Chapter 11
I/O Management and Disk Scheduling
Differences in I/O Devices

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

• Unit of transfer – Character and Block devices
  – Data may be transferred as a stream of bytes for a terminal or in larger blocks for a disk

• Data representation
  – Encoding and error-correction schemes

• Error conditions
  – Different types of errors
Device Controllers

• I/O devices have components:
  – mechanical component
  – electronic component

• The electronic component is the device controller
  – may be able to handle multiple devices

• Controller's tasks
  – convert serial bit stream to block of bytes
  – perform error correction as necessary
  – communicate with CPU
Performing I/O

• **Programmed I/O**
Process(or) is busy-waiting for the operation to complete

• **Interrupt-driven I/O**
I/O command is issued
Processor continues executing instructions

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

copy_from_user(buffer, p, count);
for (i = 0; i < count; i++) {
    while (*printer_status_reg != READY);
    *printer_data_register = p[i];
}
return_to_user(); /* p is the kernel buffer */
/* loop on every character */
/* loop until ready */
/* output one character */

Writing a string to the printer using programmed I/O --- busy waiting
Interrupts Revisited

1. Device is finished

2. Controller issues interrupt

3. CPU acks interrupt

CPU

Interrupt controller

Disk

Clock

Keyboard

Printer
Interrupt-Driven I/O

- Writing a string to the printer using interrupt-driven I/O
  - Code executed when print system call is made
  - Interrupt service procedure

```c
copy_from_user(buffer, p, count);
enable_interrugets();
while (*printer_status_reg != READY) ;
*printer_data_register = p[0];
scheduler();

if (count == 0) {
    unblock_user();
} else {
    *printer_data_register = p[i];
    count = count - 1;
    i = i + 1;
}
acknowledge_interrupt();
return_from_interrupt();
```
Figure 11.2  Typical DMA Block Diagram
Direct Memory Access (DMA)

1. CPU programs the DMA controller
2. DMA requests transfer to memory
3. Data transferred
4. Ack

Interrupt when done

Drive

Buffer

Main memory

CPU

DMA controller

Address

Count

Control

Disk controller

Bus
DMA Configurations (1)

(b) Single-bus, Integrated DMA-I/O
DMA Configurations (2)

![Diagram showing DMA configurations involving Processor, DMA, Memory, and I/O devices connected through System and I/O buses.](image-url)
I/O Using DMA

- Printing a string using DMA
  - code executed when the print system call is made
  - interrupt service procedure
<table>
<thead>
<tr>
<th>I/O-to-memory transfer through processor</th>
<th>No Interrupts</th>
<th>Use of Interrupts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programmed I/O</td>
<td></td>
<td>Interrupt-driven I/O</td>
</tr>
<tr>
<td>Direct I/O-to-memory transfer</td>
<td></td>
<td>Direct memory access (DMA)</td>
</tr>
</tbody>
</table>
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 is used to bring in additional Ready processes which is an I/O operation
Operating System Design Issues (2)

• 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
I/O Software Layers

User processes
- Make I/O call; format I/O; spooling

Device-independent software
- Naming, protection, blocking, buffering, allocation

Device drivers
- Set up device registers; check status

Interrupt handlers
- Wake up driver when I/O completed

Hardware
- Perform I/O operation
Device Drivers

User process

User program

Rest of the operating system

Printer driver

Camcorder driver

CD-ROM driver

Printer controller

Camcorder controller

CD-ROM controller

Hardware

Devices
Tasks of Device Drivers

- Accept abstract requests
- Check input parameters
- Translate from abstract to concrete
- Check if device is in use
- Issue commands to controller
- (Block)
- Check errors
- Return (error) to caller
Device-Independent I/O Software

(a) Without a standard driver interface
(b) With a standard driver interface
With or without Buffering

(a) No buffering

(b) Single buffering
I/O Buffering (1)

- Reasons for buffering
  - Processes can continue while I/O is taking place (provided data is not needed immediately)
  - Process can be swapped to hard disk (otherwise certain pages must remain in main memory during I/O, e.g. target of read) --- 1 process deadlock!?
I/O Buffering (2)

• **Block-oriented**
  – Information is stored in fixed sized blocks
  – Transfers are made as one 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
  – Input transfers made to buffer
  – Data moved to user space when needed
  – Extra data is moved into the buffer
  – User process can process data while additional data 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
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
Bounded-buffer producer-consumer model
Disk Performance Parameters (1)

• 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
Disk Performance Parameters (2)

• 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 Hardware

- Physical geometry of a disk with two zones
- A possible virtual geometry for this disk
Disk Scheduling Policies

- Seek time is the main 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 - FIFO

- 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 - SSTF

- Shortest Service/Seek 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
Shortest Seek First (SSF) disk scheduling algorithm
Disk Scheduling Policies - SCAN

- SCAN or Elevator
  - 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

• 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

![Graph showing the track number over time for C-SCAN](image)
<table>
<thead>
<tr>
<th></th>
<th>(a) FIFO</th>
<th>(b) SSTF</th>
<th>(c) SCAN</th>
<th>(d) C-SCAN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(starting at track 100)</td>
<td>(starting at track 100)</td>
<td>(starting at track 100, in the direction of increasing track number)</td>
<td>(starting at track 100, in the direction of increasing track number)</td>
</tr>
<tr>
<td>Next track accessed</td>
<td>Number of tracks traversed</td>
<td>Next track accessed</td>
<td>Number of tracks traversed</td>
<td>Next track accessed</td>
</tr>
<tr>
<td>55</td>
<td>45</td>
<td>90</td>
<td>10</td>
<td>150</td>
</tr>
<tr>
<td>58</td>
<td>3</td>
<td>58</td>
<td>32</td>
<td>160</td>
</tr>
<tr>
<td>39</td>
<td>19</td>
<td>55</td>
<td>3</td>
<td>184</td>
</tr>
<tr>
<td>18</td>
<td>21</td>
<td>39</td>
<td>16</td>
<td>90</td>
</tr>
<tr>
<td>90</td>
<td>72</td>
<td>38</td>
<td>1</td>
<td>58</td>
</tr>
<tr>
<td>160</td>
<td>70</td>
<td>18</td>
<td>20</td>
<td>55</td>
</tr>
<tr>
<td>150</td>
<td>10</td>
<td>150</td>
<td>132</td>
<td>39</td>
</tr>
<tr>
<td>38</td>
<td>112</td>
<td>160</td>
<td>10</td>
<td>38</td>
</tr>
<tr>
<td>184</td>
<td>146</td>
<td>184</td>
<td>24</td>
<td>18</td>
</tr>
<tr>
<td><strong>Average seek length</strong></td>
<td><strong>55.3</strong></td>
<td><strong>27.5</strong></td>
<td><strong>27.8</strong></td>
<td><strong>35.8</strong></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 error-detection/correction (parity, Hamming code) information
Varying degree of parallelism - read and write performances
RAID 0 (non-redundant)

(a) RAID 0 (non-redundant)
RAID 1 (mirrored)
RAID 2 (Hamming code)
RAID 3 (bit-interleaved parity)

(d) RAID 3 (bit-interleaved parity)
RAID 4 (block-level parity)
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
• Various replacement algorithms
Least Recently Used (1)

• 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 (2)

- 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
• 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