CHAPTER 6: I/O Management & Disk Scheduling

- Principles of I/O Hardware
- Principles of I/O software
- I/O software Layer
- Disk
  - Hardware
  - Formatting
  - Arm scheduling
  - Error handling
  - Stable Storage

Introduction

- Controlling and managing I/O devices and I/O operations is one of the main responsibilities of an operating system.
- OS must issue commands to the devices to work, provide a device independent interfaces between devices and the rest of the system, handle errors or exceptions, catch interrupts and so on.

Principles of I/O Hardware

- A Computer communicates with variety of I/O devices ranging from mouse, keyboard and disk to highly specialized devices like fighter plane's flying controls.
- However, one needs to digest only a few concepts to understand how operating systems facilitate deviceindependent communication with I/O devices.

I/O devices

- I/O devices can be roughly divided into
  1. Block Devices.
  2. Character Devices

1. Block Devices:

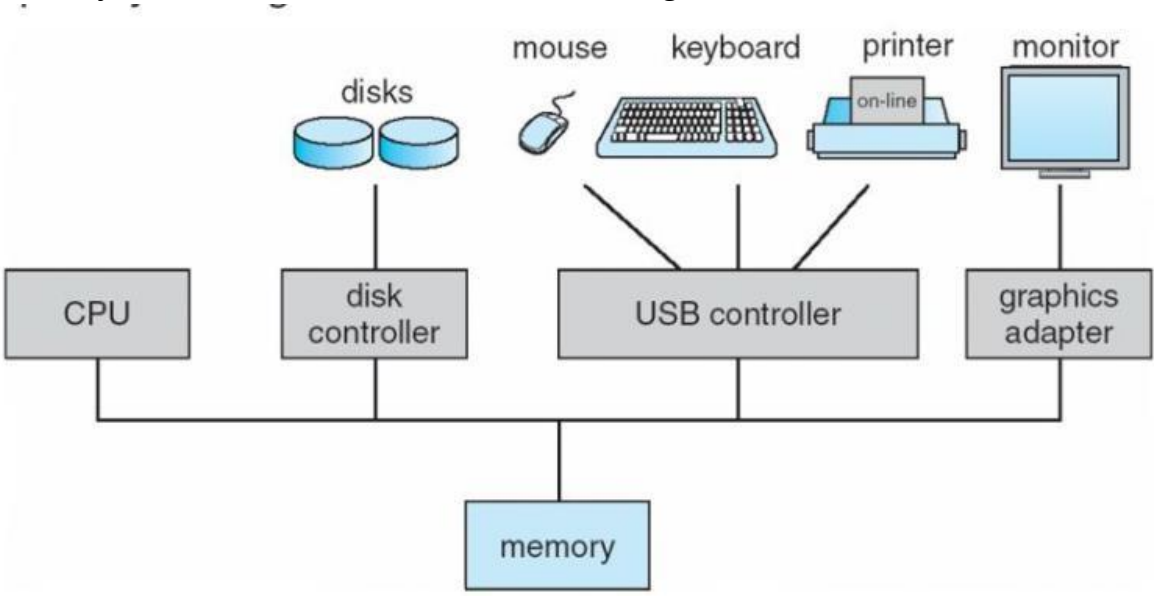
- Stores information in a fixed sized block, each one with its own address.
- common block size ranges from 512 bytes to 32,768 bytes
- data transfer takes place in blocks
- block addressable not byte addressable
- Examples: Hard disks, CD-ROMs, USB sticks etc.
- Application can interact with Block devices through block-device interface, which supports following basic system call:
  - ✓ read() – to read from device.
  - ✓ write() – to write to device.
  - ✓ seek() – to specify next block to be accessed.

2. Character Devices:

- Delivers or accepts a stream of characters, without regard to any block structure.
- Unlike block devices character devices are not addressable.
- The data transfer to/from them is performed in units of bytes.
- Examples: - Keyboards, printers, network interfaces, mice etc.
- Application can interact with Character devices through character-stream interface, which supports following basic system call:
  - ✓ get() – to read character from device.
  - ✓ put() – to write a character to device.

Device Controller

- I/O devices typically consists of two components: Electrical and Mechanical
- The electronic component is called the device controller or the adapter
- A device controller is a part of a computer system that makes sense of the signals going to, and coming from the CPU
- There are many device controllers in a computer system
- Any device connected to the computer is connected by a plug and socket, and the socket is connected to a device controller
- In personal computer, device controller usually takes the form of a chip on the parent board.
- Many controllers can handle two, four or even eight identical devices



- Device controllers is a piece of hardware that receives commands from the system bus, translates them into device actions and reads/writes the data onto the system bus
- Each controller has a few registers that are used for communicating with the CPU (control registers) along with a data buffer.
- By writing into these registers, the operating system can command the device to deliver data, accept data, switch itself on or off, or otherwise perform some action.

- By reading from these registers, the operating system can learn what the device's state is, whether it is prepared to accept a new command, and so on.
- In addition to the control registers, many devices have a data buffer that the operating system can read and write.

Principles of I/O software

- Hardware and software are like two sides of coin: one is unusable without other.
- Thus, in this section we will first discuss about the goals of I/O software and then ways to perform I/O.

Goals of I/O software

Followings are the goals of I/O software:

1) Device independence:

- It should be possible to write programs that can access any I/O device without having to specify the device in advance.
- For example, a program that reads a file as input should be able to read a file on a hard disk, a CD-ROM, a DVD, or a USB stick without having to modify the program for each different device.

