Unit

9

I/O System

9.1 I/O Hardware

9.2 Application of I/O Interface

9.3 Kernel I/O Subsystem

9.4 Disk Scheduling Algorithm

Introduction :

The Management of I/O devices is a very important part of the operating system, so important and so varied that entire I/O subsystems are devoted to its operation.An I/O interface is required whenever the I/O device is driven by the processor. The interface must have necessary logic to interpret the device address generated by the processor.Device drivers are modules that can be plugged into an OS to handle a particular device or category of similar devices. The basic mechanism of interrupt enables the CPU to respond to an asynchronous event, such as when a device controller becomes ready for service.

9.1 I/O Hardware :

I/O devices can be roughly categorized as storage, communications, user-interface, and other devices communicate with the computer via signals sent over wires or through the air. Devices connect with the computer via ports, e.g. a serial or parallel port. A common set of wires connecting multiple devices is termed a bus. Buses include rigid protocols for the types of messages that can be sent across the bus and the procedures for resolving contention issues.

9.1.1 Bus Types in I/O :

Figure below illustrates three of the four bus types commonly found in a modern PC. The PCI bus connects high-speed, high-bandwidth, devices to the memory subsystem (and the CPU.)The expansion bus connects slower low-bandwidth devices, which

Scanned by CamScanner

protection reasons. (I.e. DMA is a kernel-mode operation.)

9.2 Application of I/O Interface :

Application I/O Interface represents the structuring techniques and interfaces for the operating system to enable I/O devices to be treated in a standard, uniform way. The actual differences lies kernel level modules called device drivers which are custom tailored to corresponding devices but show one of the standard interfaces to applications.

The purpose of the device-driver layer is to hide the differences among device controllers from the I/O subsystem of the kernel, such as the I/O system calls. Following are the characteristics of I/O interfaces with respect to devices.

1) **Character stream / block :**

   A character stream device transfers bytes in one by one fashion whereas a block device transfers a complete unit of bytes.

2) **Sequential / random access :**

   A sequential device transfers data in a fixed order determined by the device, random-access device can be instructed to seek position to any of the available data storage locations.

3) **Synchronous / asynchronous :**

   A synchronous device performs data transfers with known response time where as an asynchronous device shows irregular or unpredictable response time.

Scanned by CamScanner

4) **Sharable / dedicated :**

A sharable device can be used concurrently by several processes or threads but a dedicated device cannot be used so.

5) **Speed of Operation :**

Device speeds may range from a few bytes per second to a few gigabytes per second.

6) **Read-write, Read only, or Writes Only :**

Some devices perform both input and output, while others support only one data direction that is read only.

9.3 Kernel I/O Subsystem :

Kernel I/O Subsystem is responsible to provide many services related to I/O. Following diagram shows the position of the kernel I/O subsystem :

