



# Operating System Design

Dr. Jerry Shiao, Silicon Valley University

# I/O Systems

- Role of Operating System is to Manage and Control I/O Operations and I/O Devices.
  - I/O Devices Vary in Functionality and Speed (mouse, hard drive).
  - Techniques for Performing I/O (Programmed I/O, DMA).
  - Interrupts and Programmable Interrupt Controller (PIC).
- Kernel I/O Services Provided by Operating System
  - Application I/O Interface: Abstraction at I/O Subsystem.
  - Improving efficiency: I/O Scheduling Disk Device.
  - Transforming I/O Requests to Hardware Operations
- STREAMS I/O Structure
  - Full-Duplex data path between Device Driver and User Process.
- Performance Aspects of I/O
  - Principles of Operating System Design that improves I/O Performance.

# I/O Systems

## ■ I/O Hardware

## ■ Incredible variety of I/O devices

- Storage Devices (Disk, Tapes, CDROM).
- Transmission Devices (Network Cards, Modems).
- Human-Interface Devices (Console, Keyboard, Mouse).

## ■ Common Concepts on how Software Control Devices

*External* □ Port: I/O Device cable connection point to Computer System.

- Serial Port, Parallel Port, USB Port.

*Internal.* □ BUS: Common set of wires connecting multiple I/O Devices.

- Specific protocol and specific set of messages.
- Vary in signaling methods, speed, throughput, and connection methods.
- Daisy Chain Bus: I/O Devices connected in serial on Bus.
- PCI Bus: Connects Processor – Memory Subsystem to fast I/O Devices.
  - Disk Controller, Graphics Controller.
- PCI Expansion Bus: Connects to slow I/O Devices.
  - Keyboard, Parallel Port, USB Port

# I/O Systems

- I/O Hardware

- I/O Devices: Common Concepts

- Controller: Single chip (serial-port controller) or circuit board (SCSI bus controller or host adapter) plugs into Computer.
  - Contains processor, microcode, memory for processing Bus Protocol Messages.
  - Built-In Controllers: Devices such as removable disks has attached circuit boards.

- How can the Processor give Commands and Data to a Controller to accomplish an I/O transfer?

- Controller has registers for data and control signals.
- Processor communicates with the Controller by reading and writing bit patterns in these registers.

# I/O Systems

## ■ I/O Hardware

## ■ I/O Devices: Common Concepts

- Controller: Single chip (serial-port controller) or circuit board (SCSI bus controller or host adapter) plugs into Computer.
  - Contains processor, microcode, memory for processing Bus Protocol Messages.
  - Built-In Controllers: Devices such as removable disks has attached circuit boards.

## ■ Processor Communicates to Controller:

- Port-Mapped I/O Instructions triggers bus lines to select I/O Device and read/write Device Registers.
- Memory-Mapped I/O maps Device Registers into address space of process.
- Some I/O Device uses both Direct I/O and Memory-Mapped I/O.
  - Graphics Controller has Direct I/O Ports for control operations, and large Memory-Mapped region for screen contents.
- Direct Memory Access (DMA): I/O Subsystems to access Main Memory independent of the CPU.

2 types  
(use different instructions)

# I/O Systems

## ■ I/O Function

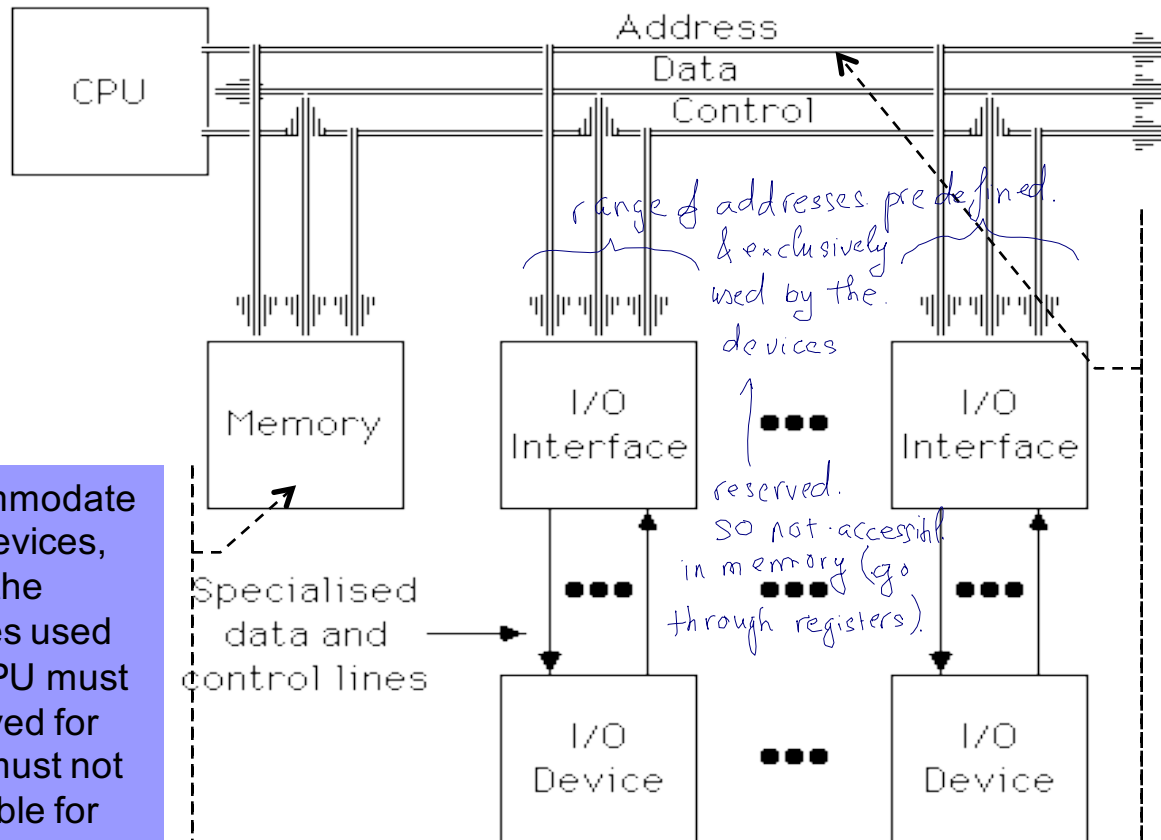
## ■ Techniques for Performing I/O

- ☐ Programmed I/O: Processor issues I/O command to I/O module. Process busy waits for the operation to complete. *for streaming data* *to trxf data to device controller* *send commands & wait for response → hold CPU*
- ☐ Interrupt-Driven I/O: Processor issues I/O command, the I/O instruction can be blocking or nonblocking. *send cmds & do sth else, do not hog up CPU*
- ☐ Direct Memory Access (DMA): DMA module controls exchange of data between main memory and I/O device controller, without the processor. *for trxf large amt of data.* *Using some CPU, allows the DMA to act on behalf of the CPU. No need the MMU to do data trxf w/o CPU usage*

	No Interrupts	Use of Interrupts
I/O-to-Memory Transfer through Processor	Programmed I/O	Interrupt-driven I/O
Direct I/O-to-Memory Transfer		Direct memory access (DMA)

# I/O Systems

- I/O Function
- Memory-Mapped I/O Structure



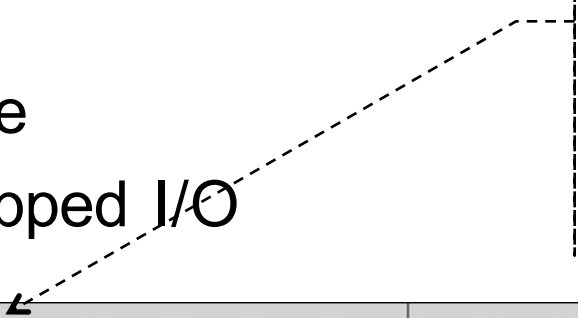
To accommodate the I/O devices, areas of the addresses used by the CPU must be reserved for I/O and must not be available for normal physical memory.

Memory-mapped I/O uses the same address bus to address both memory and I/O devices – the memory and registers of the I/O devices are mapped to (associated with) address values. So when an address is accessed by the CPU, it may refer to a portion of physical RAM, but it can also refer to memory of the I/O device. Each I/O device monitors the CPU's address bus and responds to any CPU access of an address assigned to that device, connecting the data bus to the desired device's hardware register..

# I/O Systems

- I/O Hardware
- Memory-Mapped I/O

Memory-mapped I/O uses same bus to address both memory and I/O devices. I/O devices monitor CPU's address bus and respond to any CPU access of their assigned address space, mapping the address to their hardware registers.



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)

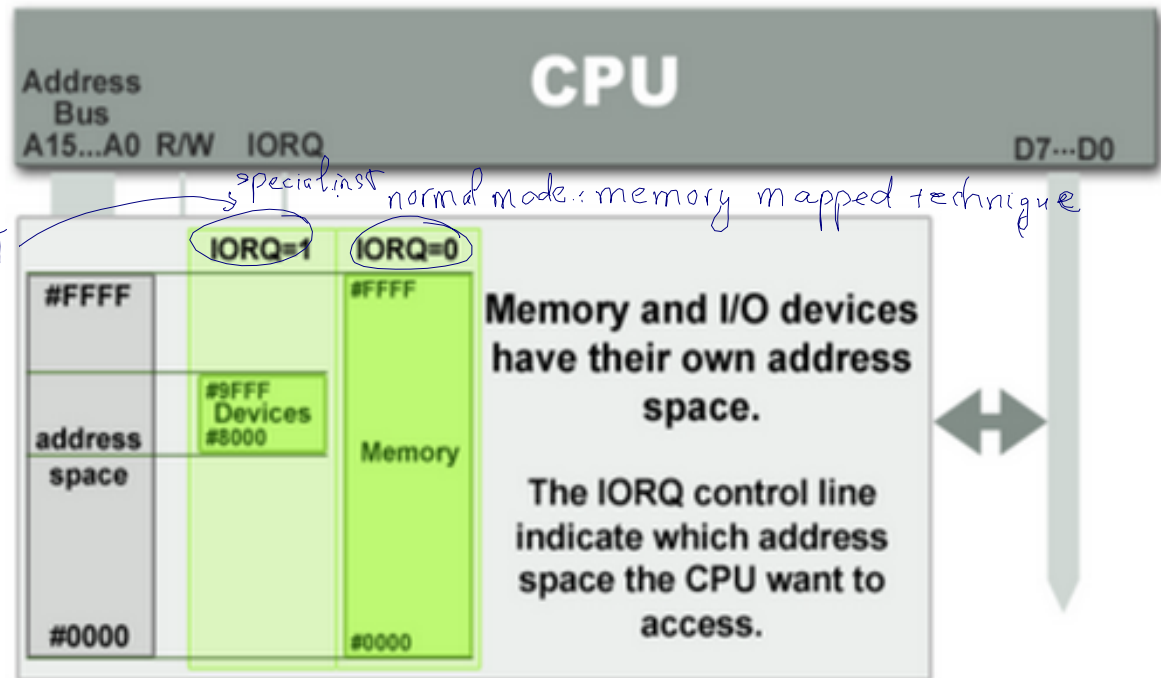


# I/O Systems

- I/O Function
- I/O Mapped I/O or Port Mapped I/O or Direct I/O
- I/O Devices mapped into separate address space.
  - Different set of signal lines for port versus memory access.
  - Read/Write require special port I/O instructions (IN and OUT Instructions).

use separate bus & separate addresses

=> use special inst: IN/OUT



# I/O Systems

- I/O Function

- I/O Mapped I/O or Port Mapped I/O or Direct I/O

- I/O Port Registers

- Data-In Register : Read by Host to get Input.
- Data-Out Register: Written by Host to send Output.
- Status Register: Read by Host to get controller state.
  - Current command completed.
  - Byte available to be read from Data-In Register.
  - Device error occurred.
- Control Register: Start command or change mode of an I/O Device.
  - Bit to select Full-Duplex or Half-Duplex.
  - Bit to enable Parity Checking.
  - Bit to select 7 or 8 Bit word Length.
  - Bit to select speed of the I/O Device.

# I/O Systems

- I/O Function
- Compare Memory-Mapped I/O and Port-Mapped I/O

## Memory-mapped IO

Same address bus to address  
memory and I/O devices

Access to the I/O devices using  
regular instructions

Most widely used I/O method

## Port-mapped IO

Different address spaces for memory  
and I/O devices

Uses a special class of CPU  
instructions to access I/O devices

x86 Intel microprocessors - IN and  
OUT instructions

# I/O Systems

## ■ I/O Hardware

## ■ Direct Memory Access (DMA)

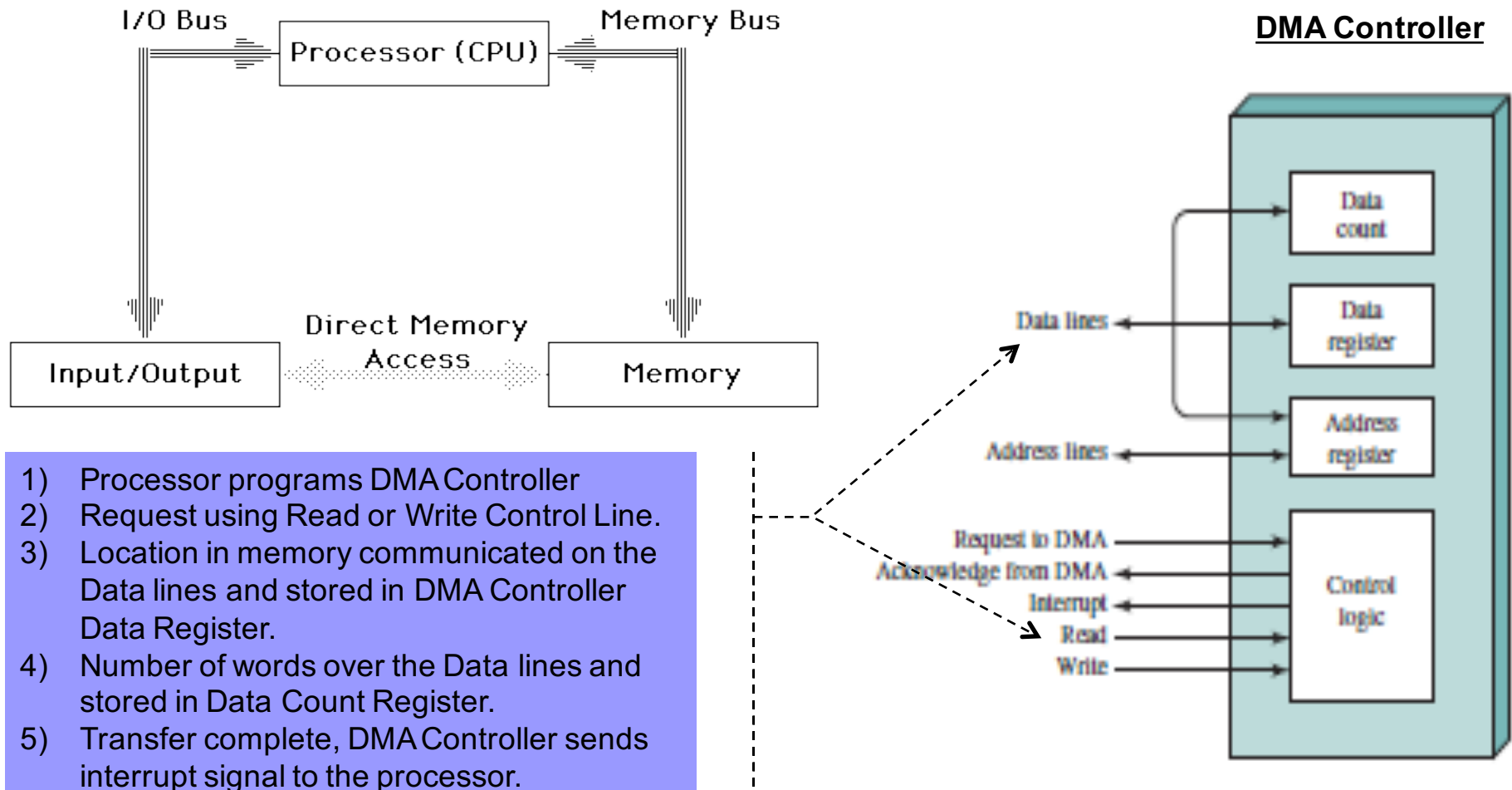
← main purpose: take over the CPU duty during data trxf

- Large data block transfers to/from I/O Device, wasteful for CPU to use controller registers (Programmed I/O).
- DMA Controller: Special-Purpose Processor operates the memory Bus directly.
- DMA Transfer: DMA Command Block with Bus addresses.
  - Pointer to source of transfer.
  - Pointer to destination of transfer.
  - Number of bytes to transfer.
- DMA Transfer: DMA-Request and DMA-Acknowledge Signals.
  - Device Controller places signal on DMA-Request wire.
  - DMA Controller seize memory BUS (CPU cannot access main memory).
  - Device Controller receives DMA-Acknowledge signal, starts sending data.
  - DMA Controller interrupts CPU when completed.

