國立陽明交通大學
NATIONAL YANG MING CHIAO TUNG UNIVERSITY

Institute of Artificial Intelligence Innovation
Department of Computer Science

Operating System

Lecture 12: I/O Systems

Shuo-Han Chen 陳碩漢
shch@nycu.edu.tw

Wed. 10:10 - 12:00 EC115 +
Fri. 11:10 – 12:00 Online

Course Schedule

W	Date	Lecture	Online	Homework
1	Sept. 4	Lec00: Couse Overview & Historical Prospective		
2	Sept. 11	Lec01: Introduction	V	
3	Sept. 18	Lec02: OS Structure	V	HW01 Due 10/5
4	Sept. 25	Lec03: Processes Concept	X	
5	Oct. 2	Typhoon – No class	V	
6	Oct. 9	Lec07: Memory Management	V	
7	Oct. 16	Lec08: Virtual Memory Management	V	HW02 Due 11/2
8	Oct. 23	Lec04: Multithreaded Programming	V	
9	Oct. 30	Midterm Exam		
10	Nov. 6	Lec05: Process Scheduling	V	Let's take a breath
11	Nov. 13	Lec06: Process Synchronization & Deadlocks	X	HW03
12	Nov. 20	School Event – No class	V	
13	Nov. 27	Lec09: File System Interface	V	
14	Dec. 4	Lec10: File System Implementation	V	HW04
15	Dec. 11	Lec11: Mass Storage System & Lec12: IO Systems	V	
16	Dec. 18	School Final Exam		

Outline

- Overview
- I/O Hardware
- I/O Methods
- Kernel I/O Subsystem
- Performance
- Application Interface

Overview

- The two main jobs of a computer
  - I/O and Computation
- I/O devices: tape, HD, mouse, joystick, network card, screen, flash disks, etc
- I/O subsystem: the methods to control all I/O devices
- Two conflicting trends
  - Standardization of HW/SW interfaces
  - Board variety of I/O devices

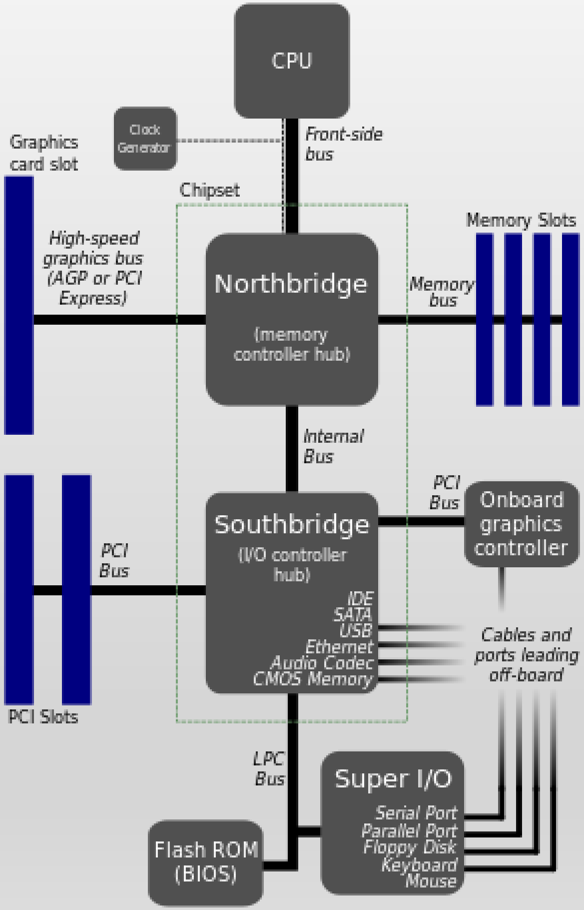
Overview

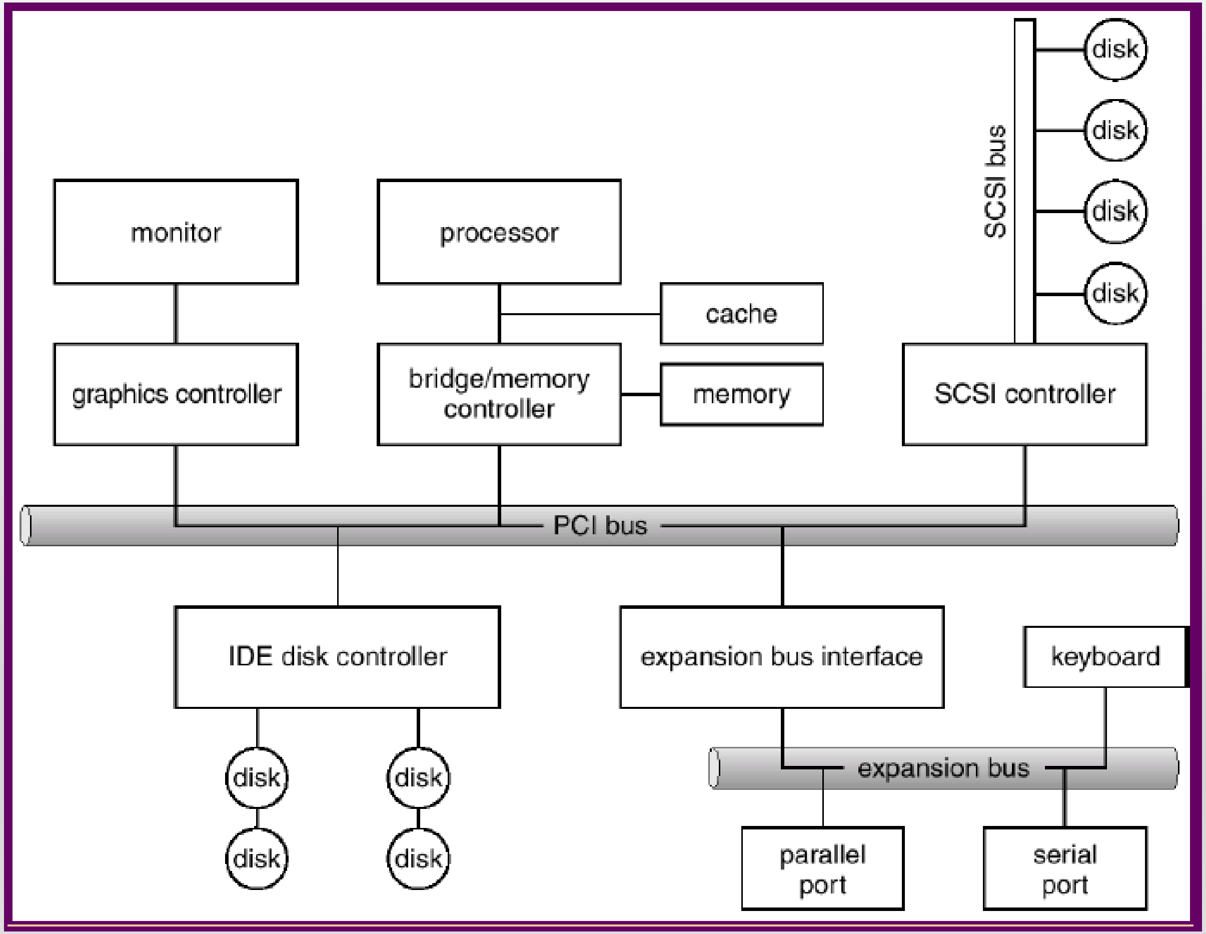
- Device drivers: a uniform device-access interface to the I/O subsystem
  - Similar to system calls between apps and OS
- Device categories
  - Storage devices: disks, tapes
  - Transmission devices: network cards, modems
  - Human-interface devices: keyboard, screen, mouse
  - Specialized devices: joystick, touchpad