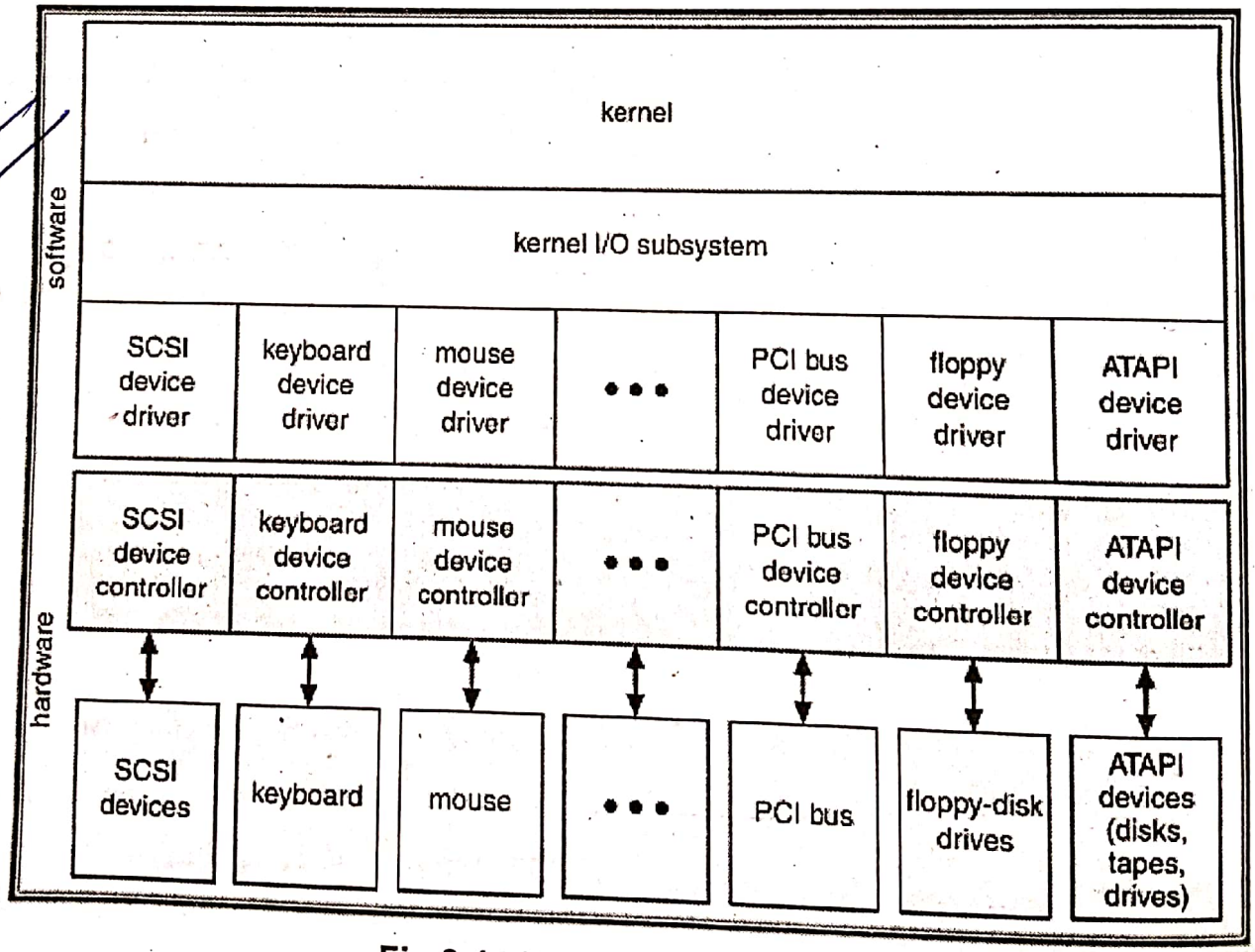


Fig 9.4 : Kernel I/O Structure

A) Service provided by the Kernel I/O Subsystem :

Following are some of the services provided by the kernel I/O subsystem :

Scanned by CamScanner

4m Kernel I/O subsystem:-

Kernel provides many services related to I/O several services like buffering, caching, spooling, device reservation, error handling are provided by kernel I/O subsystem.

① I/O scheduling:-

- I/O scheduling means to determine the order in which to execute the I/O
- I/O request are rarely executed in the order in which they are received
- scheduling can improve overall system performance & reduce AWT for I/O to complete.
- The scheduler rearranged the order of queues in the queue to improve the overall system efficiency (average response time experienced by App^n

② Buffering:-

- Kernel I/O subsystem maintains a memory area known as buffer
- A buffer is the memory area that store data while they are transferred betn two devices.

There are various reason why buffering needs to be done

One reason is to cope with a speed mismatch betn the producer & consumer of data.

Second use of buffering is to adapt betn devices that have different data transfer sizes.

Scanned by CamScanner

example 1) A file is being transferred to a disk via modem. The hard disk is much faster than the modem so to mask this difference a buffer is created in main memory to accumulate bytes received from modem.

example 2) In Networking when fragmentation & reassembly of messages is done. At the sending site a large message is fragmented in to small packets and receiving site places them in a reassembly buffer to form an image of source data

(3) Caching:

- Refers maintains ~~cache~~ cache memory which is the region of fast memory that holds copies of data
- Access to the catched copy is more efficient than the access to original

eg The instruction of currently running processes are stored on disk, catched in physical memory & copied again in CPU's secondary & primary cache

- The difference bet buffer & cache is that a buffer may hold the existing copy of a data item.
where a ~~th~~ cache just hold a copy on faster storage of an item that resides elsewhere
- Caching & buffering are distinct fun. but sometimes a region of memory can be used for both the purposes.

