Unit-6 : Device Management.

-Ankit Pangeni

DATE

(Introduction.)

Devices are the physical units for accepting the inputs and producing the output. Some device also store the data and information. OS is responsible to manage and control all the I/O operation and I/O devices.

OS 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 & processes

* Classification of I/O devices

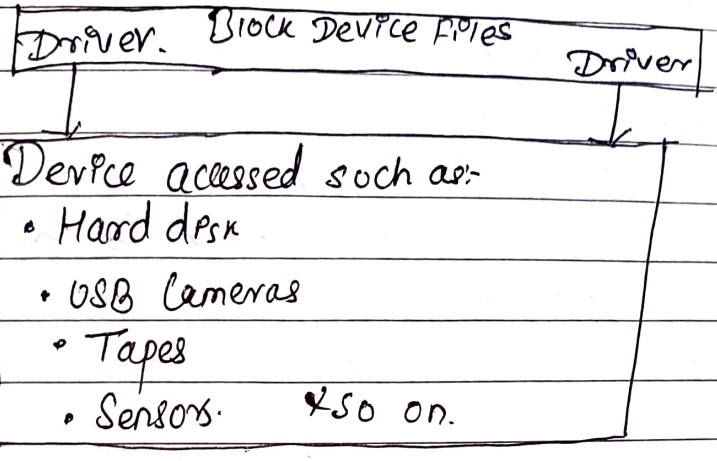
I/O devices can be divided into following 3 category

- Machine readable or Block devices

A block device is one with which the driver communicates by sending entire blocks of data. Here, information in fixed blocks with each block has its own address. Each block can

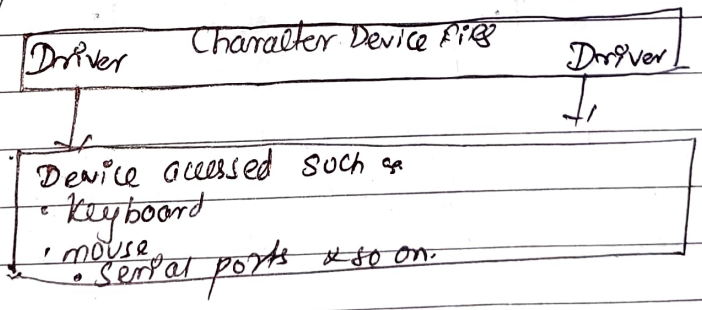
DATE

be read from / written to independently. Ex. Hard disks, USB cameras, tapes, sensors, etc.



- User readable or Character devices

A device in which the OS and drivers communicate to the computer by sending and receiving single Character bit / byte. It accepts a stream of Character with no block structure. Eg: keyboard, mouse, Serial ports, Sound cards, etc. They don't have physically addressable storage media, such as tape drive or serial ports, where I/O is normally performed in byte stream



- Communication device.

It is a hardware device capable of sending & receiving signals to allow computers to talk to other computers over the telephone. Ex: Network Interface card, wifi device, & access point. without a communication device we'd have to use a net to transfer or share data between computers

DATE

* Device Controllers

A controller is a collection of electronics that can operate a bus or a device. A single controller can handle multiple devices; Some devices have their own built 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.

Device drivers are the software to handle a particular device. OS uses device driver to handle all the input devices. A device controller works as interface with a device & a device driver. Device controller is a circuit board between device and OS but device driver is specific to an OS and hardware dependant. Basically, device drivers are softwares & device controllers are hardwares. As an interface, its main task is to convert serial bit stream to block of bytes, perform error correction as necessary.

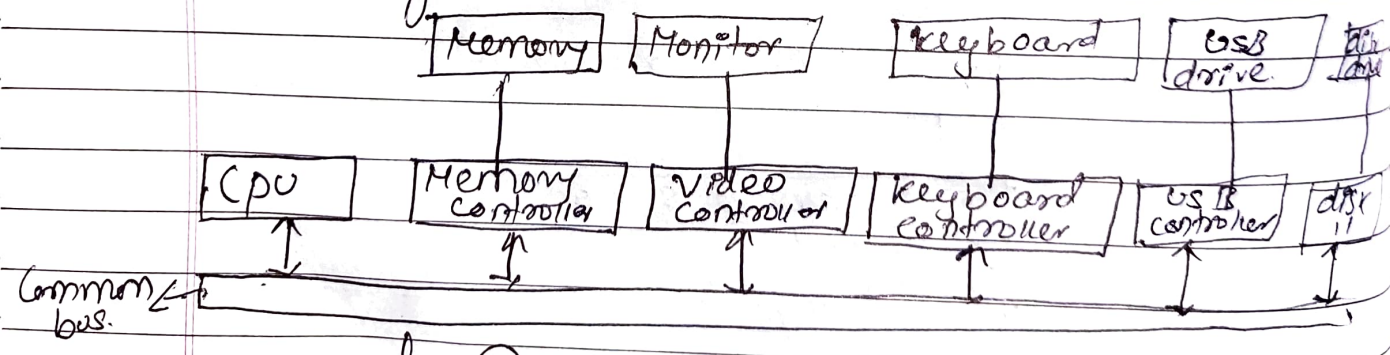


fig: Device controllers system

DATE

* Communication of cpu to I/O Device

The CPU must have a way to pass information to and from an I/O device. The approaches are

- Special Instruction I/O.

This uses CPU instructions that are specially 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

The device controller has its own control registers which can be mapped into the memory space which is known as memory mapped IO. In this I/O, all the control registers is assigned a unique address. There is no any special protection mechanism to perform I/O.

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. The advantage of this method is that every instruction which can access memory can be used to manipulate an I/O device. Memory mapped I/O is used for most high-speed I/O devices like disks, communication interface. CPU communicates with control registers & the device data buffers by:

→ using I/O port, using memory mapped IO, using hybrid system

classmate

PAGE

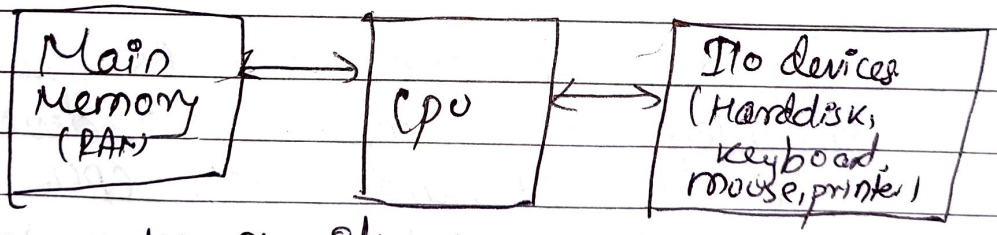
DATE

Advantage: It can be implemented in high level languages such as C and no separate protection is needed.

Disadvantage: It adds extra complexity to both hardware and OS.

DMA Operation

DMA (Direct memory access) is the data transfer technique in which the peripherals (I/O devices) manage the memory buses for direct interaction with main memory without involving the CPU. Using DMA technique, large amount of data is transferred between memory and peripheral devices without severely impacting CPU performance. During DMA transfer operation, CPU is idle & has no control of the memory buses



communicate CPU I/O without using DMA

Slow devices like keyboard will generate 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 OS would spend most of its time handling these interrupts. So, a typical computer uses DMA hardware to reduce this overhead.

classmate

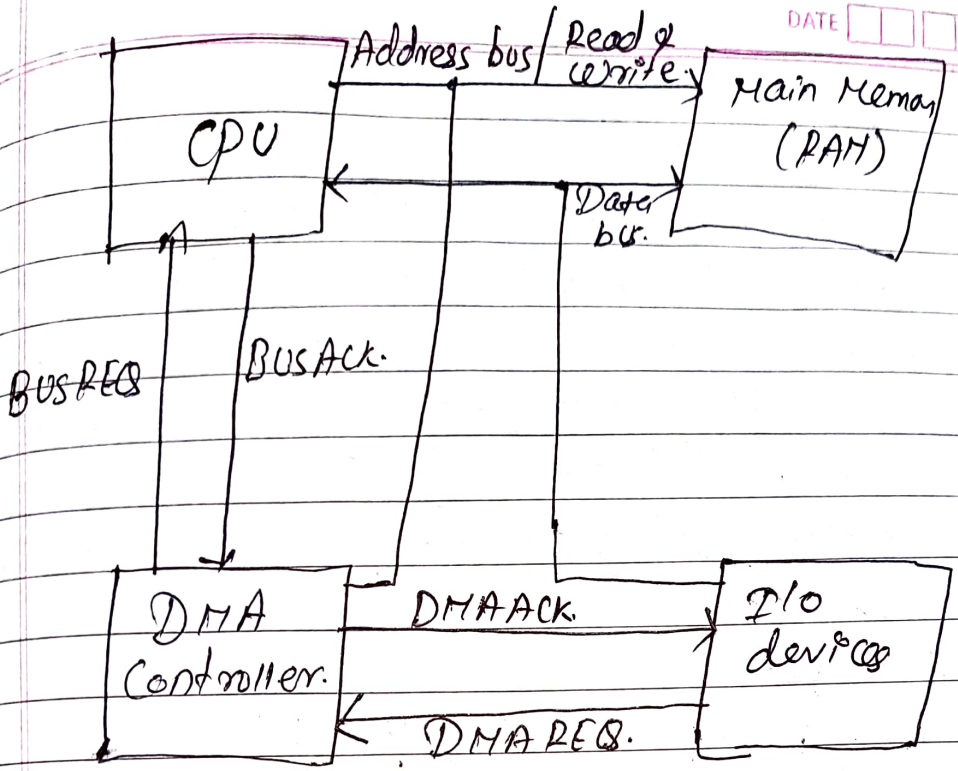


fig: DMA Transfer operation

→ CPU communicates with DMA through address & data buses.