I/O Hardware

- Port: A connection point between I/O devices and the host
  - E.g.: USB ports
- Bus: A set of wires and a well-defined protocol that specifies messages sent over the wires
  - E.g.: PCI bus
- Controller: A collection of electronics that can operate a port, a bus, or a device
  - A controller could have its own processor, memory, etc. (E.g.: SCSI controller)

Typical PC Bus Structure





Basic I/O Method (Port-mapped I/O)

- Each I/O port (device) is identified by a unique port address
- Each I/O port consists of four registers (1~4Bytes)
    - Data-in register: read by the host to get input
    - Data-out register: written by the host to send output
    - Status register: read by the host to check I/O status
    - Control register: written by the host to control the device
- Program interact with an I/O port through special I/O instructions (different from mem. access)
    - X86: IN, OUT

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)

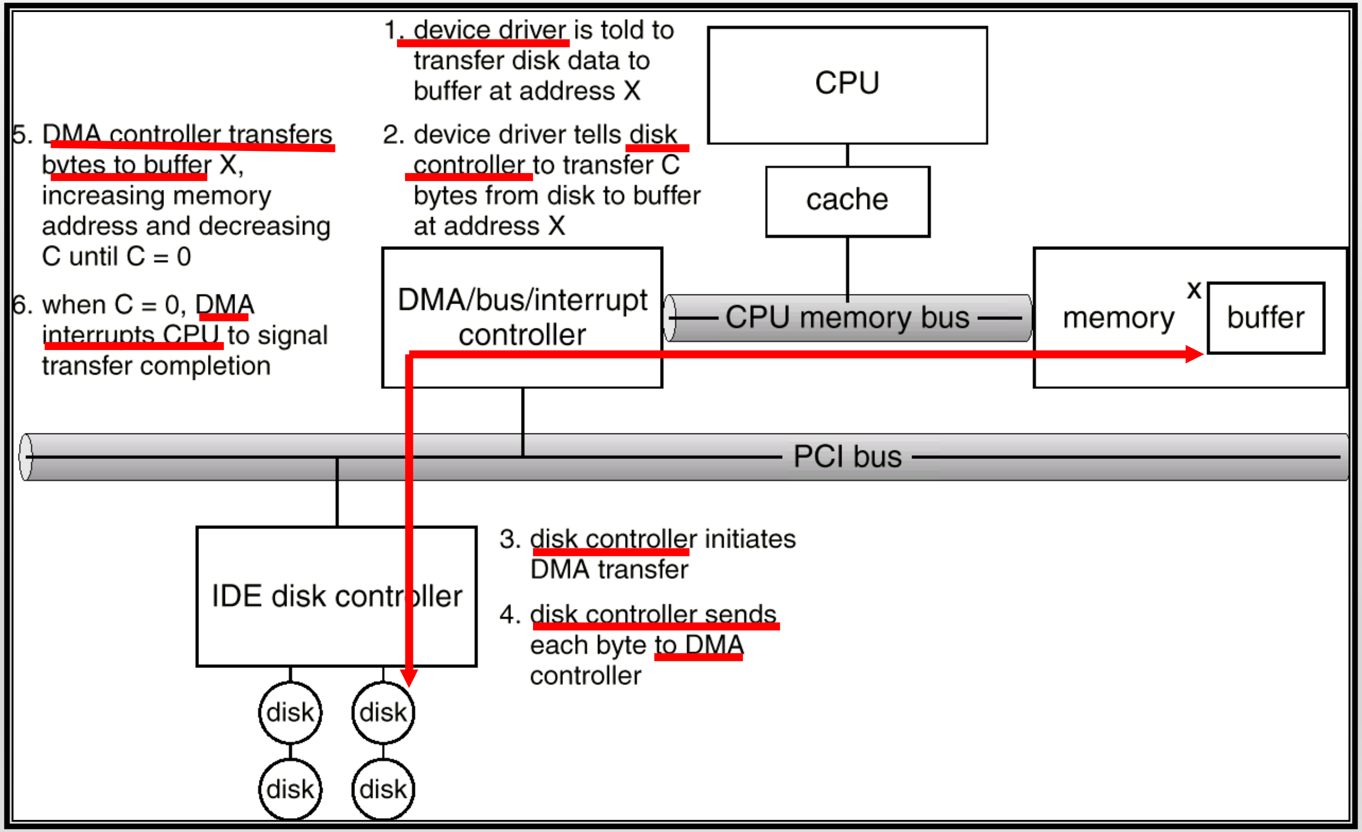
I/O Methods Categorization

- Depending on how to address a device:
  - Port-mapped I/O
    - Use different address space from memory
    - Access by special I/O instruction (e.g. IN, OUT)
  - Memory-mapped I/O
    - Reserve specific memory space for device
    - Access by standard data-transfer instruction (e.g. MOV)
    - Good: More efficient for large memory I/O (e.g. graphic card)
    - Bad: Vulnerable to accidental modification, error

