

BILKENT UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING
CS342 OPERATING SYSTEMS

LECTURE 13
INPUT/OUTPUT (I/O) SYSTEMS
(CHAPTER 13)

Dr. İbrahim Körpeoğlu
http://www.cs.bilkent.edu.tr/~korpe

References

- *The slides here are adapted/modified from the textbook and its slides: Operating System Concepts, Silberschatz et al., 7th & 8th editions, Wiley.*

REFERENCES

- Operating System Concepts, 7th and 8th editions, Silberschatz et al. Wiley.
- Modern Operating Systems, Andrew S. Tanenbaum, 3rd edition, 2009.

Outline

- I/O Hardware
- Application I/O Interface
- Kernel I/O Subsystem
- Transforming I/O Requests to Hardware Operations
- STREAMS
- Performance

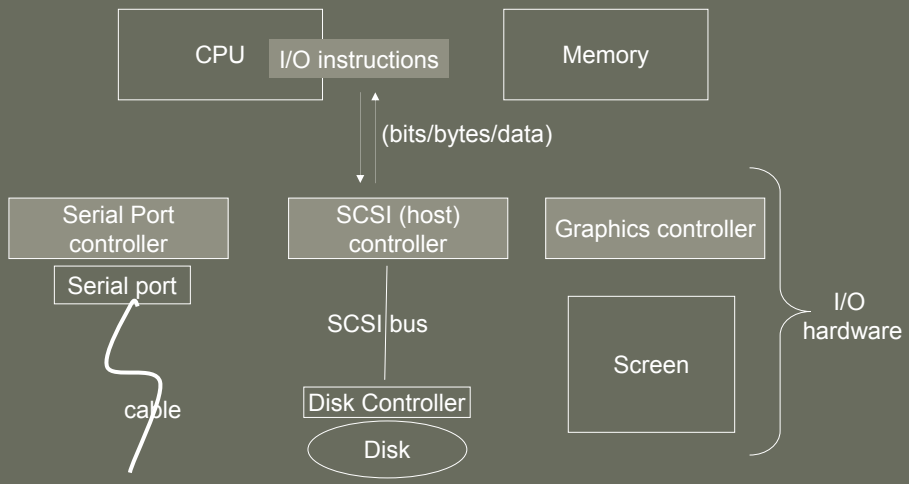
Objectives

- 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

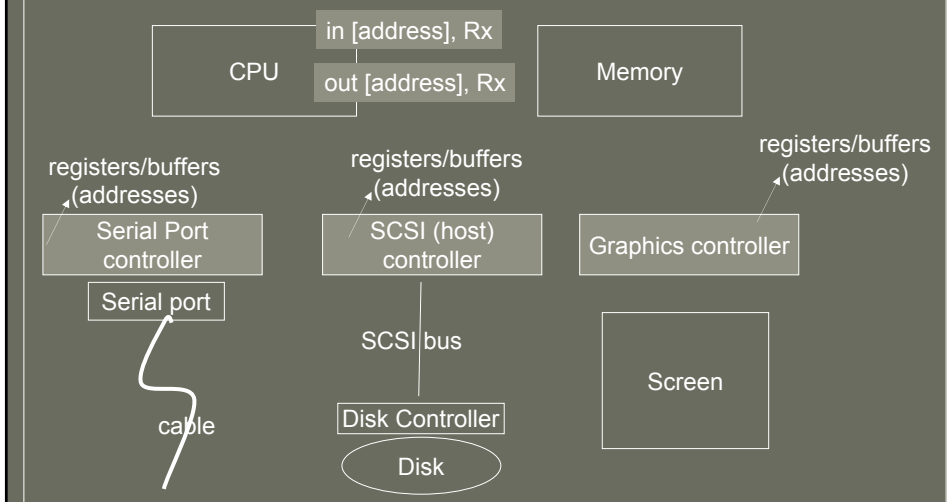
I/O Hardware

- Incredible variety of I/O devices
- Common concepts
  - Port (a connection point through computer accesses a device)
  - Bus (daisy chain or shared direct access)
    - Medium over which signals are sent/received
  - Controller (host adapter)
    - Chip that can be accessed by CPU and that controls the device/port/bus
      - PCI controller, Serial Port controller, keyboard controller, …
- I/O instructions control devices
- Devices have addresses, used by
  - Direct I/O instructions (in, out, etc)
  - Memory-mapped I/O (move, load, store)..