# I/O Systems

## ■ I/O Function

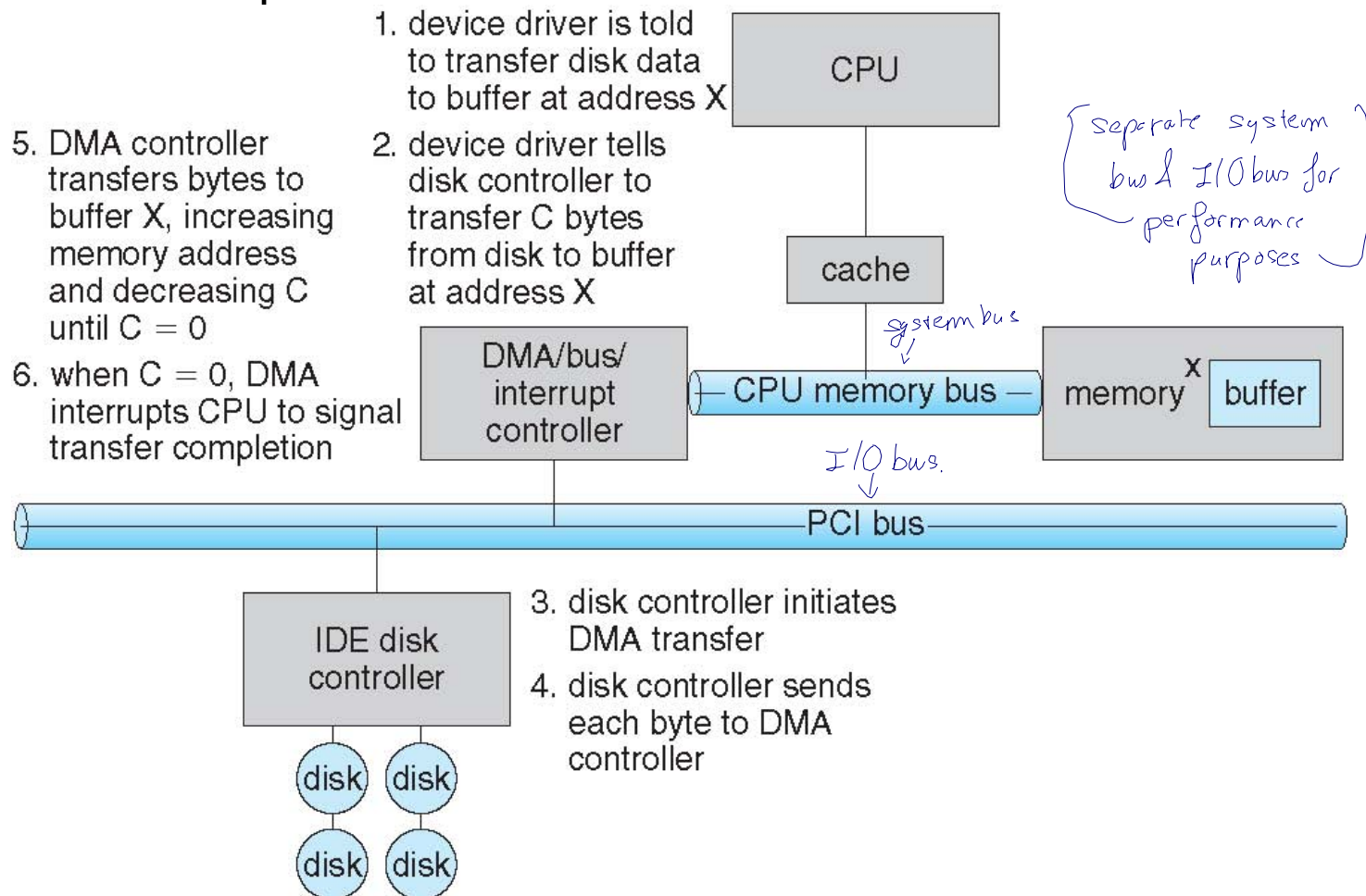
## ■ Direct Memory Access I/O Structure



# I/O Systems

## I/O Hardware

### DMA Transfer Steps:



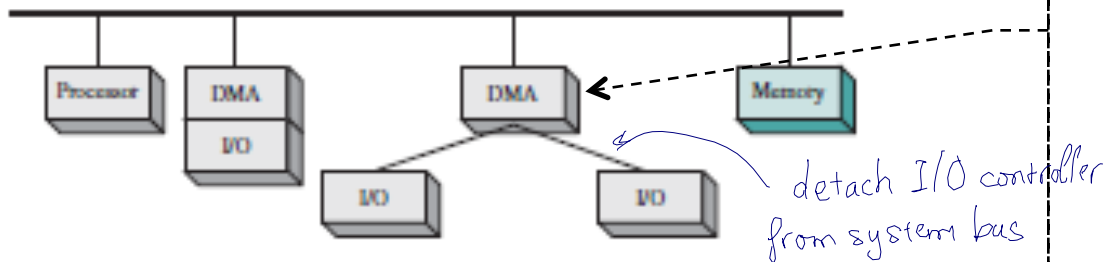
# I/O Systems

## I/O Hardware

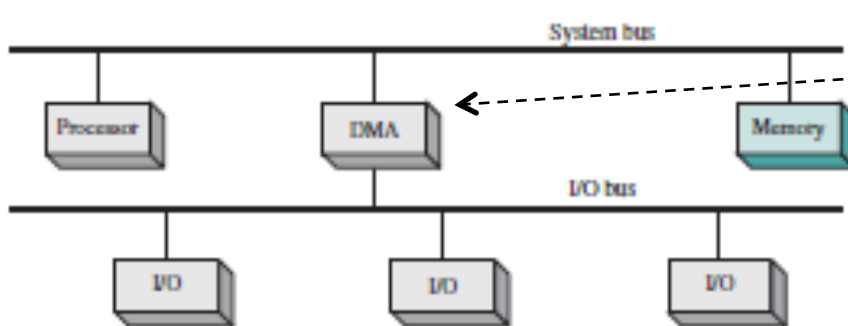
## DMA Configurations



(a) Single-bus, detached DMA



(b) Single-bus, integrated DMA-I/O



(c) I/O bus

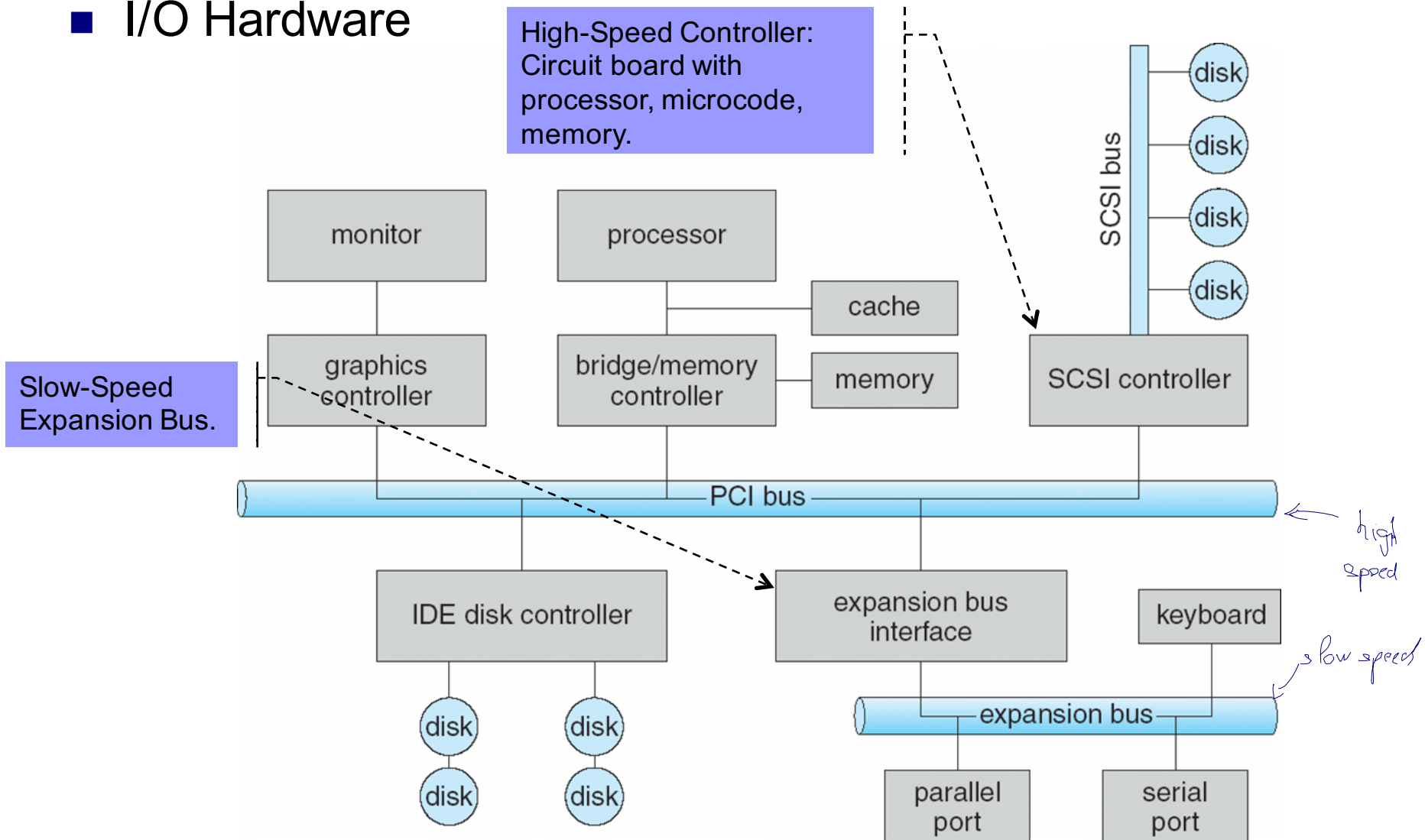
All modules share the System Bus. DMA module, acts as the "processor", uses Programmed I/O to exchange data between memory and I/O Module. Additional cycles on System Bus, transfer request to I/O Module followed by transfer from I/O Module and then to Memory.

DMA part of the I/O Module or separate module that controls I/O Modules. System Bus used by DMA Module to only exchange data with Memory and exchange control signals with the processor.

DMA Module connected to I/O Modules on separate I/O Bus. DMA Module has one interface to System Bus. System Bus used by DMA Module to only exchange data with Memory and exchange control signals with the processor.

# I/O Systems

## I/O Hardware





# I/O Systems

## ■ I/O Hardware *more of programmed I/O*

## ■ Polling: Handshaking Sequence to Communicate between Processor and Controller

- Host checks controller Status Register “Busy” Bit is clear.
  - Controller clears “Busy” Bit when its ready for next command.
- Host:
  - Sets “Write” Bit in Control Register.
  - Sets Data-Out Register with command byte.
  - Sets “Command-Ready” Bit in Control Register.
- Controller:
  - Detects “Command-Ready” Bit in Control Register.
  - Sets “Busy” Bit in Control Register.
  - Detects “Write” Bit in Control Register.
  - Reads Data-Out Register.
  - Clears “Command-Ready” Bit in Control Register.
  - Clears “Busy” Bit in Control Register.
  - Clears “Error” Bit in Control Register (Command Succeeded).

# I/O Systems

## ■ I/O Facility

## ■ What are the Operating System Objectives for I/O Facility?

## ■ Must be Efficient:

- I/O Devices slow compared to Memory and Processor.
- Multiprogramming interleaves processes that performing I/O and that require CPU cycles.
- Disk I/O receives most attention, because of swapping and File system.

## ■ Must handle all I/O Devices in a uniform manner.

- Diversity of Devices causes difficulty to achieve true generality.
- Use hierarchical, modular approach to the design of I/O function.
- Hierarchical Layer, similar to Operating System.
- Each layer performs a related subset I/O function.
- Changes/Modifications in one layer do not require changes in another.

# I/O Systems

## I/O Facility

### Logical Structure of I/O Facility: Based on Hierarchical Layers

Layers of peripheral device using byte streams (i.e. character device drivers).

Logical I/O: General I/O functions, Open, Close, Read, Write.

Device I/O: Reqs converted to I/O instructs, commands, controller orders.

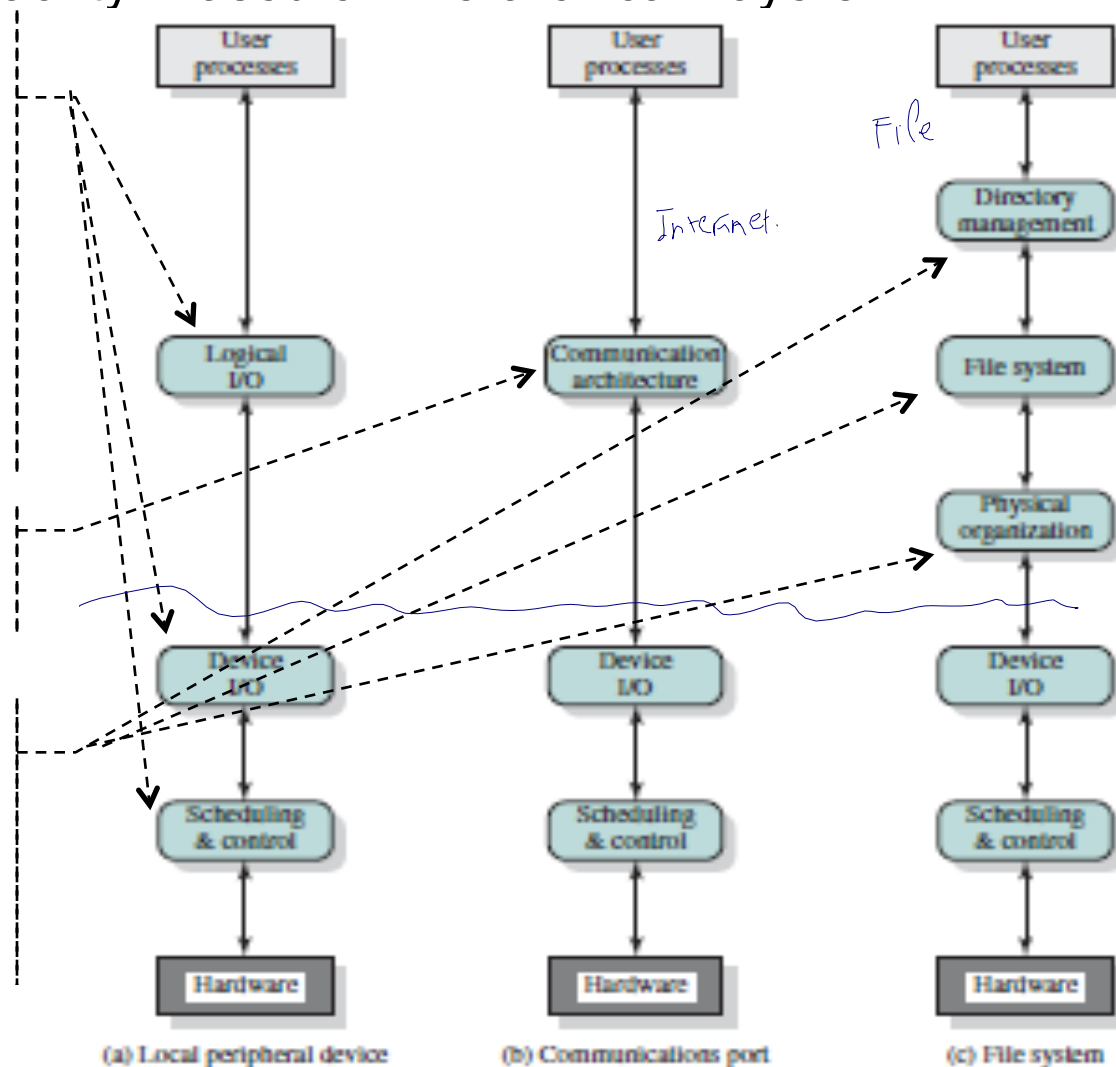
Scheduling & Control: Queuing and scheduling I/O Operations and control of the operations. Device Drivers, Interrupts, I/O Status.

Communication Architecture: Logical I/O becomes networking TCP/IP layer.

Directory Management: Organization of file system in directories.

File System: Logical structure and operations (Open, Close, Read Write).

Physical Organization: Logical to physical area (inode to sector).



# I/O Systems

## ■ I/O Hardware

## ■ Interrupts: Mechanism for the Controller to Notify CPU of Event.

- Polling inefficient, when I/O Device events are sporadic.
- CPU Hardware has Interrupt-ReQuest Line (IRQ) that is checked after every instruction is executed.
- Controller raises the IRQ when Controller has an event and must notify the CPU.
  - CPU performs state save and jumps to the Interrupt-Handler.
  - Interrupt-Handler clears interrupt and services the device.

## ■ Interrupt Handling Responsibilities:

- Defer Interrupt Handling during critical processing.
- Dispatch Interrupt Handler efficiently.
- Multilevel Interrupts: Distinguish between high-priority and low-priority interrupts.

