I/O and Operating Systems

- Controlling input/output devices is complicated by the wide variety of devices.
- The variety of different methods used to control these devices comprise the I/O subsystem.
- This wide variety of devices conflict the push toward the standardization of interfaces, leading to the use of device driver modules.
Device Controllers: An Example

The NEC PD 765 disk controller has a 16-command language, using 1 to 9 bytes in a device register. Commands include:

- Reading and writing data
- Moving the disk arm
- Formatting tracks
- Initializing, calibrating, sensing and resetting the controller and its drives.

Using A Device Controller

READ and WRITE commands require parameters such as disk block address, # of sectors per track, recording mode, intersector gap spacing, etc.

The PD 765 returns 23 status and error fields in 7 bytes.
## I/O Port Addresses

<table>
<thead>
<tr>
<th>I/O Address Range</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>000-00F</td>
<td>DMA Controller</td>
</tr>
<tr>
<td>020-021</td>
<td>Interrupt Controller</td>
</tr>
<tr>
<td>040-043</td>
<td>Timer</td>
</tr>
<tr>
<td>200-20F</td>
<td>Game Controller</td>
</tr>
<tr>
<td>2F8-2FF</td>
<td>Secondary Serial Port</td>
</tr>
<tr>
<td>320-32F</td>
<td>Hard Disk Controller</td>
</tr>
<tr>
<td>378-37F</td>
<td>Parallel Port</td>
</tr>
<tr>
<td>3D0-3DF</td>
<td>Graphics Controller</td>
</tr>
<tr>
<td>3F0-3F7</td>
<td>Floppy Disk Controller</td>
</tr>
<tr>
<td>3F8-3FF</td>
<td>Primary Serial Port</td>
</tr>
</tbody>
</table>

## Typical I/O Port

- The typical I/O port consists of 4 registers:
  - Status – contains bits read by the host system.
  - Control – contains bits written by the host system.
  - Data-in – read by the host as input.
  - Data-out – written by the host as output.
Using Polling To Start An I/O Operation

1. System interface
   - write(dev..., ...)
   - Data

2. Hardware interface
   - read function
   - write function

3. Device Controller
   - Command
   - Status
   - Data

I/O Performed By Polling

4. System interface
   - write(dev..., ...)
   - Data

5. Hardware interface
   - read function
   - write function

6. Device Controller
   - Command
   - Status
   - Data
Using Interrupts To Start An I/O Operation

1. System interface
   - `read(, dev..., ...)`
2. Hardware interface
   - `read driver`
   - `write driver`
3. Device status table
   - Device Handler
   - Interrupt Handler
4. Command | Status | Data
   - Device Controller

Completing Interrupt-Driven I/O

5. System interface
   - `read(, dev..., ...)`
6. Hardware interface
   - `read driver`
   - `write driver`
7. Device status table
   - Device Handler
8. Command | Status | Data
   - Device Controller
Maskable vs. NonMaskable Interrupts

<table>
<thead>
<tr>
<th>Vector Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>divide error</td>
</tr>
<tr>
<td>1</td>
<td>debug exception</td>
</tr>
<tr>
<td>2</td>
<td>null interrupt</td>
</tr>
<tr>
<td>3</td>
<td>breakpoint</td>
</tr>
<tr>
<td>4</td>
<td>INTO-detected overflow</td>
</tr>
<tr>
<td>5</td>
<td>bound range exception</td>
</tr>
<tr>
<td>6</td>
<td>invalid opcode</td>
</tr>
<tr>
<td>7</td>
<td>device not available</td>
</tr>
<tr>
<td>8</td>
<td>Double fault</td>
</tr>
<tr>
<td>15, 19-31</td>
<td>Intel reserved</td>
</tr>
<tr>
<td>32-255</td>
<td>Maskable interrupts</td>
</tr>
</tbody>
</table>

Direct Memory Access

- Processor
- cache
- DMA/bus/interrupt Controller
- CPU memory bus
- memory buffer
- PCI Bus
- IDE Disk Controller
- disks
Application I/O Interface