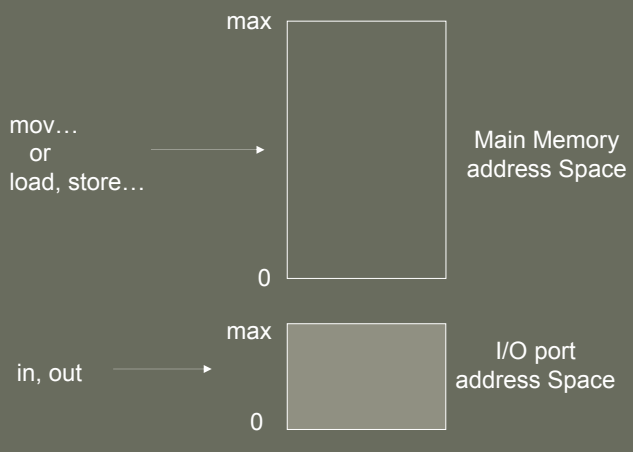
I/O hardware concepts



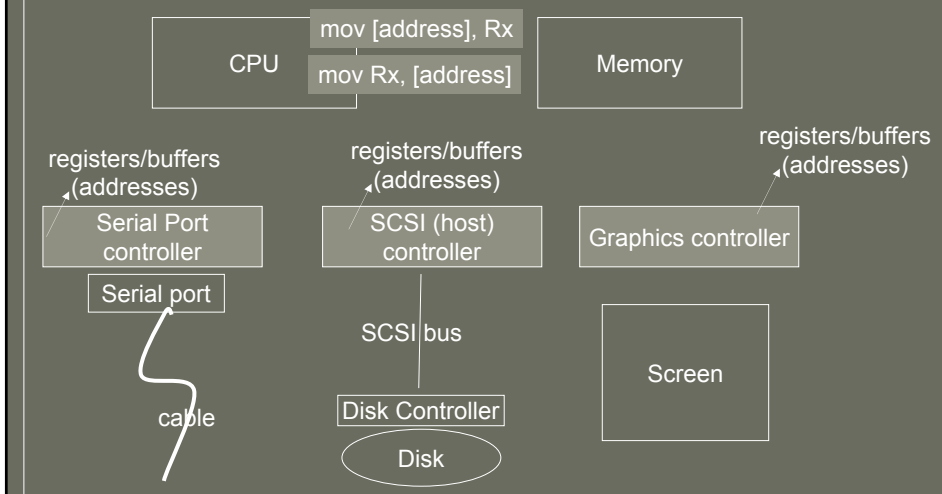
I/O hardware concepts: direct I/O



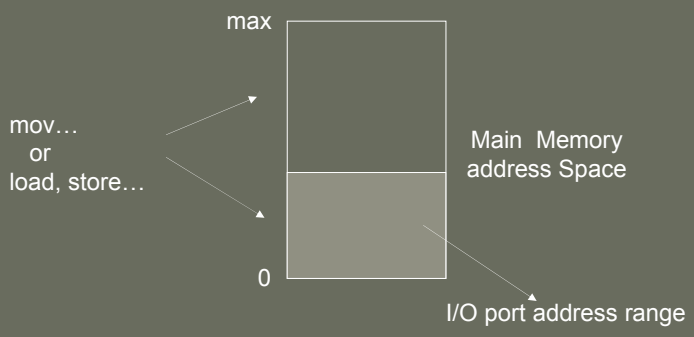
I/O hardware concepts: direct I/O



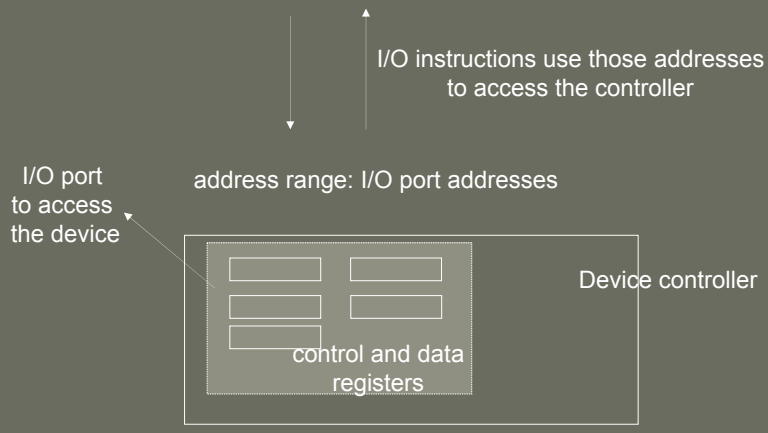
I/O hardware concepts: memory mapped I/O