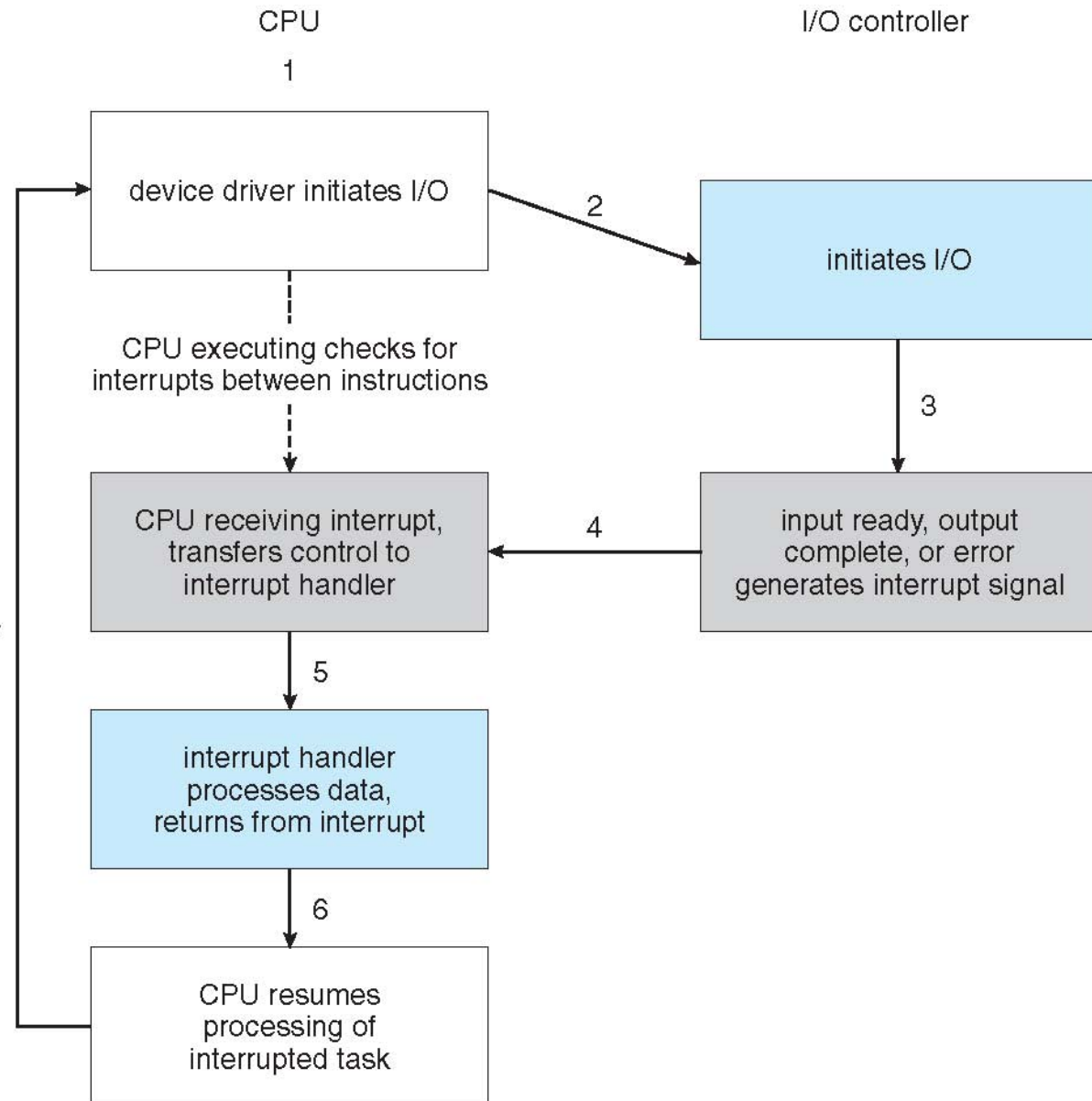
# I/O Systems

## ■ I/O Hardware

## ■ Interrupt-Driven I/O Cycle:

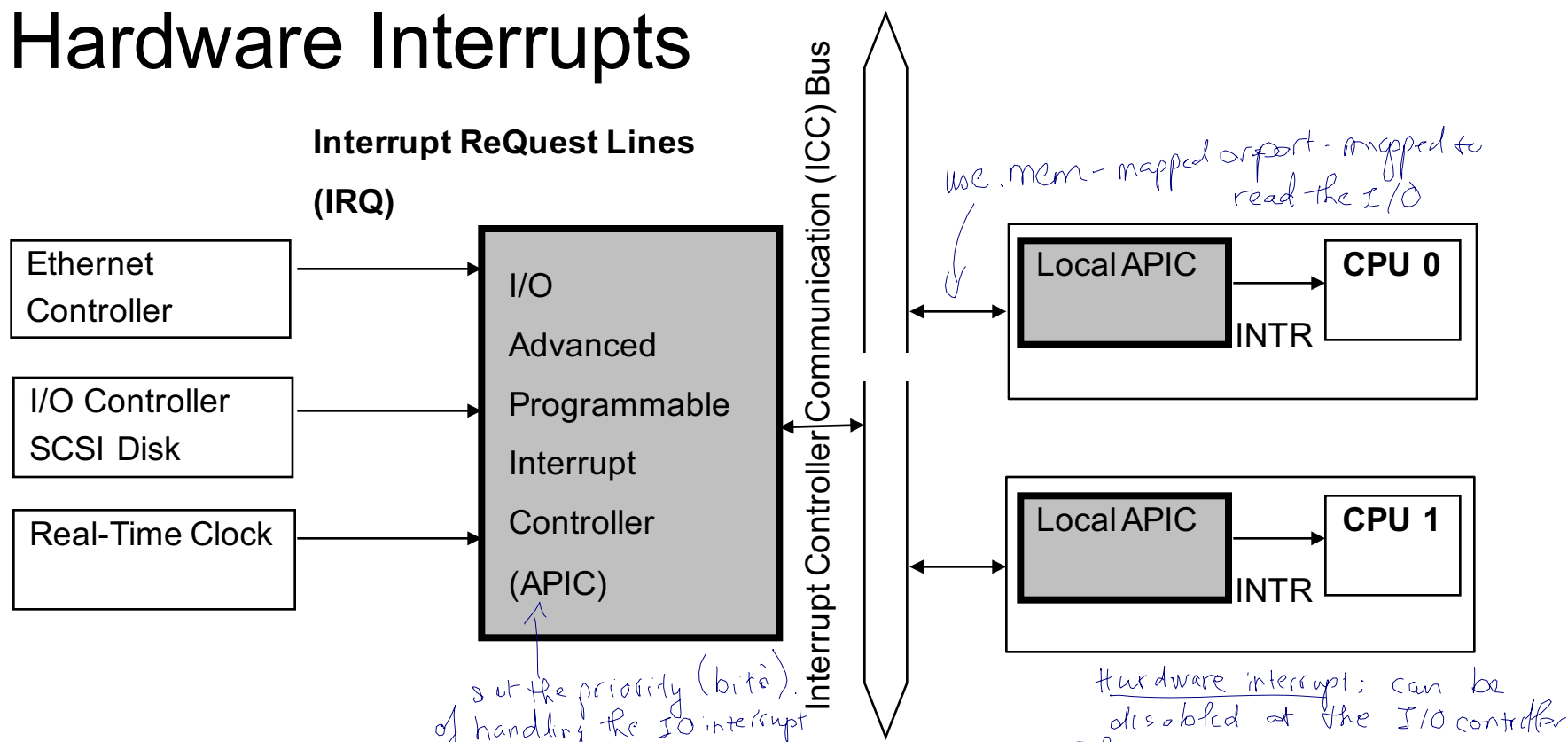
Basic interrupt mechanism enables CPU to respond to an async event, but interrupt handling feature must be expanded:

- need to defer interrupt handling during critical processing
- need to dispatch to proper interrupt
- ...



# I/O Systems

## ■ Hardware Interrupts



I/O devices: Unique or shared Interrupt ReQuest Lines (IRQs).

I/O APIC: 24 IRQ lines. Send/Receive APIC Messages over ICC BUS to Local APICs.

IRQ signals delivered to available CPUs (Static or Dynamic Distribution)

Local APIC: Mapped to interrupt vectors.

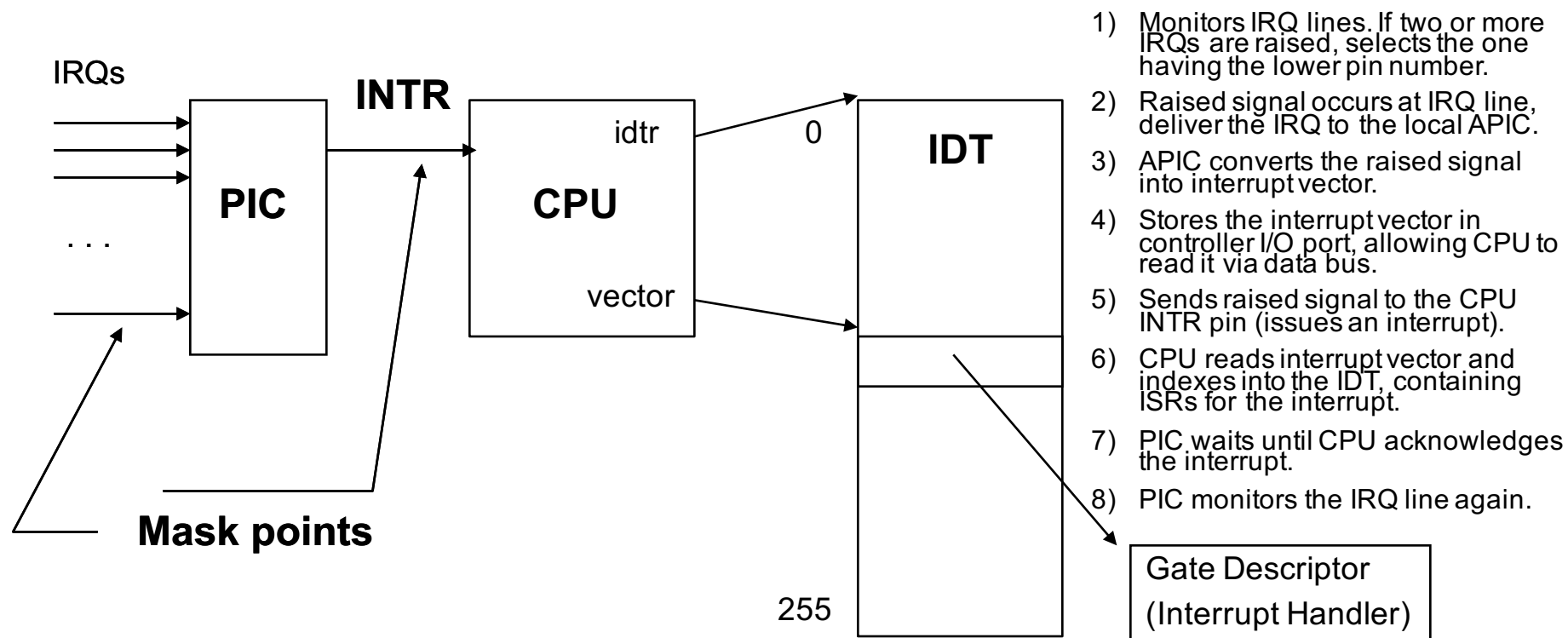
Signals processor with INTR (issues an interrupt).

## I/O Systems

- **Programmable Interrupt Controller Functionality**
  - Responsible for raising interrupt signal to the CPU when an external device sends IRQ.
    - Monitors the IRQ lines.
  - Controller translates IRQ to vector.
    - IRQ line 0 → Vector 32
      - $\text{IRQ}_n \rightarrow \text{Vector } n + 32$
    - Raises interrupt to CPU.
    - Places vector in register allowing CPU to read.
    - Waits for ACK from CPU. When ACK received, controller goes back to monitoring IRQ lines.
  - Controller prioritize multiple IRQs.
  - Controller can mask (disable) IRQs.

# I/O Systems

## ■ Programmable Interrupt Controller Functionality



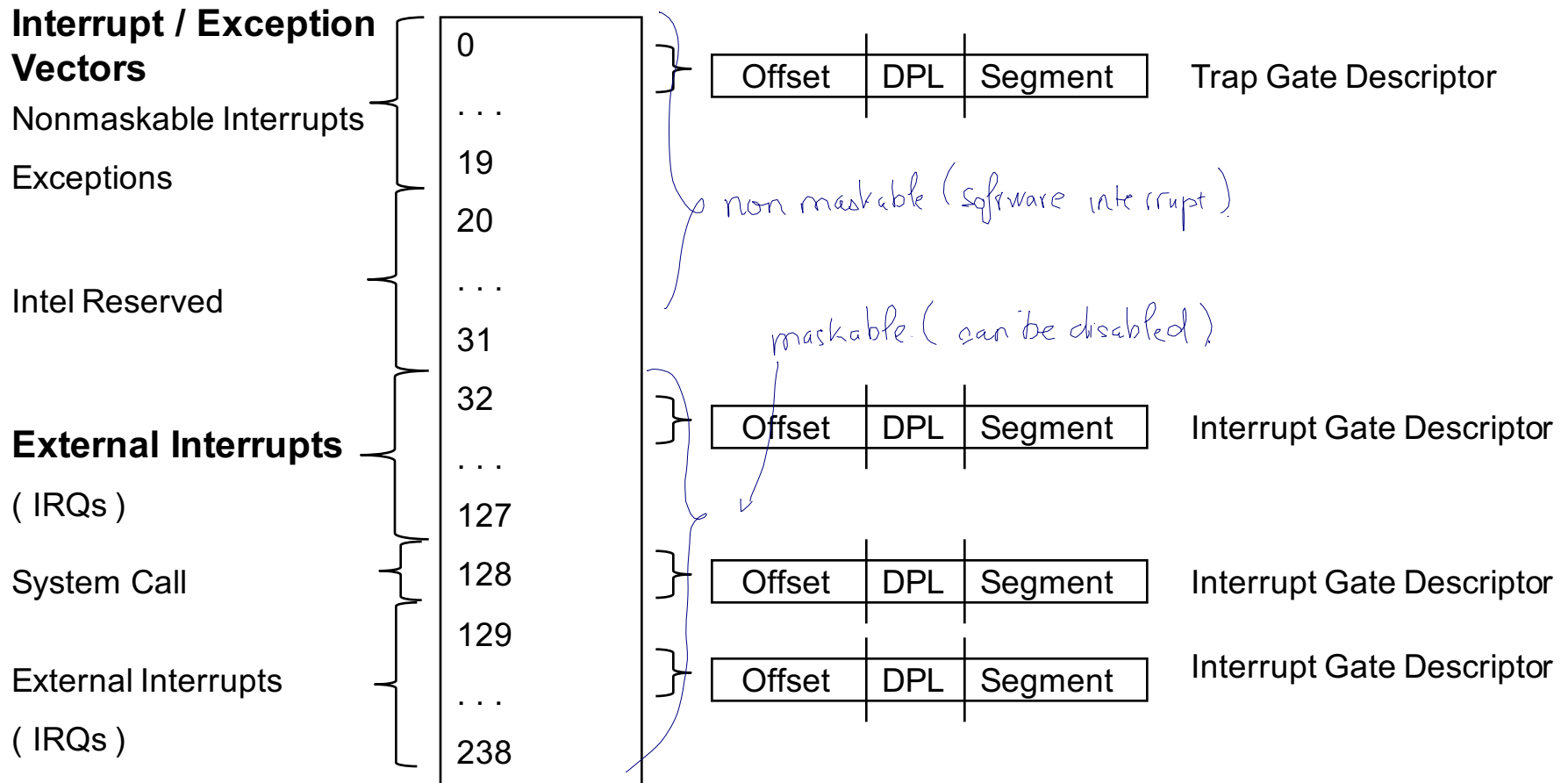
- o IRQ assignment is hardware dependent, fixed by architecture.
- o Linux device driver request IRQ when the device is opened.
- o Two devices that are not used at the same time can share IRQ.



