# I/O Management and Disk Scheduling

Chapter 11



## Categories of I/O Devices

- Human readable
  - Used to communicate with the user
  - Printers
  - Video display terminals
    - Display
    - Keyboard
    - Mouse

## Categories of I/O Devices

- Machine readable
  - Used to communicate with electronic equipment
  - Disk and tape drives
  - Sensors
  - Controllers
  - Actuators

3

## Categories of I/O Devices

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

4

# Differences in I/O Devices

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

5

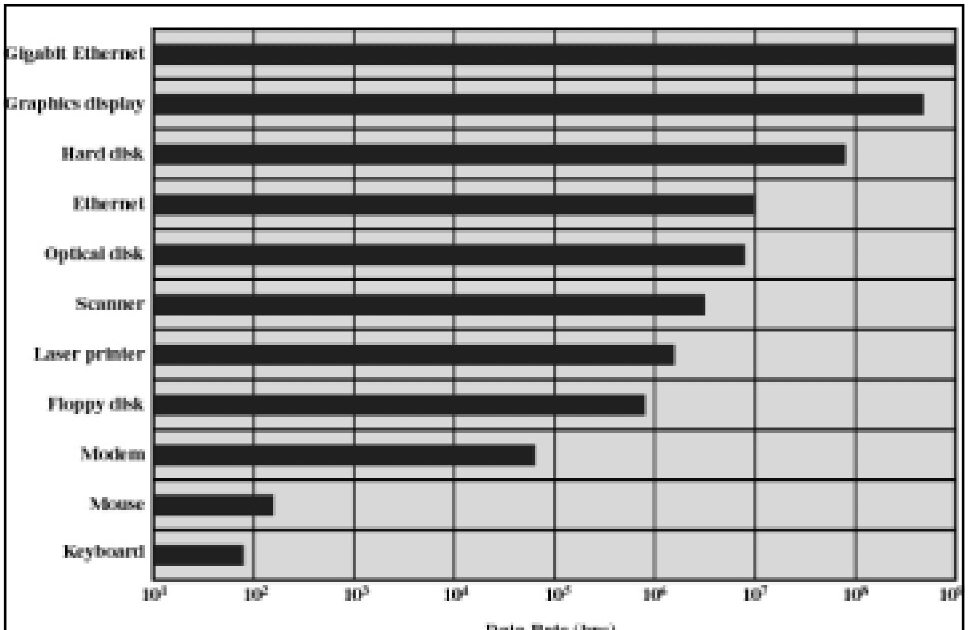



Figure 11.1 Typical I/O Device Data Rates

## 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

7

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

8

## Differences in I/O Devices

- 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
  - I/O module sends an interrupt when done

9

## Techniques for 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

10

## 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

13

## 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

14

# 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 so that processes and upper levels see devices in general terms such as read, write, open, close, lock, unlock



