CS343 - Operating Systems

Module-7A

Introduction to Input/Output Subsystem

Dr. John Jose

Assistant Professor

Department of Computer Science & Engineering

Indian Institute of Technology Guwahati, Assam.

http://www.iitg.ac.in/johnjose/

Overview of I/O Subsystem Management

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

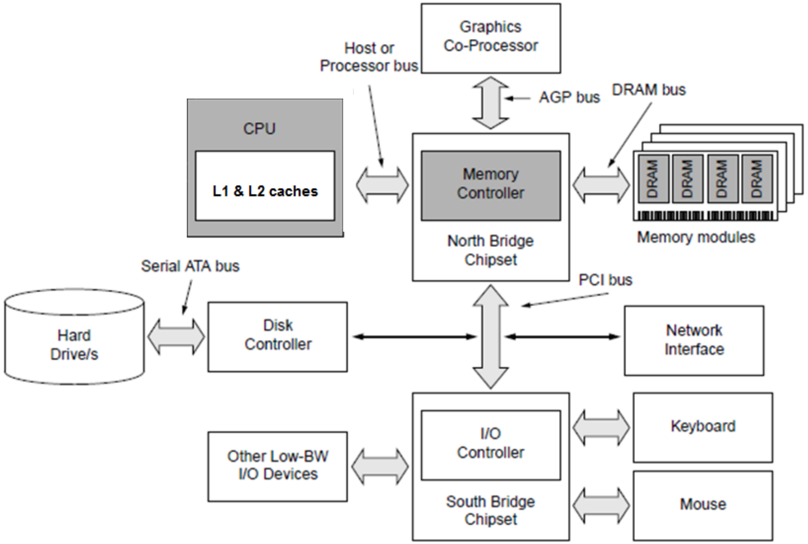
Importance and Challenges in I/O Management

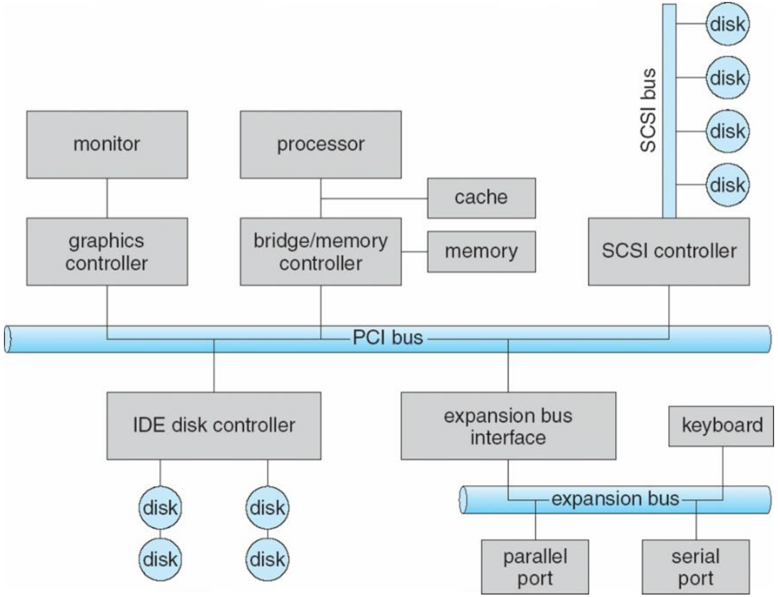
- I/O management is a major component of operating system design and operation
  - Important aspect of computer operation
  - I/O devices vary greatly
  - Various methods to control them
  - Performance management
  - New types of devices frequent
- Ports, busses, device controllers connect to various devices
- **Device drivers** encapsulate device details
  - Present uniform device-access interface to I/O subsystem

I/O Hardware

- Incredible variety of I/O devices
  - Storage
  - Transmission
  - Human-interface

PCI Bus Structure



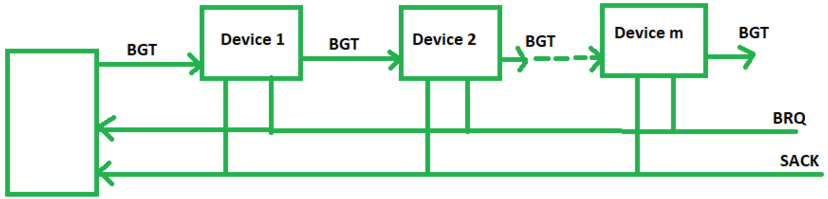


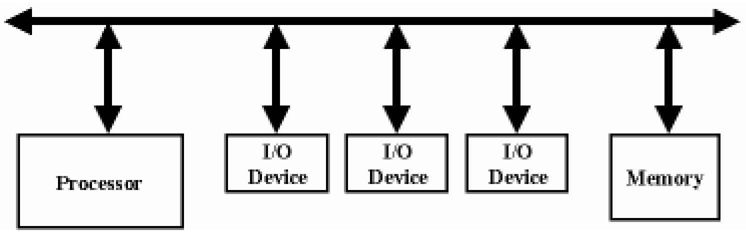
Components of I/O Subsystem

- I/O Hardware
  - ports, buses, devices, controllers
- I/O Software
  - Interrupt Handlers, Device Driver,
  - Device-Independent Software,
  - User-Space I/O Software
- I/O Data transfer mechanisms
  - Polling, Interrupt and DMAs

I/O Hardware

- Signals from I/O devices interface with computer
  - **Port** – connection point for device
  - **Bus** - **daisy chain** or shared direct access
    - **PCI** bus common in PCs and servers, PCI Express (**PCIe**)
    - **expansion bus** connects relatively slow devices





I/O Hardware

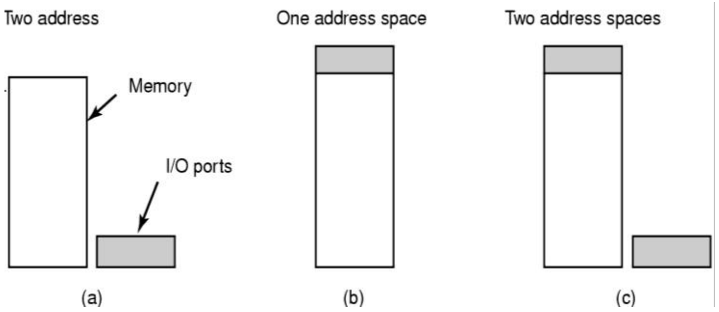
- **Controller** (**host adapter**) – electronics that operate port, bus, device
  - Sometimes integrated
  - Sometimes separate circuit board (host adapter)
  - Contains processor, microcode, private memory, bus controller, etc

