



MONASH
University

MONASH
INFORMATION
TECHNOLOGY

FIT2100 Semester 2 2019

Lecture 3:
I/O Management and Hard drives
(Reading: Stallings, Chapter 11)

WEEK 3



Lecture 10: Learning Outcomes

- Upon the completion of this lecture, you should be able to:
 - Describe the key categories of I/O devices
 - Describe the organisation of the I/O function
 - Discuss various techniques used for performing I/O and I/O buffering
 - Discuss hard drives and various disk scheduling algorithms

What are the different categories of
I/O device?

I/O Devices: Categories

- External devices that engage in I/O with computer systems can be grouped into three categories:

Human readable:

- Suitable for communicating with the computer users
- Printers, terminals, video display, keyboards, mouse

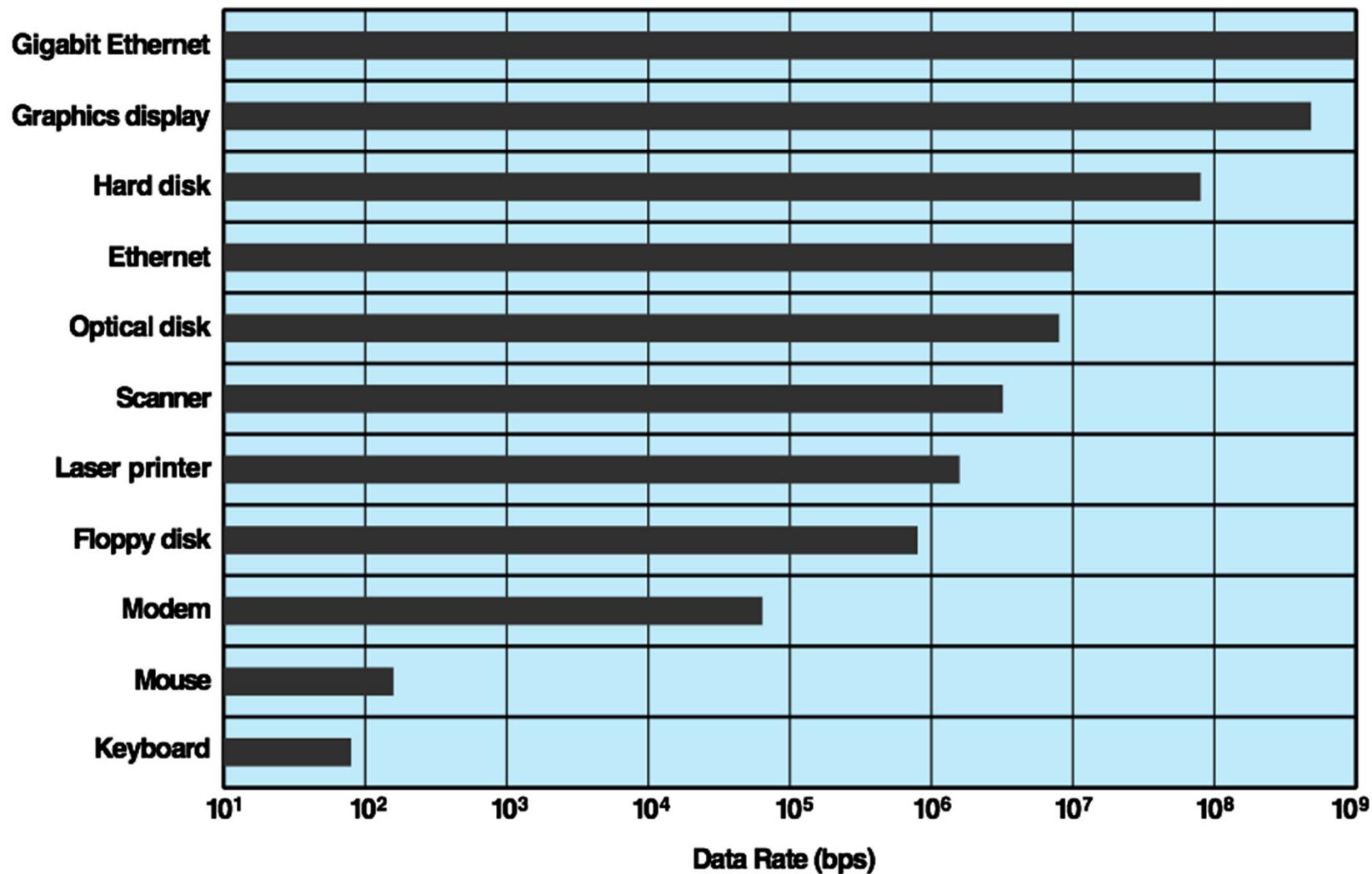
Machine readable:

- Suitable for communicating with electronic equipment
- Disk drives, USB keys, sensors, controllers, actuators

Communication:

- Suitable for communicating with remote devices
- Modems, digital line drivers, wireless interface cards

I/O Devices: Data Rates



What are the three techniques for performing I/O?

I/O: Techniques

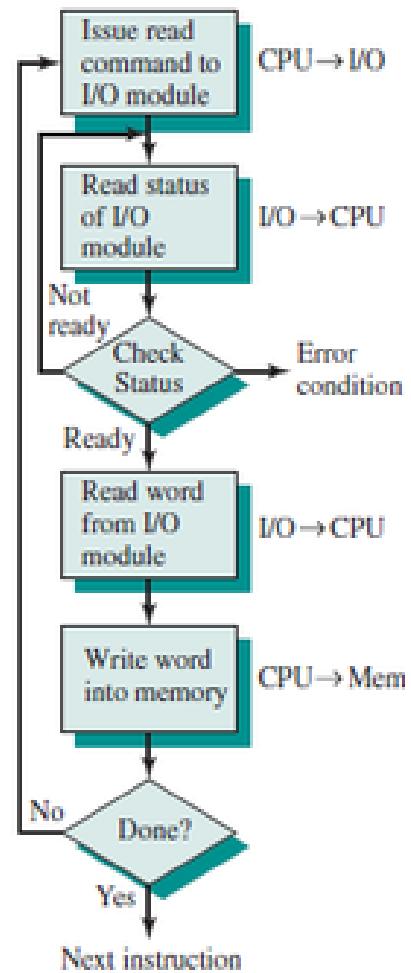
- ❑ Programmed I/O:
 - The processor issues an I/O command on behalf of a process to an I/O module — that process then *busy waits* for the I/O operation to be completed before proceeding.
- ❑ Interrupt-driven I/O:
 - The processor issues an I/O command on behalf of a process.
 - *Non-blocking* — the processor continues to execute instructions from the process that issued the I/O command.
 - *Blocking* — the next instruction the processor executes is from the OS, which will put the current process in a blocked state and schedule another process.
- ❑ Direct Memory Access (DMA):
 - DMA module controls the exchange of data between main memory and an I/O module (through the system bus).

