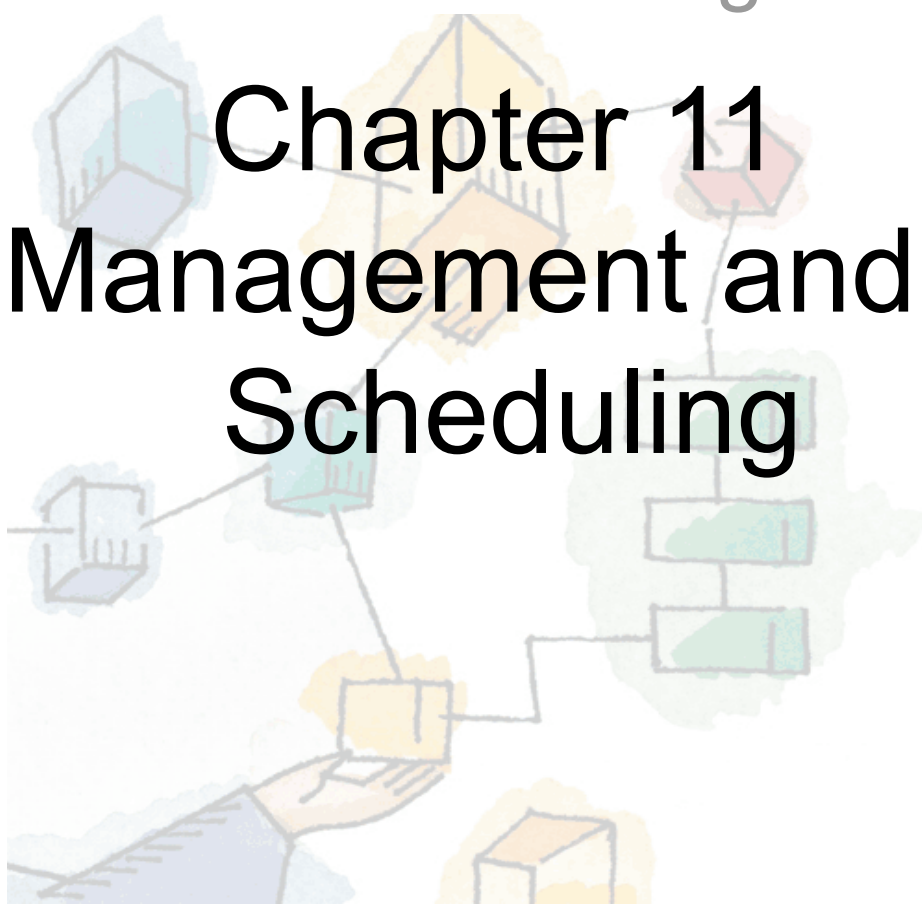


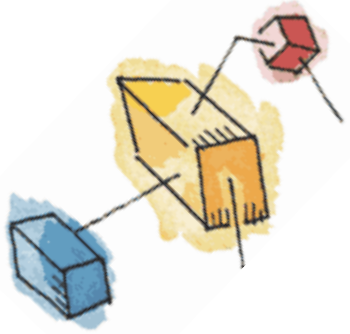
*Operating Systems:  
Internals and Design Principles, 6/E*  
William Stallings



# Chapter 11

## I/O Management and Disk Scheduling

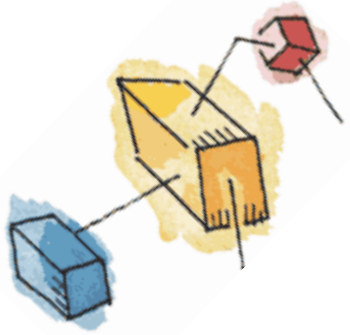
Patricia Roy  
Manatee Community College, Venice, FL  
©2008, Prentice Hall



# Categories of I/O Devices

- Human readable
  - Used to communicate with the user
  - Printers and terminals

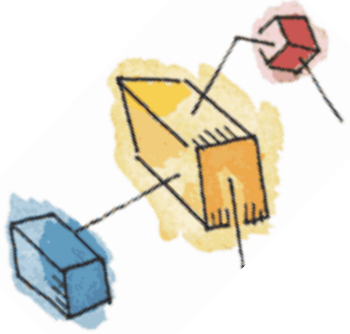




# Categories of I/O Devices

- Machine readable
  - Used to communicate with electronic equipment
  - Disk drives
  - USB keys
  - Sensors
  - Controllers
  - Actuators

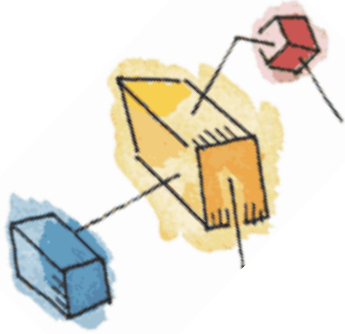




# Categories of I/O Devices

- Communication
  - Used to communicate with remote devices
  - Digital line drivers
  - Modems

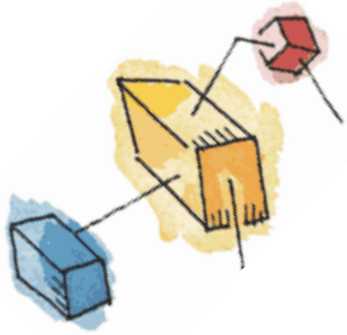




# Differences in I/O Devices

- Data rate
  - May be differences of several orders of magnitude between the data transfer rates

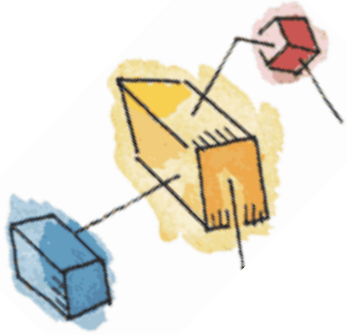




# Differences in I/O Devices

- Application
  - Disk used to store files requires file management software
  - Disk used to store virtual memory pages needs special hardware and software to support it
  - Terminal used by system administrator may have a higher priority

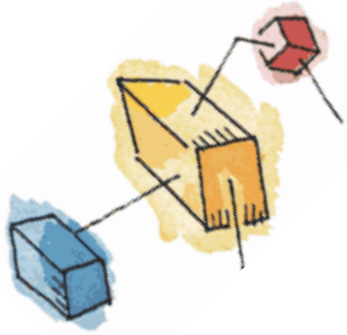




# Differences in I/O Devices

- Complexity of control
- Unit of transfer
  - Data may be transferred as a stream of bytes for a terminal or in larger blocks for a disk

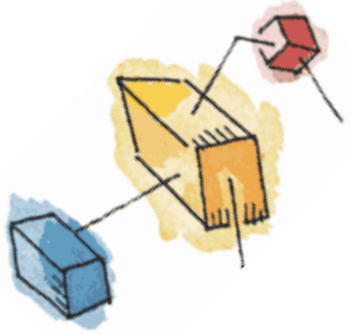




# Differences in I/O Devices

- Data representation
  - Encoding schemes
- Error conditions
  - Devices respond to errors differently





# I/O Device Data Rates

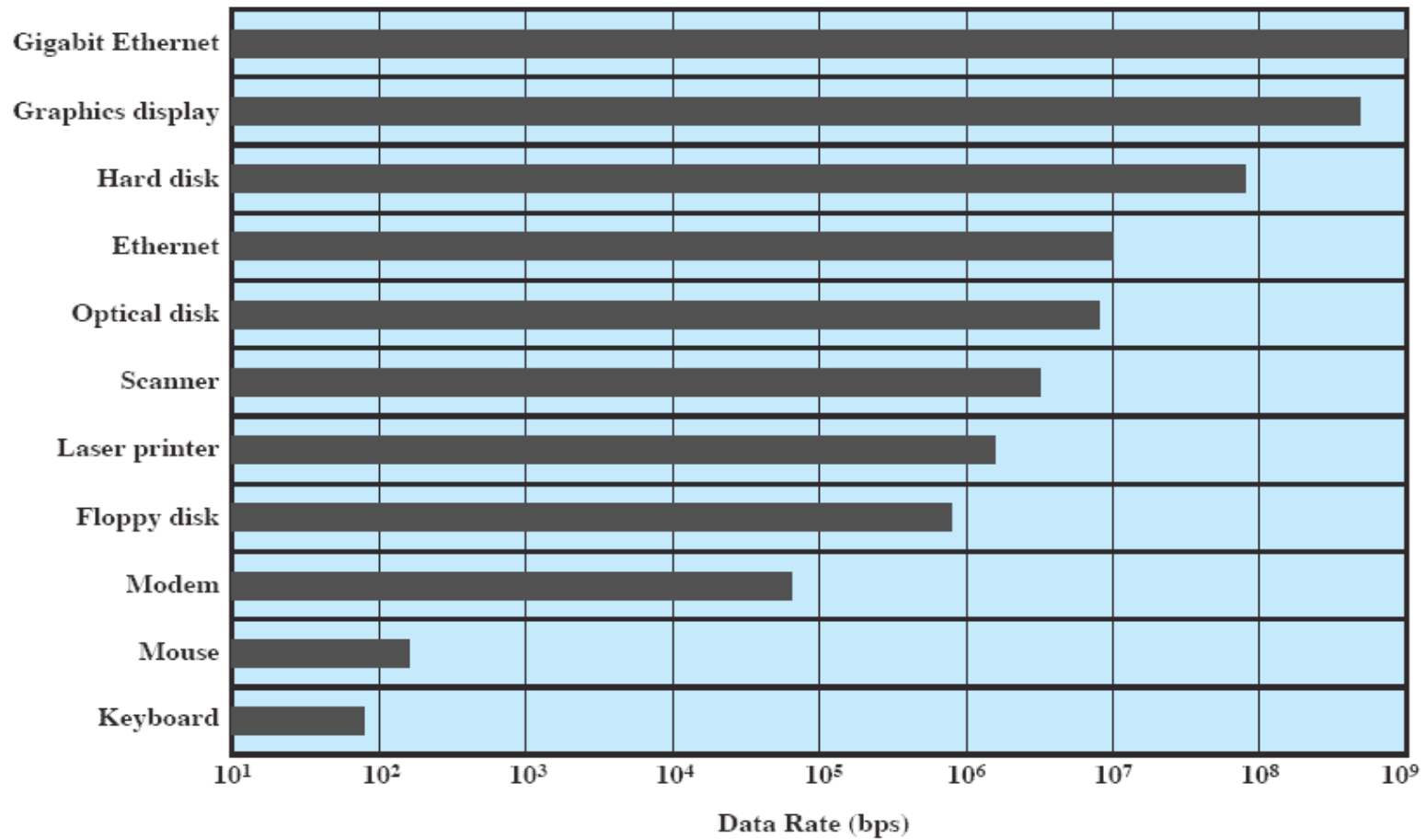
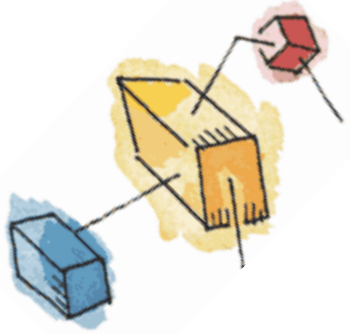