→ DMA has its own address which activates RS (Register select) and DS (DMA select) lines

→ When I/O device sends a dma request, the dma controller activates Bus Request (BR) line, and requests CPU for bus, then CPU responds with BG (Bus Grant) line.

→ DMA then puts current value of its address register into address bus, initiates Read (RD) & Write (WR) signals and sends DMA acknowledge (DMA ACK) to the I/O device

→ When BG=0, RD & WR signal allow CPU to communicate with internal DMA registers. When BG=1, DMA communicates with RAM through RD & WR line

DATE

* Interrupts

It is an external request signal that is performed by I/O device for requesting a service to the processor. so, it is a hardware mechanism in which I/O device notify the cpu for the certain task. It forces the cpu to stop what it is doing and start doing something else.

Types.

- Hardware Interrupts: They are generated by hardware devices to signal that they need some attention from the os. Cause of interrupt :- maybe they are requesting the transfer of data, or finished the transfer of data.

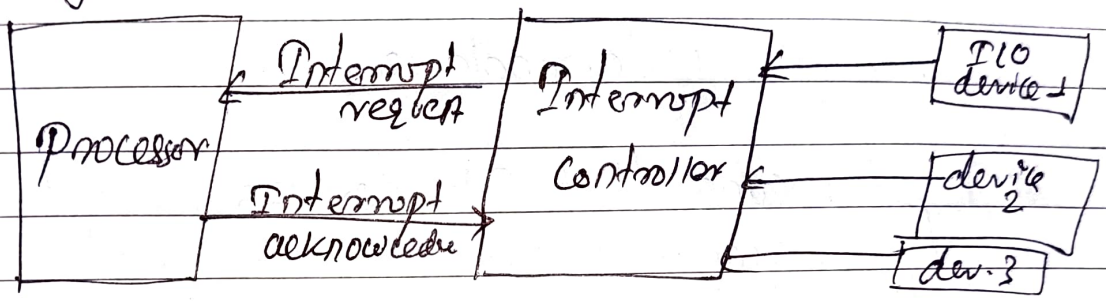
- Software Interrupts:- They are initiated by executing an instruction. It is a special system call instruction that behaves like an interrupt. It can be used by a programmer to initiate an interrupt procedure at any desired time in the program. ex:- Enable Interrupt, Disable Interrupt. etc

- Trap:- It is generated by cpu itself to indicate that some error condition occured for which assistance from the OS is needed. It is non-maskable interrupt so it can't be disabled or ignored.

DATE

Interrupts are important because, they give the user better control over the computer. Without interrupts, a user may have to wait for a given application to have a higher priority over the CPU to be run. which consumes a lot of time.

Working mechanism of Interrupts:



- There is an interrupt controller to manage & control the transfer of data between processor & device
- I/O devices send interrupt request to processor
- Processor receives interrupt request signal & stops the current activities it is doing.
- Processor sends acknowledge signal to the I/O device after saving the data in stack & register
- After receiving the acknowledge signal by input/output device, it sends the data to the processor.
- Now, the processor processes the data using ISR (Interrupt service routine)

After completion of transferring & processing the data, the processor returns back to its previous activity by retriving the data from stack & register.

PAGE

DATE

* (I/O Handling)

¥ Goals/principles of I/O Software

There are many goals of I/O Software. Some of the major principles of I/O software are:

- Device independence

It means that I/O devices should be accessible to programs without specifying the device in advance. Ex, a program that reads a file as input should be able to read a file on a ~~file~~ floppy disk, or a hard disk, or a CD-ROM without having to modify the program for each different device.

- Uniform Naming

Uniform Naming, simply be a string or an integer and not depend on the device in any way. In UNIX, all disks can be integrated in the file-system hierarchy in arbitary ways so the user need not be aware of which name corresponds to which device. It must not depend upon device in any way. All files & devices are addressed the same way; by a path name.

- Error handling

Generally, error should be handled as close as possible to the computer hardware. It

classmate

PAGE

Should try to correct the error itself if it can in case of the controller ~~reads~~ discovers a read error. And in case if it can't, then the device driver should handle it; perhaps by just trying to read the block again. In many cases, error recovery can be done transparently at a low level without the upper levels even knowing about the error.

- Synchronous (blocking) & 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 OS make asynchronous I/O available to them. The CPU starts the transfer & goes off to do something other until the interrupt arrives. In case of I/O operations are blocking the user programs are much easier to write. After a read System Call the program is automatically suspended until the data are available in the buffer. Basically it is up to the OS to make the operation that are really asynchronous look blocking to the user program.

- Buffering

Sometimes data that come off a device can't be stored directly in its final destination. Buffering sometime has a major impact on the systems I/O performance because it involves considerable copying. Eg: Data packets come from the network cannot be directly stored in physical memory. Data must be put into output buffer in advance to decouple the rate at which buffer is filled & the rate at which it is emptied, in order to avoid buffer under run.

DATE

o Sharable devices:-

The I/O devices can be sharable to the multiple users at the same time. Some users can access printer whereas other can access secondary storage at the same time. It should be noted that a single device can't be accessed by multiple user at same time.

* Handling I/O.

There are three fundamentally different ways of performing I/O operations which are listed below.

o Programmed I/O

It is the simplest type of I/O technique for the exchanges of data or any types of communication between the processor & external devices. Data are exchanged between the processor & the I/O module.

Here, the cpu (processor) has to check each I/O device in sequence and in effect "ask" each one if it needs communication with the processor. This mode decreases the system throughput. It is inefficient use of cpu. During polling, processors is busy & therefore have serious & decrement effect on system throughput. It is implemented without interrupt hardware support & it does not depend on interrupt status.

classmate

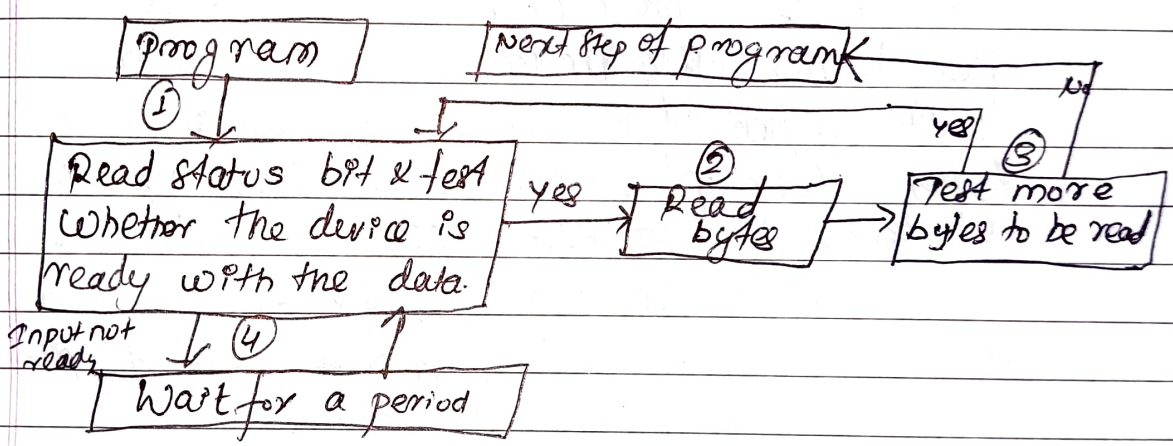
PAGE

DATE

Programmed I/O does not need initialization of stack.