# I/O Systems

## □ Interrupt Descriptor Table (IDT)

### Interrupt Descriptor Table



# I/O Systems

## I/O Interrupt Handlers

- I/O Interrupt Handler service multiple devices ( device share same IRQ).
  - Executes multiple Interrupt Service Routines (ISRs), since Interrupt Handler is not aware of which device generate IRQ.
- Actions to be performed in Interrupt Handler:
  - Critical: Must be executed immediately with maskable interrupts disabled. *← no more interrupt coming, code must be very small (Critical Section)*
    - Acknowledge interrupt to the Prog Interrupt Controller.
  - NonCritical: Must be executed immediately, but maskable interrupts are enabled.
    - Updating data structures accessible by processor.
    - Reading bar code after keyboard key pushed.
  - NonCritical Deferrable: Delayed without affecting kernel operations ( Softirqs and Tasklets).
    - Copying I/O buffer's contents to address space of process.
    - Sending keyboard line buffer to terminal handler process.

kernel  
function  
to handle  
the I/O

# I/O Systems

## ■ Interrupt Vectors

Vector Range	Use
0-19 (0x0 – 0x13)	<b>Nonmaskable interrupts and exceptions</b>
20-31 (0x14 – 0x1f)	Intel-reserved
32-127 (0x20 – 0x7f)	<b>External interrupts (IRQs)</b>
128 (0x80)	Programmed exception for system calls.
129-238 (0x81 – 0xee)	External interrupts (IRQs)
239 (0xef)	Local APIC timer interrupts.
240 (0xf0)	Local APIC termail interrupt function
241-250 (0xf1 – 0xfa)	Reserved by Linux for future use.
251 – 253 (0xfb - -0xfd)	Interprocess interrupts
254 (0xfe)	Local APIC error interrupts
255 (0xff)	Local APIC spurious interrupt

### IRQ Exception

0	Divide error
1	Debug
2	NMI
3	Breakpoint
4	Overflow
5	Bounds check
6	Invalid opcode
7	Device not available
8	Double fault
9	Coprocessor overrun
10	Invalid TSS
11	Segment not present

### IRQ INT Hardware Device

0	32	Timer
1	33	Keyboard
2	34	PIC cascading
3	35	Second serial port
4	36	First serial port
6	38	Floppy Disk
8	40	System clock
10	42	Network interface
11	43	USB port, sd card

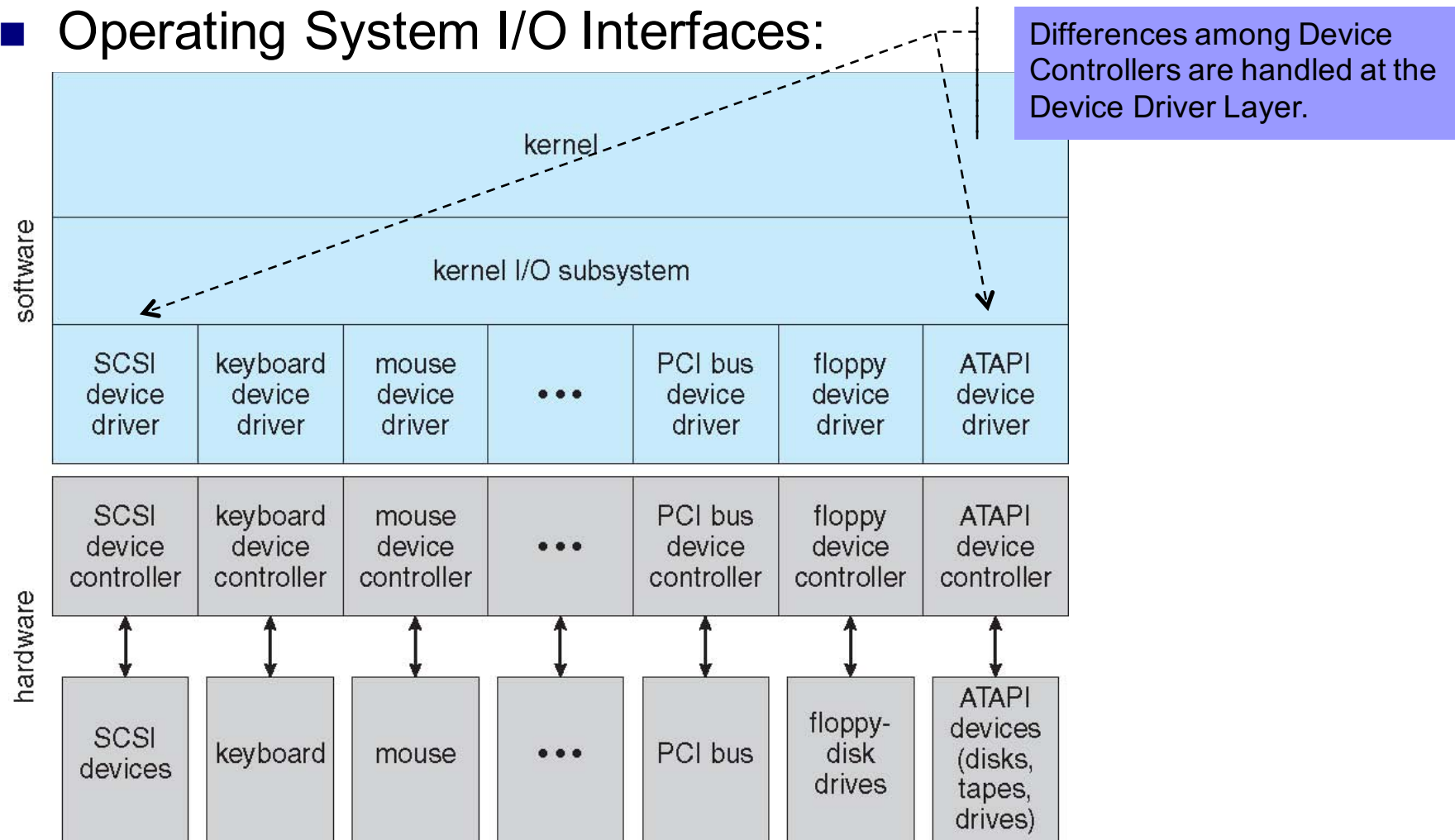
...

# I/O Systems

- Application I/O Interface
- How can I/O Devices be treated in a uniform way?
- Operating System I/O Interfaces:
  - Structuring techniques involving abstraction, encapsulation, and software layering.
  - Application can open file on a storage without knowing what kind of disk or how new devices can be added without disruption.
- Purpose of Device-Driver Interface is to hide the differences among Device Controllers from the I/O Subsystem of the Operating System.
  - I/O Subsystem independent of the hardware simplifies the Operating System.
  - I/O System Calls encapsulate device behaviors in generic classes that hide hardware differences from applications.

# I/O Systems

- Application I/O Interface
- Operating System I/O Interfaces:



# I/O Systems

- Application I/O Interface
- Operating Systems has different standards for Device-Driver Interfaces.
  - A Device can ship with multiple Device Drivers: for MS-DOS, Windows, UNIX.
- Device Characteristics vary:
  - Character-stream or block: Unit is a Byte or Block of Bytes.
  - Sequential or random access: Fixed sequential order or seek.
  - Synchronous or asynchronous: Predictable response time or irregular response time.
  - Sharable or dedicated: Concurrent or dedicated process access.
  - Speed of operation: Few bytes or few gigabytes per second.
  - Read-Write, Read Only, or Write Only: Multi- or uni-directional.

# I/O Systems

## ■ Application I/O Interface

### □ Characteristics of I/O Devices:

I/O Devices vary in different device aspects (i.e. device data transfer mode can be character or block type, or device access method could be sequential or random.

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

# I/O Systems

- Application I/O Interface
- Device Characteristic hidden by Operating System for Application Access.
- Device grouped into few conventional types with access types.
  - Block I/O Access Type.
  - Character-Stream I/O Access Type.
  - Memory-Mapped File Access Type.
  - Network Access Type.
  - Special Types: Time-of-Day Clock, Timer.
- Application access to Device Driver:
  - `ioctl ( )`: I / O Control. → sends commands to device driver → system call.  
(not common cmds)
  - Enables application to access any functionality implemented by Device Driver.



# I/O Systems

## ■ Application I/O Interface

## ■ Block Devices:

- Disk drives and other block-oriented devices (i.e. CDROM).
- **Commands expected to support: read (), write (), seek ().**
- Application access options for Block Device:
  - File-system access.
    - Operating System supported mode.
  - Raw I/O: Linear array of blocks.
    - Application performs its own buffering and locking mechanism.
    - UNIX Direct I/O.
  - Memory-Mapped File access. *← share file.*
    - Access to disk storage via Demand-Page virtual memory access.
    - Data transfers occur when access memory image.

## ■ Character Devices:

- Include keyboards, mice, serial ports.
- **Commands expected to support character: get(), put().**
- Libraries layered on top allow line-at-a-time editing.

# I/O Systems

- Application I/O Interface
- Network Devices
- Different Performance and Addressing Characteristics than Disk I/O.
  - Network Socket Interface: *← network protocol layer.*
    - Create a Socket: `socket ()`.
    - Connect Local Socket to Remote Socket: `connect ()`.
    - Listen for remote applications to connect to Local Socket: `listen ()`.
    - Listen for specific Sockets which has packet waiting to be received and which can send packets: `select ()`.
    - Send and Receive packets over the connection: `send ()`, `sendto ()`, `recv ()`, `recvfrom ()`.
    - Close Local Socket: `close ()`.
    - Socket status (Show sockets): `netstat -a`
    - Creation of Distributed Applications: Using network hardware and protocol stack.
    - UNIX Inter-process and Network Communication:
      - Half-Duplex Pipes, Full-Duplex FIFOs, Full-Duplex STREAMS, Message Queues, and Sockets.

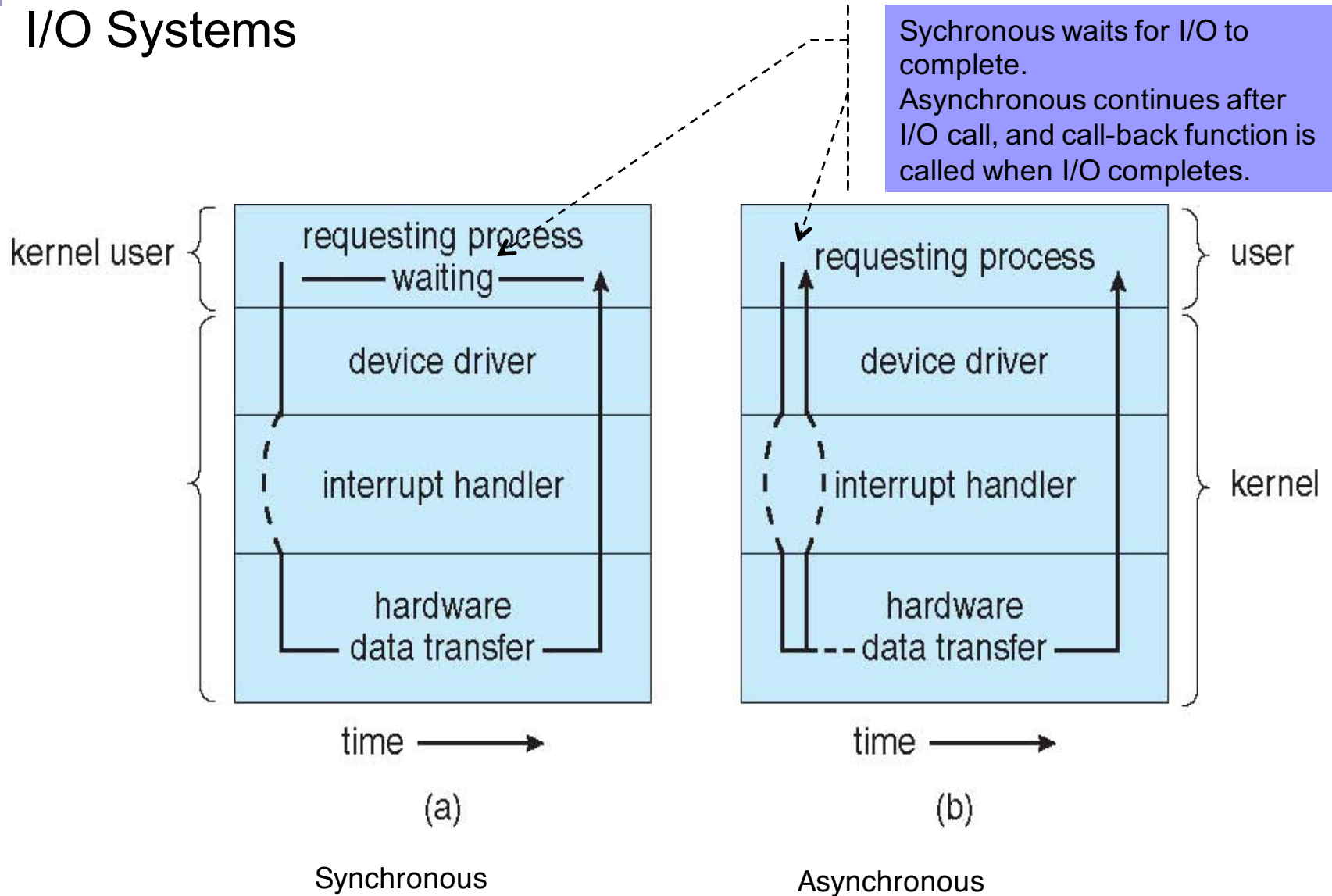
# I/O Systems

- Application I/O Interface
- Computer System Hardware Clocks and Timers:
  - Provide Current Time, Elapsed Timer, and Timer set to trigger Operation.
- Programmable Interval Timer: Hardware to measure elapsed time and trigger operations.
  - Used heavily by Operating System for Time-sensitive Applications.
- Periodic Interrrupts:
  - Disk I/O Subsystem periodic flushing of dirty cache buffers to disk.
  - Network Subsystem timeouts due to network congestion or connection failure.
  - System Calls for User processes to use timers.
  - Timer Device Drivers: Simulate Virtual Clocks to support many timer requests.
  - UNIX Scheduler:
    - Interrupt generated at each Tick to schedule CPU between multiple processes ( Time Sharing Principle ).
    - Tick is kept in kernel variable HZ ( Number of Ticks per Second ).
    - For X86 processor,  $HZ = 250$ ,  $Tick = 1 / 250 = .004 = 4 \text{ milliseconds}$ .

# I/O Systems

- I/O System Call Interface: Blocking and Non-Blocking
- Application Issues Blocking I/O: *← wait for response*
  - Process suspended until I/O system call completed.
  - Receives value of the I/O system call.
  - I/O Device action asynchronous and time unpredictable.
  - Overlap execution by using multi-threaded application, some threads perform blocking I/O system call while others continue execution.
- Application Issues Nonblocking I/O: *← wait for event / callback*
  - I/O system call returns as much data as available.
  - User application receiving keyboard and mouse input.
  - Returns quickly with count of bytes read or written or none.
- Asynchronous I/O: *← wait for I/O to complete.*
  - Process continues after making I/O system call.
    - Event or Software Interrupt.
    - Call-back routine executed outside of application's control flow.
  - I/O subsystem signals process when I/O in entirety completed (Nonblocking I/O returns data available at the time of the call).

# I/O Systems



# I/O Systems

## ■ Kernel I/O Subsystems Services

- Kernel provides many services related to I/O.
- Provides many services related to I/O:
  - I/O Scheduling: Schedule set of I/O Requests.
  - Buffering: Intermediate memory area during data transfer between two devices or device and application.
  - Caching: Region of fast memory used to hold disk sectors.
  - Spooling and Device Reservation: Manages operations that cannot be interleaved.
  - Error handling: Return status.
  - I/O Protection: System Calls to protect privileged I/O instructions.
- Kernel Data Structures: Kernel keeps state information about the use of I/O components.

# I/O Systems

## ■ Kernel I/O Subsystems Services

## ■ I/O Scheduling *← handled by the kernel*

### ■ Schedule Wait Queue of I/O Device Requests:

- I/O Device Requests placed in queue and reordered to improve system efficiency.
- Reduce average latency for I/O.
- Operating System Scheduling Algorithm must have fairness, not cause an application request to wait indefinitely (starve the request).

### ■ Wait Queue for each Device Attached to Device Status Table.

### ■ Scheduling Algorithms:

- FCFS (First Come First Serve): Simple, fair, but worst I/O service.
- SSTF (Shortest Seek Time First): Order I/O Requests such that all requests close to the current Head Position serviced first.
- SCAN: Disk Head moves from inner to outer cylinder servicing I/O Requests and then reverse directions while continuing to service.
- C-SCAN: Circular SCAN Disk Head moves from inner to cylinder and back to beginning.

# I/O Systems

- Kernel I/O Subsystems Services
- I/O Scheduling

## Device-Status Table

device: keyboard status: idle
device: laser printer status: busy
device: mouse status: idle
device: disk unit 1 status: idle
device: disk unit 2 status: busy
⋮

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

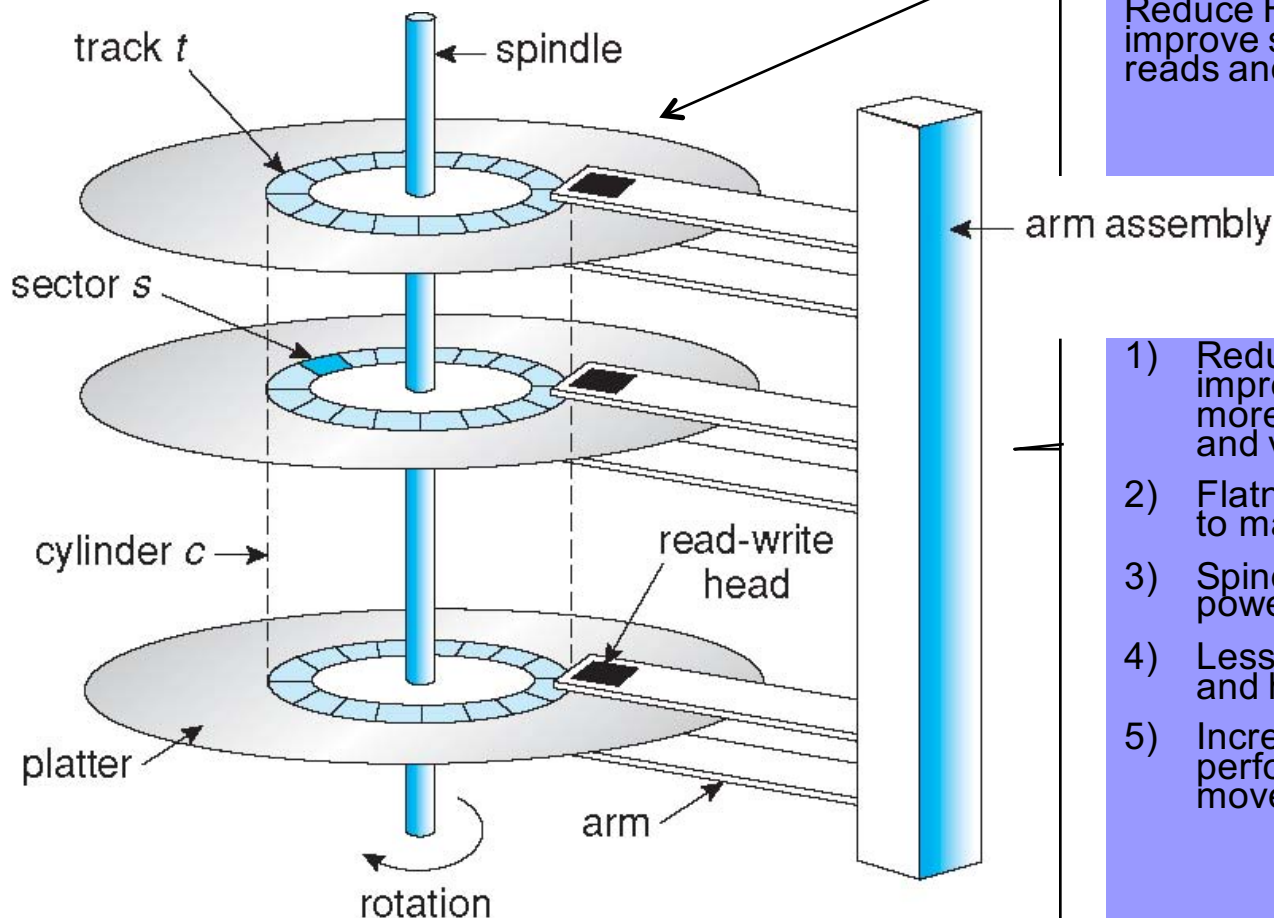
Device-Status Table:  
Entry for each I/O Device with  
Wait Queue attached.

