# 15CI06 – Operating Systems

# I/O Management

**Dr.M.Rajalakshmi**
**Associate Professor**
**Department of IT**
**Coimbatore Institute of Technology**
**Coimbatore**

# Roadmap

- I/O Devices
- Organization of the I/O Function
- Operating System Design Issues
- I/O Buffering
- Disk Scheduling

# Categories of I/O Devices

- I/O Management module manages and controls I/O operations and I/O devices.
- Difficult area of OS design
  - Difficult to develop a consistent solution due to a wide variety of devices and applications
- Three Categories of I/O devices :
  - Human readable
  - Machine readable
  - Communications

# Categories of I/O devices

- Human readable - Devices used to communicate with the computer user

Eg. - Printers and terminals, Video display, Keyboard, Mouse etc.

- Machine readable - Used to communicate with electronic equipment

Eg. Disk drives, USB keys, Sensors, Controllers, Actuators

- Communication - Used to communicate with remote devices.

Eg. Digital line drivers, Modems, Routers

# Differences in I/O Devices

Devices differ in a number of areas

- Data Transfer Rate- Data transfer rate varies from device to device. Eg. Key Board – 100 bps , Mouse – 200 to 1000 bps etc.
- Application (Support of S/W, eg. Disk needs support of file management, memory management)
- Complexity of Control – Printer requires simple interface and disk requires complex interface
- Unit of Transfer - a stream of bytes or characters (e.g., terminal I/O) or in larger blocks (e.g., disk I/O).
- Data Representation – Different encoding schemes for different devices
- Error Conditions – nature of errors reported ,their consequences, the available range of responses

