UNIT=6

Device Management:

Device management is an important function of the operating system. Device management is responsible for managing all the hardware devices of the computer system. It may also include the management of the storage device as well as the management of all the input and output devices of the computer system.

An operating system manages the devices in a computer system with the help of device controllers and device drivers. All these device controllers are connected with each other through a system bus. Device management generally performs the following:

→ Installing device and component-level drivers and related software.
→ Configuring a device so it performs as expected.
→ Implementing security measures and processes.

⊛. Classification of I/O devices:

i) **Machine readable or Block devices** → A block device is one with which the driver communicates by sending entire blocks of data. Each block has its own address and can be read from/write to independently. For example: Hard disks, USB cameras, Disk-On-Key, tapes, sensors etc.

ii) **User readable or Character devices** → A character device is one with which the driver communicates by sending and receiving single characters (bytes, octets). It accepts a stream of characters with no attention paid to block structure. It is not addressable and has no seek ability. For example: printers, graphical terminals, screen, keyboard, mouse, serial ports, sound cards etc.

iii) **Communication devices** → A communication device is a hardware device capable of transmitting an analog or digital signal over the telephone, or other communication wire, or wirelessly. The best example of a communication device is a Modem, which is capable of sending and receiving a signal to allow computers to communicate with other computers.

⊛. Device Controllers:

I/O units often consist of a mechanical component and an electronic component. The electronic component is called the device controller or adapter. A single controller can handle multiple devices; some devices have their own build-in controller. The controller has one or more registers for data and signals. The processor communicates with the controller by reading and writing bit patterns in these registers.

The controller's job is to convert the serial bit stream into a block of bytes and perform any error correction if necessary. The block of bytes is typically first assembled bit by bit, in a buffer inside the controller. After its checksum has been verified and the block has been declared to be error free, it can then be copied to main memory.

⊛. Memory-Mapped I/O:

Each control register is assigned a unique memory address to which no memory is assigned. This system is called memory-mapped I/O. In most systems, the assigned addresses are at the top of the address space or near the top of the address space.

While using memory mapped IO, OS allocates buffer in memory and informs I/O device to use that buffer to send data to the CPU. I/O device operates asynchronously with CPU, interrupts CPU when finished. Memory mapped I/O is used for high-speed I/O devices like disks, communication interfaces. CPU communicates with the control registers and the device data buffers in following three ways:

- → Using I/O port
- → Using memory mapped I/O
- → Using hybrid system.

The advantage of memory-mapped I/O is that it can be implemented in high level languages such as C and no separate protection is needed. The disadvantage is that it adds extra complexity to both hardware and OS.

⊛. DMA Operation:

Direct Memory Access (DMA) is a process for data transfer between memory and I/O, controlled by an external circuit called DMA controller, without the involvement of CPU. Most of the data that is input or output from computer is processed by the CPU, but some data does not require processing or can be processed by another device. In these situations, DMA can save processing time and is more efficient way to move data from the computers memory to other devices. For example; A PCI controller and a hard drive controller each have their own set of DMA channels.

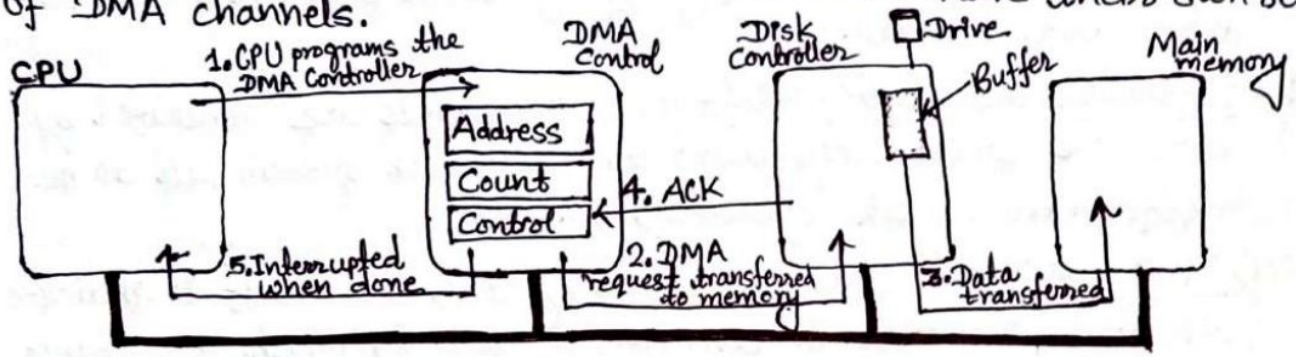


Fig: Operations of a DMA transfer

Working Procedure of DMA:

→ The CPU programs the DMA controller by its registers so it knows what to transfer where. When valid data are in disks controller buffer, DMA can begin.

→ The DMA controller initiates the transfer by issuing a read request over the bus to disk controller.

→ Data is transferred from disk controller to memory.

→ When data transfer is completed, the disk controller sends an acknowledgment signal to DMA controller. The DMA controller then increments the memory address to use and decrement the byte count. This continues until the byte count greater than 0.

→ When transfer is completed finally the DMA controller interrupt the CPU.

⊛ Interrupts:

The hardware mechanism that enables a device to notify the CPU is called an interrupt. Interrupt forces CPU to stop what it is doing and start doing something else. Interrupts are signals sent to the CPU by external devices, normally I/O devices. There are three types of interrupts:

i) Hardware interrupts → Hardware interrupts are generated by hardware devices to signal that they need some attention from the OS. They may have just received some data or they have just completed a task which the operating system provides requested, such as transferring the data between the hard drive and memory.

ii) Software interrupts → Software interrupts are generated by programs when they want to request a system call to be performed by the operating system.

iii) Traps → Traps are generated by the CPU itself to indicate that some error or condition occured for which assistance from the operating system is needed.

IO Handling:

⊛ Goals of the I/O Software:

i) Device independence → A key concept in the design of I/O software is known as device independence. It means that we should be able 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 hard disk, a DVD, or on a USB stick without having to be modified for each different device.

ii) Uniform Naming → Uniform Naming, simply be a string or an integer and not depend on the device in any way. The name of file or device should be some specific string or number. It must not depend upon device in any way. All files and devices are addressed the same way: by a path name.

iii) Error handling → Generally, errors should be handled as close as possible to the computer hardware. It should try to correct the error itself if it can in case if the controller discovers a read error. And in case if it can't then the device driver should handle it.

iv) Synchronous (blocking) and Asynchronous (interrupt-driven) transfers →

Most physical I/O is asynchronous; however, some very high performance applications need to control all the details of the I/O, so some operating systems make asynchronous I/O available to them. User programs are much easier to write if the I/O operations are blocking.

v) Buffering → Sometimes data that come off a device can't be stored directly in its final destination. Some devices have several real-time constraints, so data must be put into output buffer in advance to decouple the rate at which the buffer is filled from the rate at which it is emptied, in order to avoid buffer underruns.

vi) Sharable and Dedicated devices → Some I/O devices such as disks, can be used by many users at the same time (i.e, sharable). Other devices such as printers, have to be dedicated to a single user until that user is finished. The OS must be able to handle both shared and dedicated devices in a way that avoids problems.

⑧. Handling I/O:-

Fundamentally there are three different ways of performing I/O operations which are as follows:

i) Programmed I/O → It is the simplest form of I/O having the CPU do all the work. With programmed I/O, data are exchanged between the processor and the I/O module. The processor executes a program that ~~are exchanged between the processor and the I/O module;~~ gives it direct control of the I/O operation including sensing device status, sending a read or write command, and transferring the data. When the processor issues a command to the I/O module, it must wait until the I/O operation is complete.

