MODULE-6

I/O devices

Input Device Definition: A piece of equipment/hardware which helps us enter data into a computer is called an input device. For example keyboard, mouse, etc.

Output Device Definition: A piece of equipment/hardware which gives out the result of the entered input, once it is processed (i.e. converts data from machine language to a human-understandable language), is called an output device. For example printer, monitor, etc.

List of Input Devices

Given below is the list of the most common input devices along with brief information about each of them.

1. **Keyboard**

- A simple device comprising keys and each key denotes either an alphabet, number or number commands which can be given to a computer for various actions to be performed
- It has a modified version of typewriter keys
- The keyboard is an essential input device and computer and laptops both use keyboards to give commands to the computer

2. **Mouse**

- It is also known as a pointing device
- Using mouse we can directly click on the various icons present on the system and open up various files and programs
- A mouse comprises 3 buttons on the top and one trackball at the bottom which helps in selecting and moving the mouse around, respectively
- In case of laptops, the touchpad is given as a replacement of the mouse which helps in the movement of the mouse pointer

3. **Joy Stick**

- It is a device which comprises a stick which is attached at an angle to the base so that it can be moved and controlled
- Mostly used to control the movement in video games

- Apart from a computer system, a joystick is also used in the cockpit of an aeroplane, wheelchairs, cranes, trucks, etc. to operate them well

4. **Light Pen**

- It is a wand-like looking device which can directly be moved over the device's screen
- It is light-sensitive
- Used in conjunction with computer's cathode ray tube

5. **Microphone**

- Using a microphone, sound can be stored in a device in its digital form
- It converts sound into an electrical signal
- To record or reproduce a sound created using a microphone, it needs to be connected with an amplifier

6. **Scanner**

- This device can scan images or text and convert it into a digital signal
- When we place any piece of a document on a scanner, it converts it into a digital signal and displays it on the computer screen

7. **Barcode Reader**

- It is a kind of an optical scanner
- It can read bar codes
- A source of light is passed through a bar code, and its aspects and details are displayed on the screen

List of Output Device

The commonly used output devices have been listed below with a brief summary of what their function is and how they can be used.

1. **Monitor**

- The device which displays all the icons, text, images, etc. over a screen is called the Monitor
- When we ask the computer to perform an action, the result of that action is displayed on the monitor
- Various types of monitors have also been developed over the years

2. **Printer**

- A device which makes a copy of the pictorial or textual content, usually over a paper is called a printer

- For example, an author types the entire book on his/her computer and later gets a print out of it, which is in the form of paper and is later published
- Multiple types of printers are also available in the market, which can serve different purposes

3. **Speakers**

- A device through which we can listen to a sound as an outcome of what we command a computer to do is called a speaker
- Speakers are attached with a computer system and also are a hardware device which can be attached separately
- With the advancement in technology, speakers are now available which are wireless and can be connected using BlueTooth or other applications

4. **Projector**

- An optical device which presents an image or moving images onto a projection screen is called a projector
- Most commonly these projectors are used in auditoriums and movie theatres for the display of the videos or lighting
- If a projector is connected to a computer, then the image/video displayed on the screen is the same as the one displayed on the computer screen

5. **Headphones**

- They perform the same function as a speaker, the only difference is the frequency of sound
- Using speakers, the sound can be heard over a larger area and using headphones, the sound is only audible to the person using them
- Also known as earphones or headset

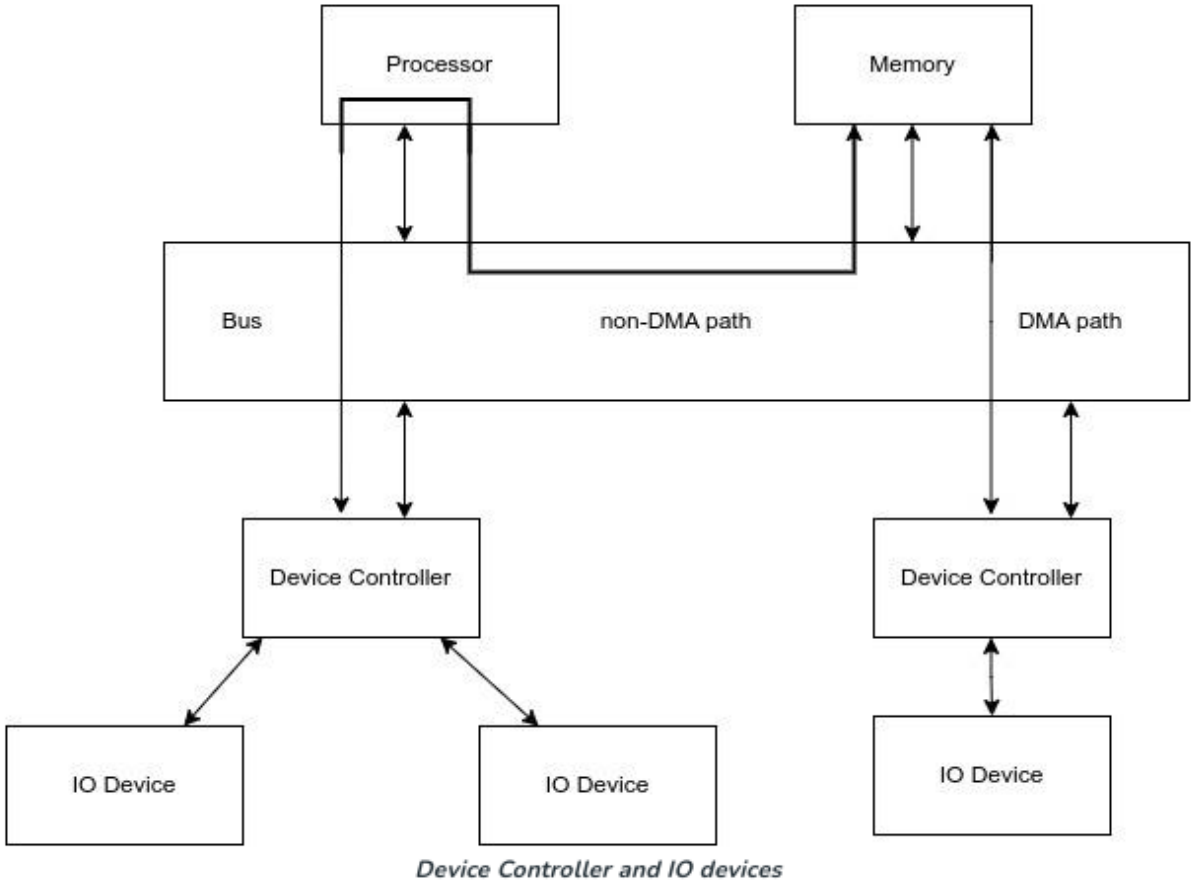
Device Controllers

In computer systems, I/O devices do not usually communicate with the operating system. The operating system manages their task with the help of one intermediate electronic device called a **device controller.**

- **Input devices** are those devices that generate data to give input to the computer system. Examples- mouse, trackball, keyboard, CD-ROM.
- **Output devices** accept data from the computer system. Examples- printer, graphics display screen, plotter.
- **Input/Output (**I/O devices**)** devices can give data as input and receive output as data from the computer system. Examples- disk, tape, writable CD.

