Operating System

By Bhavika Parmar

Unit : 2
I/O Device and File Management

- One of the important jobs of an Operating System is to manage various I/O devices including mouse, keyboards, touch pad, disk drives, display adapters, USB devices, Bit-mapped screen, LED, Analog-to-digital converter, On/off switch, network connections, audio I/O, printers etc.

- An I/O system is required to take an application I/O request and send it to the physical device, then take whatever response comes back from the device and send it to the application. I/O devices can be divided into two categories −

- **Block devices** − A block device is one with which the driver communicates by sending entire blocks of data. For example, Hard disks, USB cameras, Disk-On-Key etc.

- **Character devices** − A character device is one with which the driver communicates by sending and receiving single characters (bytes, octets). For example, serial ports, parallel ports, sounds cards etc

Device Controllers

- 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.
- The Device Controller works like an interface between a device and a device driver. I/O units (Keyboard, mouse, printer, etc.) typically consist of a mechanical component and an electronic component where electronic component is called the device controller.

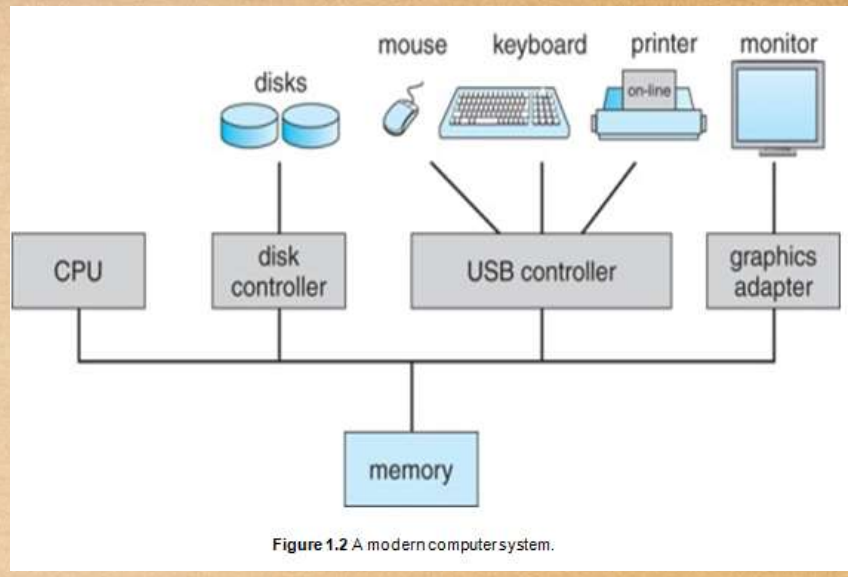
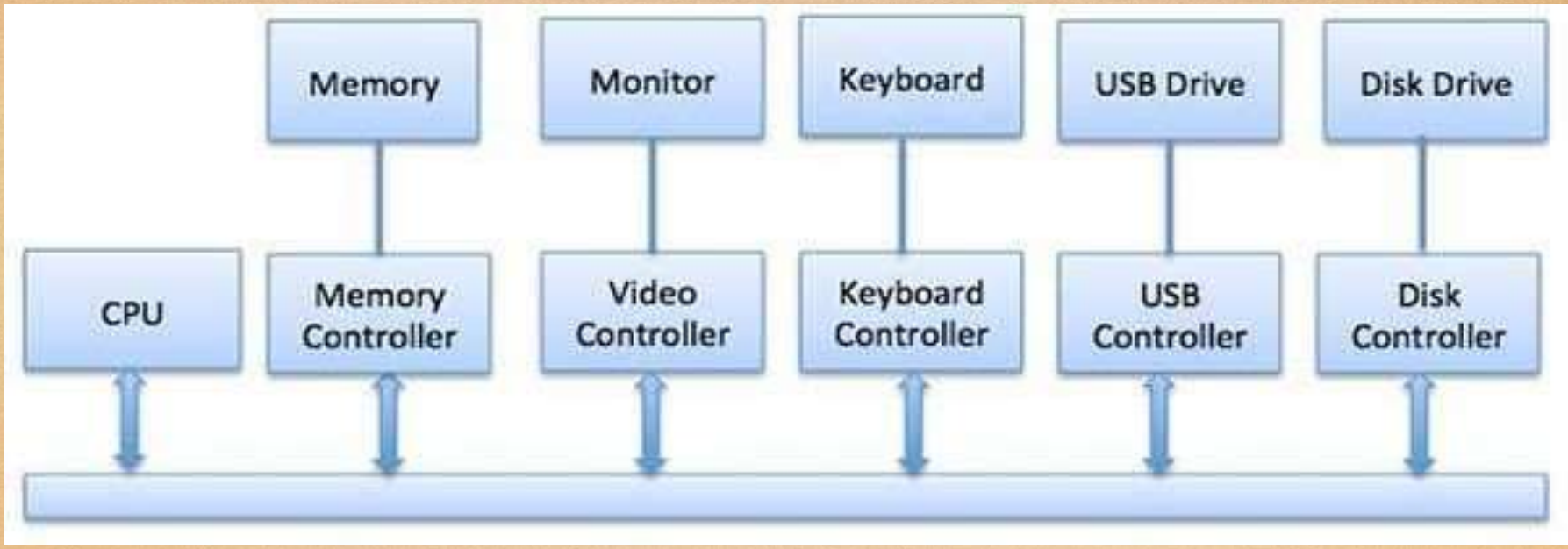


Figure 1.2 A modern computer system.

- There is always a device controller and a device driver for each device to communicate with the Operating Systems. A device controller may be able to handle multiple devices. As an interface its main task is to convert serial bit stream to block of bytes, perform error correction as necessary.

- Any device connected to the computer is connected by a plug and socket, and the socket is connected to a device controller. Following is a model for connecting the CPU, memory, controllers, and I/O devices where CPU and device controllers all use a common bus for communication.


Memory
Monitor
Keyboard
USB Drive
Disk Drive
CPU
Memory Controller
Video Controller
Keyboard Controller
USB Controller
Disk Controller

What Is a Device Driver?

- **A device driver is a software program without a user interface (UI) that manages hardware components or peripherals attached to a computer and enables them to function with the computer smoothly.**
- A device driver is a specialized software that operates a particular computer-connected device—offering a software interface to the hardware allows operating systems and other computer applications to access hardware functionalities.

Communication to I/O Devices

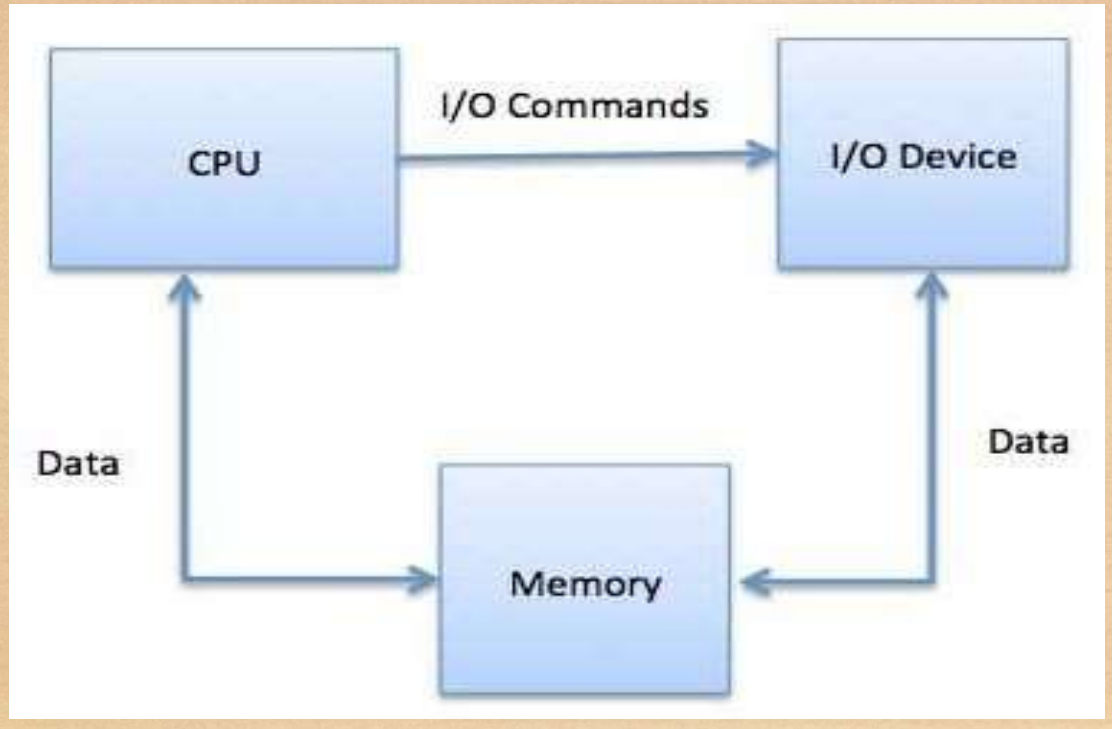
- The CPU must have a way to pass information to and from an I/O device. There are three approaches available to communicate with the CPU and Device.
- Special Instruction I/O
- Memory-mapped I/O
- Direct memory access (DMA)

Special Instruction I/O

- This uses CPU instructions that are specifically made for controlling I/O devices. These instructions typically allow data to be sent to an I/O device or read from an I/O device.

Memory-mapped I/O

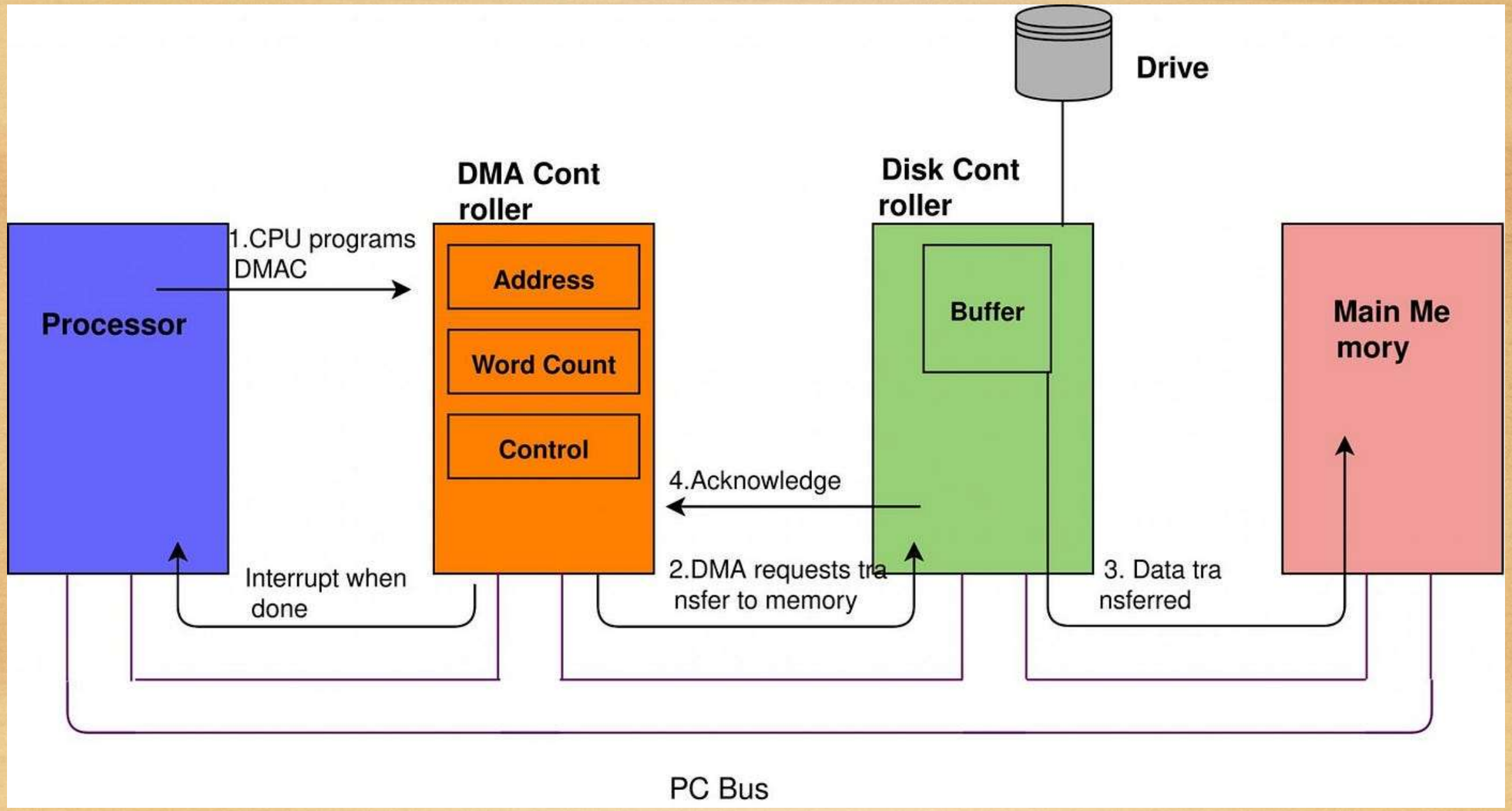
- When using memory-mapped I/O, the same address space is shared by memory and I/O devices. The device is connected directly to certain main memory locations so that I/O device can transfer block of data to/from memory without going through CPU.


CPU
I/O Commands
I/O Device
Data
Data
Memory

Direct Memory Access (DMA)

- Slow devices like keyboards will generate an interrupt to the main CPU after each byte is transferred. If a fast device such as a disk generated an interrupt for each byte, the operating system would spend most of its time handling these interrupts. So a typical computer uses direct memory access (DMA) hardware to reduce this overhead.

- Direct Memory Access (DMA) means CPU grants I/O module authority to read from or write to memory without involvement. DMA module itself controls exchange of data between main memory and the I/O device. CPU is only involved at the beginning and end of the transfer and interrupted only after entire block has been transferred.


Drive
DMA Cont
roller
Disk Cont
roller
1.CPU programs
DMAC
Processor
Address
Word Count
Control
Buffer
Main Me
mory
4.Acknowledge
Interrupt when
done
2.DMA requests tra
nsfer to memory
3. Data tra
nsferred
PC Bus

- Direct Memory Access needs a special hardware called DMA controller (DMAC) that manages the data transfers and arbitrates access to the system bus.