Note: If Programmed I/O and Interrupt driven I/O are difficult then escape and read the differences between them, that is on next page and write in exam in a way asked.

ii) Interrupt-Driven I/O → Interrupt driven I/O is a way of controlling I/O activity by a peripheral or terminal that needs to make or receive a data transfer sends a signal. This will cause a program interrupt to be set. This strategy allows the CPU to carry on with its other operations until the module is ready to transfer data.

iii) I/O using DMA → DMA uses an additional piece of hardware - a DMA controller. The DMA controller can take over the system bus and transfer data between an I/O module and memory without the involvement of CPU. Whenever the CPU wants to transfer data, it tells the DMA controller the direction of the transfer, the I/O module involved, the location of the data in the memory, and the size of the block of data to be transferred. It can then continue with other instructions and the DMA controller will interrupt it when the transfer is complete.

⊛ Differentiate between programmed I/O and Interrupt Driven I/O.

Programmed I/O	Interrupt Driven I/O.
i) In programmed I/O processor has to check each I/O devices in sequence and in effect ask each one if it needs communication with the processor.	i) In Interrupt driven I/O, processor does not have to check whether I/O device needs it service or not.
ii) In programmed I/O, processor is busy, so it decrements the system throughput.	ii) In Interrupt driven I/O, processor is not busy, so it increments the system throughput.
iii) It is implemented without interrupt hardware support.	iii) It is implemented using interrupt hardware support.
iv) It does not depend on interrupt status.	iv) Interrupt must be enabled to process interrup driven I/O.
v) It does not need initialization of stack.	v) It needs initialization of stack.
vi) System throughput decreases as number of I/O devices connected in the system increases.	vi) System throughput does not depend on the number of I/O devices connected in the system.

⊛ IO Software Layers:

IO software is organized in following four layers:

→ Interrupt handlers
→ Device drivers.
→ Device-independent OS software,
→ User level I/O software.

I. Interrupt Handlers → An interrupt handler, also known as an interrupt service routine or ISR, is a piece of software or a callback function in an OS, 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. Then it can unblock the driver that was waiting for it.

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

II. Device Drivers → Device driver refers to a special kind of software program which controls a specific hardware device that enables different hardware devices for communication with the computer's OS. Without the required device driver, the corresponding hardware device fails to work. Device drivers are operating system specific and hardware dependent.

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.

The main purpose of device drivers is to provide abstraction by acting as a translator between a hardware device and the applications that use it.

#Disk Management:

⊛. Disk Structure:

A Disk is usually divided into Tracks, Cylinders and Sectors. Hard disk drives are organized as a concentric stack of disks or 'platters'. Each platter has 2 surfaces and two read/write heads for each surface. Each platter has the same no. of tracks. The speed of the disk is measured as two parts:

i) Transfer rate → This is the rate at which the data moves from disk to the computer.

ii) Random access time → It is the sum of the seek time and rotational latency.

Disk structure key terms:

i) Seek Time → Seek time is the time taken in locating the disk arm to a specified track where the read/write request will be satisfied.

ii) Rotational latency → It is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads.

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

iv) Disk Access Time → Disk access time is given as;

Disk Access Time = Rotational Latency + Seek Time + Transfer Time

v) Disk Response Time → It is the average of time spent by a request waiting to perform its I/O operation.

vi) Transmission Time → The time taken to access the whole record is called transmission time.

Important Formulas for Numericals:

1) Disk capacity = surfaces * (tracks/surface) * (sectors/track) * (bytes/sector) [Unit = MB]

2) Data transfer rate = no. of rotations per second * (sectors/track) * (bytes/sector) * no. of surfaces. [Unit = MB/sec].

3) Average Access Time = 1/(RPM in 1 sec) * 2 [Unit = msec].

4) Average time to read a single sector = $T_s + \frac{1}{2r} + \frac{1}{r * \text{sectors per track}}$.

where, T_s = Avg. seek time
r = Rotational speed.

Note that T_s and r should be in seconds.

⇒ Expected time to read n continious sectors on the same track is given by;

$= (\text{average time to read single sector}) + \frac{1}{n*r} * (n-1)$ [Unit=ms]

⇒ Time to read n sectors scattered over the disk = (average time to read a single sector) * sectors per track.

5). Access time for n KB block = $T_s + \frac{1}{2r} + \frac{1024 * n}{r * N}$ [Unit=ms]

where, N = no. of bytes per track.