The device controller knows how to communicate with the operating system as well as how to communicate with I/O devices. So device controller is an

interface between the computer system (operating system) and I/O devices. The device controller communicates with the system using the system bus. So how the device controller, I/O devices, and the system bus is connected is shown below in the diagram.



Device Controller and IO devices

In the above diagram, some IO devices have DMA (Direct Memory access) via device controllers and some of them do not have DMA. The devices which have a DMA path to communicate with the system to access memory are much faster than devices that have a non-DMA path to access the memory. The devices have a non-DMA path via the device controller to access the memory, they have to go from the processor which means it will be scheduled by the scheduler and then when it gets loaded into RAM then it will get the CPU to execute its instruction to access memory so it is slow from devices which has a DMA.

A device controller generally can control more than one IO device but it is most common to control only a single device. Device controllers are stored in

the chip and that chip is attached to the system bus there is a connection cable from the controller to each device which is controlled by it. Generally, one controller controls one device. The operating system communicates with device controllers and the device controller communicates with devices so indirectly operating system communicates with IO devices.

Nowadays, most controllers have Direct Memory Access to execute their tasks. And because of DMA, concurrent execution of process increases and so overall performance increases.

Direct Memory Access (DMA)

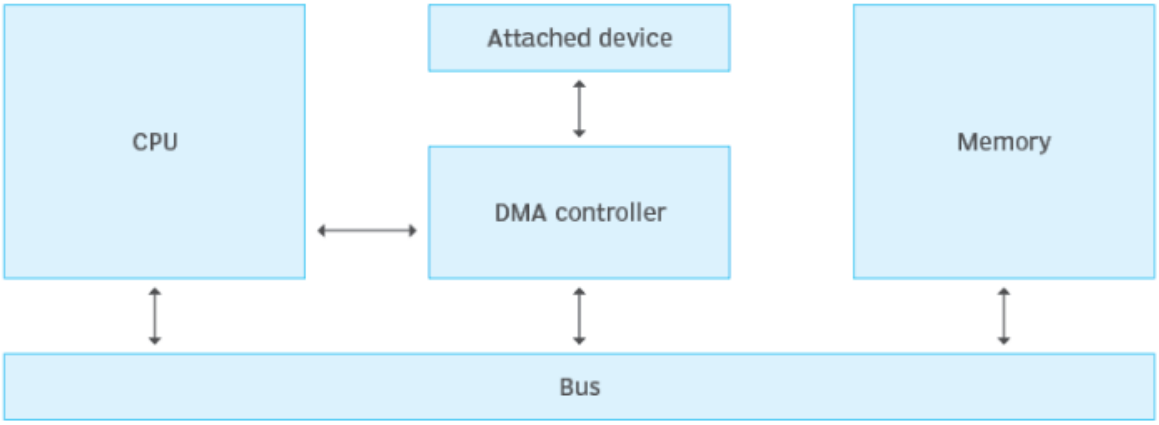
Direct Memory Access (DMA) is a capability provided by some computer bus architectures that enables data to be sent directly from an attached device, such as a disk drive, to the main memory on the computer's motherboard. The microprocessor, or central processing unit (CPU), is freed from involvement with the data transfer, speeding up overall computer operation.

DMA enables devices -- such as disk drives, external memory, graphics cards, network cards and sound cards -- to share and receive data from the main memory in a computer. It does this while still allowing the CPU to perform other tasks.

Without a process such as DMA, the computer's CPU becomes preoccupied with data requests from an attached device and is unable to perform other operations during that time. With DMA, a CPU initiates a data transfer with an attached device and can still perform other operations while the data transfer is in progress. DMA enables a computer to transfer data to and from devices with less CPU overhead.

An alternative to DMA is Ultra DMA, which provides a burst data transfer rate up to 33 megabytes per second (MBps). Hard drives that have Ultra DMA/33 also support programmed input/output (PIO) modes 1, 3 and 4, and multiword DMA mode 2 at 16.6 MBps.

Where Direct Memory Access fits in



How does DMA work?

Usually, a specified portion of memory is designated as an area to be used for direct memory access. For example, in the Industry Standard Architecture bus standard, up to 16 MB of memory can be addressed for DMA. Other bus standards might allow access to the full range of memory addresses. Peripheral component interconnect uses a bus master with the CPU delegating I/O control to the PCI controller.

DMA channels send data between an attached peripheral device and the system memory. System resources such as the CPU, memory, attached I/O devices and a DMA controller are connected through a bus line, which is also used for DMA channels. The DMA controller is used to start memory read/write cycles and to generate memory addresses.

The CPU gets the DMA controller to begin a data transfer and associate a memory address to use. The DMA controller sets the destination addresses and read/write lines to the system memory. It then changes the internal memory address with each transferred byte of data until a full block of data is transferred.

The DMA controller moves data to and from memory using one of the following methods:

- **Burst mode.** When the CPU gives the DMA controller access to the system bus, the DMA controller transfers the whole data block in one contiguous sequence. Once completed, control of the bus reverts back to the CPU. This method causes the CPU to be inactive while transferring data. The DMA controller initially gains access to the system bus using Bus Request and Bus Grant processes.
- **Cycle-stealing mode.** The DMA controller accesses the system bus in the same way as burst mode, but control of the system bus reverts to the CPU using a Bus Grant after one byte is transferred. Another Bus Request is initiated, and the process is repeated until the whole data block is transferred. The cycle-stealing transfer mode is useful in systems where controllers monitor data in real time.
- **Transparent mode.** The DMA controller transfers data only when the CPU is executing operations that don't use system buses. With this DMA transfer method, the CPU doesn't have to stop performing its operations. Transparent mode DMA operations takes the longest to transfer data blocks, but it's the most efficient mode in terms of system performance.

Types of Direct Memory Access (DMA)

There are four popular types of DMA.

- Single-Ended DMA
- Dual-Ended DMA
- Arbitrated-Ended DMA
- Interleaved DMA

Single-Ended DMA: Single-Ended DMA Controllers operate by reading and writing from a single memory address. They are the simplest DMA.

Dual-Ended DMA: Dual-Ended DMA controllers can read and write from two memory addresses. Dual-ended DMA is more advanced than single-ended DMA.

Arbitrated-Ended DMA: Arbitrated-Ended DMA works by reading and writing to several memory addresses. It is more advanced than Dual-Ended DMA.