Figure 11.1 Typical I/O Device Data Rates

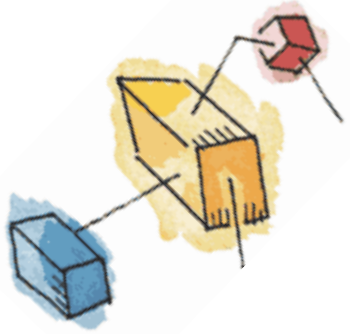




# Performing I/O

- Programmed I/O
  - Process is busy-waiting for the operation to complete
- Interrupt-driven I/O
  - I/O command is issued
  - Processor continues executing instructions



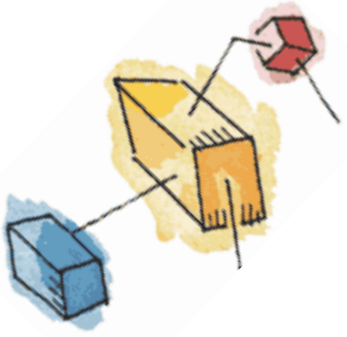


# Performing I/O

- Direct Memory Access (DMA)
  - DMA module controls exchange of data between main memory and the I/O device
  - Processor interrupted only after entire block has been transferred



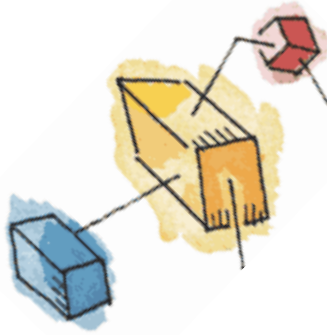
# Relationship Among Techniques



**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)

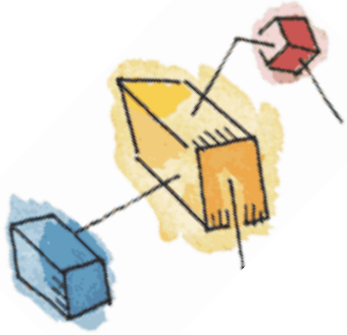




# Evolution of the I/O Function

- Processor directly controls a peripheral device
- Controller or I/O module is added
  - Processor uses programmed I/O without interrupts
  - Processor does not need to handle details of external devices

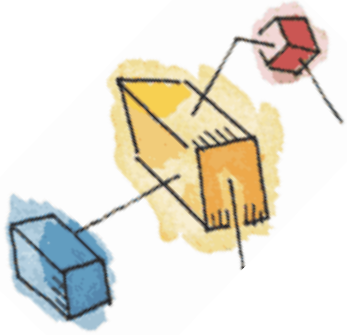




# Evolution of the I/O Function

- Controller or I/O module with interrupts
  - Processor does not spend time waiting for an I/O operation to be performed
- Direct Memory Access
  - Blocks of data are moved into memory without involving the processor
  - Processor involved at beginning and end only

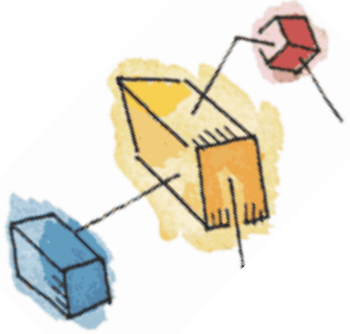




# Evolution of the I/O Function

- I/O module is a separate processor
- I/O processor
  - I/O module has its own local memory
  - Its a computer in its own right





# Direct Memory Address

- Processor delegates I/O operation to the DMA module
- DMA module transfers data directly to or from memory
- When complete DMA module sends an interrupt signal to the processor



# DMA

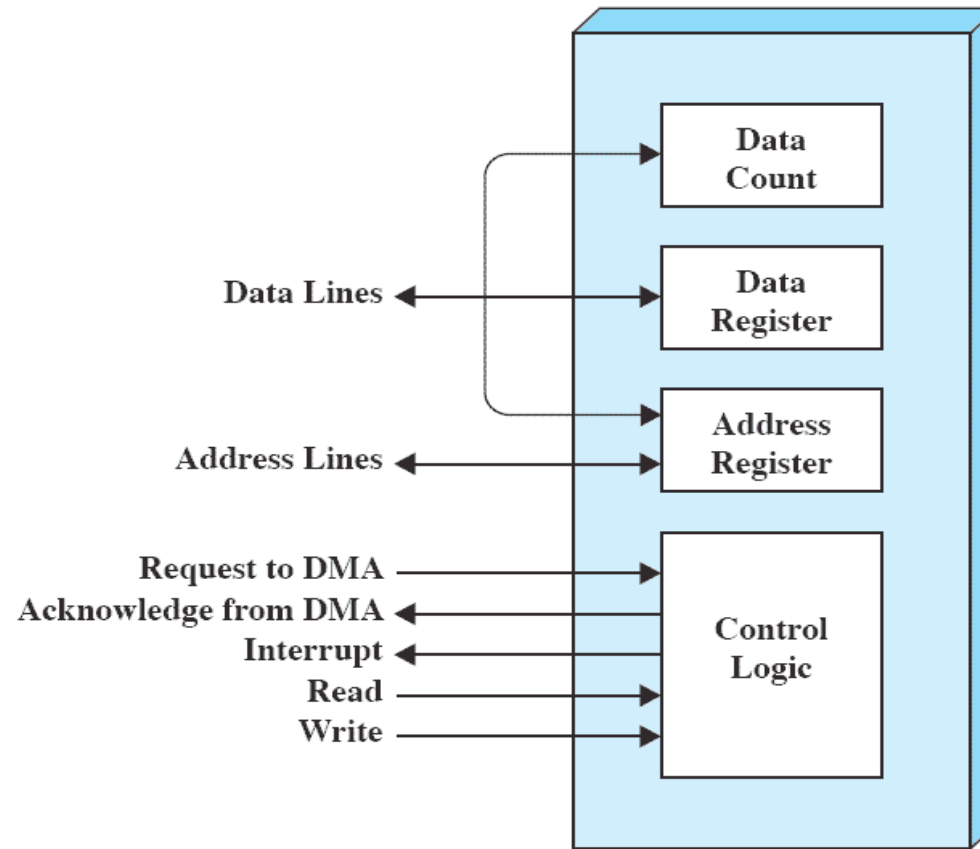
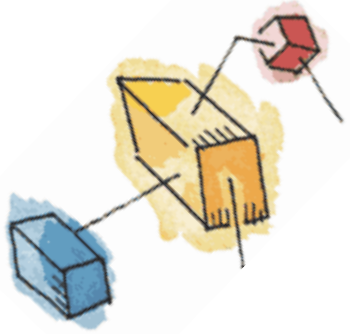
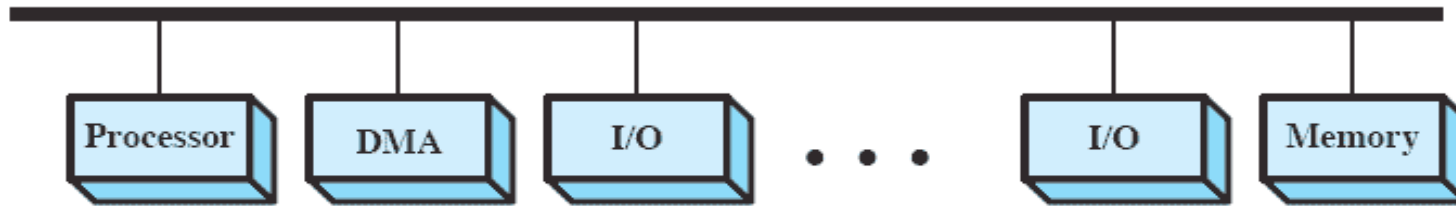