I/O Methods Categorization

- Depending on how to interact with a device:
  - Poll (busy-waiting): processor periodically check status register of a device
  - Interrupt: device notify processor of its completion
- Depending on who to control the transfer:
  - Programmed I/O: transfer controlled by CPU
  - Direct memory access (DMA) I/O: controlled by DMA controller (a special purpose controller)
    - Design for large data transfer
    - Commonly used with memory-mapped I/O and interrupt

Six-Step Process to Perform DMA (Direct Memory Access)



Review Slides (I)

- Definition of I/O port? Bus? Controller?
- I/O device and CPU communication?
  - Port-mapped vs. Memory-mapped
  - Poll vs. Interrupt
  - Programmed I/O vs. DMA
- Steps to handle an interrupt I/O and DMA request?

Kernel I/O Subsystem

I/O Subsystem

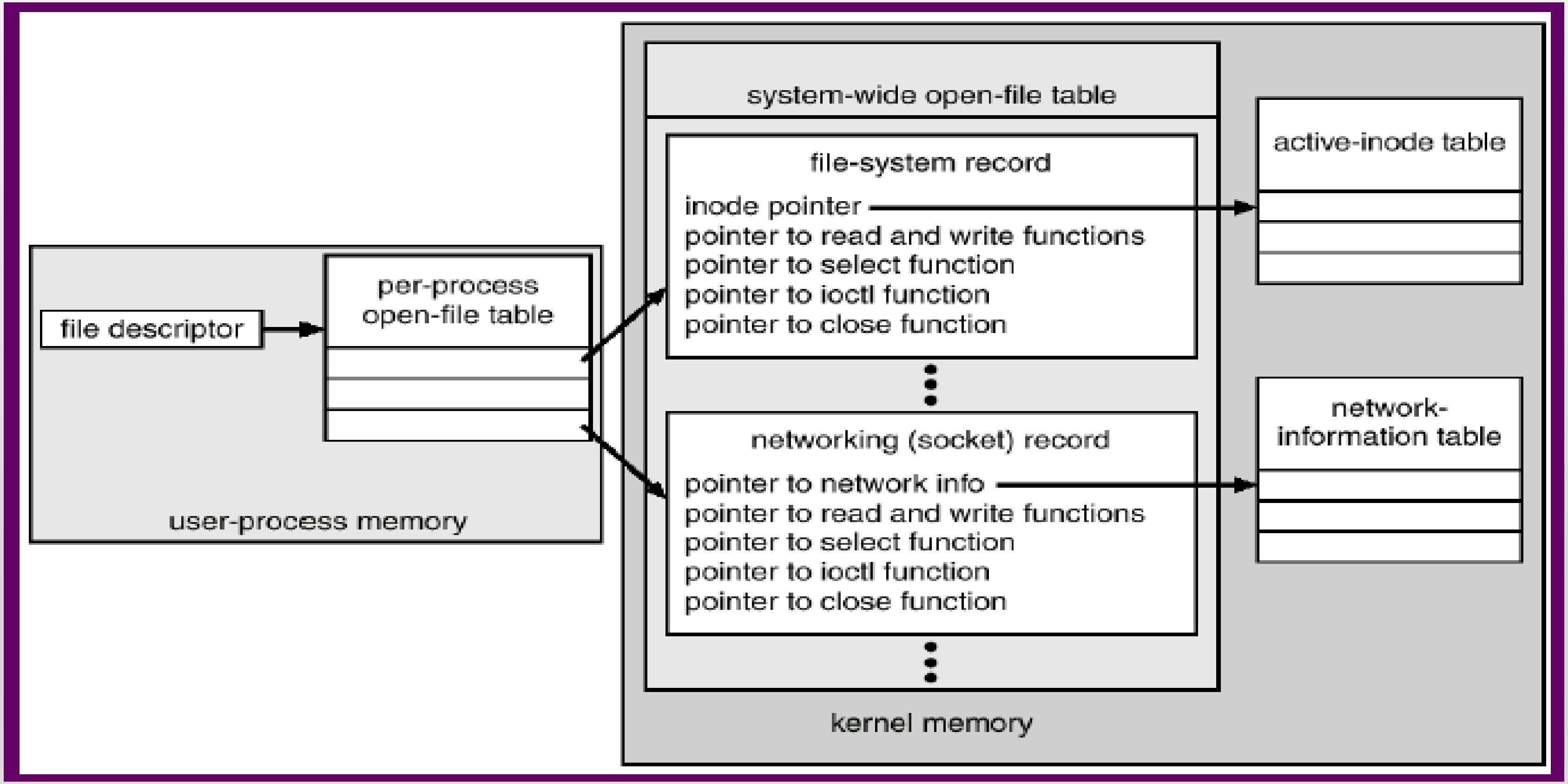
- I/O Scheduling - improve system performance by ordering the jobs in I/O queue
  - e.g. disk I/O order scheduling
- Buffering - store data in memory while transferring between I/O devices
  - Speed mismatch between devices
  - Devices with different data-transfer sizes
  - Support copy semantics