# Data Rate

- May be massive difference between the data transfer rates of devices

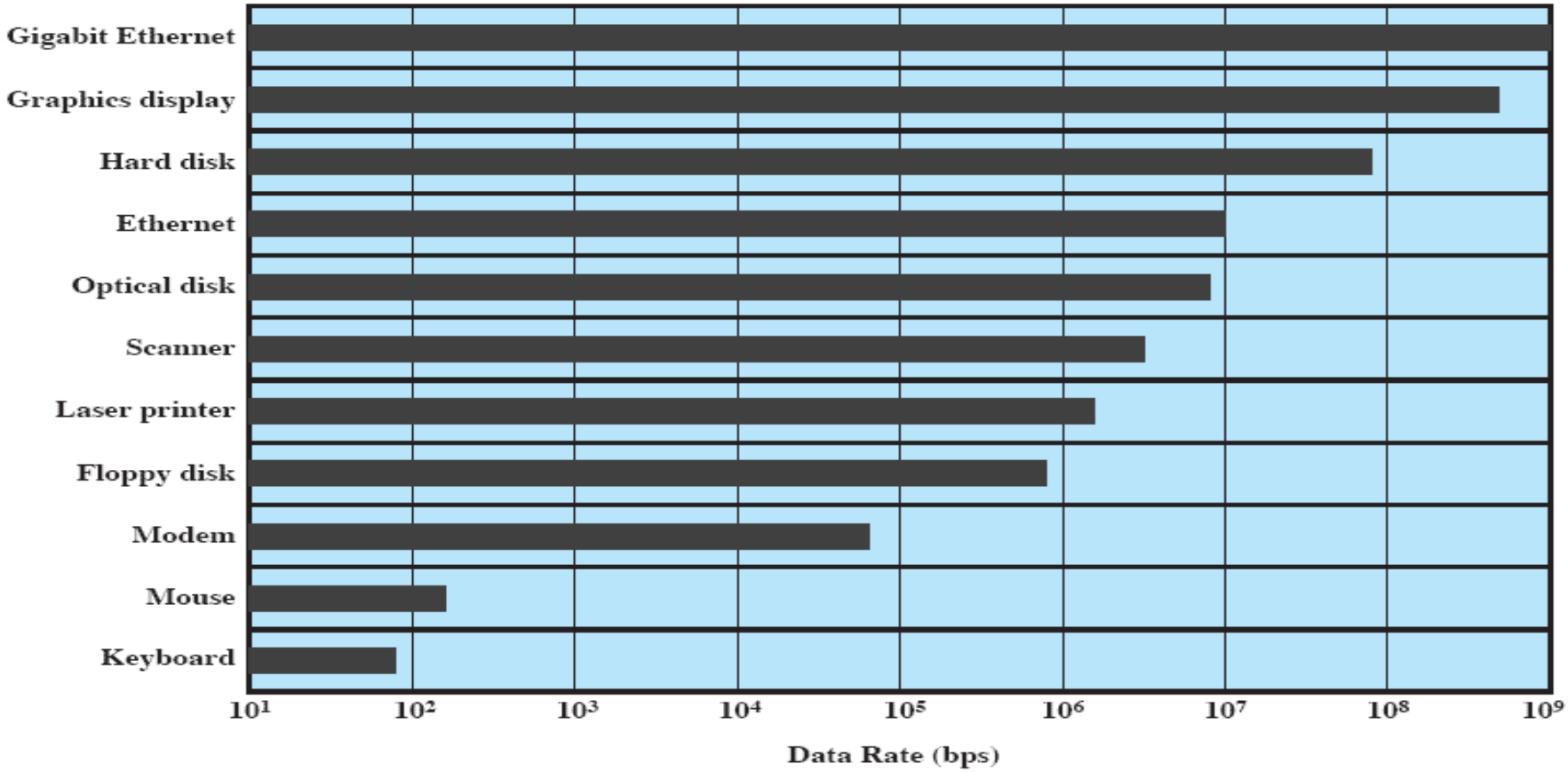



**Figure 11.1 Typical I/O Device Data Rates**

# Characteristics of I/O Devices

| aspect | variation | example |
|---|---|---|
| data-transfer mode | character<br>block | terminal<br>disk |
| access method | sequential<br>random | modem<br>CD-ROM |
| transfer schedule | synchronous<br>asynchronous | tape<br>keyboard |
| sharing | dedicated<br>sharable | tape<br>keyboard |
| device speed | latency<br>seek time<br>transfer rate<br>delay between operations | |
| I/O direction | read only<br>write only<br>read–write | CD-ROM<br>graphics controller<br>disk |

# Classification of I/O devices based on unit of data transfer

- Block oriented devices - stores and transfers information in blocks usually of fixed size
  - Data is referred by its block number
  - Commands include read, write, seek
    Eg. Disk and Tape
- Character oriented devices -stores and transfers information in a stream of bytes.
  - has no block structure
  - Commands include get and put

  E.g. keyboards, mouse, printers, terminals, serial ports

# Roadmap

- I/O Devices
- Organization of the I/O Function
- Operating System Design Issues
- I/O Buffering
- Disk Scheduling

# Techniques for performing I/O

## (I/O communication Techniques or Data transfer techniques)

- Programmed I/O
- Interrupt-driven I/O
- Direct memory access (DMA)

**Table 11.1 I/O Techniques**

| | No Interrupts | Use of Interrupts |
|---|---|---|
| **I/O-to-memory transfer through processor** | Programmed I/O | Interrupt-driven I/O |
| **Direct I/O-to-memory transfer** | | Direct memory access (DMA) |

# Programmed I/O

- When the processor is executing a program and encounters an instruction relating to I/O, it executes that instruction by issuing a command to the appropriate I/O module.
- I/O module performs the requested action and then sets the status register. It does not interrupt the processor.
- I/O module sends the request to the appropriate device (Input or Output). If the device is ready, it puts the status in the status register and the data is send to the data register.
- After the I/O instruction is invoked, the processor must checks periodically whether the I/O instruction is completed or not. Processor is in busy waiting mode.
- The CPU, while waiting, must repeatedly check the status of the I/O module, and this process is known as **Polling**.

# Programmed I/O

Programmed I/O basically works in these ways:

- CPU requests I/O operation
- I/O module performs operation
- I/O module sets status bits
- CPU checks status bits periodically
- I/O module does not inform CPU directly
- I/O module does not interrupt CPU
- CPU may wait or come back later

# Interrupt-driven I/O

- The CPU issues an I/O commands to the I/O module on behalf of a process continues to execute subsequent instructions and is interrupted by the I/O module when the I/O operation has completed.
- For input, the device interrupts the CPU when new data has arrived and is ready to be retrieved by the system processor. The actual actions to perform depend on whether the device uses I/O ports, memory mapping.
- For output, the device delivers an interrupt either when it is ready to accept new data or to acknowledge a successful data transfer.

# Interrupt-driven I/O

Below are the basic operations of Interrupt:

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

# Direct memory access (DMA)

- CPU grants I/O module authority to read from or write to memory without involvement. DMA module controls exchange of data between main memory and the I/O device. Because of DMA device can transfer data directly to and from memory, rather than using the CPU as an intermediary, and can thus relieve congestion on the bus. CPU is only involved at the beginning and end of the transfer and interrupted only after entire block has been transferred.
- Direct Memory Access needs a special hardware called DMA controller (DMAC) that manages the data transfers and arbitrates access to the system bus.

# Direct Memory Address

- The blocks of data can be transferred between the memory and I/O devices without the CPU intervention. The processor will start the DMA controller and continues with other work.
- Processor is involved only at the beginning and end of the transfer.