2) Uniform naming:

- The name of the device should simply be a string or an integer and do not depend on the device in any way.

3) Error Handling:

- In general, the error should be handled as close to the hardware as possible Propagate errors up only when lower layer cannot handle it.
- For example, if controller discovers read error, it should try to correct the error itself. If it cannot, the device driver should handle it.

4) Synchronous (blocking) Vs. Asynchronous (interrupt driven) transfers:

- It is up to the OS to make the operation that are interrupt driven look blocking to the user programs.

5) Buffering:

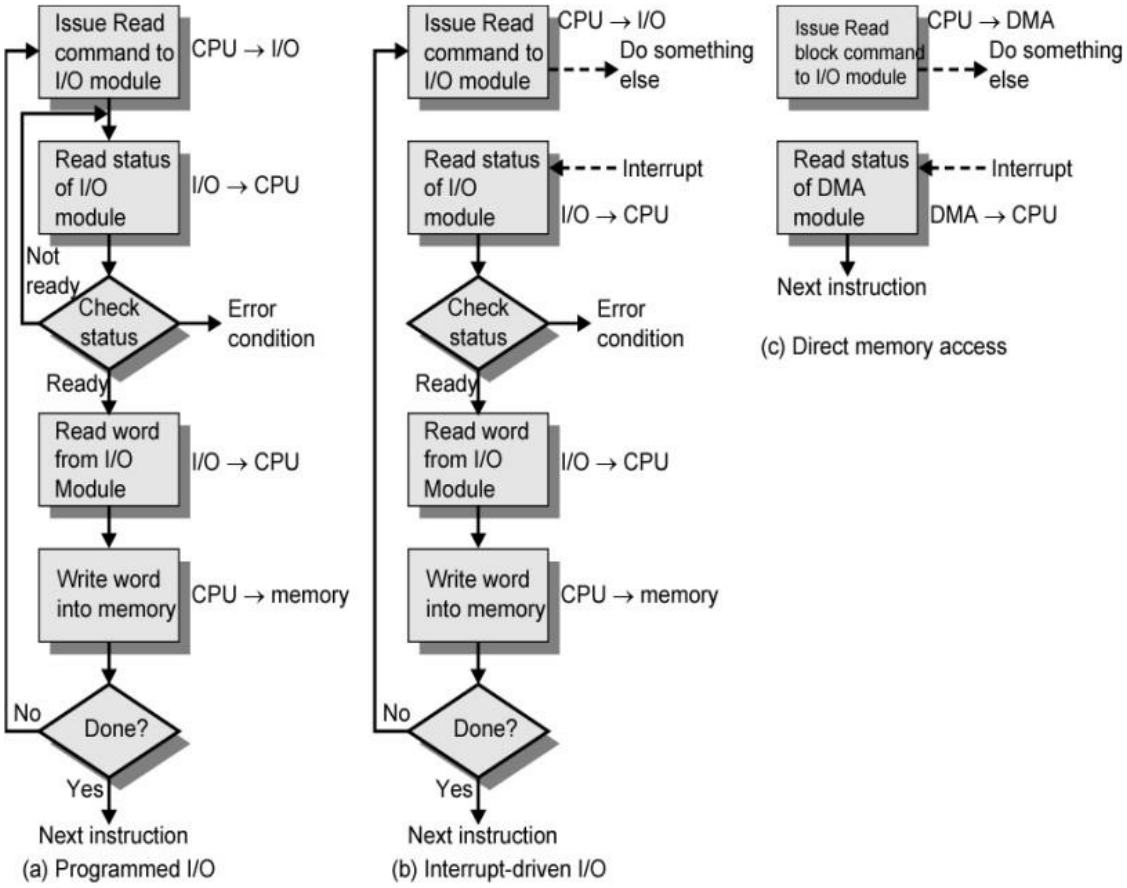
- Often, data that comes off a device cannot be stored directly to its final destination due to various reasons.
- Buffer is the intermediate destination

6) Dedicated vs. sharable device:

- Devices that can be used by many users at a time is sharable. E.g. Disk (multiple users can open files from the same disk at the same time)
- Some devices have to be dedicated to a single person. E.g. tapes.

I/O Handling

- I/O Operation can be performed in following three ways:
    1. Programmed I/O.
    2. Interrupt driven I/O.
    3. Direct Memory Access (DMA)



1) Programmed I/O:

- Programmed I/O operations are the result of I/O instructions written in the computer program.
- In programmed I/O, each data transfer in initiated by the instructions in the CPU and hence the CPU is in the continuous monitoring of the interface.
- Input instruction is used to transfer data from I/O device to CPU, store instruction is used to transfer data from CPU to memory and output instruction is used to transfer data from CPU to I/O device. This technique is generally used in very slow speed computer and is not an efficient method if the speed of the CPU and I/O is different.