⊛. 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.

Disk scheduling is important because:

i) Multiple I/O requests may arrive by different processes and only 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.

ii) Two or more request may be far from each other so can result in greater disk arm movement.

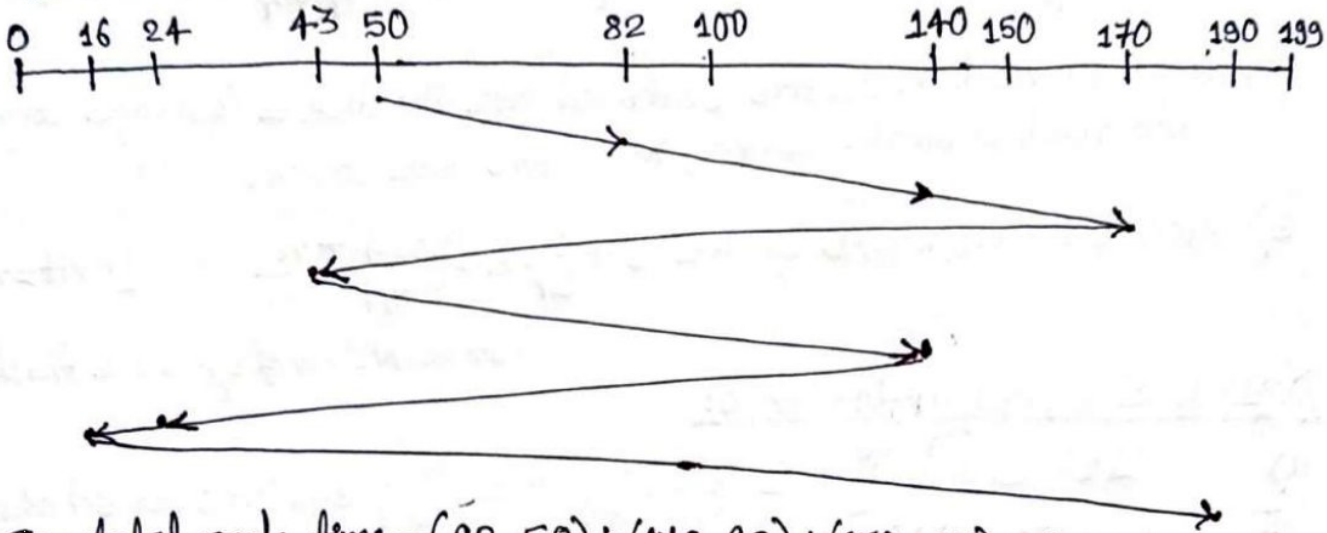
iii) Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner.

There are many disk scheduling algorithms which are as follows:-

a) **First Come-First Serve (FCFS)** → FCFS is the simplest of all the disk scheduling algorithms. In FCFS, the requests are addressed in the order they arrive in the disk queue. Since no reordering of request takes place the head may move almost randomly across the surface of the disk.

Example :- Suppose the order of request is: (82, 170, 43, 140, 24, 16, 190) and current position of Read/Write head is 50. Then find the total head movement or total seek time, if it's tail track or end is at 199.

Solution:



So, total seek time = (82-50) + (170-82) + (170-43) + (140-43)
+ (140-24) + (24-16) + (190-16)
= 642