The operating system uses the DMA hardware as follows −

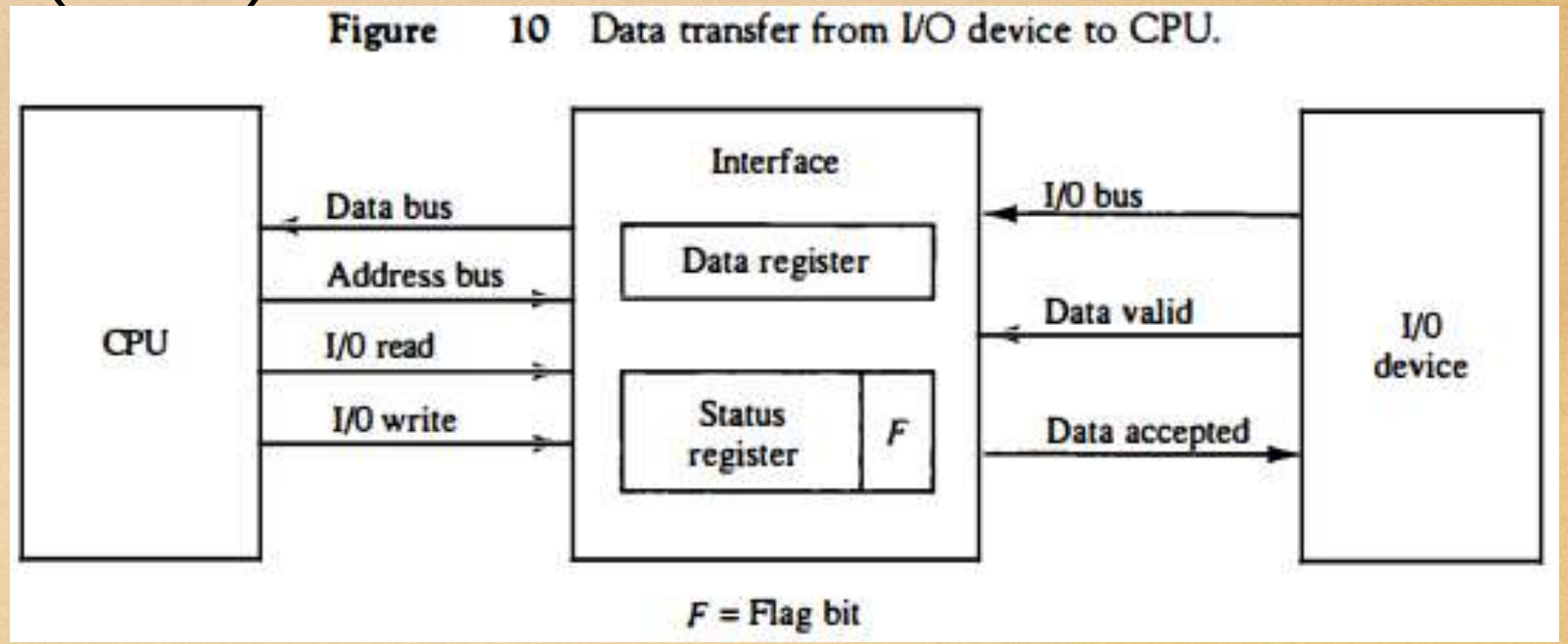
Step	Description
1	Device driver is instructed to transfer disk data to a buffer address X.
2	Device driver then instruct disk controller to transfer data to buffer.
3	Disk controller starts DMA transfer.
4	Disk controller sends each byte to DMA controller.
5	DMA controller transfers bytes to buffer, increases the memory address, decreases the counter C until C becomes zero.
6	When C becomes zero, DMA interrupts CPU to signal transfer completion.

I/O Interface (Interrupt and D M A Mode)

- The method that is used to transfer information between internal storage and external I/O devices is known as I/O interface. The CPU is interfaced using special communication links by the peripherals connected to any computer system. These communication links are used to resolve the differences between CPU and peripheral.
- **Mode of Transfer:**
- The binary information that is received from an external device is usually stored in the memory unit. The information that is transferred from the CPU to the external device is originated from the memory unit.

- Data transfer between CPU and the I/O devices may be done in different modes.
- Data transfer to and from the peripherals may be done in any of the three possible ways

1. Programmed I/O.
2. Interrupt- initiated I/O.
3. Direct memory access(DMA).



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.

- 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.

Interrupt- initiated I/O.

- 1. For this, the processor issues a **READ I/O** command to the corresponding I/O module and proceeds with some other useful tasks. It does not wait for the I/O module to get ready with the desired data.
- 2. The I/O module then processes this READ I/O command and reads the data from the addressed peripheral device. The I/O module stores the read data into its data register and issues an interrupt signal to the processor over the *control line* in the system bus.
- By sending the interrupted signal, the I/O module indicates the processor that now it is ready for transmitting the data. But, the I/O module has to wait until the processor asks for the data from the I/O module.
- When the processor requests the data from the I/O, it places the data over the data line of the system bus. Once the I/O module transfers the data to the processor it set itself ready for another I/O transfer.

- Both the methods programmed I/O and Interrupt-driven I/O require the active intervention of the processor to transfer data between memory and the I/O module, and any data transfer must transverse a path through the processor. Thus both these forms of I/O suffer from two inherent drawbacks.
  - 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.

Direct memory access(DMA).

- 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.

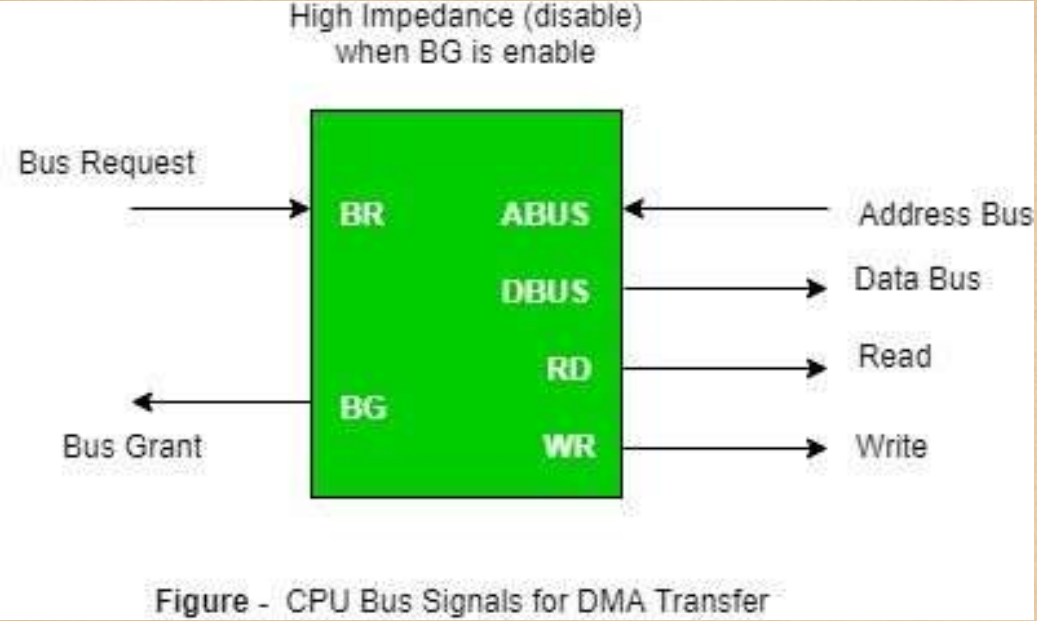


Figure - CPU Bus Signals for DMA Transfer

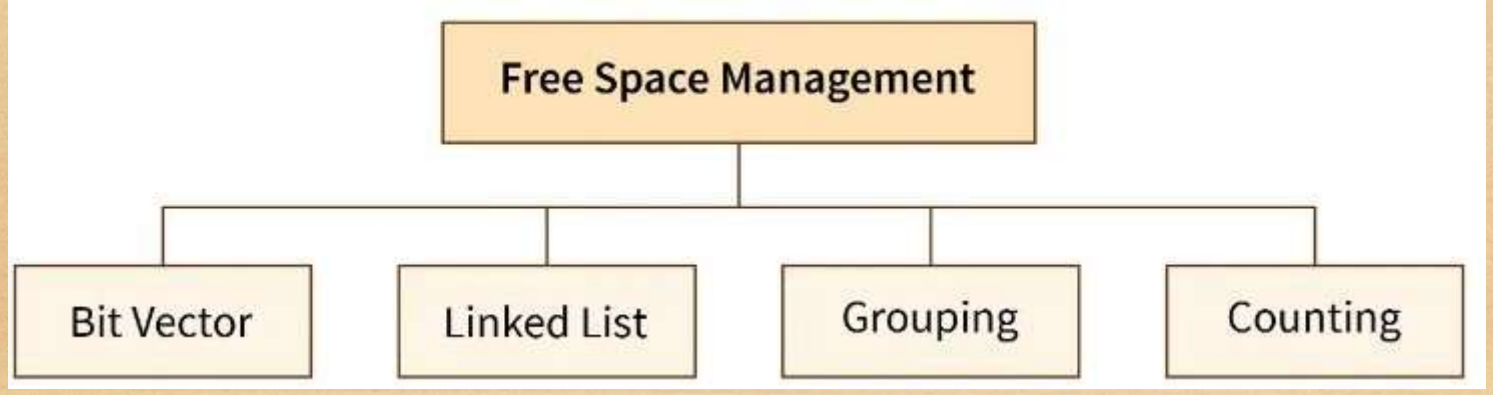
- **Bus Request :** It is used by the DMA controller to request the CPU to relinquish the control of the buses.

- **Bus Grant :** It is activated by the CPU to Inform the external DMA controller that the buses are in high impedance state and the requesting DMA can take control of the buses. Once the DMA has taken the control of the buses it transfers the data. This transfer can take place in many ways.

- Offset - an offset within an array or other data structure object is **an integer indicating the distance (displacement) between the beginning of the object and a given element or point, presumably within the same object**.

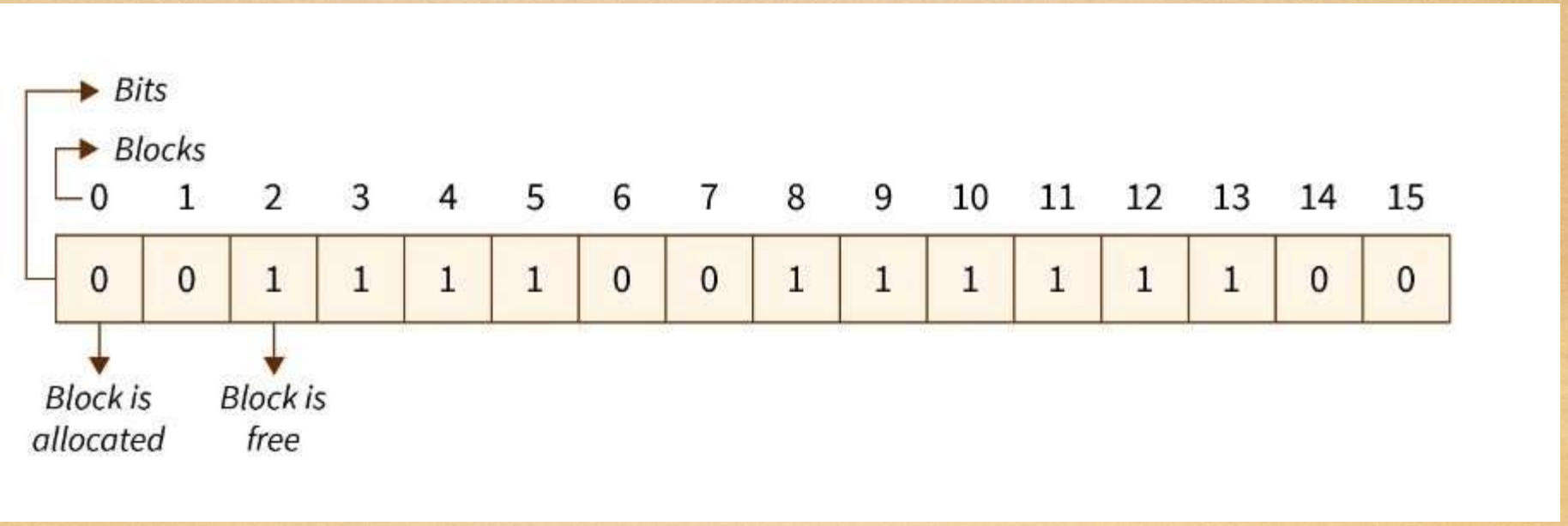
Disk space Management

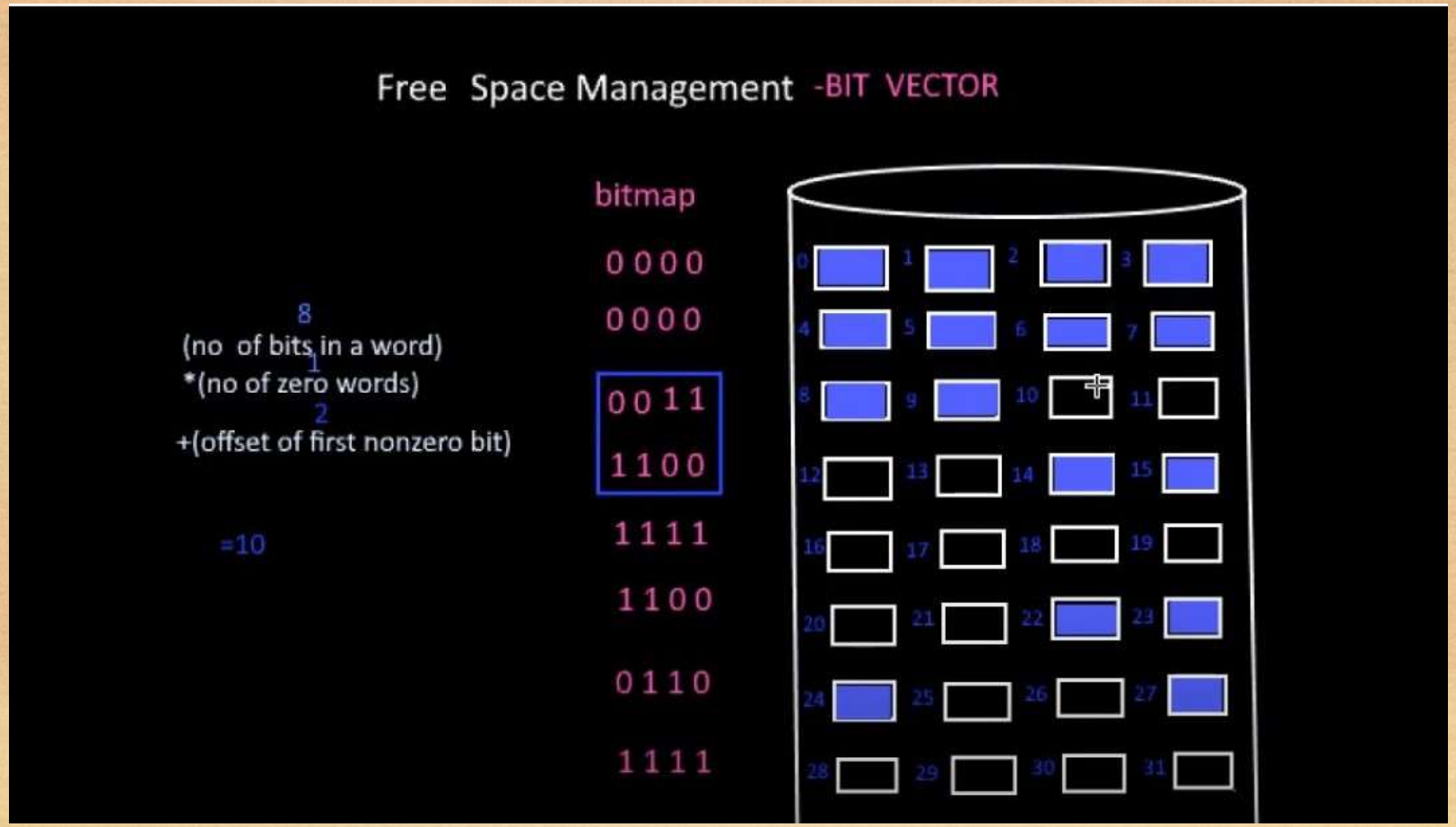
- The system keeps tracks of the free disk blocks for allocating space to files when they are created. Also, to reuse the space released from deleting the files, free space management becomes crucial. The system maintains a free space list which keeps track of the disk blocks that are not allocated to some file or directory. The free space list can be implemented mainly as:



1. Bitmap or Bit Vector :

- A bit vector is the most frequently used method to implement the free space list. A bit vector is also known as a **"Bit map"**. It is a series or collection of bits in which each bit represents a disk block. The values taken by the bits are either 1 or 0. If the block bit is 1, it means the block is empty and if the block bit is 0, it means the block is not free. It is allocated to some files. Since all the blocks are empty initially so, each bit in the bit vector represents 0.