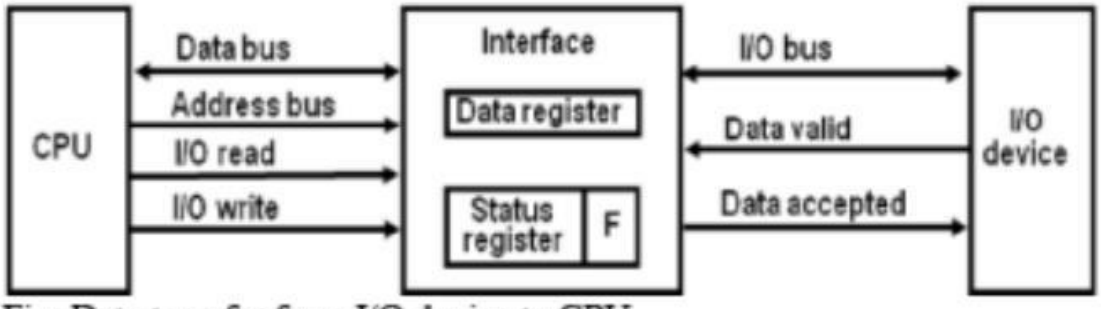


Fig: Data transfer from I/O device to CPU

- I/O device places the data on the I/O bus and enables its data valid signal
- The interface accepts the data in the data register and sets the F bit of status register and also enables the data accepted signal.
- Data valid line is disables by I/O device.
- CPU is in a continuous monitoring of the interface in which it checks the F bit of the status register.
    - If it is set i.e. 1, then the CPU reads the data from data register and sets F bit to zero.
    - If it is reset i.e. 0, then the CPU remains monitoring the interface.
- Interface disables the data accepted signal and the system goes to initial state where next item of data is placed on the data bus.

Characteristics:

- Continuous CPU involvement
- CPU slowed down to I/O speed
- Simple
- Least hardware

2) Interrupt driven I/O

- The problem with programmed I/O is that the processor has to wait a long time for the I/O module of concern to be ready for either reception or transmission of data.
- The processor, while waiting, must repeatedly interrogate the status of the I/O module. As a result, the level of the performance of the entire system is severely degraded.
- So in this approach The I/O module will then interrupt the processor to request service when it is ready to exchange data with processor. The processor then executes the data transfer, and then resumes its former processing.
- The interrupt can be initiated either by software or by hardware.

Interrupt Driven I/O basic operation

- CPU issues read command
- I/O module gets data from peripheral whilst CPU does other work
- I/O module interrupts CPU
- CPU requests data
- I/O module transfers data

Interrupt Processing from CPU viewpoint

- Issue read command
- Do other work
- Check for interrupt at end of each instruction cycle
- If interrupted:-
    - Save context (registers)
    - Process interrupt
    - Fetch data & store

3) Direct Memory Access (DMA)

- The transfer of data between the peripheral and memory without the interaction of CPU and letting the peripheral device manage the memory bus directly is termed as Direct Memory Access (DMA). The two control signals Bus Request and Bus Grant are used to fascinate the DMA transfer.
- The bus request input is used by the DMA controller to request the CPU for the control of the buses.
- When BR signal is high, the CPU terminates the execution of the current instructions and then places the address, data, read and write lines to the high impedance state and sends the bus grant signal.
- The DMA controller now takes the control of the buses and transfers the data directly between memory and I/O without processor interaction.
- When the transfer is completed, the bus request signal is made low by DMA. In response to which CPU disables the bus grant and again CPU takes the control of address, data, read and write lines. The transfer of data between the memory and I/O of course facilitates in two ways which are DMA Burst and Cycle Stealing.

1. DMA Burst:

- The block of data consisting a number of memory words is transferred at a time.

2. Cycle Stealing:

- DMA transfers one data word at a time after which it must return control of the buses to the CPU. CPU is usually much faster than I/O (DMA), thus CPU uses the most of the memory cycles
- DMA Controller steals the memory cycles from CPU
- For those stolen cycles, CPU remains idle
- For those slow CPU, DMA Controller may steal most of the memory cycles which may cause CPU remain idle long time

I/O software Layers

- I/O software is organized in four layers.
- Each layer has a well-defined function to perform and a well-defined interface to the adjacent layers.
- The functionality and interfaces differ from system to system.

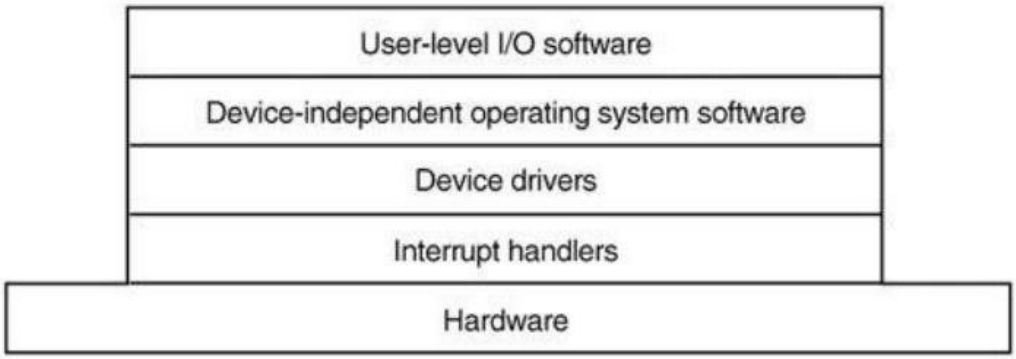


Fig: Layers of Software

1) User Level I/O

- These are application level software which use I/O system calls for the input and output. For example, in c programming, printf() and scanf() functions are used for output and input.
- Not all user level software consists of library procedures.
- Another important category is the spooling system. It is the way of dealing with delicate I/O devices in the multiprogramming system. In printer, a special process called a daemon and a special directory called a spooling directory are used to manage the output. To print a file, a process that generates the entire file to be printed and puts it in the spooling directory to use the printer's special file, to print the file in directory.