Programmed I/O

- ❑ The processor issues an I/O command on behalf of a process to an I/O module; that process then *busy waits* for the I/O operation to be completed before proceeding.
- ❑ Basic operations of Programmed I/O:
 - Processor requests an I/O operation
 - I/O module performs the operation
 - I/O module sets status bits
 - Processor checks status bits periodically
 - I/O module **does not inform or interrupt** Processor directly

Interrupt-Driven I/O

- The processor issues an I/O command on behalf of a process.
- Two possibilities: the I/O instruction from the process is either **blocking** or **non-blocking**
- Basic operations of **Interrupt-driven I/O**:
 - Processor requests a read operation
 - I/O module gets data from the I/O device while Processor does other work
 - I/O module **interrupts** Processor **when data is ready for transfer**
 - Processor requests the data from I/O module
 - I/O module transfers the data to Processor

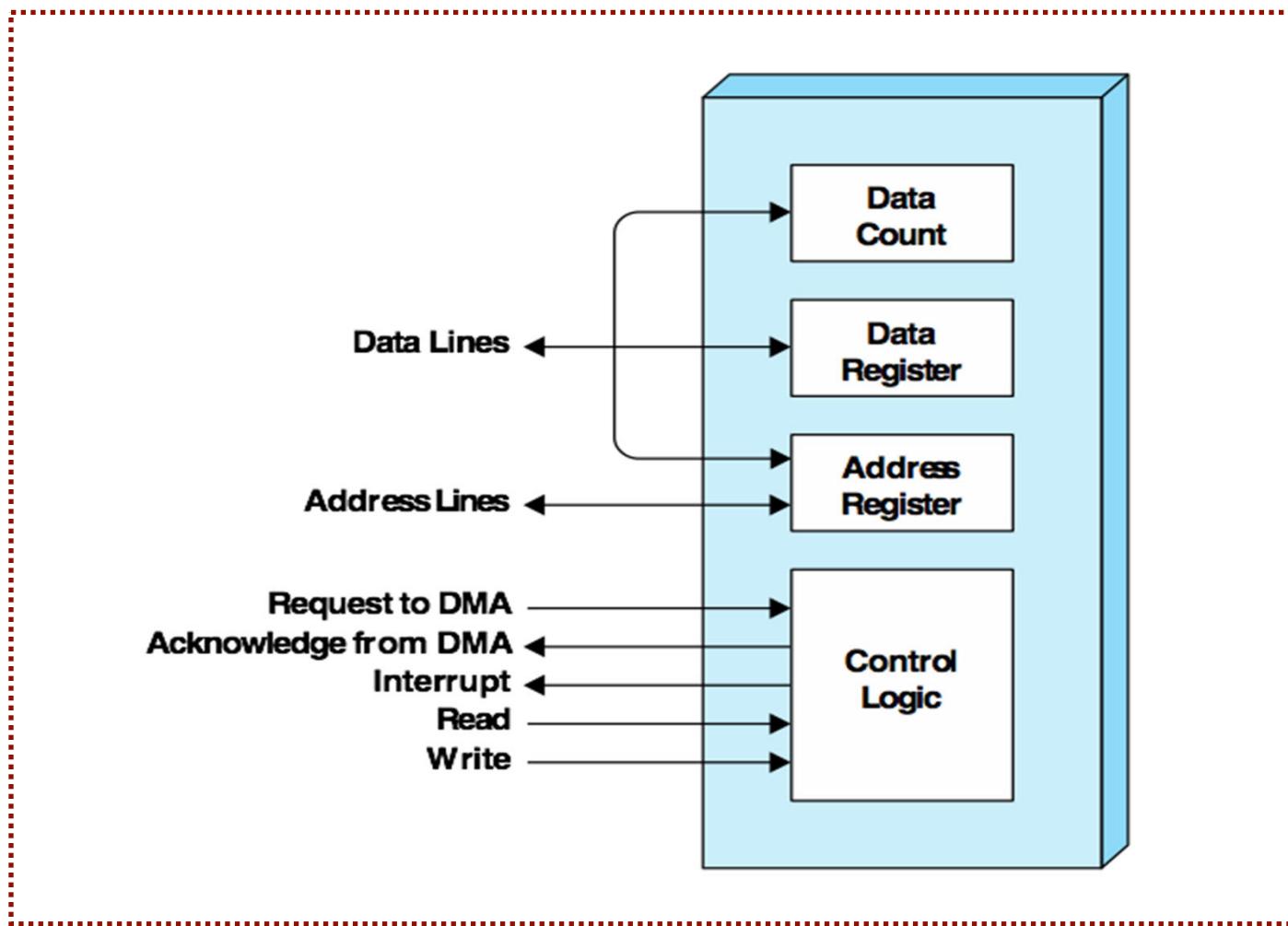


Direct Memory Access (DMA)

- DMA module controls the exchange of data between main memory and an I/O module (over the system bus).
- Basic operations of DMA:
 - Processor sends DMA a request for the transfer of a block of data
 - Processor continues with other work
 - DMA is responsible for transferring data to and from memory
 - Processor is *interrupted* only after the entire block has been transferred

How does direct memory access (DMA) work?

DMA: Block Diagram



DMA: How does it work?

When the processor wishes to read or write data it issues a command to the DMA module containing:

- whether a read or write is requested
- the address of the I/O device involved
- the starting location in memory to read/write
- the number of words to be read/written

DMA: How does it work?

- ❑ DMA transfers the entire block of data directly to and from memory **without going through the processor**
- ❑ When the transfer is complete, DMA sends an interrupt signal to the processor
- ❑ Processor is involved only at the beginning and end of the transfer
- ❑ More **efficient** than interrupt-driven or programmed I/O

Summary: Techniques for Performing I/O

	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)

What is I/O buffering?

I/O Buffering

- ❑ Perform **input transfers in advance** of requests being made and perform **output transfers some time** after the request is made.

Block-oriented device

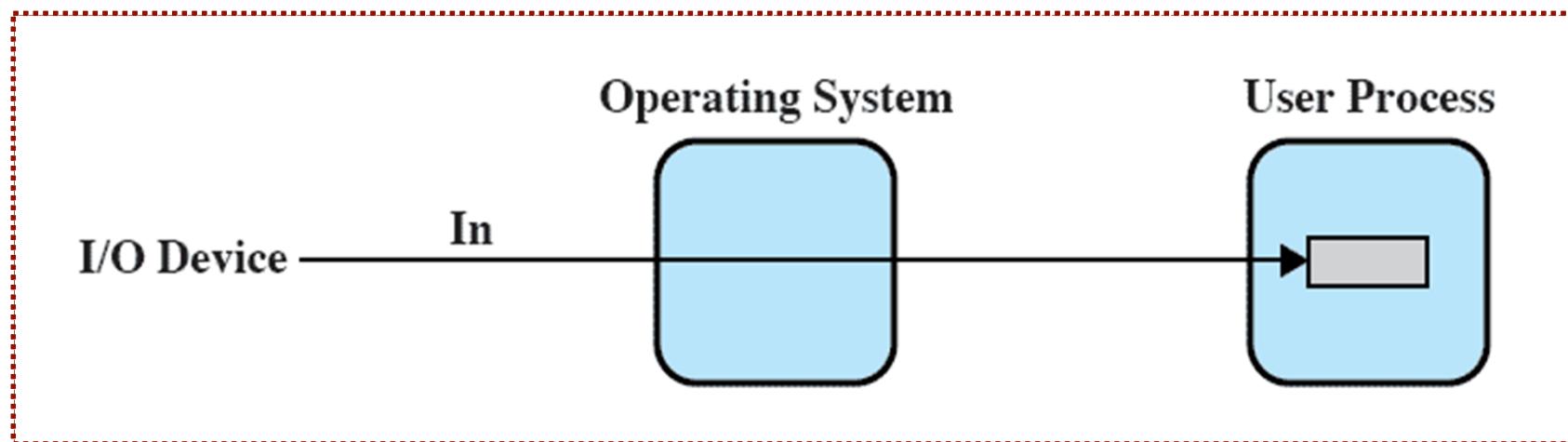
- stores information in blocks that are usually of fixed size
- transfers are made one block at a time
- possible to reference data by its block number
- disks and USB keys are examples