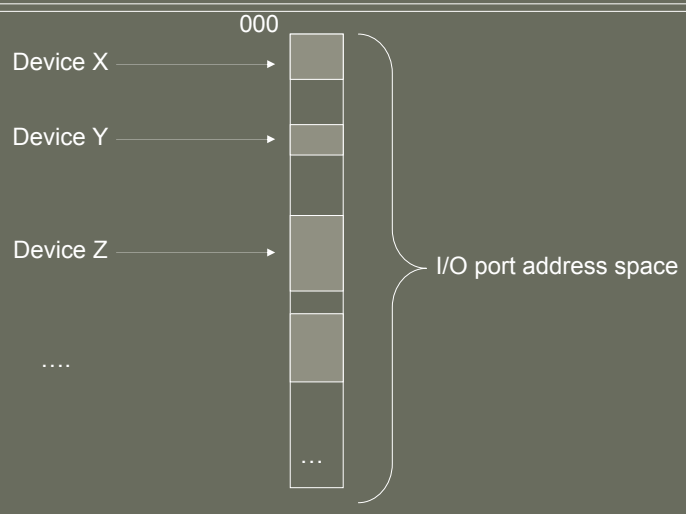
I/O hardware concepts: memory mapped I/O



I/O port concept



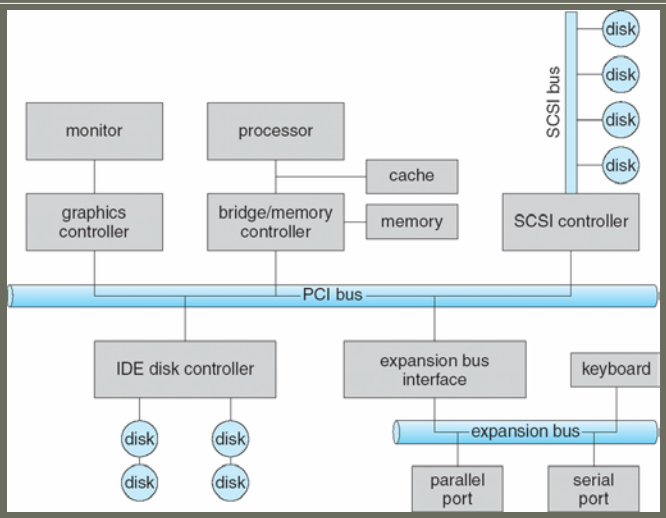
I/O port addresses



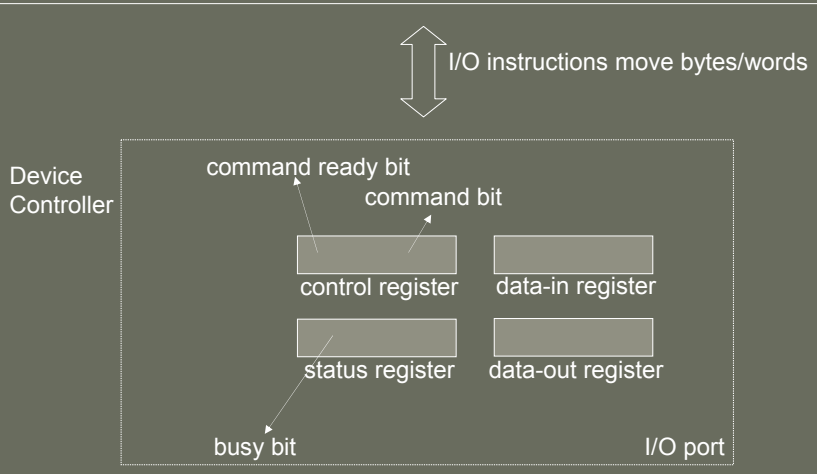
Device I/O Port Locations on PCs (partial)

I/O address range (hexadecimal)	device
000–00F	DMA controller
020–021	interrupt controller
040–043	timer
200–20F	game controller
2F8–2FF	serial port (secondary)
320–32F	hard-disk controller
378–37F	parallel port
3D0–3DF	graphics controller
3F0–3F7	diskette-drive controller
3F8–3FF	serial port (primary)