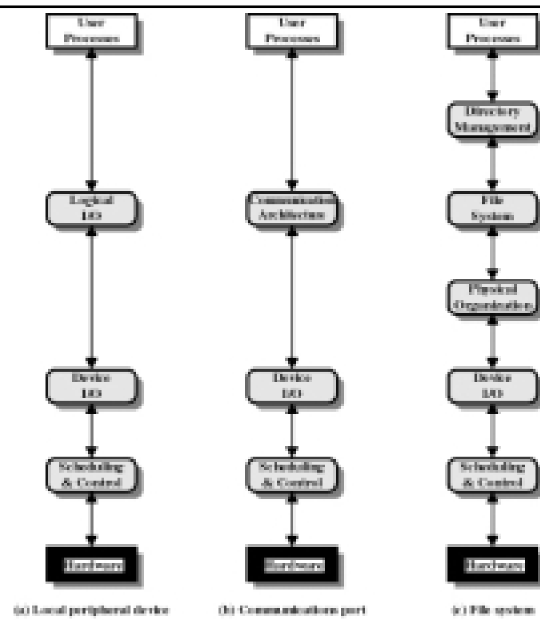



Figure 11.5 A Model of I/O Organization

## 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

17

## I/O Buffering

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

18

## 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 it takes for the beginning of the sector to reach the head

19

## Timing of a Disk I/O Transfer

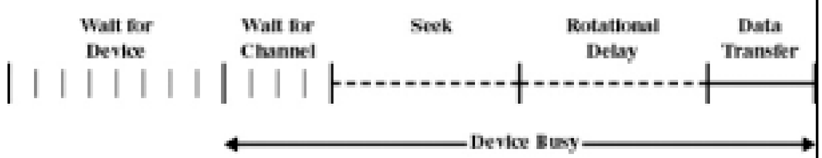



Figure 11.7 Timing of a Disk I/O Transfer

20

## 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

21

## 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, we will get the worst possible performance

22

## 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

23

## 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

24

# 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

25

# 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

26

## 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

27

## 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

28

## 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
- FSCAN
  - Two queues
  - One queue is empty for new request

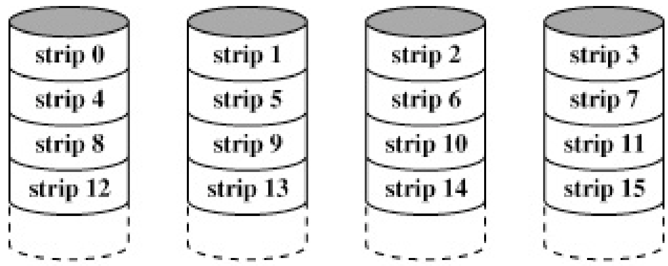
29

## Disk Scheduling Algorithms

**Table 11.3 Disk Scheduling Algorithms [WIED87]**

| Name | Description | Remarks |
|---|---|---|
| | **Selection according to requestor** | |
| RSS | 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 |

# RAID 0 (non-redundant)




(a) RAID 0 (non-redundant)

**Figure 11.9 RAID Levels** (page 1 of 2)

31

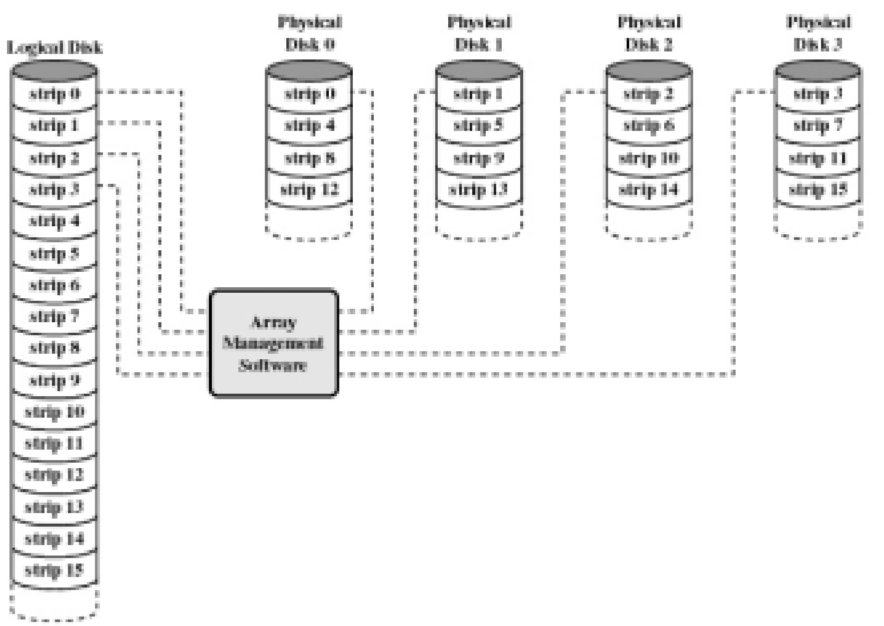
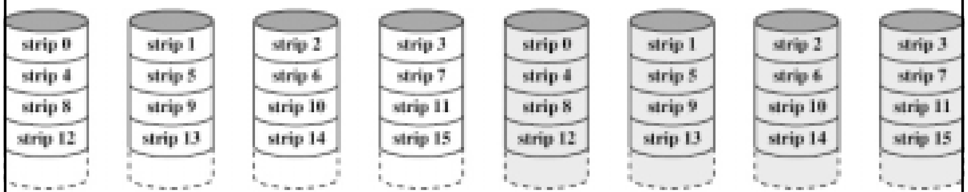



Figure 11.10 Data Mapping for a RAID Level 0 Array [MASS97]

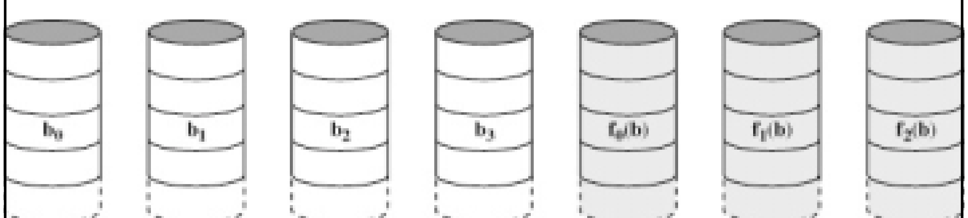
# RAID 1 (mirrored)




(b) RAID 1 (mirrored)