Figure 11.2 Typical DMA Block Diagram

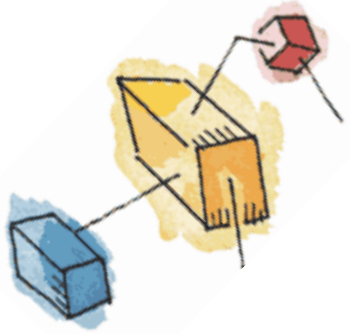


# DMA Configurations

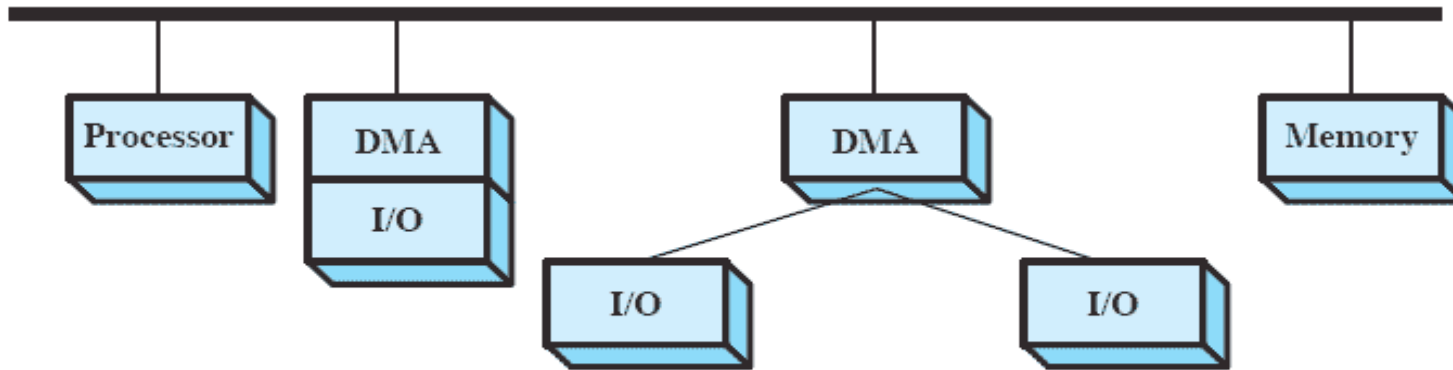


(a) Single-bus, detached DMA



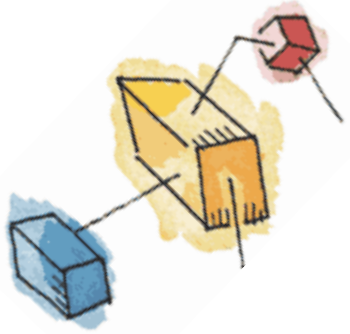


# DMA Configurations

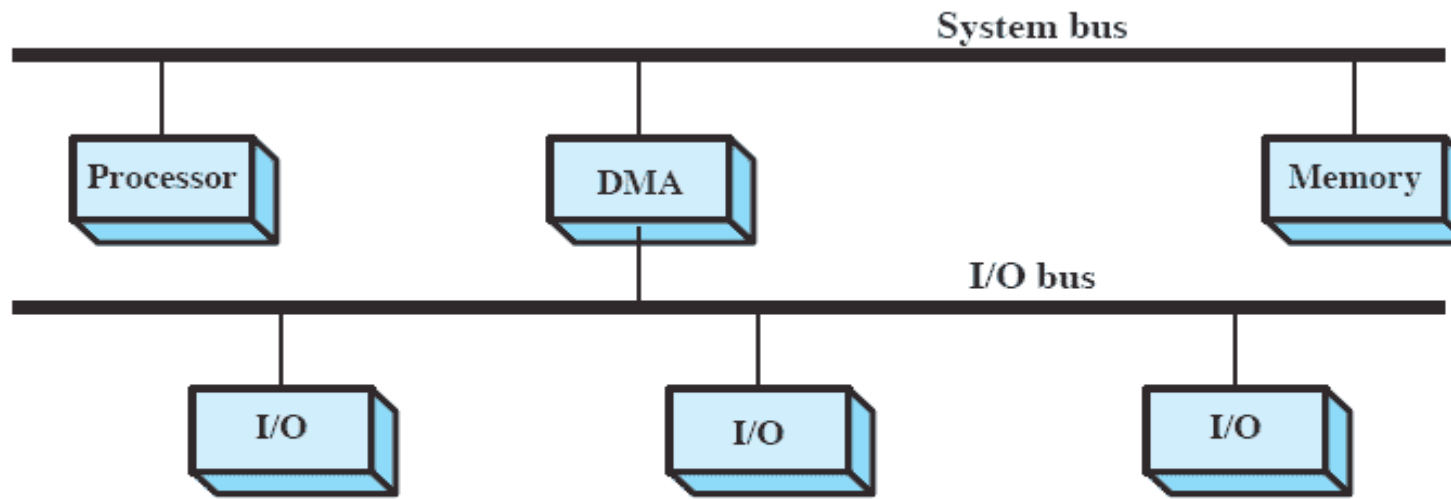


(b) Single-bus, Integrated DMA-I/O



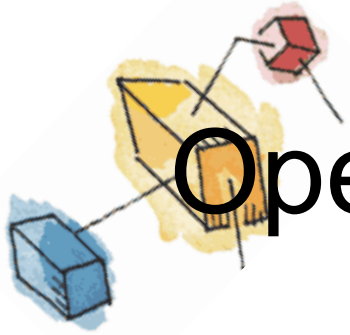


# DMA Configurations



(c) I/O bus

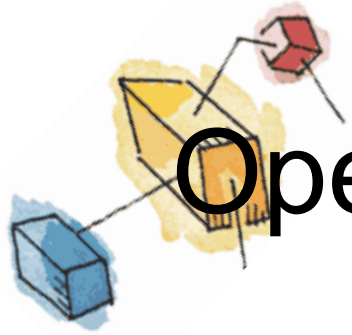




# Operating System Design Issues

- 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 is used to bring in additional Ready processes which is an I/O operation

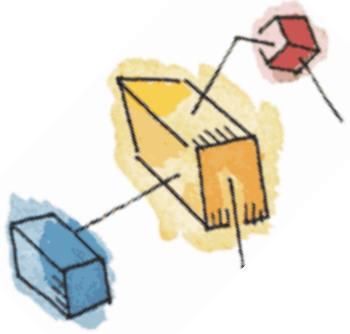




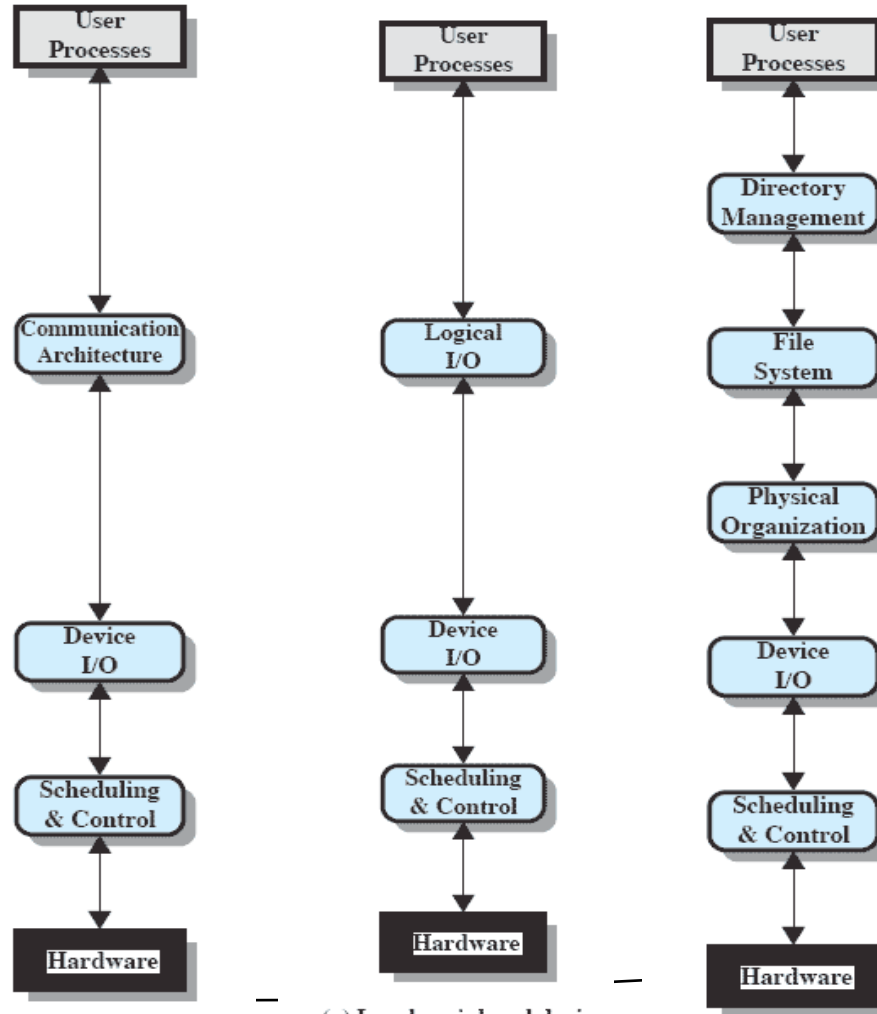
# Operating System Design Issues

- Generality
  - Desirable to handle all I/O devices in a uniform manner
  - Hide most of the details of device I/O in lower-level routines





# Model of I/O Operation

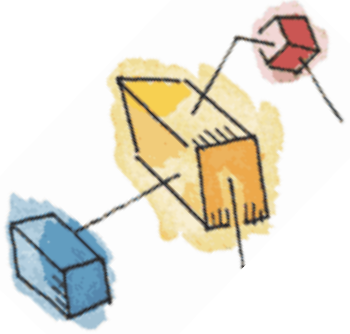


(b) Communications port

(a) Local peripheral device

(c) File system

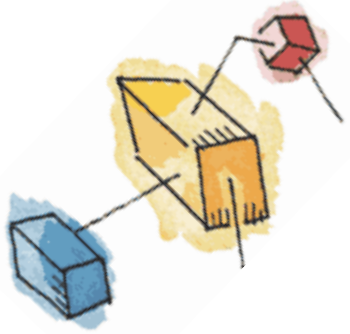




# I/O Buffering

- Reasons for buffering
  - Processes must wait for I/O to complete before proceeding
  - Certain pages must remain in main memory during I/O

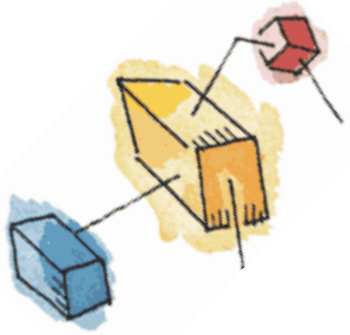




# I/O Buffering

- **Block-oriented**
  - Information is stored in fixed sized blocks
  - Transfers are made a block at a time
  - Used for disks and USB keys
- **Stream-oriented**
  - Transfer information as a stream of bytes
  - Used for terminals, printers, communication ports, mouse and other pointing devices, and most other devices that are not secondary storage

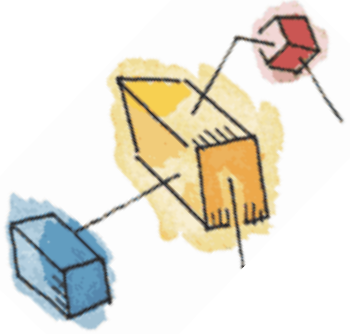




# Single Buffer

- Operating system assigns a buffer in main memory for an I/O request
- Block-oriented
  - Input transfers made to buffer
  - Block moved to user space when needed
  - Another block is moved into the buffer
    - Read ahead

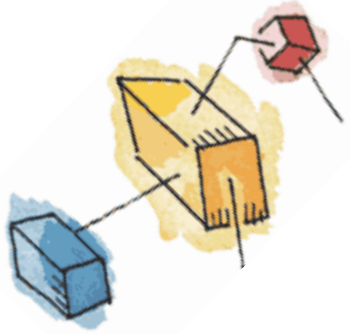




# Single Buffer

- Block-oriented
  - User process can process one block of data while next block is read in
  - Swapping can occur since input is taking place in system memory, not user memory
  - Operating system keeps track of assignment of system buffers to user processes



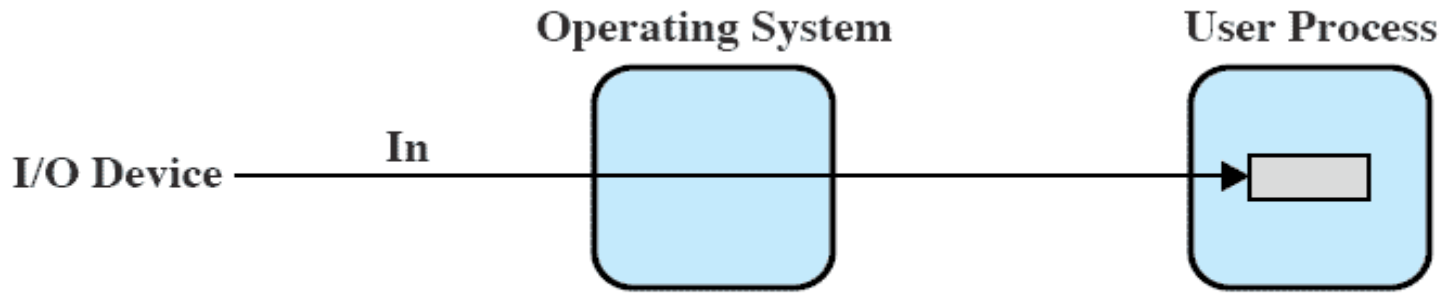
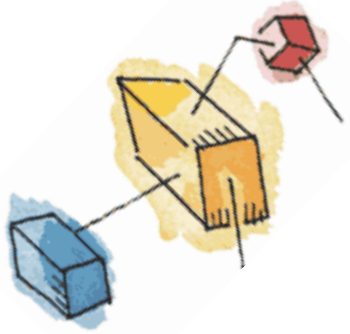