Wait Queue entries are  
ordered by Operating System  
based on I/O Scheduling  
Algorithm.



# I/O Systems

## ■ Magnetic Disks: Secondary Storage of Computer Systems



Platter Size related to performance.

Reduce Head movements and improve seek times (Faster reads and writes).

- 1) Reduce platter size, improve stiffness and more resistant to shock and vibrations.
- 2) Flatness of surface easier to manufacture.
- 3) Spindle spins faster, less-powerful motors.
- 4) Less power, reduces noise and heat.
- 5) Increases seek performance, less head movements.

# I/O Systems

## ■ Disk Structure

## ■ Mapped as one-dimensional Array of Logical Blocks

- Logical Block: Typically 512 Bytes.
- Sector 0: First sector of first track on outermost cylinder.
  - Mapped outermost cylinder to innermost cylinder.
- Disk Address: Cylinder number, Track number, Sector number
  - Difficult to translate logical block to physical sector.
    - Number of sectors per track not constant in some drives.
  - Further from center, greater number of sectors per track.
  - CLV (Constant Linear Velocity) specifies uniform density of bits per track.
    - Drive increases rotational speed when head moves from outer to inner track.
    - Used in CDROM and DVDROM.
  - CAV (Constant Angular Velocity) specifies bit density of inner track less to outer track.
    - Drive rotational speed constant.
    - Used in hard disk.

# I/O Systems

## ■ Disk Performance Parameters



SeekTime: Time required to move the disk arm to the required track.

Rotational Delay: Time required for the addressing area of the disk to rotate into a position where it is accessible by the read/write head.

Data Transfer: Transfer time to or from the disk depends on the rotation speed of the disk.

# I/O Systems

## ■ Disk Scheduling

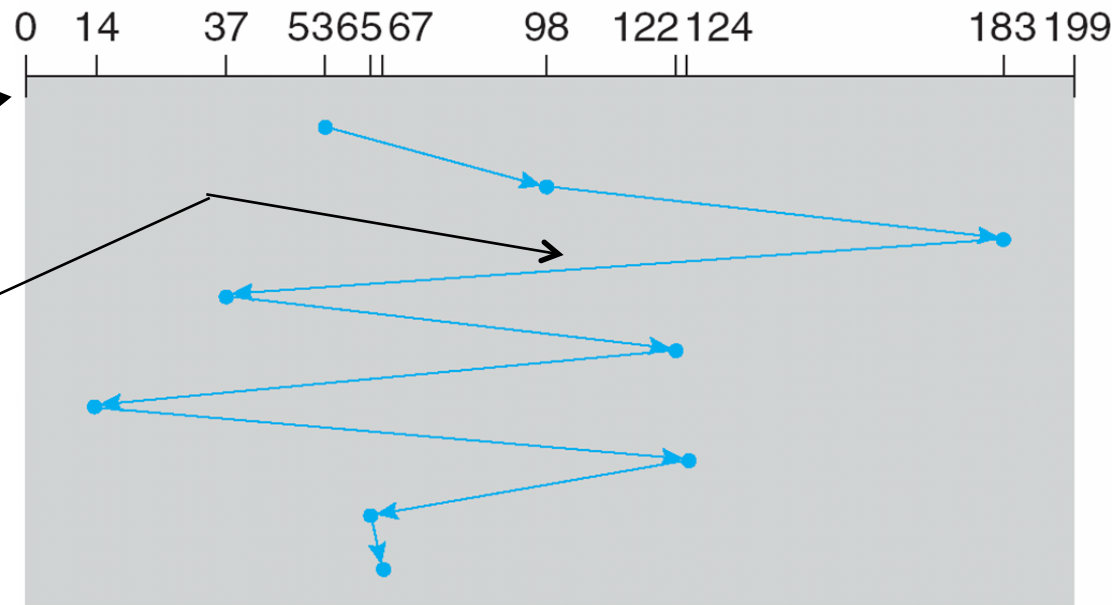
### ■ FCFS Scheduling: First Come First Serve

- Fair, but does not provide the fastest service.
- Large swings of the disk head possible, causing lengthy Access Time.

queue = 98, 183, 37, 122, 14, 124, 65, 67  
head starts at 53

Cylinders:  
Seek time is the  
Head Movement  
to a Cylinder.

Large jumps  
between  
Cylinders.



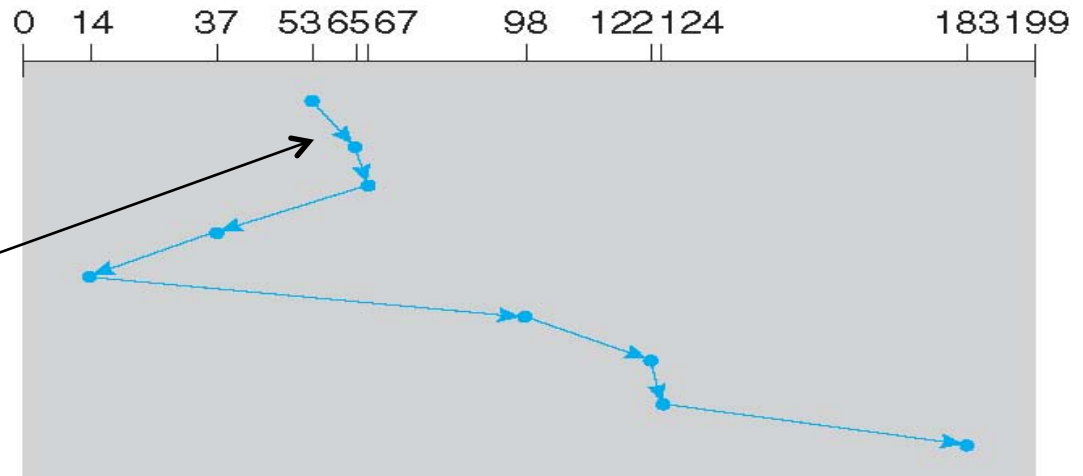
# I/O Systems

## ■ Disk Scheduling

### ■ SSTF Scheduling: Shortest Seek Time First

- Service requests closes to current Head Position before moving Head further away ( minimize Seek Time ).
- Similar to Shortest Job First Scheduling.
- Starvation of Job furthest from the current request queue.

queue = 98, 183, 37, 122, 14, 124, 65, 67  
head starts at 53



Cylinders:

Minimize Cylinder access ( seek ) time.

Servicing most recent should result in possibly no head movement (moving through a sequential file).

Favoring locality improves throughput.

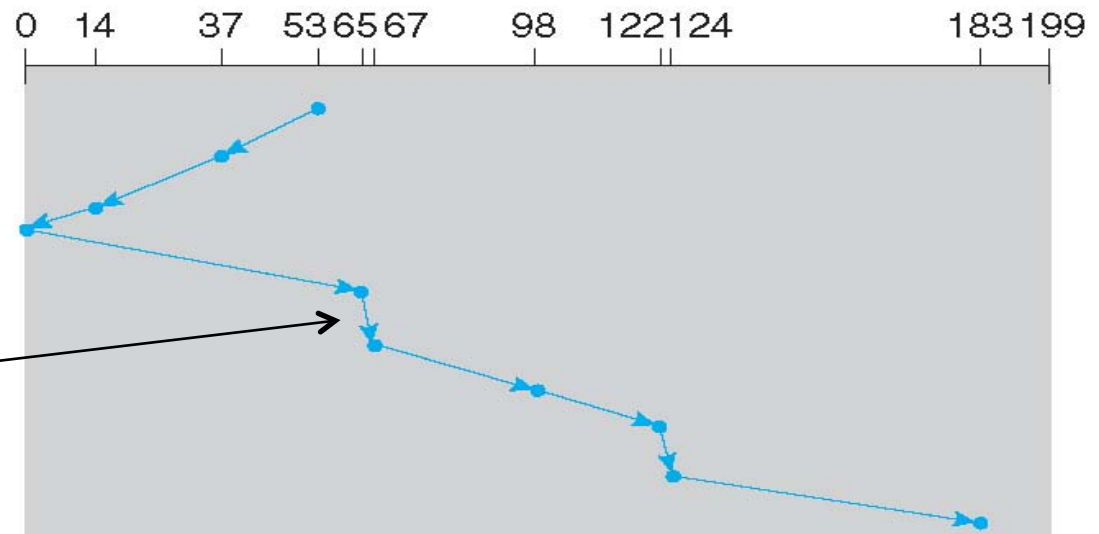
# I/O Systems

## ■ Disk Scheduling

## ■ SCAN Algorithm

- Elevator Algorithm: Disk arm starts at one end of the disk and moves towards the other end, servicing requests at each cylinder. When the end is reached, head movement is reversed.
- Requests collect at the cylinders that was just passed.

queue = 98, 183, 37, 122, 14, 124, 65, 67  
head starts at 53



Cylinders:

Access chart similar to SSTF.  
Minimize Cylinder access (seek)  
time.

# I/O Systems

## ■ Disk Scheduling

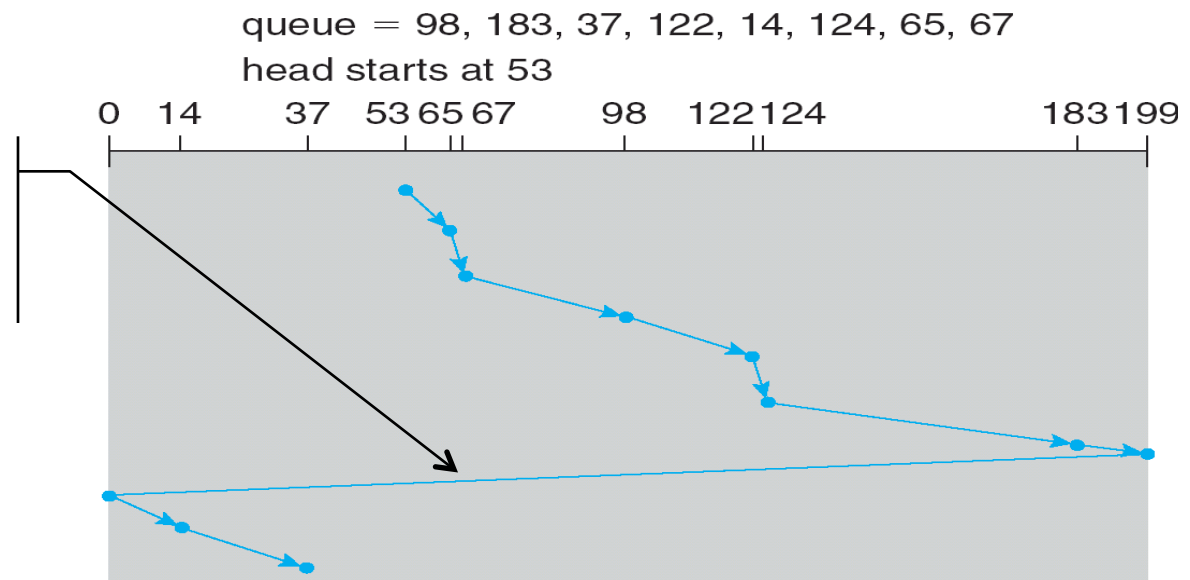
## ■ C-SCAN Algorithm

□ Variant of SCAN with a uniform wait time.

- Head moves from one end to the other end servicing requests.
- When other end reached, immediately returns to beginning before servicing requests.
- Treats cylinders as circular list that wraps from last to first cylinder.

Cylinders:

Wraps from last cylinder to first cylinder before servicing requests again.



# I/O Systems

- Disk Scheduling
- C-LOOK or LOOK Algorithm

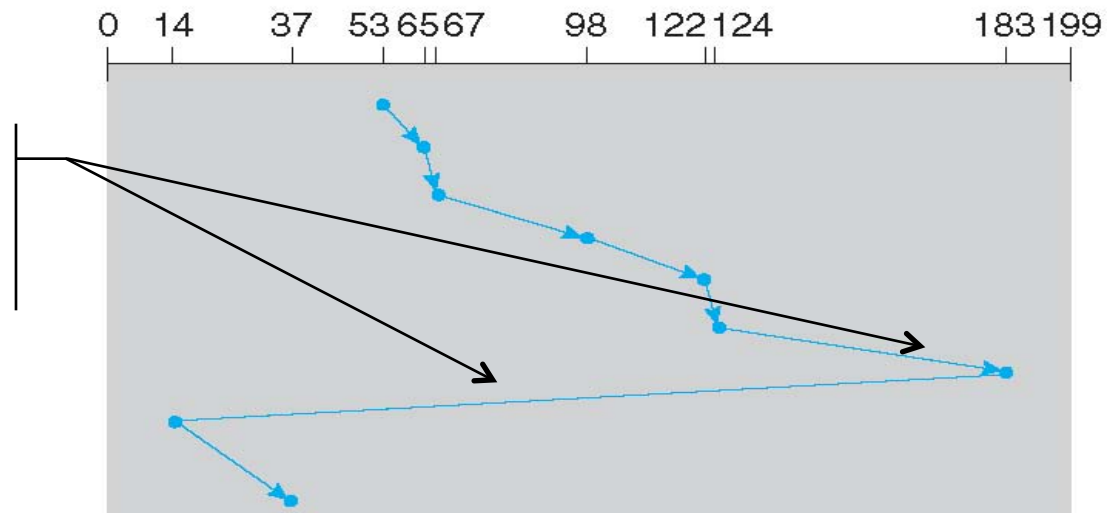
- Variant of C-SCAN

- Head moves from one end to the other end servicing requests.
    - Head only goes as far as final request in one direction before reversing directions.
    - SCAN and C-SCAN typically follows this pattern.

queue = 98, 183, 37, 122, 14, 124, 65, 67  
head starts at 53

Cylinders:

Wraps as soon as last request is reached.





# I/O Systems

## ■ Disk Scheduling

## ■ Selection of Disk-Scheduling Algorithm

- SSTF simple, performs better than FCFS.
- SCAN, C-SCAN perform better for systems with heavy disk usage.
  - Require retrieval algorithm.
- File Allocation Algorithm important.
  - Contiguous blocks will have minimum head movement.
  - Linked or Indexed File can have blocks scattered on the disk.
- Caches for directories and index blocks required, otherwise frequent directory access causes excess head movements.
- SSTF or C-LOOK ( LOOK ) typically used as default algorithm.

## ■ Disk Scheduling Algorithm written as separate module, allowing it to be replaced with a different algorithm.

# I/O Systems

## ■ Kernel I/O Subsystems Services

## ■ Buffering:

- Another means to improve efficiency of I/O is by using storage space in Main Memory.
  - Store data in memory while transferring between devices.
- To cope with device speed mismatch between producer and consumer of a data stream.
  - Modem transfer rate 1000 times slower than Disk Drive.
- To cope with device transfer size mismatch.
  - Buffer to reassemble fragmented packets in Networking.
- To maintain “copy semantics” (data buffered in system memory after system call).
  - In write () system call, application buffer copied to system buffer before returning to application.
  - Application can modify its buffer after the return.

