Chapter 11 - Lecture

Stallings - 9ed
Figure 11.1  Typical I/O Device Data Rates
Differences in I/O Devices

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

• 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
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
  – I/O module sends an interrupt when done
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
Figure 11.2  Typical DMA Block Diagram
Figure 11.3 Alternative DMA Configurations
## Relationship Among Techniques

Table 11.1  I/O Techniques

<table>
<thead>
<tr>
<th></th>
<th>No Interrupts</th>
<th>Use of Interrupts</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O-to-memory transfer</td>
<td>Programmed I/O</td>
<td>Interrupt-driven I/O</td>
</tr>
<tr>
<td>through processor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Direct I/O-to-memory</td>
<td></td>
<td>Direct memory access (DMA)</td>
</tr>
<tr>
<td>transfer</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 so that processes and upper levels see devices in general terms such as read, write, open, close, lock, unlock
Figure 11.4  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
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 (character)-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”
  - 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
  – e.g. terminal
  • Used a line at 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
  – e.g. network I/O
  • NIC (network interface card)
  • protocol stack
I/O Buffering

(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
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
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
Figure 11.6 Timing of a Disk I/O Transfer

Seek Time - 2-30ms

Rotational Latency -
- 5400 rpm (90 rps) - 0-11ms (5.6ms avg)
- 10000 rpm (167 rps) - 0-6ms (3ms avg)
- 15000 rpm (250 rps) - 0-4ms (2ms avg)

• 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
  • Transfer rate depends on rotational speed, bit density
  • Transfer rate is 1-100mB/s
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 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

• 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
## Disk Scheduling Algorithms

### Table 11.2 Comparison of Disk Scheduling Algorithms

<table>
<thead>
<tr>
<th>(a) FIFO (starting at track 100)</th>
<th>(b) SSTF (starting at track 100)</th>
<th>(c) SCAN (starting at track 100, in the direction of increasing track number)</th>
<th>(d) C-SCAN (starting at track 100, in the direction of increasing track number)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Next track accessed</strong></td>
<td><strong>Number of tracks traversed</strong></td>
<td><strong>Next track accessed</strong></td>
<td><strong>Number of tracks traversed</strong></td>
</tr>
<tr>
<td>55</td>
<td>45</td>
<td>90</td>
<td>10</td>
</tr>
<tr>
<td>58</td>
<td>3</td>
<td>58</td>
<td>32</td>
</tr>
<tr>
<td>39</td>
<td>19</td>
<td>55</td>
<td>3</td>
</tr>
<tr>
<td>18</td>
<td>21</td>
<td>39</td>
<td>16</td>
</tr>
<tr>
<td>90</td>
<td>72</td>
<td>38</td>
<td>1</td>
</tr>
<tr>
<td>160</td>
<td>70</td>
<td>18</td>
<td>20</td>
</tr>
<tr>
<td>150</td>
<td>10</td>
<td>150</td>
<td>132</td>
</tr>
<tr>
<td>38</td>
<td>112</td>
<td>160</td>
<td>10</td>
</tr>
<tr>
<td>184</td>
<td>146</td>
<td>184</td>
<td>24</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Average seek length</th>
<th>Average seek length</th>
<th>Average seek length</th>
<th>Average seek length</th>
</tr>
</thead>
<tbody>
<tr>
<td>55.3</td>
<td>27.5</td>
<td>27.8</td>
<td>35.8</td>
</tr>
</tbody>
</table>
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
• Term was invented by Patterson and Katz (Berkeley, 1994)
(a) RAID 0 (non-redundant)

(b) RAID 1 (mirrored)

(c) RAID 2 (redundancy through Hamming code)

Figure 11.8  RAID Levels (page 1 of 2)
(d) RAID 3 (bit-interleaved parity)

(e) RAID 4 (block-level parity)

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

(g) RAID 6 (dual redundancy)

Figure 11.8  RAID Levels
RAID 10

RAID 10 is sometimes also called RAID 1+0

UNIX SCR4 I/O

- Each individual device is associated with a special file
- Two types of I/O
  - Buffered
  - Unbuffered

Figure 11.12 UNIX I/O Structure
Linux I/O

• Elevator scheduler
  – Maintains a single queue for disk read and write requests
  – Keeps list of requests sorted by block number
  – Drive moves in a single direction to satisfy each request
Linux I/O

- **Deadline scheduler**
  - Uses three queues
    - Incoming requests
    - Read requests go to the tail of a FIFO queue
    - Write requests go to the tail of a FIFO queue
  - Each request has an expiration time
    - defaults for requests:
      - 0.5s for read
      - 5s for write
1. Put requests in sorted queue *and* FIFO
   • remove request from both Qs when processed
   • Schedule from sorted Q and check expiration date of FIFO entry.
   • if date has expired, schedule from FIFO until “caught up”
Linux I/O

• Anticipatory I/O scheduler
  – Delay a short period of time after satisfying a read request to see if a new nearby request can be made
Windows I/O

• Basic I/O modules
  – Cache manager
  – File system drivers
  – Network drivers
  – Hardware device drivers
Figure 11.15  Windows I/O Manager