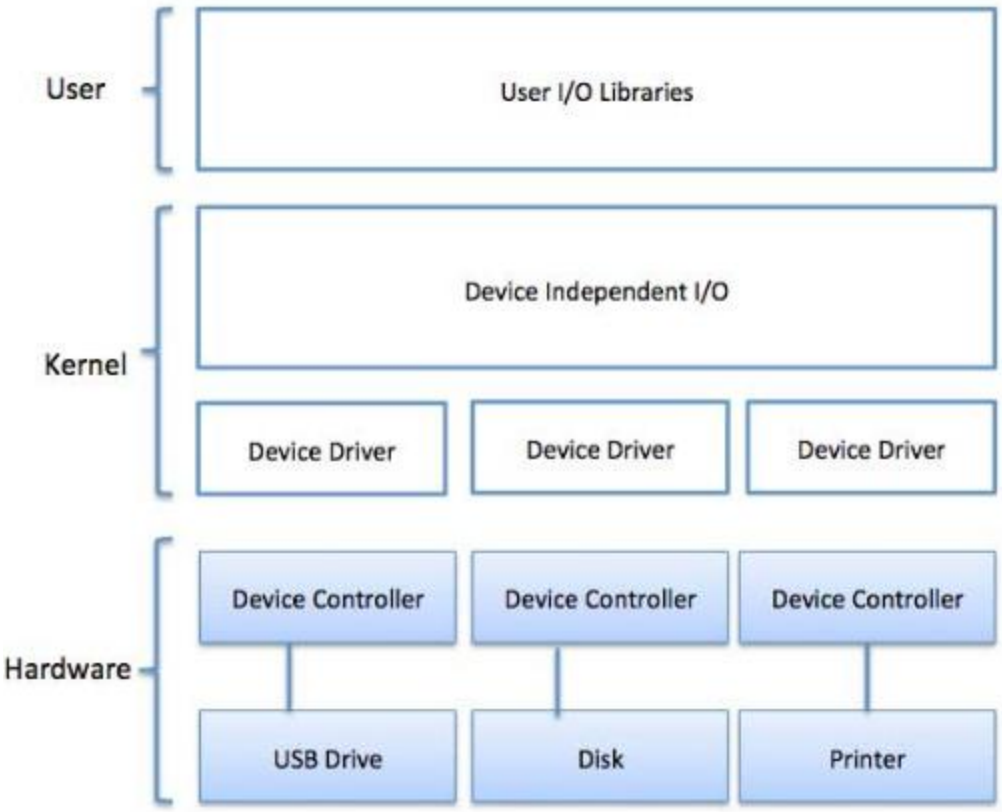
Interleaved DMA: Interleaved DMA are those DMA that read from one memory address and write from another memory address.

Principles of I/O Software:

I/O software is often organized in the following layers –

- **User Level Libraries** – This provides simple interface to the user program to perform input and output. For example, **stdio** is a library provided by C and C++ programming languages.
- **Kernel Level Modules** – This provides device driver to interact with the device controller and device independent I/O modules used by the device drivers.
- **Hardware** – This layer includes actual hardware and hardware controller which interact with the device drivers and makes hardware alive.

A key concept in the design of I/O software is that it should be device independent where it should be possible to write programs that can access any I/O device without having to specify the device in advance. For example, a program that reads a file as input should be able to read a file on a floppy disk, on a hard disk, or on a CD-ROM, without having to modify the program for each different device.



Device Drivers

Device drivers are software modules that can be plugged into an OS to handle a particular device. Operating System takes help from device drivers to handle all I/O devices. Device drivers encapsulate device-dependent code and implement a standard interface in such a way that code contains device-specific register reads/writes. Device driver, is generally written by the device's manufacturer and delivered along with the device on a CD-ROM.

A device driver performs the following jobs –

- To accept request from the device independent software above to it.
- Interact with the device controller to take and give I/O and perform required error handling

- Making sure that the request is executed successfully

How a device driver handles a request is as follows: Suppose a request comes to read a block N. If the driver is idle at the time a request arrives, it starts carrying out the request immediately. Otherwise, if the driver is already busy with some other request, it places the new request in the queue of pending requests.

Interrupt handlers

An interrupt handler, also known as an interrupt service routine or ISR, is a piece of software or more specifically a callback function in an operating system or more specifically in a device driver, whose execution is triggered by the reception of an interrupt.

When the interrupt happens, the interrupt procedure does whatever it has to in order to handle the interrupt, updates data structures and wakes up process that was waiting for an interrupt to happen.

The interrupt mechanism accepts an address — a number that selects a specific interrupt handling routine/function from a small set. In most architectures, this address is an offset stored in a table called the interrupt vector table. This vector contains the memory addresses of specialized interrupt handlers.

Device-Independent I/O Software

The basic function of the device-independent software is to perform the I/O functions that are common to all devices and to provide a uniform interface to the user-level software. Though it is difficult to write completely device independent software but we can write some modules which are common among all the devices. Following is a list of functions of device-independent I/O Software –

- Uniform interfacing for device drivers
- Device naming - Mnemonic names mapped to Major and Minor device numbers

- Device protection
- Providing a device-independent block size
- Buffering because data coming off a device cannot be stored in final destination.
- Storage allocation on block devices
- Allocation and releasing dedicated devices
- Error Reporting

User-Space I/O Software

These are the libraries which provide richer and simplified interface to access the functionality of the kernel or ultimately interactive with the device drivers. Most of the user-level I/O software consists of library procedures with some exception like spooling system which is a way of dealing with dedicated I/O devices in a multiprogramming system.

I/O Libraries (e.g., stdio) are in user-space to provide an interface to the OS resident device-independent I/O SW. For example putchar(), getchar(), printf() and scanf() are example of user level I/O library stdio available in C programming.

Kernel I/O Subsystem

Kernel I/O Subsystem is responsible to provide many services related to I/O. Following are some of the services provided.

- **Scheduling** – Kernel schedules a set of I/O requests to determine a good order in which to execute them. When an application issues a blocking I/O system call, the request is placed on the queue for that device. The Kernel I/O scheduler rearranges the order of the queue to improve the overall system efficiency and the average response time experienced by the applications.
- **Buffering** – Kernel I/O Subsystem maintains a memory area known as **buffer** that stores data while they are transferred between two devices or between a device with an application operation. Buffering is done to cope with a speed mismatch

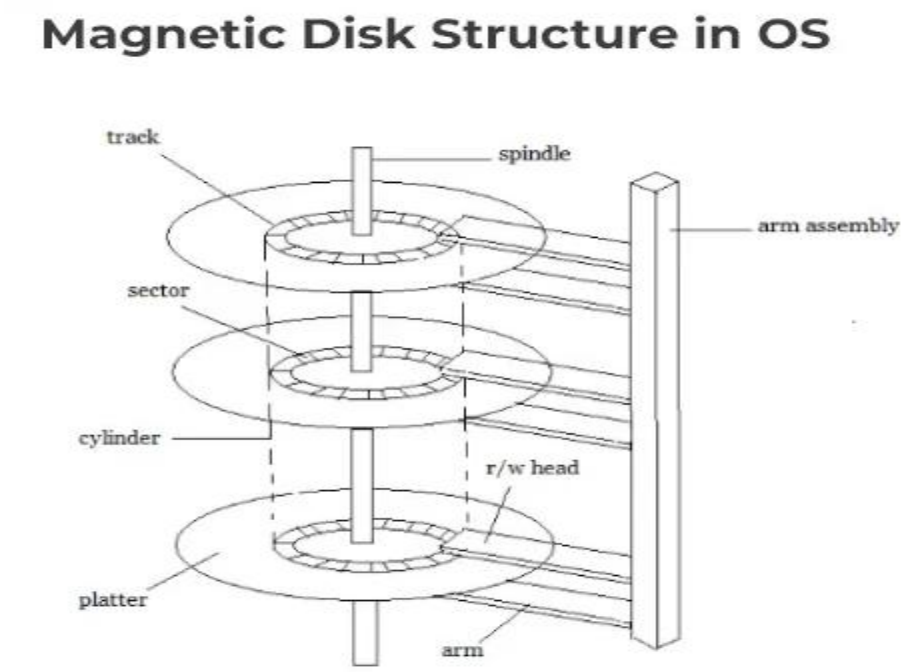
between the producer and consumer of a data stream or to adapt between devices that have different data transfer sizes.