A Typical PC Bus Structure



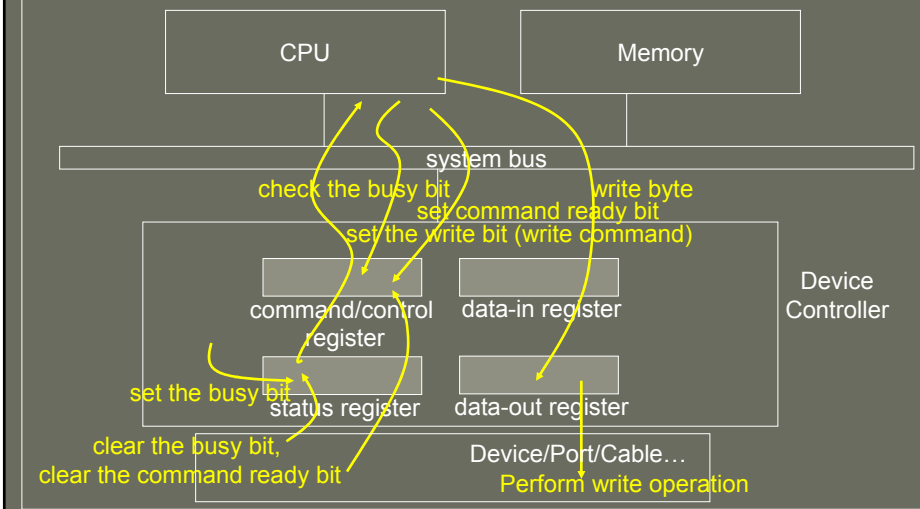
A typical I/O port



Interacting with the Device controller

- Host (CPU+Memory) and Device Controller interaction (data transfers and control) can be in one of 3 ways:
  - Polling
  - Interrupt driven I/O
  - Interrupt driven with help of DMA

Example: polling based writing



Example: polling based writing

CPU	Controller
1. read and check the busy bit 2. if busy go to 1. 3. set the write bit in command register 4. Write byte (word) into data out register 5. Set the command ready bit 6. Go to 1(maybe after doing something else)	• notices command ready bit set • set the busy bit • controller reads the command (it is write command), gets byte from data out register and write the byte out (this may take time) • clears the busy bit • clears the command ready bit • clears the error bit

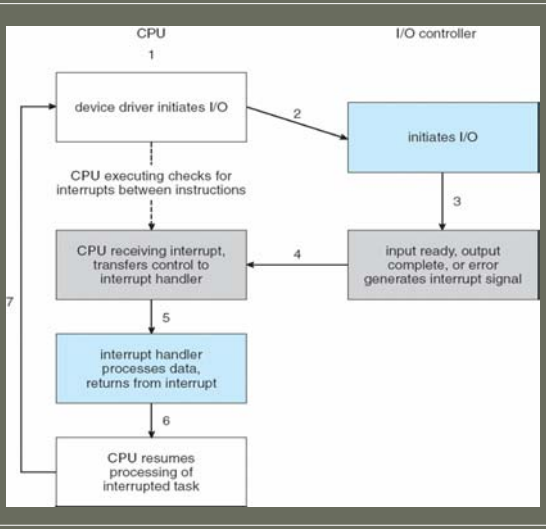
Transferring Data between host and device: Polling

- Determines state of device
  - command-ready
  - busy
  - Error
- Busy-wait cycle to wait for I/O from device

Transferring Data between host and 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 the correct handler
  - Based on priority
  - Some nonmaskable
- Interrupt mechanism also used for exceptions

Interrupt-Driven I/O Cycle



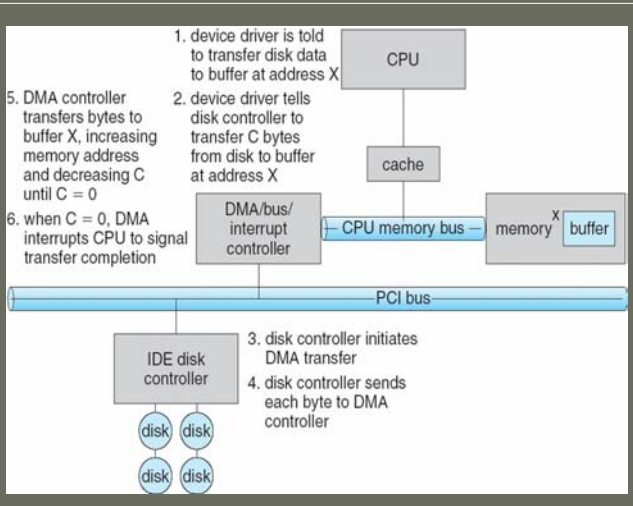
Intel Pentium Processor Event-Vector Table