Stream-oriented device

- transfers data in and out as a stream of bytes
- no block structure
- terminals, printers, communications ports, and most other devices that are not secondary storage are examples

I/O Buffering: No Buffer

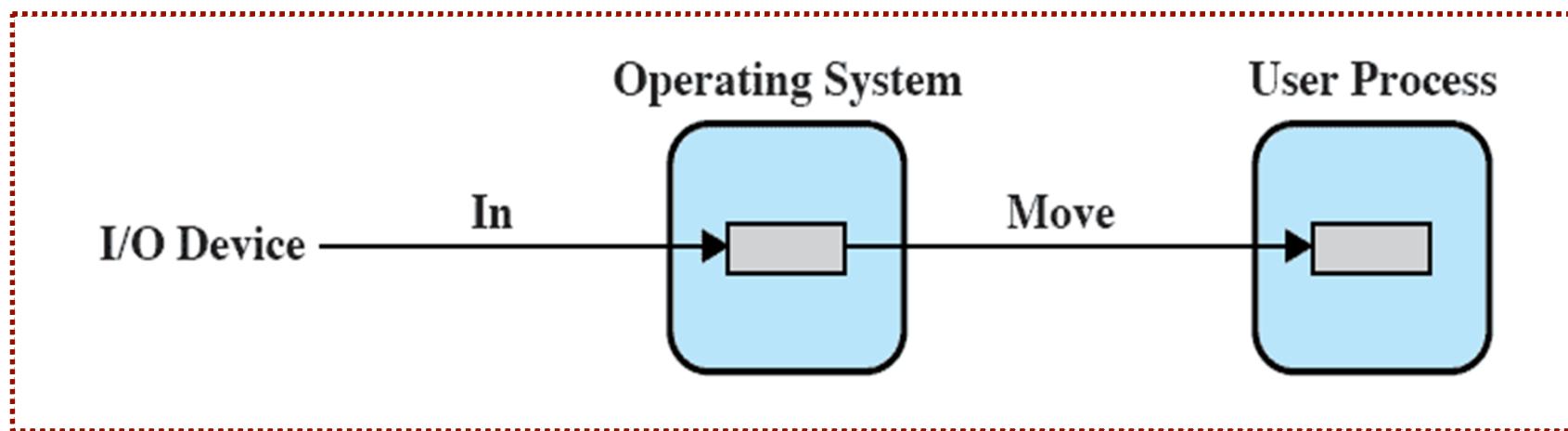
- Without a buffer, OS directly accesses the device when it needs to.



I/O Buffering: Single Buffer

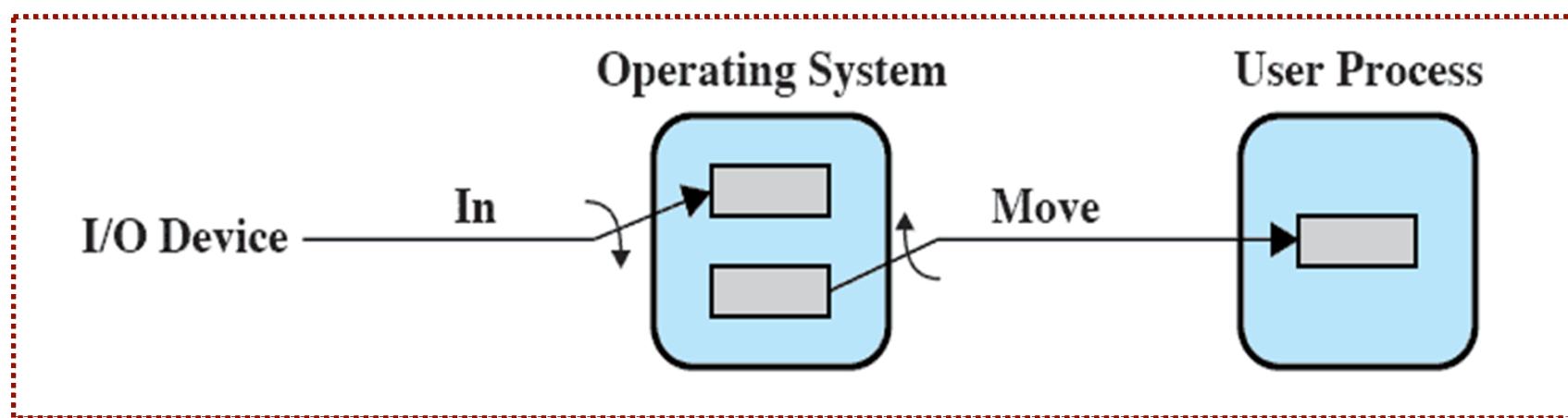
- When a user process issues an I/O request, OS assigns a buffer in main memory for an I/O request.

Reading ahead or anticipated input



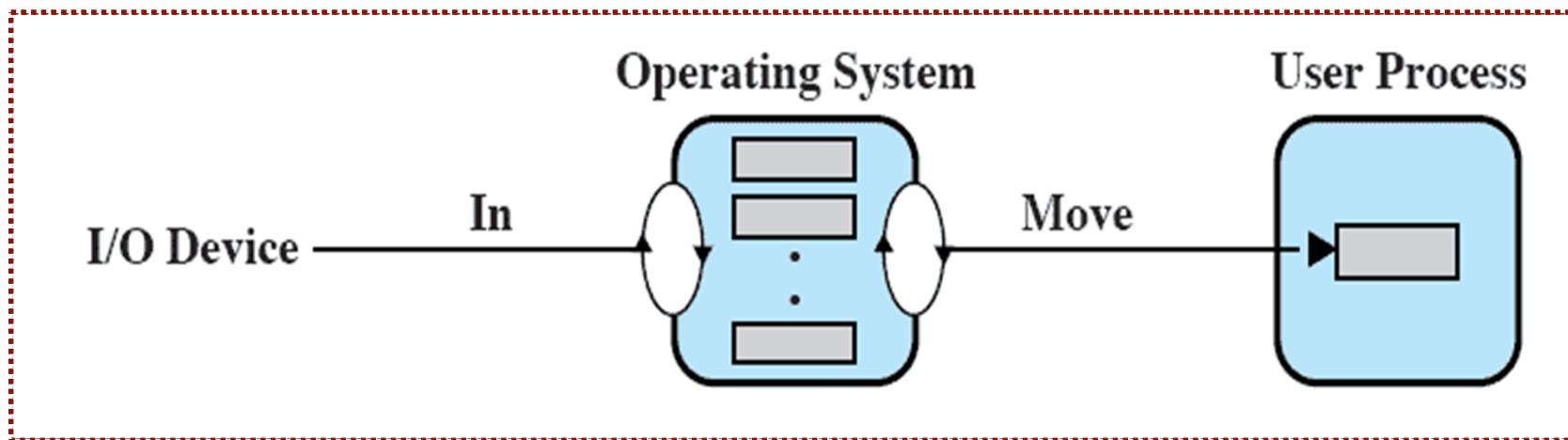
I/O Buffering: Double Buffer

- ❑ 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
- ❑ Also known as **buffer swapping**