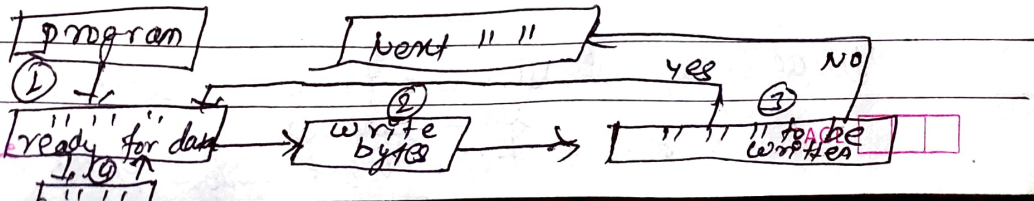
Programmed I/O Mode:- Input Data Transfer

- Each input is read after first testing whether the device is ready with the input.
- The program waits for the ready status by repeatedly testing the status bit & till all targeted bytes is read from the input device.
- The program is in busy state only after the device gets ready else in wait state



Programmed I/O Mode:- Output Data Transfer

- Each output written after 1st testing whether the device is ready to accept the byte at its output register or output buffer is empty
- The program waits for the ready status by repeatedly testing the status bit(s) & till all targeted bytes are written.
- The program in busy state only after the device gets ready else waits state.



DATE

Advantages: o Simple to implement
• Very little hardware support.

Disadvantage: o Busy waiting
o Tied up cpu for long period with no use for work

o Interrupt-Driven I/O

This mode of I/O data transfer uses interrupt facility. In this method, the CPU doesn't monitor peripherals. I/O devices generate interrupt signal to CPU when it is ready for data transfer. When interrupt signal is received by CPU, then it stops its ongoing task & serves I/O devices. After serving I/O devices, it resumes its previous job. It is efficient use of cpu and increases the system throughput.

It removes the need of CPU to continually pole input devices to see if it must ready any data. When an input device has data, then the appropriate I/O module can interrupt the CPU to request a data transfer

Advantages: • It is faster than programmed I/O
o It is more efficient

Disadvantages: • Can be tricky to write if using a low level language
• Can be tough to get various pieces to work together.

classmate

PAGE

DATE

I/O Using DMA

Although interrupt driven I/O is much more efficient than program controlled I/O, all data is still transferred through the CPU. This will be inefficient if large quantities of data are being transferred between the peripheral and memory. The transfer will be slower than necessary, & the CPU will be unable to perform any other actions while it is taking place. Many systems use additional strategy known as direct memory access (DMA)

[write definition from previous page]. 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 & main memory without the intervention of the CPU. Whenever CPU wants to transfer data, it tells the DMA controller the direction of the transfer, the I/O module involved, the location of data in memory & size of block of data to be transferred. It can then ~~continue~~ continue with other instructions & the DMA controller will interrupt it when the transfer is complete.

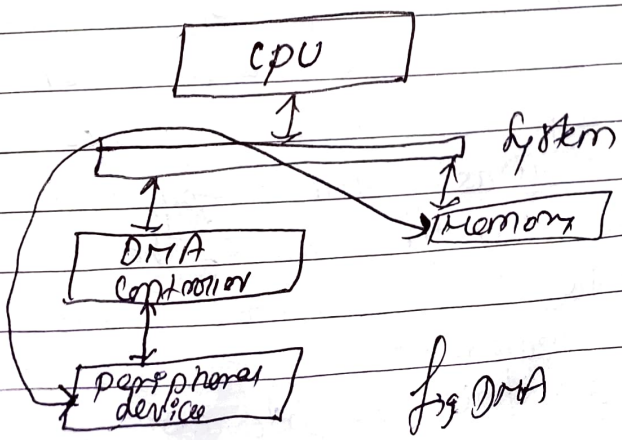


fig DMA

PAGE

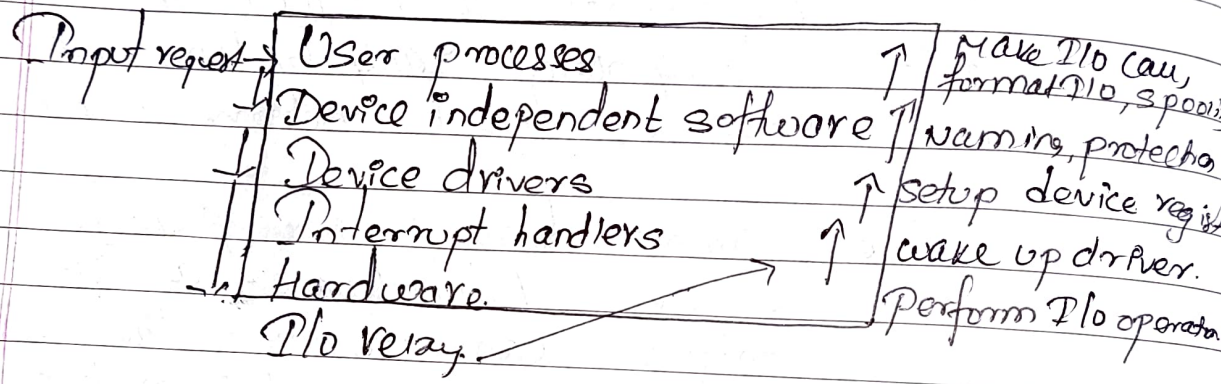
classmate

DATE

* I/O Software layers

Basically, I/O software is organized in the four layers

- Interrupt handlers
- Device drivers
- Device-independent input/output software.
- User-space input/output software.



- Interrupt handlers

An interrupt handler, also known as an interrupt service routine (ISR), is a piece of software or more specifically a callback function in an OS, 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 & wakes up process that was waiting for an interrupt to happen. Basically the job of the interrupt handler is to service the device & stop it from interrupting. Once the handler resumes, the CPU resumes what it was doing before the interrupt occured

CLASSMATE

PAGE

DATE

- Device Drivers

Device drivers are a device-specific code just for controlling the input/output device that is attached to the computer system. They are the software modules that can be plugged into an OS to handle a particular device. OS takes help from device driver to handle all I/O devices.

Basically, a device driver is a special kind of software program that controls a specific hardware device attached to a computer. They are essential for a computer to work properly. They are necessary to permit a computer to interface and interact with specific drivers. They define the messages & mechanisms whereby the computer (OS & apps) can access the device or make requests for the device to fulfil.

- Device Independent I/O Software.

It is a I/O software in which I/O devices should be accessible to programs without specifying the device in advance. It performs I/O functions common to all devices & provides a uniform interface user-level softwares. Though writing completely device-independent software is difficult, modules that are common among all devices can be written relatively easily.

Some functions: Uniform interfacing, device naming, device security, buffering, storage allocation & reporting of errors.

PAGE

- User space I/O software.

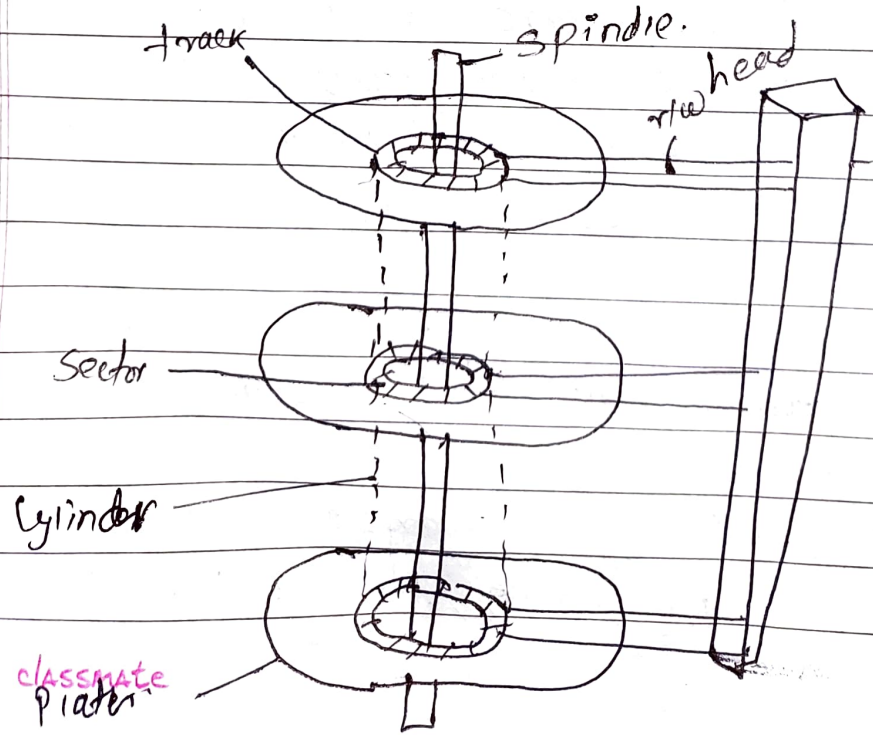
These are the libraries that are linked with the user programs & even whole programs running outside the kernel.

(Disk Management)

* Disk Structure.