# Single Buffer

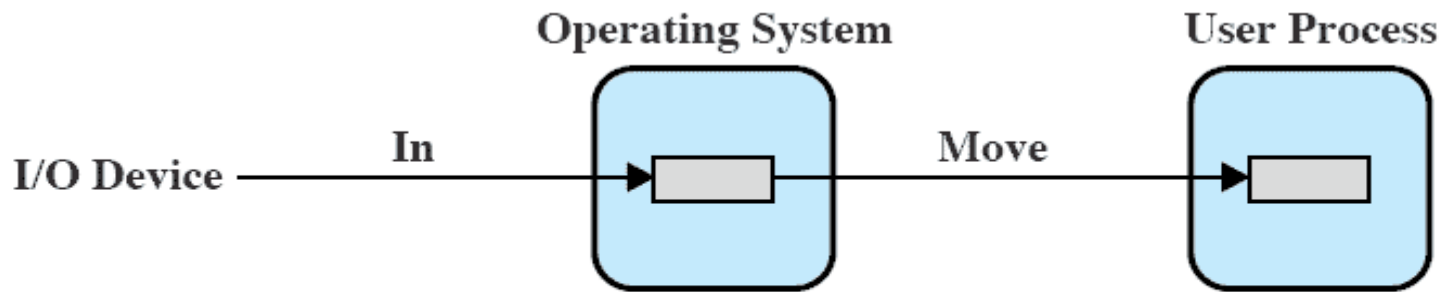
- Stream-oriented
  - Used a line at a time
  - User input from a terminal is one line at a time with carriage return signaling the end of the line
  - Output to the terminal is one line at a time



# Single Buffer

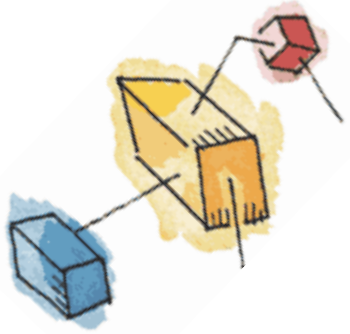


(a) No buffering



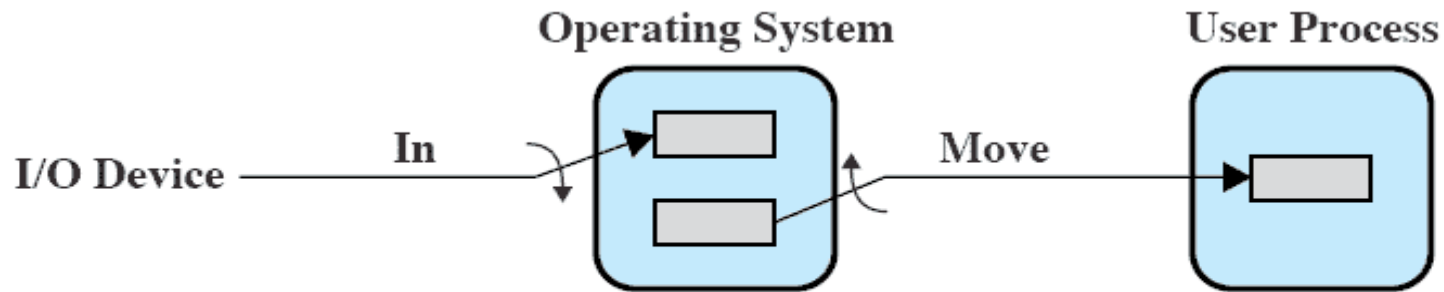
(b) Single buffering





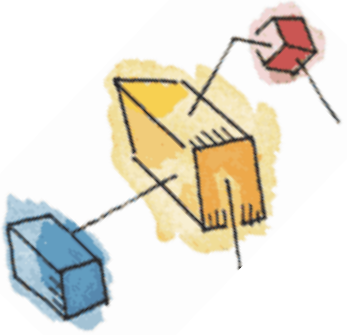
# Double Buffer

- 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



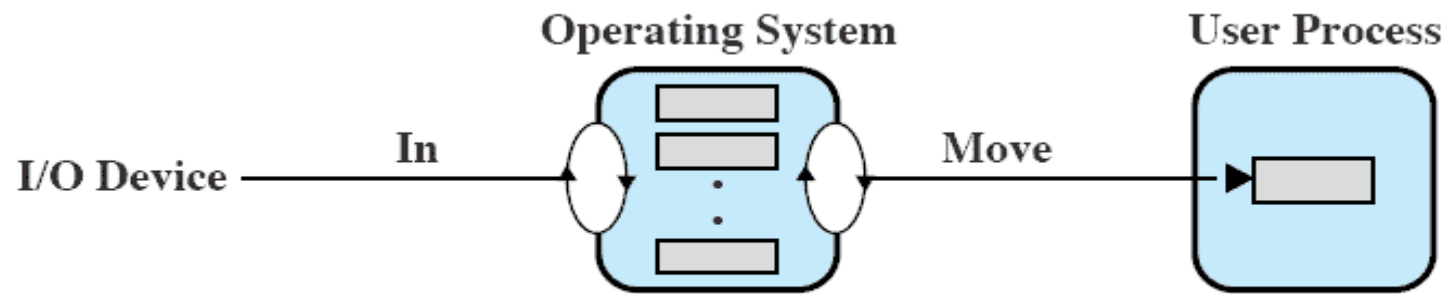
(c) Double buffering





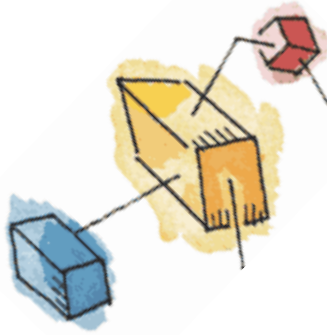
# Circular Buffer

- 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





# Disk Performance Parameters

- To read or write, the disk head must be positioned at the desired track and at the beginning of the desired sector
- Seek time
  - Time it takes to position the head at the desired track
- Rotational delay or rotational latency
  - Time its takes for the beginning of the sector to reach the head





# Timing of Disk I/O Transfer

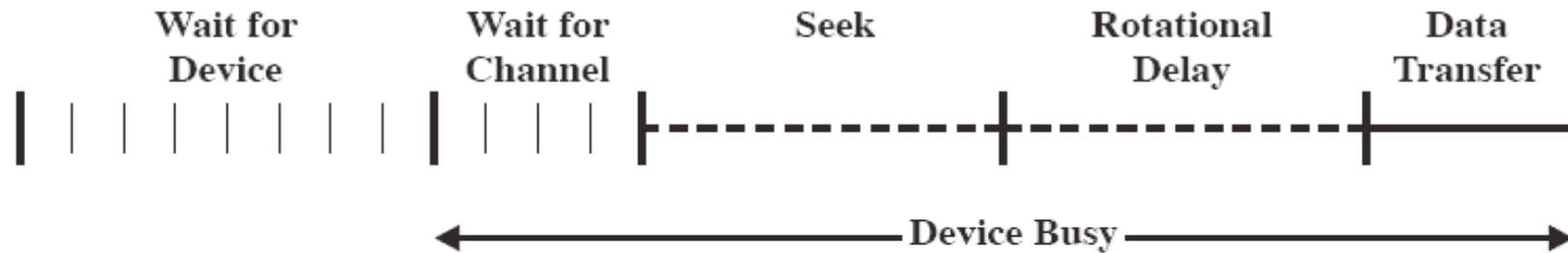


Figure 11.6 Timing of a Disk I/O Transfer

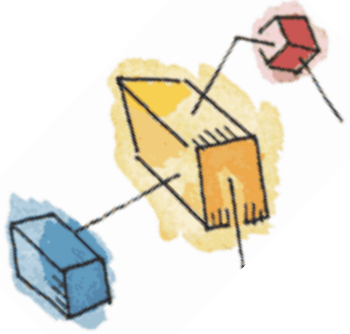




# Disk Performance Parameters

- Access time
  - Sum of seek time and rotational delay
  - The time it takes to get in position to read or write
- Data transfer occurs as the sector moves under the head

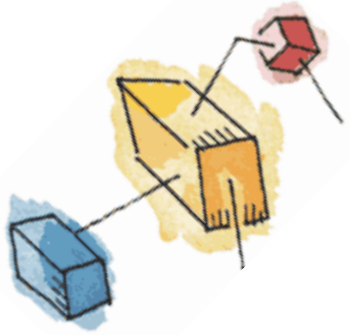




# Disk Scheduling Policies

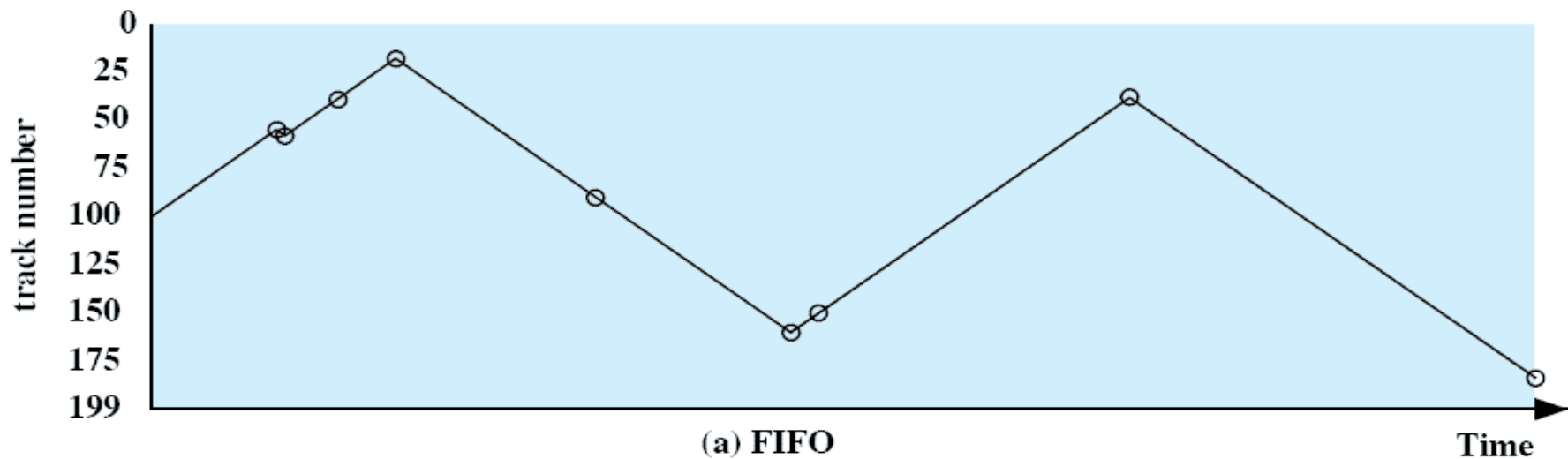
- Seek time is the reason for differences in performance
- For a single disk there will be a number of I/O requests
- If requests are selected randomly, get poor performance

