

Input/Output (I/O) Systems

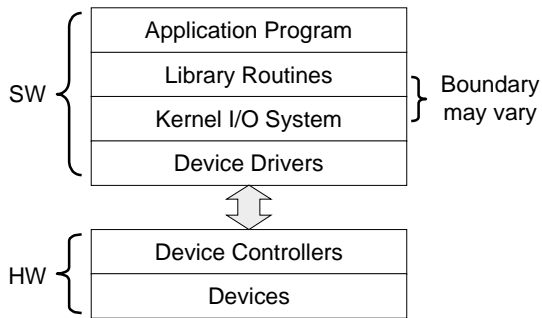
• **Have already discussed:**

- I/O hardware, bus, controller
- Polling, interrupts, DMA
- Software interrupts (traps), timers

• **Now, the OS part:**

- I/O API, I/O modes, buffering
- Device types, spooling, allocation
- I/O request handling

Typical OS I/O Architecture



Input/Output API

• **Application Program Interface**

• **Various levels**

- Language syntax/libraries
- System libraries
- Actual system (kernel) calls

• **Tends to abstract operations**

- Mask differences between devices

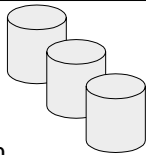
Device Characteristics

- **Transfer mode**
 - Character
 - Block
- **Access**
 - Sequential
 - Random
- **Schedule**
 - Synchronous
 - Asynchronous
- **Sharing**
 - Dedicated
 - Sharable
- **Speed**
- **I/O direction**
 - Read
 - Write
 - Read/write

Examples?

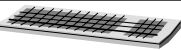
Block Devices

- **Disk drives, etc.**
- **Access mode**
 - Raw block read/writes
 - Mediated through file system
- **Memory-mapped I/O**
 - Through virtual memory system

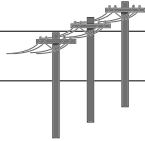


Character Stream Devices

- **Keyboard, modem, etc.**
- **Transfer mode**
 - Raw character read/writes (0-n bytes)
 - Buffered
 - Intra-line editing
 - Previous line recall
- **May not really be “characters”**
 - E.g., analog signals, audio, mouse



Network Devices



•Two meanings

- Actual network interface (Ethernet?)
- Pseudo-device (represents a link)

•Socket interface

- Appears as a character device
- “Out of band” control channel
- “Well-known” addresses (ports)

Blocking I/O Operations

•Process initiates I/O

•Blocked from execution

- Waits in a queue?

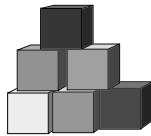
Possible to overlap I/O and other processing?

•I/O completes

•Process awakened

- Returns to “ready” state

•I/O data processed



Non-Blocking I/O Operations

•Process/thread initiates I/O

•Continues to execute

- Running or ready state

•I/O completion notification

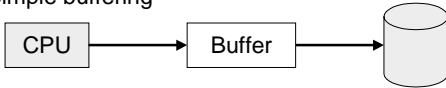
- I/O status in data structure (polling)
- Event flag (can “wait” on flag)
- Asynchronous trap routine
 - Like UNIX signal handler

Data Buffering

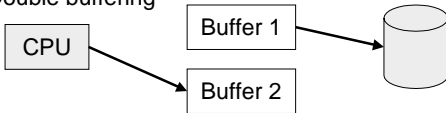
- **Application data buffer**
 - In program virtual address space
 - Lock in memory during actual I/O?
- **What if device is very slow?**
 - E.g., keyboard input
- **May use OS buffer temporarily**
 - Allow application to be swapped out
 - Copy data between buffers

Buffering Options

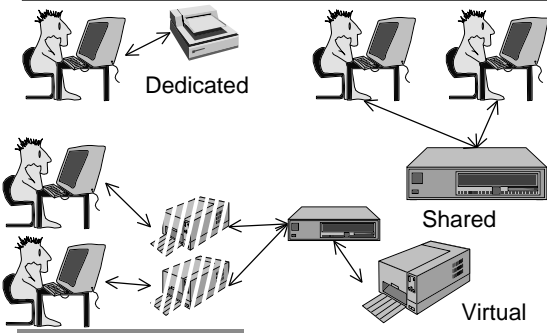
Simple buffering



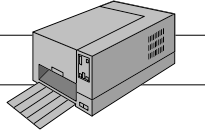
Double buffering



Device Sharing



Spooling



- **Printer (for example)**
 - A dedicated device - one job at a time
 - Would like to share it
- **Create a virtual device**
 - Appears dedicated to a process
 - Data actually buffered
- **Managed by a system process**
 - Print queue and spooler

Device Allocation

- **True dedicated device**
 - E.g., A/D, D/A, audio card
- **Only one process can use it**
 - Otherwise, destructive conflict results
- **Allocate (reserve) before use**
 - Process may block if not available
- **Deallocate (free) when done**
 - Make available to next waiting process

I/O Request Processing