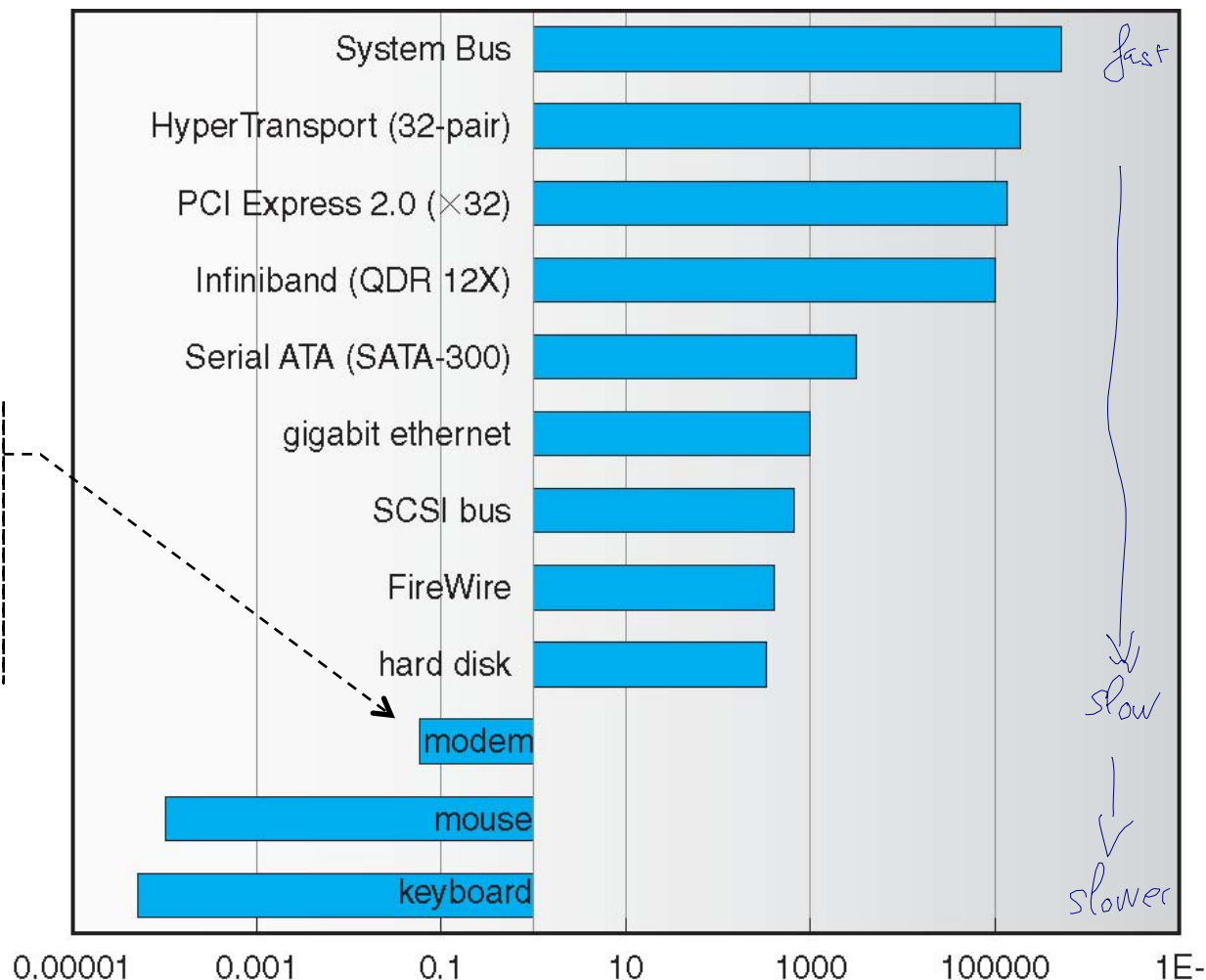
# I/O Systems

## ■ Kernel I/O Subsystems Services

## ■ Buffering:

## ■ Device Transfer Rate.

Disparity in Device Transfer Rates in Sun Enterprise 6000 device. Modem transfer rate is 1000 times slower than Hard Disk.



# I/O Systems

## ■ Kernel I/O Subsystems Services

## ■ I/O Buffering Schemes (Input):

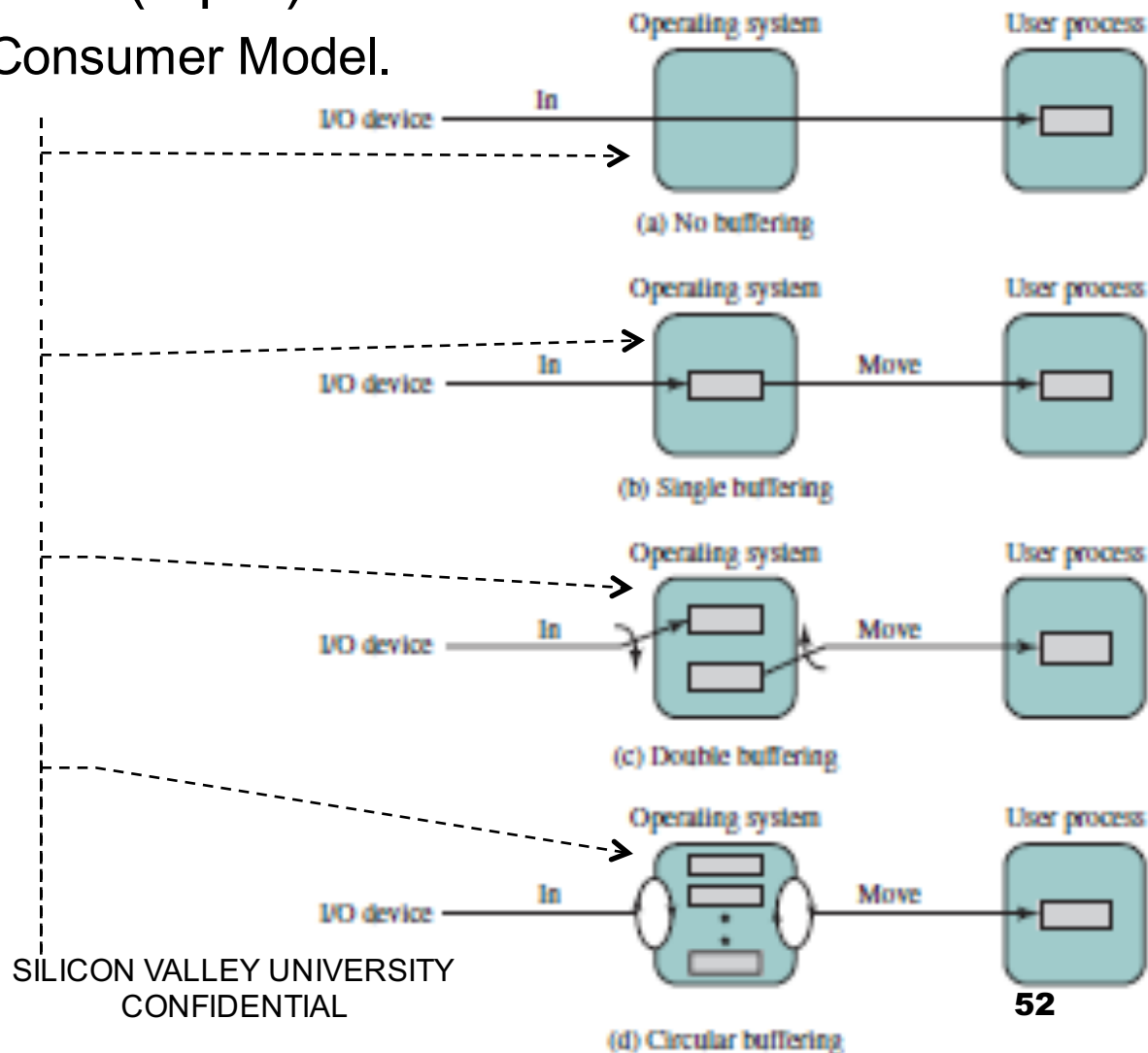
- Follows Producer-Consumer Model.

Process hung waiting for slow I/O.  
Cannot swap because user memory receiving data is locked, even though I/O request is queued.

After transfer, process moves block into user space and requests another block (read ahead). Data usually access sequentially.

Block-Oriented transfers, improvement, but complex. Stream-Oriented transfers, similar to single buffering scheme.

Process must handle rapid burst of I/O. I/O Operation must keep up with the process.  
Bounded-Buffer Producer-Consumer Model.



# I/O Systems

## ■ Kernel I/O Subsystems Services

## ■ Caching

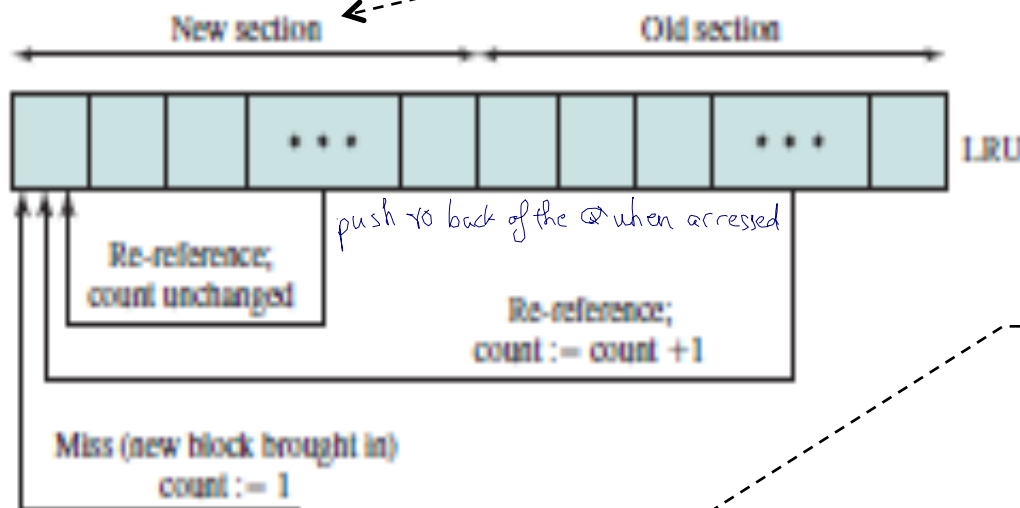
- Fast Memory holding **copy** of data.
  - Instructions of current process.
  - Translation Look-Aside Buffer ( TLB ) supporting Paging Logical Address to Physical Address mapping.
- Buffers used as cache: May hold only existing copy of data item.
  - Data buffered in memory for I/O efficiency with large data transfers.

## ■ Disk Cache

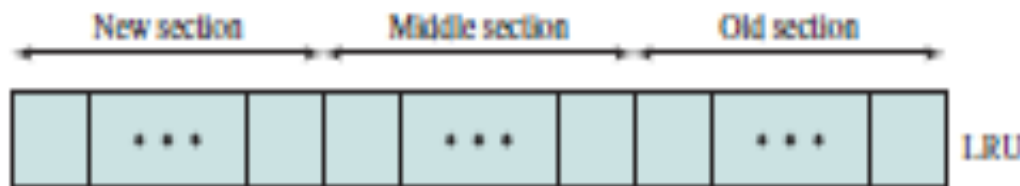
- Buffer in Main Memory for Disk Sectors.
- I/O Request for a Disk Sector, request is satisfied from cache.
  - Locality of reference, subsequent reference will be to the same data block.
- Sector Replacement Strategy:
  - Least Recently Used (LRU): Replace block in the cache longest with no reference.
  - Least Frequently Used (LFU): Counter for each block referenced, replace block with fewest reference.

# I/O Systems

- Kernel I/O Subsystems Services
- Disk Cache: LRU



(a) FIFO



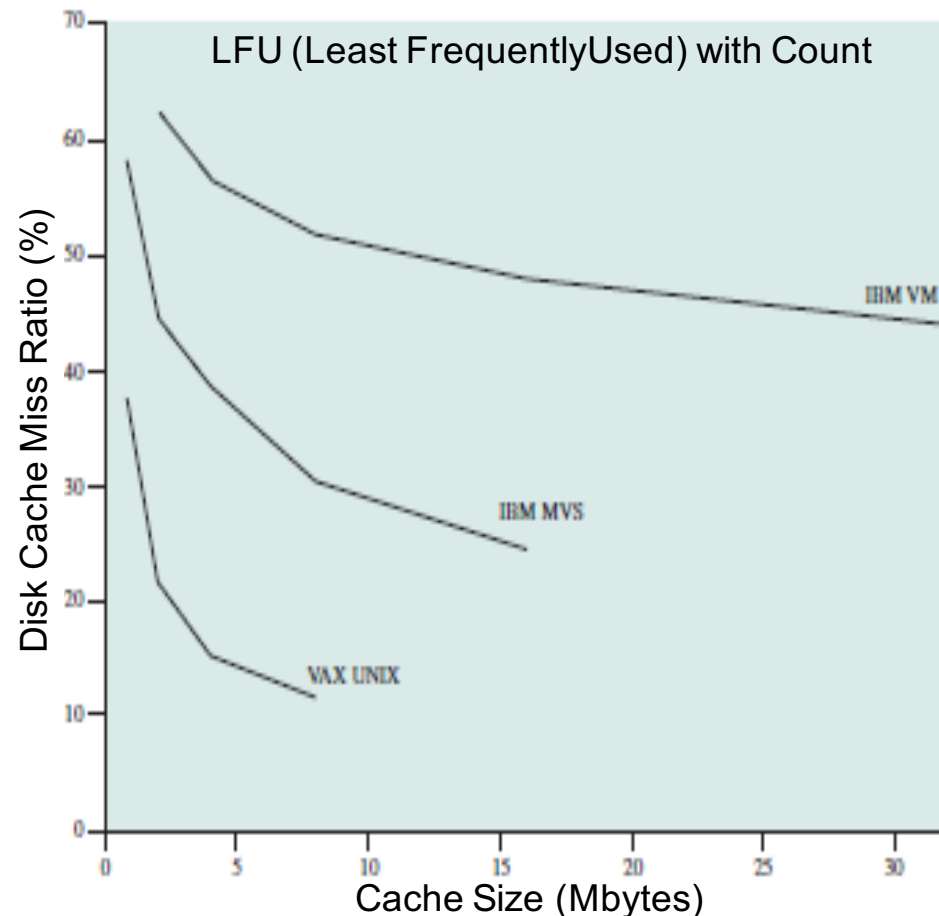
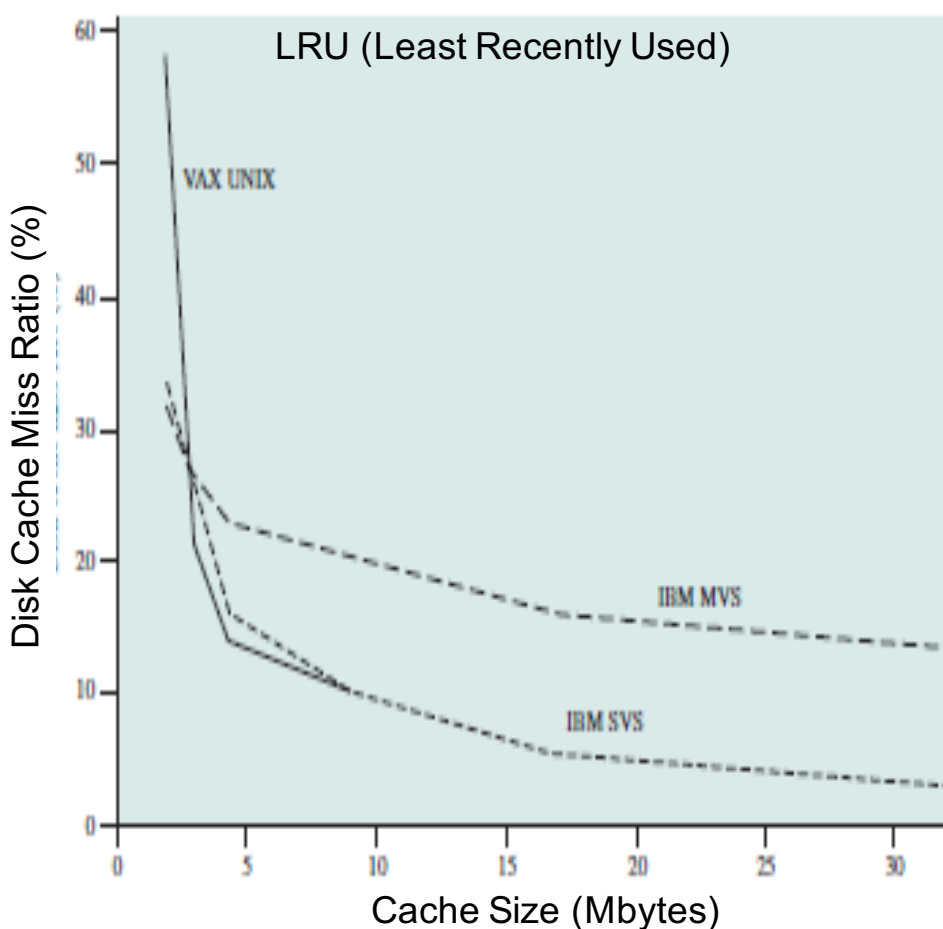
(b) Use of three sections

Blocks accessed frequently does not increase their reference count.  
New Section: Block referenced moved to top of queue, but count not incremented.  
Old Section: Block referenced moved to top of queue and count incremented.

New Section blocks cannot increment until out of New Section.  
Allow blocks to build up reference count.  
Middle Section: Block cannot be replaced while in Middle section. Count will be incremented.  
Old Section: Same as before.

# I/O Systems

- Kernel I/O Subsystems Services
- Disk Cache: LRU



# I/O Systems

## ■ Kernel I/O Subsystems Services

## ■ Spooling ← order the requests that come in

- I/O Devices that cannot handle requests from multiple concurrent applications: CANNOT accept interleaved data streams.
  - Printer Device can service only one print job at a time.
  - Operating System intercepts I/O requests to the printer and spools to a separate Disk File, instead of sending to the printer. Spooled file is then queued for output to the printer.
  - Tape Drives uses spooling to coordinate concurrent output.



# I/O Systems

## ■ Kernel I/O Subsystems Services

## ■ Device Reservation

- Device (printers) cannot multiplex I/O Requests of concurrent apps.
- Operating System ( IBM VMS ) provides exclusive access to device.
  - Application must allocate an idle Device.
  - De-allocate Device when no longer needed.
- Operating System can allow ONLY one application to access device.
  - Windows NT has system call to wait until a device becomes available.
- Application can assume responsibility to avoid Deadlock.