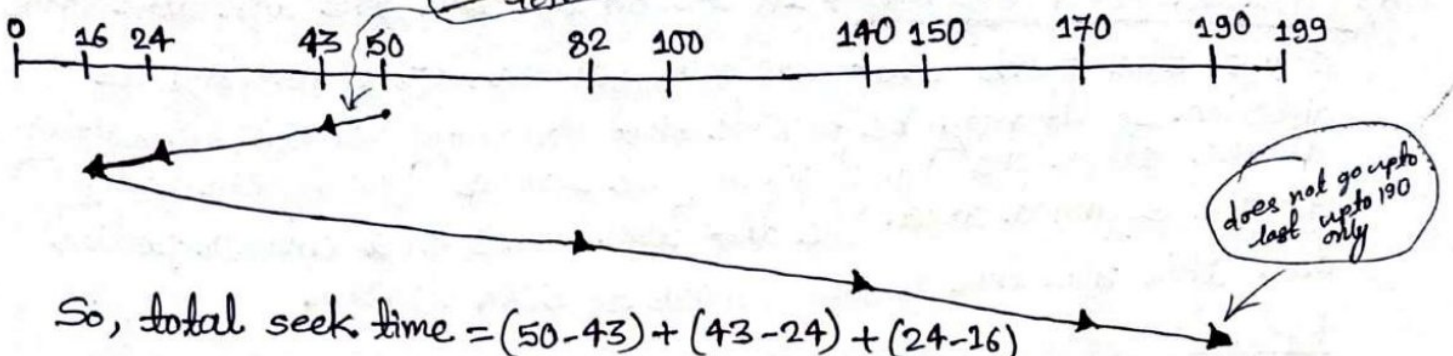
Advantages:
- → Every request gets a fair chance.
- → No indefinite postponement.

Disadvantages:
- → Does not try to optmize seek time.
- → May not provide the best possible service.

b) **Shortest Seek Time first (SSTF)** → In SSTF, requests having shortest seek time are executed first. So, the seek time of every request is calculated in advance in 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.

Example: (We will take same question of FCFS, since it will be easier and will also help to compare which one is better).

Solution:



So, total seek time = (50-43) + (43-24) + (24-16)
+ (82-16) + (140-82) + (170-40) + (190-170)
= 208

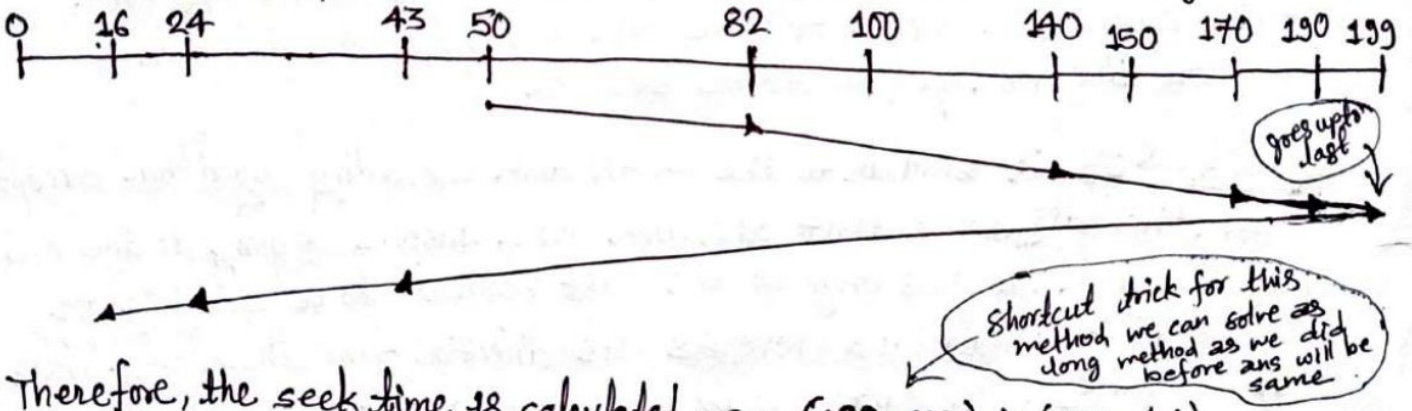
Advantages:
→ Average response time decreases.
→ Throughput increases.

Disadvantages:
→ Overhead to calculate seek time in advance.
→ Can cause starvation for a request if it has higher seek time as compared to incomming requests.
→ High variance of response time as SSTF favours only some requests.

c) Elevator (SCAN) → In SCAN algorithm the disk arm moves into a particular direction and services the requests comming in it's path and after reaching the end of disk, it reverses its direction and again services the request arriving in its path.

Example: [We are taking same example]

Solution: Suppose that the disk arm should move "towards larger value".