I/O Subsystem

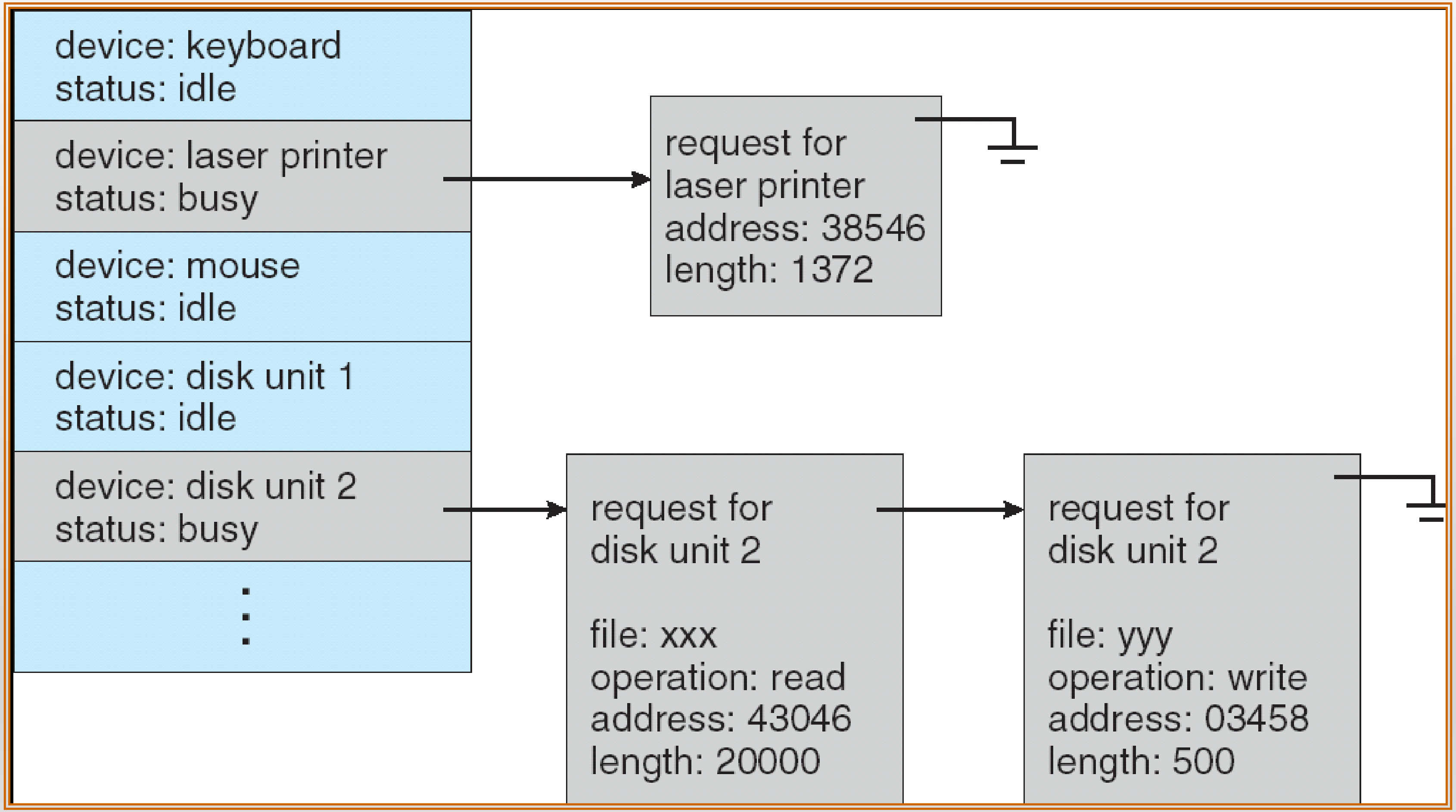
- Caching - fast memory that holds copies of data
  - Always just a copy
  - Key to performance
- Spooling - holds output for a device
  - e.g. printing (cannot accept interleaved files)
- Error handling - when I/O error happens
  - e.g. SCSI devices returns error information
- I/O protection
  - Privileged instructions

UNIX I/O Kernel Data Structure

- Linux treats all I/O devices like a file



Device-status Table



Blocking and Nonblocking I/O

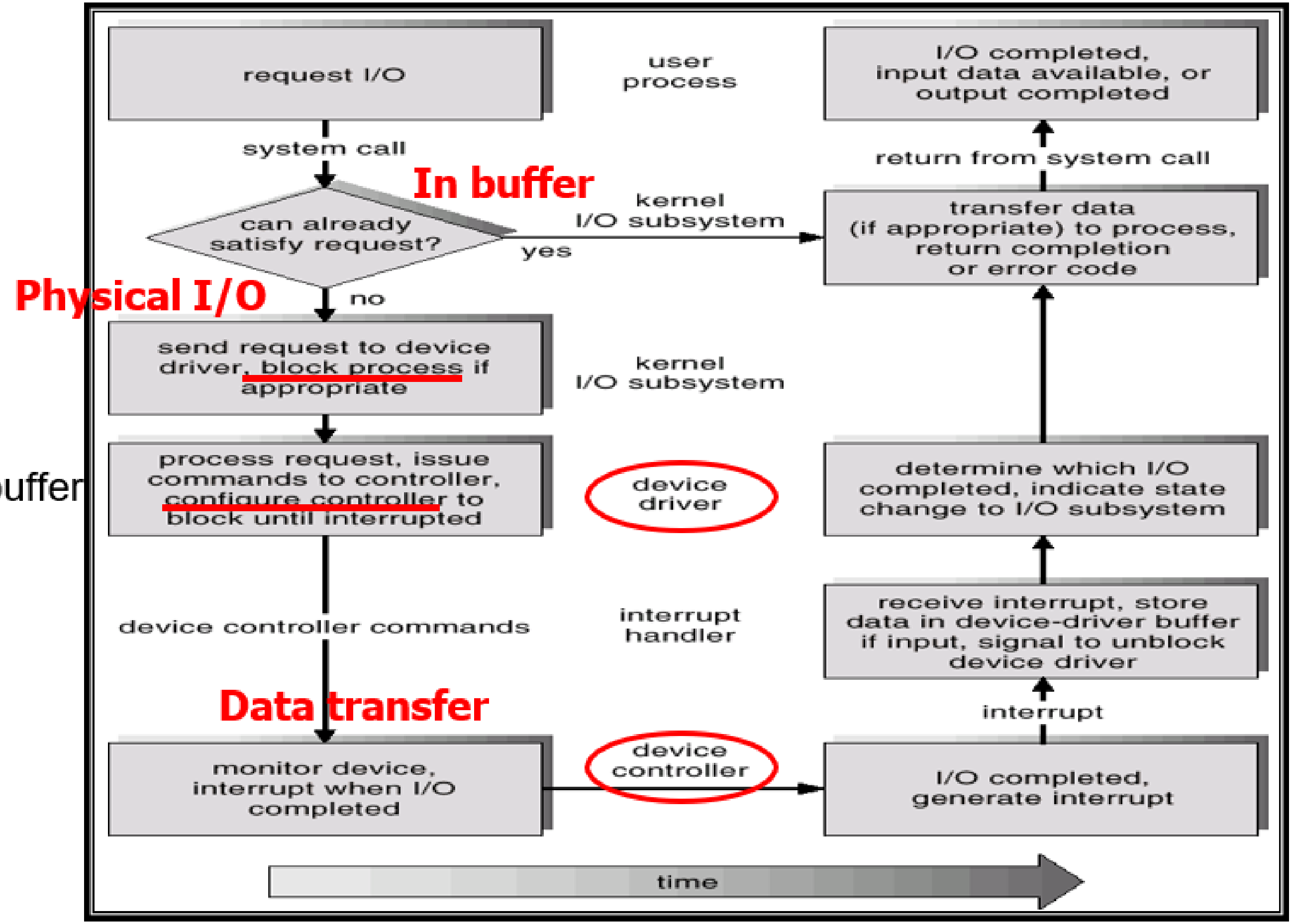
- Blocking - process suspended until I/O completed
  - Easy to use and understand
  - Insufficient for some needs
  - Use for synchronous communication & I/O
- Nonblocking
  - Implemented via multi-threading
  - Returns quickly with count of bytes read or written
  - Use for asynchronous communication & I/O