Scanned by CamScanner

(a) Spooling & Device Reservation:-

- Spool is buffer that holds output for a device, such as printer that cannot accept interleaved data streams
- The spooling system copies the queued spool files to the ~~the~~ printer one at a time. In some operating system spooling is managed by system daemon process. In other OS it is handled by an in kernel thread

eg The printer can print only one job at a time but more than one may want to print concurrently at the same time but output should not be mixed together
- The OS solves this problem by intercepting all o/p to the printer each appln output spooled to the disk file & spooling system maintains the queue of such spooled files
- When printer finishes the job it is doing the spooling system assigns next job from the queue to the printer
- The OS provides control interface that enables users & system Administrator to view the queue; to remove unwanted job before they are printed & to suspend printing while the printer is serviced

(b) Error Handling:-

An OS that uses protected memory can guard against many kind of hardware & application error so complete system failure is not the usual result of any minor prob

Scanned by CamScanner

MAHAGURU | Page No. | Date

(2) devices & I/O transfer can fail in many ways either for transient reasons such as network becoming overloaded or for permanent reasons such as disk controller becoming defective.

for eg read() failure results in read() retry. and Network send() failure results in retend()

- Generally an I/O system call will return 1 bit information about the status of the call signifying either success or failure
- Some system use addition bits to give more detailed information of the error occured & type of error etc

Scanned by CamScanner

1) **Scheduling :**

Kernel schedules a set of I/O requests to determine a good order in which to execute them. When an application issues a blocking I/O system call, the request is placed on the queue for that device. The Kernel I/O scheduler rearranges the order of the queue to improve the overall system efficiency and the average response time experienced by the applications.

2) **Buffering :**

Kernel I/O Subsystem maintains a memory area known as buffer that stores data while they are transferred between two devices or between devices with an application operation. Buffering is done to cope with a speed mismatch between the producer and consumer of a data stream or to adapt between devices that have different data transfer sizes.

3) **Caching :**

Kernel maintains cache memory which is region of fast memory that holds copies of data. Access to the cached copy is more efficient than access to the original.

4) **Spooling and Device Reservation :**

A spool is a buffer that holds output for a device, such as a printer, that cannot accept interleaved data streams. The spooling system copies the queued spool files to the printer one at a time. In some operating systems, spooling is managed by a system daemon process. In other operating systems, it is handled by an in kernel thread.

5) **Error Handling :**

An operating system that uses protected memory can guard against many kinds of hardware and application errors.

9.4 Disk Scheduling Algorithm :

For a multiprogramming system with many processes, the disk queue may often have several pending requests. Thus, when one request is completed, the operating system chooses which pending request to service next. How does the operating system make this choice? Any one of several disk-scheduling algorithms can be used as solution to this situation. Following are types of disk scheduling algorithms which are used in multi-programming system :

Scanned by CamScanner

requests.

Solved Problems (Disk Scheduling Algorithms) :

Example :

Given the queue - 95, 180, 34, 119, 11, 123, 62, 64 with the Read-write head initially at the track 50 and the tail track being at 199, calculate total head movements using all Disk scheduling algorithms.

Solution :

a) **FCFS:**

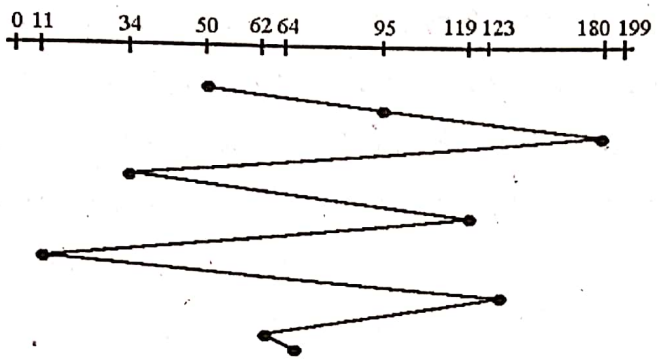


Fig 9.10 : FCFS

To determine the number of head movements, simply find the number of tracks it took to move from one request to the next. For this case it went from 50 to 95 to 180 and so on. From 50 to 95 it moved 45 tracks. It had a total head movement of 640 tracks.

b) **SSTF :**

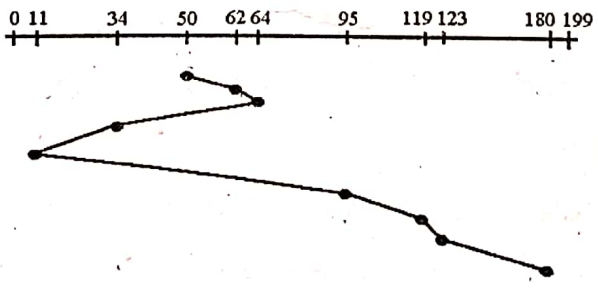


Fig 9.11 : SSTF

Scanned by CamScanner

Starting at 50, the next shortest distance would be 62 instead of 34 since, it is only 12 tracks away from 62 and 16 tracks away from 34. The process would continue until all the process is taken care of. The next case would be to move from 62 to 64 instead of 34 since there are only 2 tracks between them and not 18 if it were go to the other way. Although, this seems to be a better service being that it moved a total of 236 tracks.