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

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

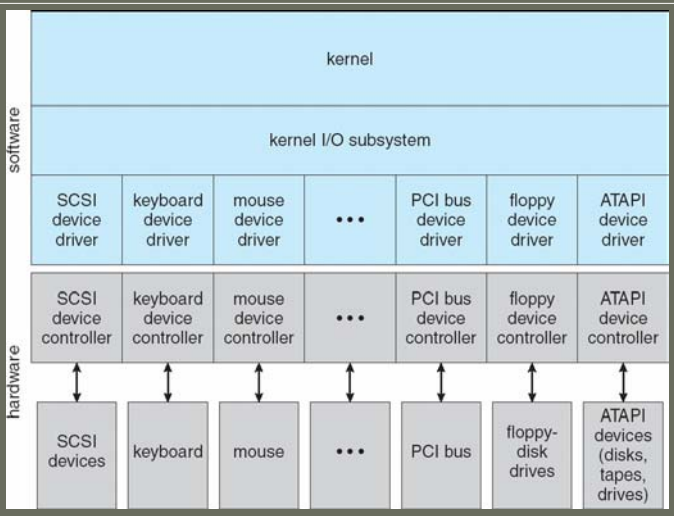
Six Step Process to Perform DMA Transfer



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



Characteristics of I/O Devices

aspect	variation	example
data-transfer mode	character block	terminal disk
access method	sequential random	modem CD-ROM
transfer schedule	synchronous asynchronous	tape keyboard
sharing	dedicated sharable	tape keyboard
device speed	latency seek time transfer rate delay between operations	
I/O direction	read only write only read–write	CD-ROM graphics controller disk

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)