## ■ Error Handling

- Operating System compensates for I/O Device transient failures.
  - Disk read ( ) failure results internally in a read retry.
  - Network send ( ) failure results internally in a resend.
- I/O System Call returns error code when I/O Device request fails.
  - UNIX/Linux: Variable “errno” returns error code to application.
- Device Drivers provide detailed error information in System Logs.
  - UNIX/Linux: /var/log/messages

# I/O Systems

- Kernel I/O Subsystems Services

- 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.

- Must be done through Operating System.

- I/O must be performed via system calls.

- Operating System checks for valid request.

- Memory-Mapped and I/O Port memory locations must be protected.

- Exception to the rule:

- Graphic games and video editing and playback software need direct access to memory-mapped graphics controller memory.

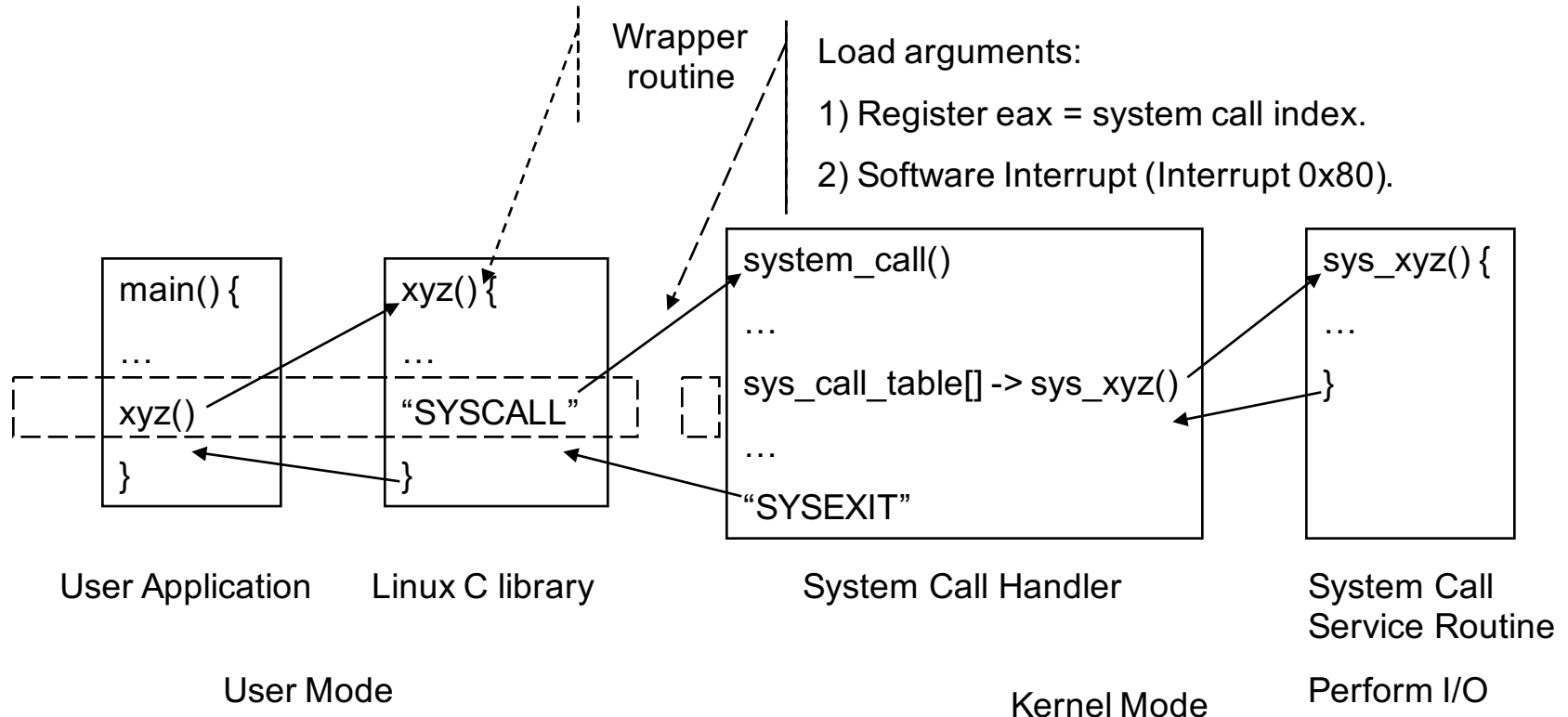
- Graphics memory allocated to one process at a time.

# I/O Systems

## ■ Kernel I/O Subsystem Services

## ■ I/O Protection

- System Call – Explicit request to the kernel made through a software interrupt. Service provided in the kernel, cross the user-space / kernel boundary.
- Linux C library – Provides the wrapper routines to issue a system call.



# I/O Systems

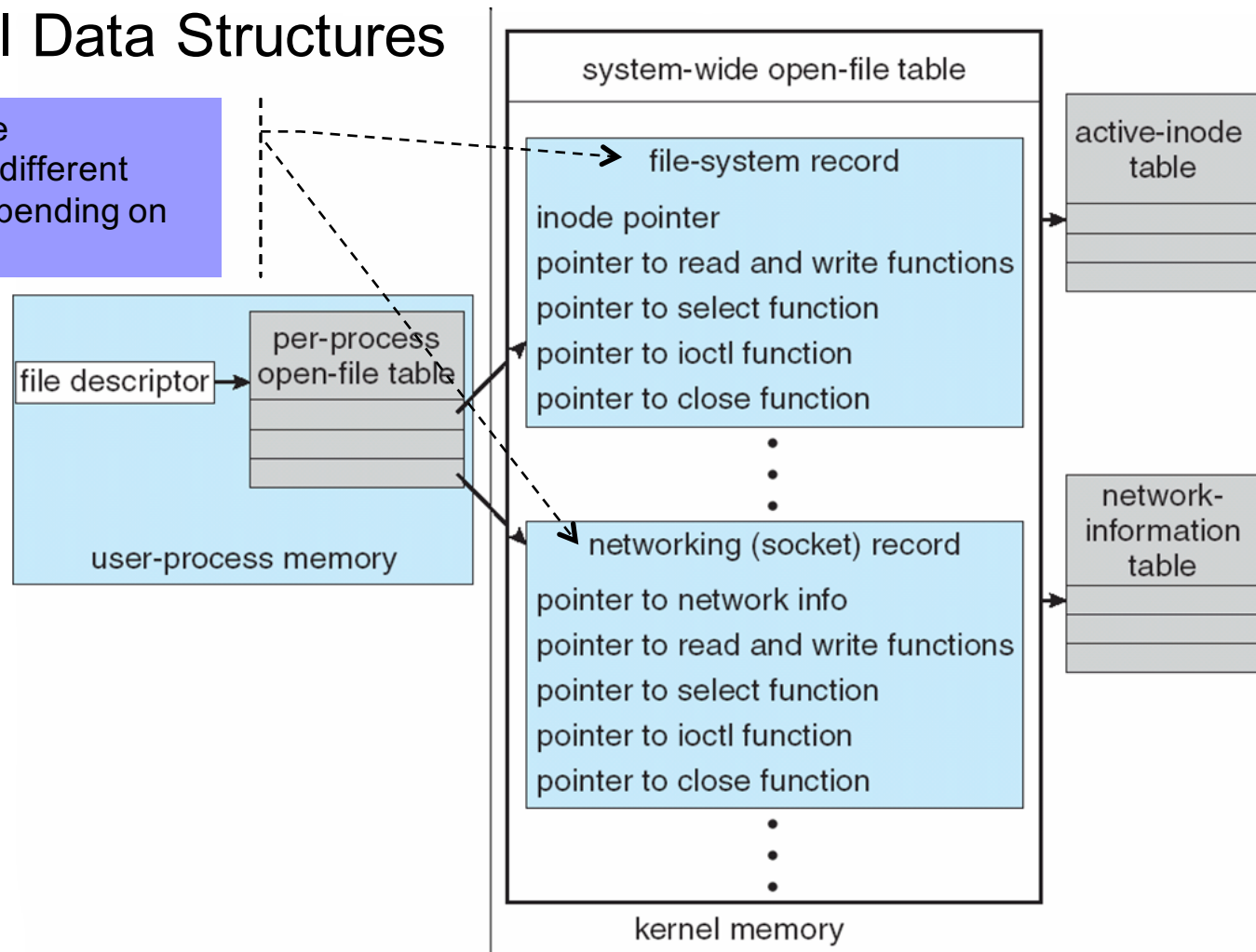
- Kernel I/O Subsystems Services
- Kernel Data Structures
- Kernel keeps state info for I/O components in in-Kernel Data Structures.
  - Including open file tables, network connections, character device state.
- Open-File Record: Contains Dispatch Table holding pointers to routines, depending on the type of file.
  - Each supports same functions, but semantics differ.
- Some Operating Systems use object-oriented methods and message passing to implement I/O.
  - Windows NT (Hybrid OS based on Mach MicroKernel)  
Messaging → I/O Request → Kernel → I/O Manager → Device Driver

# I/O Systems

## ■ Kernel I/O Subsystems Services

## ■ Kernel Data Structures

Open-File Table  
Entries contain different  
information, depending on  
File Type.



# I/O Systems

- Transforming I/O Requests to Hardware Operations
- How is a File Name connected by Operating System to a Disk Sector?
- MS-DOS:
  - First part of File Name preceding colon is the specific hardware device ( “c:<filename>” where “c” represents primary disk ).
  - Specific device name must be specified ( “c:<filename>” ).
- UNIX
  - Device name in regular File-System name.
  - Mount Table: Associates prefixes of path names with device names.
    - Longest matching prefix; entry in Mount Table gives device name.
  - Device name in the File-System Directory structure has a <major, minor> device number, instead of an inode number.
    - Major device number identifies Device Driver to be called to handle I/O to the device.
    - Minor device number used by Device Driver to index into Device Table containing port address or memory-mapped I/O address of device controller.

# I/O Systems

- Transforming I/O Requests to Hardware Operations

- MS-DOS:

- 1) MS-DOS file name uses “:” to separates device name from file-system name space (i.e. “c:<file name>”).
- 2) File name maps to a number in the File Directory.
- 3) Number is an entry in the File Allocation Table.
- 4) Table entry is the first disk block and all the blocks for the file is found by following the chain of disk blocks.

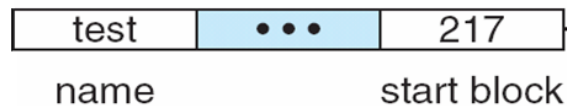
- UNIX:

- 1) UNIX file name has device name incorporated into file-system name space: Path name map into Mount Table to the device and File System.
- 1) File name maps into an inode number in the File Directory.
- 2) Inode number index into Inode Table.
- 3) Table entry contains space-allocation information.

# I/O Systems

## ■ MS-DOS File Allocation Table ( FAT )

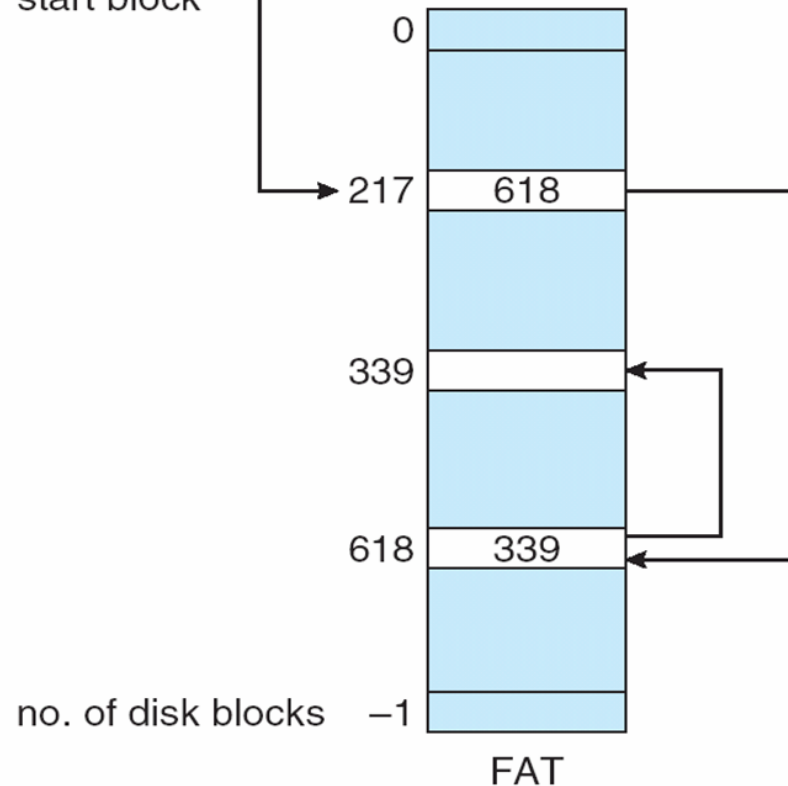
directory entry



**Directory Entry** contains offset to the start of the File in the FAT.

### **File Allocation Table ( FAT ):**

Each entry represents a physical block on disk and each entry points to the next entry (block).





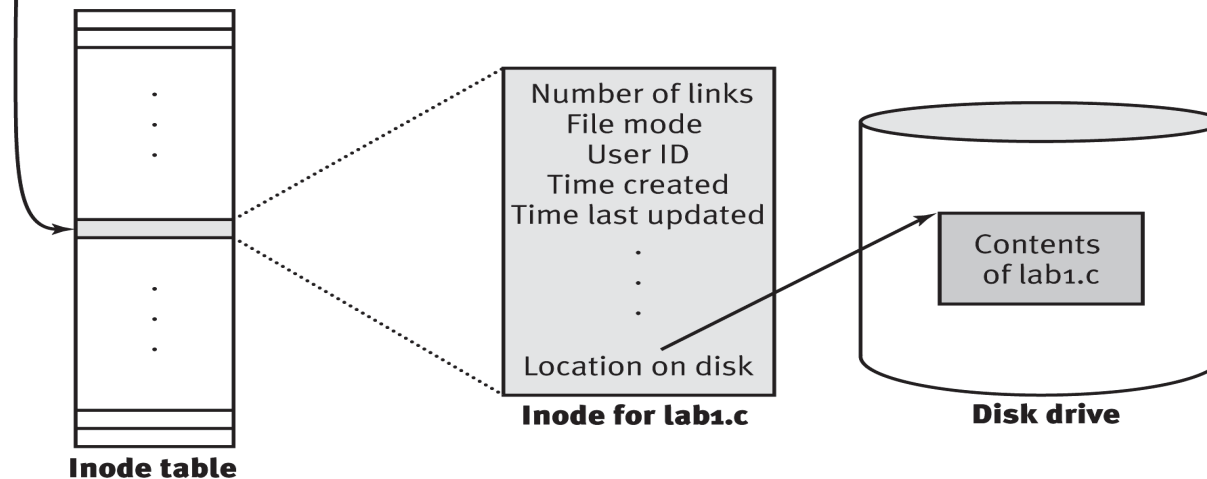
# I/O Systems

## ■ The UNIX File System

**Contents of the directory**  
**~/courses/ee446/labs**

1076	.
2083	..
13059	lab1.c
17488	lab2.c
18995	lab3.c

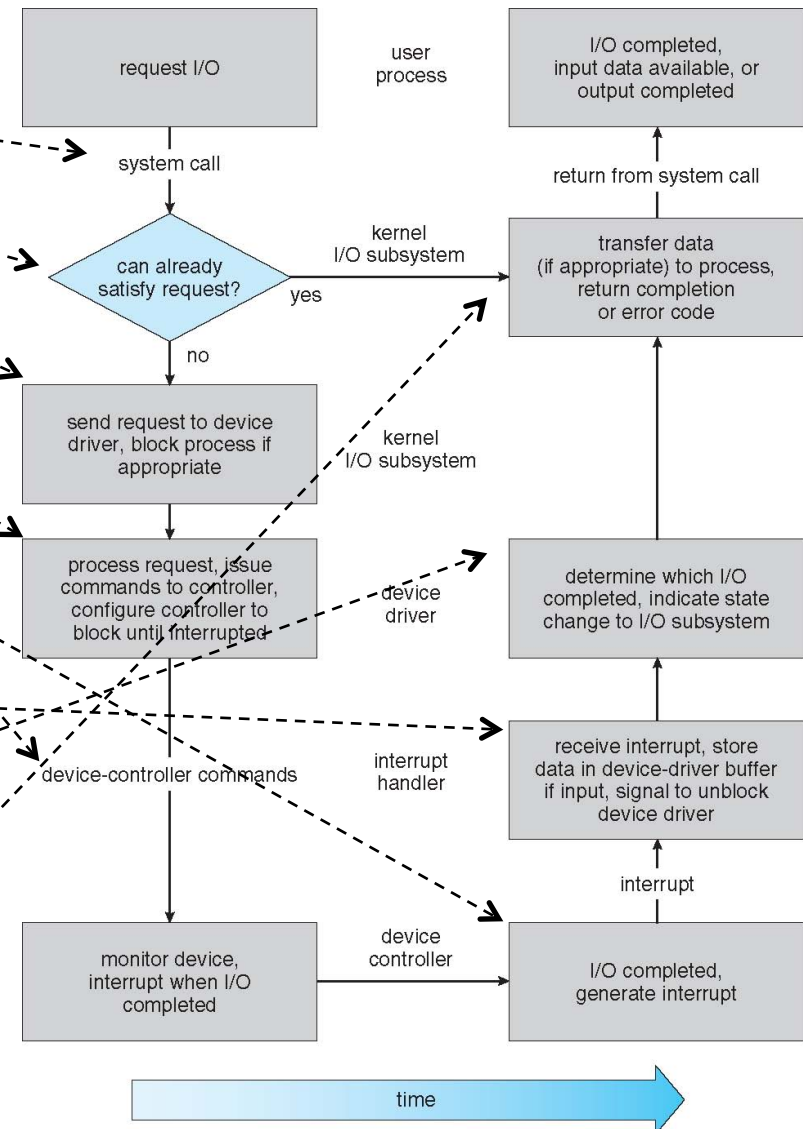
- 1) Directory contains array of entries <inode #, filename>.
- 2) Entry placed in Inode Table in RAM when file is opened.
- 3) Indexing into Inode Table returns the entry of the Inode containing the block location of file on disk.