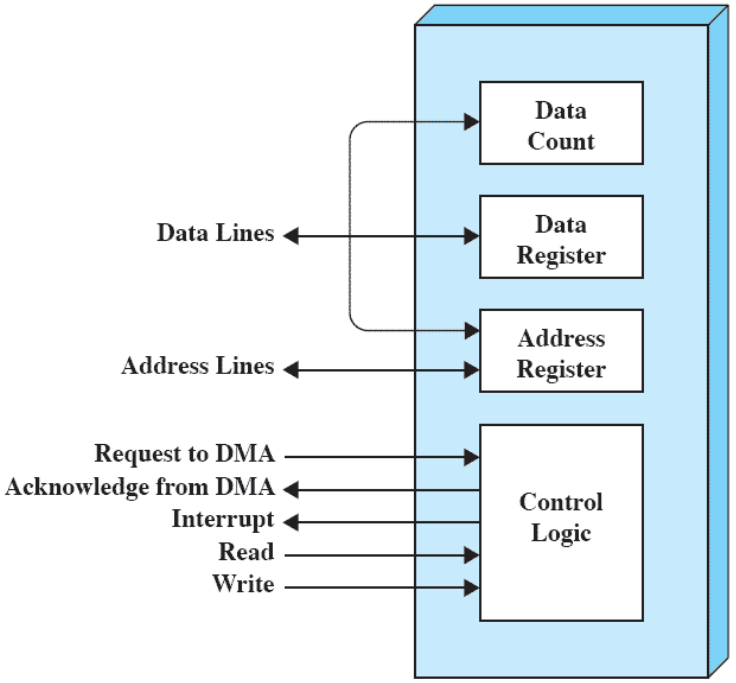
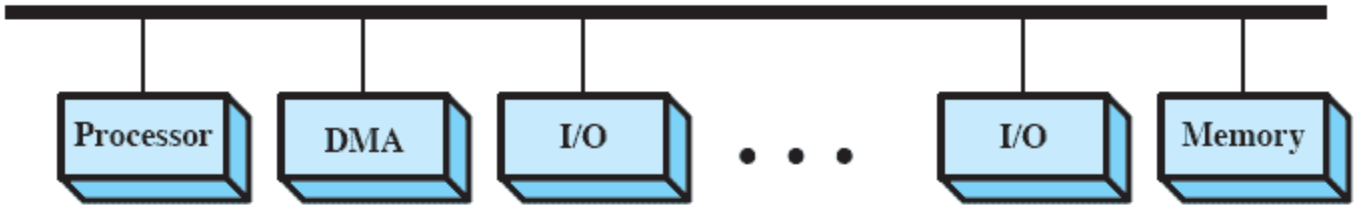



Figure 11.2 Typical DMA Block Diagram

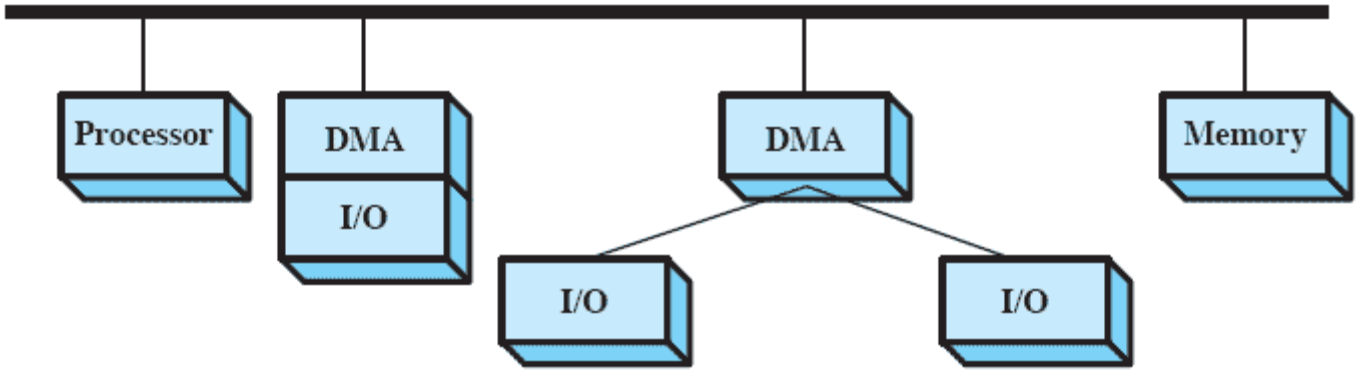
# DMA Configurations: Single Bus




(a) Single-bus, detached DMA

- DMA can be configured in several ways
- Shown here, all modules share the same system bus

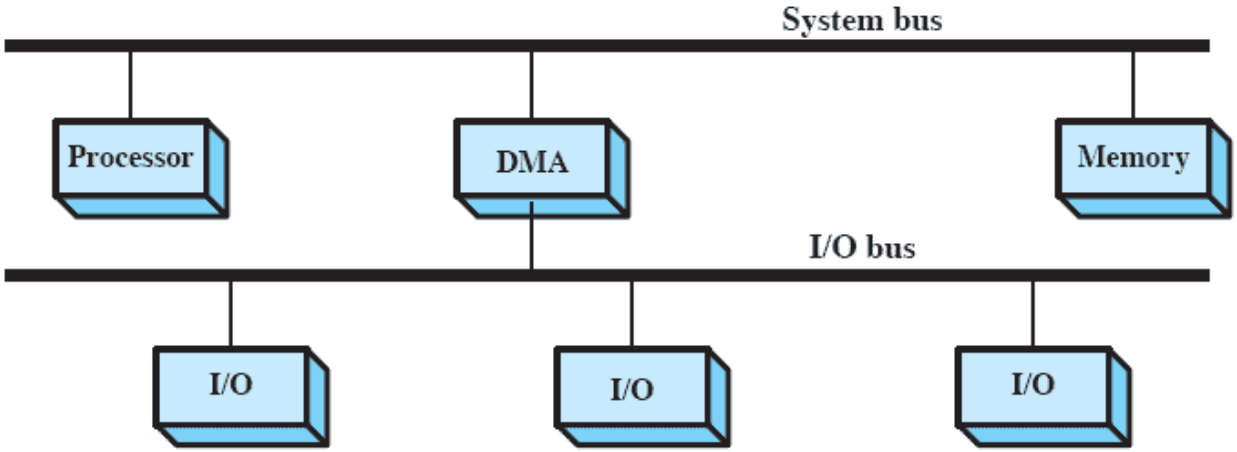
# DMA Configurations: Integrated DMA & I/O