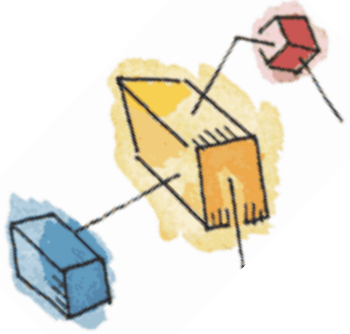




# Disk Scheduling Policies

- First-in, first-out (FIFO)
  - Process request sequentially
  - Fair to all processes
  - Approaches random scheduling in performance if there are many processes

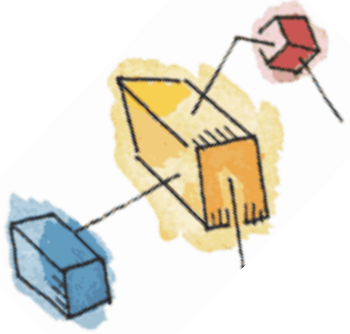




# Disk Scheduling Policies

- Priority
  - Goal is not to optimize disk use but to meet other objectives
  - Short batch jobs may have higher priority
  - Provide good interactive response time

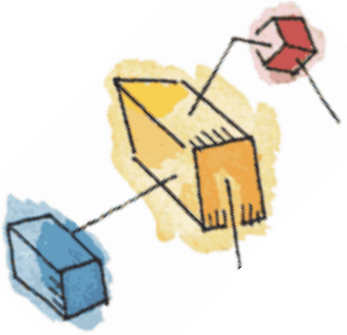




# Disk Scheduling Policies

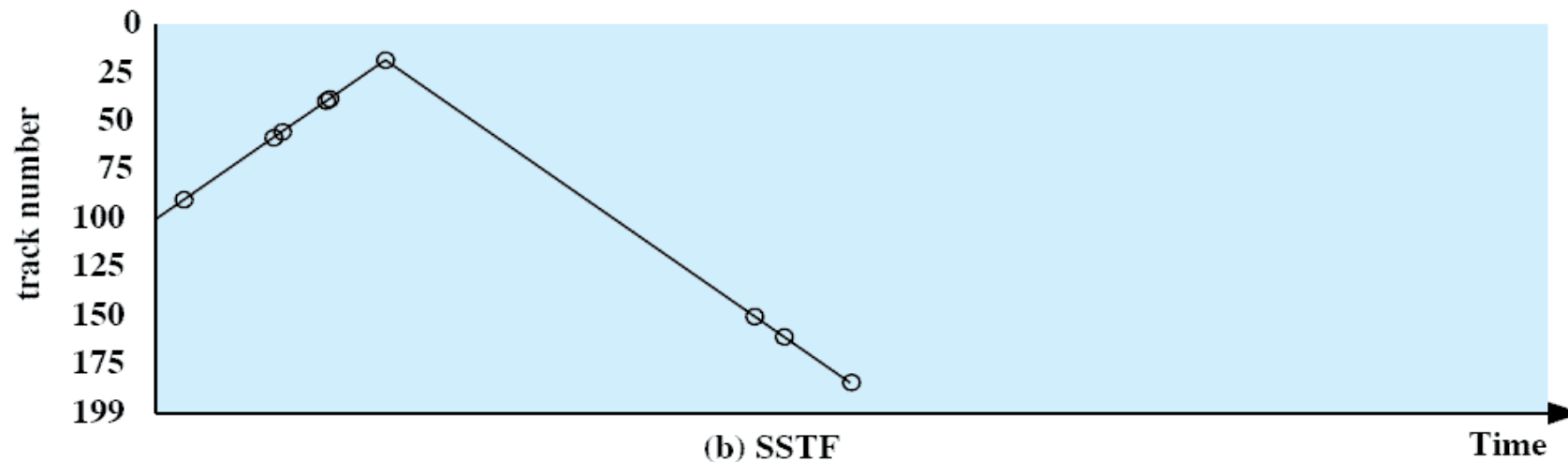
- Last-in, first-out
  - Good for transaction processing systems
    - The device is given to the most recent user so there should be little arm movement
  - Possibility of starvation since a job may never regain the head of the line

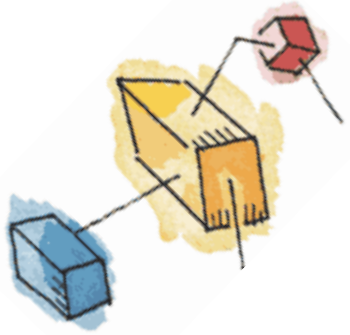




# Disk Scheduling Policies

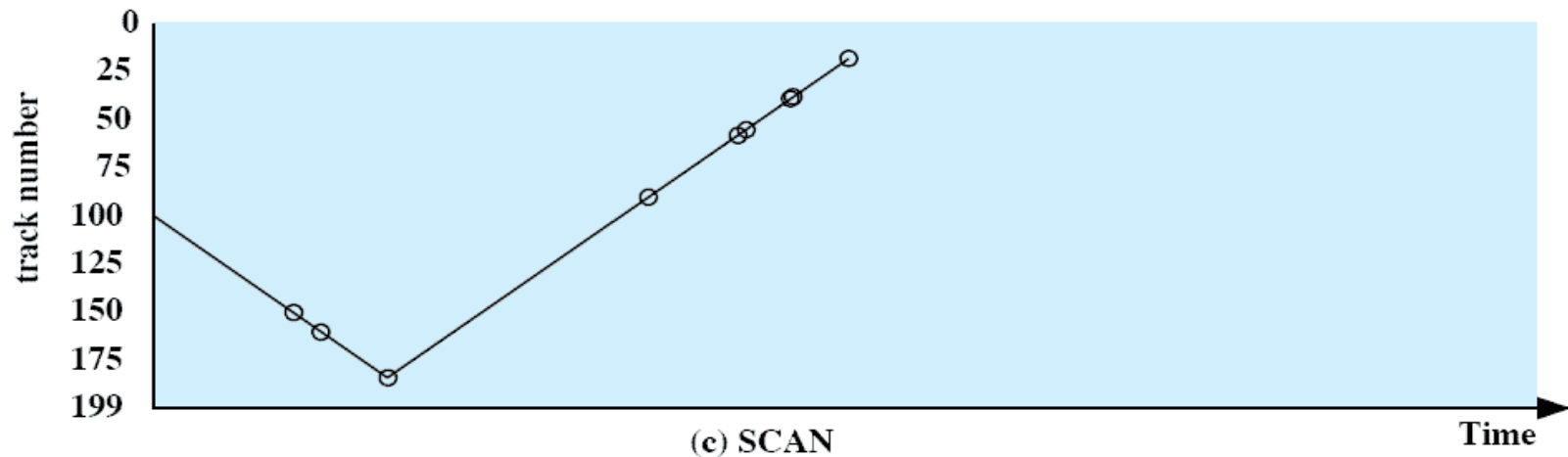
- Shortest Service Time First
  - Select the disk I/O request that requires the least movement of the disk arm from its current position
  - Always choose the minimum seek time

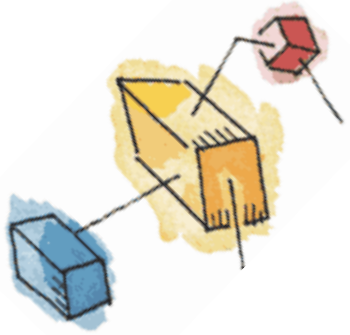




# Disk Scheduling Policies

- SCAN
  - Arm moves in one direction only, satisfying all outstanding requests until it reaches the last track in that direction
  - Direction is reversed

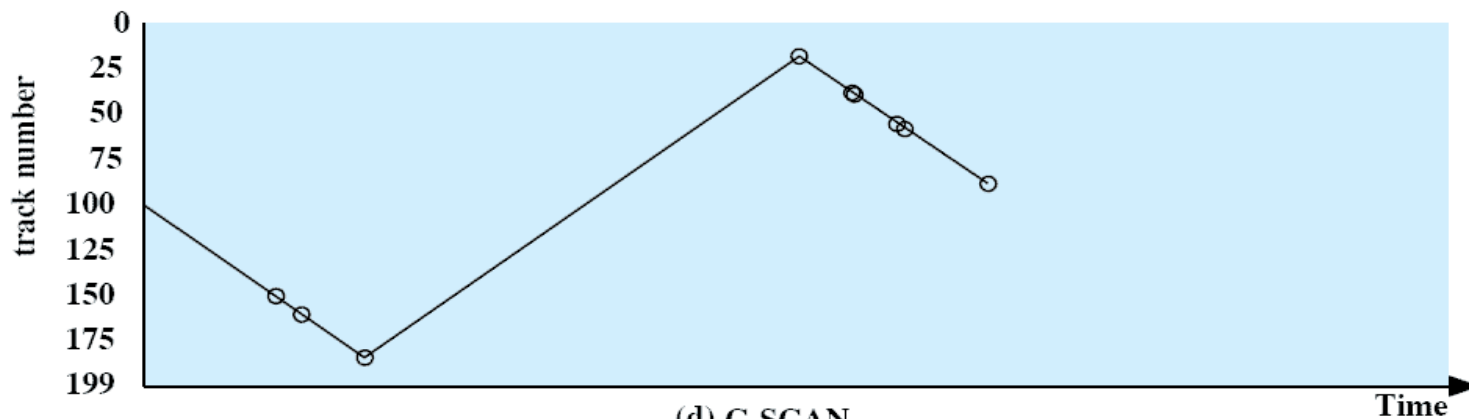




# Disk Scheduling Policies

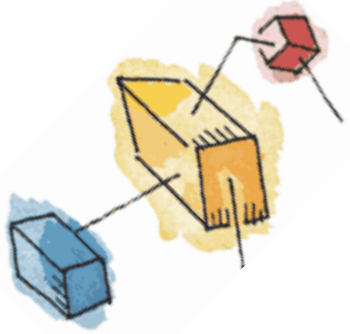
- C-SCAN

- 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



(d) C-SCAN

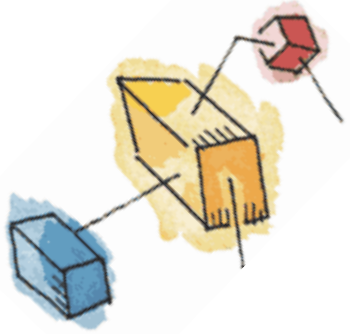




# Disk Scheduling Policies

- N-step-SCAN
  - Segments the disk request queue into subqueues of length N
  - Subqueues are processed one at a time, using SCAN
  - New requests added to other queue when queue is processed