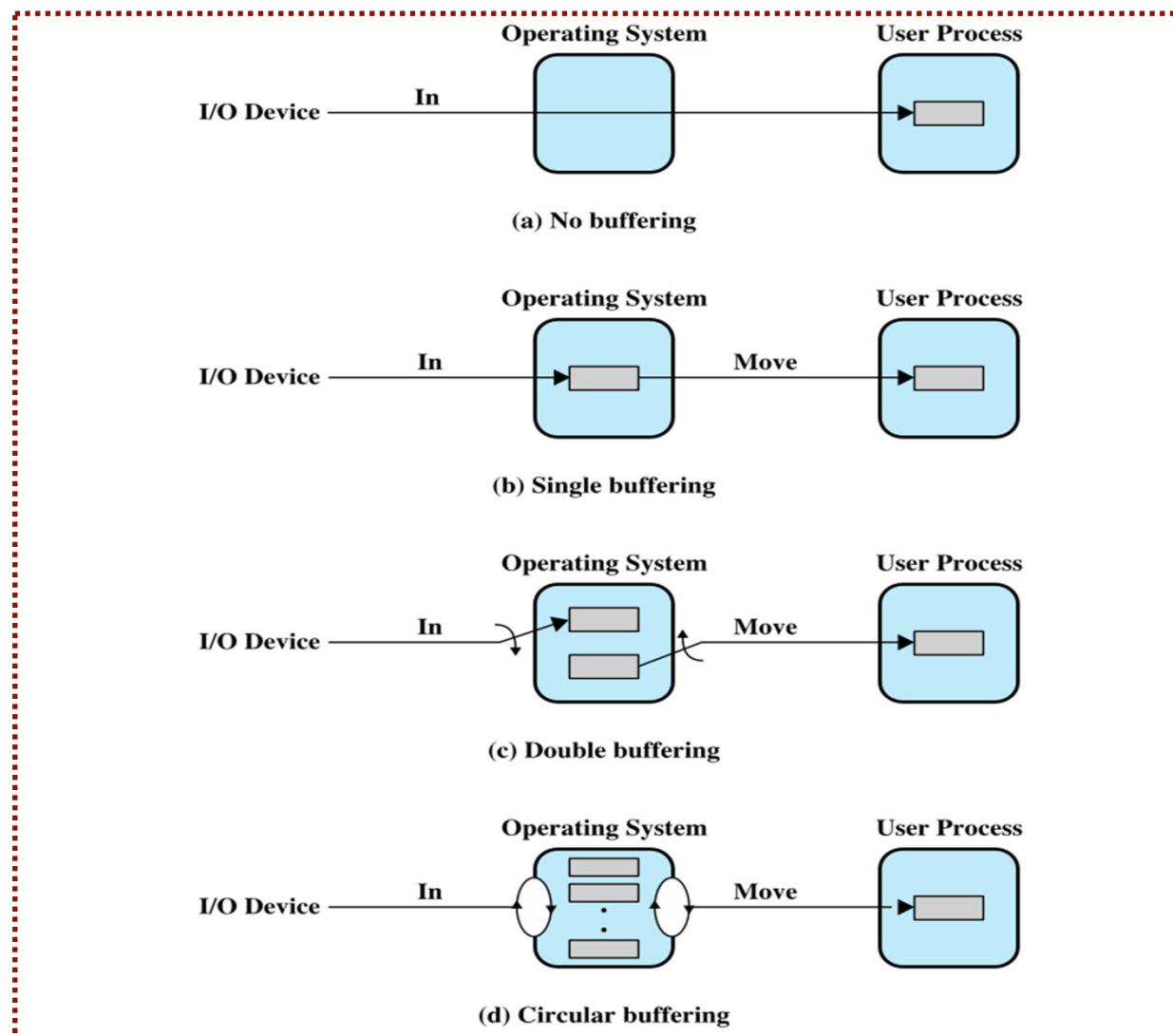


I/O Buffering: Circular Buffer

- ❑ Two or more buffers are used
- ❑ Each individual buffer is one unit in a circular buffer
- ❑ Used when I/O operation must keep up with the process



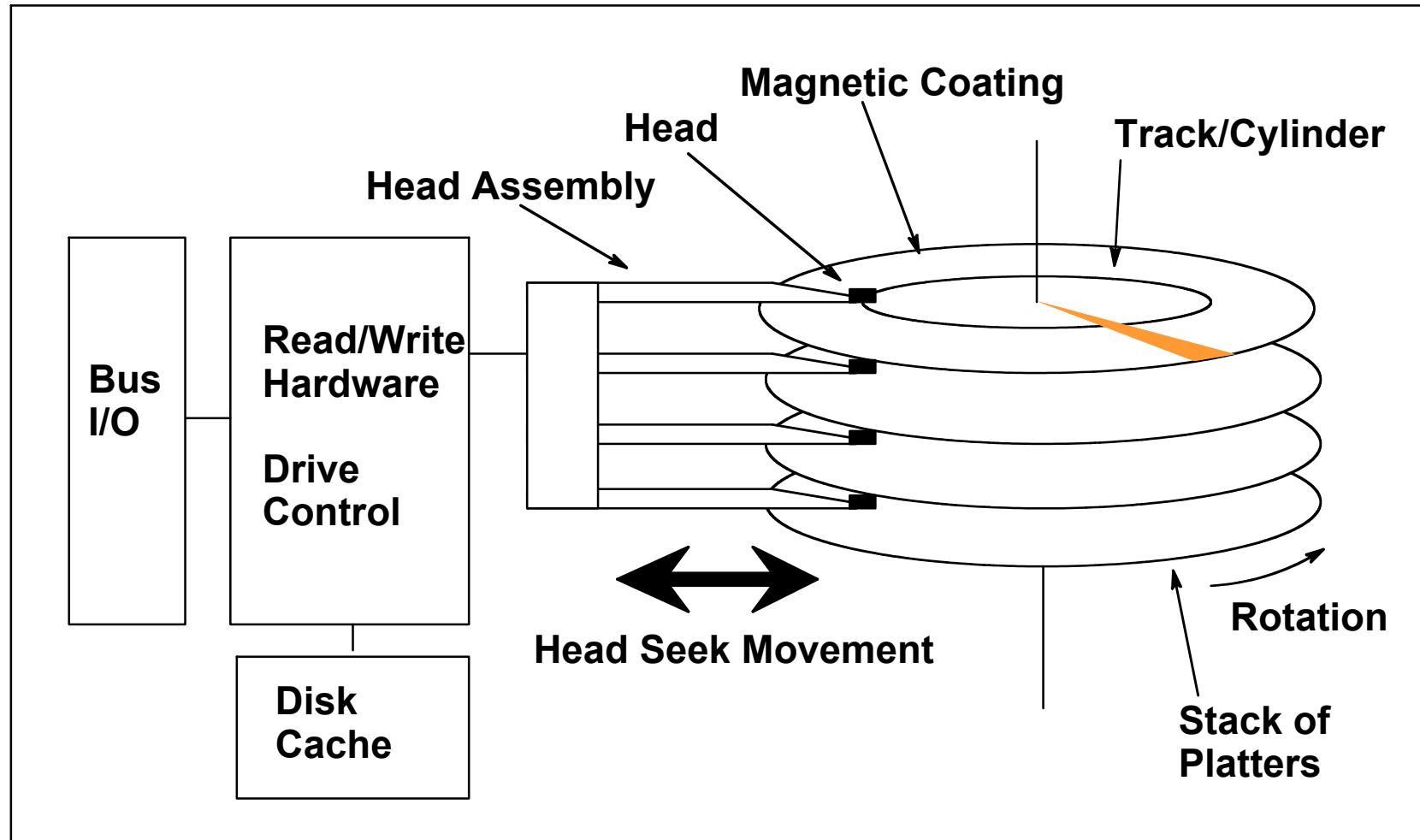
Summary: I/O Buffering Schemes (Input)