Life Cycle of An I/O Request

Check buffer cache

Move process from run queue to wait queue

Allocate kernel buffer
Schedule I/O



Performance

- I/O is a major factor in system performance
  - It places heavy demands on the CPU to execute device driver code
  - The resulting context switches stress the CPU and its hardware caches
  - I/O loads down the memory bus during data copy between controllers and physical memory, …
  - Interrupt handling is a relatively expensive task
    - Busy-waiting could be more efficient than interrupt- driven if I/O time is small

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

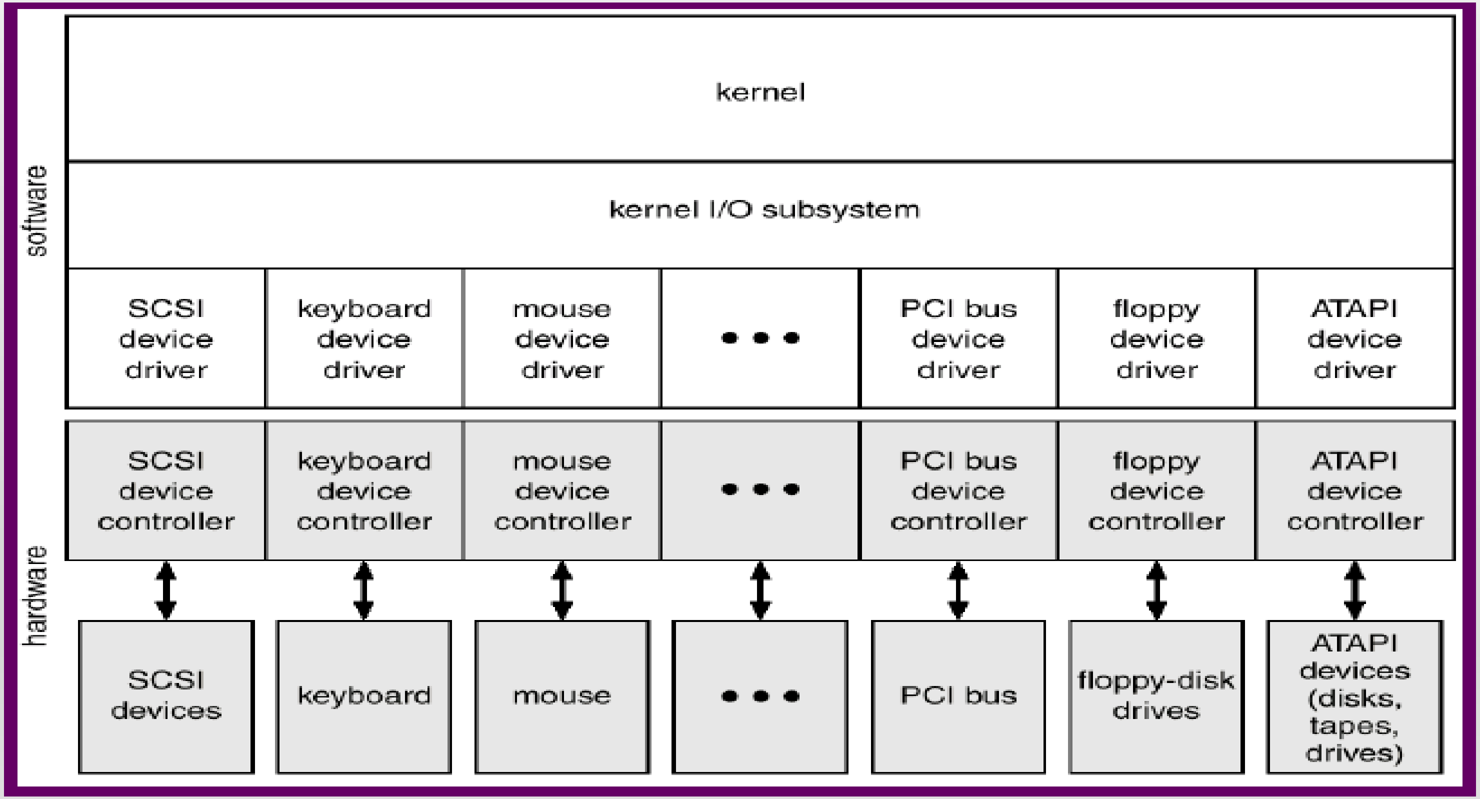
Review Slides (II)

- What are the key I/O services
  - Scheduling
  - Cache
  - Buffering
  - Spooling
  - Error handling
  - I/ protection
- How to improve system performance?

