Module 12: I/O Systems

- I/O Hardware
- Application I/O Interface
- Kernel I/O Subsystem
- Transforming I/O Requests to Hardware Operations
- 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
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-drive I/O Cycle**

1. **CPU** executes checks for interrupts between instructions.
2. **Device driver initiates I/O**.
3. **I/O controller initiates I/O**.
4. **CPU receiving interrupt, transfers control to interrupt handler**.
5. **Interrupt handler processes data, returns from interrupt**.
6. **CPU resumes processing of interrupted task**.
7. **Input ready, output complete, or error generates interrupt signal**.

---

**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
  - Synchronous or asynchronous
  - Sharable or dedicated
  - Speed of operation
  - read-write, read only, or write only
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 include socket interface
  - Separates network protocol from network operations
  - Includes select functionality
- Approaches vary widely (pipes, FIFOs, streams, queues, mailboxes)
**Clocks and Timers**

- Provide current time, elapsed time, timer
- An 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
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 - holds 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
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. User process requests I/O.
2. Kernel I/O subsystem processes request, issues commands to controller, and configures controller to block until interrupted.
3. Device controller commands device.
4. Device controller interrupts device handler.
5. Device handler determines which I/O completed, indicates state change to I/O subsystem.
6. I/O subsystem transfers data (if appropriate) to process, returns completion or error code.
7. User process terminates 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

Intercomputer communications

[Diagram showing intercomputer communications process]
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