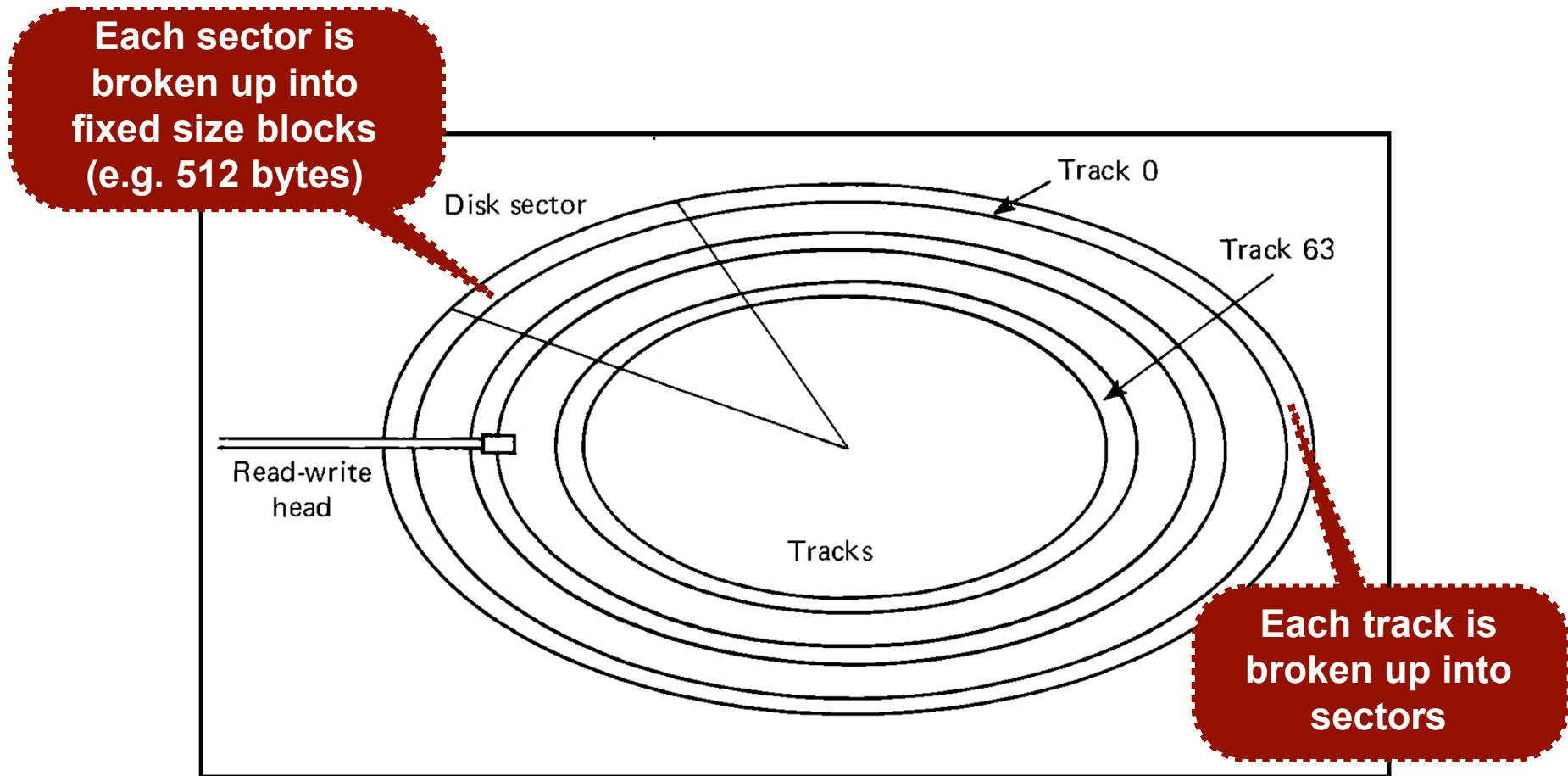
How does a hard drive work?

SATA hard drive with cover removed





Magnetic Disk: Movable Head System



[Source: <http://www.jklp.org/profession/books/mix/c05.html>]

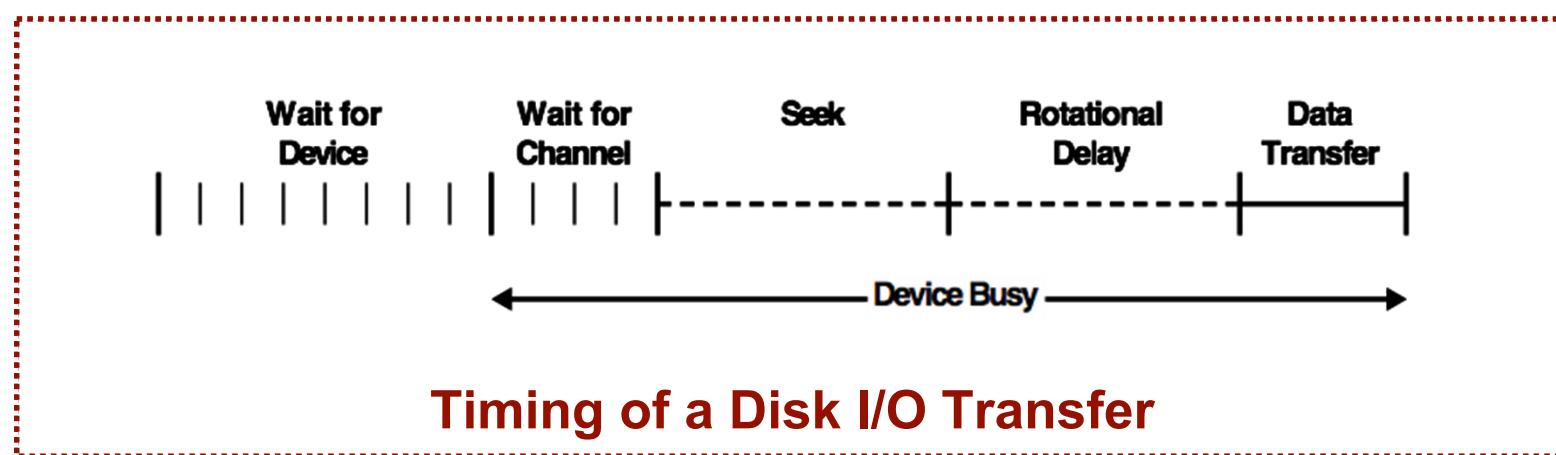
Read/Write: Disk Head Positioning

- When the disk drive is operating, the disk is rotating at constant speed
- To read or write: **the head must be positioned at the desired track and at the beginning of the desired sector on that track**
- **Track selection:** involves moving the head in a movable-head system or electronically selecting one head on a fixed-head system
- **Seek time:** the time it takes to position the head (disk arm) at the track on a movable-head system
- **Rotational delay:** the time it takes for the beginning of the sector to reach the head
- **Access time:** the sum of the seek time (if any) and the rotational delay

How does disk scheduling work?