2) Device Independent Operating system software

- Although some of the I/O software is device specific, other parts of it are device independent.
- The basic function of device-independent software is to perform the I/O functions that are common to all devices and provide uniform interface to user-level software.
- Functions of device independent I/O software
  - Uniform interfacing for device drivers
  - Buffering
  - Error Reporting
  - Allocating and releasing dedicated devices
  - Providing a device-independent block size

3) Device Driver

- Device specific code for controlling the device is called device driver.
- Written by device's manufacturer and delivered along with the device.
- Device driver is the software part that communicates to the device controller, giving it commands and accepting response.

Functions of Device Driver

- Accept read/write request from device independent I/O software above it.
- Initialize the device if needed.
- Manage its power requirements and log events
- Check whether the input parameters are valid 〉 Translate parameters from abstract to concrete terms (e.g., linear block number to CHS(Cylinder Head, Sector) for disk)
- Check if the device is currently in use. If it is request will be queued for later processing.

Working Mechanism

- Firstly command sequence is determined.
- After knowing issued command sequences, it starts writing them into controller's device registers.
- After that, it checks to see if the controller accepted the command and is prepared to accept next command.
- Driver blocks (or waits) till the controller does some work for it.
- It blocks itself until the interrupt comes into unblock it.
- Interrupt is sent to unblock the device driver.

- After operation has been completed, driver checks for error and if everything is all right, the driver passes data to the device independent software.
- Finally, it returns some status information for error reporting back the caller.
- If none of the request is in queue, the driver blocks waiting for next request.

4) Interrupt Handler

Steps:

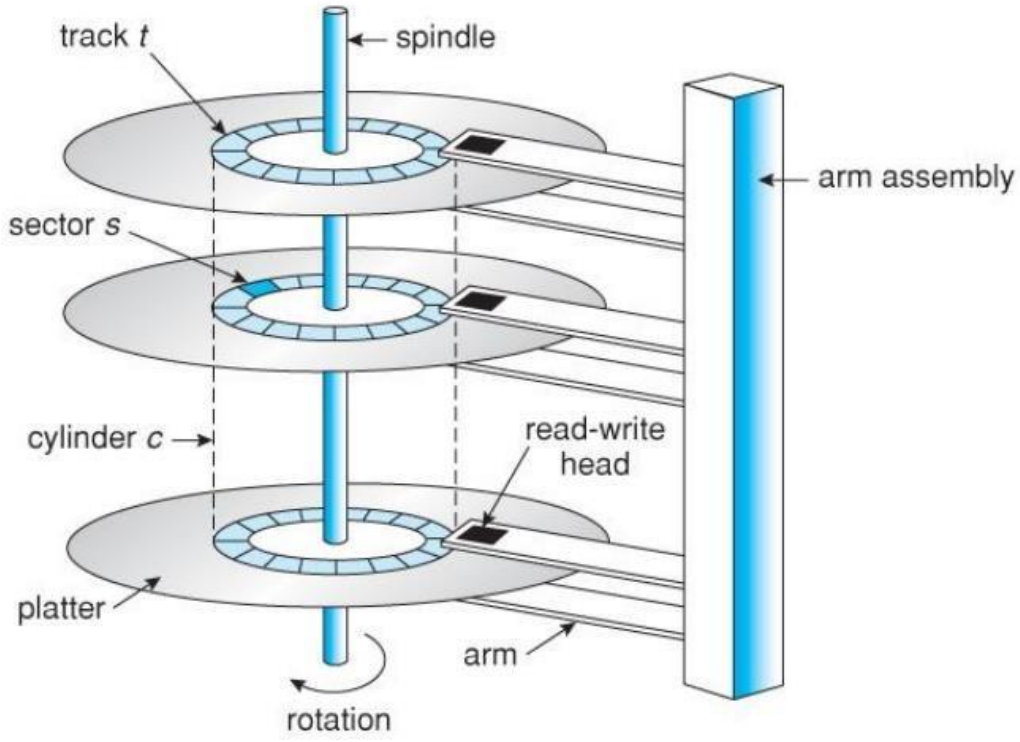
- Save any registers (including PSW) not already saved by interrupt hardware.
- Set up a context for the interrupt service procedure.(this may include Page table, TLB)
- Set up a stack for interrupt service procedure.
- Acknowledge the interrupt controller.
- Copy the registers from where they were saved (possibly some stack) to the process table
- Run the interrupt service procedure.
- Choose which process to run next.
- Set up the MMU context for the process to run next.
- Load the new process' registers, including its PSW.
- Start running the new process

Disk

- Disks are the information storage media
- Advantages of disks over main memory:
  - Storage capacity of disk is greater than main memory
  - Disks are non-volatile (i.e. unlike main memory, the information on disk is not lost even after power off).
- Disks come in variety of types:
  - Magnetic Disks: Hard Disks, Floppy disks
  - Optical Disks: CD-ROMs, DVDs etc.

Disk Hardware

- A disk consists of several platters
- Each disk platter has a flat circular shape, like a CD
- The two surfaces of a platter are covered with a magnetic material
- A read-write head "flies" just above each surface of every platter
- The heads are attached to a disk arm that moves all the heads as a unit
- The surface of a platter is logically divided into circular tracks, which are subdivided into sectors The set of tracks that are at one arm position makes up a cylinder
- There may be thousands of concentric cylinders in a disk drive, and each track may contain hundreds of sectors.