c) SCAN :

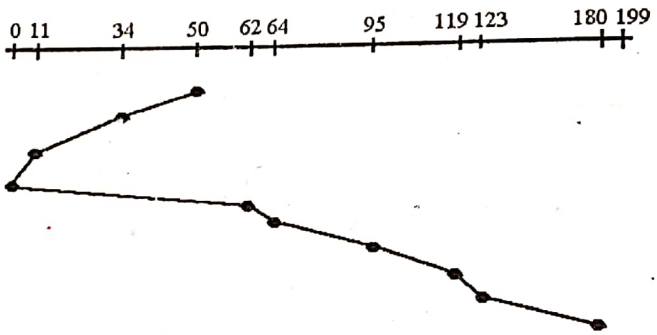


Fig 9.12 : SCAN

This approach works like an elevator does. It scans down towards the nearest end and then when it hits the bottom it scans up servicing the requests that it didn't get going down. If a request comes in after it has been scanned it will not be serviced until the process comes back down or moves back up. This process moved a total of 230 tracks.

d) C-SCAN :

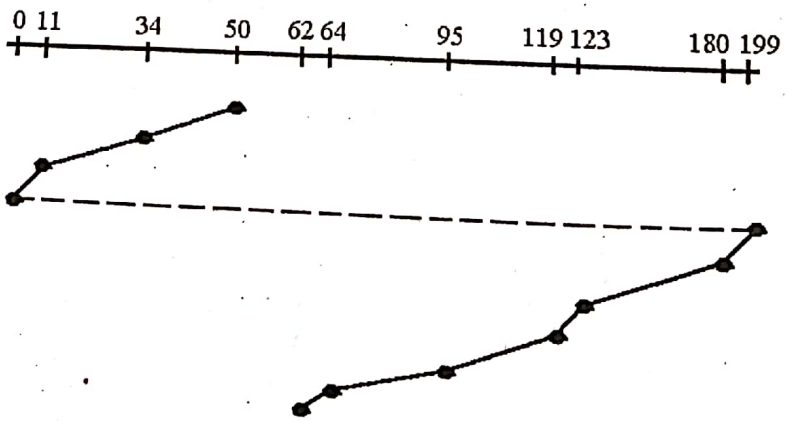


Fig 9.13 : C-SCAN

Scanned by CamScanner

The total head movement for this algorithm is only 187 tracks.

e) **C-LOOK :**

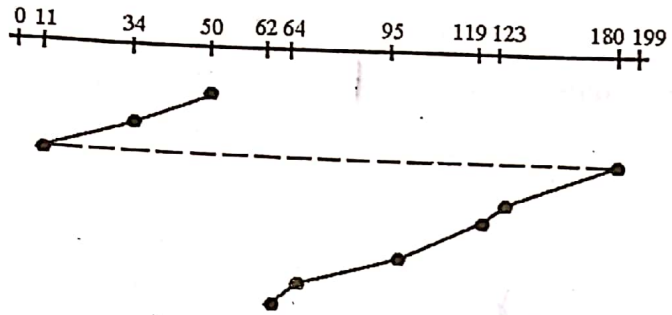


Fig 9.14 : C-LOOK

In this the scanning doesn't go past the last request in the direction that it is moving. It too jumps to the other end but not all the way to the end. Just to the furthest request. C-SCAN had a total movement of 187 but this scan (C-LOOK) reduced it down to 157 tracks.

Example 2 :

Given the Work Queue: 23, 89, 132, 42, 187 and there are 200 cylinders numbered from 0 – 199. The disk head stars at number 100. Find the total number of head movements using FCFS, SSTF, SCAN, C-SCAN, LOOK and C-LOOK algorithms.