Disk Performance: Parameters

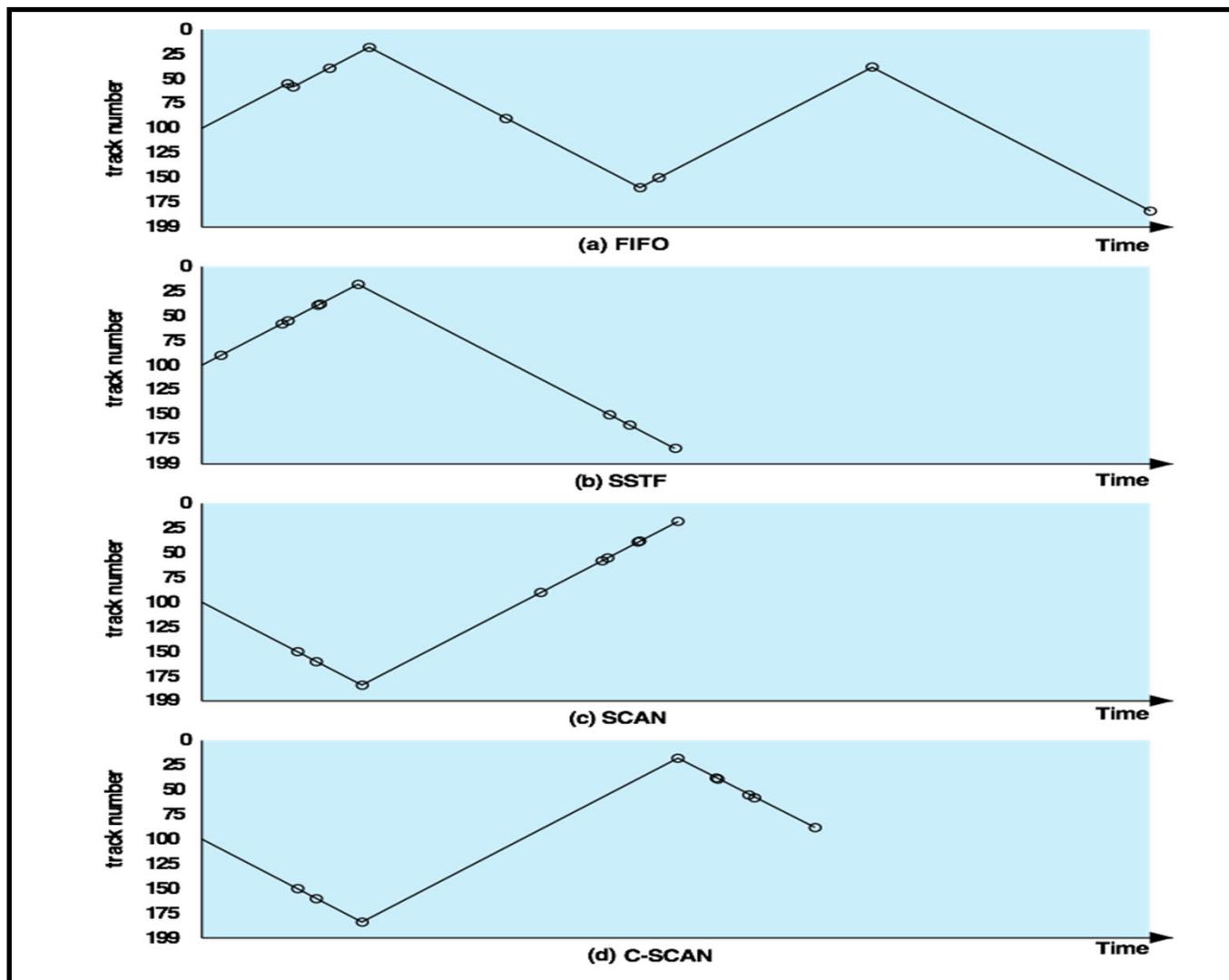
- The actual details of disk I/O operation depend on:
 - Computer system
 - Operating system
 - Nature of the I/O channel and disk controller hardware



Disk Scheduling: Algorithms

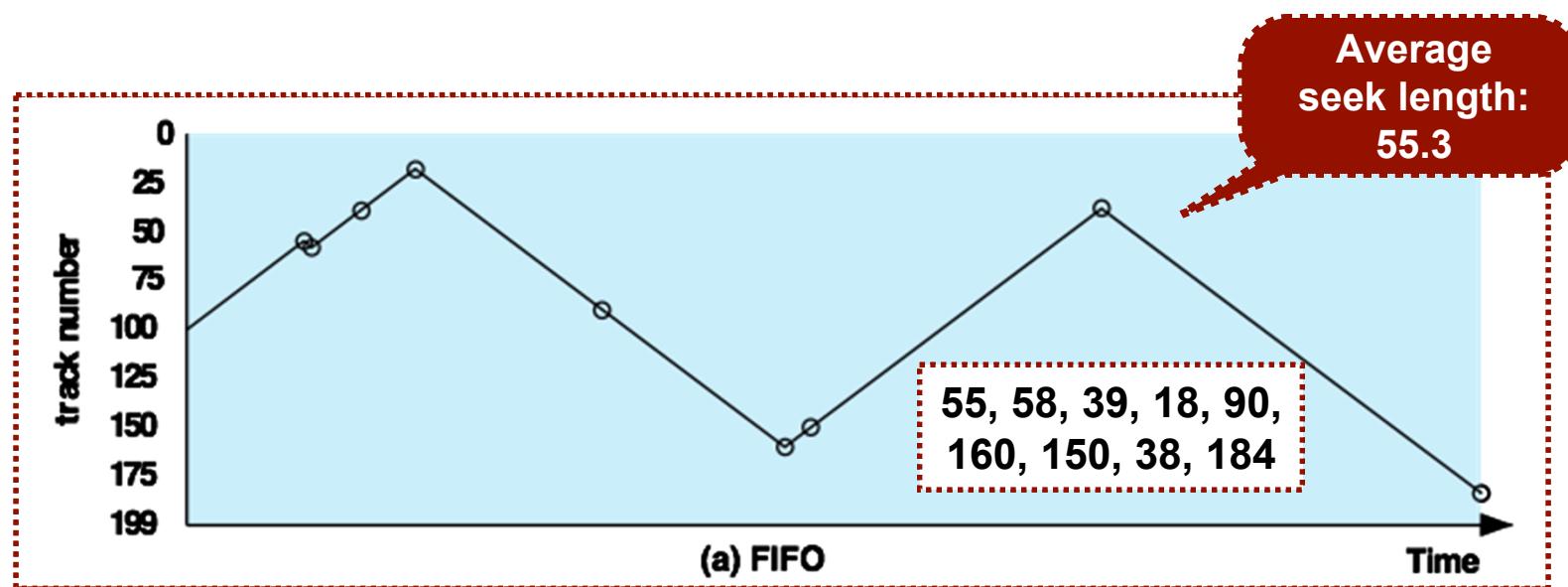
Name	Description	Remarks
Selection according to requestor		
Random	Random scheduling	For analysis and simulation
FIFO	First in first out	Fairest of them all
PRI	Priority by process	Control outside of disk queue management
LIFO	Last in first out	Maximize locality and resource utilization
Selection according to requested item		
SSTF	Shortest service time first	High utilization, small queues
SCAN	Back and forth over disk	Better service distribution
C-SCAN	One way with fast return	Lower service variability
N-step-SCAN	SCAN of N records at a time	Service guarantee
FSCAN	N-step-SCAN with $N =$ queue size at beginning of SCAN cycle	Load sensitive

Disk Scheduling: Comparison of Algorithms



Disk Scheduling: First In First Out (FIFO)

- ❑ Processes in sequential order
- ❑ Fair to all processes
- ❑ Approximates random scheduling in performance if there are many processes competing for the disk