Free Space Management -BIT VECTOR
bitmap
0000
0000
0011
1100
1111
1100
0110
1111
8
(no of bits in a word)
1
*(no of zero words)
2
+(offset of first nonzero bit)
=10
0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15
16 17 18 19
20 21 22 23
24 25 26 27
28 29 30 31

- **"Free block number"** can be defined as that block which does not contain any value.*i.e.*, they are free blocks. The formula to find a free block number is :
- [Block number = (number of bits per words)*(number of 0-value word) + Offset of first 1 bit].
- We will consider the first 8 bits group (00111100) to constitute a non-zero word since all bits are not 0 here. **"Non-zero word"** is that word that contains the bit value '1' (block that is not free). Here, the first non-zero word is the third block of the group. So, the offset will be ‘3’.
- EX. (00001110)

- **Advantages of Bit vector method :**

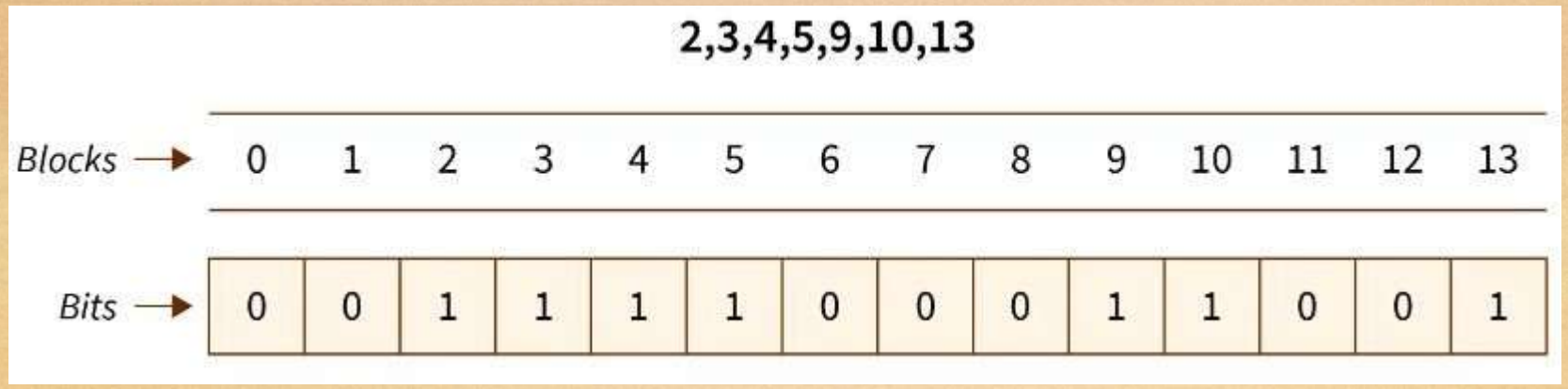
1. Simple and easy to understand.
2. Consumes less memory.
3. It is efficient to find free space.

- **Disadvantages of Bit vector method :**

4. The operating system goes through all the blocks until it finds a free block. (block whose bit is 0).
5. It is not efficient when the disk size is large.

2. Linked List :

- A **linked list** is another approach for free space management in an operating system. In it, all the free blocks inside a disk are linked together in a **"linked list"**. These free blocks on the disk are linked together by a pointer. These pointers of the free block contain the address of the next free block and the last pointer of the list points to null which indicates the end of the linked list. This technique is not enough to traverse the list because we have to read each disk block one by one which requires I/O time.

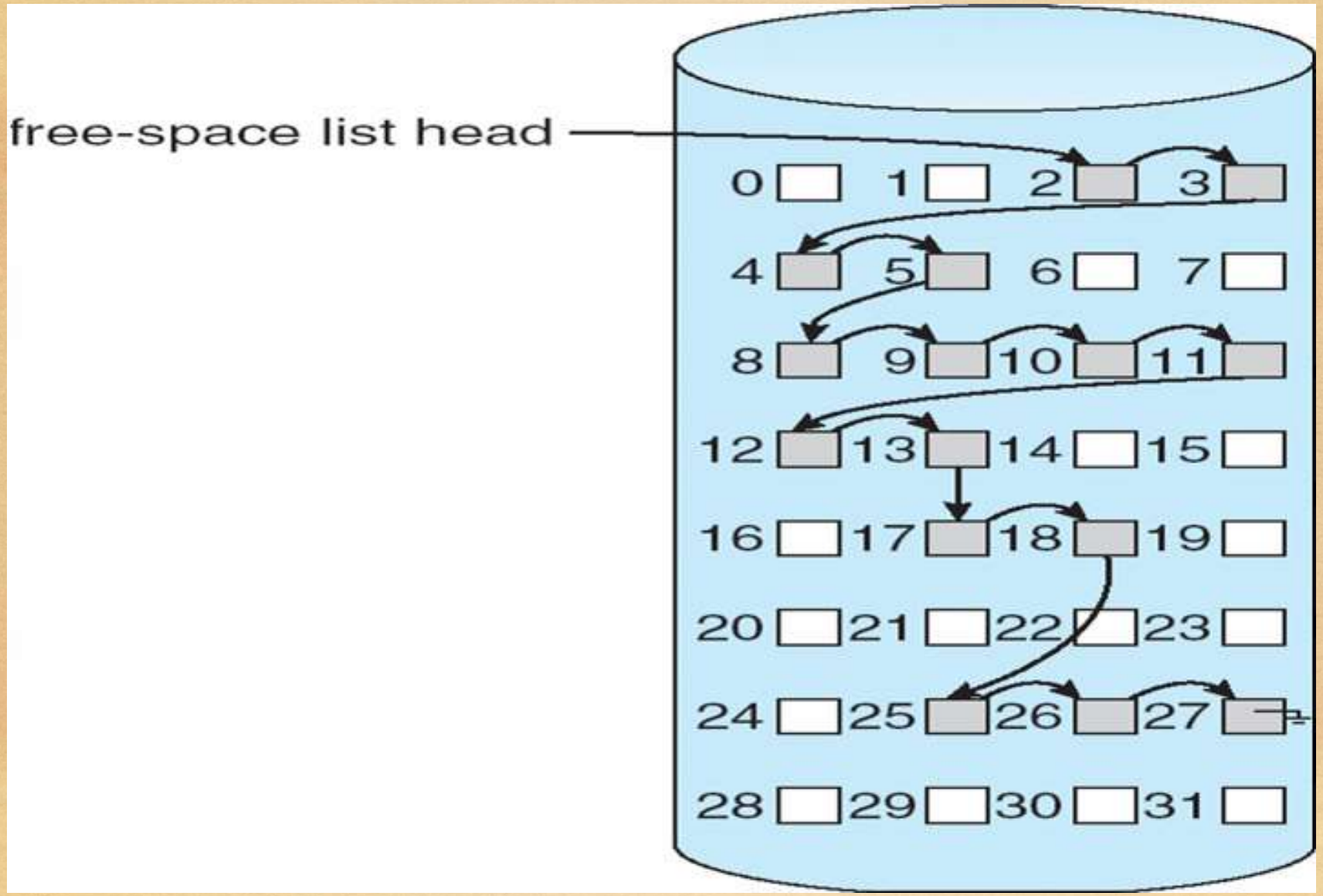


- In the above example, block 2 is the first free block and after that block 3 is the next free block, and then block 4 is free. Block 2 contains a pointer that indicates block 3 and blocks 3 contains a pointer that indicates the next free block 4. This continues until the pointer reached the last free block.
- **Advantage of the Linked list :**

1. In this method, available space is used efficiently.
2. As there is no size limit on a linked list, a new free space can be added easily.

- **Disadvantages :**

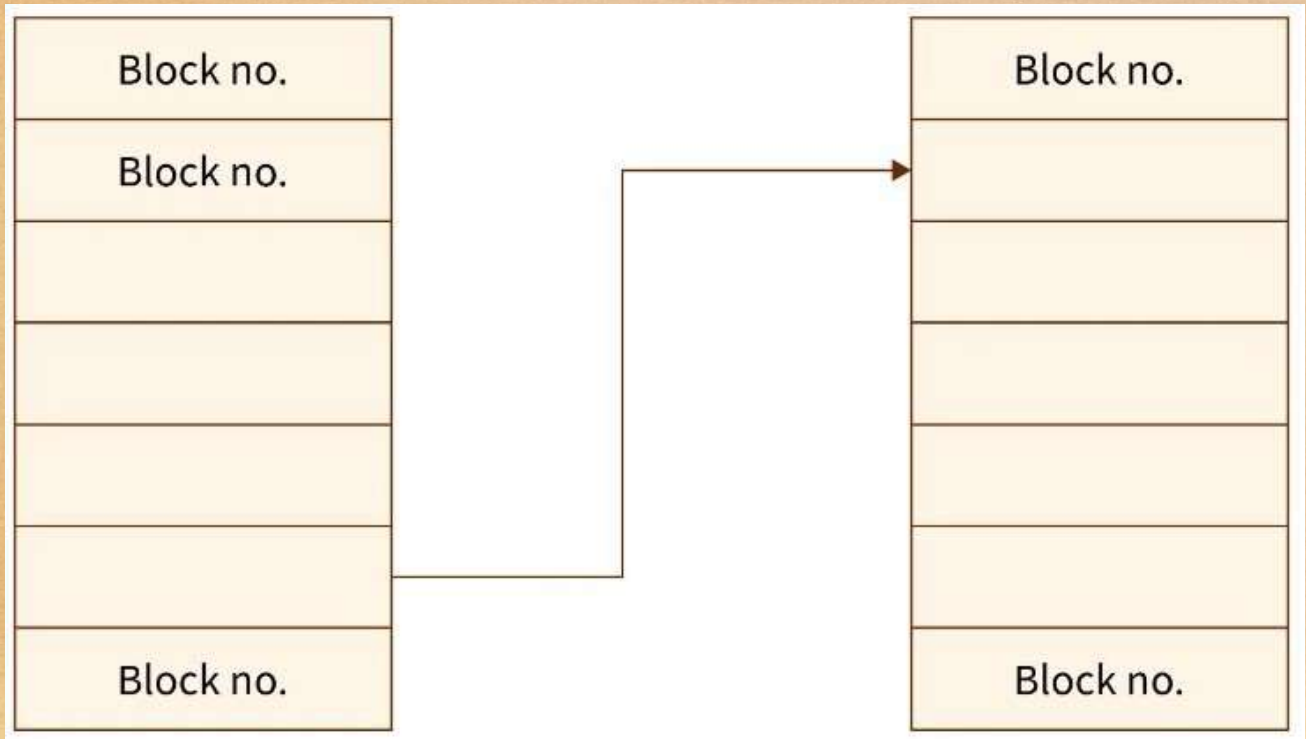
3. In this method, the overhead of maintaining the pointer appears.
4. the Linked list is not efficient when we need to reach every block of memory.


free-space list head
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31

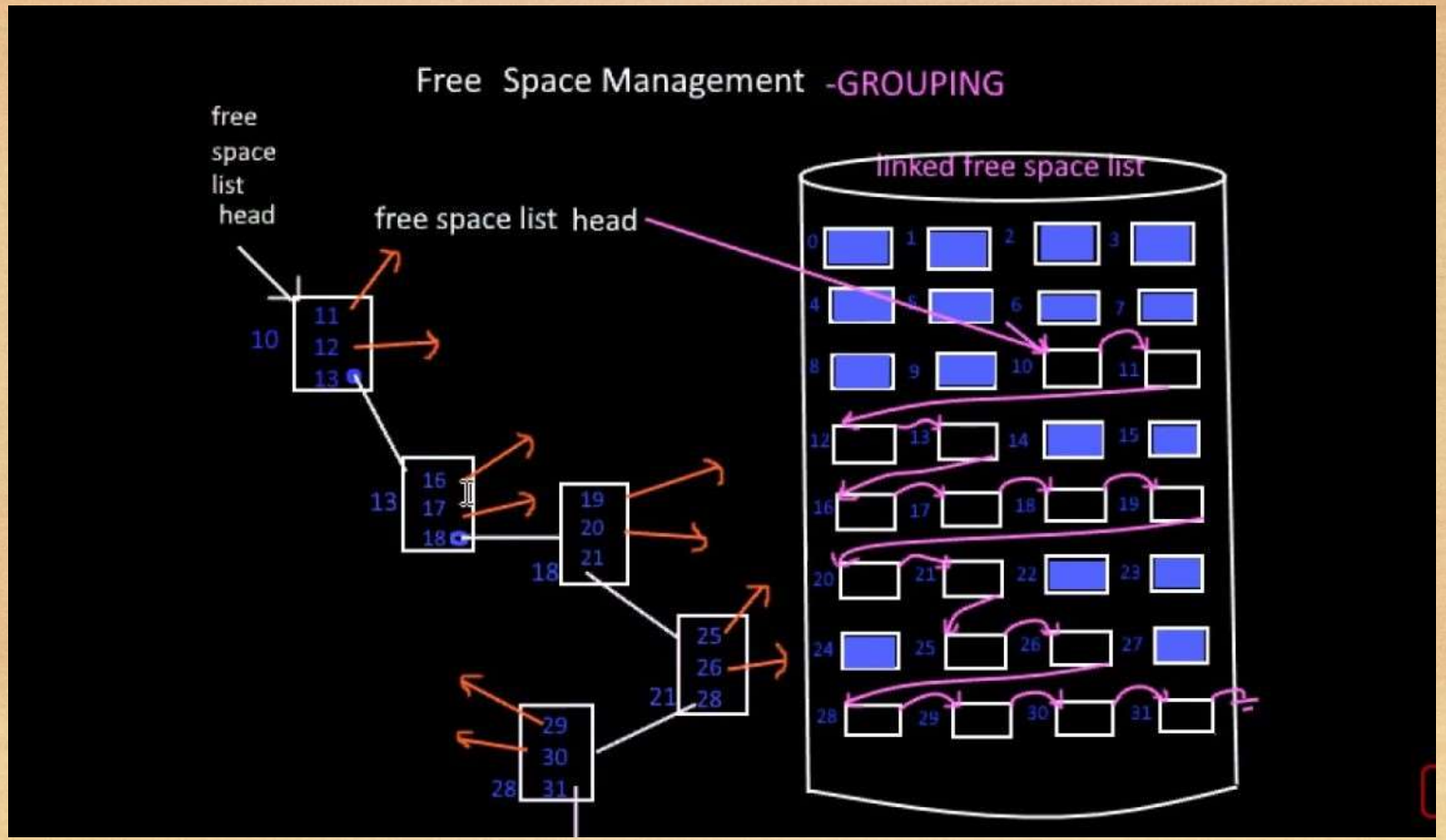
Grouping :

- The grouping technique is also called the **"modification of a linked list technique"**. In this method, first, the free block of memory contains the addresses of the n-free blocks. And the last free block of these n free blocks contains the addresses of the next n free block of memory and this keeps going on. This technique separates the empty and occupied blocks of space of memory.