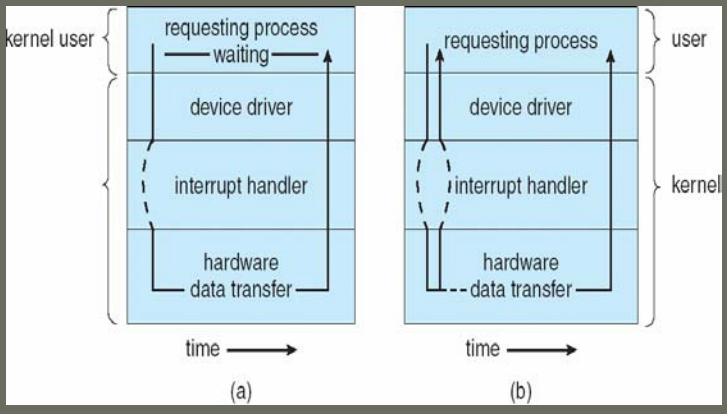
Clocks and Timers

- 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

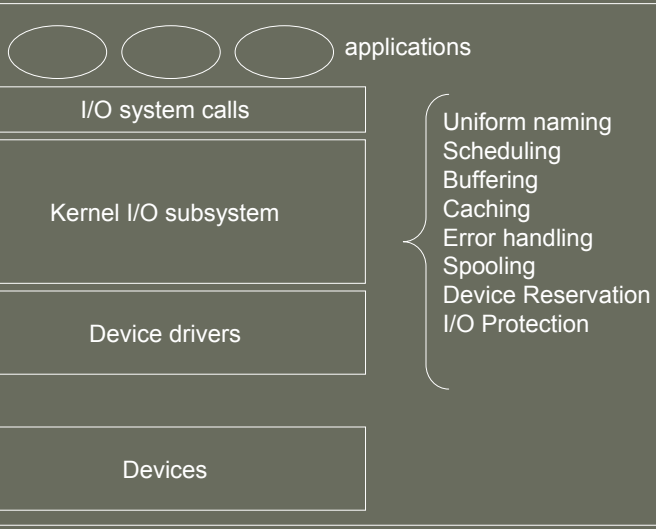
Blocking and Nonblocking I/O

- 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



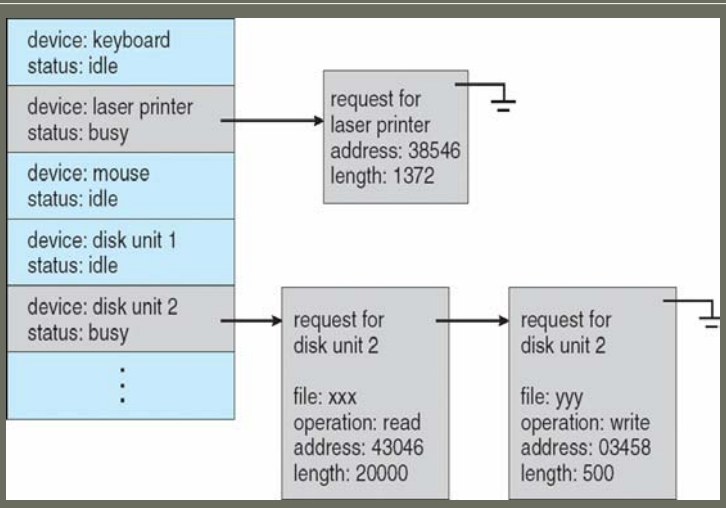
Kernel I/O Subsystem



Kernel I/O Subsystem

- Scheduling
  - Some I/O request ordering via per-device queue
  - Some OSs try fairness

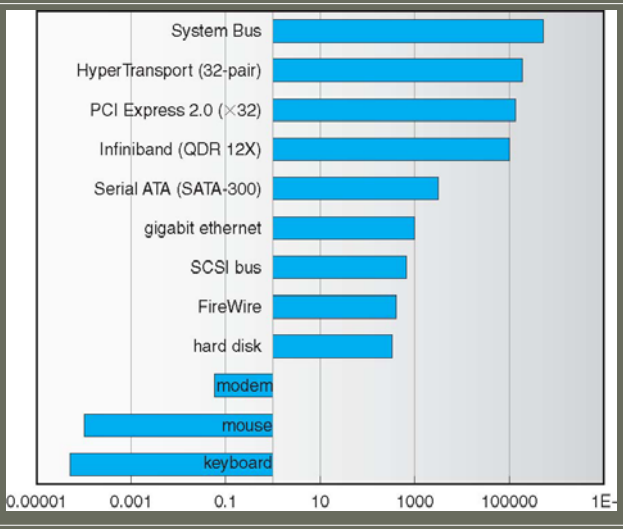
Device-status Table



Kernel I/O Subsystem

- 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”

Sun Enterprise 6000 Device-Transfer Rates



Kernel I/O Subsystem

- Caching - fast memory holding copy of data
  - Always just a copy
  - Key to performance

Kernel I/O Subsystem

- Spooling - hold output for a device
  - If device can serve only one request at a time
  - i.e., Printing

Kernel I/O Subsystem

- Device reservation - provides exclusive access to a device
  - System calls for allocation and deallocation
  - Watch out for deadlock

