I/O Systems
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
A Typical PC Bus Structure

- Monitor
- Processor
- Graphics controller
- Bridge/memory controller
- Cache
- Memory
- IDE disk controller
- Expansion bus interface
- Expansion bus
- SCSI bus
- Disk
- Keyboard
- Parallel port
- Serial port
### 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
   - CPU executing checks for interrupts between instructions
     - CPU receiving interrupt, transfers control to interrupt handler
       - interrupt handler processes data, returns from interrupt
         - CPU resumes processing of interrupted task

2. I/O controller
   - initiates I/O
     - input ready, output complete, or error generates interrupt signal

3. CPU
   - device driver initiates I/O
   - CPU executing checks for interrupts between instructions
     - CPU receiving interrupt, transfers control to interrupt handler
       - interrupt handler processes data, returns from interrupt
         - CPU resumes processing of interrupted task
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

<table>
<thead>
<tr>
<th>Hardware devices</th>
<th>Hardware controllers</th>
<th>Software interface</th>
<th>Software drivers</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCSI devices</td>
<td>keyboard device</td>
<td>keyboard device</td>
<td>SCSI device</td>
</tr>
<tr>
<td>mouse</td>
<td>device</td>
<td>device</td>
<td>device driver</td>
</tr>
<tr>
<td>floppy-disk</td>
<td>drive</td>
<td>floppy device</td>
<td>floppy device</td>
</tr>
<tr>
<td>drives</td>
<td></td>
<td>driver</td>
<td>ATAPI device</td>
</tr>
<tr>
<td>controllers</td>
<td></td>
<td></td>
<td>driver driver</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **SCSI device**
- **keyboard device**
- **mouse device**
- **PCI bus device**
- **floppy device**
- **ATAPI device**

**Kernel I/O Subsystem**

**Kernel**
## 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 random</td>
<td>modem CD-ROM</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 delay</td>
<td></td>
</tr>
<tr>
<td></td>
<td>between operations</td>
<td></td>
</tr>
<tr>
<td>I/O direction</td>
<td>read only</td>
<td>CD-ROM graphics</td>
</tr>
<tr>
<td></td>
<td>write only</td>
<td>controller 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

- Differ sufficiently 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 attempt to provide 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”
<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
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 errors, such as disk read failure, 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

kernel

system call $n$

user program

 case $n$

CIS 415, Spring 2015

IO Systems
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
I/O Requests → 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. User process requests I/O.
2. System call is made.
3. Can I/O request already be satisfied? (Yes/No)
   - Yes: I/O completed, transfer data (if appropriate) to process, return completion or error code.
   - No: Send request to device driver, block process if appropriate.
4. Process request, issue commands to controller, configure controller to block until interrupted.
5. Monitor device, interrupt when I/O completed.
6. Receive interrupt, store data in device-driver buffer if input, signal to unblock device driver.
7. I/O completed, generate interrupt.
8. Return from system call.
9. Time arrow indicates flow of events.
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)

Increased flexibility