When the "grouping technique" is applied, block 3 will store the addresses of blocks 4, 5, and 6 because 'block 3' is the first free block. In the same way, block 6 will store the addresses of blocks 9, 10, and 11 because block 6 is the first occupied block.



Block 3 -> 4, 5, 6
Block 6 -> 9, 10, 11
Block 11 -> 12, 13, 14


Free Space Management -GROUPING
free space list head
free space list head
linked free space list
10
11
12
13
13
16
17
18
18
19
20
21
21
25
26
28
28
29
30
31

- **Advantage of the Grouping method :**

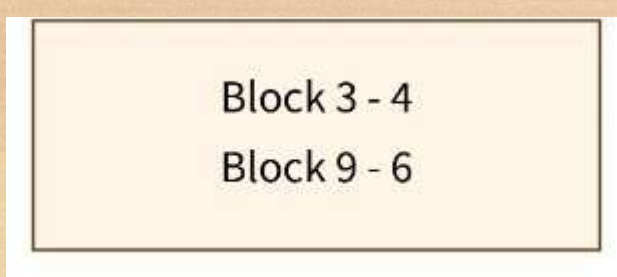
1. By using this method, we can easily find addresses of a large number of free blocks easily and quickly.

- **Disadvantage :**

1. We need to change the entire list if one block gets occupied.

Counting :

- In memory space, several files are created and deleted at the same time. For which memory blocks are allocated and de-allocated for the files. Creation of files occupy free blocks and deletion of file frees blocks. When there is an entry in the free space, it consists of two parameters- **"address of first free disk block (a pointer)"** and **" a number 'n' "**.
- **Example :**
- Let us take an example where a disk has 16 blocks in which some blocks are empty and some blocks are occupied as given below :



- When the **"counting technique"** is applied, the block number 3 will represent block number 4 because block 3 is the first free block. Then, the block stores the number of free blocks .*i.e*. - there are 4 free blocks together. In the same way, the first occupied block number 9 will represent block number 10 and keeps the number of rest occupied blocks i.e.- there are 6 occupied blocks as shown in the above figure.
- **Advantages :**

1. In this method, a bunch of free blocks take place fastly.
2. The list is smaller in size.

- **Disadvantage :**

1. In the counting method, the first free block stores the rest free blocks, so it requires more space.

Allocation and Disk Arm Scheduling Methods

- As we know, a process needs two type of time, CPU time and IO time. For I/O, it requests the Operating system to access the disk. However, the operating system must be fair enough to satisfy each request and at the same time, operating system must maintain the efficiency and speed of process execution. The technique that operating system uses to determine the request which is to be satisfied next is called disk scheduling.
- 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

- '**Disk Scheduling Algorithm**' is an algorithm that keeps and manages input and output requests arriving for the disk in a system. As we know, for executing any process memory is required. And when it comes to accessing things from a hard disk, the process becomes very slow as a hard disk is the slowest part of our computer. There are various methods by which the process can be scheduled and can be done efficiently.

What is Disk Scheduling Algorithms in OS?

- "**Disk Scheduling Algorithms**" in an operating system can be referred to as a manager of a grocery store that manages all the incoming and outgoing requests for goods of that store. He keeps a record of what is available in-store, what we need further, and manages the timetable of transaction of goods.
- The '**Disk Structure in OS**' is made in such a way that there are many layers of storage blocks on the disk. And when we need to access these data from disk, we initialize a 'request' to the system to give us the required data. These requests are done on a large scale. So, there is a large number of input and output requests coming to the disk. The operating system manages the timetable of all these requests in a proper algorithm. This is called as "Disk Scheduling Algorithm in OS".

- This algorithm helps OS to maintain an efficient manner in which input-output requests can be managed to execute a process. It manages a proper order to deal with sequential requests to the disk and provide the required data. Since multiple requests are approaching the disk, this algorithm also manages the upcoming requests of the future and does a lining up of requests.

Why is Disk Scheduling needed?

- In our system, **multiple requests** are coming to the disk simultaneously which will make a queue of requests. This queue of requests will result in an increased waiting time of requests. The requests wait until the underprocessing request executes. To overcome this queuing and manage the timing of these requests, 'Disk Scheduling' is important in our Operating System.

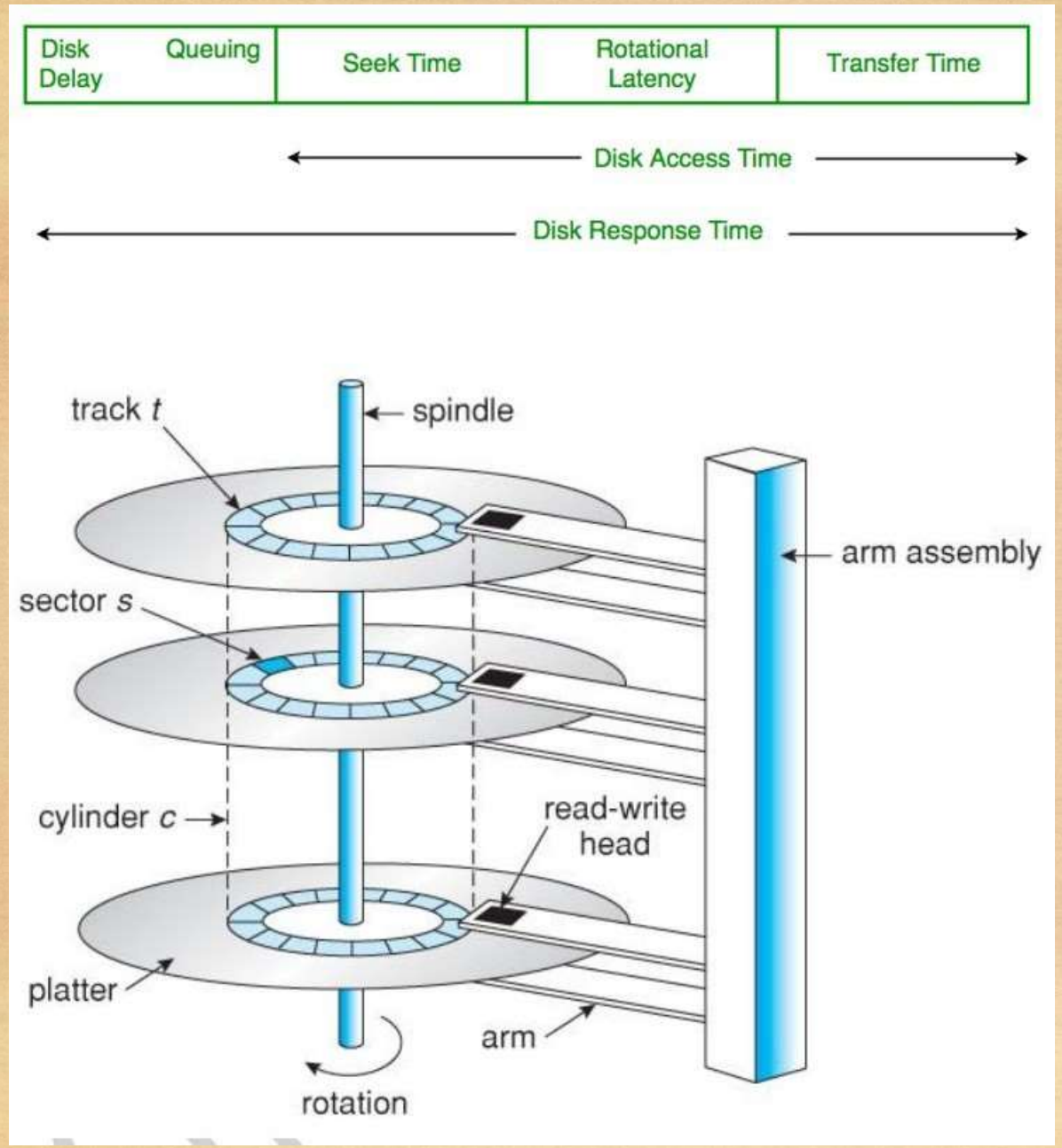
Important terms related to disk scheduling

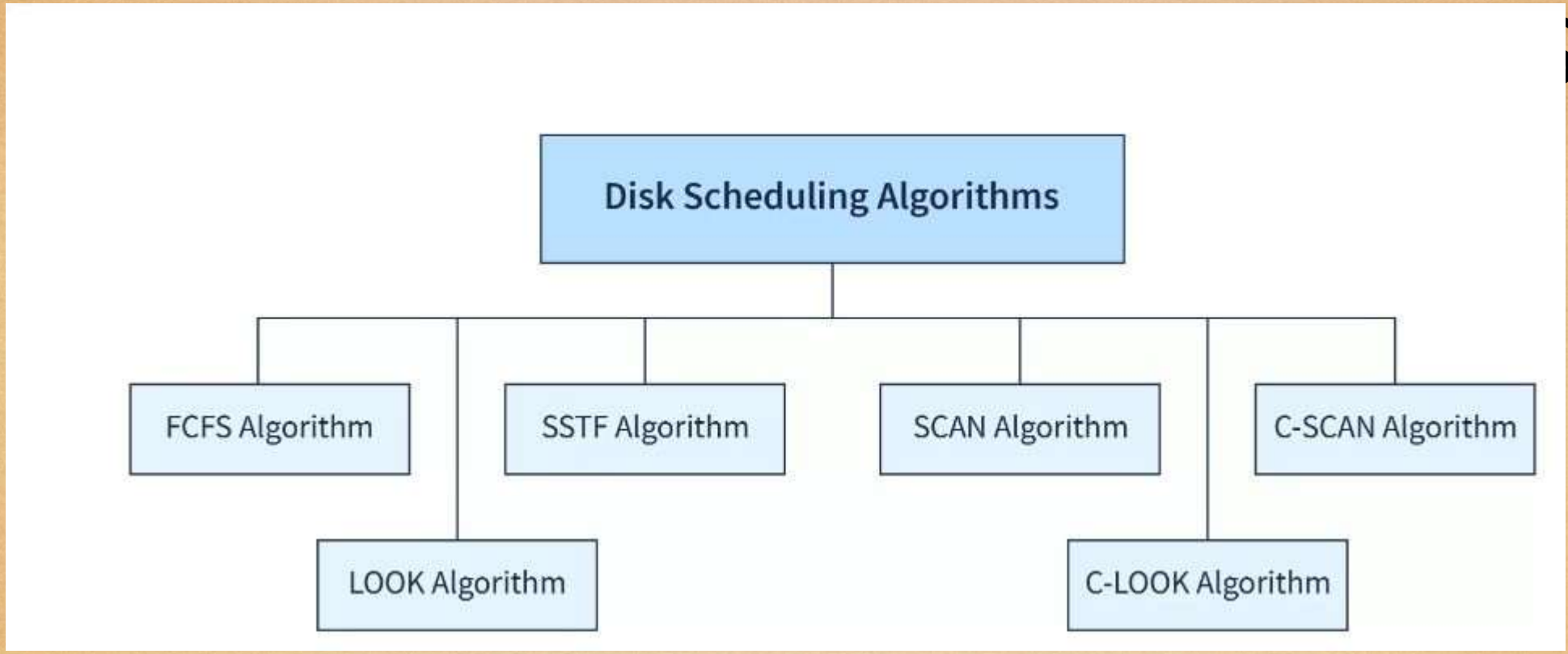
1. **Seek Time**: To access these data according to the request, the disk arm moves and find the required block. The time taken by the arm in doing this search is known as "Seek Time".
2. **Rotational Latency**: The required data block needs to move at a particular position from where the read/write head can fetch the data. So, the time taken in this movement is known as "Rotational Latency". This rotational time should be as less as possible so, the algorithm that will take less time to rotate will be considered a better algorithm.
3. **Transfer Time**: When a request is made from the user side, it takes some time to fetch these data and provide them as output. This taken time is known as "Transfer Time".

4. **Disk Access Time**: It is defined as the total time taken by all the above processes. Disk access time =(seek time +rotational latency time +transfer time)

5. **Disk Response Time**: The disk processes one request at a single time. So, the other requests wait in a queue to finish the ongoing process of request. The average of this waiting time is called "Disk Response Time".

6. **Starvation**: Starvation is defined as the situation in which a low-priority job keeps waiting for a long time to be executed. The system keeps sending high-priority jobs to the disk scheduler to execute first.