- **Caching** – Kernel maintains cache memory which is region of fast memory that holds copies of data. Access to the cached copy is more efficient than access to the original.
- **Spooling and Device Reservation** – A spool is a buffer that holds output for a device, such as a printer, that cannot accept interleaved data streams. The spooling system copies the queued spool files to the printer one at a time. In some operating systems, spooling is managed by a system daemon process. In other operating systems, it is handled by an in kernel thread.
- **Error Handling** – An operating system that uses protected memory can guard against many kinds of hardware and application errors.

Magnetic Disk Structure

The length of the tracks near the centre is less than the length of the tracks farther from the centre.

- Each track is further divided into **sectors.**
- Spindle revolves the platters and is controlled by r/w unit of OS. Some advanced spindles have capability to only revolve a particular disk and keep others intact.
- Arm Assembly is there which keeps a pointy r/w head on each disk to read of write on a particular disk.
- A world cylinder may also be used at times to refer disk stack.



The speed of the disk is measured as two parts:

- **Transfer rate:** This is the rate at which the data moves from disk to the computer.
- **Random access time:** It is the sum of the seek time and rotational latency.

Seek time is the time taken by the arm to move to the required track. **Rotational latency** is defined as the time taken by the arm to reach the required sector in the track.

Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.

Disk Access Time is:

Seek Time + Rotational Latency + Transfer Time

Disk Response time is the average of all the tasks starting from disk i/o request till the time when first data output from disks is sent to the system (This is given wrong on GeeksforGeeks)

Even though the disk is arranged as sectors and tracks physically, the data is logically arranged and addressed as an array of blocks of fixed size. The size of a block can

be **512** or **1024** bytes. Each logical block is mapped with a sector on the disk, sequentially. In this way, each sector in the disk will have a logical address.

Disk Scheduling Algorithms

Disk scheduling is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O Scheduling.

Importance of Disk Scheduling in Operating System

- Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller. Thus other I/O requests need to wait in the waiting queue and need to be scheduled.
- Two or more requests may be far from each other so this can result in greater disk arm movement.
- Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner.

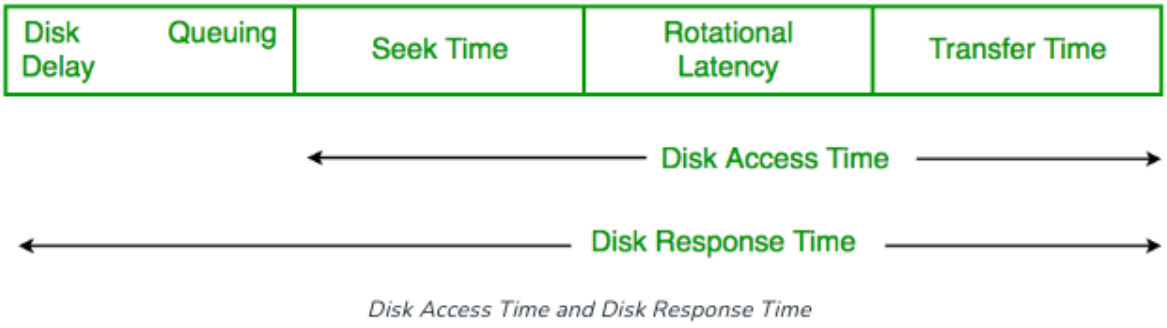
Here, we will cover the following Topics in Disk Scheduling.

- *Important Terms Associated with Disk Scheduling*
- *Disk Scheduling Algorithms*
    - *FCFS (First Come First Serve)*
    - *SSTF (Shortest Seek Time First)*
    - *SCAN (Elevator Algorithm)*
    - *C-SCAN (CIrcular SCAN)*
    - *LOOK*
    - *C-LOOK*
    - *RSS*
    - *LIFO (Last-In First-Out)*
    - *N-Step SCAN*
    - *F-SCAN*

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

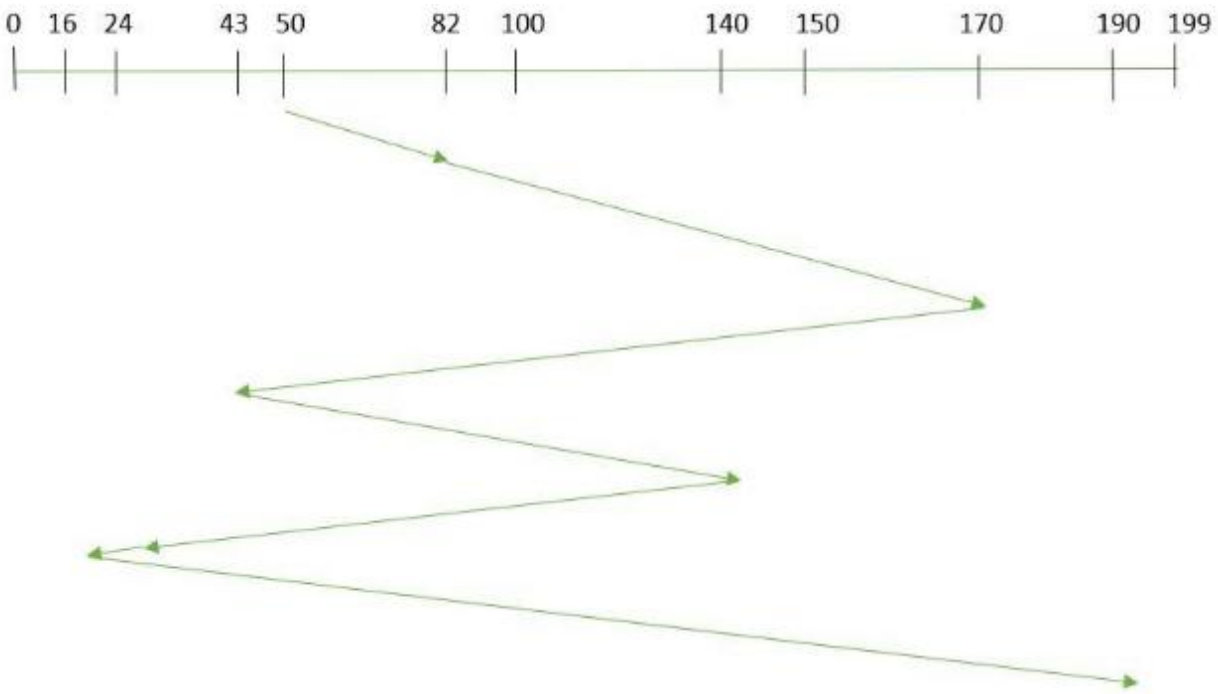
- **Disk Response Time:** Response Time is the average time spent by a request waiting to perform its I/O operation. The average *Response time* is the response time of all requests. *Variance Response Time* is the measure of how individual requests are serviced with respect to average response time.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

Disk Scheduling Algorithms

There are several Disk Several Algorithms. We will discuss each one of them.

