Chapter 13: I/O Systems

- I/O Hardware
- Application I/O Interface
- Kernel I/O Subsystem
- Transforming I/O Requests to Hardware Operations
- Streams
- Performance

I/O Hardware

- Incredible variety of I/O devices
- Common concepts
  - Port
  - Bus (daisy chain or shared direct access)
  - Controller (host adapter)
- I/O instructions control devices
- Devices have addresses, used by
  - Direct I/O instructions
  - Memory-mapped I/O

A Typical PC Bus Structure
### Device I/O Port Locations on PCs (partial)

<table>
<thead>
<tr>
<th>I/O address range (hexadecimal)</th>
<th>device</th>
</tr>
</thead>
<tbody>
<tr>
<td>000-00F</td>
<td>DMA controller</td>
</tr>
<tr>
<td>028-021</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>218-21F</td>
<td>serial port (secondary)</td>
</tr>
<tr>
<td>300-30F</td>
<td>hard-disk controller</td>
</tr>
<tr>
<td>378-37F</td>
<td>parallel port</td>
</tr>
<tr>
<td>300-30F</td>
<td>graphics controller</td>
</tr>
<tr>
<td>39F-3FF</td>
<td>diskette drive controller</td>
</tr>
<tr>
<td>2FB-3FF</td>
<td>serial port (primary)</td>
</tr>
</tbody>
</table>

### Polling

- Determines state of device
  - command-ready
  - busy
  - Error
- Busy-wait cycle to wait for I/O from device

### Interrupts

- CPU Interrupt request line triggered by I/O device
- Interrupt handler receives interrupts
- Maskable to ignore or delay some interrupts
- Interrupt vector to dispatch interrupt to correct handler
  - Based on priority
  - Some unmaskable
- Interrupt mechanism also used for exceptions
Interrupt-Driven I/O Cycle

1. Device does I/O
2. CPU requests service for I/O
3. I/O controller
4. I/O service
5. Return

Intel Pentium Processor Event-Vector Table

<table>
<thead>
<tr>
<th>vector number</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>disable error</td>
</tr>
<tr>
<td>1</td>
<td>bus error</td>
</tr>
<tr>
<td>2</td>
<td>I/O abend</td>
</tr>
<tr>
<td>3</td>
<td>segmentation fault</td>
</tr>
<tr>
<td>4</td>
<td>illegal instruction</td>
</tr>
<tr>
<td>5</td>
<td>general protection exception</td>
</tr>
<tr>
<td>6</td>
<td>memory page fault</td>
</tr>
<tr>
<td>7</td>
<td>processor limit</td>
</tr>
<tr>
<td>8</td>
<td>real address violation</td>
</tr>
<tr>
<td>9</td>
<td>segment not present</td>
</tr>
<tr>
<td>10</td>
<td>stack fault</td>
</tr>
<tr>
<td>11</td>
<td>page fault</td>
</tr>
<tr>
<td>12</td>
<td>breakpoint exception</td>
</tr>
<tr>
<td>13</td>
<td>page fault</td>
</tr>
<tr>
<td>14</td>
<td>virtual address violation</td>
</tr>
<tr>
<td>15</td>
<td>trap</td>
</tr>
<tr>
<td>16</td>
<td>illegal instruction</td>
</tr>
<tr>
<td>17</td>
<td>alignment check</td>
</tr>
<tr>
<td>18</td>
<td>external</td>
</tr>
<tr>
<td>19-255</td>
<td>reserved</td>
</tr>
</tbody>
</table>

Direct Memory Access

- Used to avoid programmed I/O for large data movement
- Requires DMA controller
- Bypasses CPU to transfer data directly between I/O device and memory
### Six Step Process to Perform DMA Transfer

1. Device driver reads block from buffer in device controller.
2. Device driver tells DMA controller to transfer.
3. DMA controller takes control of address B.
4. DMA controller reads buffer from memory block.
5. DMA controller loads buffer into the device controller.
6. Device controller sends data to device.

### Application I/O Interface

- I/O system calls encapsulate device behaviors in generic classes.
- Device-driver layer hides differences among I/O controllers from kernel.
- Devices vary in many dimensions:
  - Character-stream or block
  - Sequential or random-access
  - Shareable or dedicated
  - Speed of operation
  - Read-write, read only, or write only