Therefore, the seek time is calculated as = (199-50) + (199-16)
= 332

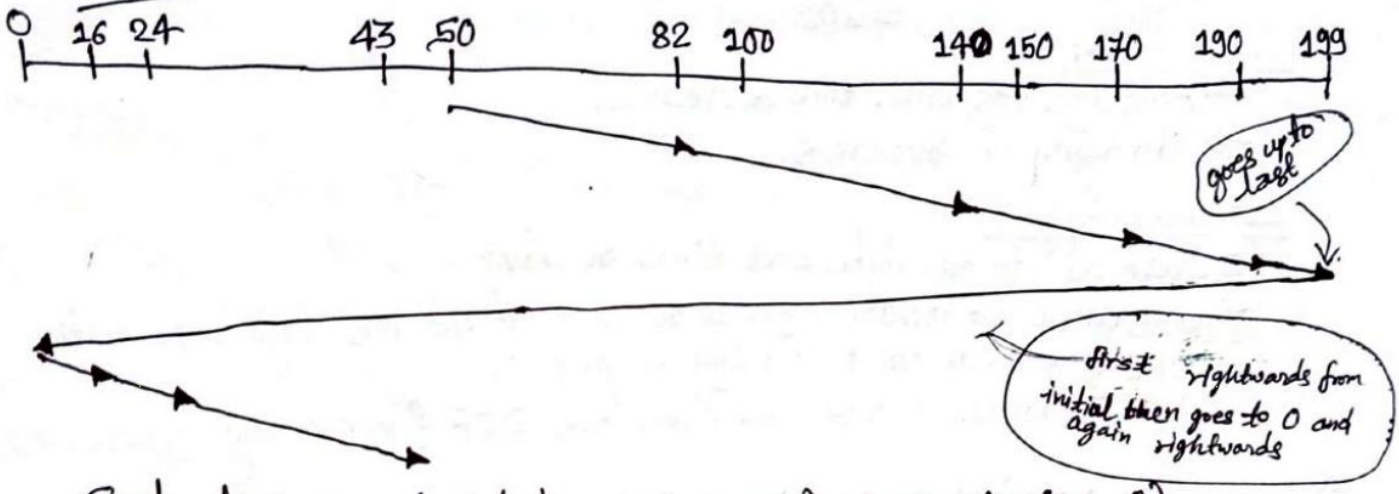
Advantages:
→ High throughput
→ Low variance of response time
→ Average response time.

Disadvantages:
→ Long waiting time for requests for locations just visited by disk arm.

d) Circular-SCAN (C-SCAN) → In C-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. The disk arm moves in a circular fashion and this algorithm is also similar to SCAN algorithm.

Example: [Same example question].

Solution:



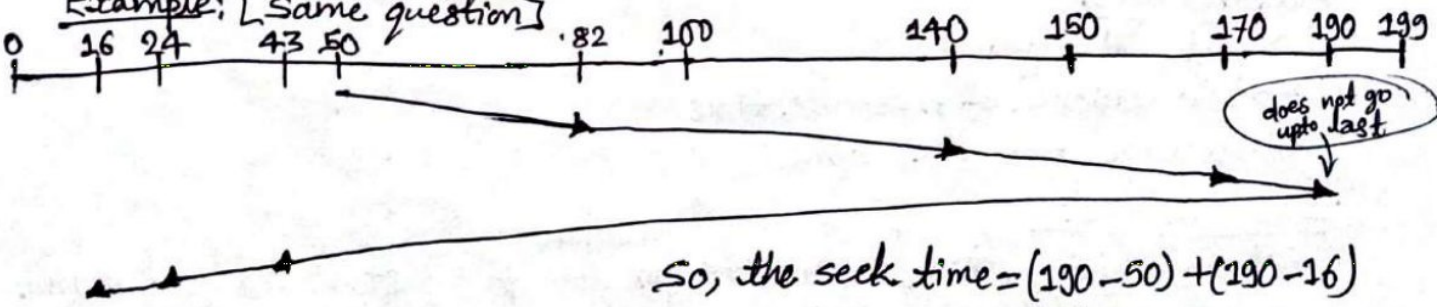
Seek time is calculated as = (199-50)+(199-0)+(43-0)
= 391

Advantages:
- → Provides uniform waiting time.
- → Provides better response time.