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 disk arm)
=
(82-50)+(170-82)+(170-43)+(140-43)+(140-24)+(24-16)+(190-16) =642
```

Advantages of FCFS

Here are some of the advantages of First Come First Serve.

- Every request gets a fair chance
- No indefinite postponement

Disadvantages of FCFS

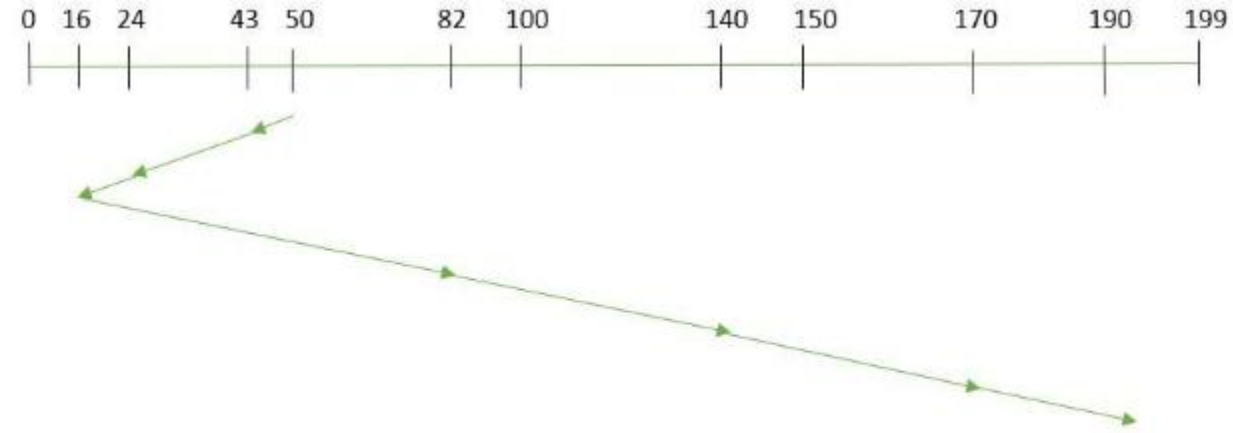
Here are some of the disadvantages of First Come First Serve.

- Does not try to optimize seek time
- May not provide the best possible service

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
```

Advantages of Shortest Seek Time First

Here are some of the advantages of Shortest Seek Time First.

- The average Response Time decreases
- Throughput increases

Disadvantages of Shortest Seek Time First

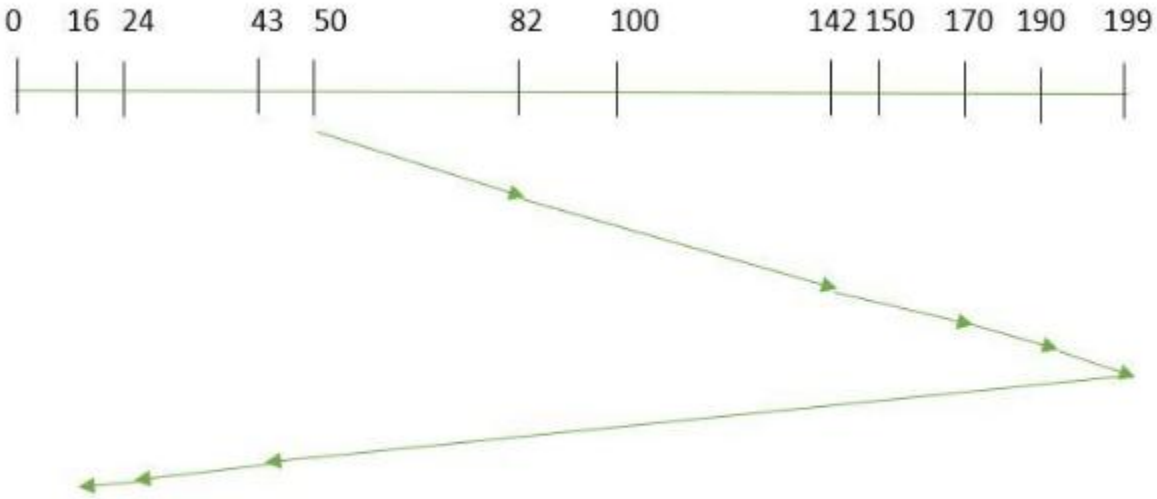
Here are some of the disadvantages of Shortest Seek Time First.

- Overhead to calculate seek time in advance
- Can cause Starvation for a request if it has a higher seek time as compared to incoming requests
- The high variance of response time as SSTF favors only some requests

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
```

Advantages of SCAN Algorithm

Here are some of the advantages of the SCAN Algorithm.

- High throughput
- Low variance of response time
- Average response time

Disadvantages of SCAN Algorithm

Here are some of the disadvantages of the SCAN Algorithm.

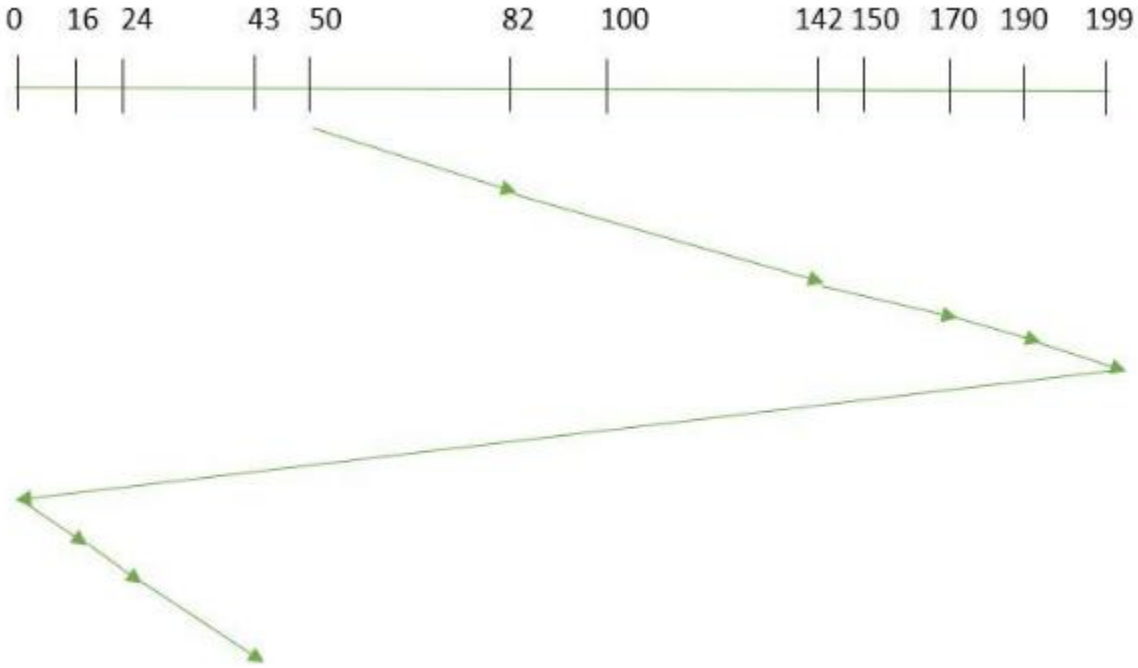
- Long waiting time for requests for locations just visited by disk arm

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
```

Advantages of C-SCAN Algorithm
Here are some of the advantages of C-SCAN.