(b) Single-bus, Integrated DMA-I/O

- Direct Path between DMA and I/O modules
- This substantially cuts the required bus cycles

# DMA Configurations: I/O Bus




(c) I/O bus

- Reduces the number of I/O interfaces in the DMA module

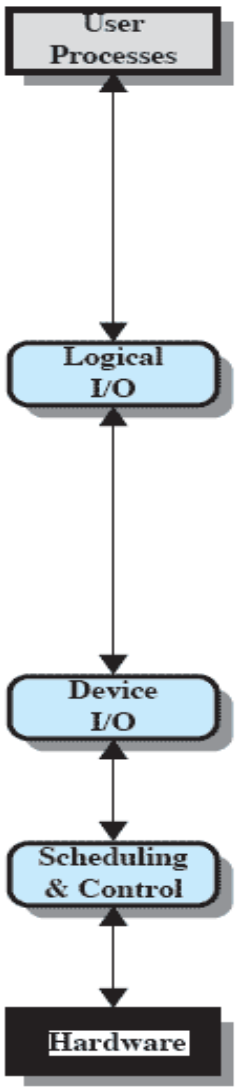
# Evolution of the I/O Function

1. Processor directly controls a peripheral device
2. Controller or I/O module is added
   - Processor uses programmed I/O without interrupts
   - Processor does not need to handle details of external devices
3. Controller or I/O module with interrupts
   - Efficiency improves as processor does not spend time waiting for an I/O operation to be performed
4. Direct Memory Access
   - Blocks of data are moved into memory without involving the processor
   - Processor involved at beginning and end only
5. I/O module is a separate processor
   - CPU directs the I/O processor to execute an I/O program in main memory.
6. I/O processor
   - I/O module has its own local memory
   - Commonly used to control communications with interactive terminals

# Operating System Design Issues

1. Efficiency
   - Most I/O devices extremely slow compared to main memory
   - Use of multiprogramming allows for some processes to be waiting on I/O while another process executes
   - I/O cannot keep up with processor speed
   - Swapping used to bring in ready processes but this is an I/O operation itself
2. Generality
   - For simplicity and freedom from error it is desirable to handle all I/O devices in a uniform manner
   - Hide most of the details of device I/O in lower-level routines
   - Difficult to completely generalize, but can use a hierarchical modular design of I/O functions

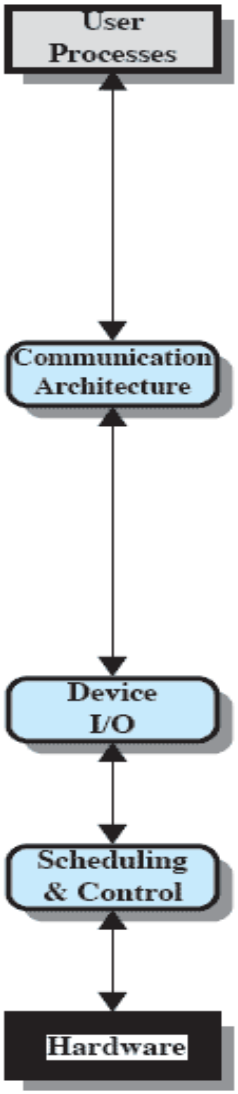
# Local Structure of peripheral device




(a) Local peripheral device

- Logical I/O:
  - Deals with the device as a logical resource
  - Deals with the device in terms of device identifier and simple commands such as open, close, read and write.
- Device I/O:
  - Converts requested operations into sequence of I/O instructions
  - Buffering techniques can be used to improve the utilization
- Scheduling and Control
  - Performs actual queuing and control operations

# Local Structure of Communications Port




(b) Communications port

Similar to previous but the logical I/O module is replaced by a communications architecture,

- This consist of a number of layers.
- An example is TCP/IP,

# Local Structure of File System

(c) File system

- Directory management
  - Concerned with user operations affecting directory files such as add, delete, etc.
- File System
  - Logical structure and operations that can be specified by the users such as open, close, read, write. Access rights are also managed at this layer.
- Physical organisation
  - Converts logical names ( logical references to files and records ) to physical addresses (considering track and sector structure )

# I/O Buffering

- Processes must wait for I/O to complete before proceeding
- It may be more efficient to perform input transfers in advance of requests being made and to perform output transfers some time after the request is made.
- Buffering is done for many reasons :

- **To cope up with speed mismatch between the processor and sec. Storage devices.**
- **To adopt between devices that have different data-transfer sizes**
- **In Computer networking buffers are used for fragmentation and reassembling of messages.**