Application I/O Interface

A Kernel I/O Structure

- Device drivers: a uniform device-access interface to the I/O subsystem; hide the differences among device controllers from the I/O sub-system of OS



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 readĐwrite	CD-ROM graphics controller disk

I/O Device Class

- Device class is fairly standard across different OS
  - Block I/O
  - Char-stream I/O
  - Memory-mapped file access
  - Network sockets
  - Clock & timer interfaces
- Back-door interfaces (e.g. ioctl())
  - Enable an application to access any functionality implemented by a device driver without the need to invent a new system call

Block & Char Devices

- Block devices: disk drives
    - system calls: read( ), write( ), seek( )
    - Memory-mapped file can be layered on top
- Char-stream devices: mouse, keyboard, serial ports
    - system calls: get( ), put( )
    - Libraries layered on top allow line editing

Network Devices

- Varying enough from block and character to have own interface
    - System call: send(), recv(), select()
    - select() returns which socket is waiting to send or receive, eliminates the need of busy waiting
- Many other approaches
    - pipes, FIFOS, STREAMS, message queues

Reading Material & HW

- 13.1 - 13.6
- Problem Set
    - 13.2
    - 13.5
    - 13.6
    - 13.8

Interrupt Vector Table

- Intel Pentium Processor:

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
19Ð31	(Intel reserved, do not use)
32Ð255	maskable interrupts

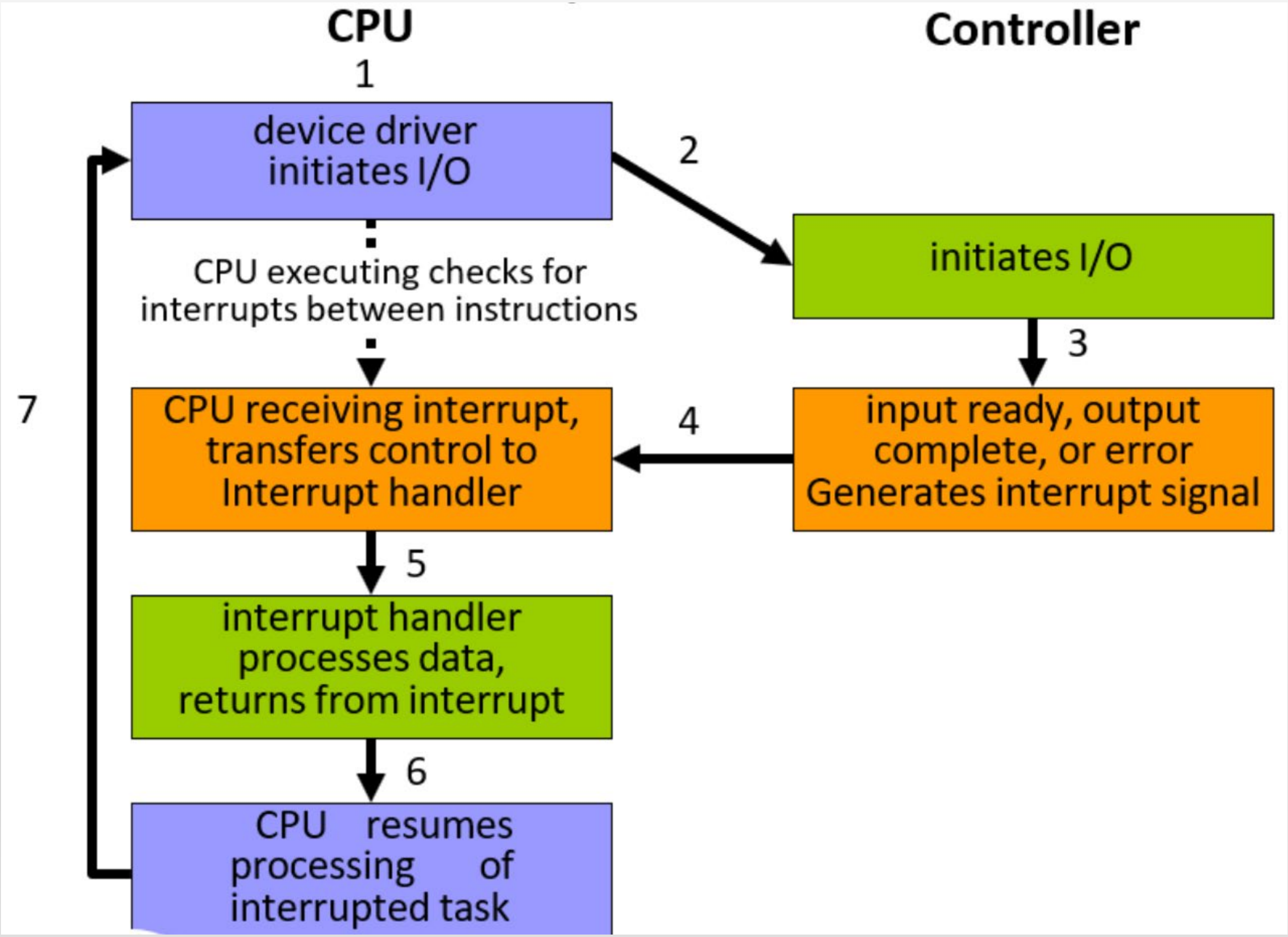
CPU and device Interrupt handshake

1. Device asserts interrupt request (IRQ)
2. CPU checks the interrupt request line at the beginning of each instruction cycle
3. Save the status and address of interrupted process
4. CPU acknowledges the interrupt and search the interrupt vector table for interrupt handler routines
5. CPU fetches the next instruction from the interrupt handler routine
6. Restore interrupted process after executing interrupt handler routine