Disadvantages:
- → It causes more seek movements as compared to SCAN algorithm.
- → It causes the head to move till the end of the disk even if there are no requests to be serviced.

e) LOOK → It is similar to the SCAN disk scheduling algorithm except for the difference that the disk arm inspite of going to the end of the disk, it goes only to the last request to be serviced in front of the head and reverses its direction from there only. Thus it prevents the extra delay which occured due to the unnecessary traversal to the end of the disk.

Example: [Same question]



So, the seek time = (190-50)+(190-16)
= 314.

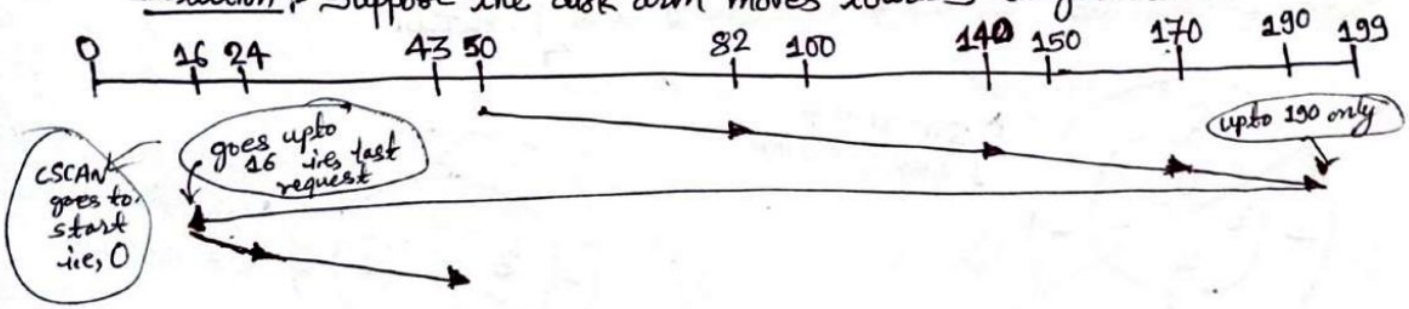
Advantages:
→ It provides better performance compared to SCAN.
→ It does not lead to starvation.

Disadvantages:
→ There is an overhead of finding the end requests.
→ It causes long waiting time for the cylinders just visited by the head.

f) C-LOOK → Circular-LOOK is an improved version of LOOK algorithm. It is similar to CSCAN disk scheduling algorithm. The difference is that the disk arm in spite of going to the end goes only to the last request to be serviced.

Example: [Same example question].

Solution: Suppose the disk arm moves towards larger value.



So, the seek time = (190-50) + (196-16) + (43-16)
= 341

Advantages:
→ It provides better performance compared to CSCAN
→ It does not lead to starvation.

Disadvantage:
→ There is an overhead of finding the end requests.

⊛. Disk Formatting:

Disk formatting is a process to configure the data-storage devices such as hard-drive, floppy disk and flash drive when we are going to use them for the very first time. Disk formatting is usually required when new operating system is going to be used by the user. It is also done when there is space issue and we require additional space for the storage of more data in the drives. When we format the disk then the existing files within the disk are also erased. Disk formatting has also the capability to erase the bad applications and various sophisticated viruses. The frequent formatting of disk decreases the life of hard-drive.