Accessing Data from disk

- Data on a magnetic disk is recorded on the surface of circular tracks with the help of read/write head, which is mounted on an arm assembly.
- These head can be multiple in numbers to access the adjacent tracks simultaneously, thus making disk access faster.
- The transfer of data between memory and disk drive is handled by a Disk Controller, which interface the disk drive to computer system.
- The process of accessing data comprises three steps:
  1. Seek
  2. Rotation
  3. Data Transfer

1) Seek

- As soon as the disk unit receives the read/write command, the read/write heads are positioned on specific track on the disk platter.
- The time taken in doing so is known as seek time.
- Seek time depends on how fast the hardware moves the arm.
- Seek time of modern disk ranges between 6 to 15 milliseconds.

2) Rotation

- Once the head are positioned on desire tracks, the head of specific platter is activated. Since the disk is rotating constantly, the head has to wait for the required.

3) Data Transfer

- After waiting period for desired data location ends, the Read/Write head transfer data to or from the disk to memory.
- The rate at which data is Read from or Written to the disk is known as data transfer rate.
- The data transfer rate depends upon the rotational speed of the disk.
- If the disk has rotational speed of 6000 rpm (rotation per minute), having sectors 125 and 512 bytes/sector, the data transfer rate per revolution will be 125 x 512 = 64000 bytes. Hence total transfer rate per second will be (64000 x 6000/60) bytes/minute or 6.4 MB/Sec.
- Disk I/O time = seek time + rotational delay + transfer time

Disk Arm scheduling Algorithms

- The main goal of disk scheduling is to minimize the seek time.
- We know that Disk: **I/O time = seek time + rotational delay + transfer time**
- For most disk, seek time dominates the other two times.
- So, reducing the mean seek time can improve system performance time substantially.
- The idea is to permute the order of the disk requests form the order that they arrive from the users to an order that reduces the length and number of seeks.
- We will look at several disk arm scheduling algorithms for doing so.

Algorithms

- Following are the various Disk Scheduling Algorithms.
  1. First Come First Served (FCFS) Algorithm
  2. Shortest Seek Time First (SSTF) Algorithm
  3. SCAN Algorithm
  4. Circular SCAN (C-SCAN) Algorithm
  5. LOOK Algorithm
  6. Circular LOOK (C-LOOK) Algorithm

1) First Come First Served (FCFS) Algorithm

- 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

Advantages

- It is simple, easy to understand and implement.
- It does not cause starvation to any request.

Disadvantages

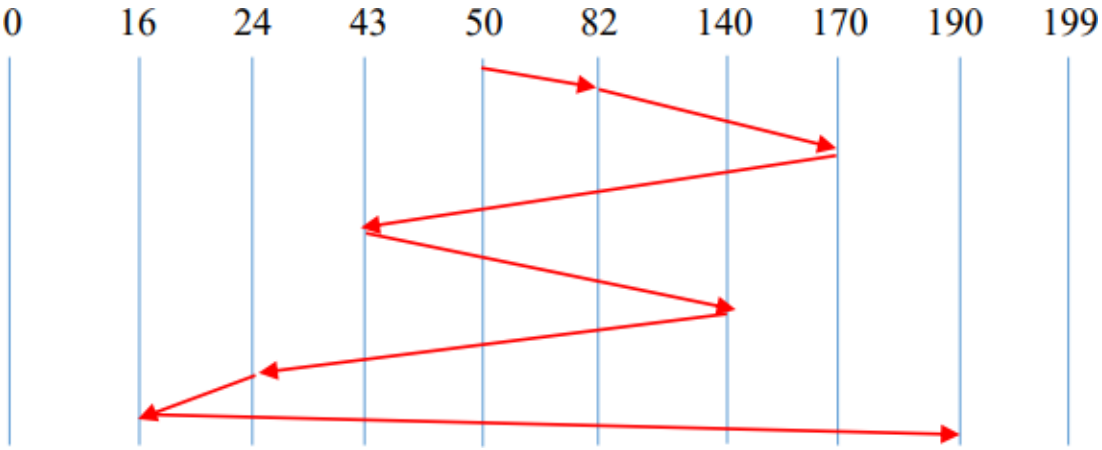
- It results in increased total seek time.
- It is inefficient.

Example 1.

Suppose that a disk has 200 cylinders, numbered from 0 to 199. The read/write head of drive is currently serving a request at 50, and the previous request was at cylinder 40. The queue of pending request, in FIFO order is:

82, 170, 43, 140, 24, 16, 190

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending request, for FCFS Algorithm?



Total head Movement = (82 – 50) + (170 – 82) + (170 – 43) + (140 – 43) + (140 – 24) + (24 – 16) + (190 – 16) Cylinder

=642 Cylinder.

2) Shortest Seek Time First (SSTF) Algorithm

- This algorithm services that request next which requires least number of head movements from its current position regardless of the direction.
- It breaks the tie in the direction of head movement.

Advantages

- It reduces the total seek time as compared to FCFS.
- It provides increased throughput.
- It provides less average response time and waiting time.

