access Method

Another method is *direct access method* also known as *relative access method*. A fixed-length logical record that allows the program to read and write record rapidly. in no particular order. The direct access is based on the disk model of a file since disk allows random access to any file block. For direct access, the file is viewed as a numbered sequence of block or record. Thus, we may read block 14

then block 59, and then we can write block 17. There is no restriction on the order of reading and writing for a direct access file.

A block number provided by the user to the operating system is normally a *relative block number*, the first relative block of the file is 0 and then 1 and so on.




*Direct Access Method*

### Index Sequential method

It is the other method of accessing a file that is built on the top of the sequential access method. These methods construct an index for the file. The index, like an index in the back of a book, contains the pointer to the various blocks. To find a record in the file, we first search the index, and then by the help of pointer we access the file directly.

### Key Points Related to Index Sequential Method

- It is built on top of Sequential access.
- It control the pointer by using index.

## File Directories

The collection of files is a file directory. The directory contains information about the files, including attributes, location, and ownership. Much of this information,

especially that is concerned with storage, is managed by the operating system. The directory is itself a file, accessible by various file management routines.

Below are information contained in a device directory.

- Name
- Type
- Address
- Current length
- Maximum length
- Date last accessed
- Date last updated
- Owner id
- Protection information

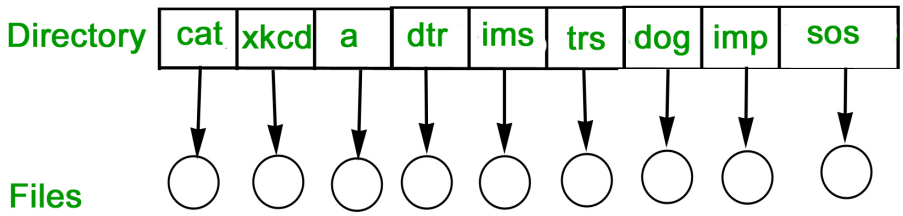
**The operation performed on the directory are:**

- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system

## Single-Level Directory

In this, a single directory is maintained for all the users.

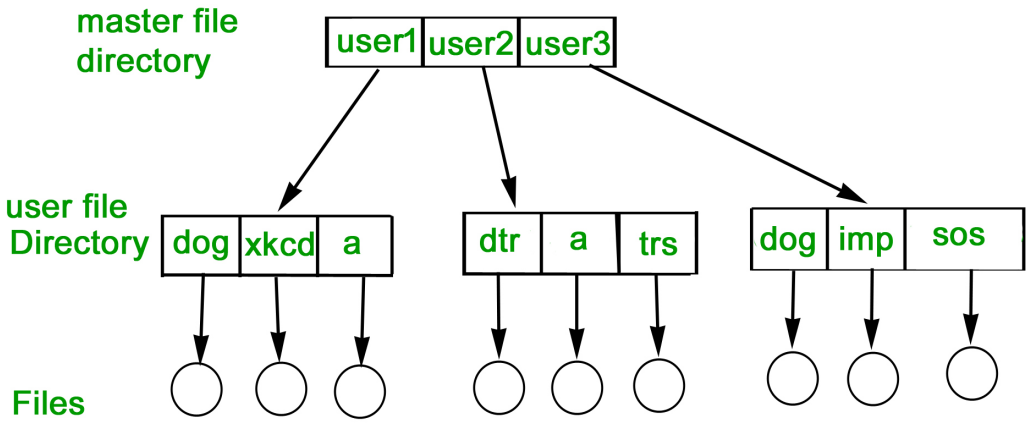
- **Naming Problem:** Users cannot have the same name for two files.
- **Grouping Problem:** Users cannot group files according to their needs.