In modern computers, most of the secondary storage is in the form of magnetic disks. Hence, knowing the structure of a magnetic disk is necessary to understand how data in the disk is accessed by the computer.

Each modern disk contains concentric tracks and each track is divided into multiple sectors. The disks are usually arranged as a 1D array of blocks, where blocks are the smallest storage unit. Blocks can also be called as sectors.



Note: The same tracks on all the surfaces are cylinder

DATE

Some key terms used in disk structure

- **Seek time :-** Time taken in locating the disk arm to a specified track where the read/write request will be satisfied.

- **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. So, the disk scheduling algorithm that gives minimum rotational latency is better.

- **Transfer time :-** Time to transfer the data. It depends on the rotating speed of the disk & no. of bytes to be transferred.

- **Disk access time**
  Disk access time = Rotational latency + seek time + Transfer time.

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

- **Transmission time.**
  Time taken to access the whole record is called transmission time.

PAGE

classmate

NOR: Here, seek time, controller time & the amount of data to be transferred is not given, we consider all the three terms as 0.

DATE

Some numerical problems:-

Q. Consider a hard disk with: 4 surfaces, 64 tracks/surface, 128 sectors/track, 256 bytes/sector.

1. What is the capacity of the hard disk?

→ Disk capacity = surfaces * tracks/surface * sector/track * bytes/sector

= 4 x 64 x 128 x 256

= 8 388 608 bytes

= 8 MB

2. The disk is rotating at 3600 RPM, what is the data transfer rate?

⇒ Here, 60 sec → 3600 rotations

∴ 1 sec → 60 rotation.

Data transfer rate = no. of rotations per sec. x track capacity x no. of surfaces

= 60 x 128 x 256 x 4

= 7864320 bytes/sec

= 7.5 MB/sec.

3. Disk is rotating at 3600 RPM, what is average access time?

⇒ Average access time = Average rotational delay/latency

Rotational latency = 60 sec = 3600 rotations [1 sec = 60 rot]

∴ Rotation latency = (1/60 sec) = 0.0167 sec = 16.67 msec

∴ Average rotational latency = 16.67 / 2 = 8.33 msec

Average access time = 8.33 msec.

PAGE

DATE

Q. Disk has an average seek time of 5ms, a rotational speed of 15,000 rpm and 500 sectors per track. What is the average access time to read a single sector? What is the expected time to read 500 contiguous sectors on the same track? What is the expected time to read 500 sectors scattered over the disk?

∋ Given,

Average seek time (T_s) = 5ms = 5×10^{-3} sec.

(r) Rotational speed = 15,000 rpm = $\frac{15000}{60}$ = 250 rps

(n) Sectors per track = 500.

∴ Average access time to read a single sector =

$$= T_s + \frac{1}{2r} + \frac{1}{r + \text{Sectors p. track}}$$

$$= 5 \times 10^{-3} + \frac{1}{2 \times 250} + \frac{1}{250 + 500}$$

$$= 7.008 \times 10^{-3} \text{ sec.}$$

$$= 7.008 \text{ ms.}$$

∴ Expected time to read 500 contiguous sectors on the same track

$$= 7.008 + \frac{1}{\text{Rot. speed} \times \text{sectors p. track}} (\text{sector p. track} - 1)$$

$$= 7.008 + \frac{1}{250 \times 500} \times 499$$

$$= 11 \text{ ms.}$$

∴ Time to read 500 sectors scattered over the disk

= Average access time x no. of sector

= 7.008 x 500 = 3504 ms

= 3.504 sec

Note: Access time for n KB block = $Ts + \frac{1}{2r} + \frac{1024 \times n}{r \times N}$

DATE

Q. Consider a disk with a mean seek time of 8ms, a rotational rate of 15,000 rpm, and 262,144 bytes per track. What are the access times for block sizes of 1KB, 2KB and 4KB, respectively?

⇒ Given,
Average seek time (Ts) = 8ms = 8×10^{-3} sec.
Rotational rate (r) = 15,000 rpm = $\frac{15000}{60}$ = 250 rps
No. of bytes per track (N) = 262144

∴ Access time for 1 KB block = $Ts + \frac{1}{2r} + \frac{1024}{r \times N}$

$= 8 \times 10^{-3} + \frac{1}{500} + \frac{1024}{250 \times 262144}$

$= 10.0156$ ms

Access time for 2KB block = $Ts + \frac{1}{2r} + \frac{1024 \times 2}{r \times N}$

$= 8 \times 10^{-3} + \frac{1}{500} + \frac{1024 \times 2}{250 \times 262144}$

$= 10.03125$ ms

Access time for 4KB block = $8 \times 10^{-3} + \frac{1}{500} + \frac{1024 \times 4}{250 \times 262144}$

$= 10.0625$ ms

DATE

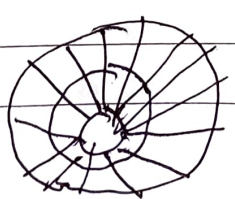
Disk formatting

Preamble | Data | ECC.

It is the configuring process of a data storage media such as a hard drive, floppy disk or flash drive for initial usage. Any existing files on the drive would be erased with disk formatting. It is usually done before initial installation or before installation of a new OS. It is also done if there is a requirement for additional storage in the computer.

Before a disk can store a data, it must be divided into sectors that the disk controller can read and write, called low level formatting. This sector typically consists of preamble, data and ECC (error correcting code). The preamble contains the cylinder & the sector numbers & the ECC contains information that can be used to recover from read error. The size depends upon the manufacturer, depending on reliability.

- Low level formatting fills the disk with a special data structure for each sector.
- Data structure consists of three fields :- header, data area and trailer.
- Header & trailer contain information used by the disk controller.
- Sector no. and ECC contained in header & trailer.
- For writing data ECC is updated & for reading ECC is calculated (read & write are done to a sector)



Direction of disk rotation

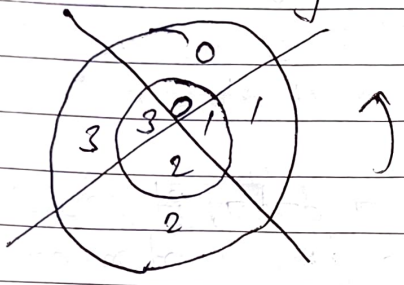
classmate

PAGE

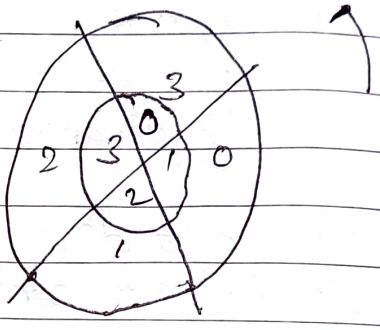
DATE

* Cylinder Skew

If the position of sector 0 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 continuous operation without losing data.



No Skew



Sector Skew

* Interleaving

It is the process through which gaps are placed between two sectors on the platter of a disk. This was done in earlier days when computers were not quick enough to read continuous streams of data. Without interleaving there would be no gap between the sectors & data would arrive immediately before reading unit is ready. Which causes complete rotation of disk would be required again to read the same data. The interleaving ratio was not fixed & can be set by the end user depending on their system specs. Nowdays, the ratio of interleaving is 1:1. ie. no interleaving used.

DATE

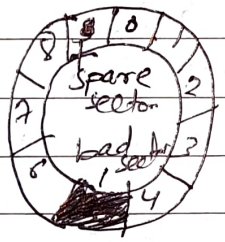
* Error handling

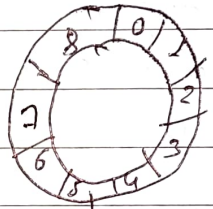
Manufacturing defects introduce bad sectors; if defect is small (say few bits). Then the hard drive is shipped & ECC corrects the error every time the sector is accessed. If error is bigger, it can't be masked.

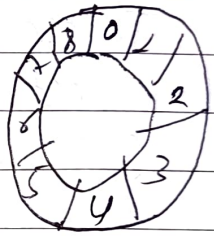
There are 2 ways for error correction.

a) when one of the sectors is bad, the controller remaps between the ~~bad~~ bad & spare sectors.

b) If controller can't remap, the os 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 same data in different places on multiple hard disks to protect data in case of a drive failure. It works by placing data on multiple disks & allowing I/O operations to overlap in a balanced way, improving performance

DATE

- Stripping in RAID: process of dividing large data into multiple disks for faster access
- Mirroring in RAID: Mechanism in which the same data is written to another disk drive
- Parity in RAID: Method used to rebuild data in case of failure of one of the disks
- Hotspares in RAID: It is an extra drive added to the disk array, to increase the fault tolerance

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

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

DATE

* Disk Scheduling:

Disk scheduling is done by OS to schedule I/O requests arriving for the disk. It is also known as I/O scheduling.
A hard disk drive is a collection of plates called platters. Surface of each platter is divided into circular tracks. Each tracks is divided into smaller pieces called sectors. Group of tracks positioned on top of each other form a cylinder. There is a head connected to an arm for each surface, which handles all I/O operations. For each I/O request, first head is selected. It is then moved over the destination track. The disk is then rotated to position the desired sector under the head & finally I/O read/write operation is performed.