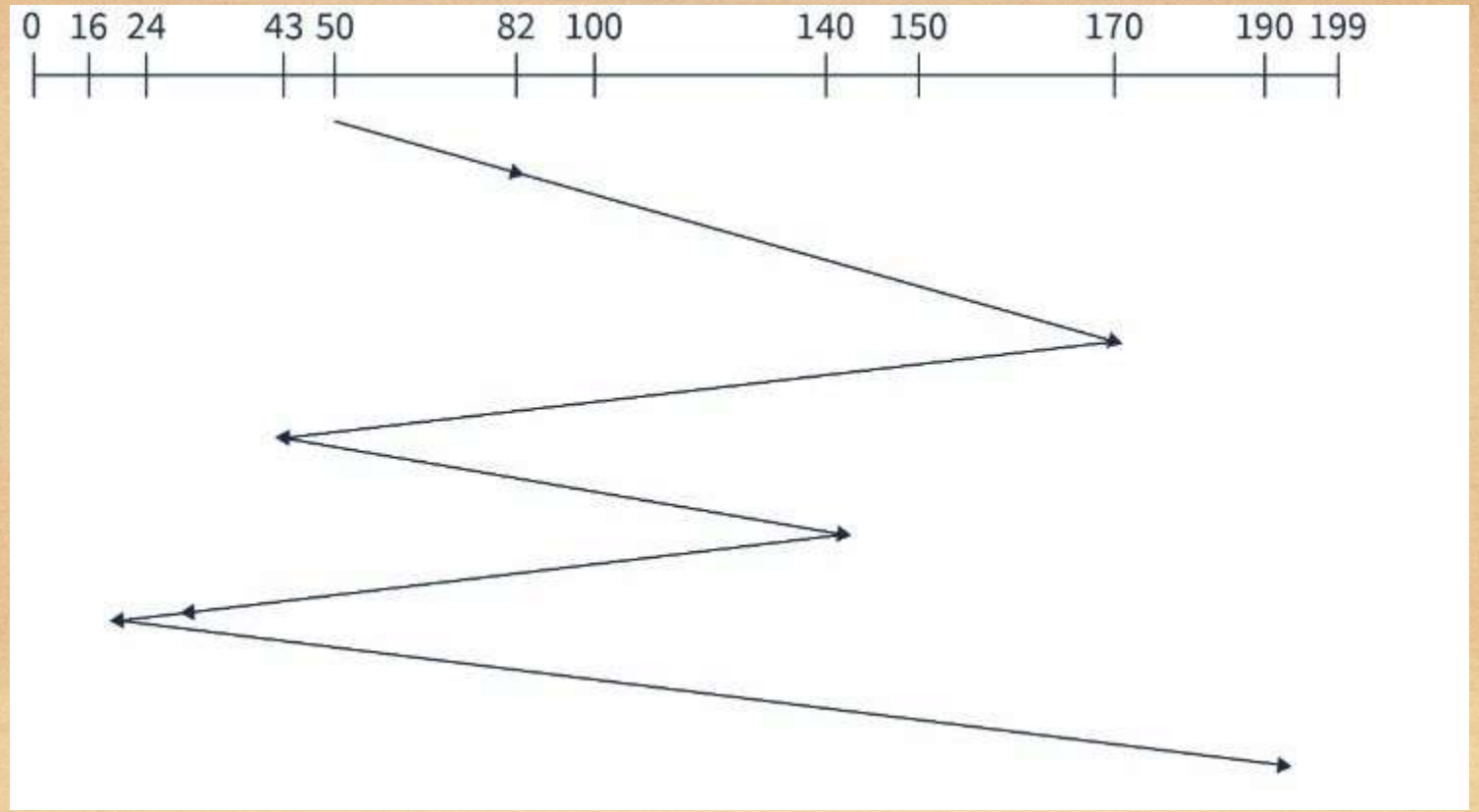

Disk Delay
Queuing
Seek Time
Rotational Latency
Transfer Time
Disk Access Time
Disk Response Time
track t
spindle
sector s
arm assembly
cylinder c
read-write head
platter
arm
rotation


Disk Scheduling Algorithms
FCFS Algorithm
SSTF Algorithm
SCAN Algorithm
C-SCAN Algorithm
LOOK Algorithm
C-LOOK Algorithm

1. FCFS disk scheduling algorithm-

- It stands for '**first-come-first-serve**'. As the name suggests, the request which comes first will be processed first and so on. The requests coming to the disk are arranged in a proper sequence as they arrive. Since every request is processed in this algorithm, so there is no chance of 'starvation'.

Example: Suppose a disk having 200 tracks (0-199). The request sequence(82,170,43,140,24,16,190) of disk are shown as in the given figure and the head start is at request 50.



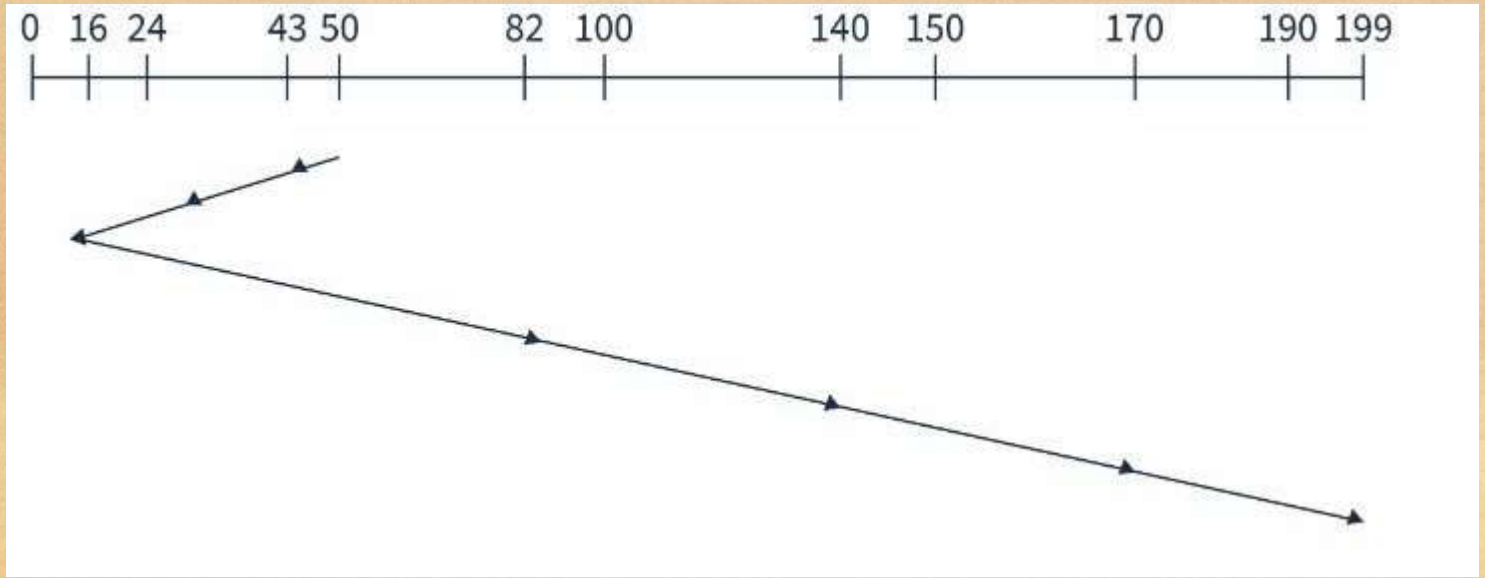
- **Explanation**: In the above image, we can see the head starts at position 50 and moves to request 82. After serving them the disk arm moves towards the second request that is 170 and then to the request 43 and so on. In this algorithm,, the disk arm will serve the requests in arriving order. In this way, all the requests are served in arriving order until the process executes.
- "Seek time" will be calculated by adding the head movement differences of all the requests:
- **Seek time**= "(82-50) +(170-82) +(170-43) +(140-43) +(140-24) + (24-16) +(190-16) =642

- **Advantages**:
  - Implementation is easy.
  - No chance of starvation.
- **Disadvantages**:
  - 'Seek time' increases.
  - Not so efficient.

2. SSTF disk scheduling algorithm-

- It stands for '**Shortest seek time first**'. As the name suggests, it searches for the request having the least 'seek time' and executes them first. This algorithm has less 'seek time' as compared to FCFS Algorithm.

- **Example**: : Suppose a disk having 200 tracks (0-199). The request sequence(82,170,43,140,24,16,190) are shown in the given figure and the head position is at 50.



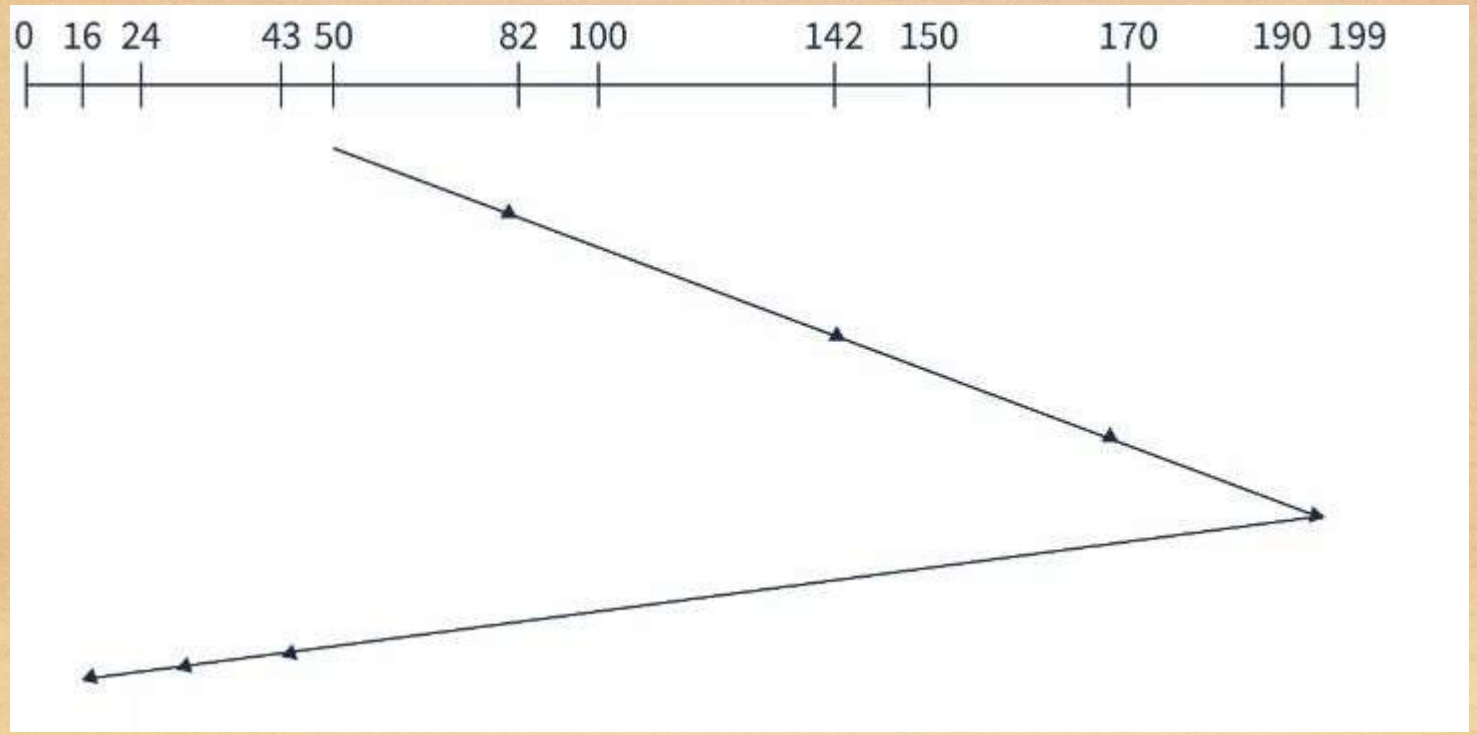
- The disk arm searches for the request which will have the least difference in head movement. So, the least difference is (50-43). Here the difference is not about the shortest value but it is about the shortest time the head will take to reach the nearest next request. So, after 43, the head will be nearest to 24, and from here the head will be nearest to the request 16, After 16, the nearest request is 82, so the disk arm will move to serve request 82 and so on.
- Hence, Calculation of Seek Time =(50-43) +(43-24) +(24-16) + (82-16) +(140-82) +(170-140) +(190-170) =208

- **Advantages**:
  - In this algorithm, disk response time is less.
  - More efficient than FCFS.
- **Disadvantages**:
  - Less speed of algorithm execution.
  - Starvation can be seen.

3. SCAN disk scheduling algorithm:

- In this algorithm, the head starts to scan all the requests in a direction and reaches the end of the disk. After that, it reverses its direction and starts to scan again the requests in its path and serves them. Due to this feature, this algorithm is also known as the "Elevator Algorithm".

- **Example**: Suppose a disk having 200 tracks (0-199). The request sequence(82,170,43,140,24,16,190) are shown in the given figure and the head position is at 50. The 'disk arm' will first move to the larger values.

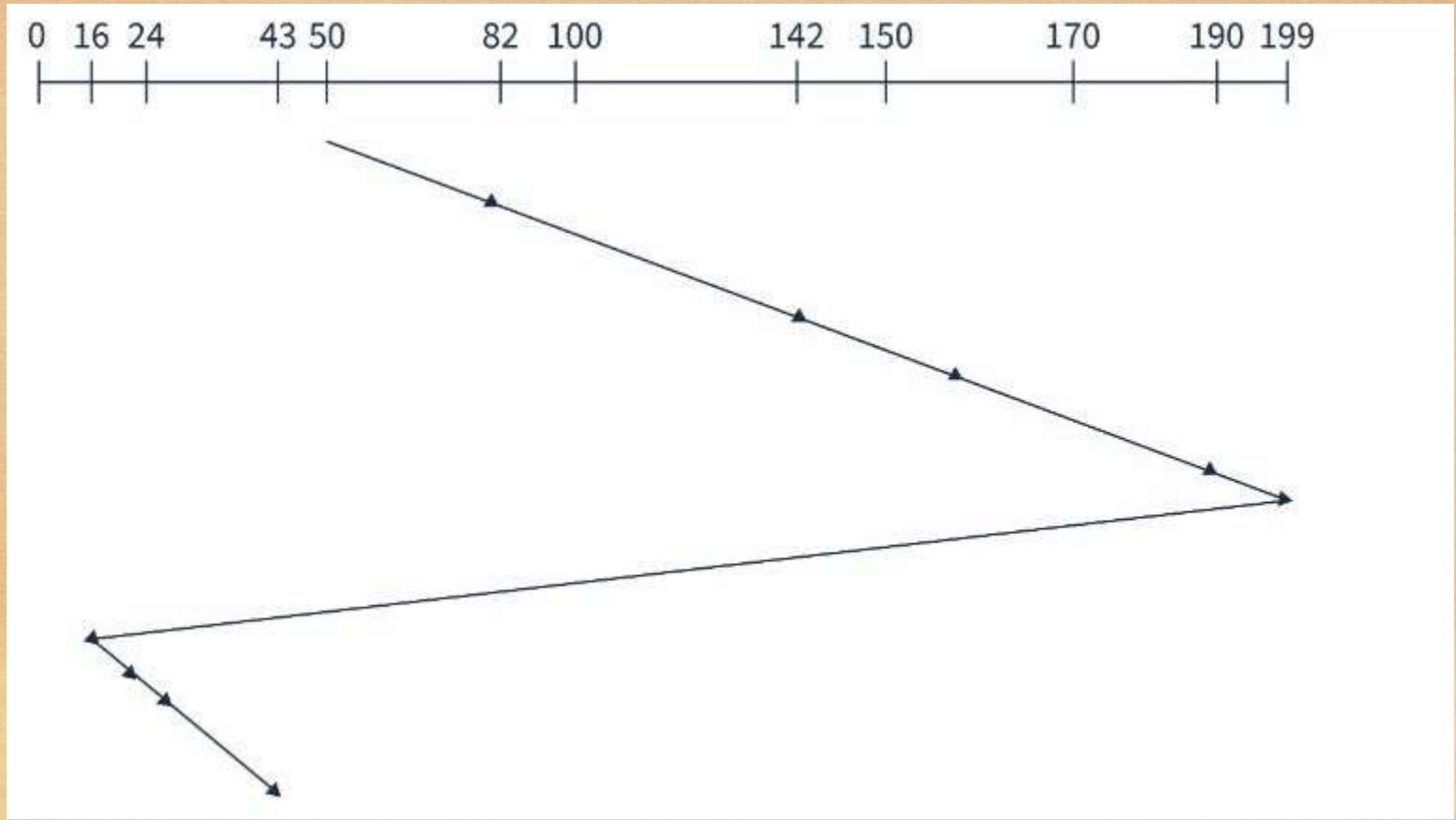


- In the above image, we can see that the disk arm starts from position 50 and goes in a single direction until it reaches the end of the disk i.e.- request position 199. After that, it reverses and starts servicing in the opposite direction until reached the other end of the disk. This process keeps going on until the process is executed. Hence, Calculation of 'Seek Time' will be like: (199-50) +(199-16) =332
- **Advantages**:
  - Implementation is easy.
  - Requests do not have to wait in a queue.
- **Disadvantage**:
  - The head keeps going on to the end even if there are no requests in that direction.