**Figure 11.9 RAID Levels** (page 1 of 2)

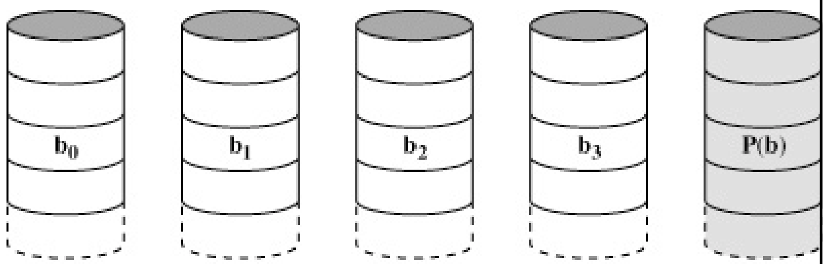
# RAID 2 (redundancy through Hamming code)




(c) RAID 2 (redundancy through Hamming code)

**Figure 11.9 RAID Levels** (page 1 of 2)

## RAID 3 (bit-interleaved parity)

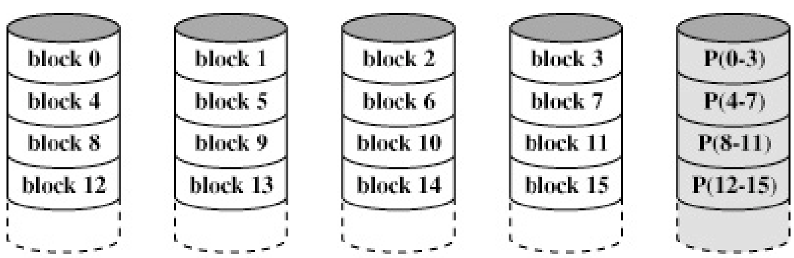



(d) RAID 3 (bit-interleaved parity)

**Figure 11.9 RAID Levels** (page 2 of 2)

35

## RAID 4 (block-level parity)

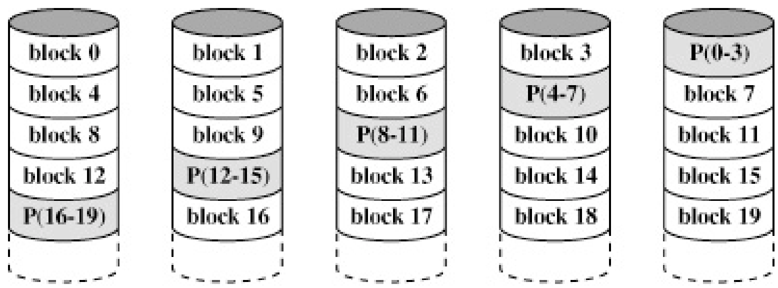



(e) RAID 4 (block-level parity)

**Figure 11.9 RAID Levels** (page 2 of 2)

36

## RAID 5 (block-level distributed parity)

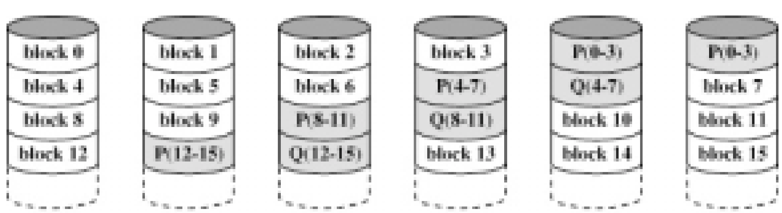



(f) RAID 5 (block-level distributed parity)

**Figure 11.9 RAID Levels** (page 2 of 2)

37

## RAID 6 (dual redundancy)




(g) RAID 6 (dual redundancy)

**Figure 11.9 RAID Levels** (page 2 of 2)

38

## Disk Cache

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

39

## 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

40

## 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

41

## 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
- Block with smallest count is selected for replacement
- Some blocks may be referenced many times in a short period of time and then not needed any more

42

# UNIX SVR4 I/O

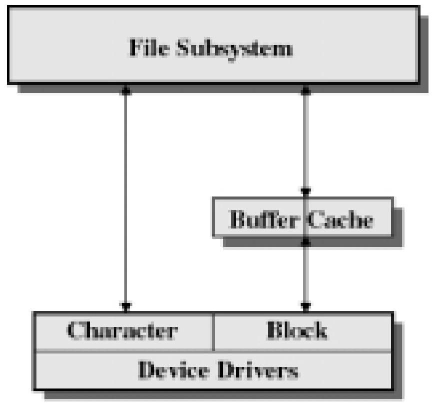



Figure 11.14 UNIX I/O Structure



# Windows 2000 I/O

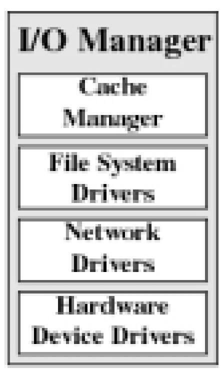



Figure 11.16 Windows 2000 I/O Manager