Kernel

Kernel I/O Subsystem

<table>
<thead>
<tr>
<th>SCSI Device Driver</th>
<th>keyboard Device Driver</th>
<th>mouse Device Driver</th>
<th>…</th>
<th>PCI Bus Device Driver</th>
<th>Floppy Disk Device Driver</th>
<th>ATAPI Device Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCSI Device Controller</td>
<td>keyboard Device Controller</td>
<td>mouse Device Controller</td>
<td>…</td>
<td>PCI Bus Device Controller</td>
<td>Floppy Disk Device Controller</td>
<td>ATAPI Device Controller</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SCSI Devices</th>
<th>keyboard</th>
<th>mouse</th>
<th>…</th>
<th>PCI Bus</th>
<th>Floppy Disk Drive</th>
<th>ATAPI Devices</th>
</tr>
</thead>
</table>

I/O Device Characteristics

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Variation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data-Transfer Mode</td>
<td>Character Block</td>
<td>Terminal Disk</td>
</tr>
<tr>
<td>Access Method</td>
<td>Sequential Random</td>
<td>Modem CD-ROM</td>
</tr>
<tr>
<td>Transfer Schedule</td>
<td>Synchronous Asynchronous</td>
<td>Tape Keyboard</td>
</tr>
<tr>
<td>Sharing</td>
<td>Dedicated Sharable</td>
<td>Tape Keyboard</td>
</tr>
<tr>
<td>I/O Direction</td>
<td>Read only Write Only Read-Write</td>
<td>CD-ROM Graphics controller Disk</td>
</tr>
</tbody>
</table>
Block and Character Devices

- The essential operations of block devices are **read**, **write** and **seek**.
- **Raw I/O** – accessing a block devices as an array of blocks
- Character-stream devices such as keyboard use the basic operations **get** and **put**.

Network Devices

- Sockets are a network interface used by many operating systems.
- Sockets operations includes:
  - creating a socket
  - connecting local socket to a remote address
  - listening for remote application to connect into local sockets
  - sending and receiving packets
Clocks and Timers

• Computer clocks and timers have 3 basic operations:
  – Giving the current time
  – Giving the elapsed time
  – Triggering operation $X$ at time $T$

Blocking and Nonblocking I/O

• When an application calls for input or output, the operating system blocks the process until the I/O operation is completed. Such a system call is called a *blocking system call*.
• Some application cannot work properly if system calls block the process, e.g., video applications. These use *nonblocking system calls*. 
Implementing Nonblocking I/O

- Nonblocking I/O requires that execution and I/O occur concurrently. This can be implemented by writing multithreaded programs.
- An alternative involves nonblocking system calls which do not wait for I/O to be completed.

Kernel I/O Subsystem

The I/O subsystem provides these services:
- I/O Scheduling
- Buffering
- Caching
- Spooling
- Device Reservation
- Error handling
I/O Scheduling

- The order in which input/output requests are made is rarely the best order in which to perform them.
- I/O request scheduling algorithms seek to maximize performance while guaranteeing that all input/output are fulfilled within a reasonable time frame.

Buffering

[Bar chart showing device transfer rates on a logarithmic scale]
Caching

• A cache is a copy of input/output data in memory that can be accessed more quickly than what is stored externally.
• Although buffering and caching are distinct functions, there are cases where the same area of memory is used for both purposes, e.g., when reading disk data.

Spooling and Device Reservation

• A spool is a buffer for output intended for a device that cannot accept interleaved data streams.
• Printers are an example of such devices.
Error Handling

- Operating systems should be able to handle transient hardware errors. This is usually done by trying the failed operation again.
- I/O system calls return a bit of information that indicates whether the operation was a success.
- In some cases either the operating system or the hardware itself may provide more specific information about a failure.

Kernel Data Structures
Transforming I/O Requests to Hardware Operations

- request I/O
  - system call
    - yes (can already satisfy request)
    - no
      - send request to device driver, block process if appropriate
      - process request, issue commands to controller, configure controller to block until interrupted
      - device controller commands
      - monitor device, interrupt when I/O is completed

- user process
  - kernel I/O subsystem
  - device driver
  - interrupt handler
  - keyboard device controller

- I/O completed
  - input data available (if input)
    - system call return
      - transfer data (if appropriate) to process, return completion or error code
      - process request, issue commands to controller, configure controller to block until interrupted
      - receive interrupt
        - store data in driver buffer if input
      - signal to unblock device driver
      - I/O is completed, generate interrupt

Performance

- I/O is a major factor in system performance.
- Interrupt handling is computationally expensive; anything that limits it can increase system performance.
- Network traffic can seriously affect system performance.
- Some systems use front-end processor to terminal I/O.