4. C-SCAN disk scheduling algorithm:

- It stands for "Circular-Scan". This algorithm is almost the same as the Scan disk algorithm but one thing that makes it different is that 'after reaching the one end and reversing the head direction, it starts to come back. The disk arm moves toward the end of the disk and serves the requests coming into its path. After reaching the end of the disk it reverses its direction and again starts to move to the other end of the disk but while going back it does not serve any requests.

- **Example**: Suppose a disk having 200 tracks (0-199). The request sequence(82,170,43,140,24,16,190) are shown in the given figure and the head position is at 50.

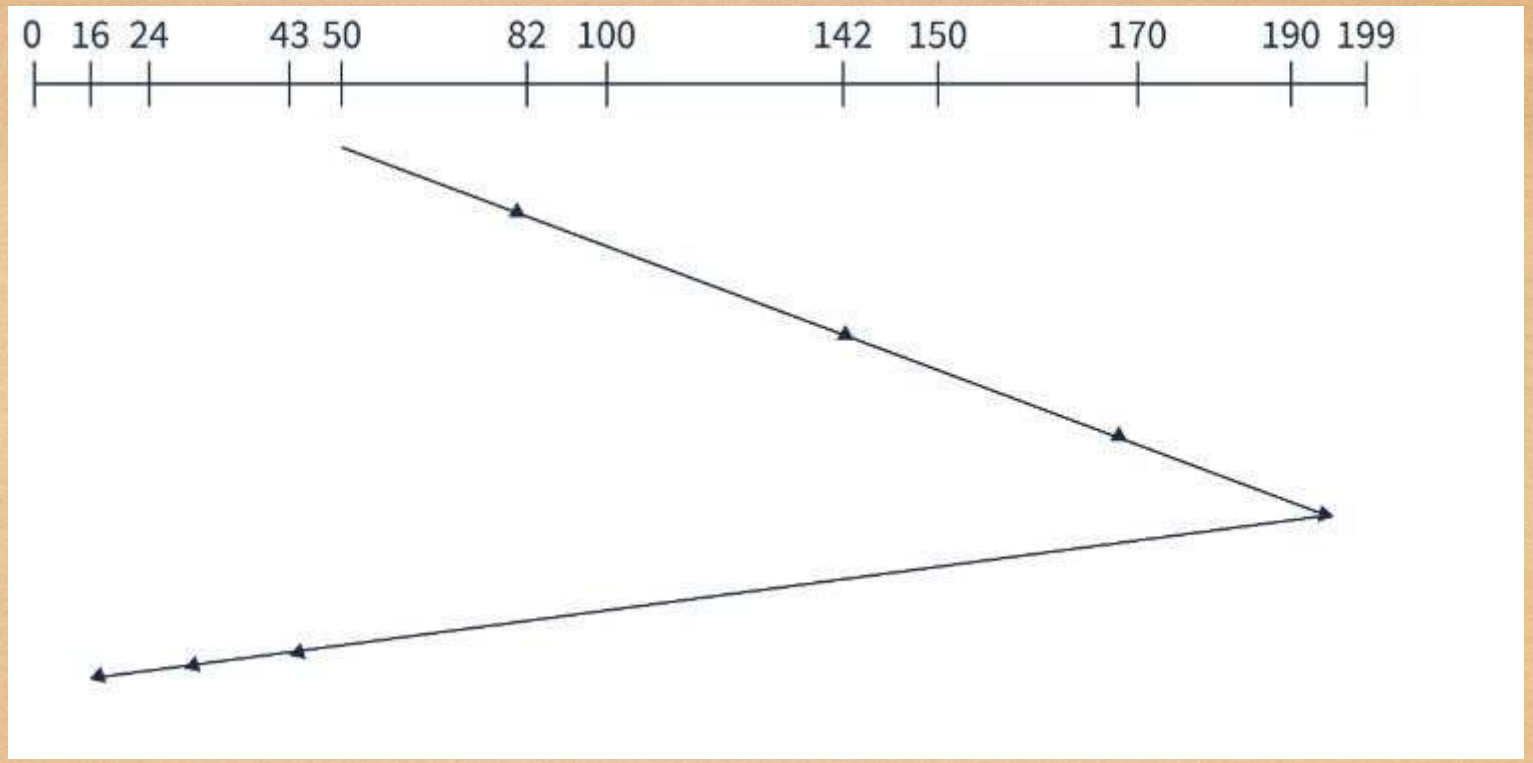


- In the above figure, the disk arm starts from position 50 and reached the end(199), and serves all the requests in the path. Then it reverses the direction and moves to the other end of the disk i.e.- 0 without serving any task in the path. After reaching 0, it will again go move towards the largest remaining value which is 43. So, the head will start from 0 and moves to request 43 serving all the requests coming in the path. And this process keeps going.
- Hence, Seek Time will be =(199-50) +(199-0) +(43-0) =391
- **Advantages**:
  - The waiting time is uniformly distributed among the requests.
  - Response time is good in it.
- **Disadvantages**:
  - The time taken by the disk arm to locate a spot is increased here.
  - The head keeps going to the end of the disk.

5. LOOK the disk scheduling algorithm:

- In this algorithm, the disk arm moves to the 'last request' present and services them. After reaching the last requests, it reverses its direction and again comes back to the starting point. It does not go to the end of the disk, in spite, it goes to the end of requests.

- **Example** a disk having 200 tracks (0-199). The request sequence(82,170,43,140,24,16,190) are shown in the given figure and the head position is at 50.

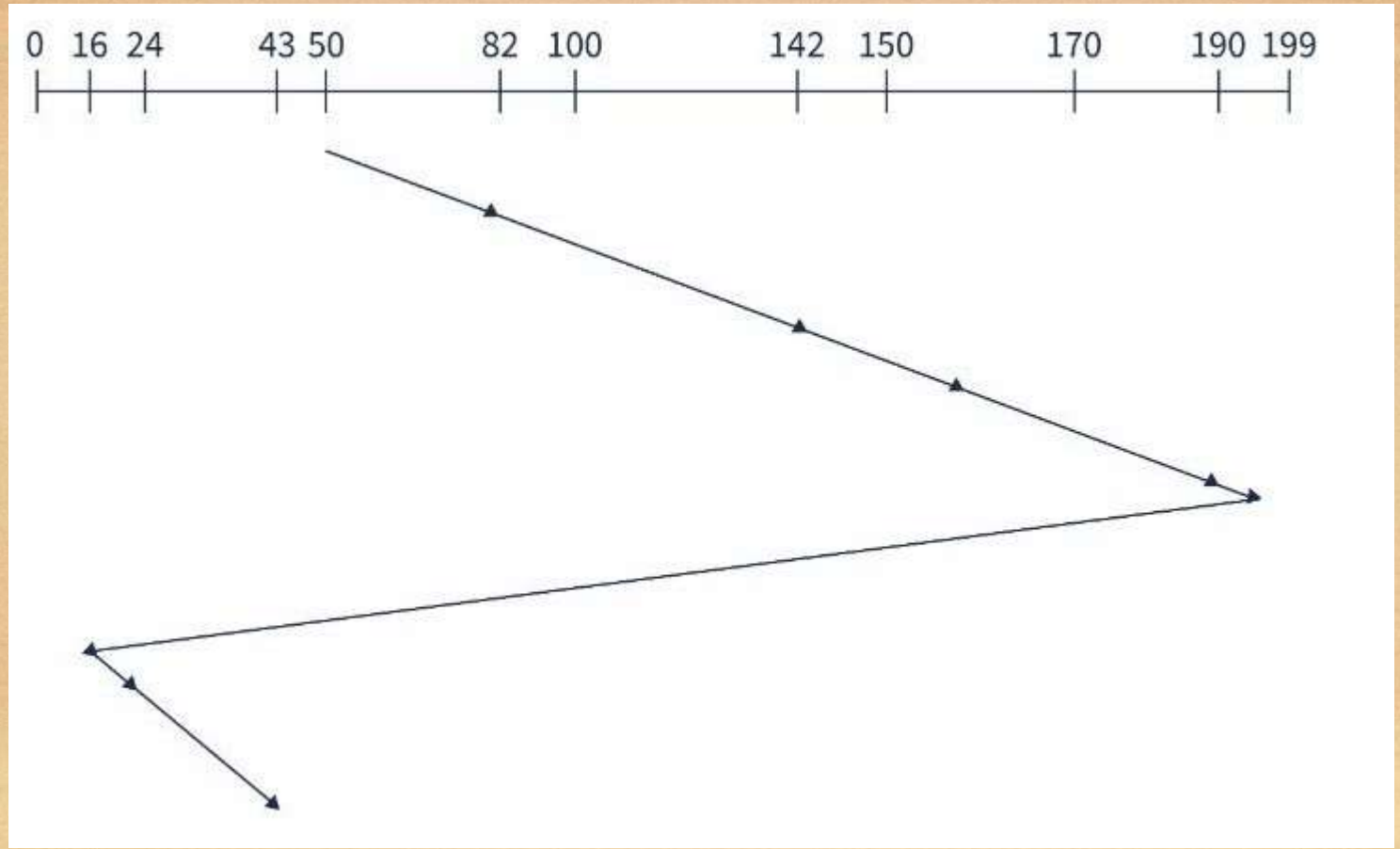


- **Explanation**: The disk arm is starting from 50 and starts to serve requests in one direction only but in spite of going to the end of the disk, it goes to the end of requests i.e.-190. Then comes back to the last request of other ends of the disk and serves them. And again starts from here and serves till the last request of the first side. Hence, **Seek time** =(190-50) +(190-16) =314
- **Advantages**:
  - Starvation does not occur.
  - Since the head does not go to the end of the disk, the time is not wasted here.
- **Disadvantage**:
  - The arm has to be conscious to find the last request.

6. C-LOOK disk scheduling algorithm:

- The C-Look algorithm is almost the same as the Look algorithm. The only difference is that after reaching the end requests, it reverses the direction of the head and starts moving to the initial position. But in moving back, it does not serve any requests.

- **Example**: Suppose a disk having 200 tracks (0-199). The request sequence(82,170,43,140,24,16,190) are shown in the given figure and the head position is at 50.



- The disk arm is starting from 50 and starts to serve requests in one direction only but in spite of going to the end of the disk, it goes to the end of requests i.e.-190. Then comes back to the last request of other ends of a disk without serving them. And again starts from the other end of the disk and serves requests of its path. Hence, Seek Time =(190-50) +(190-16) +(43-16) =341
- **Advantages**:
  - The waiting time is decreased.
  - If there are no requests till the end, it reverses the head direction immediately.
  - Starvation does not occur.
  - The time taken by the disk arm to find the desired spot is less.
- **Disadvantage**:
  - The arm has to be conscious about finding the last request.

File

- A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.

File Structure

- A File Structure should be according to a required format that the operating system can understand.
- A file has a certain defined structure according to its type.
- A text file is a sequence of characters organized into lines.
- A source file is a sequence of procedures and functions.
- An object file is a sequence of bytes organized into blocks that are understandable by the machine.
- When operating system defines different file structures, it also contains the code to support these file structure. Unix, MS-DOS support minimum number of file structure.

- A File Structure needs to be predefined format in such a way that an operating system understands. It has an exclusively defined structure, which is based on its type.
- Three types of files structure in OS:
- A text file: It is a series of characters that is organized in lines.
- An object file: It is a series of bytes that is organized into blocks.
- A source file: It is a series of functions and processes.

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.

File types- name, extension

File Type	Usual extension	Function
Executable	exe, com, bin or none	ready-to-run machine- language program
Object	obj, o	complied, machine language, not linked
Source code	c. p, pas, 177, asm, a	source code in various languages
Batch	bat, sh	Series of commands to be executed
Text	txt, doc	textual data documents
Word processor	doc,docs, tex, rrf, etc.	various word-processor formats
Library	lib, h	libraries of routines
Archive	arc, zip, tar	related files grouped into one file, sometimes compressed.

File Access Methods

- File access is a process that determines the way that files are accessed and read into memory. Generally, a single access method is always supported by operating systems. Though there are some operating system which also supports multiple access methods.
- **Three file access methods are:**
- Sequential access
- Direct random access
- Index sequential access

- **Sequential Access**
- In this type of file access method, records are accessed in a certain pre-defined sequence. In the sequential access method, information stored in the file is also processed one by one. Most compilers access files using this access method.
- **Random Access**
- The random access method is also called direct random access. This method allow accessing the record directly. Each record has its own address on which can be directly accessed for reading and writing.

- **Indexed Sequential Access**
- This type of accessing method is based on simple sequential access. In this access method, an index is built for every file, with a direct pointer to different memory blocks. In this method, the Index is searched sequentially, and its pointer can access the file directly. Multiple levels of indexing can be used to offer greater efficiency in access. It also reduces the time needed to access a single record.

File Attributes

- A file has a name and data. Moreover, it also stores meta information like file creation date and time, current size, last modified date, etc. All this information is called the attributes of a file system.
- Here, are some important File attributes used in OS:
- **Name:** It is the only information stored in a human-readable form.
- **Identifier**: Every file is identified by a unique tag number within a file system known as an identifier.
- **Location:** Points to file location on device.
- **Type:** This attribute is required for systems that support various types of files.
- **Size**. Attribute used to display the current file size.
- **Protection**. This attribute assigns and controls the access rights of reading, writing, and executing the file.
- **Time, date and security:** It is used for protection, security, and also used for monitoring

Protection in File System

- In computer systems, alot of user’s information is stored, the objective of the operating system is to keep safe the data of the user from the improper access to the system. Protection can be provided in number of ways. For a single laptop system, we might provide protection by locking the computer in a desk drawer or file cabinet. For multi-user systems, different mechanisms are used for the protection.

Types of Access :

- The files which have direct access of the any user have the need of protection. The files which are not accessible to other users doesn't require any kind of protection. The mechanism of the protection provide the facility of the controlled access by just limiting the types of access to the file. Access can be given or not given to any user depends on several factors, one of which is the type of access required. Several different types of operations can be controlled:
- **Read –** Reading from a file.
- **Write –** Writing or rewriting the file.
- **Execute –** Loading the file and after loading the execution process starts.
- **Append –** Writing the new information to the already existing file, editing must be end at the end of the existing file.
- **Delete –** Deleting the file which is of no use and using its space for the another data.
- **List –** List the name and attributes of the file.