**Sender side – Fragmentation**

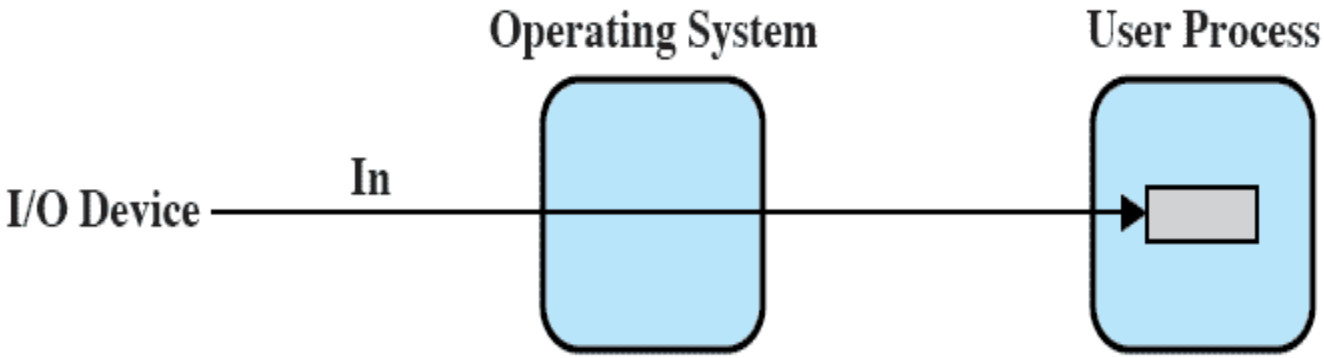
**Receiver side - reassembling**

# Types of Buffering

1. No buffering
2. Single Buffering
3. Double Buffering
4. Circular Buffering

# No Buffer

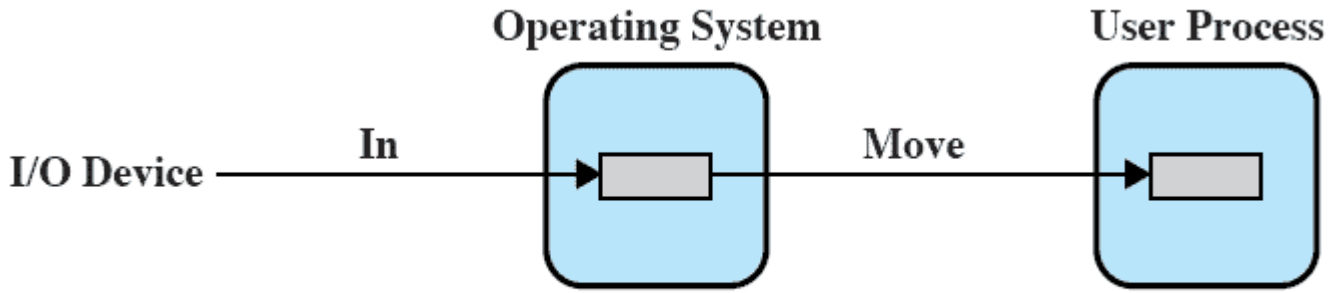
- Without a buffer, the OS directly access the device as and when it needs




(a) No buffering

# Single Buffering

- Operating system assigns a buffer in main memory for an I/O request
- Input transfers made to buffer
- Block moved to user space when needed
- The next block is moved into the buffer
  - *Read ahead* or *Anticipated Input*
- Often a reasonable assumption as data is usually accessed sequentially




(b) Single buffering

Block-oriented Buffering

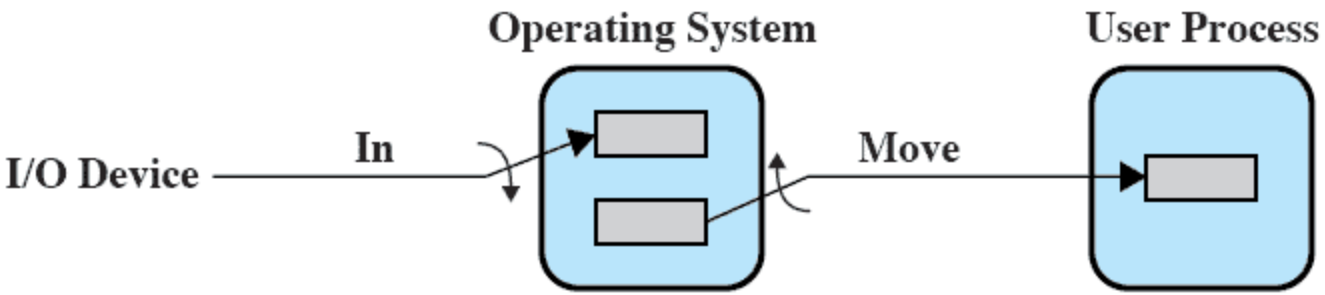
- Information is stored in fixed sized blocks
- Transfers are made a block at a time
  - Can reference data  as block number
- Used for disks and USB keys

Stream-Oriented Buffering

- Transfer information as a stream of bytes (Line-at-time or Byte-at-a-time)
- Terminals often deal with one line at a time with carriage return signaling the end of the line
- Byte-at-a-time suites devices where a single keystroke may be significant
  - Also sensors and controllers