I/O Hardware

- I/O instructions control devices
- Devices usually have registers where device driver places commands, addresses, and data to write, or read data from registers after command execution
  - Data-in register, data-out register, status register, control register
  - Typically 1-4 bytes, or FIFO buffer
- Devices have addresses, used by I/O instructions

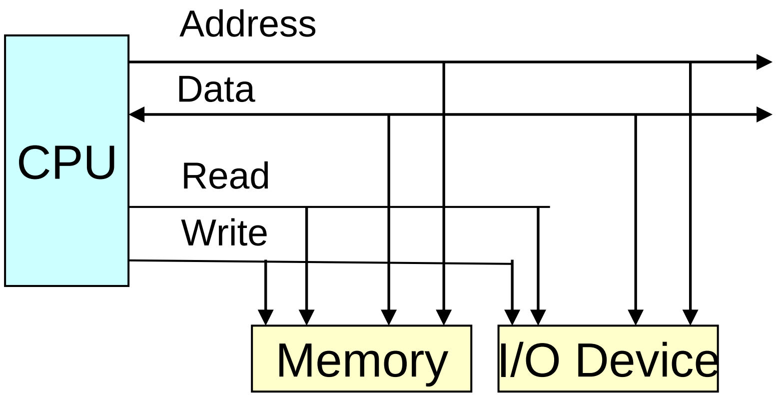
I/O Mapping

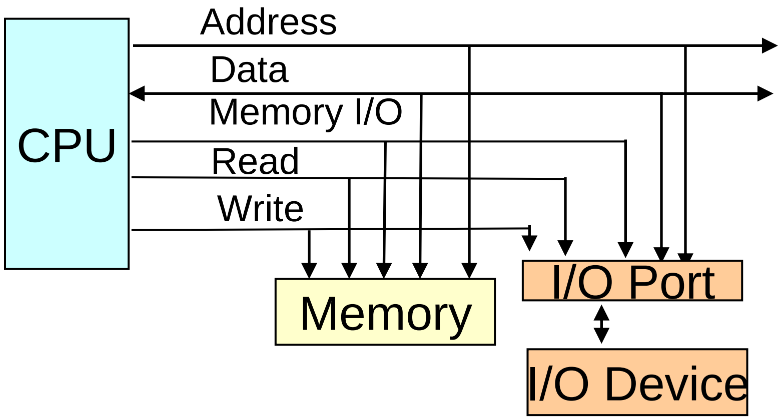
- Memory mapped I/O
  - Devices and memory share an address space
  - I/O looks just like memory read/write
  - No special commands for I/O
  - Large selection of memory access commands available
- Isolated I/O (I/O mapped I/O)
  - Separate address spaces
  - Need I/O or memory select lines
  - Special commands for I/O; Limited set



I/O Mapping

- CPU needs to talk to I/O
- **Memory mapped I/O**
  - Devices mapped to reserved memory locations - like RAM
  - Uses load/store instructions just like accesses to memory
- **I/O mapped I/O**
  - Special bus line
  - Special instructions





I/O Basics

- I/O module interface I/O to CPU and Memory
- **I/O controller ←→ I/O devices ports**
  - Transfers data to/from device
  - Synchronizes operations with software
- **Status/ control registers**: device status, errors
- **Data registers**
  - Write: CPU/RAM data ➔ device [eg Transmit]
  - Read: CPU🡸 device [eg Receive]

Functions of I/O Module

- Control & Timing
- Processor Communication
- Device Communication
- Data Buffering
- Error Detection (e.g., extra parity bit)

Basic I/O Steps

- CPU checks I/O module device status
- I/O module returns status
- If ready, CPU requests data transfer by sending a command to the I/O module
- I/O module gets a unit of data (byte, word, etc.) from device
- I/O module transfers data to CPU
- Variations of these steps for different I/O mechanisms like poling, interrupt and DMA based I/O.

Thank you

johnjose@iitg.ac.in
http://www.iitg.ac.in/johnjose/

CS343 - Operating Systems

Module-7B

Data Transfer Techniques in I/O Subsystem

Dr. John Jose

Assistant Professor

Department of Computer Science & Engineering

Indian Institute of Technology Guwahati, Assam.

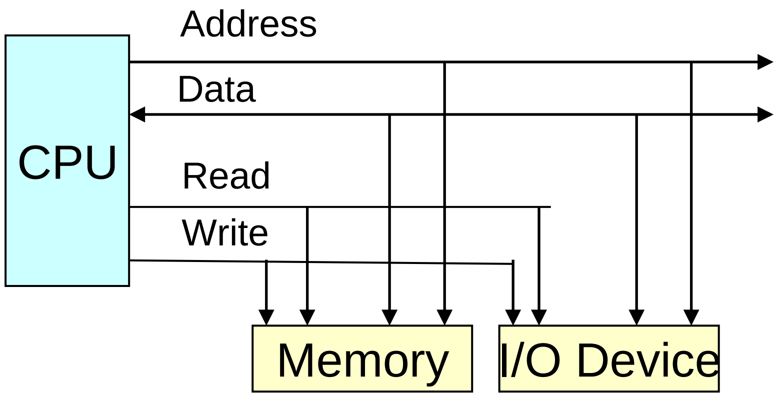
http://www.iitg.ac.in/johnjose/

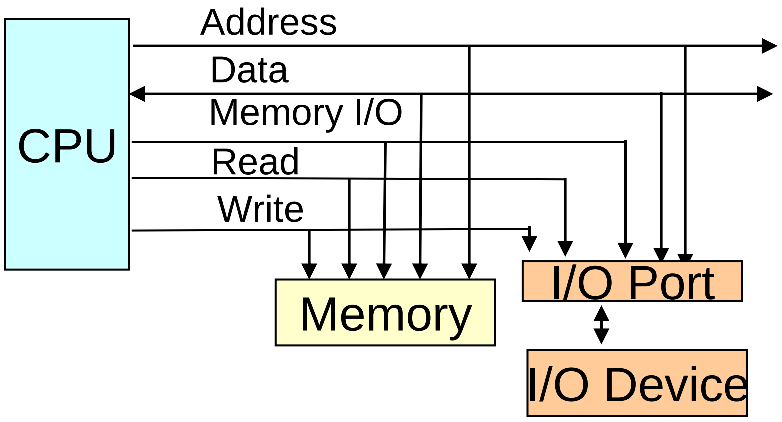
Objectives

- 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 various data transfer schemes