There are two objectives of any disk scheduling algo:

- Maximize the throughput : the average no. of requests satisfied per time unit.
- Minimize the seek time : time taken to reach upto desired track.

Some major disk scheduling algorithms are:

1. First Come - First Serve (FCFS)
2. Shortest - Seek Time First (SSTF)
3. Elevator (SCAN)
4. Circular scan (C-SCAN)
5. LOOK
6. C-LOOK

PAGE

DATE

1. First Come-First Serve (FCFS)

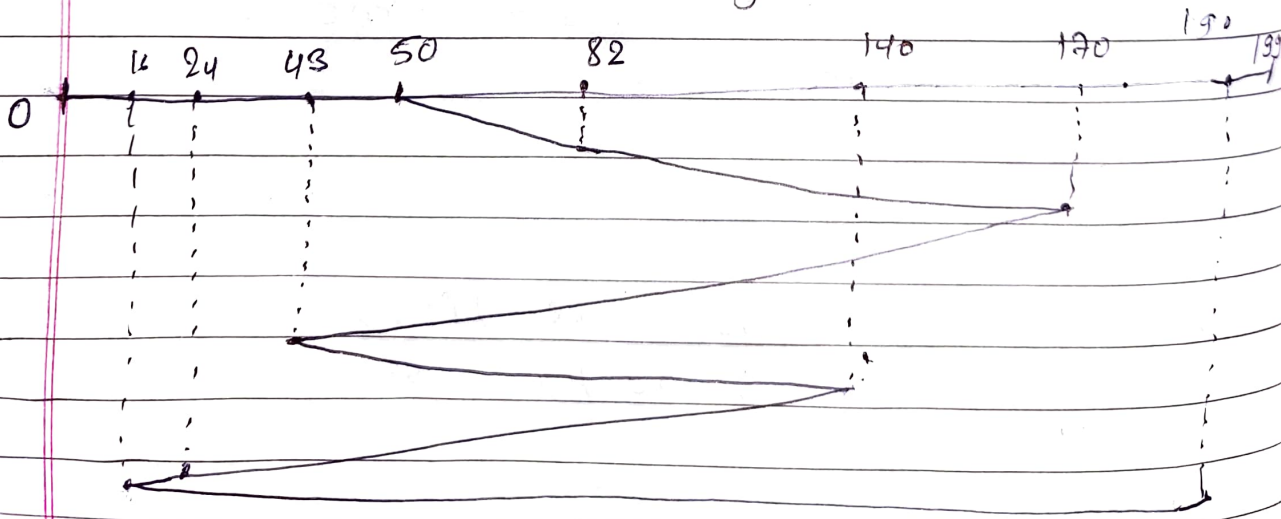
It is the simplest of all disk scheduling algorithms. The requests are addressed in the order they arrive in the disk queue. This aims to minimize seek time/response time with little regard for throughput.

Features
- → Perform operations in order requested
- → no reordering of work queue.
- → no starvation: every request is serviced
- →

Advantages
- → Every request gets fair chance
- → no indefinite postponement

Disadvantages
- → doesn't try to optimize seek time
- → poor performance

Q. A disk contains 200 tracks (0- 199). Request queue contains track no. 82, 170, 43, 140, 24, 16, 190 respectively. Current position of R/W head = 50. Calculate total number of track movements by R/W head.



DATE

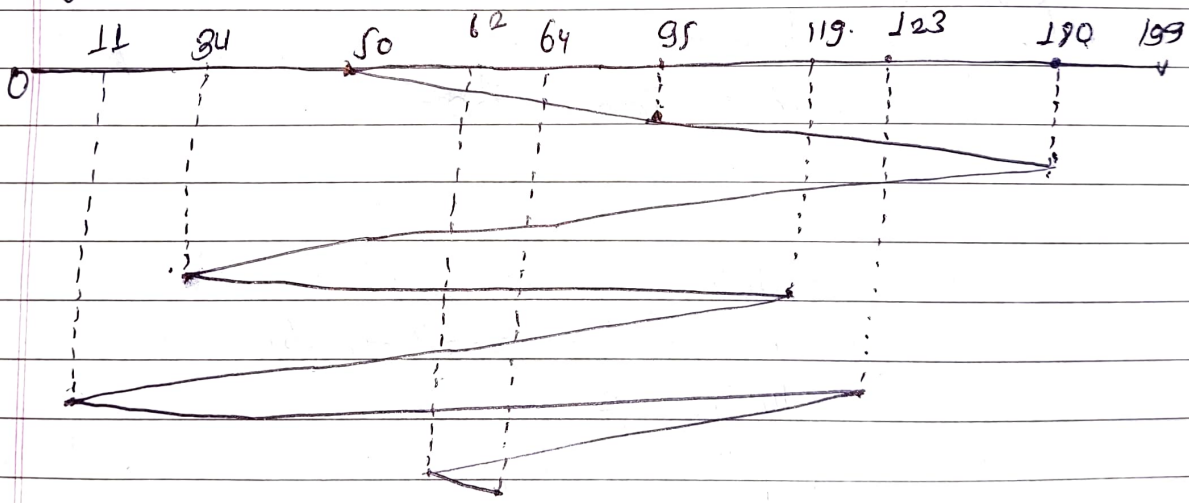
∴ Total disk/head movements or track movements or seek count by R/w head

$= (82-50)+(190-82)+(170-43)+(140-43)+$
$(140-24)+(24-16)+(190-16)$
$= 642$

or

$= (170-50)+(170-43)+(140-43)+(140-16)+(190-16)$
$= 642$

Q. Given the following queue {95, 180, 34, 119, 11, 123, 62, 64} with R/w head at track 50 and the tail track being at 199. Then find total head movement by using FCFS disk scheduling algorithm



∴ Total disk/head movements

$= (95-50)+(180-95)+(180-34)+(119-34)+$
$(119-11)+(123-11)+(123-62)+(64-62)$
$= 644$ tracks

or

$= (180-50)+(180-34)+(119-34)+(119-11)+(123-11)$
$+(123-62)+(64-62)$
$= 644$

classmate

PAGE

DATE

2. 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.

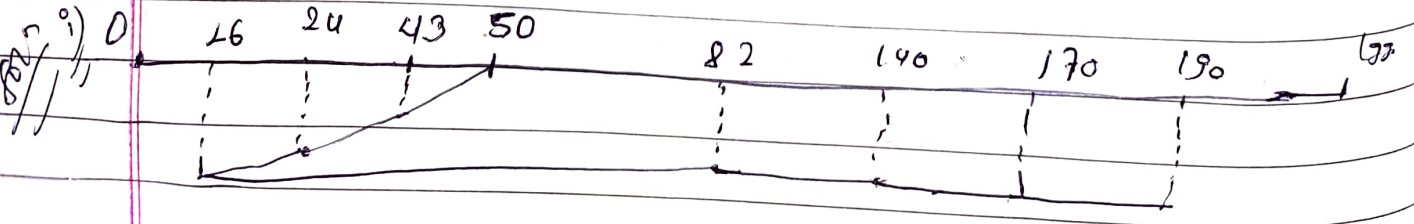
Advantages
- → Improvement over FCFS as it decreases average response time & increases the throughput of system

Disadvantage
- → Overhead to calculate seek time in advance.
- → Can cause starvation for higher seek time requests
- → High variance of response time.

Q. A disk contains 200 tracks (0-199). Request queue contains track no. 82, 170, 43, 140, 24, 16, 190 respectively. Current position of R/W head is 50.

i) Calculate total no of tracks movement by R/W head using SSTF
ii) If R/W head takes 1 ms to move from one track to another then what is total time taken?

