I/O Systems

CS 256/456
Dept. of Computer Science, University of Rochester

I/O Hardware

<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>2 Kbps</td>
</tr>
<tr>
<td>Telephone channel</td>
<td>96 Kbps</td>
</tr>
<tr>
<td>Dual-DEC lines</td>
<td>16 Kbps</td>
</tr>
<tr>
<td>Laser printer</td>
<td>100 Kbps</td>
</tr>
<tr>
<td>Scanner</td>
<td>400 Kbps</td>
</tr>
<tr>
<td>IEEE 1394</td>
<td>1.25 MB/sec</td>
</tr>
<tr>
<td>USB (Universal Serial Bus)</td>
<td>1.5 MB/sec</td>
</tr>
<tr>
<td>Digital camcorder</td>
<td>4 MB/sec</td>
</tr>
<tr>
<td>EIDE (UltraDMA)</td>
<td>66 MB/sec</td>
</tr>
<tr>
<td>44x40 CD-ROM</td>
<td>6 MB/sec</td>
</tr>
<tr>
<td>Fast Ethernet</td>
<td>12.5 MB/sec</td>
</tr>
<tr>
<td>EISA bus</td>
<td>16.7 MB/sec</td>
</tr>
<tr>
<td>IDE (ATA-2) disk</td>
<td>16.7 MB/sec</td>
</tr>
<tr>
<td>FlexRay (IEEE 1394)</td>
<td>50 MB/sec</td>
</tr>
<tr>
<td>XGA Monitor</td>
<td>60 MB/sec</td>
</tr>
<tr>
<td>10Gigabit Ethernet</td>
<td>75 MB/sec</td>
</tr>
<tr>
<td>SCSI Ultra 2 disk</td>
<td>80 MB/sec</td>
</tr>
<tr>
<td>Gigabit Ethernet</td>
<td>175 MB/sec</td>
</tr>
<tr>
<td>Lithium tape</td>
<td>220 MB/sec</td>
</tr>
<tr>
<td>PCI bus</td>
<td>538 MB/sec</td>
</tr>
<tr>
<td>Sun Gigaplane X2 bus</td>
<td>20 GB/sec</td>
</tr>
</tbody>
</table>

Some typical device, network, and bus data rates:

Device Controllers

- I/O devices have mechanical component & electronic component
- The electronic component is the device controller
  - It contains control logic, command registers, status registers, and on-board buffer space

I/O Ports & Memory-Mapped I/O

I/O methods:
- Separate I/O and memory space; special I/O commands (IN/OUT)
- Memory-mapped I/O

Issues with them:
- Convenience/efficiency when use high-level language;
- Protection mechanisms;
- Special data access schemes: TSL
- Data caching
Direct Memory Access (DMA)

- Are the addresses CPU sends to the DMA controller virtual or physical addresses?
- Can the disk controller directly read data into the main memory (bypassing the controller buffer)?

How is I/O accomplished?

- **Polling-based**
  - CPU spins and polls the I/O until it completes

- **Interrupt-driven**
  - CPU initiates I/O and then does something else; gets notified when the I/O is done (interrupts)

Polling-based I/O

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

Interrupt-Driven I/O

```c
void copy_from_user(char *buffer, int count)
{
    // (a)
    char *start = buffer;
    while (count-- > 0)
    {
        *buffer = (*buffer) + 'A';
        buffer += 1;
    }
    return;
}
```
Interrupt Handlers
1. Save registers of the old process
2. Set up context for interrupt service procedure (switch from the user space to kernel space: MMU, stack, ...)
3. Run service procedure; when safe, re-enable interrupts
4. Run scheduler to choose the new process to run next
5. Set up context (MMU, registers) for process to run next
6. Start running the new process

How much cost is it? Is it a big deal?
For Gigabit Ethernet, each packet arrives once every 12us.

I/O Software Layers
- Device-dependent OS I/O software; directly interacts with controller hardware
- Interface to upper-layer OS code is standardized.

Device Drivers
- Device driver is the device-specific part of the kernel-space I/O software. It also includes interrupt handlers.
- Device drivers must run in kernel mode. Why?
  - The crash of a device driver brings down the whole system.
  - Device drivers are probably the bugiest part of the OS. Why?

Upper-level I/O Software
- Device independence
  - reuse software as much as possible across different types of devices
- Buffering
  - data coming off a device is stored in intermediate buffers
- Need for buffering
  - speed matching with I/O devices
  - caching
  - speculative I/O
Disclaimer

- Parts of the lecture slides contain original work of Abraham Silberschatz, Peter B. Galvin, Greg Gagne, Andrew S. Tanenbaum, and Gary Nutt. The slides are intended for the sole purpose of instruction of operating systems at the University of Rochester. All copyrighted materials belong to their original owner(s).