Access Control :

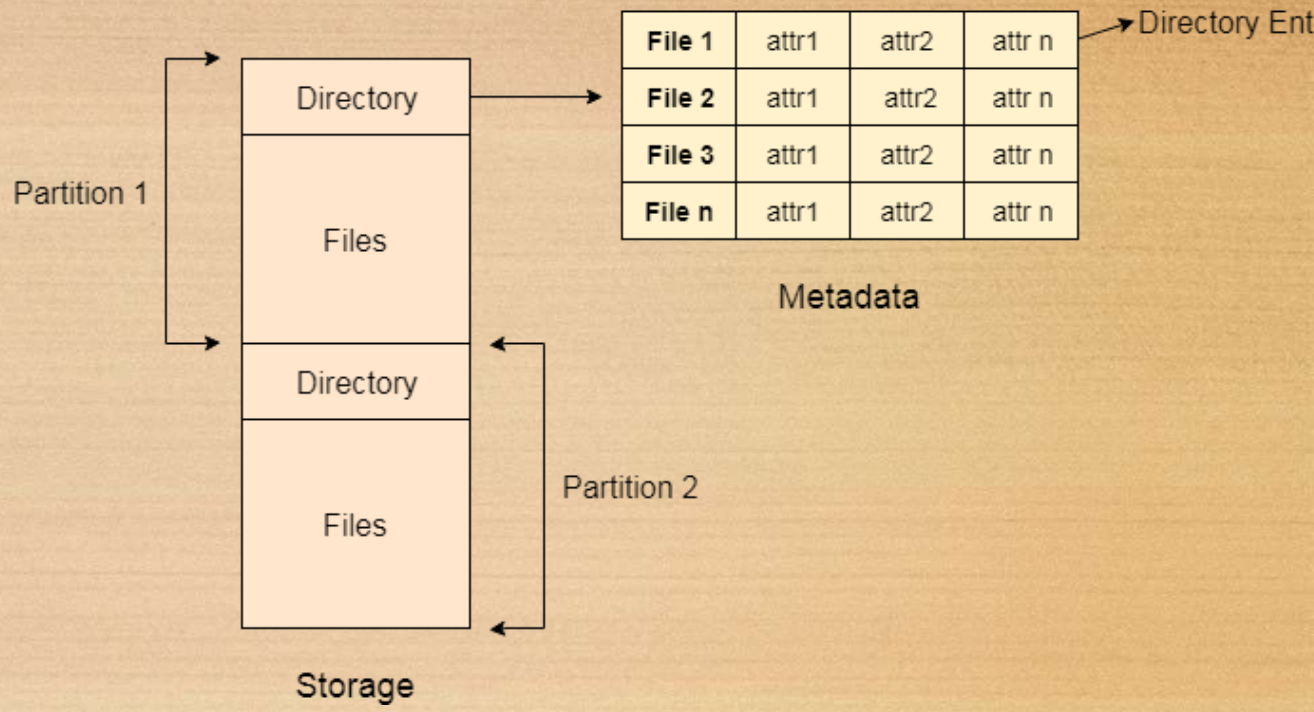
- There are different methods used by different users to access any file. The general way of protection is to
- associate identity-dependent access with all the files and directories an list called ACCESS CONTOL LIST which specify the names of the users and the types of access associate with each of the user. The main problem with the access list is their length. If we want to allow everyone to read a file, we must list all the users with the read access. This technique has two undesirable consequences:
- Constructing such a list may be tedious and unrewarding task, especially if we do not know in advance the list of the users in the system.

Directory

What is a directory?

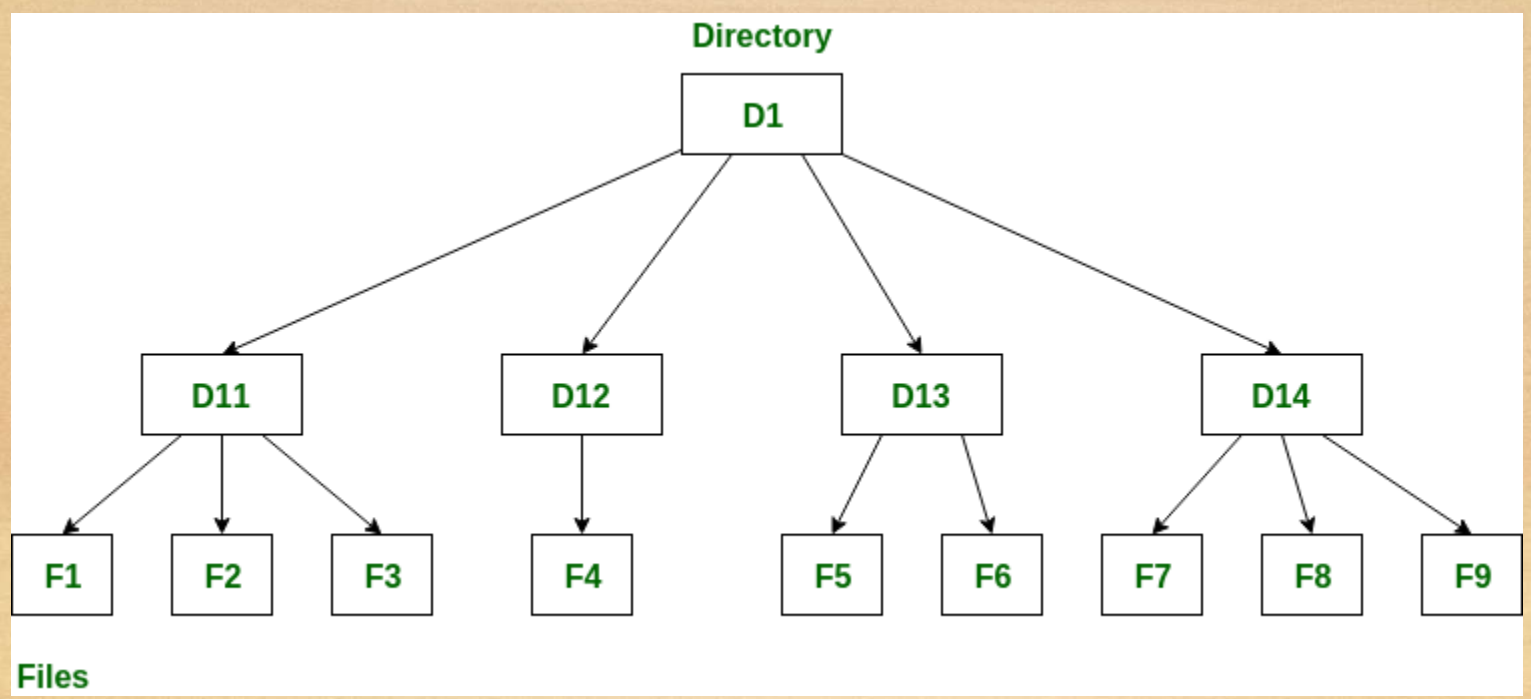
- Directory can be defined as the listing of the related files on the disk. The directory may store some or the entire file attributes.
- To get the benefit of different file systems on the different operating systems, A hard disk can be divided into the number of partitions of different sizes. The partitions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



Directory Structure

- A **directory** is a container that is used to contain folders and files. It organizes files and folders in a hierarchical manner.
- In other words, directories are like folders that help organize files on a computer.

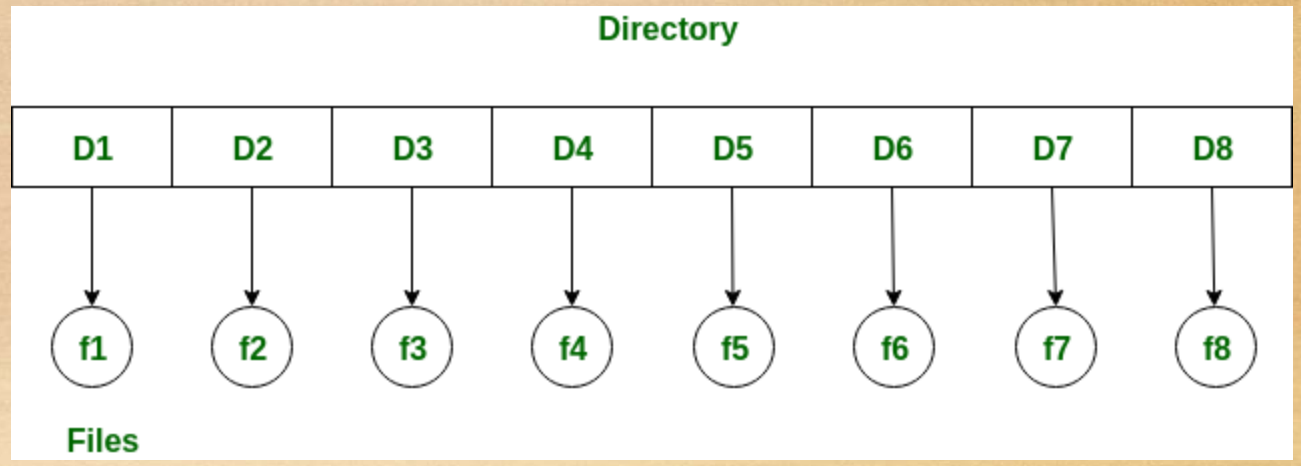


Types of Directory Structure

1. Single-Level Directory
2. Two-Level Directory
3. Tree Structure/ Hierarchical Structure
4. Acyclic Graph Structure

1. Single-Level Directory :

- The single-level directory is the **simplest directory structure**.
- In it, all files are contained in the same directory which makes it easy to support and understand.

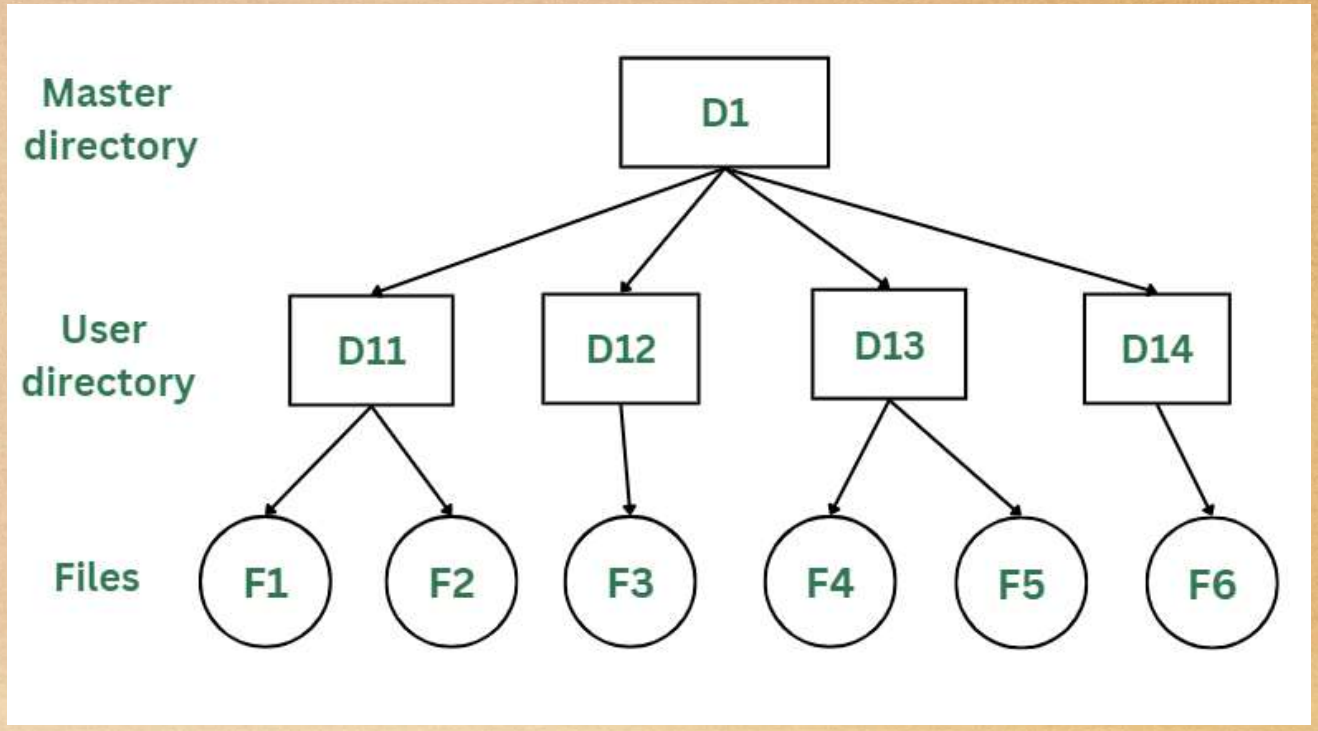


Types of Directory Structure

2. Two-Level Directory :

- single level directory often leads to confusion of files names among different users. The solution to this problem is to create a **separate directory for each user**.

- In the two-level directory structure, each user has their own **user files directory (UFD).** The UFDs have similar structures, but each lists only the files of a single user. System's **master file directory (MFD)** is searched whenever a new user id is created.

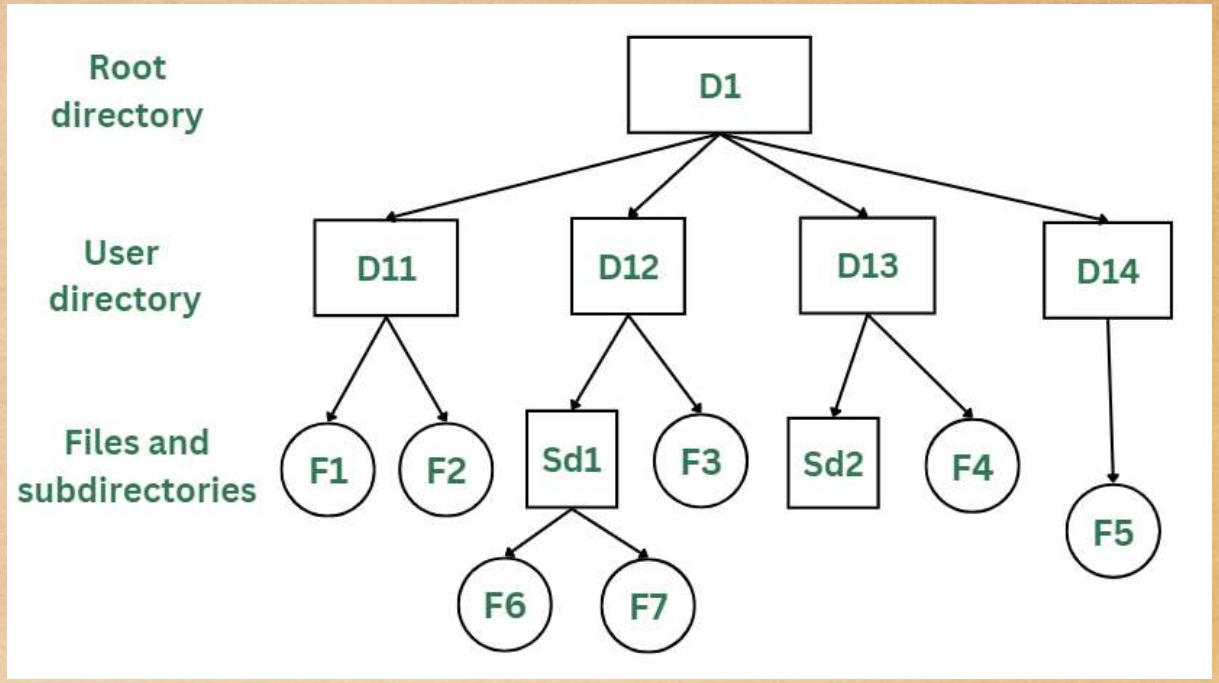


Types of Directory Structure

3. Tree Structure/ Hierarchical Structure :

- Tree directory structure of operating system is most commonly used in our personal computers. User can create files and subdirectories too, which was a disadvantage in the previous directory structures.

