CSMC 412
Operating Systems
Prof. Ashok K Agrawala

© 2006 Ashok Agrawala
Set 13
I/O Systems

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

• Explore the structure of an operating system’s I/O subsystem
• Discuss the principles of I/O hardware and its complexity
• Provide details of the performance aspects of I/O hardware and software
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
# 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>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>serial port (secondary)</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>diskette-drive controller</td>
</tr>
<tr>
<td>3F8–3FF</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 **non-maskable**

• Interrupt mechanism also used for exceptions
Interrupt-Driven I/O Cycle

1. CPU
   - device driver initiates I/O

2. Initiates I/O

3. Input ready, output complete, or error generates interrupt signal

4. CPU receiving interrupt, transfers control to interrupt handler
   - interrupt handler processes data, returns from interrupt
   - CPU resumes processing of interrupted task
## Intel Pentium Processor Event-Vector Table

<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>9</td>
<td>coprocessor segment overrun (reserved)</td>
</tr>
<tr>
<td>10</td>
<td>invalid task state segment</td>
</tr>
<tr>
<td>11</td>
<td>segment not present</td>
</tr>
<tr>
<td>12</td>
<td>stack fault</td>
</tr>
<tr>
<td>13</td>
<td>general protection</td>
</tr>
<tr>
<td>14</td>
<td>page fault</td>
</tr>
<tr>
<td>15</td>
<td>(Intel reserved, do not use)</td>
</tr>
<tr>
<td>16</td>
<td>floating-point error</td>
</tr>
<tr>
<td>17</td>
<td>alignment check</td>
</tr>
<tr>
<td>18</td>
<td>machine check</td>
</tr>
<tr>
<td>19-31</td>
<td>(Intel reserved, do not use)</td>
</tr>
<tr>
<td>32-255</td>
<td>maskable interrupts</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 is told to transfer disk data to buffer at address X
2. Device driver tells disk controller to transfer C bytes from disk to buffer at address X
3. Disk controller initiates DMA transfer
4. Disk controller sends each byte to DMA controller
5. DMA controller transfers bytes to buffer X, increasing memory address and decreasing C until C = 0
6. When C = 0, DMA interrupts CPU to signal transfer completion
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
  – Sharable or dedicated
  – Speed of operation
  – read-write, read only, or write only
A Kernel I/O Structure

- **Software**
  - kernel
  - kernel I/O subsystem
    - SCSI device driver
    - keyboard device driver
    - mouse device driver
    - ... (continues)
    - PCI bus device driver
    - floppy device driver
    - ATAPI device driver

- **Hardware**
  - SCSI device controller
  - keyboard device controller
  - mouse device controller
  - ... (continues)
  - PCI bus device controller
  - floppy device controller
  - ATAPI device controller

- Devices:
  - SCSI devices
  - keyboard
  - mouse
  - ... (continues)
  - PCI bus
  - floppy-disk drives
  - ATAPI devices (disks, tapes, drives)
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 block</td>
<td>terminal disk</td>
</tr>
<tr>
<td>access method</td>
<td>sequential</td>
<td>modem CD-ROM</td>
</tr>
<tr>
<td></td>
<td>random</td>
<td></td>
</tr>
<tr>
<td>transfer schedule</td>
<td>synchronous</td>
<td>tape keyboard</td>
</tr>
<tr>
<td></td>
<td>asynchronous</td>
<td></td>
</tr>
<tr>
<td>sharing</td>
<td>dedicated</td>
<td>tape keyboard</td>
</tr>
<tr>
<td></td>
<td>sharable</td>
<td></td>
</tr>
<tr>
<td>device speed</td>
<td>latency</td>
<td></td>
</tr>
<tr>
<td></td>
<td>seek time</td>
<td></td>
</tr>
<tr>
<td></td>
<td>transfer rate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>delay between operations</td>
<td></td>
</tr>
<tr>
<td>I/O direction</td>
<td>read only</td>
<td>CD-ROM graphics controller disk</td>
</tr>
<tr>
<td></td>
<td>write only</td>
<td></td>
</tr>
<tr>
<td></td>
<td>read–write</td>
<td></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
- **Programmable interval timer** 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
Two I/O Methods

(a) Synchronous

(b) Asynchronous
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”
# Device-status Table

<table>
<thead>
<tr>
<th>Device</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyboard</td>
<td>idle</td>
</tr>
<tr>
<td>Laser printer</td>
<td>busy</td>
</tr>
<tr>
<td>Mouse</td>
<td>idle</td>
</tr>
<tr>
<td>Disk unit 1</td>
<td>idle</td>
</tr>
<tr>
<td>Disk unit 2</td>
<td>busy</td>
</tr>
</tbody>
</table>

- Request for laser printer
  - Address: 38546
  - Length: 1372

- Request for disk unit 2
  - File: xxx
  - Operation: read
  - Address: 43046
  - Length: 20000

- Request for disk unit 2
  - File: yyy
  - Operation: write
  - Address: 03458
  - Length: 500
Sun Enterprise 6000 Device-Transfer Rates

- Gigaplane bus
- SBUS
- SCSI bus
- Fast Ethernet
- Hard disk
- Ethernet
- Laser printer
- Modem
- Mouse
- Keyboard
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
I/O Protection

• User process may accidentally or purposefully attempt to disrupt normal operation via illegal I/O instructions
  – All I/O instructions defined to be privileged
  – I/O must be performed via system calls
    • Memory-mapped and I/O port memory locations must be protected too
Use of a System Call to Perform I/O

1. trap to monitor
2. perform I/O
3. return to user

- system call $n$
- read
- case $n$
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

- **file descriptor**
  - per-process open-file table
  - user-process memory

- **system-wide open-file table**
  - file-system record
    - inode pointer
    - pointer to read and write functions
    - pointer to select function
    - pointer to ioctl function
    - pointer to close function

- **networking (socket) record**
  - pointer to network info
  - pointer to read and write functions
  - pointer to select function
  - pointer to ioctl function
  - pointer to close function

- **active-inode table**
- **network-information table**

- **kernel memory**
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 in Unix System V and beyond

- 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

![Diagram of intercomputer communication process]

- Character typed
- System call completes
- Network packet received
- Network adapter
- Device driver
- Kernel
- User process

Sending system:
- Context switch
- Sending system

Receiving system:
- Context switch
- Receiving system
- Network subdaemon
- Kernel
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

- increased time (generations)
- increased efficiency
- increased development cost
- increased abstraction

new algorithm

application code

kernel code

device-driver code

device-controller code (hardware)

device code (hardware)

increased flexibility