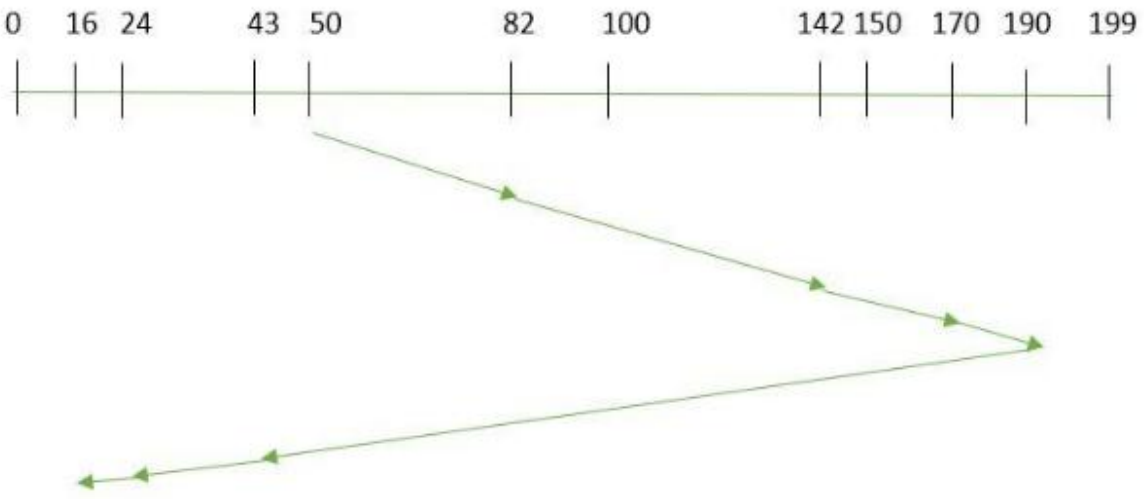
- Provides more uniform wait time compared to SCAN.

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:

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:

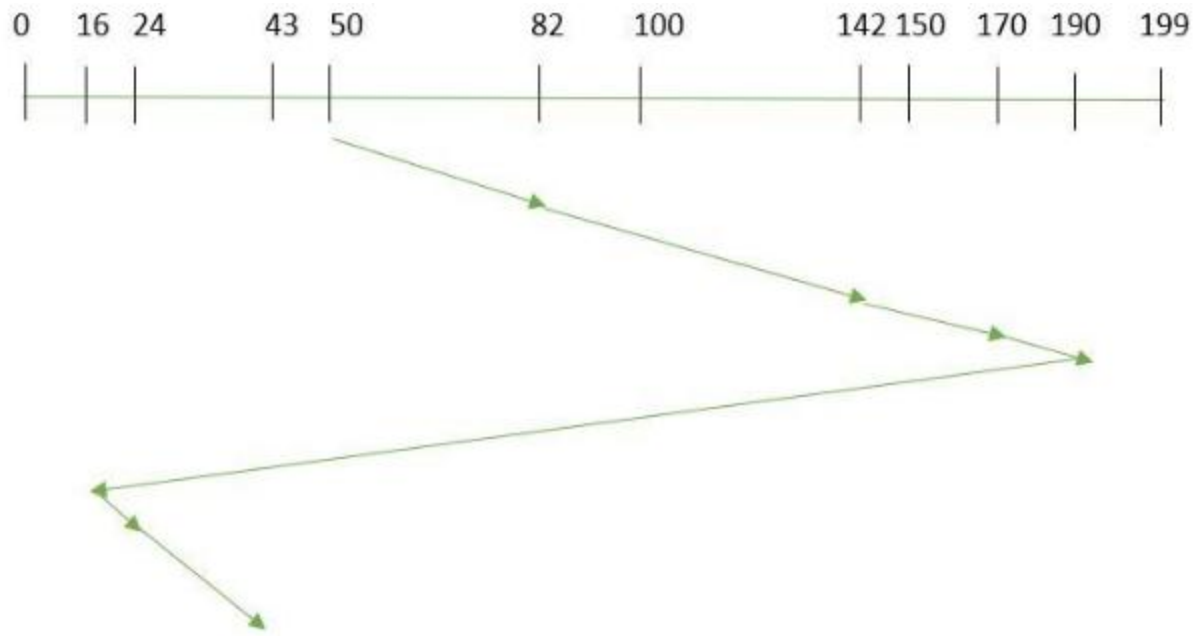
```
= (190-50) + (190-16) = 314
```

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
```

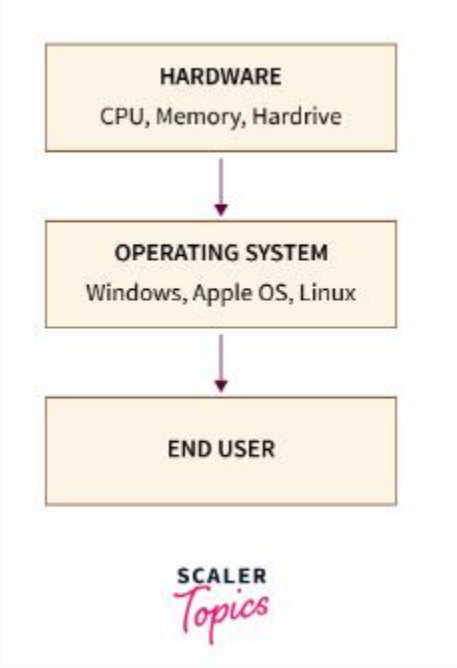
Concept of File

File management is one of the basic but important features provided by the operating system. File management in the operating system is nothing but software that handles or manages the files (binary, text, pdf, docs, audio, video, etc.) present in computer software.

The file system in the operating system is capable of managing individual as well as groups of files present in the computer system. The file system in the operating system tells us about the location, owner, time of creation and modification, type, and state of a file present on the computer system.

File Management in an Operating System?

Before learning about the file system in the operating system, let us first get a good grasp of operating systems and files.
Files is a collection of co-related information that is recorded in some format (such as text, pdf, docs, etc.) and is stored on various storage mediums such as flash drives, hard disk drives (HDD), magnetic tapes, optical disks, and tapes, etc. Files can be read-only or read-write. Files are simply used as a medium for providing input(s) and getting output(s).
Now, an Operating System is nothing but a software program that acts as an interface between the hardware, the application software, and the users. The main aim of an operating system is to manage all the computer resources. So, we can simply say that the operating system gives a platform to the application software and other system software to perform their task.
Refer to the diagram below to understand the value and working of the operating system.



The features of an operating system are:

1. Provide security to system and application software.
2. memory management.
3. disk management.

4. I/O operations.
5. file management, etc.

Some common types of file systems include:

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

File Access Methods

When a file is used, information is read and accessed into computer memory and there are several ways to access this information of 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.

There are three ways to access a file into a computer system: Sequential-Access, Direct Access, Index sequential Method.

1. **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, 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-* read the next position of the file and automatically advance a file pointer, which keeps track I/O location. Similarly, for the -write *next-* append to the end of the file and advance to the newly written material.
1.

   - Data is accessed one record right after another record in an order.
   - When we use read command, it move ahead pointer by one

- When we use write command, it will allocate memory and move the pointer to the end of the file
- Such a method is reasonable for tape.

Advantages of Sequential Access Method :

- It is simple to implement this file access mechanism.
- It uses lexicographic order to quickly access the next entry.
- It is suitable for applications that require access to all records in a file, in a specific order.
- It is less prone to data corruption as the data is written sequentially and not randomly.
- It is a more efficient method for reading large files, as it only reads the required data and does not waste time reading unnecessary data.
- It is a reliable method for backup and restore operations, as the data is stored sequentially and can be easily restored if required.

Disadvantages of Sequential Access Method :

- If the file record that needs to be accessed next is not present next to the current record, this type of file access method is slow.
- Moving a sizable chunk of the file may be necessary to insert a new record.
- It does not allow for quick access to specific records in the file. The entire file must be searched sequentially to find a specific record, which can be time-consuming.
- It is not well-suited for applications that require frequent updates or modifications to the file. Updating or inserting a record in the middle of a large file can be a slow and cumbersome process.
- Sequential access can also result in wasted storage space if records are of varying lengths. The space between records cannot be used by other records, which can result in inefficient use of storage.

2.Direct Access –

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.

Advantages of Direct Access Method :

- The files can be immediately accessed decreasing the average access time.
- In the direct access method, in order to access a block, there is no need of traversing all the blocks present before it.

3.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:

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

4.Relative Record Access –

Relative record access is a file access method used in operating systems where records are accessed relative to the current position of the file pointer. In this method, records are located based on their position relative to the current record, rather than by a specific address or key value.

Key Points of Relative Record Access:

Relative record access is a random access method that allows records to be accessed based on their position relative to the current record.

This method is efficient for accessing individual records but may not be suitable for files that require frequent updates or random access to specific records.

Relative record access requires fixed-length records and may not be flexible enough for some applications.

This method is useful for processing records in a specific order or for files that are accessed sequentially.

Advantages of Relative Record Access:

Random Access: Relative record access allows random access to records in a file. The system can access any record at a specific offset from the current position of the file pointer.

Efficient Retrieval: Since the system only needs to read the current record and any records that need to be skipped, relative record access is more efficient than sequential access for accessing individual records.

Useful for Sequential Processing: Relative record access is useful for processing records in a specific order. For example, if the records are sorted in a specific order, the system can access the next or previous record relative to the current position of the file pointer.

Disadvantages of Relative Record Access:
Fixed Record Length: Relative record access requires fixed-length records. If the records are of varying length, it may be necessary to use padding to ensure that each record is the same length.

Limited Flexibility: Relative record access is not very flexible. It is difficult to insert or delete records in the middle of a file without disrupting the relative positions of other records.

Limited Application: Relative record access is best suited for files that are accessed sequentially or with some regularity, but it may not be appropriate for files that are frequently updated or require random access to specific records.

5.Content Addressable Access-