Interrupt Prioritization

- Maskable interrupt: interrupt with priority lower than current priority is not recognized until pending interrupt is complete
- Non-maskable interrupt (NMI): highest- priority, never masked
  - Often used for power-down, memory error

Interrupt-Driven I/O



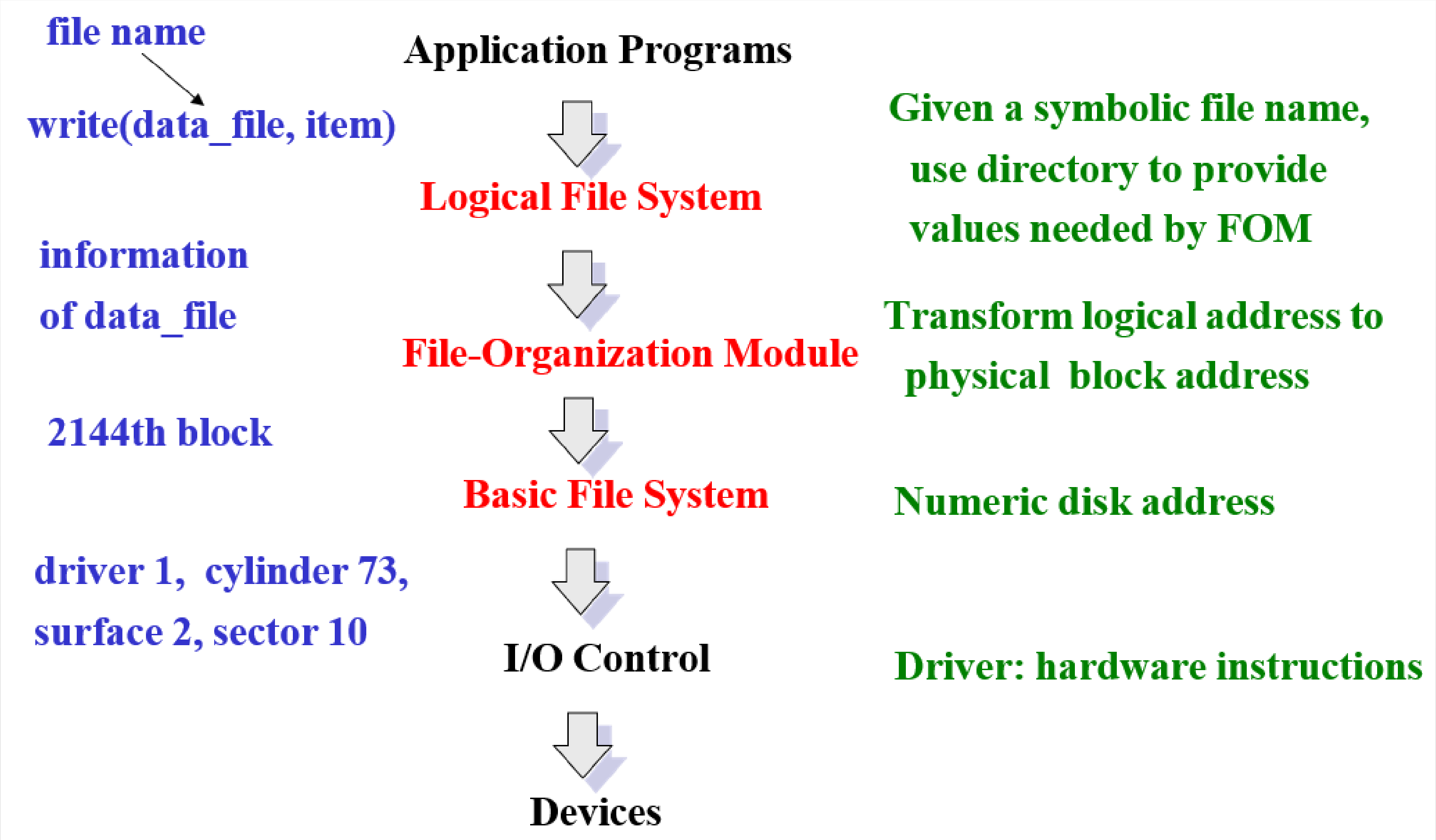
Summary of Services in I/O Subsystem

- The management of the name space for files and devices
- Access control to files and devices
- Operation control
- File system space allocation
- Disk allocation
- Buffering, caching, and spooling
- I/O scheduling
- Device status monitoring, error handling, and failure recovery
- Device driver configuration and initialization