I/O Mapping

- CPU needs to talk to I/O
- **Memory mapped I/O**
  - Devices mapped to reserved memory locations - like RAM
  - Uses load/store instructions just like accesses to memory
- **I/O mapped I/O**
  - Special bus line
  - Special instructions





I/O Data Transfer techniques

- Polled I/O
- Interrupt-Driven I/O
- Direct Memory Access (DMA)

Polled I/O

- CPU periodically check I/O status (polling)
  - If device ready, do operation
  - If error, take action
- CPU has direct control over I/O
  - Sensing status
  - Read/write commands
  - Transferring data
- CPU waits for I/O module to complete operation
- Wastes CPU time

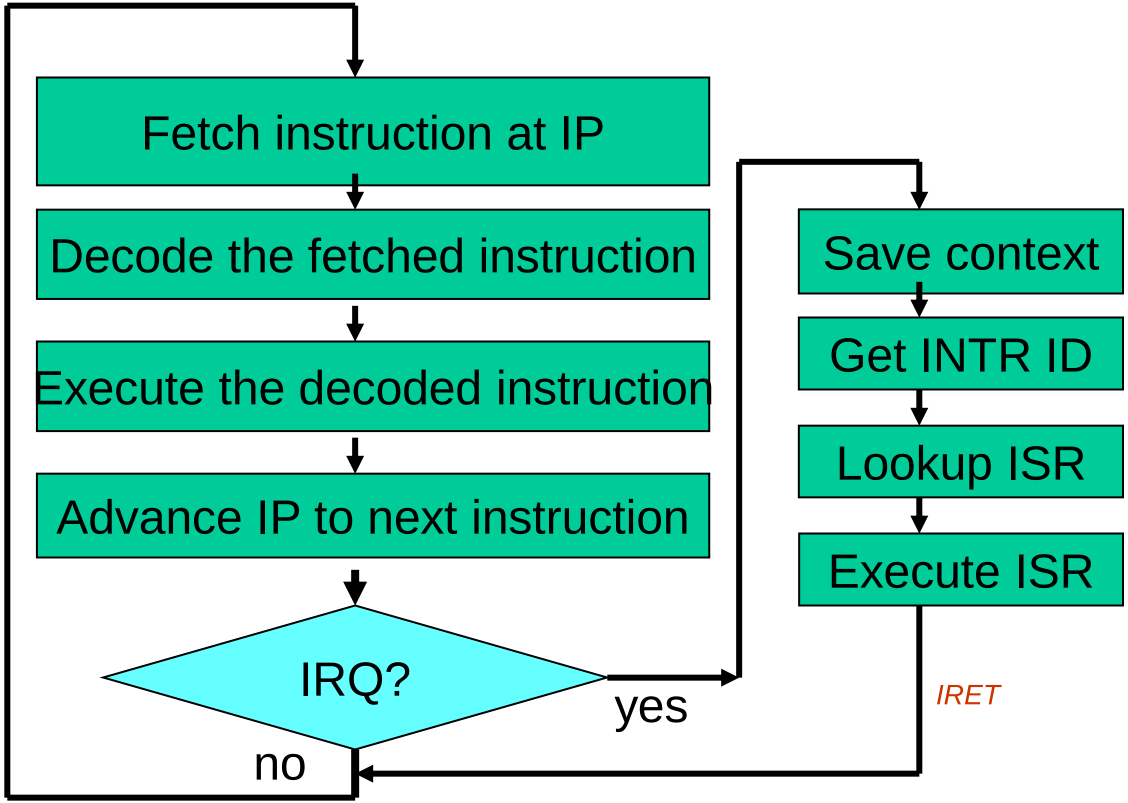
Steps in Polled I/O

- CPU requests I/O operation
- I/O module performs operation
- I/O module sets status bits
- CPU checks status bits periodically (polling)
- CPU may wait or come back later
- I/O module does not inform CPU directly
- I/O module does not interrupt CPU

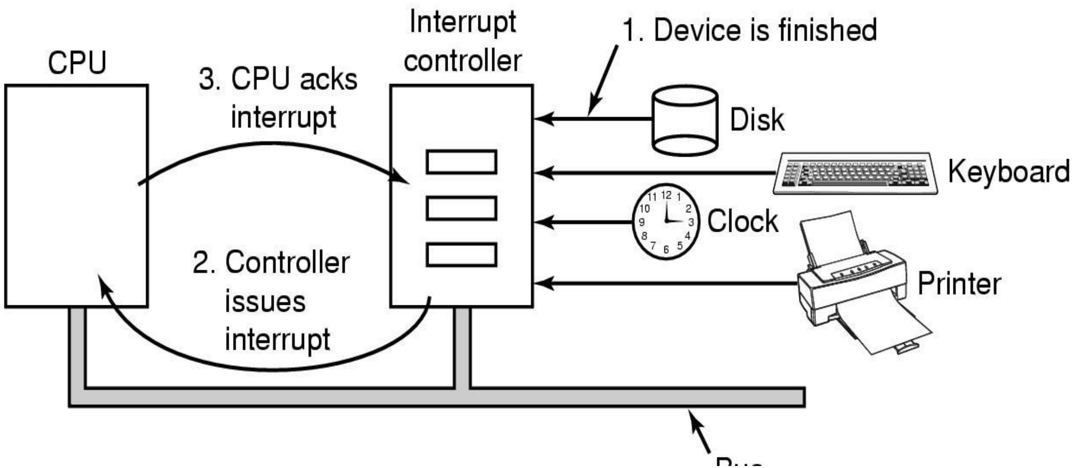
Interrupts

- Polling can happen in 3 instruction cycles
  - Read status, extract status bit, branch if status is shows done.
  - How to be more efficient if status is done infrequently?
- CPU **Interrupt-request line** triggered by I/O device
  - Checked by processor after each instruction
- **Interrupt handler** receives interrupts
  - **Maskable** to ignore or delay some interrupts