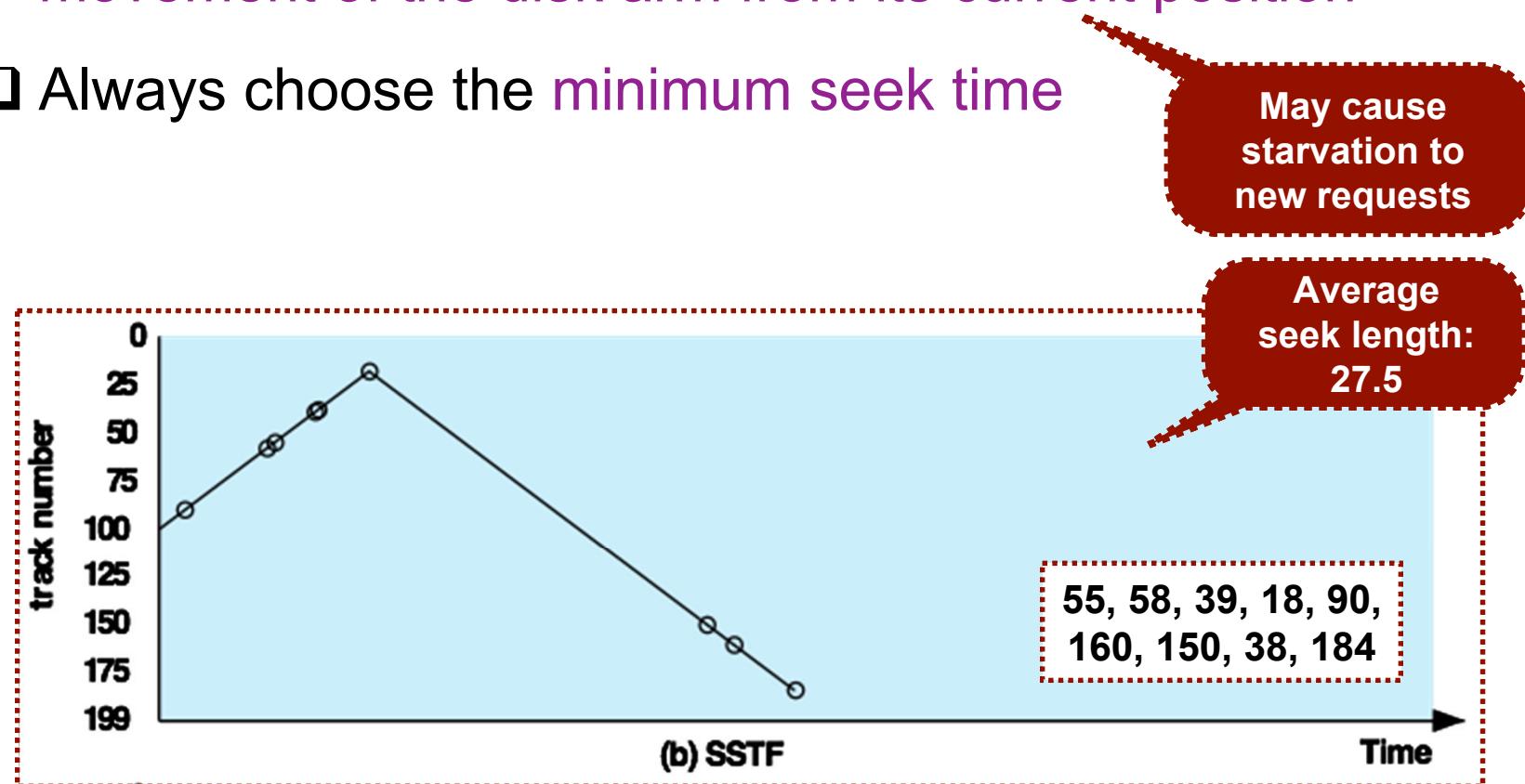


Disk Scheduling: Priority (PRI)

- ❑ Control of the scheduling is outside the control of disk management software
- ❑ **Goal:** not to optimise disk utilisation but **to meet other OS objectives**
- ❑ Short batch jobs and interactive jobs are given higher priority
- ❑ Provides **good interactive response time**
- ❑ Longer jobs may have to wait an **excessively long time**
- ❑ Countermeasures on the part of users

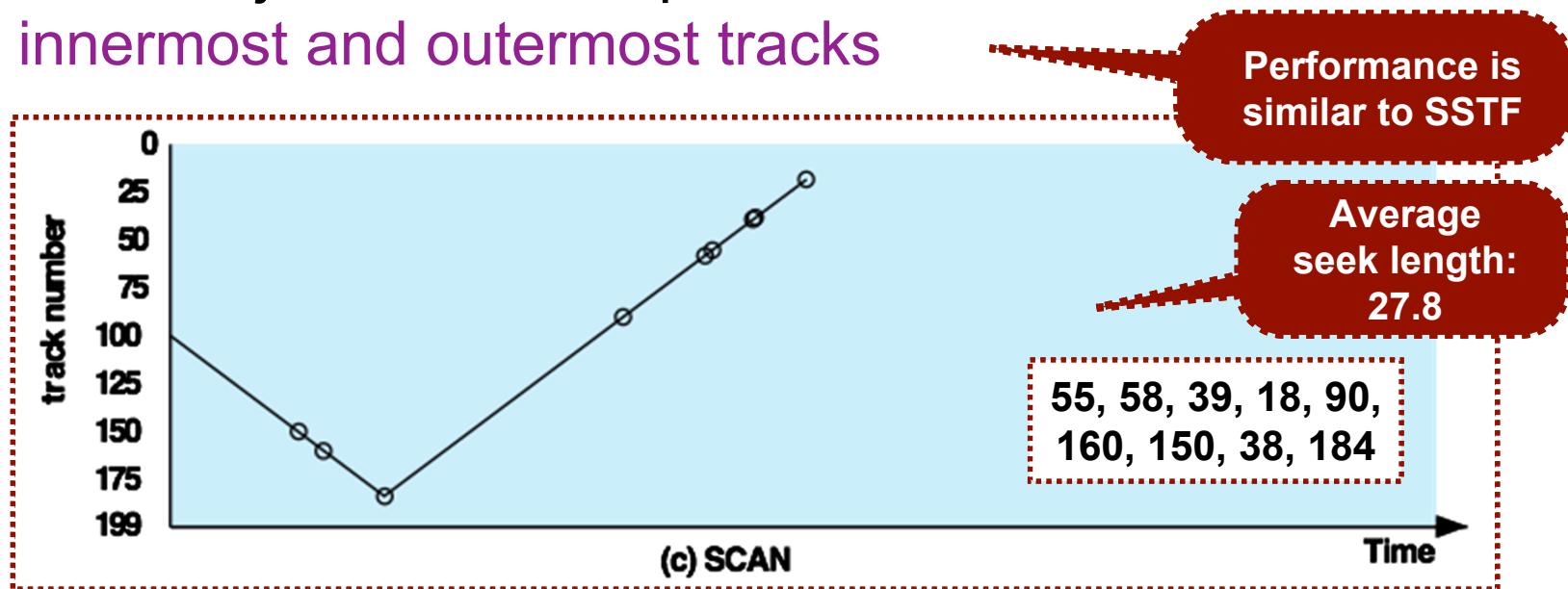
Disk Scheduling: Shortest Service Time First (SSTF)

- Select the disk I/O request that requires the least movement of the disk arm from its current position
- Always choose the minimum seek time



