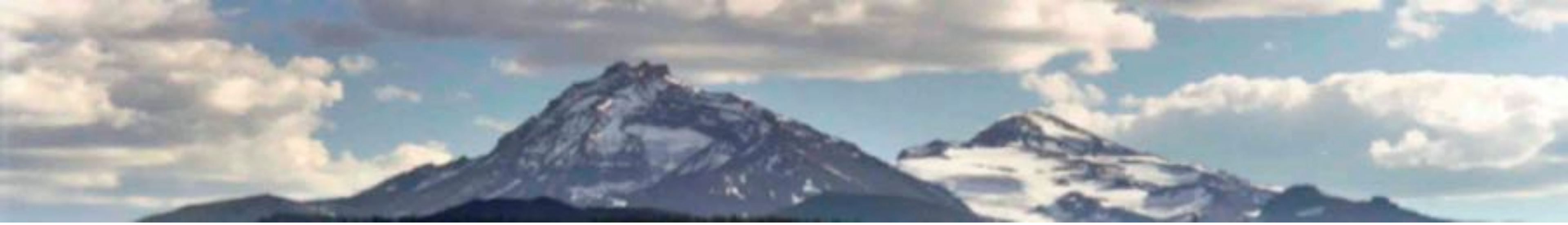


CIS 415
Operating Systems
Introduction

Prof. Allen D. Malony
Department of Computer and Information Science
Spring 2020

UNIVERSITY OF OREGON

Logistics

- Do not post links to a PDF version of the Operating Systems Concepts book online
  - OSC is not free and it is copyrighted material
  - It is a violation of copyright to provide a link to a PDF
  - UO can get in big trouble!
  - You can get in big trouble!
- Labs on Friday!
  - Begin working on Lab 1 assignment
- Project 1 to be posted by COB on Friday if not sooner

Objectives

- Operating systems
  - What is it?
  - What motivates it?
  - Bit of history
  - Overview
- Review of computer systems organization

What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware

What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs
  - Make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner
- Ok, do I need an OS just to print something?

```
main()
{
    printf("This is not a printout ...\n");
}
```

Without an OS

- Get printer manual
  - Figure out how to send messages to it
- Write the program
  - Put the character string "This is not a printout …" in a memory buffer
  - Do the stuff printer requires to send buffer to it
  - Go into endless loop
    - wait for someone to turn off computer eventually
- Get hold of a computer
  - It has to be all yours!
- Translate your program into machine code
  - Has to be specific to the computer you are using
- Figure out how to get program into memory
  - Font panel switches
  - Burn a ROM
  - Punch cards
- Somehow start program
- Turn off computer when done

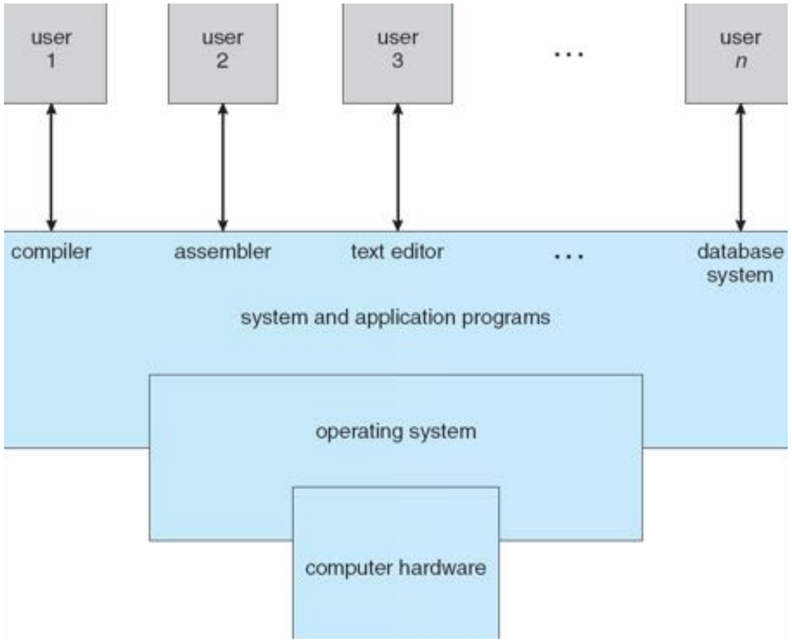
YUCK!!!