I/O Requests to Hardware Operations

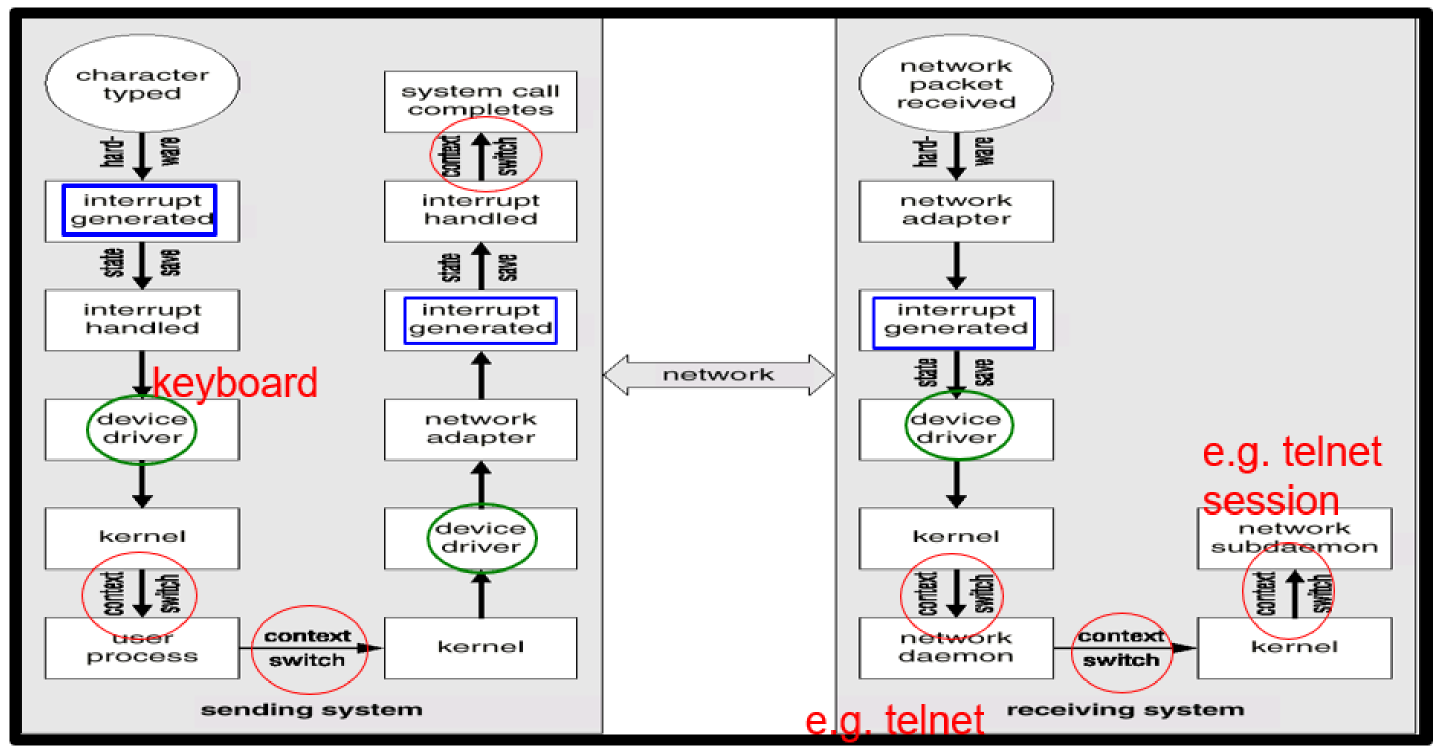
- Consider reading a file from disk for a process
  - Determine device holding file
  - Translate name to device representation
  - Physically read data from disk into buffer
  - Make data available to requesting process
  - Return control to process

Layered File System revisited



Intercomputer Communications

- Network traffic could cause high context switch rate
- Interrupt generated during keyboard & network I/O
- Context switch occurs between prog. & kernel (drivers)



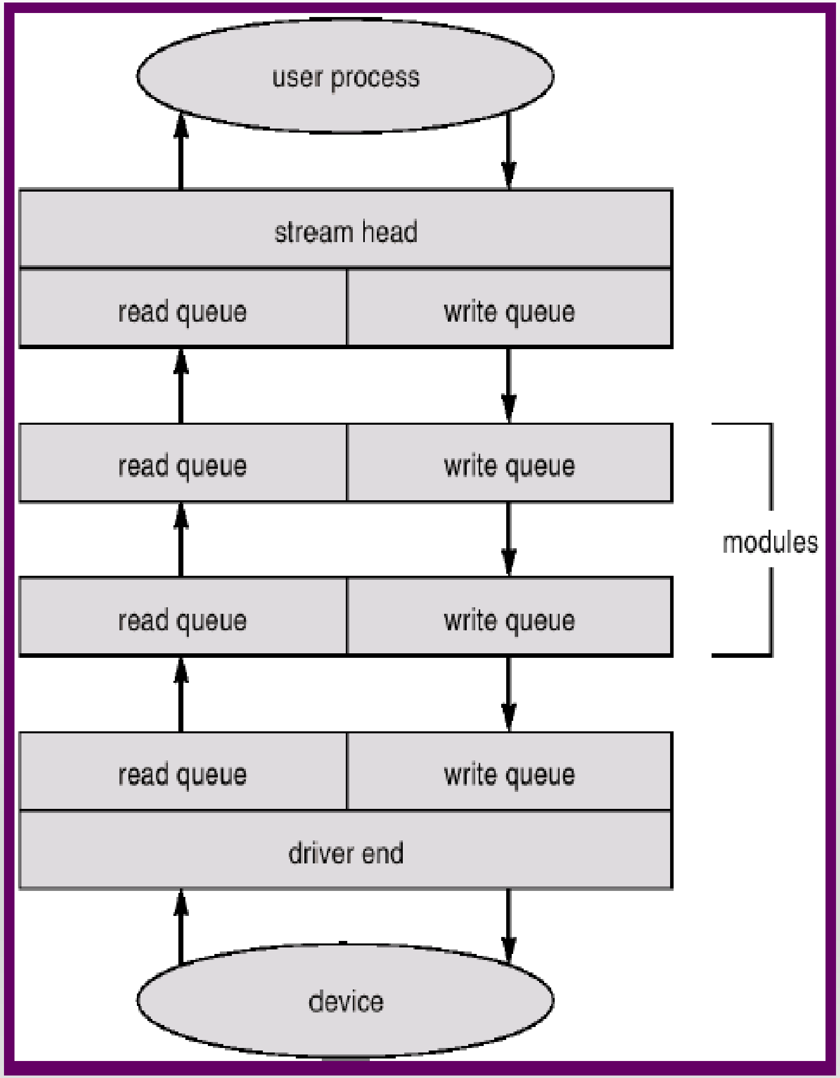
STREAMS

- A full-duplex communication channel between a user-level process and a device

- STREAM provides a framework for a modular and incremental approach to writing device drivers and network protocols

The STREAM Structure

- A STREAM consists of
  - STREAM head interfaces with user process
  - Driver end interfaces with the device
  - zero or more STREAM modules between them
- Each module contains a read and a write queue
- Message passing is used to communicate between queues



Q & A

Thank you for your attention