Interrupt Service Routine



Interrupt Driven I/O

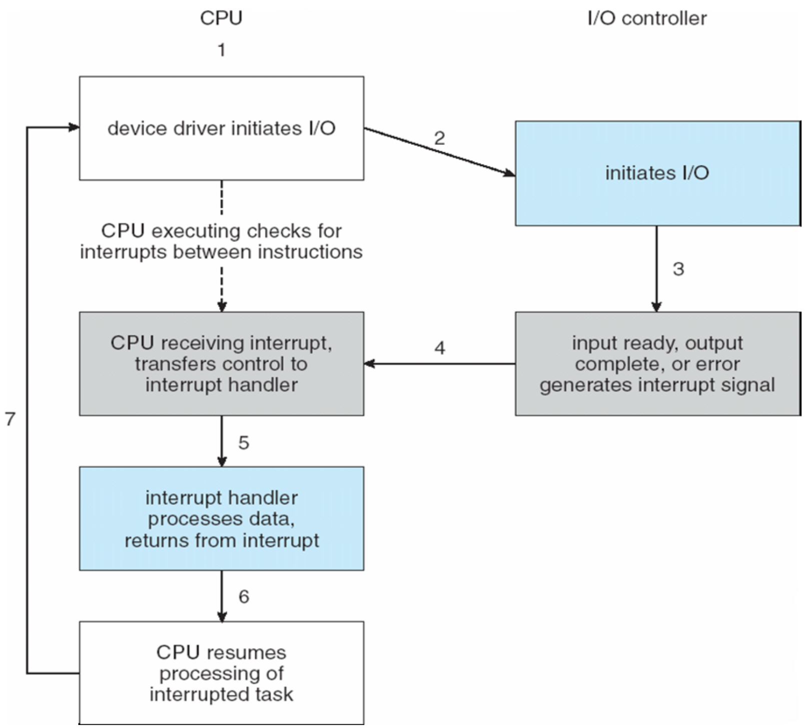


- ❖ I/O module gets data from peripheral while CPU continues other work.
- ❖ I/O module interrupts CPU when data is ready.
- ❖ CPU requests data
- ❖ I/O module transfers data

Interrupt Driven I/O

- I/O device issues an interrupt to indicate that it needs attention of CPU
- Interrupts are special signals initialed by I/O devices to catch the attention of the processor.
- Overcomes CPU waiting
- No repeated CPU checking of device
- An I/O interrupt is asynchronous w.r.t. instruction execution
- Is not associated with any instruction so doesn’t prevent any instruction from completing

Interrupt-Driven I/O Cycle



Interrupt Driven I/O

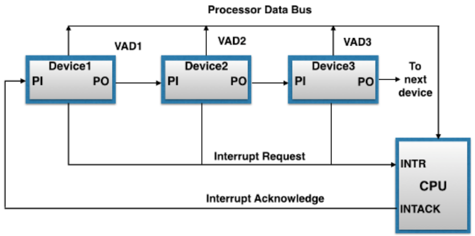
- Advantages
  - Relieves the processor from having to continuously poll for an I/O event; user program progress is only suspended during the actual transfer of I/O data to/from user memory space
- Disadvantages
  - Special hardware is needed to indicate the I/O device causing the interrupt and to save the necessary information prior to servicing the interrupt and to resume normal processing after servicing the interrupt

Challenges in Interrupt Driven I/O

- How do you identify the module issuing the interrupt?
  - Need a way to identify the device generating the interrupt
- How do you deal with multiple interrupts?
  - Can have different urgencies (so need a way to prioritize them)

Identifying Interrupting Module

- CPU asks each module in turn (Slow) Daisy Chain or Hardware poll
  - Interrupt Acknowledge sent down a chain
  - Module responsible places vector on bus
  - CPU uses vector to identify handler routine



- Vectored Interrupt
  - **Interrupt vector** to dispatch interrupt to correct handler

Ex: Intel Pentium Processor Event-Vector Table

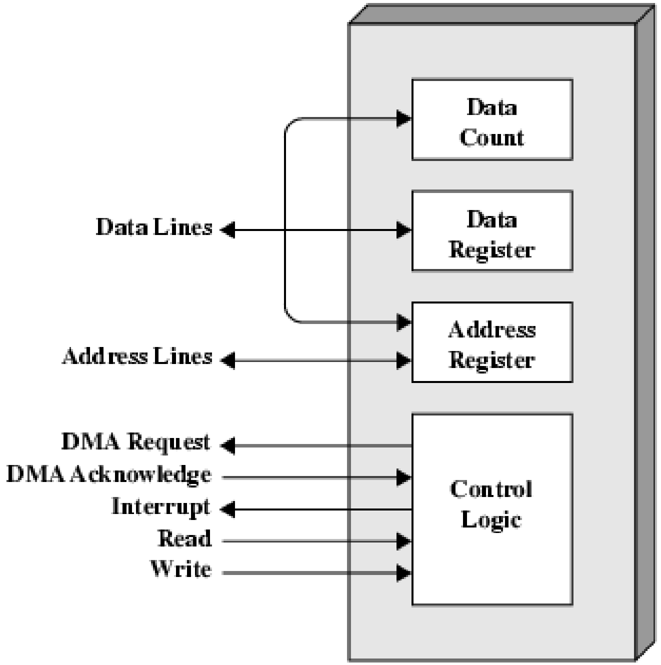
vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
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