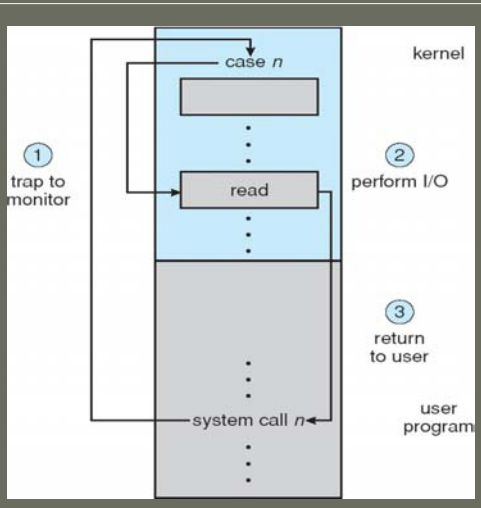
Error Handling

- 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

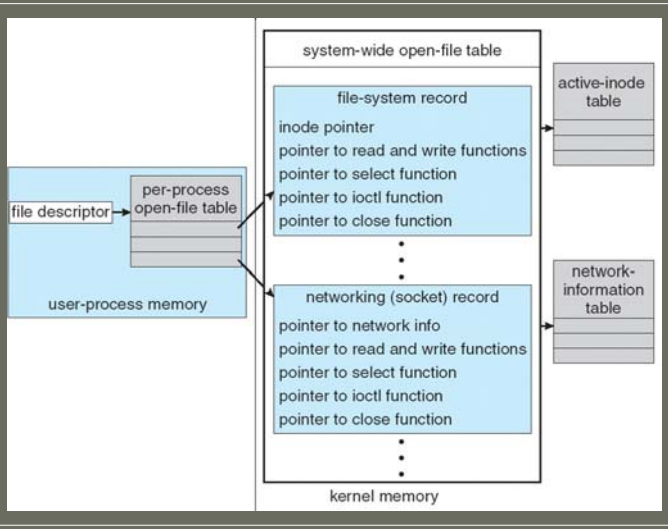
Use of a System Call to Perform I/O



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

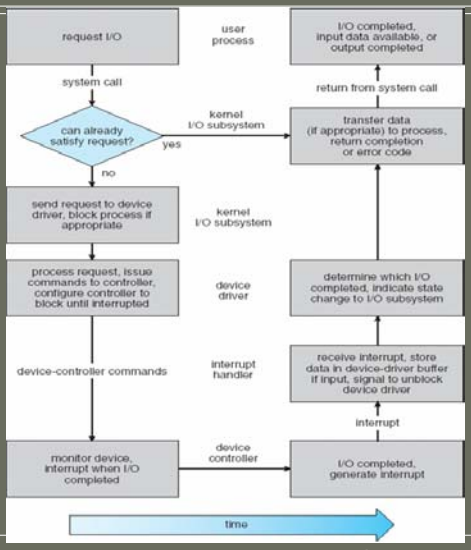
UNIX I/O Kernel Structure



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

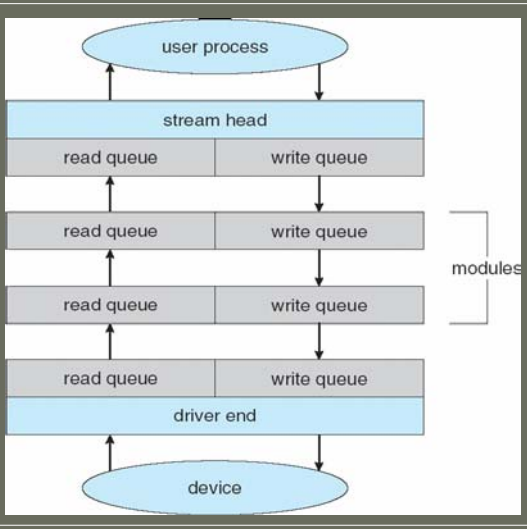
Life Cycle of An I/O Request



STEAMS

- 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

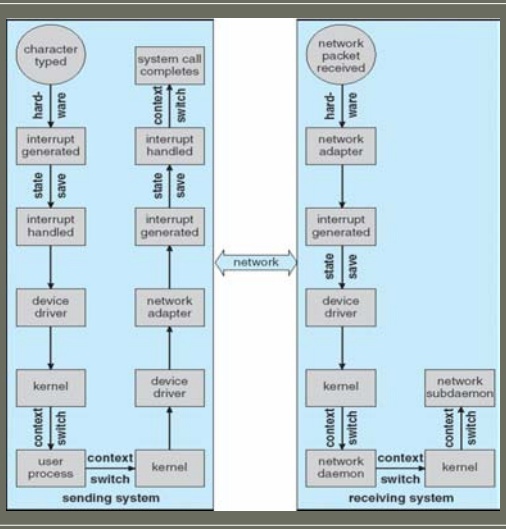
The STREAMS Structure



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

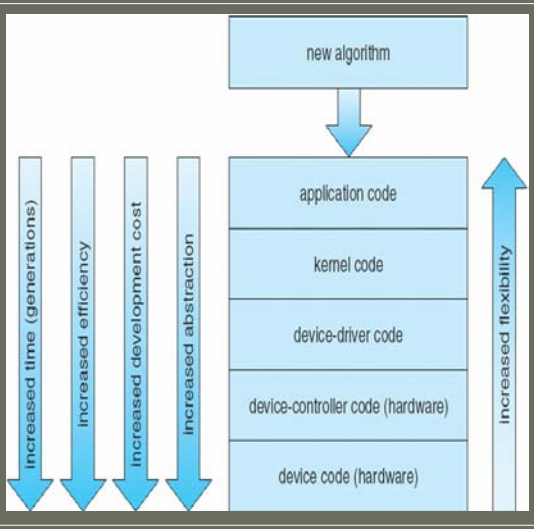
Intercomputer Communications



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



End of lecture