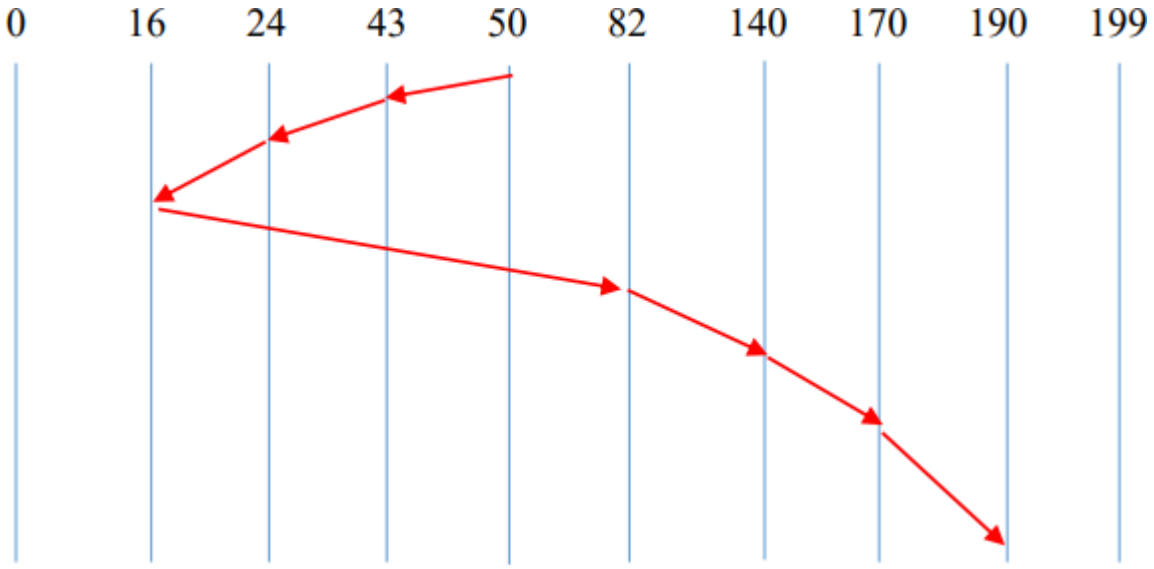
Disadvantages

- There is an overhead of finding out the closest request.
- The requests which are far from the head might starve for the CPU.
- Switching the direction of head frequently slows down the algorithm.

Example 1. Suppose that a disk has 200 cylinders, numbered from 0 to 199. The read/write head of drive is currently serving a request at 50, and the previous request was at cylinder 40. The queue of pending request , in FIFO order is:

82, 170, 43, 140, 24, 16, 190

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending request, for SSTF Algorithm?



Total head Movement = (50 - 43) + (43 – 24) + (24 – 16) + (82 – 16) + (140 – 82) + (170 – 140) + (190 – 170) Cylinder

= 208 Cylinder

3) SCAN Algorithm

- As the name suggests, this algorithm scans all the cylinders of the disk back and forth.
- Head starts from one end of the disk and move towards the other end servicing all the requests in between.
- After reaching the other end, head reverses its direction and move towards the starting end servicing all the requests in between.
- The same process repeats.
- SCAN Algorithm is also called as Elevator Algorithm.
- This is because its working resembles the working of an elevator.

Advantages

- It is simple, easy to understand and implement.
- It does not lead to starvation.

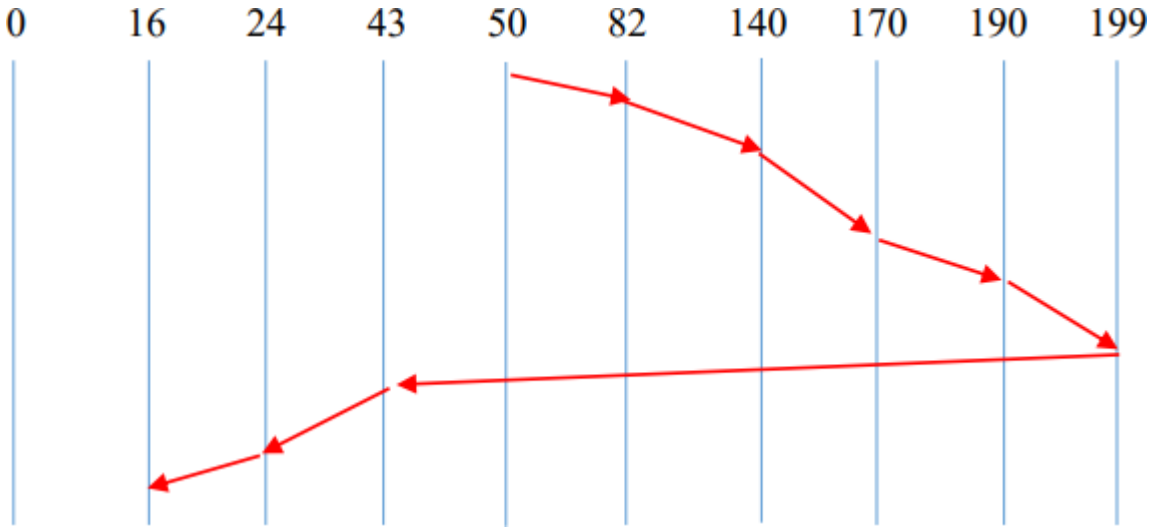
Disadvantages

- It causes the head to move till the end of the disk even if there are no requests to be serviced.

Example 1. Suppose that a disk has 200 cylinders, numbered from 0 to 199. The read/write head of drive is currently serving a request at 50, and the previous request was at cylinder 40. The queue of pending request , in FIFO order is:

82, 170, 43, 140, 24, 16, 190

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending request, for SCAN Algorithm?