Solution :

a) **FCFS :**

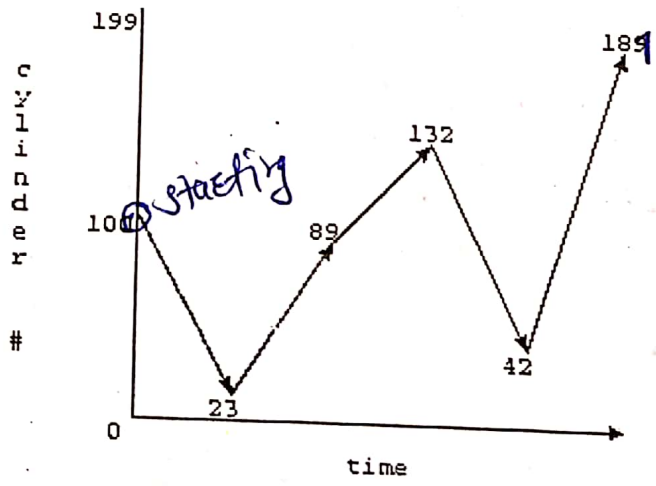


Fig 9.15 : FCFS

1) Moving from 100 to 23 requires movement over 77 cylinders.
2) Then from 23 to 89 requires movement over 66 cylinders.

Scanned by CamScanner

3) Now from 89 to 132 requires movement over 42 cylinders.
4) From 132 to 42 requires movement over 90 cylinders.
5) And at last from 42 to 187, requires movement over 145 cylinders.
6) Thus total head movements are 420.

b) **SSTF :**

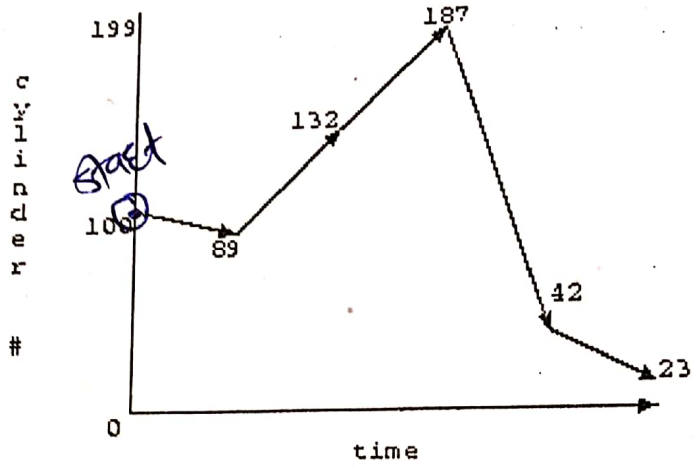


Fig 9.16 : SSTF

1) Moving from 100 to 89 requires movement over 11 cylinders.
2) Then from 89 to 132 requires movement over 43 cylinders.
3) Now from 132 to 187 requires movement over 55 cylinders.
4) From 187 to 42 requires movement over 145 cylinders.
5) And at last from 42 to 23, requires movement over 19 cylinders.
6) Thus total head movements are 418.

c) **SCAN :**

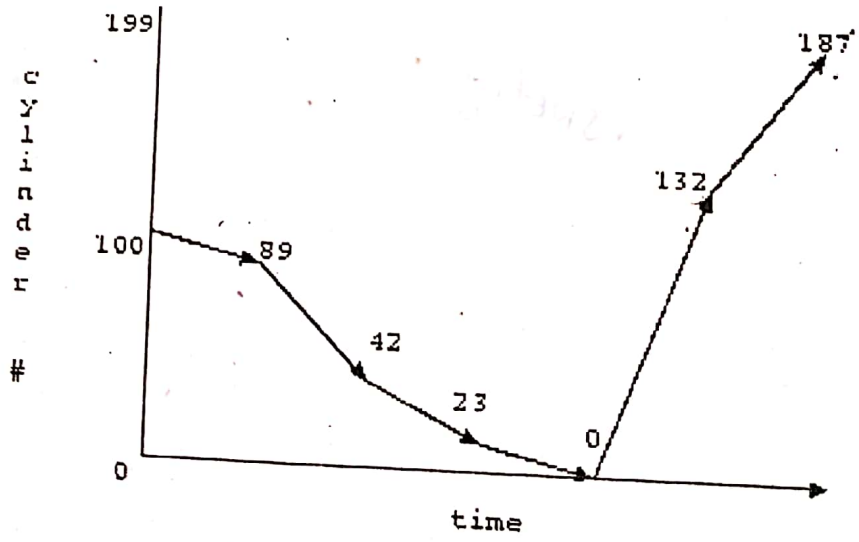


Fig 9.17 : SCAN

Scanned by CamScanner

1) Moving from 100 to 89 requires movement over 11 cylinders.
2) Then from 89 to 42 requires movement over 47 cylinders.
3) Now from 42 to 23 requires movement over 19 cylinders.
4) From 23 to 0 requires movement over 23 cylinders.
5) From 0 to 132, requires movement over 132 cylinders.
6) And at last, from 132 to 187 requires movement over 55 cylinders.
7) Thus total head movements are 287.