# Disk Scheduling Policies

- FSCAN
  - Two subqueues
  - One queue is empty for new requests

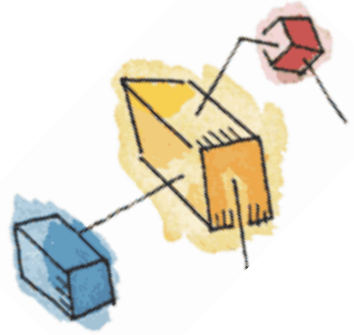




# Disk Scheduling Algorithms

(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
<b>Average seek length</b>	55.3	<b>Average seek length</b>	27.5	<b>Average seek length</b>	27.8	<b>Average seek length</b>	35.8

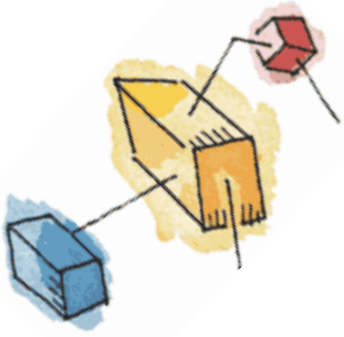




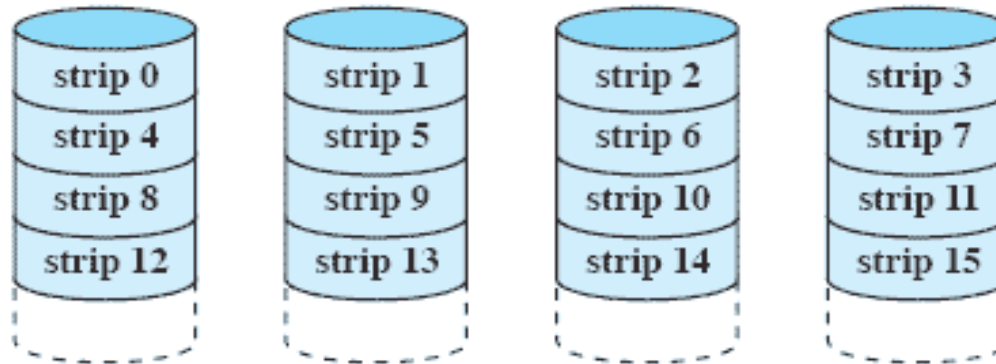
# RAID

- Redundant Array of Independent Disks
- Set of physical disk drives viewed by the operating system as a single logical drive
- Data are distributed across the physical drives of an array
- Redundant disk capacity is used to store parity information



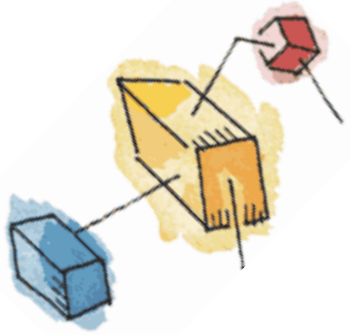


# RAID 0 (nonredundant)

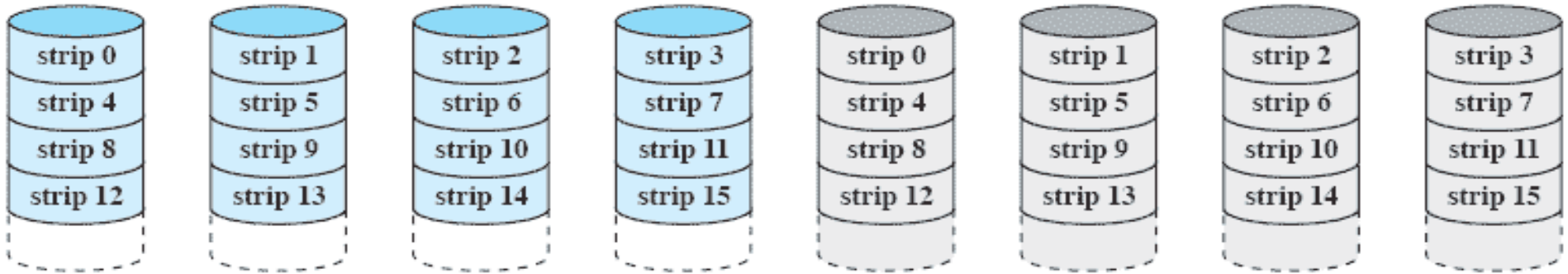


(a) RAID 0 (non-redundant)





# RAID 1 (mirrored)

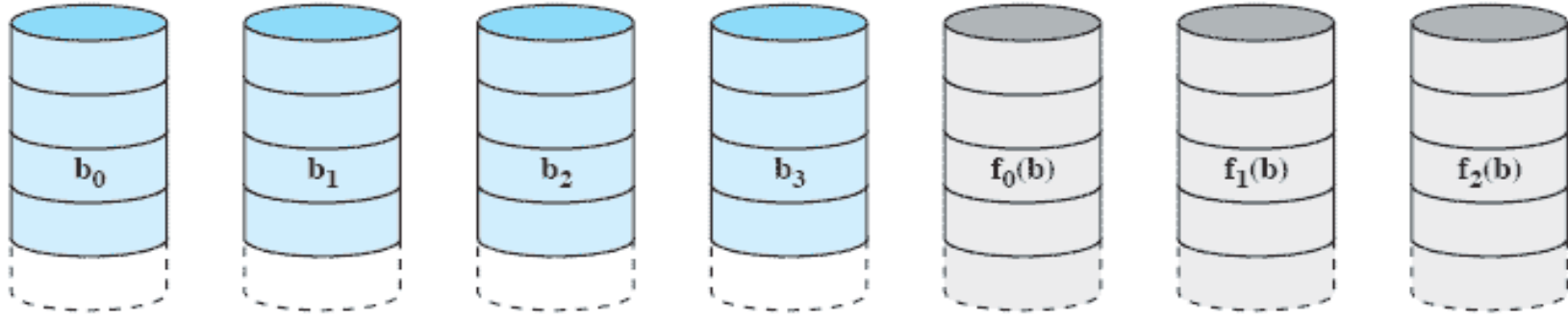


(b) RAID 1 (mirrored)



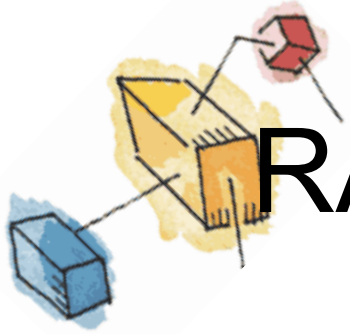


# RAID 2 (redundancy through Hamming code)

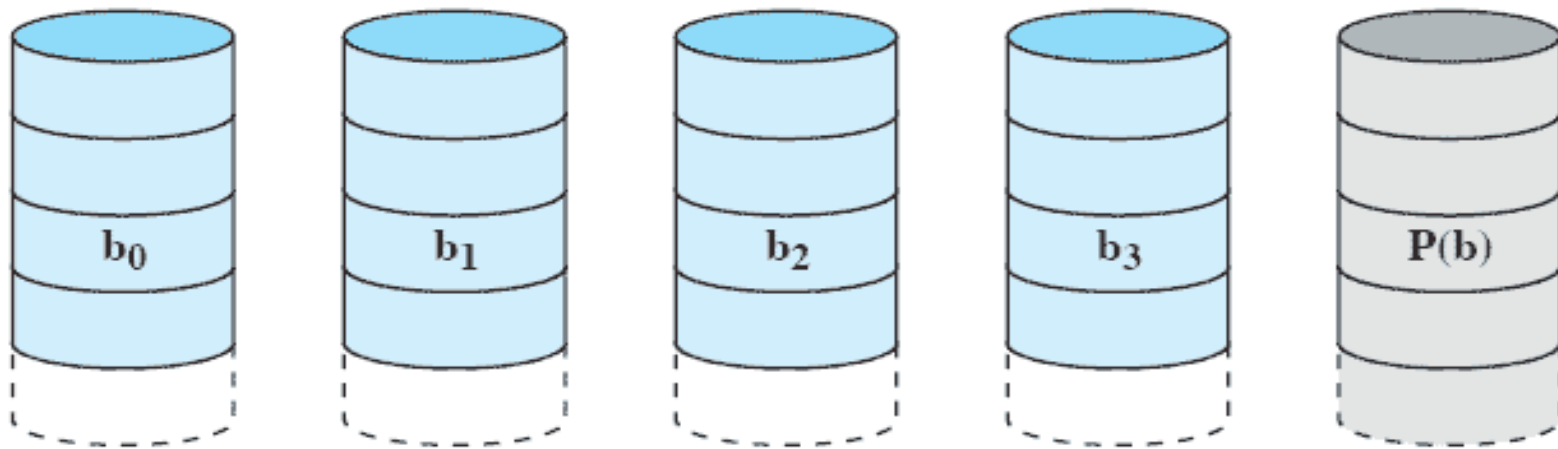


(c) RAID 2 (redundancy through Hamming code)