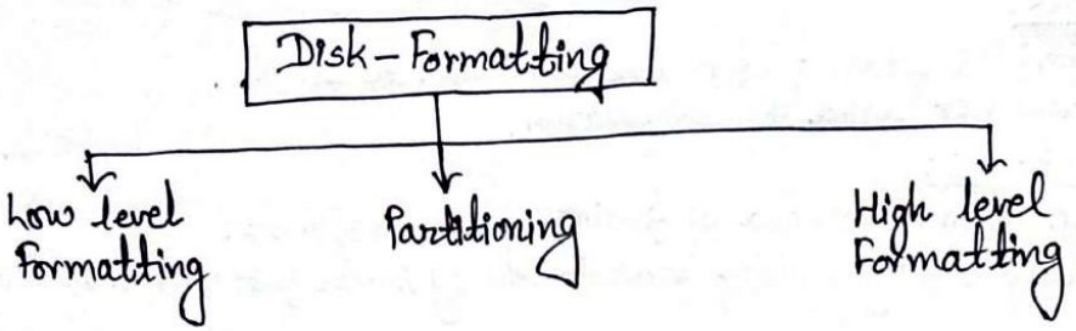


Fig: Formatting process of disk.

a) Cylinder Skew → If the position of sector s_i on each track is offset from the previous track then such an offset is called cylinder skew. It allows the disk to read multiple tracks in one continious operation without losing data.

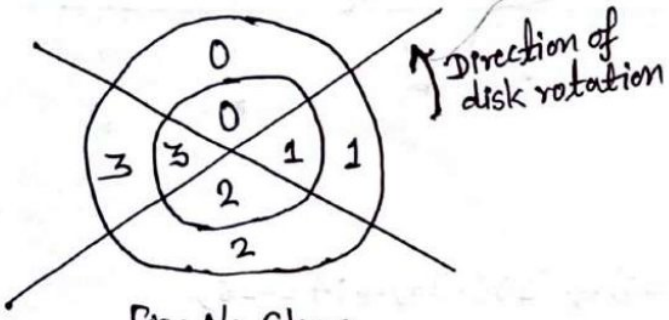


Fig: No Skew.

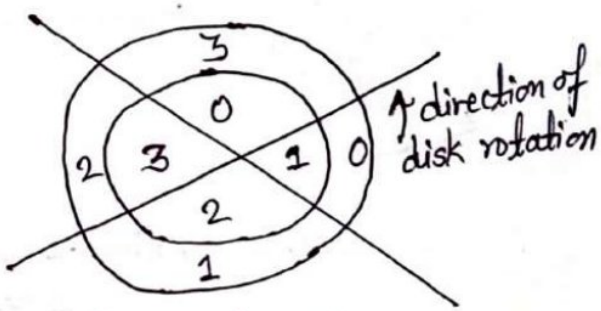


Fig: Sector Skew.

b) Interleaving → Interleave is the process through which gaps are placed between two sectors on the platter of a disk. This was done in earlier days when the computers were not quick enough to read continious streams of data.

Without interleaving there would be no gaps between the sectors and data would arrive immediately before the reading unit is ready. Due to this to read the same data a complete rotation of the disk would be required again. The interleaving ratio was not fixed and can be set by the end user depending on their system specs. Nowdays the ratio of interleaving is 1:1 i.e, no interleaving is used.

c) Error Handling → Manufacturing defects introduce bad sectors; If defect is small (say few bits). Then the hard drive is shipped and ECC corrects the error every time the sector is accessed. If error is bigger it cannot be masked. There are two ways for error correction:

i) When one of the sectors is bad the controller remaps between the bad and spare sectors.
ii) If controller cannot remap, the operating system does the same thing in software.

⊛. Redundant Array of Independent Disks (RAID):

RAID is a technique which makes use of a combination of multiple disks instead of using a single disk for increased performance, data redundancy or both. It is the way of storing the same data in different places on multiple hard disks to protect data in the case of a drive failure. RAID works by placing data on multiple disks and allowing I/O operations to overlap in a balanced way, improving performance.

Stripping in RAID → The process of dividing large data into multiple disks for faster access is called stripping in RAID.

Mirroring in RAID → It is a mechanism in which the same data is written to another disk drive.

Parity in RAID → It is a method used to rebuild data in case of failure of one of the disks.

Hot Spares in RAID → Hot Spare is an extra drive added to the disk array, to increase the fault tolerance.

There are various RAID levels that can be used. Selecting the suitable raid level for our application depend on following things:

→ We can select a raid level based on the performance that it provides.
→ ~~RAID~~ RAID level based on the level of redundancy it provides.
→ RAID level based on read and write operations.