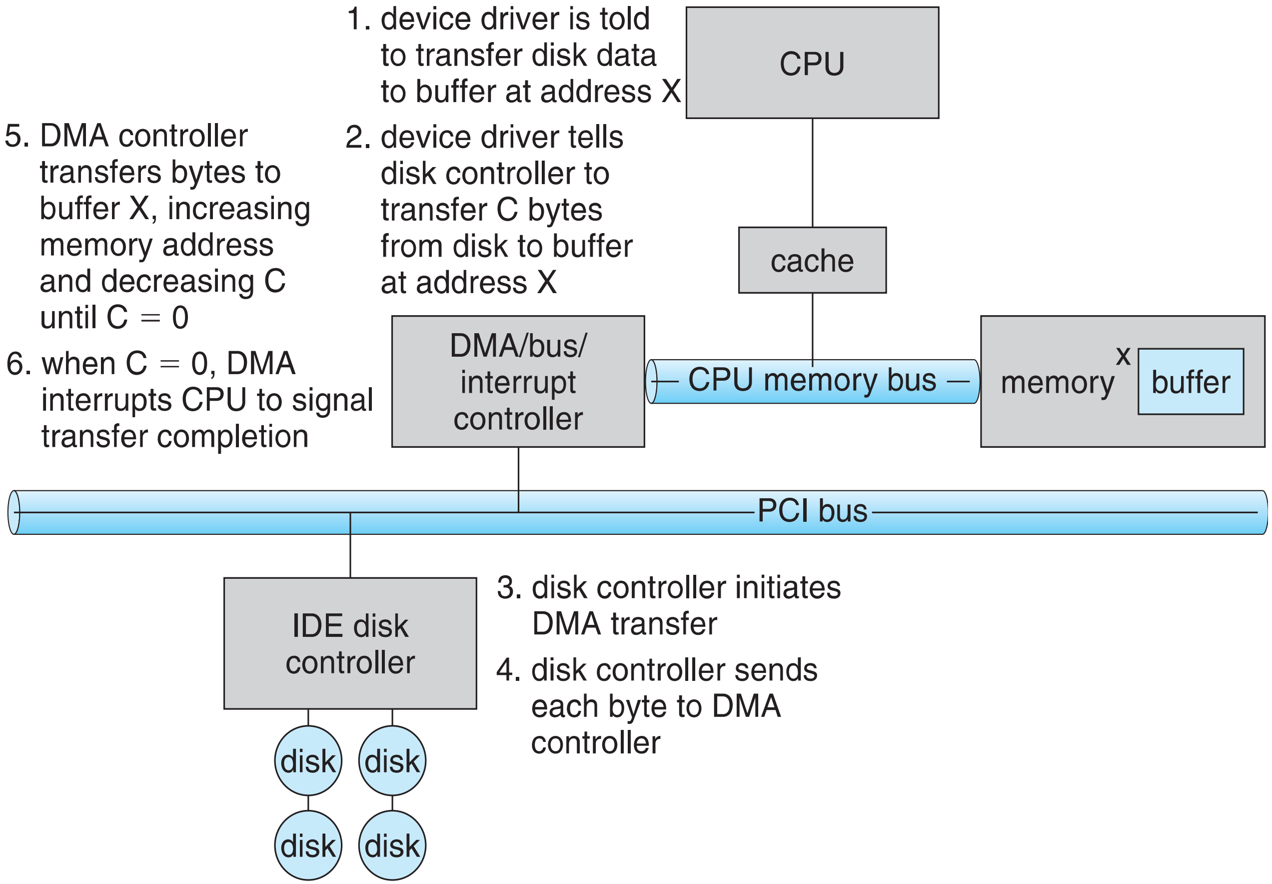
- Interrupt driven and programmed I/O require active CPU intervention
- For high-bandwidth devices (like disks) interrupt-driven I/O would consume a lot of processor cycles
- Bypasses CPU to transfer data directly between I/O device and memory
- OS writes DMA command block into memory
    - Source and destination addresses, Read or write mode
    - Count of bytes, Writes location of command block to DMA controller
- DMA is an additional module (hardware) on bus
- DMA controller takes over from CPU for I/O operations

DMA Module Diagram

- CPU tells DMA controller:-
    - Read/Write
    - Device address
    - Starting address of memory block for data
    - Amount of data to be transferred
- CPU carries on with other work
- DMA controller deals with transfer
- DMA controller sends interrupt when finished



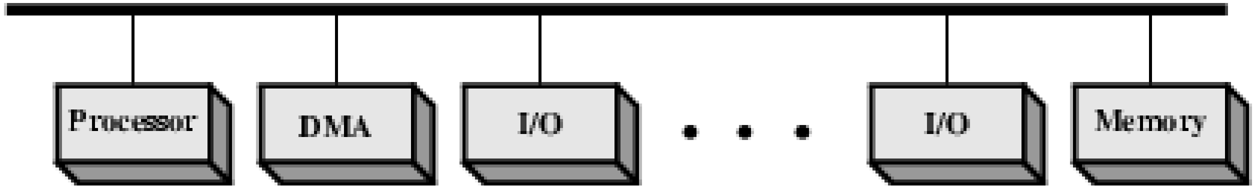
Disk to Memory Copy via DMA



Modes of DMA operation

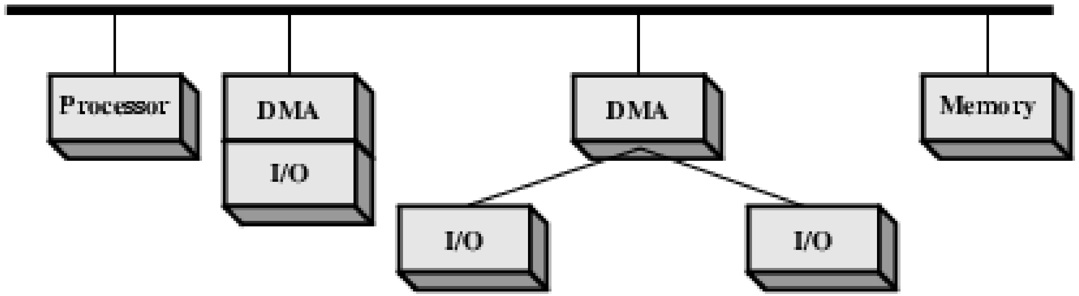
- **Cycle Stealing**
  - DMA controller acquires control of bus
  - Transfers a single word and releases the bus
  - The CPU is slowed down due to bus contention
  - Responsive but not very efficient
- **Burst Mode**
  - DMA Controller acquires control of bus
  - Transfers all the data and then only releases the bus
  - The CPU is suspended or works with cache
  - Efficient but interrupts may not be serviced in a timely way

DMA Configurations



- Single Bus, Detached DMA controller
- Each transfer uses bus twice
    - I/O to DMA then DMA to memory
- CPU is suspended twice