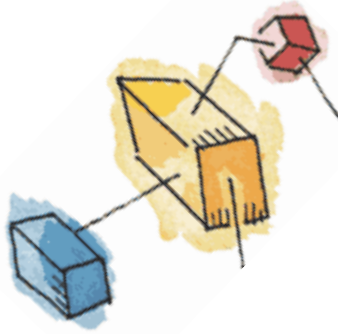


# RAID 3 (bit-interleaved parity)

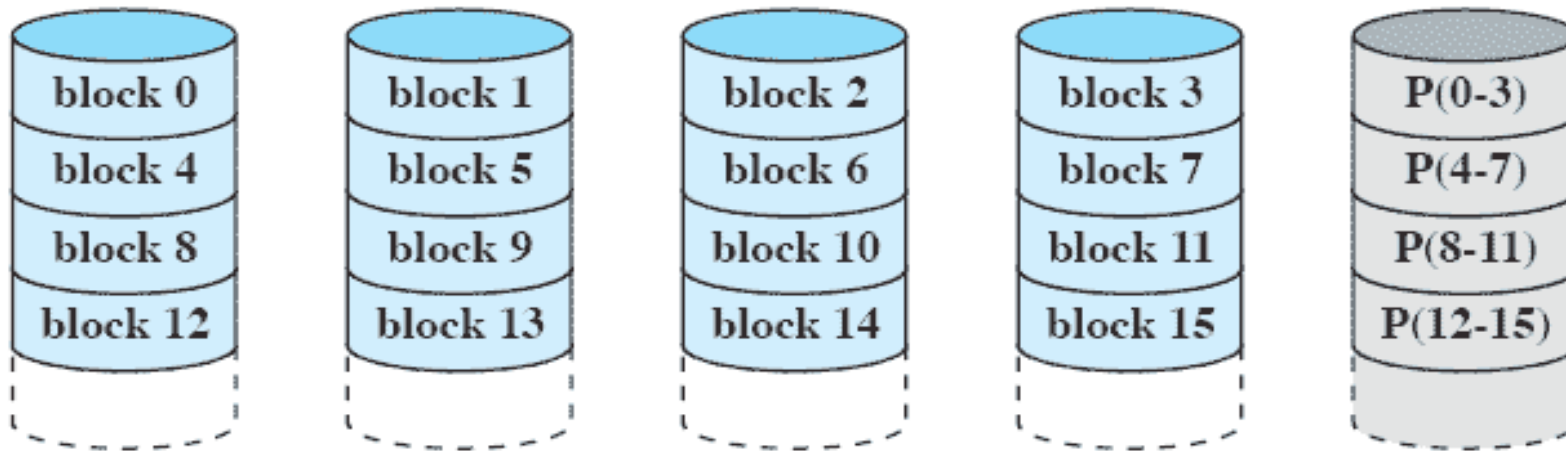


(d) RAID 3 (bit-interleaved parity)



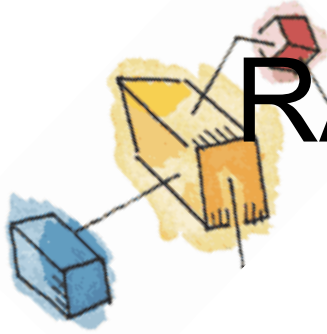


# RAID 4 (block-level parity)

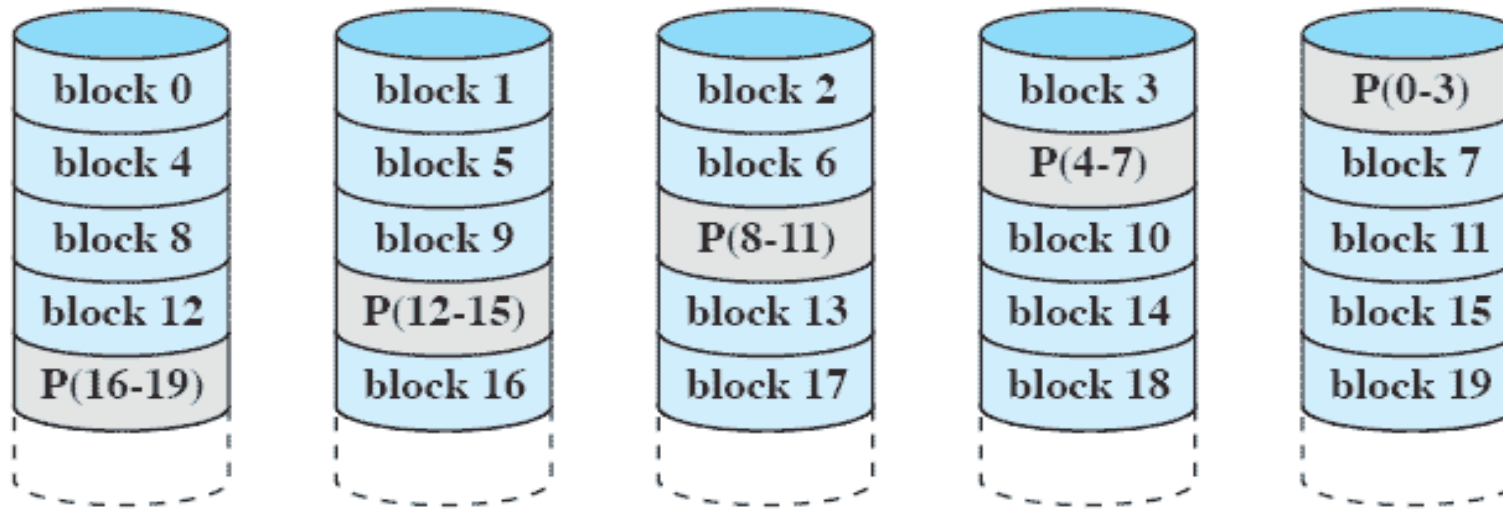


(e) RAID 4 (block-level parity)



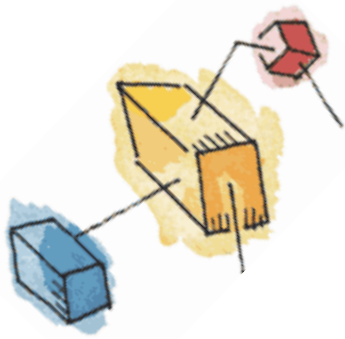


# RAID 5 (block-level distributed parity)

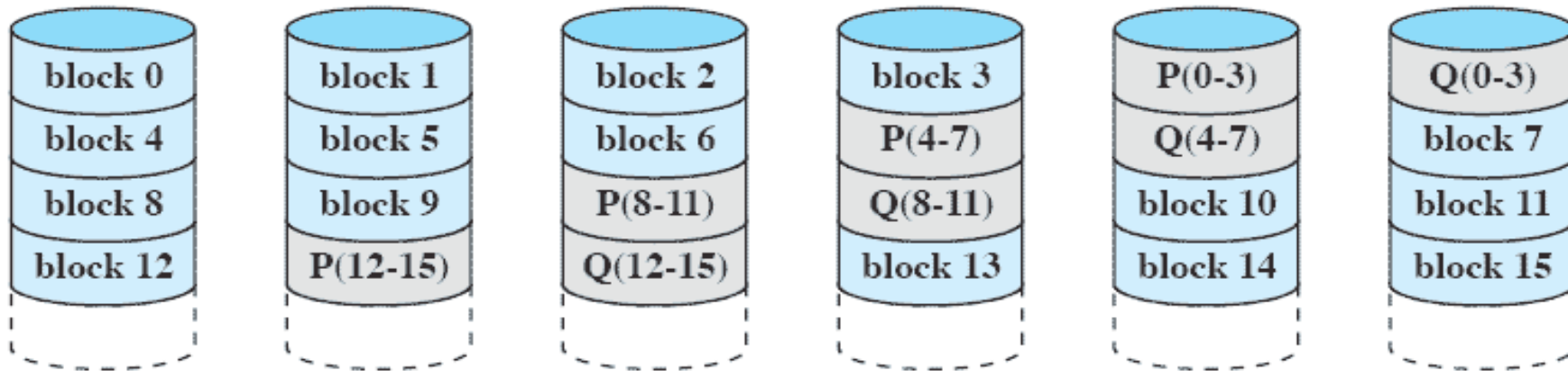


(f) RAID 5 (block-level distributed parity)



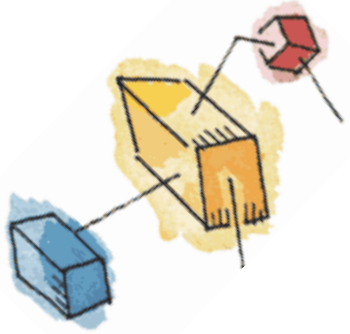


# RAID 6 (dual redundancy)



(g) RAID 6 (dual redundancy)

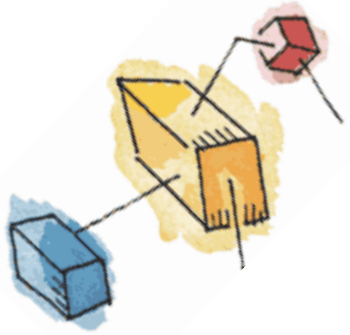




# Disk Cache

- Buffer in main memory for disk sectors
- Contains a copy of some of the sectors on the disk

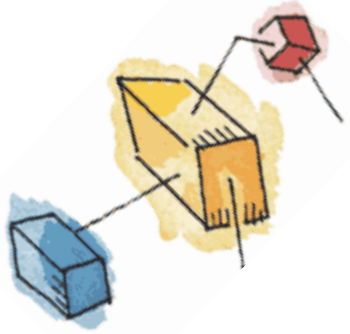




# Least Recently Used

- The block that has been in the cache the longest with no reference to it is replaced
- The cache consists of a stack of blocks
- 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

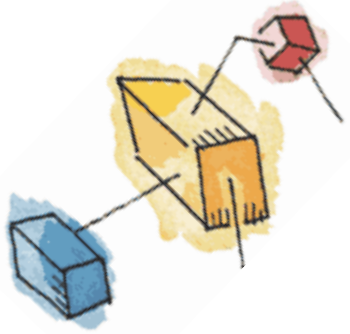




# Least Recently Used

- The block on the bottom of the stack is removed when a new block is brought in
- Blocks don't actually move around in main memory
- A stack of pointers is used

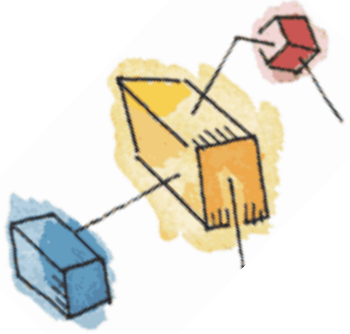




# Least Frequently Used

- The block that has experienced the fewest references is replaced
- A counter is associated with each block
- Counter is incremented each time block accessed