# Double Buffering

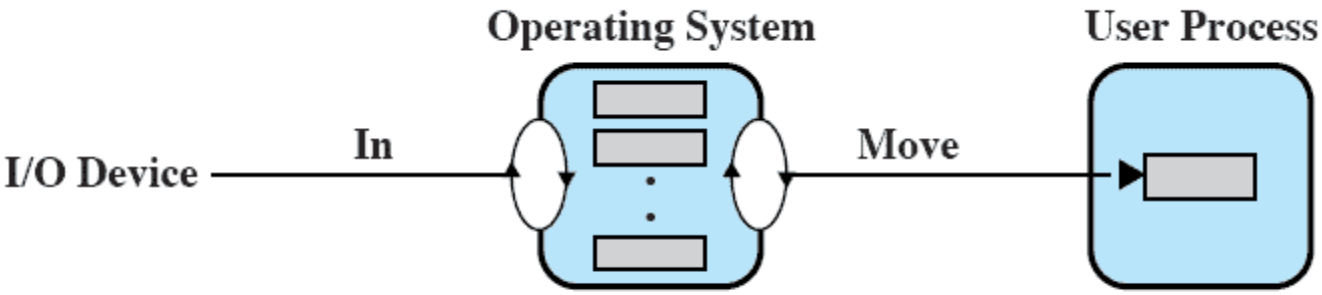
- Use two system buffers instead of one
- A process can transfer data to or from one buffer while the operating system empties or fills the other buffer
- Double buffering or buffer swapping




(c) Double buffering

# Circular Buffering

- More than two buffers are used
- Each individual buffer is one unit in a circular buffer
- Used when I/O operation must keep up with process




(d) Circular buffering

# Buffer Limitations

- Buffering smoothes out peaks in I/O demand.
  - But with enough demand eventually all buffers become full and their advantage is lost
- However, when there is a variety of I/O and process activities to service, buffering can increase the efficiency of the OS and the performance of individual processes.

# Disk Scheduling

The operating system is responsible for using hardware efficiently. When we consider the disk, disk drives - disk I/O operation depends on

- Computer system
- Operating System
- the nature of I/O channel and disk controller hardware

Disk Performance Parameters

1. Seek time
2. Rotational latency
3. Data Transfer Time

1. Seek time : is the time for the disk arm to move the heads to the desired track containing the desired sector.

*2.* Rotational latency (delay): is the additional time waiting for the disk to rotate the desired sector to the disk head.

Access Time = Seek Time + rotational Latency

3. Transfer Time : The time required for the data transfer

The transfer time to or from the disk depends on the rotation speed of the disk in the following fashion.

$$T = \frac{b}{rN}$$

where

$T$ = transfer time

$b$ = number of bytes to be transferred

$N$ = number of bytes on a track

$r$ = rotation speed, in revolutions per second

Thus, the total average access time can be expressed as

$$T_a = T_s + \frac{1}{2r} + \frac{b}{rN}$$

where $T_s$ is the average seek time.

Note : For problems refer book

# Disk Scheduling Policies

In a multiprogramming environment the OS maintains a queue of requests for each I/O device. So for a single disk there will be a number of I/O requests ( reads or writes) from various processes in the queue.

If we select the items in the random order then we can expect that the tracks to be visited will occur randomly giving poor performance.

Each request made up of

- whether input or output
- disk address ( disk track no. (cylinder No.), sector etc.)
- memory address for the transfer
- amount of information to be transferred (byte count)

If the desired disk drive and controller are available the request can be serviced immediately. If the drive or controller is busy, any new requests for service will need to be placed on the queue of pending requests for that drive.

# Disk Scheduling Policies

## Types

1. First-in, first-out (FIFO) or FCFS scheduling
2. Shortest Service Time First (SSTF) scheduling
3. SCAN scheduling
4. C-SCAN scheduling