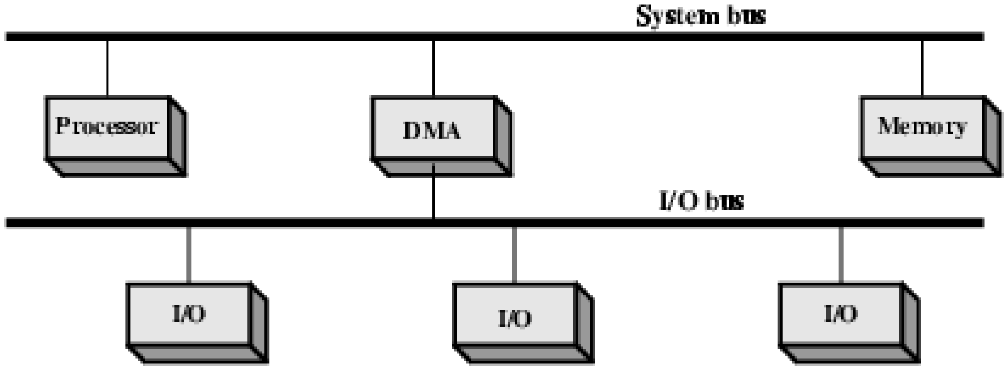
DMA Configurations



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

- Single Bus, Integrated DMA controller
- Controller may support >1 device
- Each transfer uses bus once
  - DMA to memory
- CPU is suspended once

DMA Configurations



(c) I/O bus

- Separate I/O Bus
- Bus supports all DMA enabled devices
- Each transfer uses bus once
  - DMA to memory
- CPU is suspended once

Thank you

johnjose@iitg.ac.in
http://www.iitg.ac.in/johnjose/

CS343 - Operating Systems

Module-7C

I/O Device Classifications and Performance

Dr. John Jose

Assistant Professor

Department of Computer Science & Engineering

Indian Institute of Technology Guwahati, Assam.

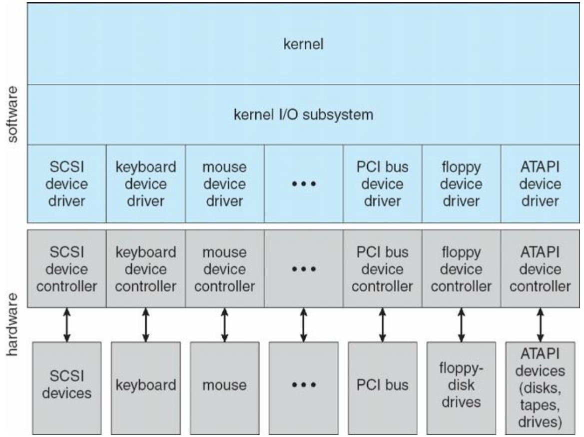
http://www.iitg.ac.in/johnjose/

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 various types of I/O devices

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
- New devices use already-implemented protocols. No extra work.
- Each OS has its own I/O subsystem structures and device driver frameworks



Kernel I/O Structure

Category of I/O Devices

- 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**

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

Characteristics of I/O Devices

- Broadly I/O devices can be grouped by the OS into
  - Block I/O
  - Character I/O (Stream)
  - Memory-mapped file access
  - Network sockets

Block and Character Devices

- Block devices include disk drives
  - Commands include read, write, seek
  - Raw I/O, direct I/O, or file-system access
  - File mapped to virtual memory and brought via demand paging
  - DMA
- Character devices include keyboards, mice, serial ports
  - Commands include **get()**, **put()**
  - Buffering and editing services

Network Devices

- Linux, Unix, Windows and many others include **socket** interface
  - Separates network protocol from network operation
  - Includes **select()** functionality to choose sockets for send and receive
- Pipes, FIFOs, streams, queues, mailboxes

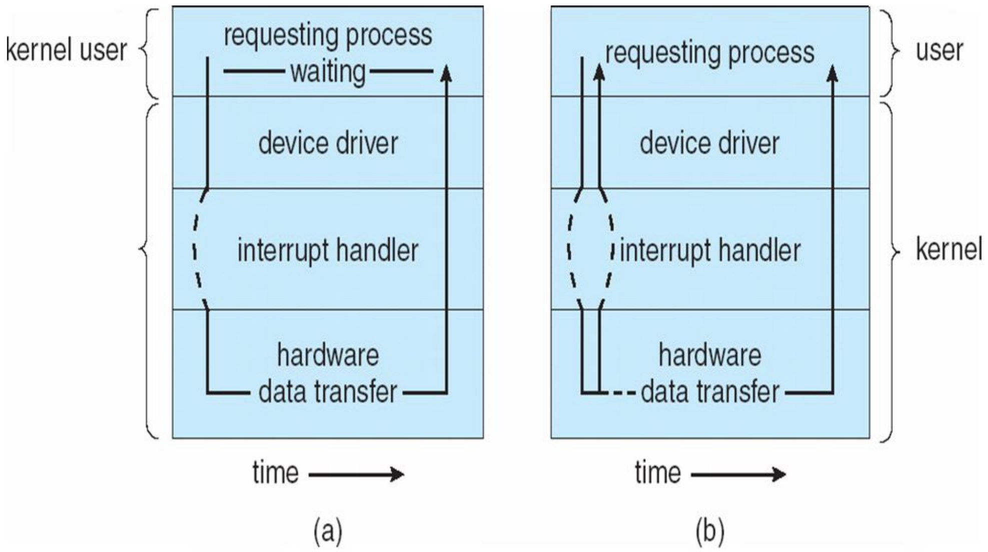
Clocks and Timers

- Provide current time, elapsed time, set a timer to trigger ***X*** and ***T***
- Normal resolution about 1/60 second
- Some systems provide higher-resolution timers
- Programmable interval timer used for timings, periodic interrupts

Nonblocking and Asynchronous I/O

- **Blocking** - process suspended until I/O completed
  - Application from running to waiting and to ready states
- **Nonblocking** - I/O call returns as much as available I/O data
  - User interface, data copy (buffered I/O)
  - Implemented via multi-threading
  - Returns quickly with count of bytes read or written
  - **select()** to find if data ready then **read()** or **write()** to transfer
- **Asynchronous** - process runs while I/O executes
  - Difficult to use
  - I/O subsystem signals process when I/O completed