d) LOOK :

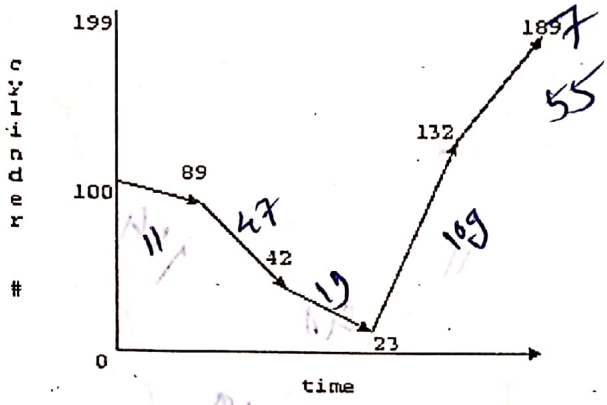


Fig 9.18 : LOOK

1) Moving from 100 to 89 requires movement over 11 cylinders.
2) Then from 89 to 42 requires movement over 47 cylinders.
3) Now from 42 to 23 requires movement over 19 cylinders.
4) From 23 to 132 requires movement over 109 cylinders.
5) And at last from 132 to 187, requires movement over 55 cylinders.
6) Thus total head movements are 241.

e) C-SCAN :

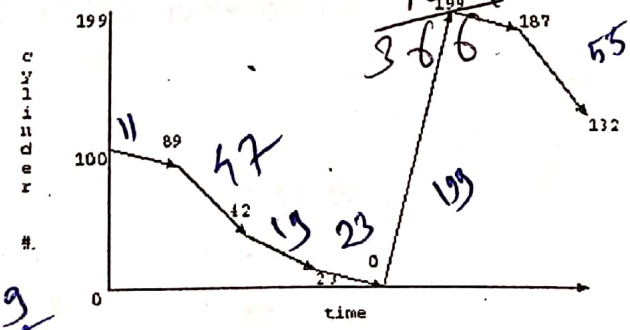


Fig 9.19 : C-SCAN

Scanned by CamScanner

1) Moving from 100 to 89 requires movement over 11 cylinders.
2) Then from 89 to 42 requires movement over 47 cylinders.
3) Now from 42 to 23 requires movement over 19 cylinders.
4) From 23 to 0 requires movement over 23 cylinders.
5) From 0 to 199, requires movement over 199 cylinders.
6) From 199 to 187 requires movement over 12 cylinders.
7) And at last from 187 to 132 requires movement over 55 cylinders.
8) Thus total head movements are 366.

f) **C-LOOK :**

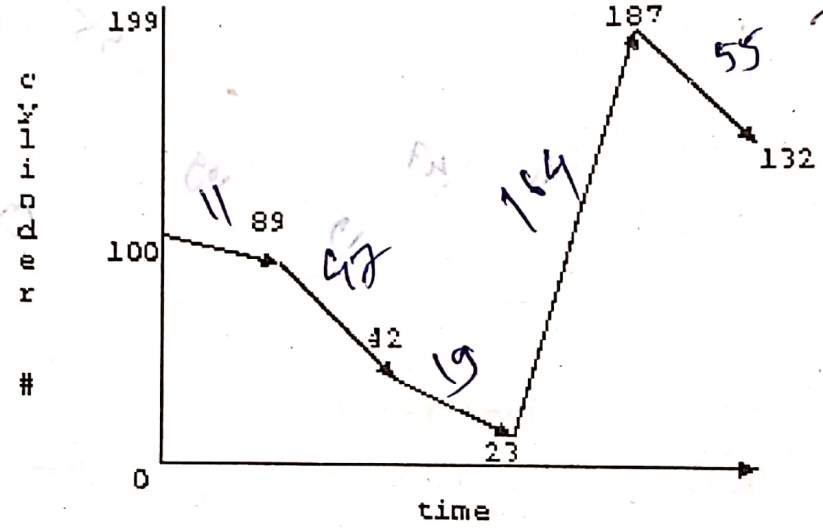


Fig 9.20 : C-LOOK

1) Moving from 100 to 89 requires movement over 11 cylinders.
2) Then from 89 to 42 requires movement over 47 cylinders.
3) Now from 42 to 23 requires movement over 19 cylinders.
4) From 23 to 187 requires movement over 164 cylinders.
5) And at last from 187 to 132, requires movement over 55 cylinders.
6) Thus total head movements are 296.

Scanned by CamScanner