Total head Movement = (82 - 50) + (140 – 82) + (170 – 140) + (190 – 170) + (199 – 190) + (199 – 43) + (43 – 24) + (24 – 16) Cylinder

= 332 Cylinder.

4) Circular SCAN (C-SCAN) Algorithm

- Circular-SCAN Algorithm is an improved version of the SCAN Algorithm.
- Head starts from one end of the disk and move towards the other end servicing all the requests in between.
- After reaching the other end, head reverses its direction.
- It then returns to the starting end without servicing any request in between.
- The same process repeats.

Advantages

- The waiting time for the cylinders just visited by the head is reduced as compared to the SCAN Algorithm.
- It provides uniform waiting time.
- It provides better response time.

Disadvantages

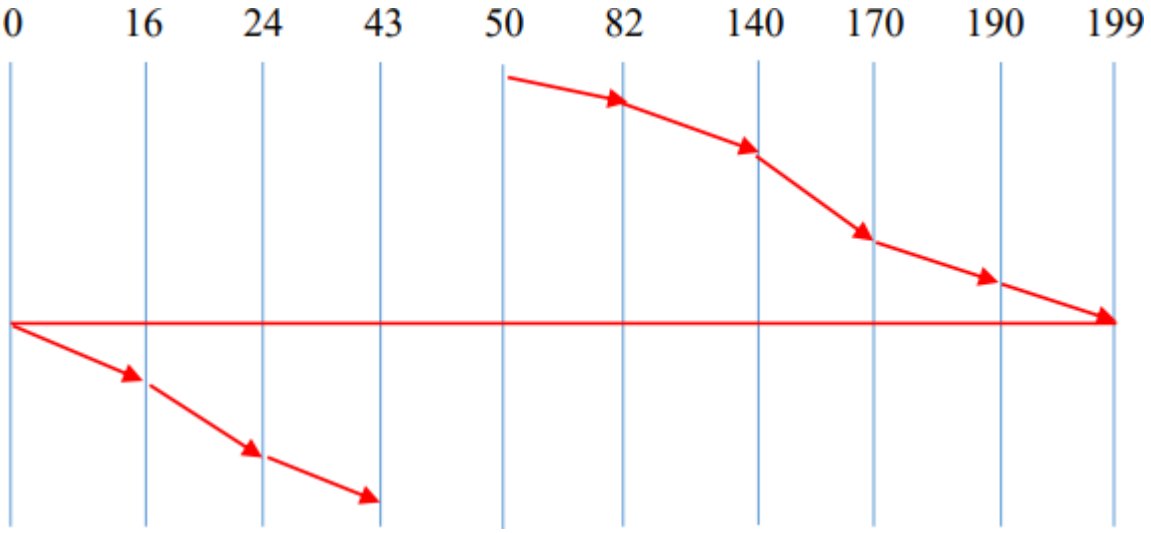
- It causes more seek movements as compared to SCAN Algorithm.
- It causes the head to move till the end of the disk even if there are no requests to be serviced.

Example 1.

Suppose that a disk has 200 cylinders, numbered from 0 to 199. The read/write head of drive is currently serving a request at 50, and the previous request was at cylinder 40. The queue of pending request, in FIFO order is:

82, 170, 43, 140, 24, 16, 190

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending request, for C-SCAN Algorithm?



Total head Movement = (82 - 50) + (140 – 82) + (170 – 140) + (190 – 170) + (199 – 190) + (199 – 0) + (16 – 0) + (24 – 16) + (43 – 24) Cylinder

= 391 Cylinder

5) LOOK Algorithm

- LOOK Algorithm is an improved version of the SCAN Algorithm.
- Head starts from the first request at one end of the disk and 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 first request at the starting end servicing all the requests in between.
- The same process repeats.

Advantages

- It does not causes the head to move till the ends of the disk when there are no requests to be serviced.
- It provides better performance as compared to SCAN Algorithm.
- It does not lead to starvation.

Disadvantages

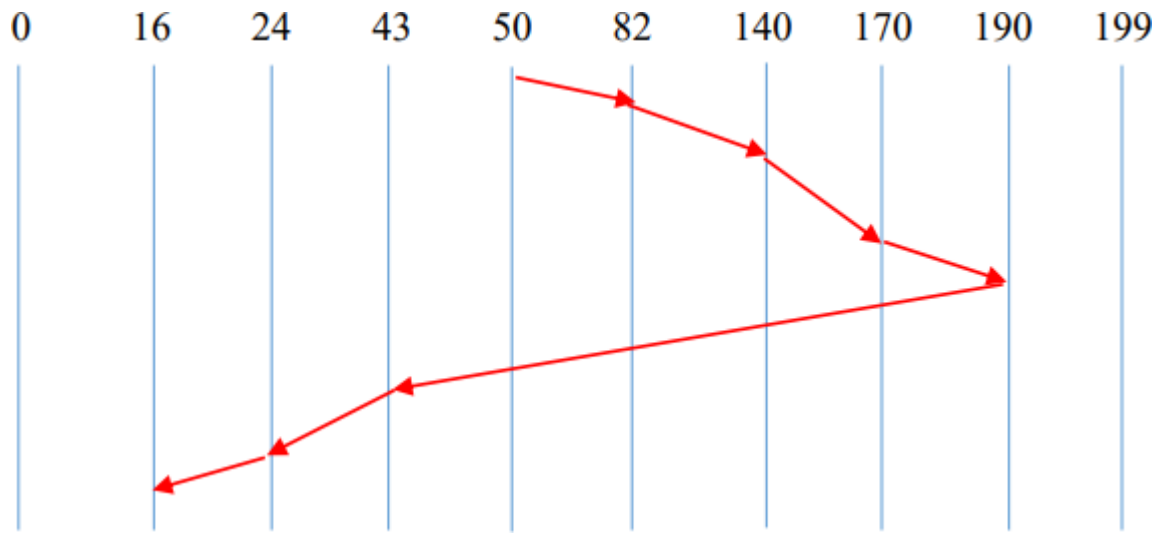
- There is an overhead of finding the end requests.