Two I/O Methods



Synchronous Asynchronous

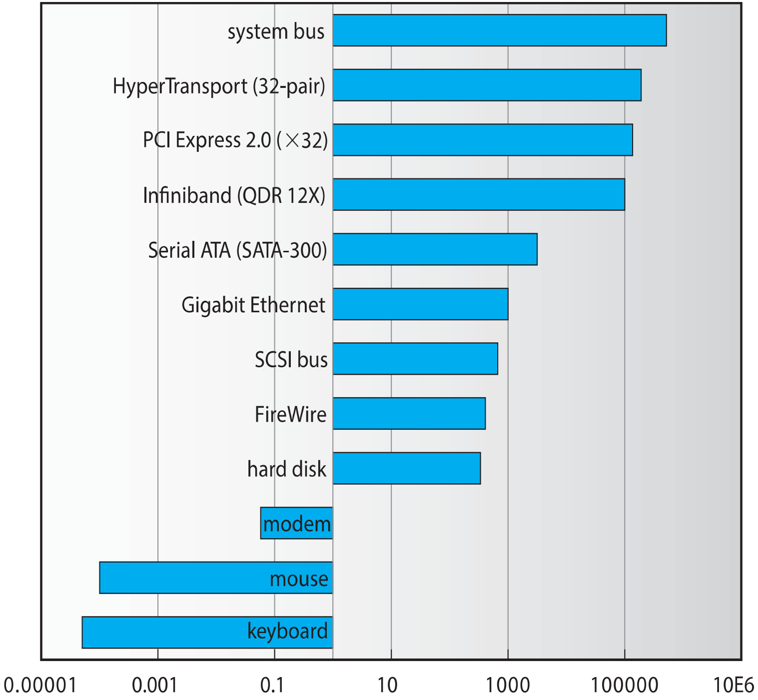
Kernel I/O Subsystem

- ❖ **Scheduling**
  - ❖ Some I/O request ordering via per-device queue
  - ❖ Some OSs try fairness
  - ❖ Some implement Quality Of Service

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
- **Double buffering** – two copies of the data
  - Kernel and user
  - Varying sizes
  - Full / being processed and not-full / being used
  - Copy-on-write can be used for efficiency in some cases

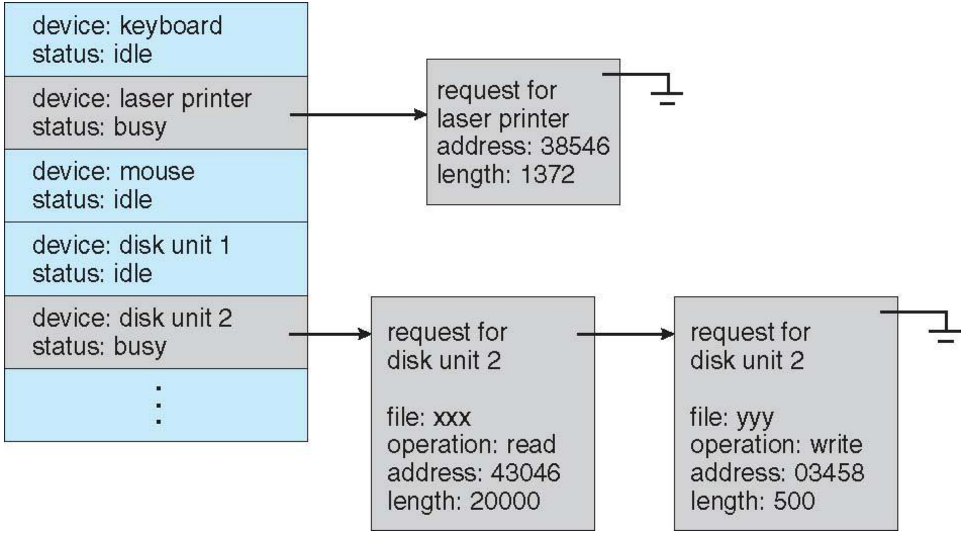
Sun Enterprise 6000 Device-Transfer Rates



Kernel I/O Subsystem

- **Caching** - faster device holding copy of data
  - Always just a copy
  - Key to performance
  - Sometimes combined with buffering
- **Spooling** - hold 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 de-allocation
  - Watch out for deadlock

Device-Status Table

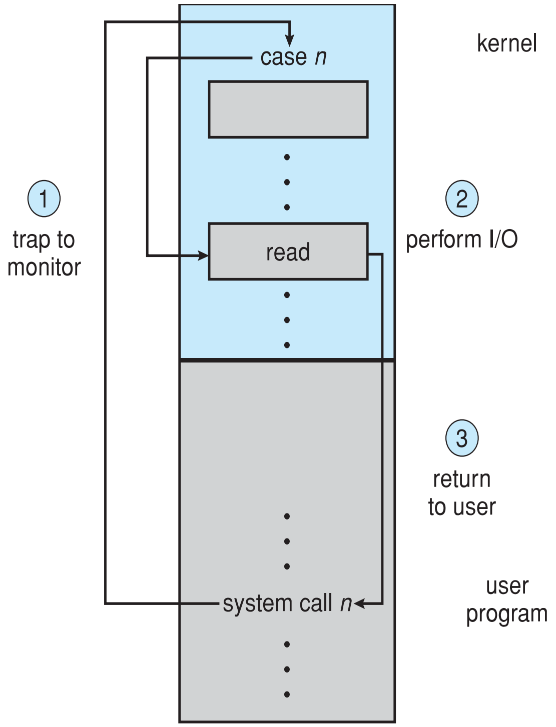


Error Handling

- OS can recover from disk read, device unavailable, transient write failures
  - Retry a read or write, for example
  - Some systems more advanced – Solaris FMA, AIX
    - Track error frequencies, stop using device with increasing frequency of retry-able errors
- Most return an error number or code when I/O request fails
- System error logs hold problem reports

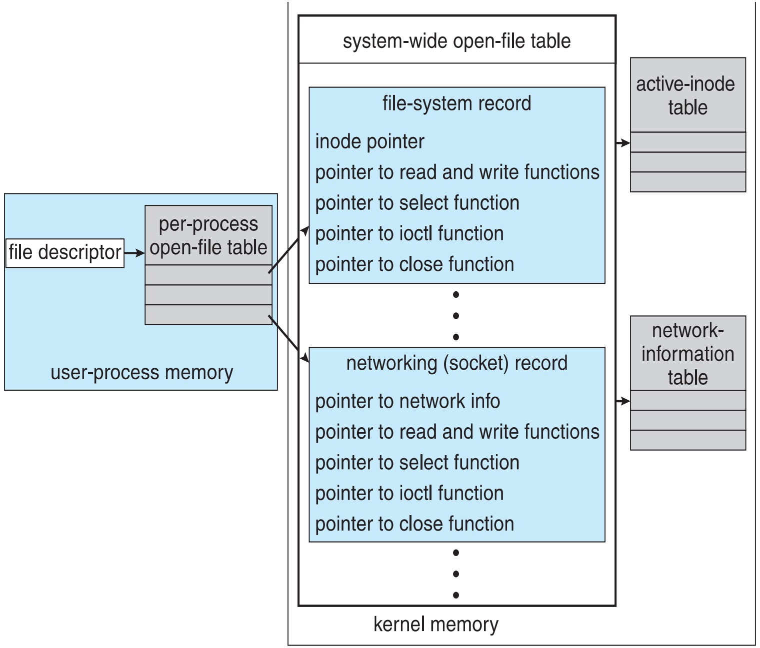
System Call to Perform I/O

- All I/O instructions defined to be privileged
- I/O must be performed via system calls