# I/O Systems

## Life Cycle of I/O Request

- 1) Process issues Blocking read () system call to file descriptor of opened file.
- 2) System-call in Kernel checks if data already in buffer cache.
- 3) Physical I/O performed. Process removed from Run Queue and placed in Wait Queue for the device and I/O Request scheduled.
- 4) Kernel buffers allocated and schedules I/O.
- 5) Device Controller perform data transfer.
- 6) Driver poll for status and data or sets up DMA transfer. When managed by DMA controller, interrupt generated when transfer completes.
- 7) Interrupt Handler receives interrupt (Interrupt-Vector Table), stores data, signals Device Driver.
- 8) Device Driver determines which I/O Request has completed, signals I/O Subsystem.
- 9) Kernel transfer data or return codes to the address space of the requesting process. Process moved from Wait Queue to the Ready Queue.



# I/O Systems

## ■ STREAMS

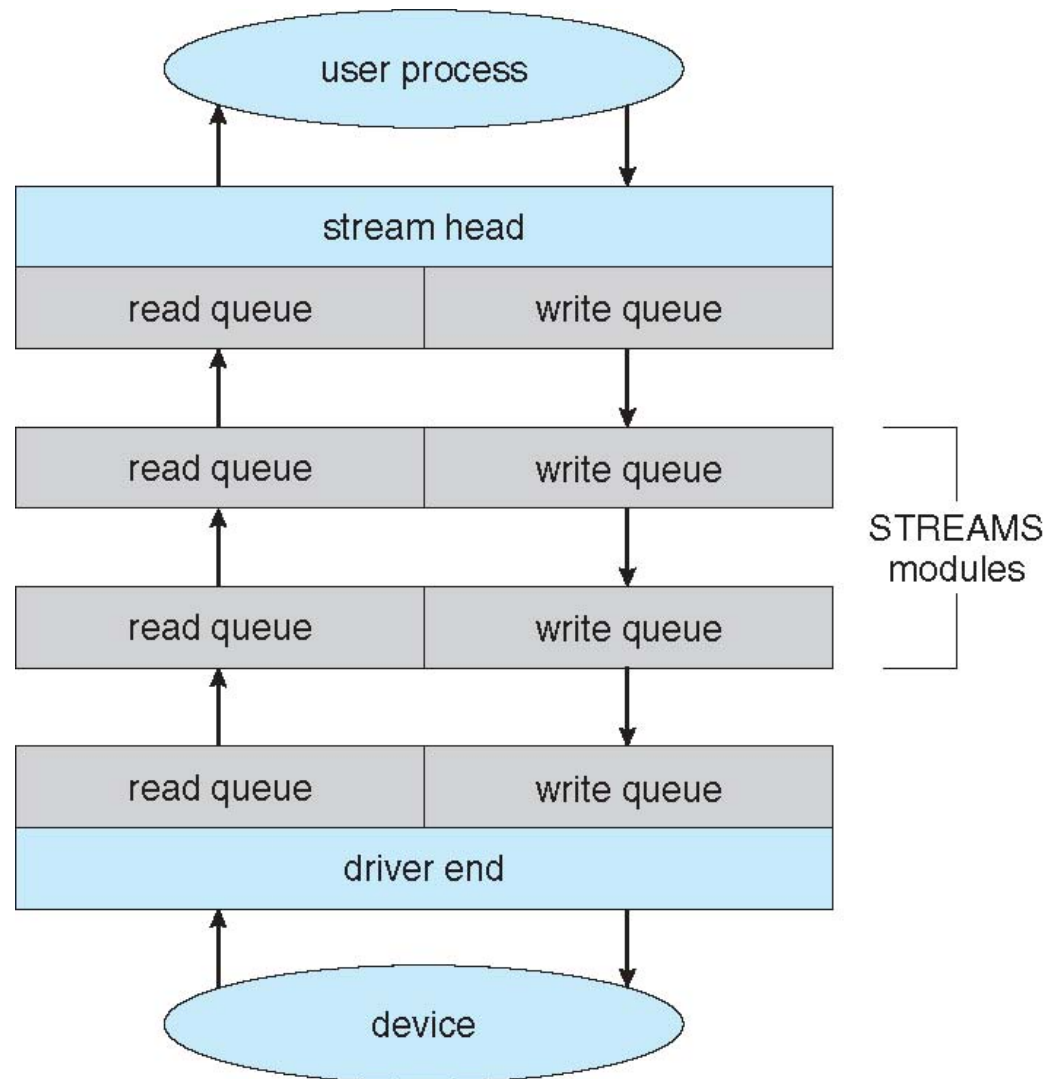
- **STREAM** – In UNIX System V, a full-duplex communication channel between a user-level process and a Device Driver.
  - Intended to support different kinds of character-based I/O.
- A STREAM consists of:
  - STREAM Head interfaces with the User Process.
  - STREAM End interfaces with the Device Driver.
  - Zero or more STREAM modules between them.
- Each module contains a **read queue** and a **write queue**.
- Message is used to communicate between queues.
- Modules are “pushed” onto a stream or “popped” from a stream by `ioctl ( )`.
- User process writes data to device: `write ( )` or `putmsg ( )`.
- User process reads data from device: `read ( )` or `getmsg ( )`.

# I/O Systems

## ■ STREAMS

## ■ STREAMS

### Structure



# I/O Systems



## ■ Performance

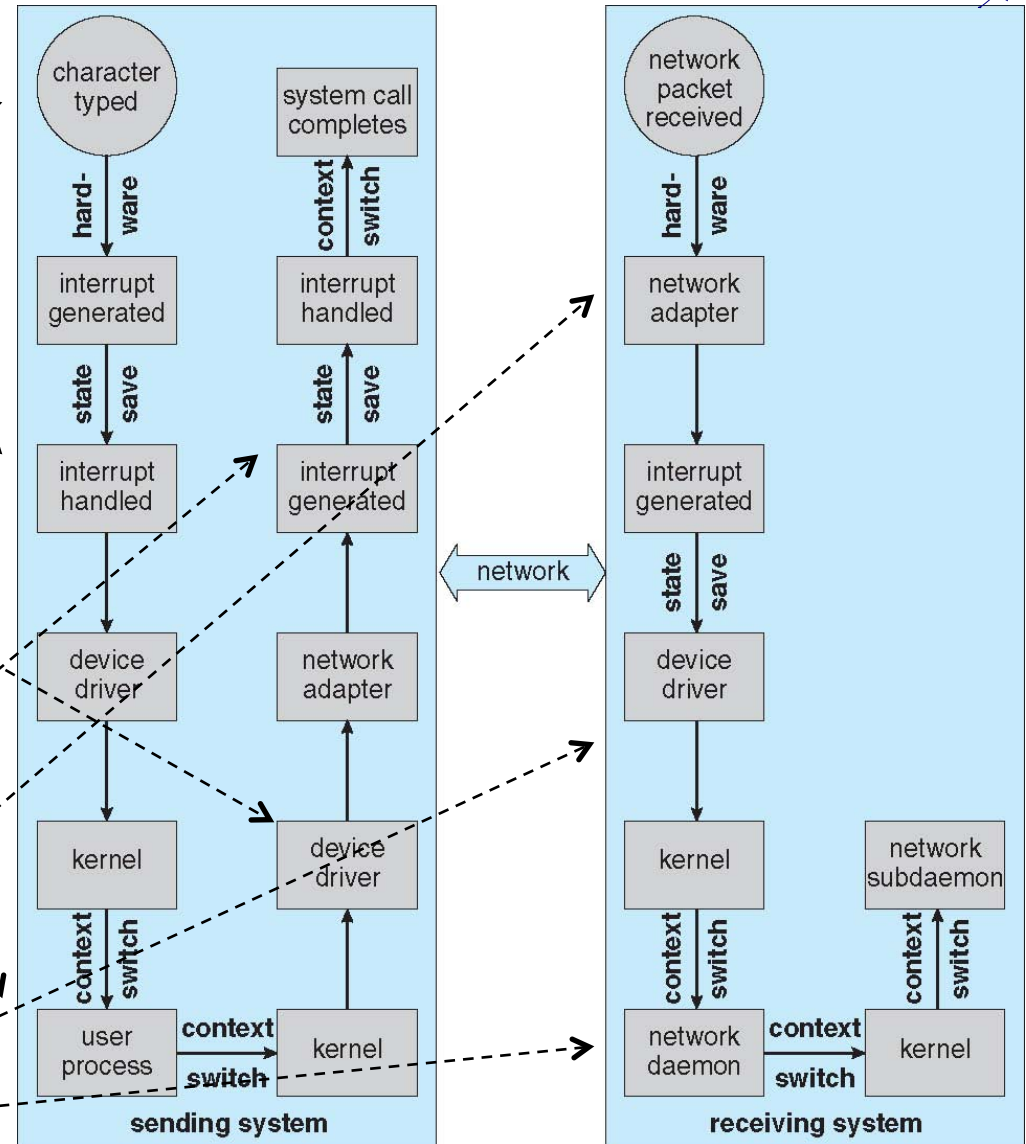
### ■ I/O a major factor in system performance:

- Demands on the CPU to execute Device Driver.
- Schedule processes fairly and efficiently.
- Context switches, due to interrupts, stress CPU and hardware caches.
- I/O exposes inefficiencies in Kernel Interrupt Handlers.
  - Interrupt causes state change, executes Interrupt Handler, and restore state.
  - Staying in critical section (interrupt disabled) too long.
  - Programmed I/O can be more efficient than interrupt driven I/O, when busy-wait cycles not excessive.
- Memory bus unavailable to Kernel during data copies between controllers and physical memory.
- Network traffic especially stressful, causing high context switch rate.

# I/O Systems

## ■ Performance

- 1) Local user types character.
- 2) Keyboard Interrupt Handler calls Device Driver and user process is awoken.
- 3) User process issues Network I/O System Call.
- 4) Kernel Network Layer receives packet and builds network packet for Network Device Driver.
- 5) Network Device Driver sends packet to Network Controller .
- 6) Network Controller generates local interrupt to complete System Call.
- 7) Remote system Network Controller rcvs packet, generates interrupt.
- 8) Network Dev Drv passes pkt to kernel to Net Dmn.



# I/O Systems



## ■ Performance

## ■ Improve Efficiency of I/O

- ☐ Reduce number of context switches.
- ☐ Reduce data copying.
- ☐ Reduce interrupts by using large transfers, smart controllers, polling (with minimized busy waiting).
- ☐ Concurrent CPU and I/O transfer using DMA-knowledgeable controllers.
- ☐ I/O Channel with dedicated CPU to buffer data while CPU running processes (mainframes).
- ☐ Balance CPU, memory, bus, and I/O performance for highest throughput.

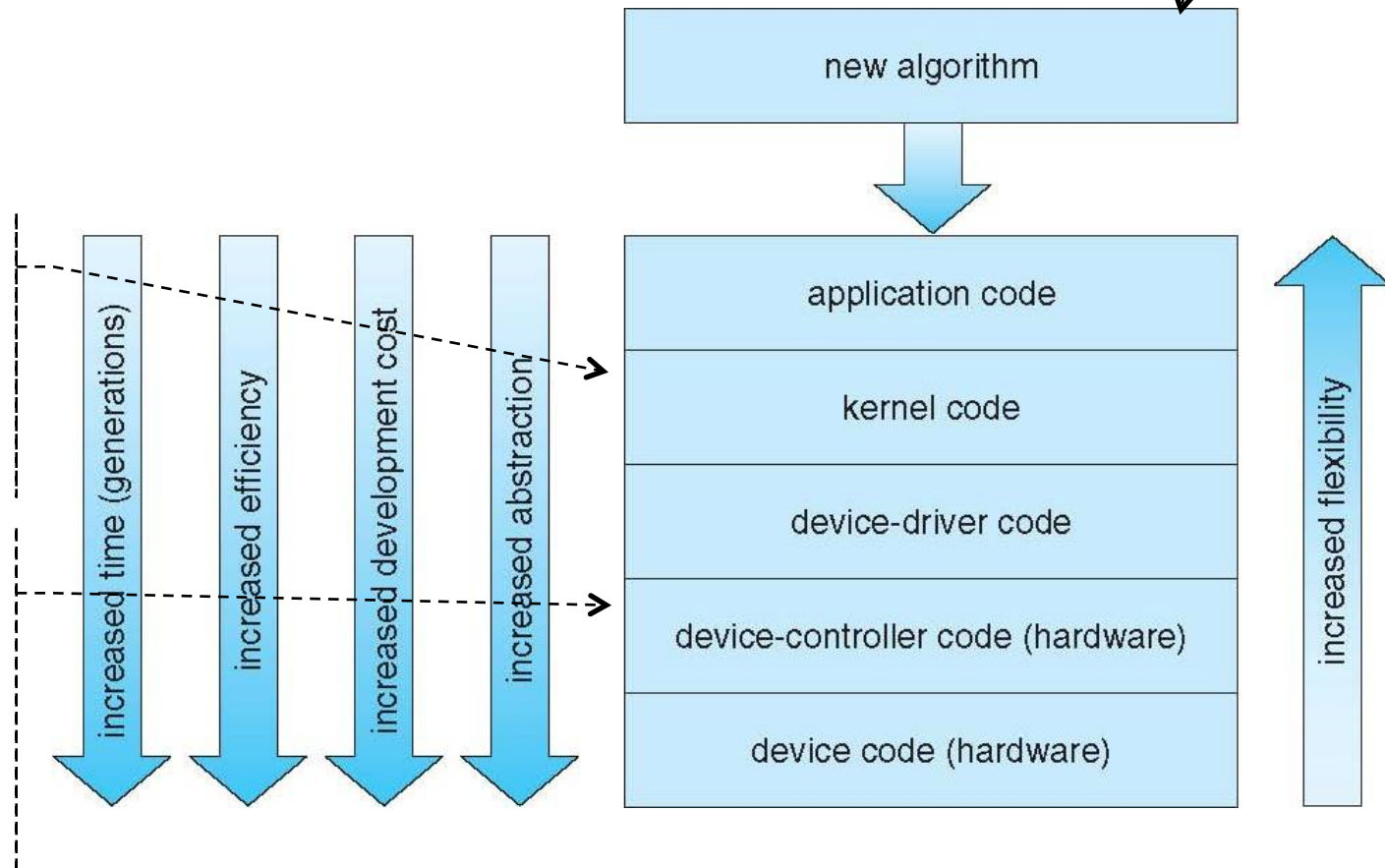
# I/O Systems

- Performance
- Improve Efficiency of I/O

I/O Algorithms starts at the application level, where flexibility is high, but inefficient. Overhead of context switches and cannot access Kernel data structures and Kernel functionality.

Code is moved to the Kernel for performance and priority. More complexity in the Kernel and code has to be stable.

Highest performance would require specialized hardware. Hardware changes time consuming (months instead of days) and minimum flexibility.





# I/O Systems

## ■ Summary

## ■ Common Concepts for I/O:

- Basic Hardware Elements: Port, Bus, I/O Device Controller.
- Programmed I/O for moving data between Devices and Main Memory.
  - Polling or Interrupt.
  - Interrupt Request Line and Interrupt Handlers.
- DMA Controllers for large data block transfers.

## ■ Device Driver: Kernel Module that controls a Device.

- System Call interface hides different categories of hardware:
  - Block Drivers, Character Drivers, Memory-Mapped Files, Network Sockets, Programmed Timers.
  - Nonblocking (used by applications and Kernel), and Blocking (used by applications) System Calls.

# I/O Systems

[end on June 23rd]

- Summary
- I/O Subsystem coordinates services to applications via uniform interface provided by Device Drivers.
  - Access Control to Files and Devices.
  - File-System Space Allocation.
  - Buffering, Caching, Spooling.
  - I/O Scheduling.
  - Device-Status Monitoring, Error Handling, and Failure Recovery.
- I/O Subsystem Name Translation.
  - Maps full path file names to Device Drivers and Device Addresses (UNIX).
  - Maps with explicit device (MS-DOS).
- STREAMS is framework for Device Drivers and Network Protocols.

# I/O Systems

- Summary
- I/O System Calls has Overhead affecting efficiency.
- Software Layers between application and physical Device.
  - Context Switching, crossing Kernel protective boundary.
  - Signal and interrupt handling to service Device.
  - CPU and Memory System load caused by copying data between Kernel buffers and application buffers.