CSCI-GA.2250-001

Operating Systems

Lecture 10: I/O Part: I

Mohamed Zahran (aka Z)
mzahran@cs.nyu.edu
http://www.mzahran.com
A Simple Definition

• The main concept of I/O is to move data from/to I/O devices to the processor using some modules and buffers.

• This is the way the processor deals with the outside world.

• Examples: mouse, display, keyboard, disks, scanners, speakers, etc.
The OS and I/O

• The OS controls all I/O devices
  – Issue commands to devices
  – Catch interrupts
  – Handle errors

• Provides an interface between the devices and the rest of the system
I/O Devices

• Block device
  – Stores information in fixed-size blocks
  – Each block has its own address
  – Transfers in one or more blocks
  – Example: Hard-disks, CD-ROMs, USB sticks

• Character device
  – Delivers or accepts stream of character
  – Is not addressable
  – Example: mice, printers, network interfaces
OS

Applications

File System

Low-Level Interface

I/O Devices
<table>
<thead>
<tr>
<th>Device</th>
<th>Data rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyboard</td>
<td>10 bytes/sec</td>
</tr>
<tr>
<td>Mouse</td>
<td>100 bytes/sec</td>
</tr>
<tr>
<td>56K modem</td>
<td>7 KB/sec</td>
</tr>
<tr>
<td>Scanner</td>
<td>400 KB/sec</td>
</tr>
<tr>
<td>Digital camcorder</td>
<td>3.5 MB/sec</td>
</tr>
<tr>
<td>802.11g Wireless</td>
<td>6.75 MB/sec</td>
</tr>
<tr>
<td>52x CD-ROM</td>
<td>7.8 MB/sec</td>
</tr>
<tr>
<td>Fast Ethernet</td>
<td>12.5 MB/sec</td>
</tr>
<tr>
<td>Compact flash card</td>
<td>40 MB/sec</td>
</tr>
<tr>
<td>FireWire (IEEE 1394)</td>
<td>50 MB/sec</td>
</tr>
<tr>
<td>USB 2.0</td>
<td>60 MB/sec</td>
</tr>
<tr>
<td>SONET OC-12 network</td>
<td>78 MB/sec</td>
</tr>
<tr>
<td>SCSI Ultra 2 disk</td>
<td>80 MB/sec</td>
</tr>
<tr>
<td>Gigabit Ethernet</td>
<td>125 MB/sec</td>
</tr>
<tr>
<td>SATA disk drive</td>
<td>300 MB/sec</td>
</tr>
<tr>
<td>Ultrium tape</td>
<td>320 MB/sec</td>
</tr>
<tr>
<td>PCI bus</td>
<td>528 MB/sec</td>
</tr>
</tbody>
</table>
I/O Units

Mechanical component  
The Device Itself

Electronic Component  
Device Controller

Diagram showing components connected to a bus.
Controller and Device

- Each controller has few registers used to communicate with CPU
- By writing/reading into/from those registers, the OS can control the devices.
- There are also data buffers in the device that can be read/written by the OS
How does CPU communicate with control registers and data buffers?

Two main approaches

- I/O port space
- Memory-mapped I/O
I/O Port Space

- Each control register is assigned an I/O port number
- The set of all I/O ports form the I/O port space
- I/O port space is protected
Memory-Mapped I/O

- Map control registers into the memory space
- Each control register is assigned a unique memory address
Advantages of Memory-Mapped I/O

- Device drivers can be written entirely in C (since no special instructions are needed)
- No special protection is needed from OS, just refrain from putting that portion of the address space in any user’s virtual address space
- Every instruction that can reference memory can also reference control registers
Disadvantages of Memory-Mapped I/O

- Caching a device control register can be disastrous
- Hardware complications
Direct Memory Access (DMA)

- It is not efficient for the CPU to request data from I/O one byte at a time.
- DMA controller has access to the system bus independent of the CPU.
Interrupts

• When an I/O device has finished the work given to it, it causes an interrupt.
Precise Interrupts

- Makes handling interrupts much simplers
- Has 4 properties
  - The program counter (PC) is saved in known place
  - All instructions before the one pointed by PC have fully executed
  - No instruction beyond the one pointed by PC has been executed
  - The execution state of the instruction pointed to by the PC is known
Precise Interrupt

Imprecise Interrupt
I/O Software

- Device independence
- Uniform naming
- Error handling
  - Should be handled as close to the hardware as possible
- Synchronous vs asynchronous (interrupt-driven)
- Buffering
- Sharable verses dedicated devices
Three Ways of Doing I/O

- Programmed I/O
- Interrupt-driven I/O
- I/O Using DMA
Programming I/O

• CPU does all the work
• Busy-waiting (polling)

Example:

```c
void my_function()
{
    int i;
    for (i = 0; i < count; i++)
    {
        while (*printer_status_reg != READY);
        *printer_data_register = p[i];
    }
    return;
}
```
Interrupt-Driven I/O

• Waiting for a device to be ready, the process is blocked and another process is scheduled.

• When the device is ready it raises an interrupt.
I/O Using DMA

• DMA does the work instead of the CPU
• Let the DMA do its work and then interrupts
Conclusions

• The OS provides an interface between the devices and the rest of the system.
• The hardware: CPU, programmable interrupt controller, DMA, device controller, and the device itself.
• Sections: 5.1 and 5.2