Example 1.

Suppose that a disk has 200 cylinders, numbered from 0 to 199. The read/write head of drive is currently serving a request at 50, and the previous request was at cylinder 40. The queue of pending request, in FIFO order is:

82, 170, 43, 140, 24, 16, 190

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending request, for LOOK Algorithm?



Total head Movement = (82 - 50) + (140 – 82) + (170 – 140) + (190 – 170) + (190 – 43) + (43 – 24) + (24 – 16) Cylinder

= 314 Cylinder

6) Circular LOOK (C-LOOK) Algorithm

- Circular-LOOK Algorithm is an improved version of the LOOK Algorithm.
- Head starts from the first request at one end of the disk and 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 first request at the starting end without servicing any request in between.
- The same process repeats.

Advantages

- It does not causes the head to move till the ends of the disk when there are no requests to be serviced.
- It provides better performance as compared to LOOK Algorithm.
- It does not lead to starvation.

Disadvantages

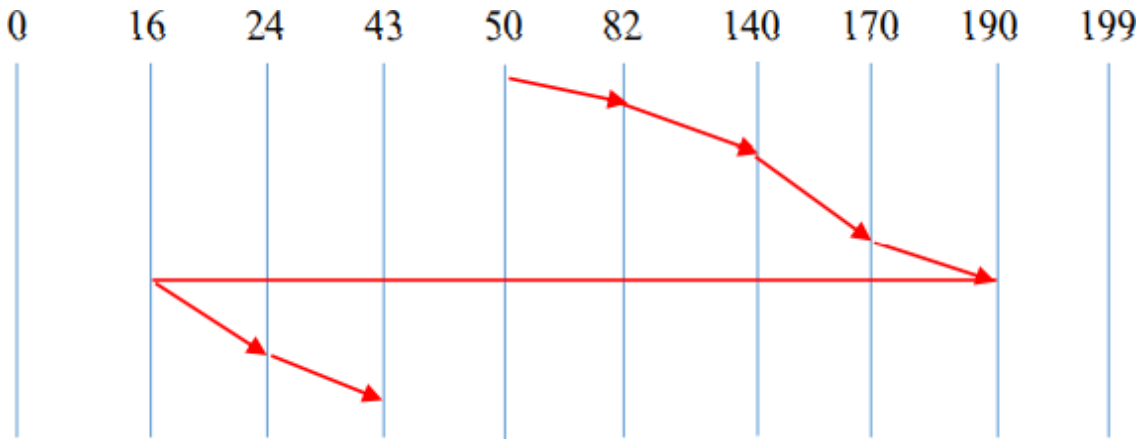
- There is an overhead of finding the end requests.

Example 1.

Suppose that a disk has 200 cylinders, numbered from 0 to 199. The read/write head of drive is currently serving a request at 50, and the previous request was at cylinder 40. The queue of pending request, in FIFO order is:

82, 170, 43, 140, 24, 16, 190

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending request, for C-LOOK Algorithm?



Total head Movement = (82 - 50) + (140 – 82) + (170 – 140) + (190 – 170) + (16 – 0) + (24 – 16) + (43 – 24) Cylinder

= 341 Cylinder

Questions for Practice

1. Suppose that a disk has 200 cylinders, numbered from 0 to 199. The read/write head of drive is currently serving a request at 53, and the previous request was at cylinder 40. The queue of pending request, in FIFO order is:

98, 183, 37, 122, 14, 124, 65, 67

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending request, for following Algorithm?

1) First Come First Served (FCFS) Algorithm
2) Shortest Seek Time First (SSTF) Algorithm
3) SCAN Algorithm
4) Circular SCAN (C-SCAN) Algorithm
5) LOOK Algorithm
6) Circular LOOK (C-LOOK) Algorithm

2. Suppose that a disk has 200 tracks, numbered from 0 to 199. The read/write head of drive is currently serving a request at track143, and has just finished the request at track 125. The queue of pending request , in FIFO order is:

86, 147, 91, 177, 94, 150, 102, 175, 130

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending request, for following Algorithm? Also calculate average head movements for each algorithms.

1) First Come First Served (FCFS) Algorithm
2) Shortest Seek Time First (SSTF) Algorithm
3) SCAN Algorithm
4) Circular SCAN (C-SCAN) Algorithm.
5) LOOK Algorithm.

3. Suppose that a disk has 100 tracks, numbered from 0 to 99. The read/write head of drive is currently serving a request at track 20 and assume 19 was accessed before 20.. The queue of pending request , in FIFO order is:

5, 25, 18, 3, 39, 8 and 35.

Suppose seek takes 5 ms per cylinder moved. Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending request, for following Algorithm? Also calculate seek time needed to serve these request for each algorithm.

1) First Come First Served (FCFS) Algorithm
2) Shortest Seek Time First (SSTF) Algorithm
3) SCAN Algorithm
4) LOOK Algorithm.