Kernel Data Structures

- Kernel keeps state information for I/O components, including open file tables, network connections, character device state



Power Management

- Not strictly domain of I/O, but much is I/O related
- Computers and devices use electricity, generate heat, frequently require cooling
- OS can help manage and improve use
- Mobile computing has power management

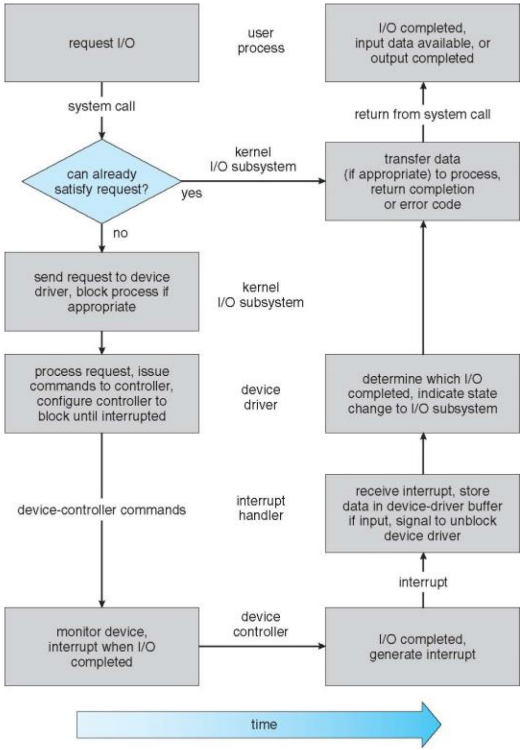
Power Management on Mobile Platforms

- Understands relationship between components
- Build device tree representing physical device topology
- System bus -> I/O subsystem -> {flash, USB storage}
- Device driver tracks state of device, whether in use
- Unused component – turn it off
- All devices in tree branch unused – turn off branch
- Wake locks – like other locks but prevent sleep of device when lock is held
- Power collapse – put a device into very deep sleep
    - Only awake enough to respond to external stimuli

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

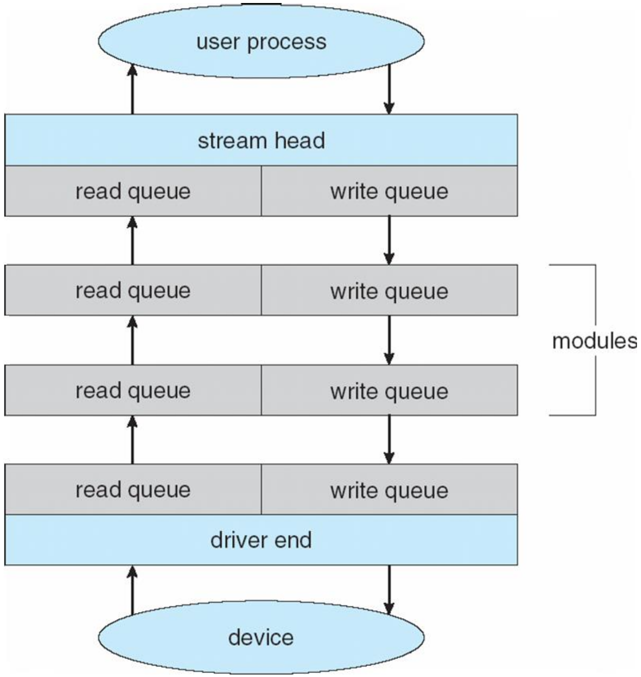
Life Cycle of An I/O Request



STREAMS

- **STREAM** – a full-duplex communication channel between a user-level process and a device in Unix System
- 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
  - **Flow control** option to indicate available or busy
- Asynchronous internally, synchronous where user process communicates with stream head

The STREAMS Structure



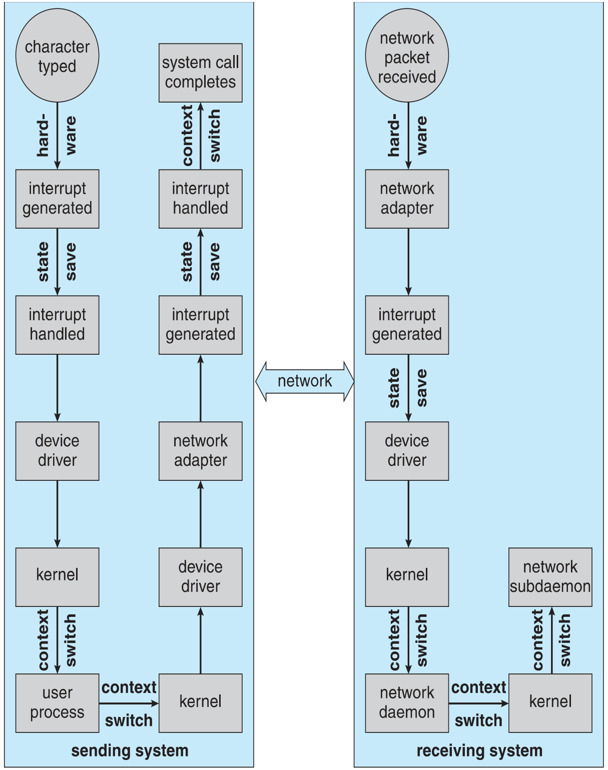
Performance

- 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

Improving Performance

- Reduce number of context switches
- Reduce data copying
- Reduce interrupts by using large transfers, smart controllers, polling
- Use DMA
- Use smarter hardware devices
- Balance CPU, memory, bus, and I/O performance for highest throughput
- Move user-mode processes / daemons to kernel threads

Intercomputer Communications



Thank you

johnjose@iitg.ac.in
http://www.iitg.ac.in/johnjose/