Chapter 13: I/O Systems

肖 卿 俊
办公室：计算机楼212室
电邮：csqjxiao@seu.edu.cn
主页：http://cse.seu.edu.cn/PersonalPage/csqjxiao
电话：025-52091022
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
# 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 nonmaskable
- Interrupt mechanism also used for exceptions
Interrupt-Driven I/O Cycle

1. CPU
   - device driver initiates I/O
2. I/O controller
   - initiates I/O
3. CPU executing checks for interrupts between instructions
4. CPU receiving interrupt, transfers control to interrupt handler
   - interrupt handler processes data, returns from interrupt
5. CPU resumes processing of interrupted task
   - input ready, output complete, or error generates interrupt signal
## 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

- **Kernel I/O Subsystem**
  - kernel
  - SCSI device driver
  - keyboard device driver
  - mouse device driver
  - PCI bus device driver
  - floppy device driver
  - ATAPI device driver

- **Hardware**
  - SCSI devices
  - keyboard
  - mouse
  - 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></td>
<td>block</td>
<td></td>
</tr>
<tr>
<td>access method</td>
<td>sequential random</td>
<td>modem CD-ROM</td>
</tr>
<tr>
<td></td>
<td>random</td>
<td></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></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</td>
</tr>
<tr>
<td></td>
<td>write only</td>
<td>disk</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
Blocking and Nonblocking I/O

- **Asynchronous** - process runs while I/O executes
  - Difficult to use
  - I/O subsystem signals process when I/O completed
Two I/O Methods

Synchronous

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>

### Requests
- **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
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 $n$

user program

kernel

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
  - User-process memory
  - Per-process open-file table
  - 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
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
Life Cycle of An I/O Request

1. **Request I/O**
   - User process
   - System call

2. **Can already satisfy request?**
   - Yes
   - Kernel I/O subsystem
   - Transfer data (if appropriate) to process, return completion or error code
   - Return from system call

   - No
   - Send request to device driver, block process if appropriate

3. **Process request, issue commands to controller, configure controller to block until interrupted**
   - Device-controller commands
   - Receive interrupt, store data in device-driver buffer if input, signal to unblock device driver
   - Interrupt

4. **Monitor device, interrupt when I/O completed**
   - Device controller
   - I/O completed, generate interrupt

5. **Determine which I/O completed, indicate state change to I/O subsystem**

6. **Receive interrupt, store data in device-driver buffer if input, signal to unblock device driver**

7. **Send request to device driver, block process if appropriate**

8. **Process request, issue commands to controller, configure controller to block until interrupted**

9. **Monitor device, interrupt when I/O completed**

10. **I/O completed, input data available, or output completed**
    - Return from system call
    - Transfer data (if appropriate) to process, return completion or error code
    - Return from system call
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
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)