## Two-Level Directory

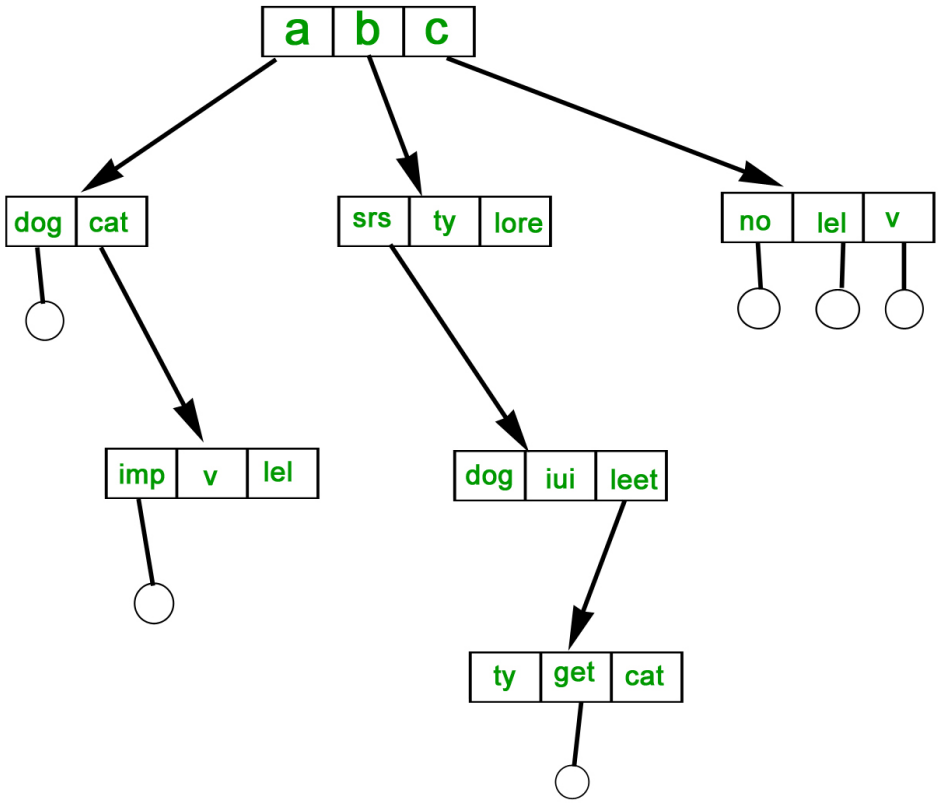
In this separate directories for each user is maintained.

- **Path Name**: Due to two levels there is a path name for every file to locate that file.
- Now, we can have the same file name for different users.
- Searching is efficient in this method.




## Tree-Structured Directory

The directory is maintained in the form of a tree. Searching is efficient and also there is grouping capability. We have absolute or relative path name for a file.




# File Sharing in OS

File Sharing in an Operating System(OS) denotes how information and files are shared between different users, computers, or devices on a network; and files are units of data that are stored in a computer in the form of documents/images/videos or any others types of information needed.

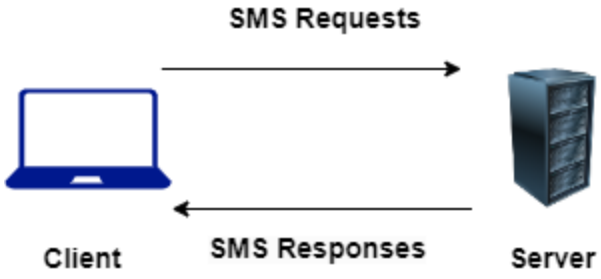
## Various Ways to Achieve File Sharing

Let's see the various ways through which we can achieve file sharing in an OS.

## 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/printer on a network. SMB is now the standard way for seamless file transfer method and printer sharing.

**Example:** Imagine in a company where the employees have to share the files on a particular project . Here SMB is employed to share files among all the windows based operating system.orate on projects. SMB/CIFS is employed to share files between Windows-based computers. Users can access shared folders on a server, create, modify, and delete files.

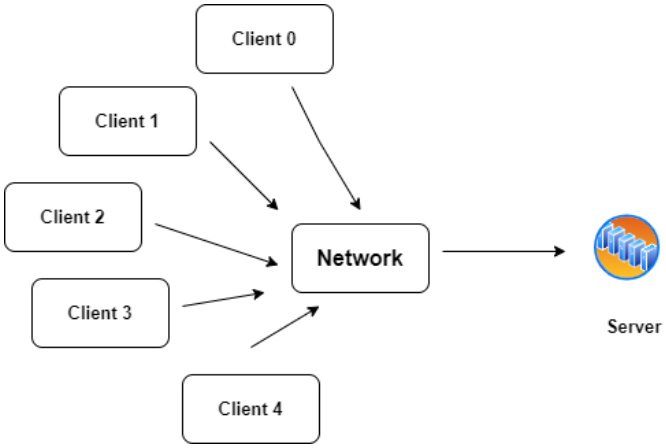



*SMB File Sharing*

## 2. Network File System (NFS)

NFS is a distributed based file sharing protocol mainly used in Linux/Unix based operating System. It allows a computer to share files over a network as if they were based on local. It provides a efficient way of transfer of files between servers and clients.