### A Kernel I/O Structure
Characteristics of I/O Devices

<table>
<thead>
<tr>
<th>aspect</th>
<th>variation</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>data-transfer mode</td>
<td>character</td>
<td>terminal</td>
</tr>
<tr>
<td></td>
<td>block</td>
<td>disk</td>
</tr>
<tr>
<td>access method</td>
<td>sequential</td>
<td>modem</td>
</tr>
<tr>
<td></td>
<td>random access</td>
<td>CD-ROM</td>
</tr>
<tr>
<td>transfer schedule</td>
<td>synchronous</td>
<td>tape</td>
</tr>
<tr>
<td></td>
<td>asynchronous</td>
<td>keyboard</td>
</tr>
<tr>
<td>sharing</td>
<td>dedicated</td>
<td>tape</td>
</tr>
<tr>
<td></td>
<td>shareable</td>
<td>keyboard</td>
</tr>
<tr>
<td>device speed</td>
<td>memory</td>
<td>tape</td>
</tr>
<tr>
<td></td>
<td>seek</td>
<td>keyboard</td>
</tr>
<tr>
<td></td>
<td>line</td>
<td>dedicate</td>
</tr>
<tr>
<td></td>
<td>delay</td>
<td>seek</td>
</tr>
<tr>
<td>I/O direction</td>
<td>read-only</td>
<td>dedicated</td>
</tr>
<tr>
<td></td>
<td>write-only</td>
<td>seek</td>
</tr>
<tr>
<td></td>
<td>read/write</td>
<td>ROM</td>
</tr>
<tr>
<td></td>
<td>block</td>
<td>disk</td>
</tr>
</tbody>
</table>

Block and Character Devices

- Block devices include disk drives
  - Commands include read, write, seek
  - Raw I/O or file-system access
  - Memory-mapped file access possible
- Character devices include keyboards, mice, serial ports
  - Commands include get, put
  - Libraries layered on top allow line editing

Network Devices

- Varying enough from block and character to have own interface
- Unix and Windows NT/9x/2000 include socket interface
  - Separates network protocol from network operation
  - Includes select functionality
- Approaches vary widely (pipes, FIFOs, streams, queues, mailboxes)
Clocks and Timers

- Provide current time, elapsed time, timer
- If programmable interval time used for timings, periodic interrupts
- ioctl (on UNIX) covers odd aspects of I/O such as clocks and timers

Blocking and Nonblocking I/O

- Blocking - process suspended until I/O completed
  - Easy to use and understand
  - Insufficient for some needs
- Nonblocking - I/O call returns as much as available
  - User interface, data copy (buffered I/O)
  - Implemented via multi-threading
  - Returns quickly with count of bytes read or written
- Asynchronous - process runs while I/O executes
  - Difficult to use
  - I/O subsystem signals process when I/O completed

Kernel I/O Subsystem

- Scheduling
  - Some I/O request ordering via per-device queue
  - Some OSs try fairness
- Buffering - store data in memory while transferring between devices
  - To cope with device speed mismatch
  - To cope with device transfer size mismatch
  - To maintain “copy semantics”
**Kernel I/O Subsystem**

- **Caching** - fast memory holding copy of data
  - Always just a copy
  - Key to performance
- **Spooling** - hold output for a device
  - If device can serve only one request at a time
    - I.e., Printing
- **Device reservation** - provides exclusive access to a device
  - System calls for allocation and deallocation
  - Watch out for deadlock

**Error Handling**

- OS can recover from disk read, device unavailable, transient write failures
- Most return an error number or code when I/O request fails
- System error logs hold problem reports
Kernel Data Structures

- Kernel keeps state info for I/O components, including open file tables, network connections, character device state.
- Many, many complex data structures to track buffers, memory allocation, “dirty” blocks.
- Some use object-oriented methods and message passing to implement I/O.

UNIX I/O Kernel Structure

I/O Requests to Hardware Operations

- Consider reading a file from disk for a process:
  - Determine device holding file
  - Translate name to device representation
  - Physically read data from disk into buffer
  - Make data available to requesting process
  - Return control to process
STREAMS

- STREAM – a full-duplex communication channel between a user-level process and a device

- A STREAM consists of:
  - STREAM head interfaces with the user process
  - driver end interfaces with the device
  - zero or more STREAM modules between them.

- Each module contains a read queue and a write queue

- Message passing is used to communicate between queues

The STREAMS Structure
Performance

- I/O a major factor in system performance:
  - Demands CPU to execute device driver, kernel I/O code
  - Context switches due to interrupts
  - Data copying
  - Network traffic especially stressful

Intercomputer Communications

Improving Performance

- Reduce number of context switches
- Reduce data copying
- Reduce interrupts by using large transfers, smart controllers, polling
- Use DMA
- Balance CPU, memory, bus, and I/O performance for highest throughput
Device-Functionality Progression

- new algorithm
- application code
- kernel code
- device-driver code
- device-controller code (hardware)
- device code (hardware)

Increased efficiency
Increased abstractions
Decreased development cost
Decreased packaging
Decreased packaging

Effective packaging