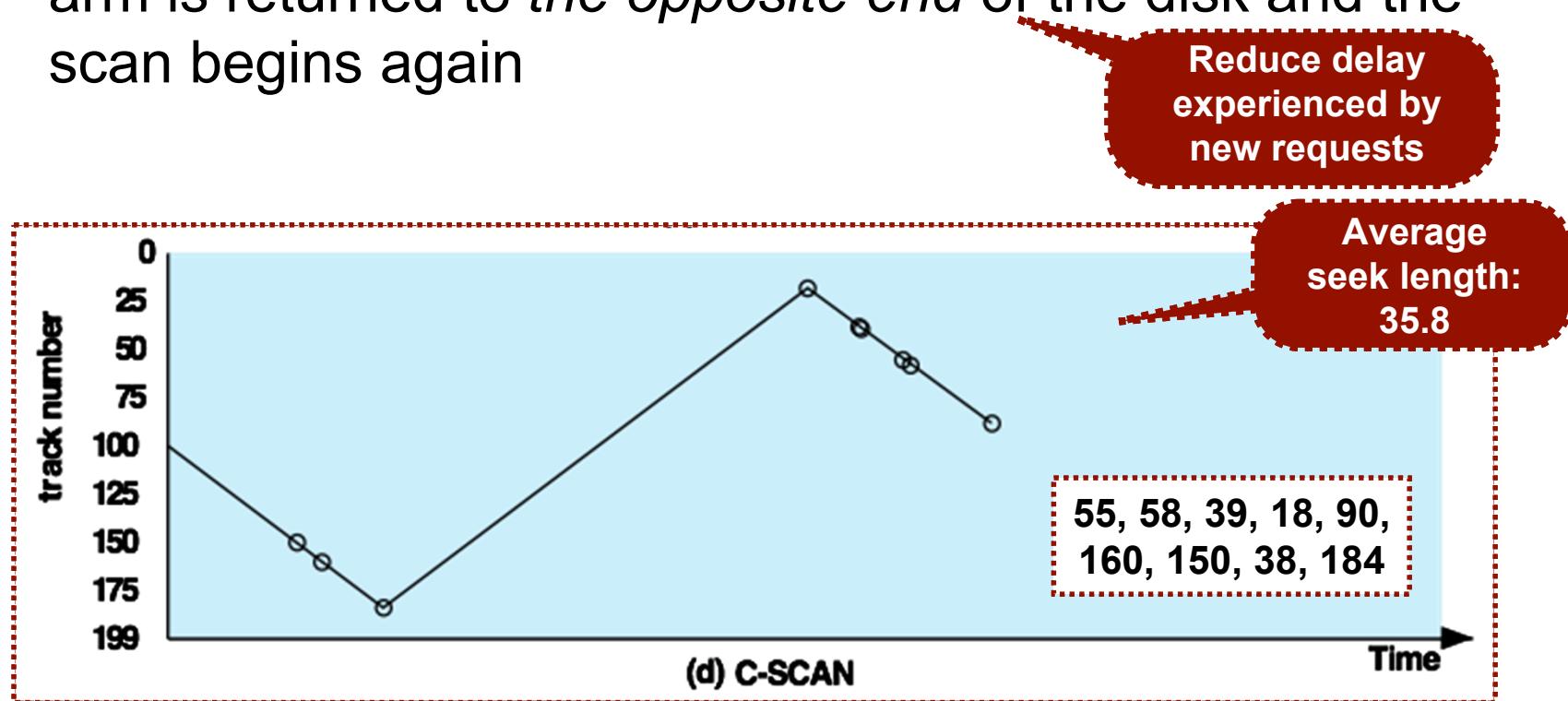
Disk Scheduling: SCAN (Elevator Algorithm)

- The head (disk arm) moves in **one direction only**
- Satisfies all outstanding requests until it reaches the last track in that direction then the direction is *reversed*
- Favours jobs whose request are for **tracks nearest to both innermost and outermost tracks**



Disk Scheduling: C-SCAN (Circular 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



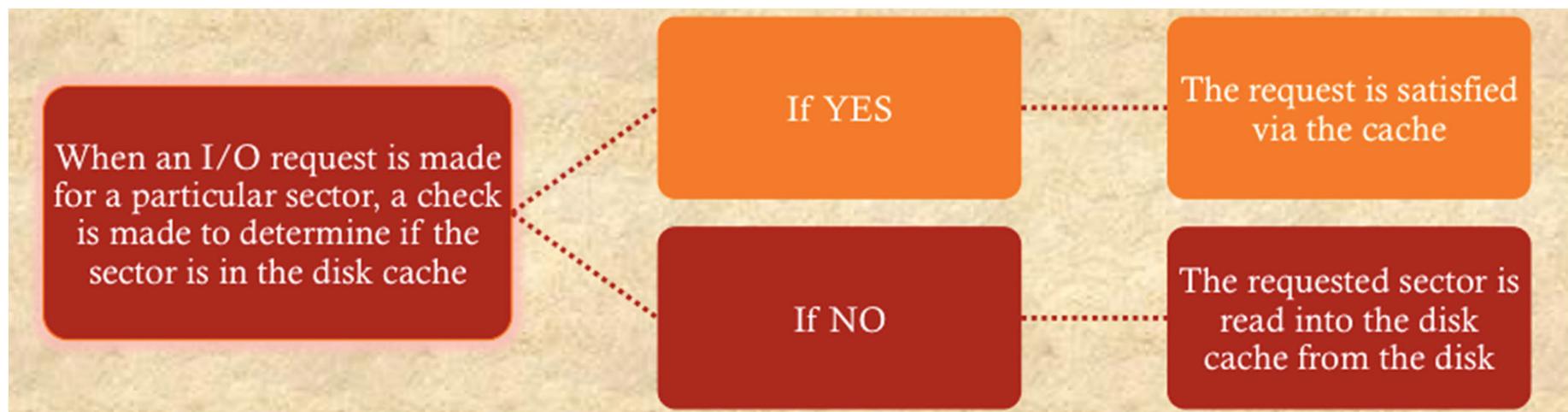
What is a disk cache?*

*Reading from Stallings: Chapter 11 (11.7)

Disk Cache

- ❑ **Cache memory:** used to apply to a memory that is smaller and faster than main memory and that is interposed between main memory and the processor
- ❑ Reduces average memory access time by exploiting the **principle of locality**
- ❑ **Disk cache:** a buffer in main memory for disk sectors
 - Newer hard drives also come with a level of disk cache memory built in.
- ❑ Contains a copy of some of the sectors on the disk

Disk Cache



Cache Block Replacement: Least Recently Used (LRU)

- ❑ Most commonly used algorithm that deals with the design issue of **replacement strategy**
- ❑ The block that has been in the cache **the longest with no reference** to it is replaced
- ❑ A **stack of pointers** reference the cache
 - **Most recently referenced block** is on the top of the stack
 - When a block is referenced or brought into the cache, it is placed on the top of the stack

Cache Block Replacement: Least Frequently Used (LFU)

- ❑ The block that has experienced the fewest references is replaced
- ❑ A counter is associated with each block
- ❑ The counter is incremented each time the block is accessed
- ❑ When replacement is required, the block with the smallest count is selected

I/O Management in Unix*

*Reading from Stallings: Chapter 11 (11.8)

WHAT IS A DRIVER?

- ❑ Program code embedded in operating system
 - In Linux, driver ‘modules’ can be loaded or unloaded from the kernel depending on the user’s needs
- ❑ Provides an interface for interacting with a hardware device
- ❑ In Linux, driver interfaces appear as virtual ‘files’ in the /dev/ directory
 - These ‘files’ can be written to or read from in order to access or modify hardware devices

Summary of Lecture 3

- ❑ I/O architecture is an interface between the computer system and the outside world.
- ❑ A key aspect of I/O is the use of buffers that are controlled by I/O utilities rather than by application processes.
- ❑ Buffering smoothes out the differences between the speeds.
- ❑ Disk I/O has the greatest impact on overall system performance.
- ❑ Two of the most widely used approaches are disk scheduling and the disk cache.
- ❑ A disk cache is a buffer, usually kept in main memory, that functions as a cache of disk block between disk memory and the rest of main memory.

Reading from Stallings: Chapter 11: 11.1-11.5;
11.7-11.8 (additional reading)