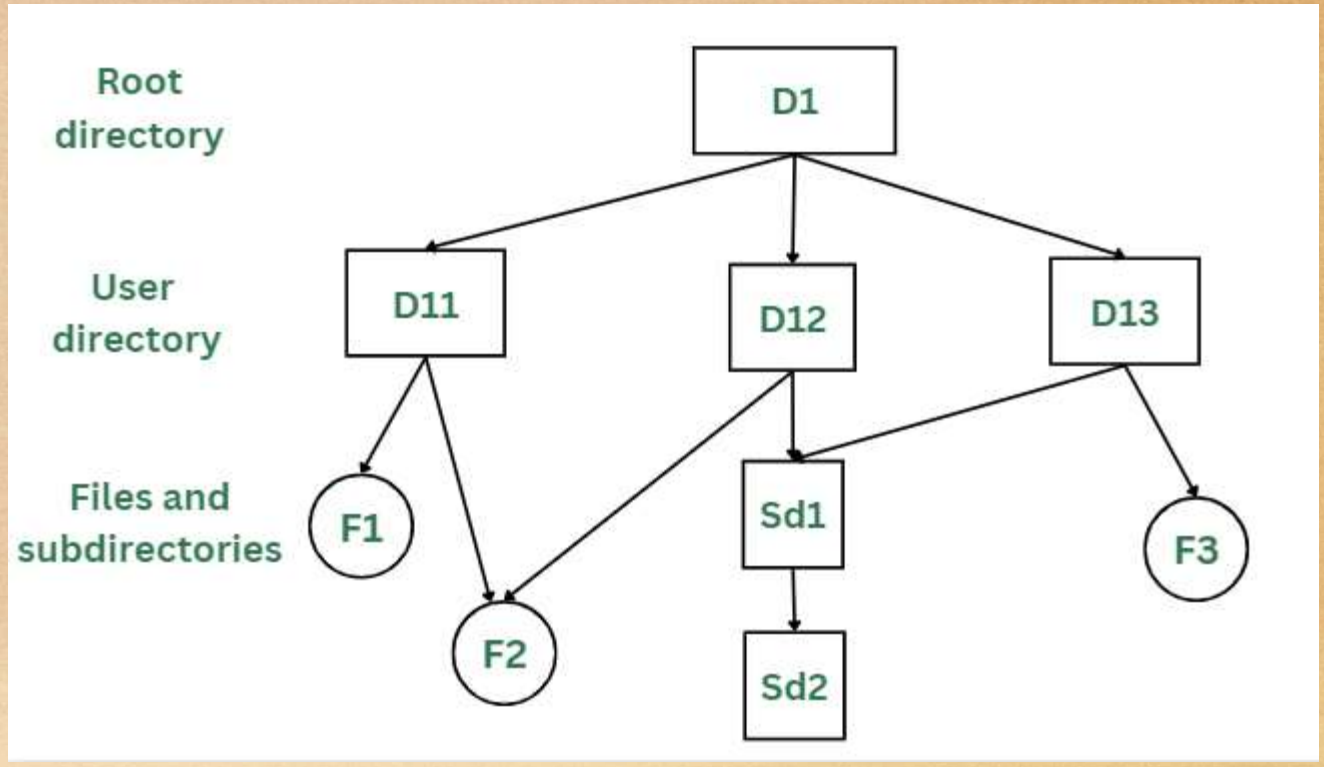
- This directory structure resembles a real tree upside down, where the **root directory** is at the peak. This root contains all the directories for each user. The users can create subdirectories and even store files in their directory.



Types of Directory Structure

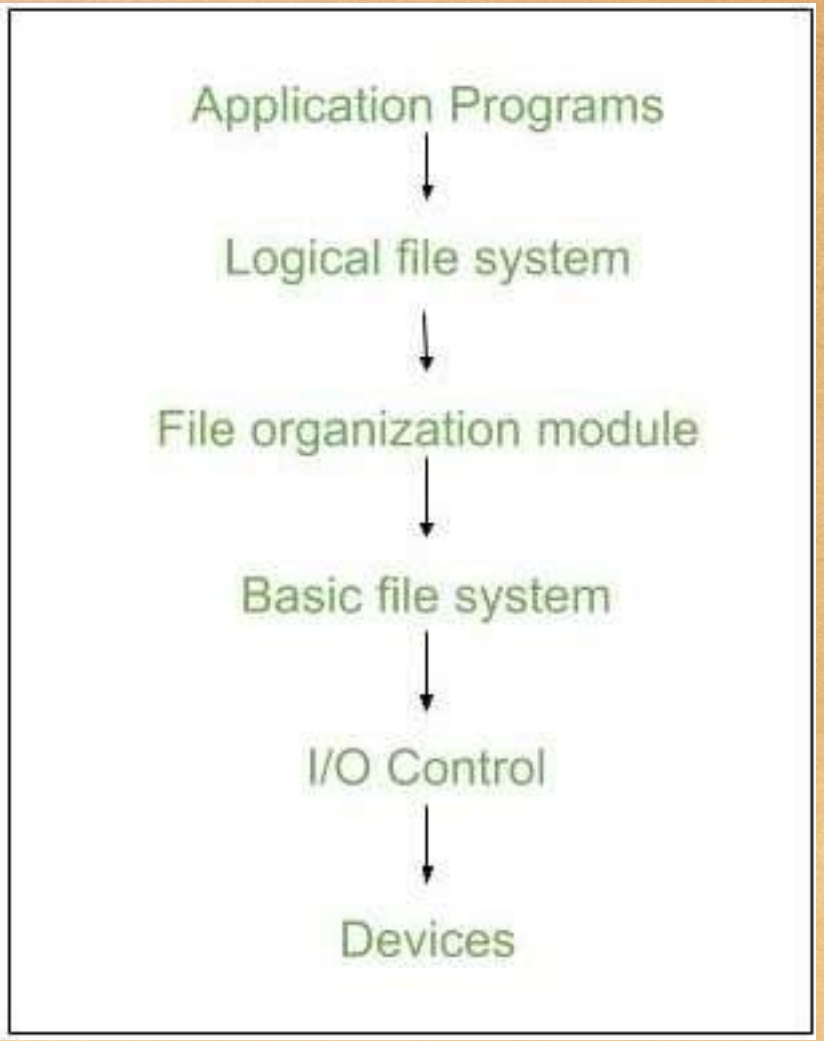
4. Acyclic Graph Structure :

- Acyclic graph directory structure, where a file in one directory can be accessed from multiple directories.
- In this way, the files could be shared in between the users. It is designed in a way that multiple directories point to a particular directory or file with the help of links.



File system Management

- The file system resides on secondary storage and provides efficient and convenient access to the disk by allowing data to be stored, located, and retrieved.
- 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.
- A file system in an operating system is organized into multiple layers, each responsible for different aspects of file management and storage.



File system Optimization

1. Disk-Space Management :

- Since all the files are normally stored on disk one of the main concerns of file system is management of disk space.

❑ Using a linked list:

Using a linked list of disk blocks with each block holding as many free disk block numbers as will fit.

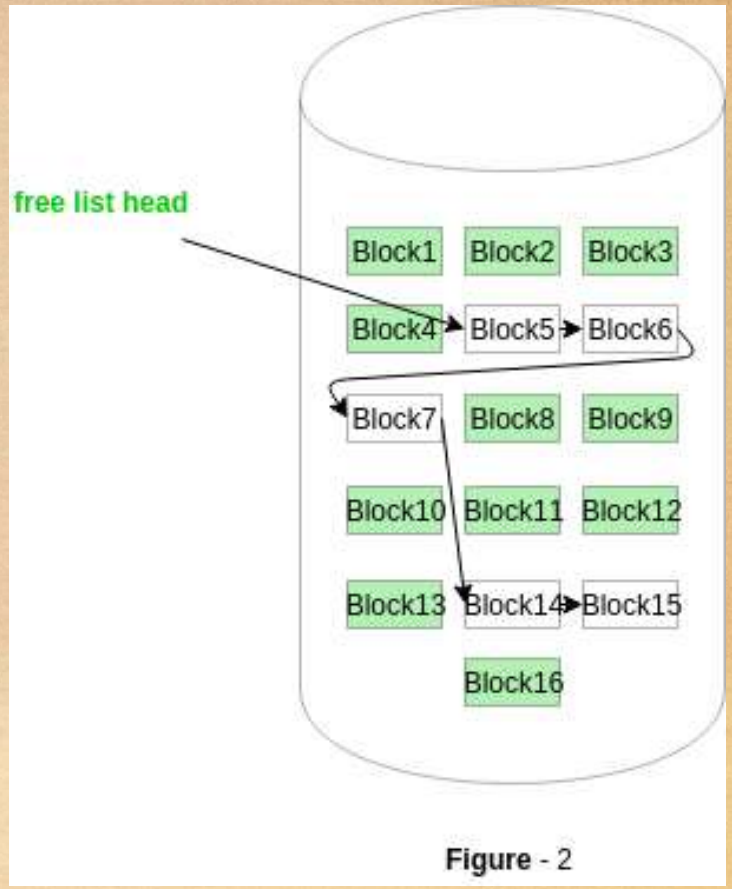


Figure - 2

❑ Bitmap:

A disk with n blocks has a bitmap with n bits. Free blocks are represented using 1's and allocated blocks as 0's as seen below in the figure.

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: 1111000111111001.

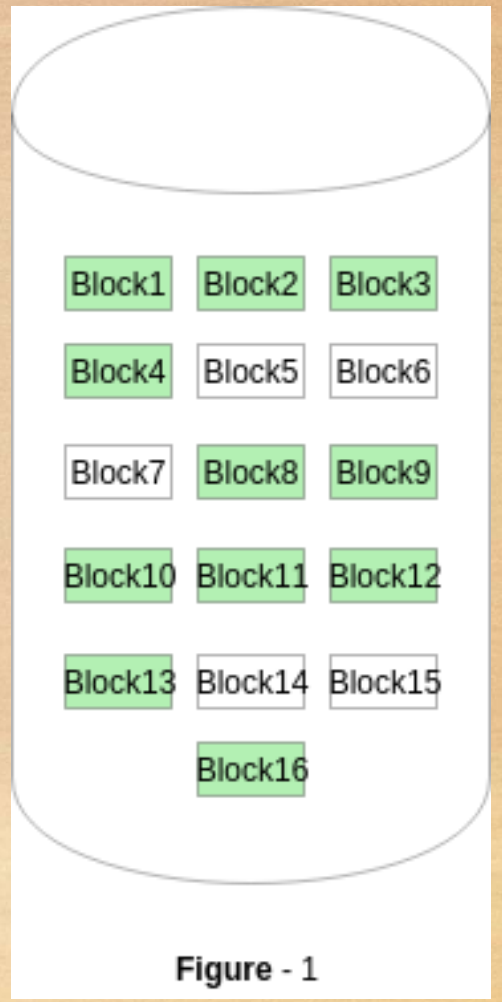


Figure - 1

❑ **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.

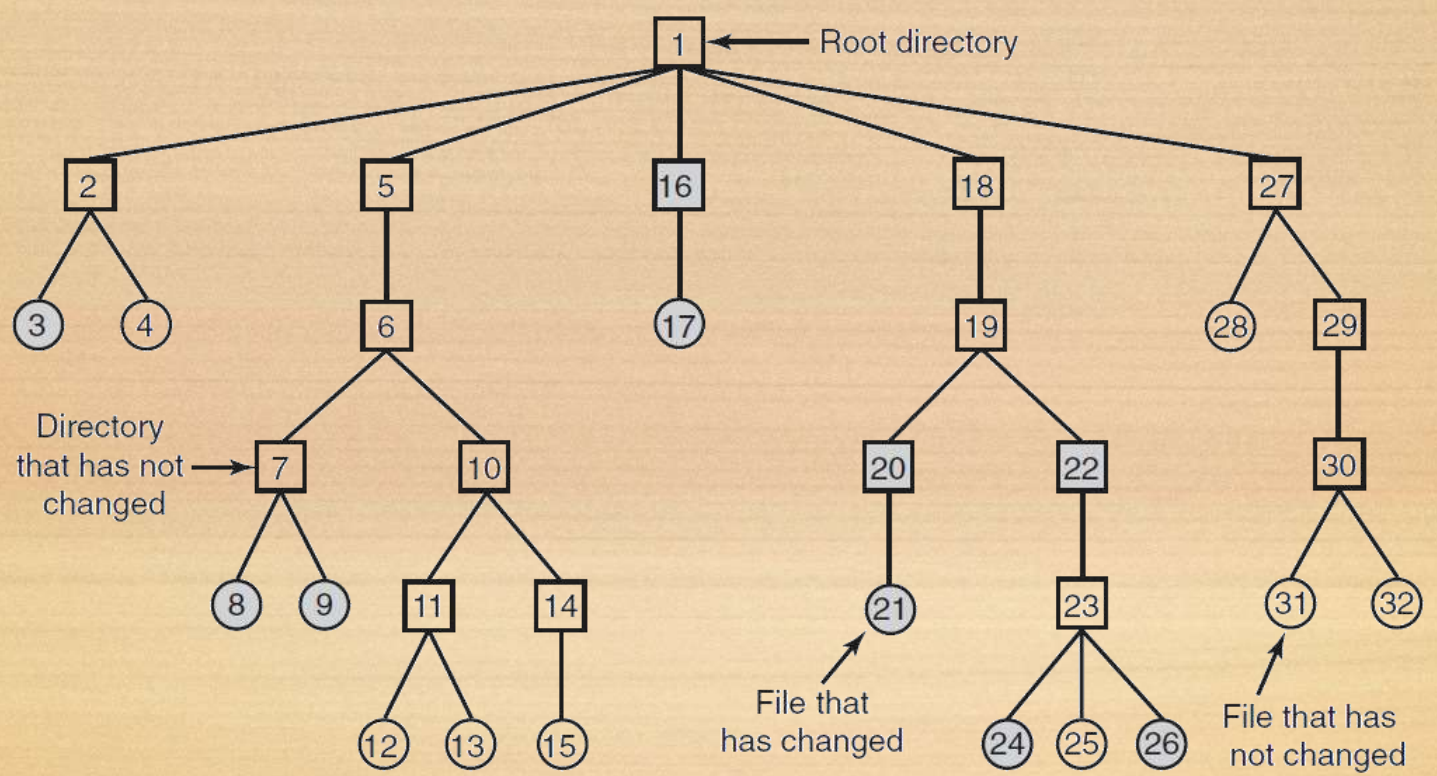
❑ **Counting :**

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:

Address of first free disk block.
A number n.

2. File-system Backups :

- If a computer's file system is irrevocably lost, whether due to hardware or software restoring all the information will be difficult, time consuming and in many cases impossible. So it is adviced to always have file-system backups.
- Backing up files is time consuming and as well occupies large amount of space, so doing it efficiently and conveniety is important. Below are few points to be considered before creating backups for files.



2. File-system Backups :

- If a computer's file system is irrevocably lost, whether due to hardware or software restoring all the information will be difficult, time consuming and in many cases impossible. So it is adviced to always have file-system backups.
- Backing up files is time consuming and as well occupies large amount of space, so doing it efficiently and conveniety is important. Below are few points to be considered before creating backups for files.