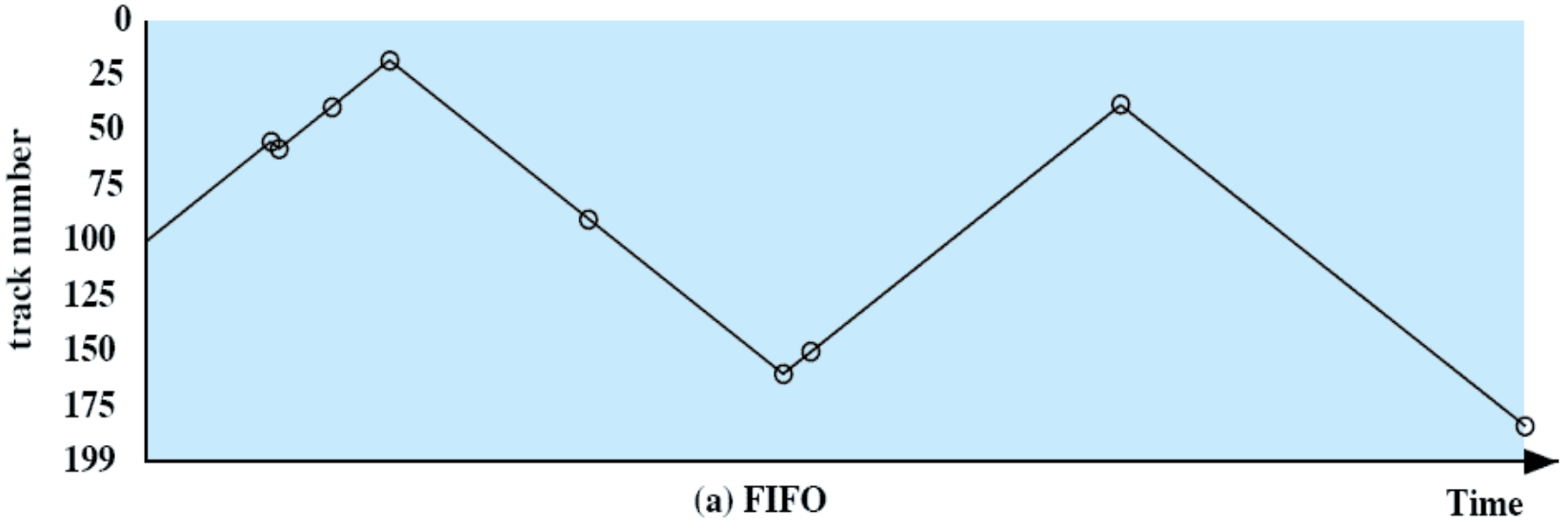
- To compare various schemes, consider a disk head is initially located at track 100.
  - assume a disk with 200 tracks and that the disk request queue has random requests in it.
- The requested tracks, in the order received by the disk scheduler, are
  - 55, 58, 39, 18, 90, 160, 150, 38, 184.

# First-in, first-out (FIFO)

- First request is serviced first
- Simplest form of scheduling
- It does not provide fast response

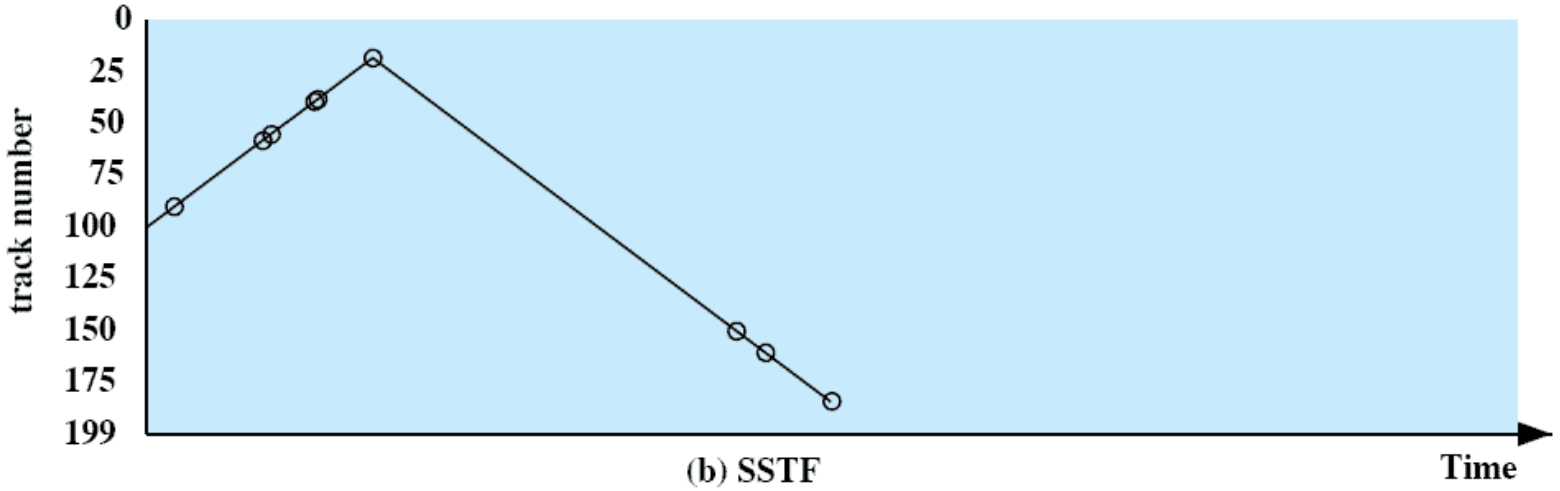
Requested tracks

55, 58, 39, 18, 90, 160, 150, 38, 184



# Shortest Service Time First (SSTF)

- Selects the request with the minimum seek time from the current head position.
- Always choose the minimum seek time
- SSTF scheduling may cause starvation of some requests
- Select the disk I/O request that requires the least movement of the disk arm from its current position
- Provide better performance than FIFO.



# SCAN

- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.
- Sometimes called the *elevator algorithm* since the disk arm behaves like an elevator in abuilding, first servicing all requests going up and then reversing to service requests the otherway.
- Arm moves in one direction only, satisfying all outstanding requests until it reaches the last track in that direction then the direction is reversed

# LOOK

- Refinement of SCAN scheduling algorithm is LOOK scheduling Policy.
- With LOOK scheduling, the arm is required to move in one direction satisfying all requests until it reaches the last requested track, then it reverses the direction without going all the way to the end of the disk.