Content-addressable access (CAA) is a file access method used in operating systems that allows records or blocks to be accessed based on their content rather than their address. In this method, a hash function is used to calculate a unique key for each record or block, and the system can access any record or block by specifying its key.

Keys in Content-Addressable Access:
Unique: Each record or block has a unique key that is generated using a hash function.

Calculated based on content: The key is calculated based on the content of the record or block, rather than its location or address.

Advantages of Content-Addressable Access:
Efficient Search: CAA is ideal for searching large databases or file systems because it allows for efficient searching based on the content of the records or blocks.

Flexibility: CAA is more flexible than other access methods because it allows for easy insertion and deletion of records or blocks.

Data Integrity: CAA ensures data integrity because each record or block has a unique key that is generated based on its content.

Disadvantages of Content-Addressable Access:
Overhead: CAA requires additional overhead because the hash function must be calculated for each record or block.

Collision: There is a possibility of collision where two records or blocks can have the same key. This can be minimized by using a good hash function, but it cannot be completely eliminated.

Limited Key Space: The key space is limited by the size of the hash function used, which can lead to collisions and other issues.

File Type

It refers to the ability of the operating system to differentiate various types of files like text files, binary, and source files. However, Operating systems like MS_DOS and UNIX has the following type of files:

Character Special File

It is a hardware file that reads or writes data character by character, like mouse, printer, and more.

Ordinary files

- These types of files stores user information.
- It may be text, executable programs, and databases.
- It allows the user to perform operations like add, delete, and modify.

Directory Files

- Directory contains files and other related information about those files. Its basically a folder to hold and organize multiple files.

Special Files

- These files are also called device files. It represents physical devices like printers, disks, networks, flash drive, etc.

Operations on file

- **Creating-** Creating a file involves two steps. First, check if there is free space. You must create a new directory file entry if you have enough space.
- **Opening**- This is a common operation performed on a file. After you create a file, you need to open it before you can perform any file processing operations. If the user wants to open the file, call a system call to instruct the operating system to open the file and pass the file name to the file
- **Writing**- To write a file, you need to know two things. One is the name of the file and the other is the information or data written to the file
- **Deleting**- To delete a file, first browse the directory of the named file, then free up the file area and delete the directory entry.
- **Truncating**- To truncate a file, delete only the contents of the file, but don't delete its file structure.
- **Close File-** After **editing** the **file, you need to close** the file **to free up any permanent** changes made and occupied **resources. Finally,** the internal **descriptor** created when the file was **opened is released.**

Directories Strcture

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

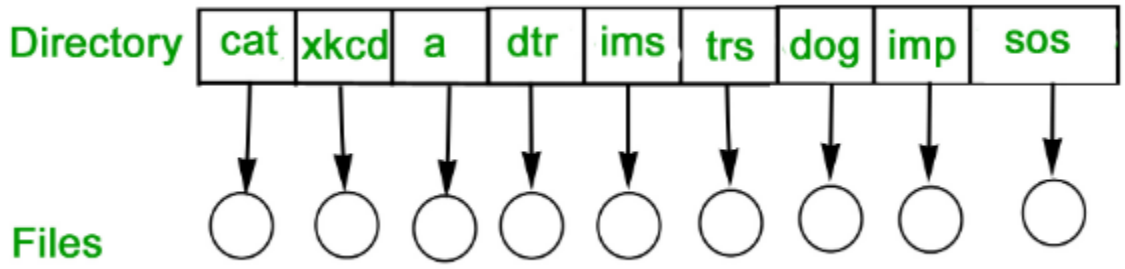
Advantages of Maintaining Directories

- **Efficiency:** A file can be located more quickly.
- **Naming:** It becomes convenient for users as two users can have same name for different files or may have different name for same file.
- **Grouping:** Logical grouping of files can be done by properties e.g. all java programs, all games etc.

Single-Level Directory

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

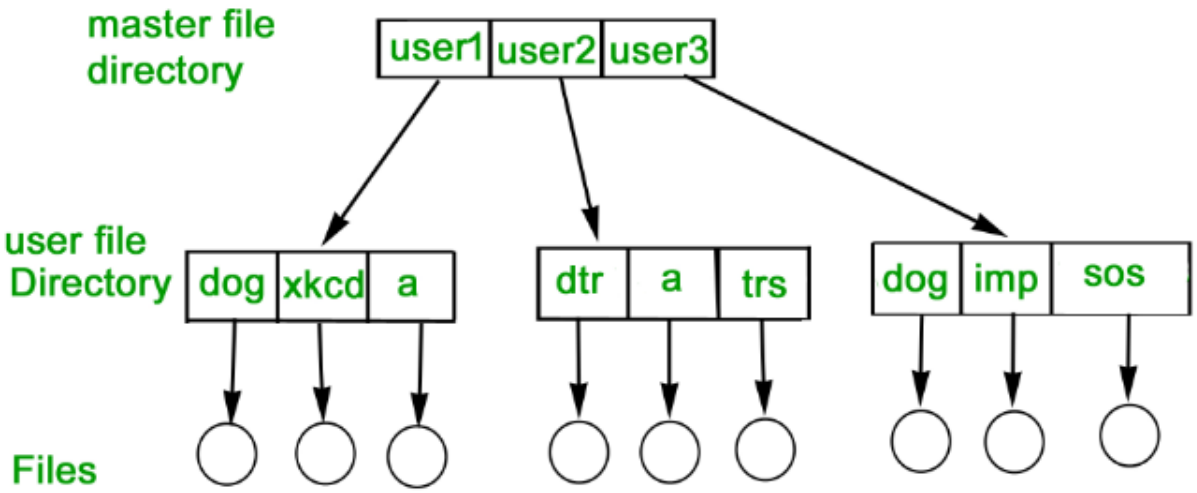
- **Naming problem:** Users cannot have the same name for two files.
- **Grouping problem:** Users cannot group files according to their needs.



Two-Level Directory

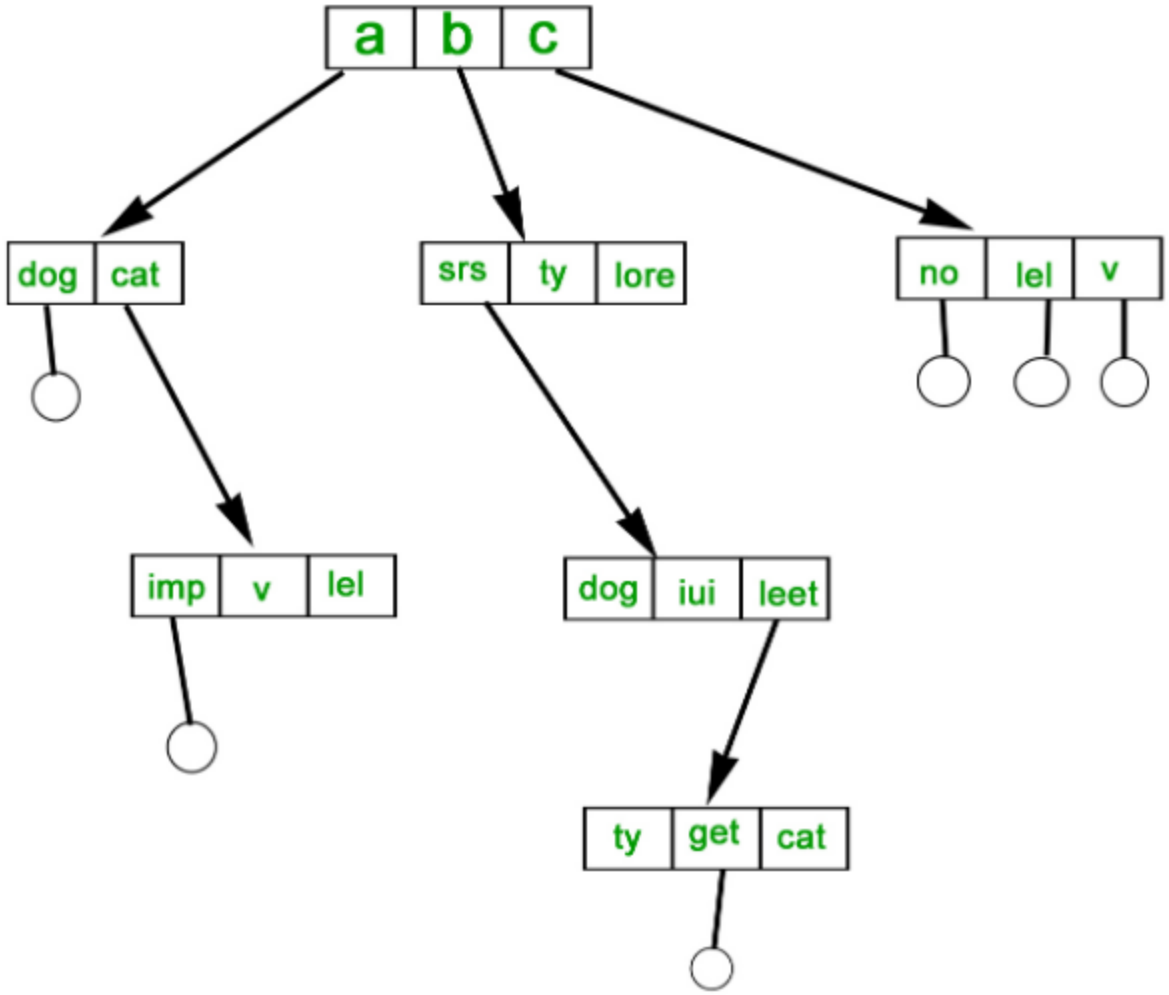
In this separate directories for each user is maintained.

- **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 Allocation Methods

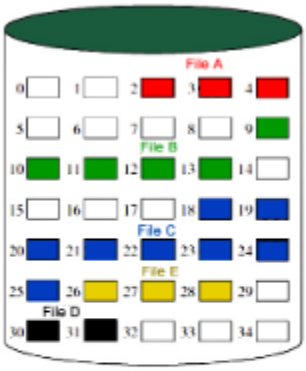
There are several types of file allocation methods. These are mentioned below.

- Continuous Allocation
- Linked Allocation(Non-contiguous allocation)
- Indexed Allocation

Continuous Allocation

A single continuous set of blocks is allocated to a file at the time of file creation. Thus, this is a pre-allocation strategy, using variable size portions. The file allocation table needs just a single entry for each file, showing the starting block and the length of the file. This method is best from the point of view of the individual sequential file. Multiple blocks can be read in at a time to improve I/O performance for sequential processing. It is also easy to retrieve a single block.

For example, if a file starts at block b, and the ith block of the file is wanted, its location on secondary storage is simply b+i-1.



File allocation table

File name	Start block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3

Disadvantages of Continuous Allocation

- External fragmentation will occur, making it difficult to find contiguous blocks of space of sufficient length. A compaction algorithm will be necessary to free up additional space on the disk.
- Also, with pre-allocation, it is necessary to declare the size of the file at the time of creation.

Linked Allocation(Non-Contiguous Allocation)

Allocation is on an individual block basis. Each block contains a pointer to the next block in the chain. Again the file table needs just a single entry for each file, showing the starting block and the length of the file. Although pre-allocation is possible, it is more common simply to allocate blocks as needed. Any free block can be added to the chain. The blocks need not be continuous. An increase in file size is always possible if a free disk block is available. There is no external

fragmentation because only one block at a time is needed but there can be internal fragmentation but it exists only in the last disk block of the file.

Disadvantage Linked Allocation(Non-contiguous allocation)

- Internal fragmentation exists in the last disk block of the file.
- There is an overhead of maintaining the pointer in every disk block.
- If the pointer of any disk block is lost, the file will be truncated.
- It supports only the sequential access of files.

Indexed Allocation

It addresses many of the problems of contiguous and chained allocation. In this case, the file allocation table contains a separate one-level index for each file: The index has one entry for each block allocated to the file. The allocation may be on the basis of fixed-size blocks or variable-sized blocks. Allocation by blocks eliminates external fragmentation, whereas allocation by variable-size blocks improves locality. This allocation technique supports both sequential and direct access to the file and thus is the most popular form of file allocation.