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
Figure 11.1  Typical I/O Device Data Rates
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 other instructions

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

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

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 */
Interrupts Revisited

1. Device is finished

- Disk
- Clock
- Keyboard
- Printer
Interrupt-Driven I/O

copy_from_user(buffer, p, count);
enable_interrups();
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();

(a) (b)

• Writing a string to the printer using interrupt-driven I/O
  – Code executed when print system call is made
  – Interrupt service procedure
Direct Memory Access (DMA)

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

Interrupt when done
Address
Count
Control
Drive
Buffer
Main memory
Bus
DMA Configurations (1)

(b) Single-bus, Integrated DMA-I/O
DMA Configurations (2)
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></th>
<th>No Interrupts</th>
<th>Use of Interrupts</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>I/O-to-memory transfer</strong></td>
<td>Programmed I/O</td>
<td>Interrupt-driven I/O</td>
</tr>
<tr>
<td><strong>through processor</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Direct I/O-to-memory</strong></td>
<td></td>
<td>Direct memory access (DMA)</td>
</tr>
<tr>
<td><strong>transfer</strong></td>
<td></td>
<td></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
• **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

I/O request

Layer

I/O reply

I/O functions

Make I/O call; format I/O; spooling

Naming, protection, blocking, buffering, allocation

Set up device registers; check status

Wake up driver when I/O completed

Perform I/O operation

User processes

Device-independent software

Device drivers

Interrupt handlers

Hardware
Device Drivers

User space

User program

Kernel space

Rest of the operating system

Printer driver

Camcorder driver

CD-ROM driver

Hardware

Printer controller

Camcorder controller

CD-ROM controller

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
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
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 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
SSF (2)

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
<table>
<thead>
<tr>
<th>(a) FIFO</th>
<th>(b) SSTF</th>
<th>(c) SCAN</th>
<th>(d) C-SCAN</th>
</tr>
</thead>
<tbody>
<tr>
<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>
<td></td>
</tr>
<tr>
<td><strong>Next track accessed</strong></td>
<td><strong>Next track accessed</strong></td>
<td><strong>Next track accessed</strong></td>
<td><strong>Next track accessed</strong></td>
</tr>
<tr>
<td><strong>Number of tracks traversed</strong></td>
<td><strong>Number of tracks traversed</strong></td>
<td><strong>Number of tracks traversed</strong></td>
<td><strong>Number of tracks traversed</strong></td>
</tr>
<tr>
<td>55</td>
<td>90</td>
<td>150</td>
<td>150</td>
</tr>
<tr>
<td>58</td>
<td>58</td>
<td>160</td>
<td>160</td>
</tr>
<tr>
<td>39</td>
<td>55</td>
<td>184</td>
<td>184</td>
</tr>
<tr>
<td>18</td>
<td>39</td>
<td>90</td>
<td>38</td>
</tr>
<tr>
<td>90</td>
<td>38</td>
<td>58</td>
<td>18</td>
</tr>
<tr>
<td>160</td>
<td>18</td>
<td>55</td>
<td>38</td>
</tr>
<tr>
<td>150</td>
<td>150</td>
<td>39</td>
<td>58</td>
</tr>
<tr>
<td>38</td>
<td>160</td>
<td>38</td>
<td>55</td>
</tr>
<tr>
<td>184</td>
<td>184</td>
<td>18</td>
<td>90</td>
</tr>
<tr>
<td><strong>Average seek length</strong></td>
<td><strong>Average seek length</strong></td>
<td><strong>Average seek length</strong></td>
<td><strong>Average seek length</strong></td>
</tr>
<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
RAID 0 (non-redundant)
RAID 1 (mirrored)
RAID 2 (Hamming code)

(c) RAID 2 (redundancy through Hamming code)
RAID 3 (bit-interleaved parity)
RAID 4 (block-level parity)

(e) RAID 4 (block-level parity)
RAID 5 (block-level distributed parity)
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 (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

New Section

Old Section

MRU

Re-reference; count unchanged

LRU

Re-reference; count := count + 1

Miss (new block brought in)

count := 1

(a) FIFO