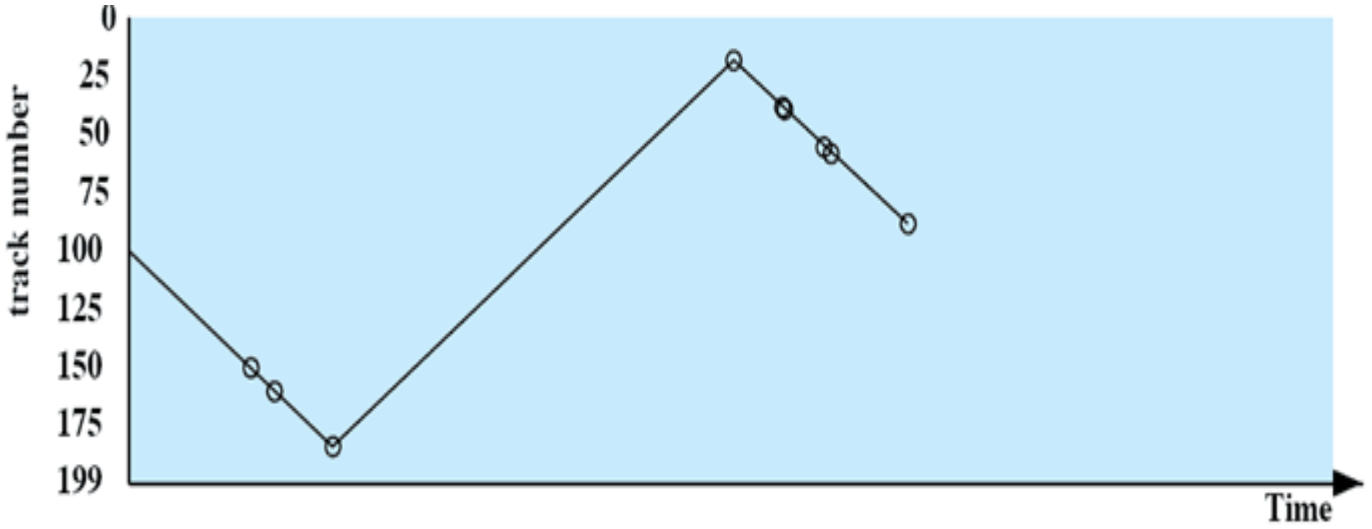
Note : Refer the diagram from notes or in the video

# C-SCAN

- Restricts scanning to one direction only
- When the last track has been visited in one direction, the arm is returned to the opposite end of the disk and the scan begins again
- Provides a more uniform wait time than SCAN.
- The head moves from one end of the disk to the other. servicing requests as it goes. When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip.
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one.

# C-LOOK

- Refinement of C-SCAN scheduling algorithm is C-LOOK scheduling Policy.
- Arm goes only as far as the last request in each direction, then it reverses the direction immediately, without going all the way to the end of the disk.



# Performance Compared

## Comparison of Disk Scheduling Algorithms

| (a) FIFO (starting at track 100) | | (b) SSTF (starting at track 100) | | (c) SCAN (starting at track 100, in the direction of increasing track number) | | (d) C-SCAN (starting at track 100, in the direction of increasing track number) | |
|---|---|---|---|---|---|---|---|
| **Next track accessed** | **Number of tracks traversed** | **Next track accessed** | **Number of tracks traversed** | **Next track accessed** | **Number of tracks traversed** | **Next track accessed** | **Number of tracks traversed** |
| 55 | 45 | 90 | 10 | 150 | 50 | 150 | 50 |
| 58 | 3 | 58 | 32 | 160 | 10 | 160 | 10 |
| 39 | 19 | 55 | 3 | 184 | 24 | 184 | 24 |
| 18 | 21 | 39 | 16 | 90 | 94 | 18 | 166 |
| 90 | 72 | 38 | 1 | 58 | 32 | 38 | 20 |
| 160 | 70 | 18 | 20 | 55 | 3 | 39 | 1 |
| 150 | 10 | 150 | 132 | 39 | 16 | 55 | 16 |
| 38 | 112 | 160 | 10 | 38 | 1 | 58 | 3 |
| 184 | 146 | 184 | 24 | 18 | 20 | 90 | 32 |
| **Average seek length** | 55.3 | **Average seek length** | 27.5 | **Average seek length** | 27.8 | **Average seek length** | 35.8 |

# Multiple Disks

- Disk I/O performance may be increased by spreading the operation over multiple read/write heads
  - Or multiple disks
- Disk failures can be recovered if parity information is stored

# Disk Cache

- Buffer in main memory for disk sectors
- Contains a copy of some of the sectors on the disk
- When an I/O request is made for a particular sector,
  - a check is made to determine if the sector is in the disk cache.
- A number of ways exist to populate the cache

# Selecting a Disk-Scheduling Algorithm

- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk.
- Performance depends on the number and types of requests.
- Requests for disk service can be influenced by the file-allocation method.
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary.
- Either SSTF or LOOK is a reasonable choice for the default algorithm.

# MS-DOS Disk Layout