Sol: i)



Note: If tie occurs in seek time, for SSTF, then use FCFS

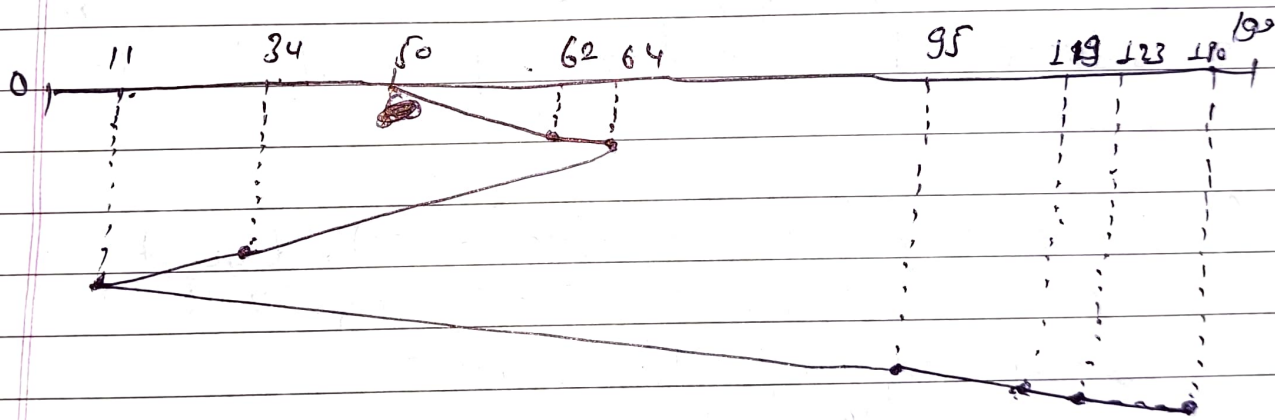
DATE

∴ Total track movement = (50 - 16) + (190 - 16)
= 34 + 174
= 208 tracks.

And, total time taken = 208 x 1 ns = 208 ns

Q Given the following queue {95, 180, 34, 119, 11, 123, 62, 64} with the R/W head initially at the track 50 and the tail track being at 199. Then find the total head movement using SSTF disk scheduling algorithm

A)



∴ Total disk movement = (64 - 50) + (64 - 11) + (180 - 11)
= 236 tracks

Note: Response time = Seek time.

DATE

3. Elevator (SCAN) (Note: It goes to end of disk in only one direction only)

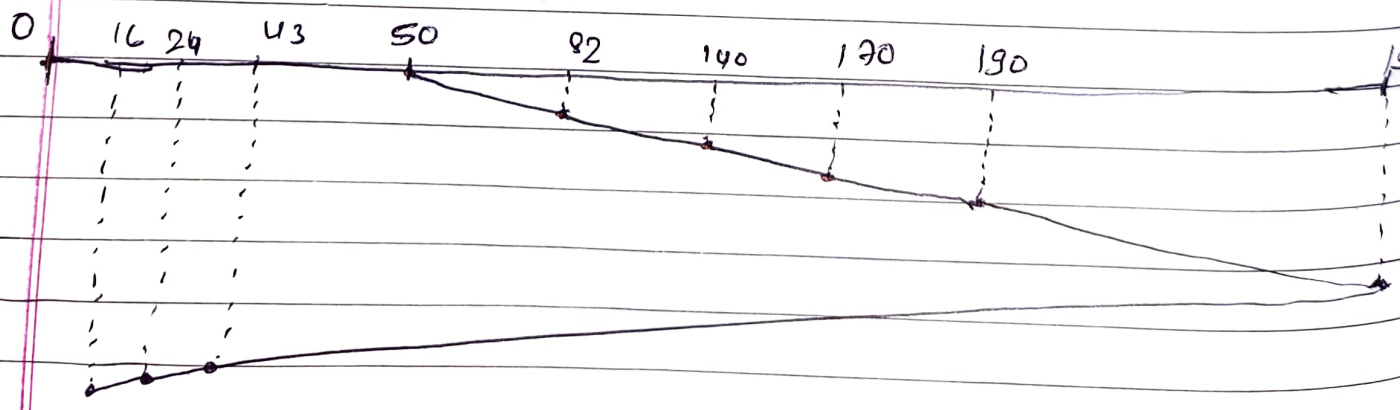
In SCAN algorithm the disk arm moves into a particular direction & services the requests coming in its path & after reaching the end of disk, it reverses its direction & again services the request arriving in its path. So, it works like an elevator. As a result, requests at the midrange are serviced more & those arriving behind the disk arm will have to wait.

Advantages	Disadvantages
→ High throughput	→ Long waiting time for requests for locations just visited by disk arm.
→ Low variance of response time	
→ Average response time.	

Q. A disk contains 200 tracks (0-199). Request queue contains track no. 82, 170, 43, 140, 24, 16, 190 respectively. Current position of R/W head = 50 (Direction left (right to left) or right (left to right))

i) Calculate the total no of track movements by R/W head using SCAN

Let's use direction right (left to right) i.e. to larger direction



∴ Total track movement = (199 - 50) + (199 - 16)
= 332 tracks

classmate

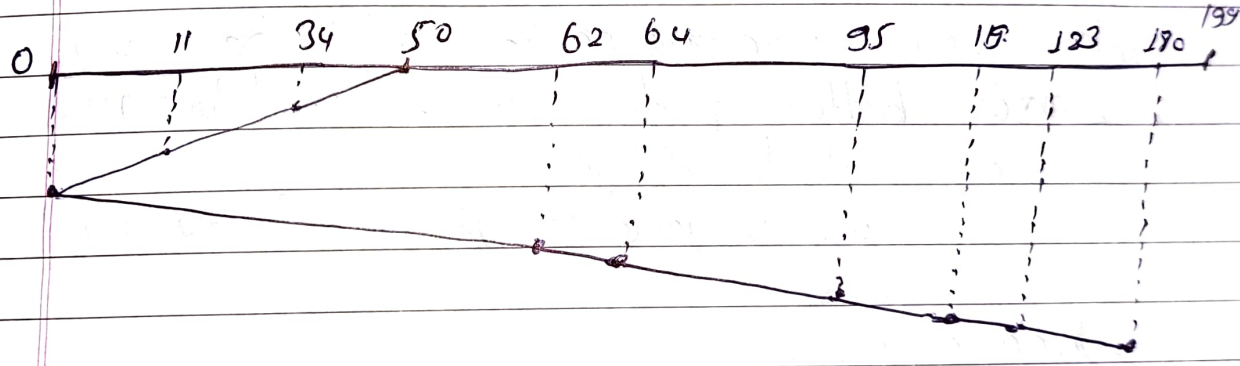
PAGE

DATE

9) If R/W head takes 1ns to move from one track to another then total time taken is?

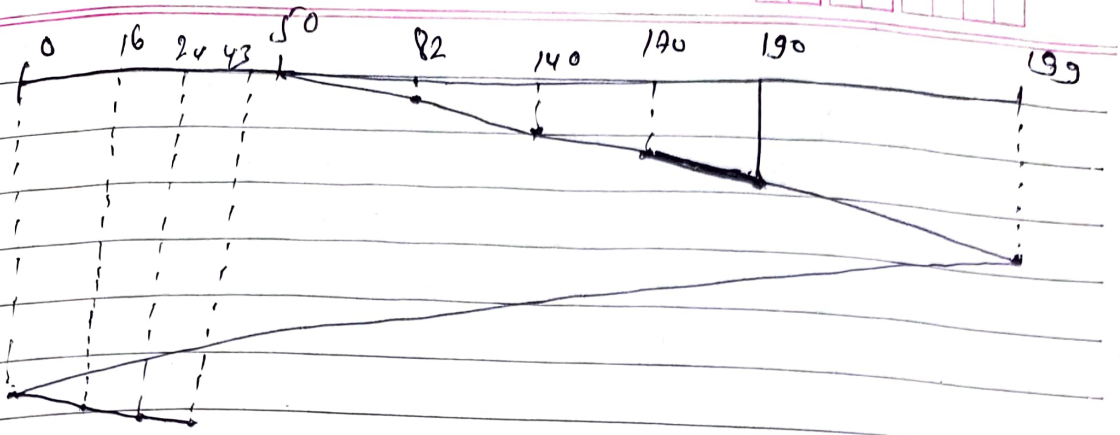
⇒ Total time taken = 332 × 1 ns
= 332 ns

Q Given the following queue {95, 180, 34, 119, 11, 123, 62, 64} with the R/W head initially at track 50 and the trail track being at 199. Find total head movement by using SCAN disk sch. alg.
⇒ Direction (left)



