I/O Management
Intro
Chapter 5

Learning Outcomes
• A high-level understanding of the properties of a variety of I/O devices.
• An understanding of methods of interacting with I/O devices.
• An appreciation of the trend towards offloading more I/O handling to devices themselves.

I/O Devices
• There exists a large variety of I/O devices:
  – Many of them with different properties
  – They seem to require different interfaces to manipulate and manage them
    • We don’t want a new interface for every device
    • Diverse, but similar interfaces leads to code duplication
• Challenge:
  – Uniform and efficient approach to I/O

Categories of I/O Devices (by usage)
• Human interface
  – Used to communicate with the user
  – Printers, Video Display, Keyboard, Mouse
• Machine interface
  – Used to communicate with electronic equipment
  – Disk and tape drives, Sensors, Controllers, Actuators
• Communication
  – Used to communicate with remote devices
  – Ethernet, Modems, Wireless

I/O Device Handling
• Data rate
  – May be differences of several orders of magnitude between the data transfer rates
  – Example: Assume 1000 cycles/byte I/O
    • Keyboard needs 10 KHz processor to keep up
    • Gigabit Ethernet needs 100 GHz processor…..
Sample Data Rates

<table>
<thead>
<tr>
<th>Device</th>
<th>Data rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mechanical</td>
<td>16 KB/sec</td>
</tr>
<tr>
<td>Mouse</td>
<td>100 KB/sec</td>
</tr>
<tr>
<td>SD card</td>
<td>7 MB/sec</td>
</tr>
<tr>
<td>Telephone channel</td>
<td>8 KB/sec</td>
</tr>
<tr>
<td>Dual EDDY head</td>
<td>16 MB/sec</td>
</tr>
<tr>
<td>Laser printer</td>
<td>100 MB/sec</td>
</tr>
<tr>
<td>Scanner</td>
<td>80 MB/sec</td>
</tr>
<tr>
<td>Gigabit Ethernet</td>
<td>1.4 GB/sec</td>
</tr>
<tr>
<td>USB 3.0 external</td>
<td>3.2 GB/sec</td>
</tr>
<tr>
<td>Video camera</td>
<td>6 MB/sec</td>
</tr>
<tr>
<td>IDE drive</td>
<td>12 MB/sec</td>
</tr>
<tr>
<td>DVD drive</td>
<td>24 MB/sec</td>
</tr>
<tr>
<td>IDE ATA disk</td>
<td>16.6 MB/sec</td>
</tr>
<tr>
<td>Exabyte 2 (ATAPI)</td>
<td>50 MB/sec</td>
</tr>
<tr>
<td>IDE drive</td>
<td>80 MB/sec</td>
</tr>
<tr>
<td>Serial/ISA 12-bit</td>
<td>28 MB/sec</td>
</tr>
<tr>
<td>PCI/PCI-X 64-bit</td>
<td>50 MB/sec</td>
</tr>
<tr>
<td>Ethernet 10/100/1000</td>
<td></td>
</tr>
<tr>
<td>Gigabit Ethernet</td>
<td>1 GB/sec</td>
</tr>
<tr>
<td>I/O bus</td>
<td>50 MB/sec</td>
</tr>
<tr>
<td>Bus bandwidth/MB/s (bus)</td>
<td>90 MB/sec</td>
</tr>
</tbody>
</table>

I/O Device Handling Considerations

- 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
    - `lp0: printer on fire!`
  - Expected error rate also differs

I/O Device Handling Considerations

- Layering
  - Need to be both general and specific, e.g.
  - Devices that are the same, but aren’t the same
    - Hard-disk, USB disk, RAM disk
  - Interaction of layers
    - Swap partition and data on same disk
    - Two mice
  - Priority
    - Keyboard, disk, network

Bus Architectures

(a) A single-bus architecture
(b) A dual-bus memory architecture

Intel IXP420
Interrupts

- Devices connected to an Interrupt Controller via lines on an I/O bus (e.g., PCI).
- Interrupt Controller signals interrupt to CPU and is eventually acknowledged.
- Exact details are architecture specific.

I/O Interaction

- Also called polling, or busy waiting
- I/O module (controller) performs the action, not the processor
- Sets appropriate bits in the I/O status register
- No interrupts occur
- Processor checks status until operation is complete
  - Wastes CPU cycles

Programmed I/O

- Processor is interrupted when I/O module (controller) ready to exchange data
- Processor is free to do other work
- No needless waiting
- Consumes a lot of processor time because every word read or written passes through the processor

Direct Memory Access

- Transfers data directly between Memory and Device
- CPU not needed for copying

Direct Memory Access

- Transfers a block of data directly to or from memory
- An interrupt is sent when the task is complete
- The processor is only involved at the beginning and end of the transfer
DMA Considerations

- Reduces number of interrupts
- Less (expensive) context switches or kernel entry-exits
- Requires contiguous regions
- Copying
- Scatter-gather
- Synchronous/Asynchronous
- Shared bus must be arbitrated
  - CPU cache reduces (but not eliminates) CPU need for bus

The Process to Perform DMA Transfer

Evolution of the I/O Function

- Processor directly controls a peripheral device
  - Example: CPU controls a flip-flop to implement a serial line

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
Evolution of the I/O Function

- Direct Memory Access
  - Blocks of data are moved into memory without involving the processor
  - Processor involved at beginning and end only

![Diagram showing CPU, Memory, UART, and Serial Line connections]

Evolution of the I/O Function

- I/O module has a separate processor
  - Example: SCSI controller
  - Controller CPU executes SCSI program code out of main memory

![Diagram showing CPU, Memory, SCSI Controller, and SCSI Cable connections]

Evolution of the I/O Function

- I/O processor
  - I/O module has its own local memory, internal bus, etc.
  - It's a computer in its own right
  - Example: Myrinet 10 gigabit NIC

![Diagram showing CPU, Memory, Myrinet Controller, and Bus connections]