**Example:** Many Programmer/Universities/Research Institution uses Unix/Linux based Operating System. The Institutes puts up a global server datasets using NFS. The Researchers and students can access these shared directories and everyone can collaborate on it.



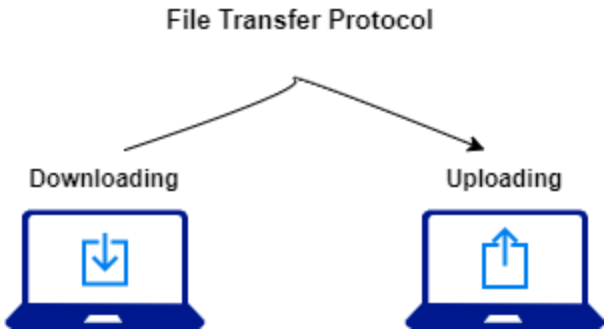
*NFS File Sharing*

## 3. File Transfer Protocol (FTP)

It is the most common standard protocol for transferring of the files between a client and a server on a computer network. FTPs supports both uploading and downloading of the files, here we can download,upload and transfer of files from Computer A to Computer B over the internet or between computer systems.

**Example:** Suppose the developer makes changes on the server. Using the FTP protocol, the developer connects to the server they can update the server with new website content and updates the existing file over there.

Read more about FTP: **FTP and it's implementation**




*FTP File Sharing*

## 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 that with others, and providing access from many users. It includes collaboration in realtime file sharing and version control access.

Ex: Several students working on a project and they can use Google Drive to store and share for that purpose. They can access the files from any computer or mobile devices and they can make changes in realtime and track the changes over there.

*Cloud Based File Sharing*

These all file sharing methods serves different purpose and needs according to the requirements and flexibility of the users based on the operating system.

## Secondary Storage Management Overview

Secondary storage refers to non-volatile memory like hard disks, SSDs, and optical drives used for storing data persistently. The operating system manages secondary storage to ensure efficient data access, reliability, and space utilization.

---

### Key Responsibilities of Secondary Storage Management

1. **Space Allocation**:
    - The OS allocates space for files and directories.
    - Methods include **contiguous**, **linked**, or **indexed allocation**.
2. **Free Space Management**:
    - Tracks unused space on the storage device.
    - Techniques include:
        - Bitmaps: Each bit represents a block (0 = free, 1 = allocated).
        - Linked Lists: Free blocks are connected using pointers.
3. **File System Support**:

- The OS implements file systems (e.g., NTFS, ext4) to organize and manage secondary storage.

4. **Disk Scheduling**:
   - Optimizes the order of read/write operations to minimize seek time (e.g., **FCFS**, **SSTF**, **SCAN** algorithms).
5. **Data Integrity and Reliability**:
   - Prevents data loss through backup mechanisms and redundancy techniques (e.g., **RAID**).

---

## Storage Allocation Techniques

1. **Contiguous Allocation**:
   - Files occupy contiguous blocks on disk.
   - **Advantage**: Fast sequential access.
   - **Disadvantage**: External fragmentation.
2. **Linked Allocation**:
   - Each file is a linked list of disk blocks.
   - **Advantage**: No fragmentation.
   - **Disadvantage**: Slower random access due to pointer traversal.
3. **Indexed Allocation**:
   - An index block contains pointers to the actual file blocks.
   - **Advantage**: Efficient for large files.
   - **Disadvantage**: Overhead of maintaining index blocks.

---

## Free Space Management

Efficient free space tracking ensures quick allocation of storage when needed.

1. **Bitmaps**:
   - Use a bit array where each bit represents a block.

- **Advantage**: Compact representation.
- **Disadvantage**: Requires processing to find consecutive free blocks.

2. **Linked Lists**:
   - Free blocks are connected in a linked list.
   - **Advantage**: Simple to implement.
   - **Disadvantage**: Traversal overhead.

---

## Data Reliability Techniques

1. **RAID (Redundant Array of Independent Disks)**:
   - A data storage technology combining multiple physical disks for performance or redundancy.
   - RAID levels include:
     - **RAID 0**: Striping (performance).
     - **RAID 1**: Mirroring (redundancy).
     - **RAID 5**: Distributed parity (balance of both).
2. **Backup and Recovery**:
   - Regular data backups prevent loss due to hardware failure or corruption.