Total disk movements: (50-0) + (180-0)
= 230 tracks

PAGE



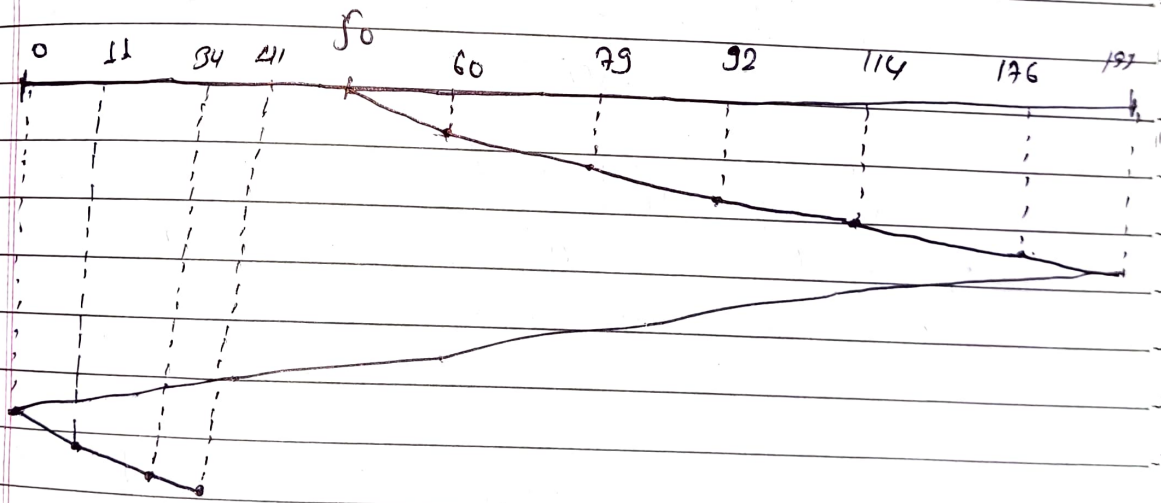
Total movement = (199-50) + (199-0) + (43-0)

or Seek count = 391 tracks

Q. Request sequence = {176, 79, 34, 60, 92, 11, 41, 114}

Initial head position = 50

Direction = right (left to right)



∴ Total Seek count = (199-50) + (199-0) + (41-0)

= 389 tracks

4) Circular SCAN (C-SCAN)

In 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 pending at the ~~scanned area~~ other end or there may be zero or few requests pending at the scanned area. These situations are avoided in CSCAN algorithm.

The disk arm instead of reversing its direction goes to the other end of the disk & starts servicing the requests from there.

Advantage.

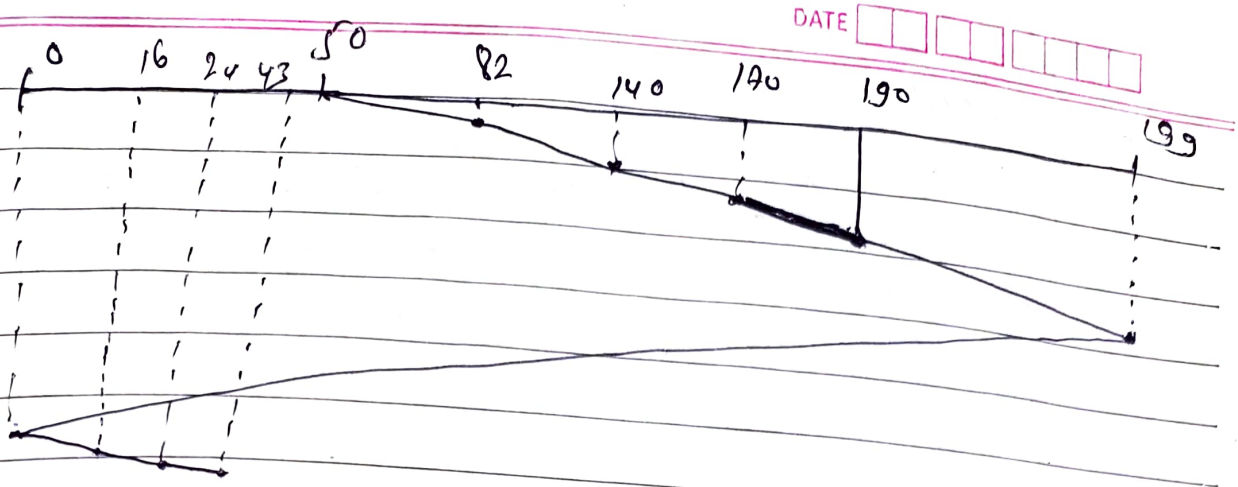
→ Waiting time for cylinders just visited by the head is reduced as compared to SCAN algo.
→ Provides uniform waiting time & better response time.

Disadvantage

→ Causes more seek movements as compared to SCAN
→ Causes the head to move till the end of the disk even if there are no requests to be serviced.

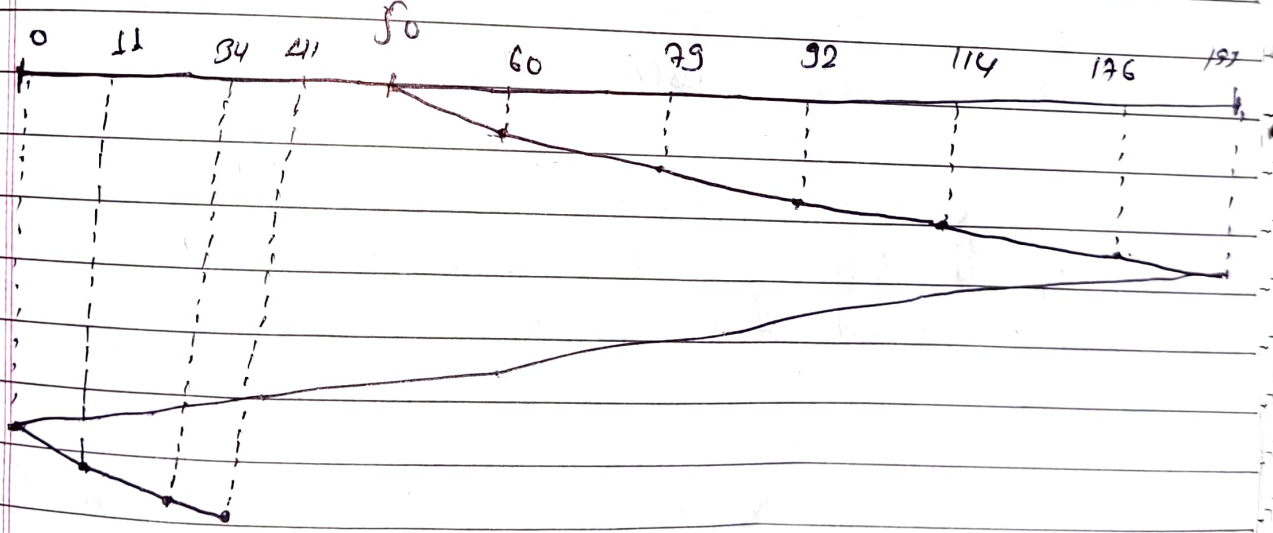
Q. A disk contains 200 tracks (0-199). Request queue contains track no. 82, 170, 43, 140, 24, 16, 190 respectively. Current position of head is 50. Calculate total no. of head movements by R/W head using C-SCAN.

→ Direction is towards large value (right)



Total movement = (199-50) + (199-0) + (43-0)
or Total Seek count = 391 tracks

Q Request sequence = {176, 79, 34, 60, 92, 11, 41, 114}
Initial head position = 50
Direction = right (left to right)



∴ Total Seek count = (199-50) + (199-0) + (41-0)
= 389 tracks

5. LOOK

It is similar to the SCAN disk scheduling algorithm except the difference is that the disk arm in spite of going to the end of the disk goes only to last request to be serviced infront of the head & then reverses its direction from there only. Thus it prevents the extra delay which occured due to unnecessary traversal to the end of the disk.

Advantages

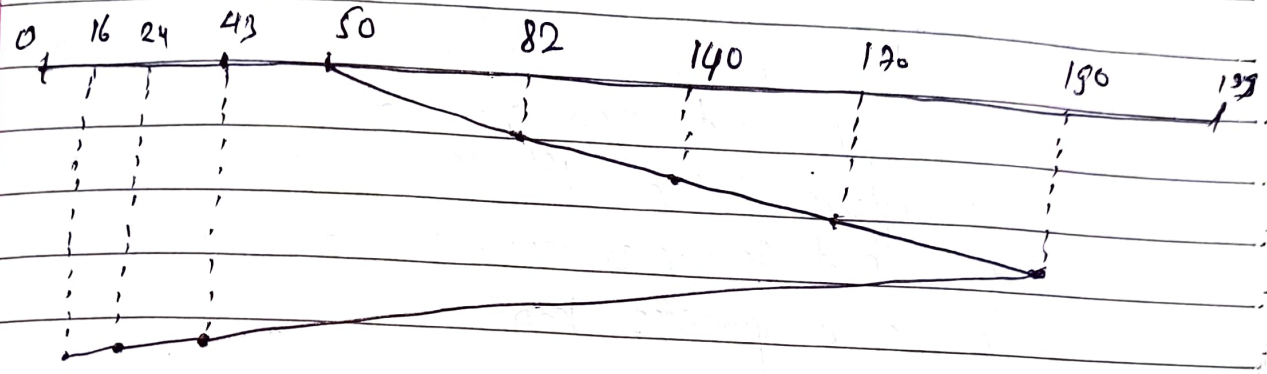
- Doesn't cause the head to move till the ends of the disk when there are no requests to be serviced.
- Provides better performance compared to SCAN.
- Doesn't lead to starvation.
- Provides low variance in response time & waiting time.