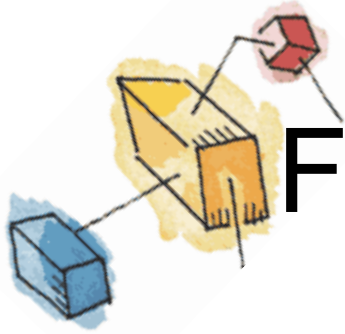




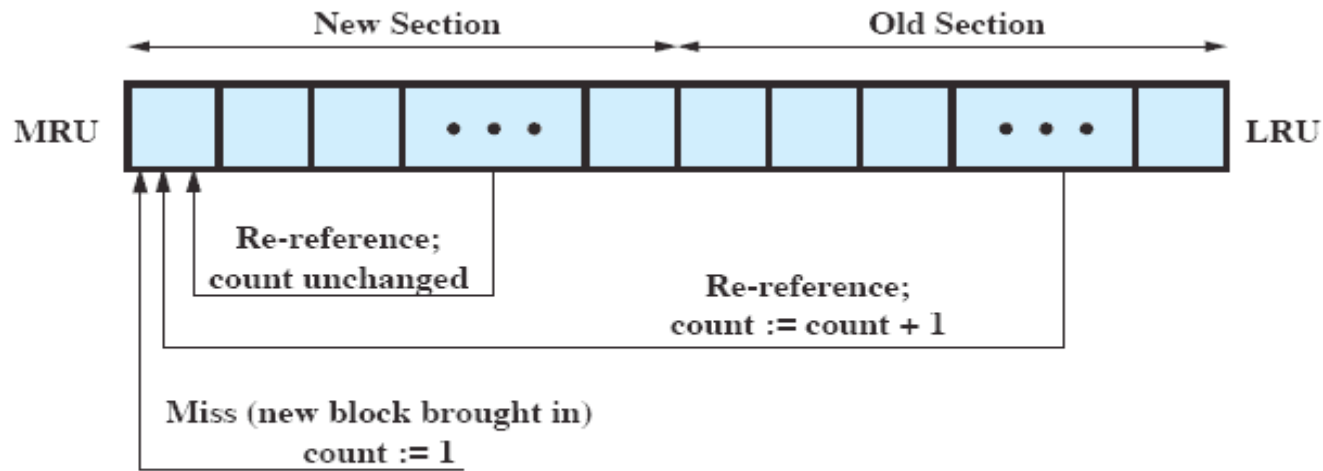
# Least Frequently Used

- Block with smallest count is selected for replacement
- Some blocks may be referenced many times in a short period of time and the reference count is misleading

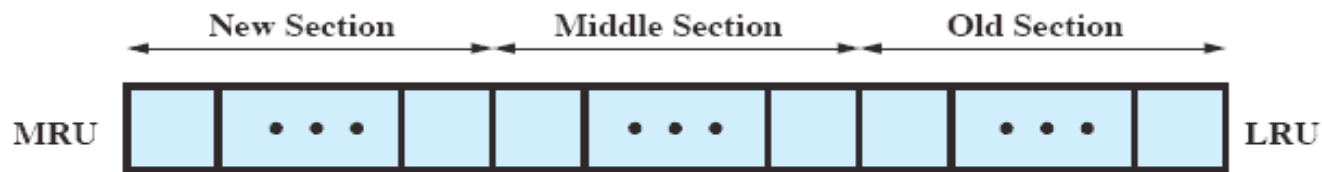




# Frequency-Based Replacement

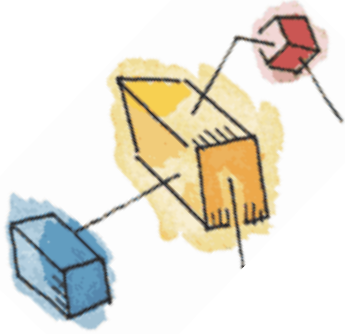


(a) FIFO



(b) Use of three sections





# Disk Cache Performance

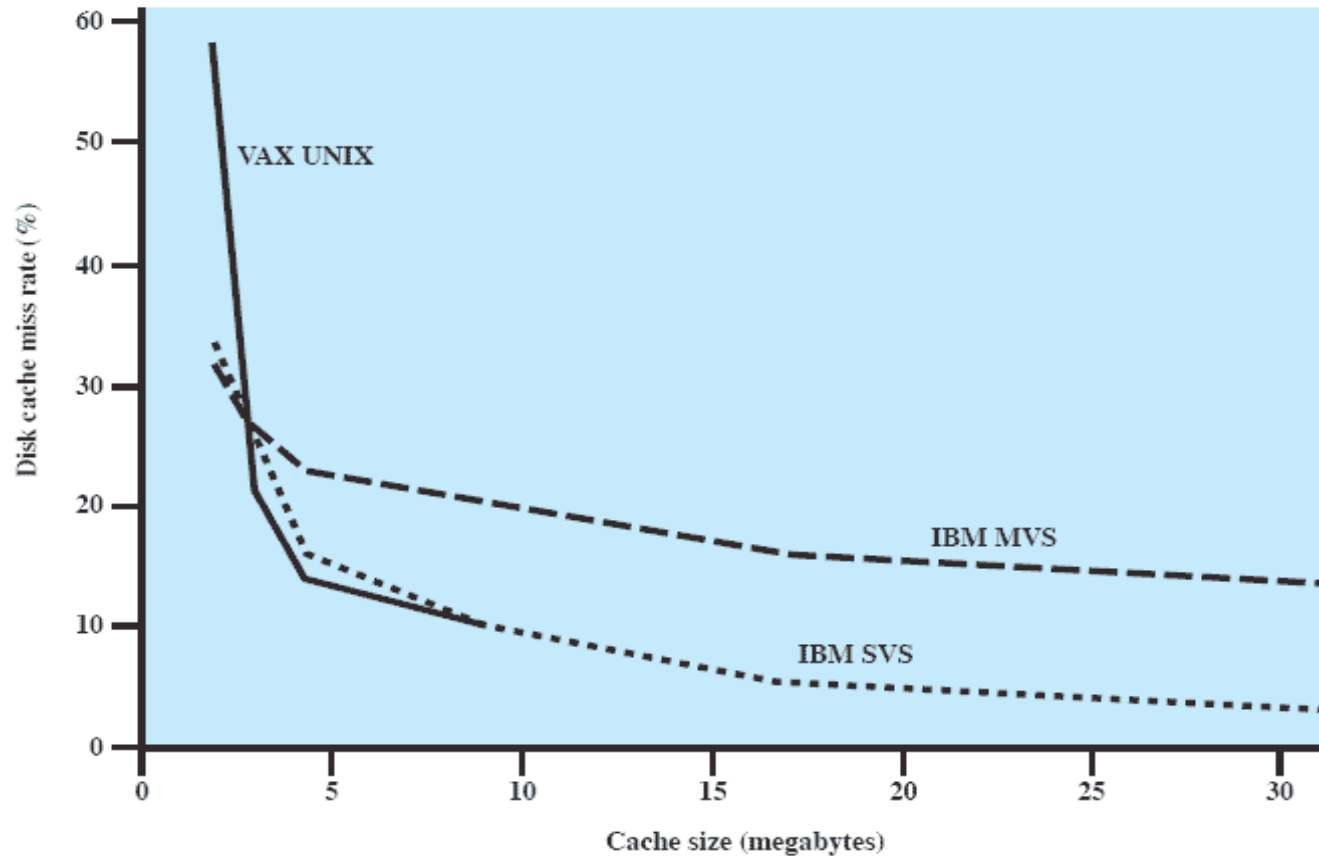
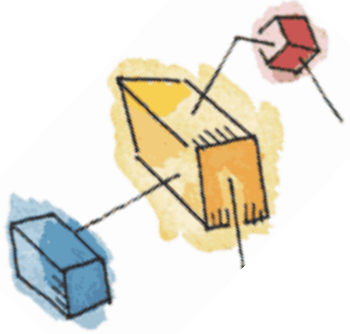


Figure 11.10 Some Disk Cache Performance Results Using LRU





# Disk Cache Performance

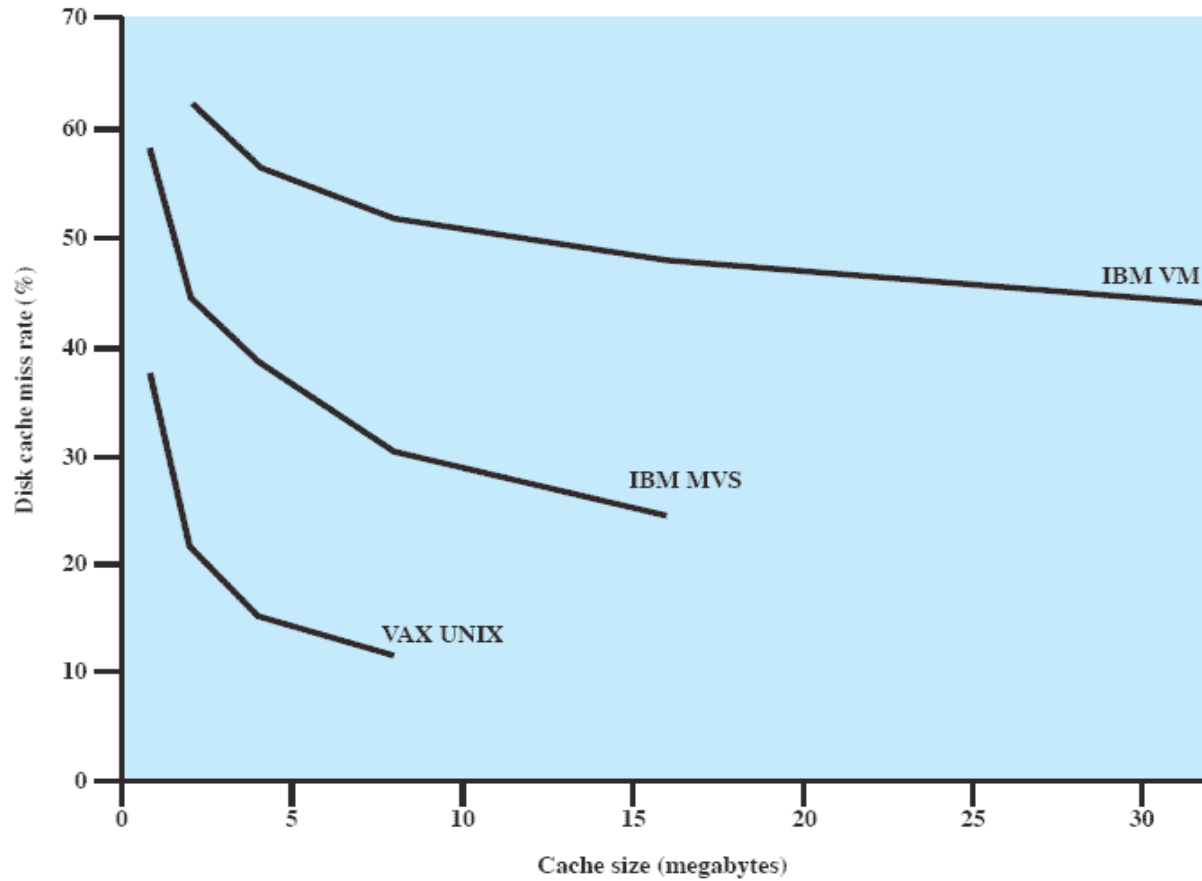


Figure 11.11 Disk Cache Performance Using Frequency-Based Replacement [ROBI90]