Disadvantages.

- Overhead of finding the end requests.
- Causes long waiting time for the cylinders just visited by the head

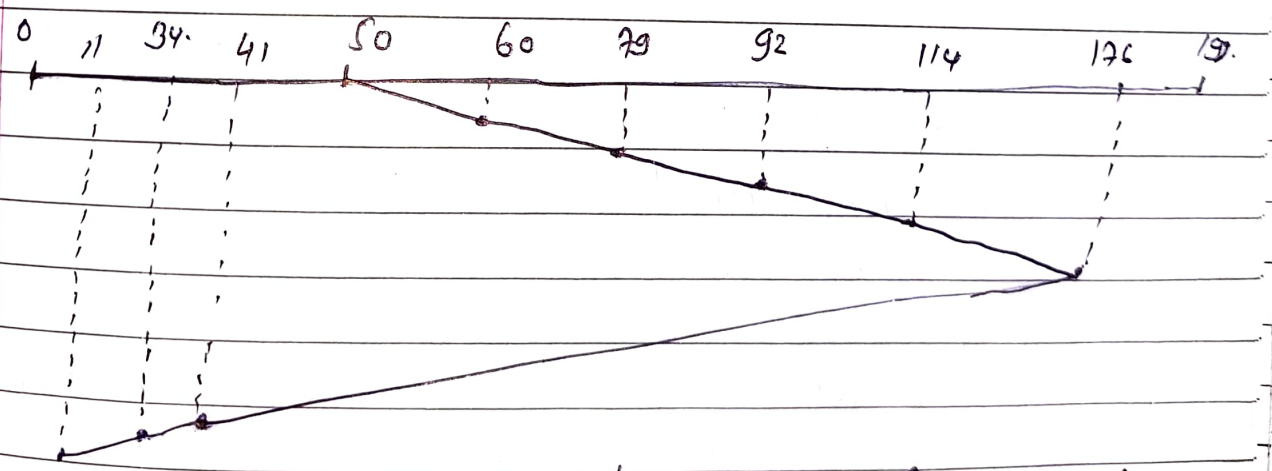
DATE

Q A disk contains 200 tracks (0-199). Request queue contains track no. 82, 190, 43, 140, 24, 16, 190 respectively. Current position of R/W head is 50. Calculate total no. of tracks movement by R/W head. Using look disk scheduling algorithm. (direction right)



∴ Total no. of track movements = (190-50) + (190-16)
= 314

Q. Request sequence = {176, 79, 34, 60, 92, 11, 41, 114}
Initial head position = 50
Direction: right



Total seek count / track mov. = (176-50) + (176-11)
= 291 tracks

classmate

PAGE

DATE

6. C-Look

Circular-look algorithm is an improved version of Look algorithm. Head starts from the 1st request at the one end of the disk & moves towards the last request at the other end servicing all the requests in between. After reaching the last request at the other end, head reverses its direction. It then returns to the 1st request at the starting end without servicing any requests in between. The same process repeat.

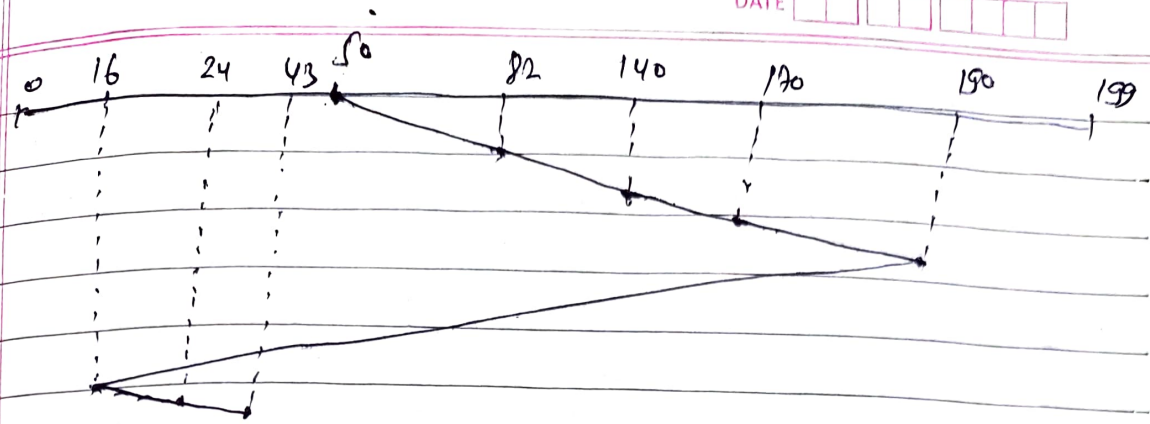
Advantages

→ Doesn't cause the head to move till the end of the disk when there are no requests to be serviced
→ reduces waiting time for cylinders just visited by head.
→ provides better performance compared to Look.
→ Doesn't lead to starvation
→ provides low variance in response & waiting time

Disadvantages

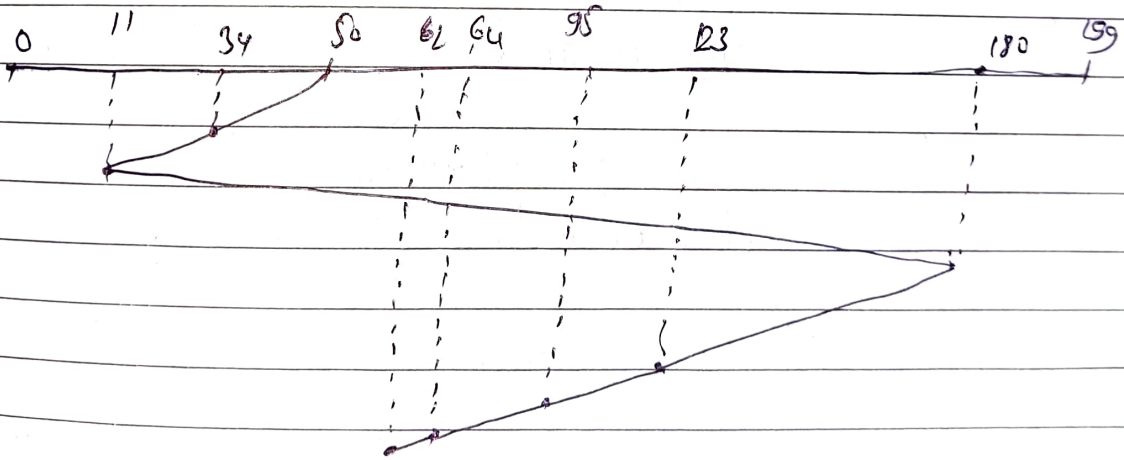
→ There is an overhead of finding the end requests

Q. A disk contains 200 tracks (0-199) request queue contains track no. 82, 170, 43, 140, 24, 16, 190 respectively. Current position of R/W head = 50. Calculate total no of tracks movement by R/W head using CLOOK. Direction (right) (towards large value)



∴ Total no of track movements = (190-50) + (190-16) + (43-16)
= 341 tracks.

Q. Given the following queue {95, 180, 34, 119, 11, 123, 62, 64} with the R/W head initially at track 50 & the tail track being at 199. Find total head movement using C-LOOK. (Direction (left))



Total disk movement/head/seek = (50-11) + (180-11) + (180-62)
= 326 tracks

Questions asked from this chapter

Q. Discuss the concept of interleaving of disk. Why is it important? Explain with suitable example. (2078 - 5 marks)

Q. What is the main objective of disk scheduling algorithms? Why SSTF is not practically feasible? Assume that we have disk with 100 tracks & currently head is at track number 35. What will be the seek time for the algorithms SCAN and LOOK for processing IO requests queue: 52, 67, 27, 11, 43, 85, 18, 75, 92, 8 ? (2076 - 10 marks)

Q. What is direct memory access? Discuss its working principle. (2075 - 5 marks) (2075 - 5 marks) (2071 - 5 marks) Write its advantages. (said)

Q. What do you mean by interrupt? Explain the working mechanism of interrupt controller. (2073 - 5 marks, 2066 - 5 marks) What is the advantage of interrupt over polling (2074 - 5 marks)

Q. How do you manage free disk space? (2076 - 5 marks)

Q. What do you mean by disk management? Explain the error handling & formatting operation on the disk. (2073 - 6 marks)

Q. When programmed IO is suitable than other IO handling techniques? Explain the process of IO handling using DMA (2076 - 5 marks)