With an OS

- Put program in file
  - Put character string "This is not a printout …" in a memory buffer
  - Issue system call to send buffer to printer
- Compile the program
- Tell OS to run the program file
  - Turn on the printer first
- That's everything!
- Uh, ok, but what's a system call?

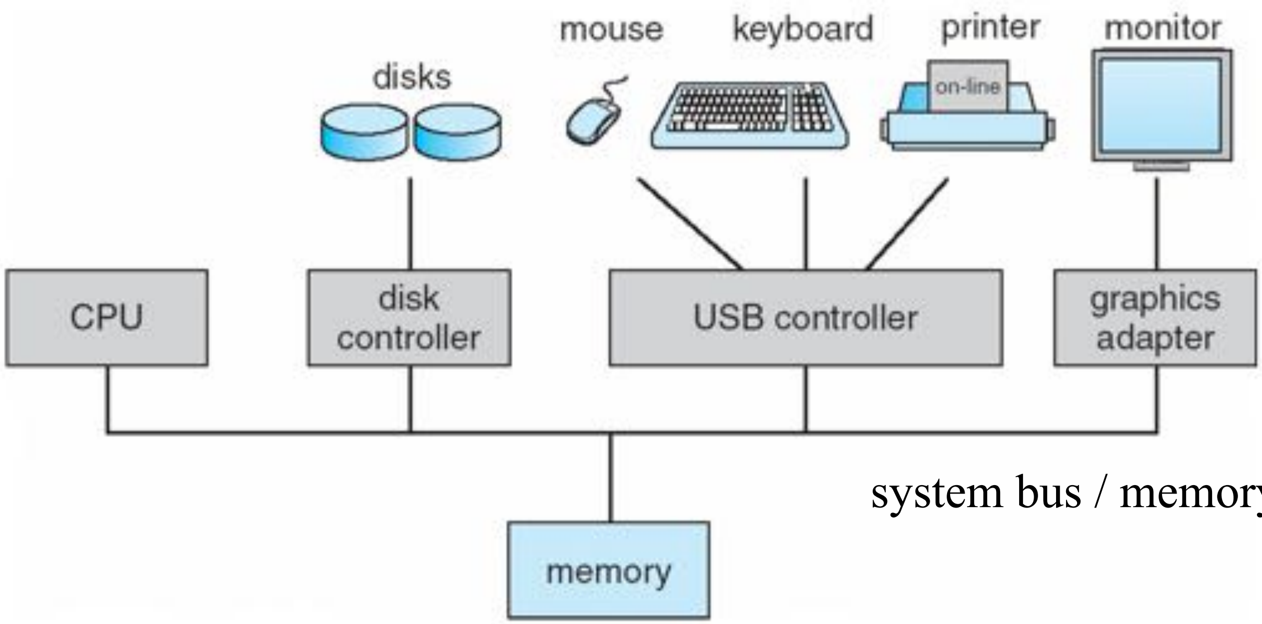
Computer System Structure

- Computer system divided into four components:
  - Hardware – provides basic computing resources
    - CPU, memory, I/O devices
  - Operating system
    - controls and coordinates use of hardware among various applications and users
  - Application programs
    - use system resources to solve computing problems
    - word processors, compilers, web browsers, database systems, video games, …
  - Users
    - People, machines, other computers



Basic Computer System Organization

- Computer-system operation
  - One or more CPUs, device controllers connect through common bus providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory cycles



system bus / memory bus

Computer System Architecture

- Most systems use a single general-purpose processor
  - Most systems have special-purpose processors as well
- Multiprocessors systems growing in importance
  - Advantages include:
    - increased throughput
    - economy of scale
    - increased reliability – graceful degradation or fault tolerance
  - Two types:
    - asymmetric multiprocessing – each processor has a specific task
    - symmetric multiprocessing – each processor performs all tasks
  - Can be used to run parallel applications

Computer System Operation

- I/O devices and the CPU can execute concurrently
- Each device controller is in charge of a device type
- Each device controller has a local buffer
- CPU moves data from/to main memory to/from local buffers associated with I/O devices
- I/O is from the device to local buffer of controller
- Device controller informs CPU that it has finished its operation by causing an interrupt

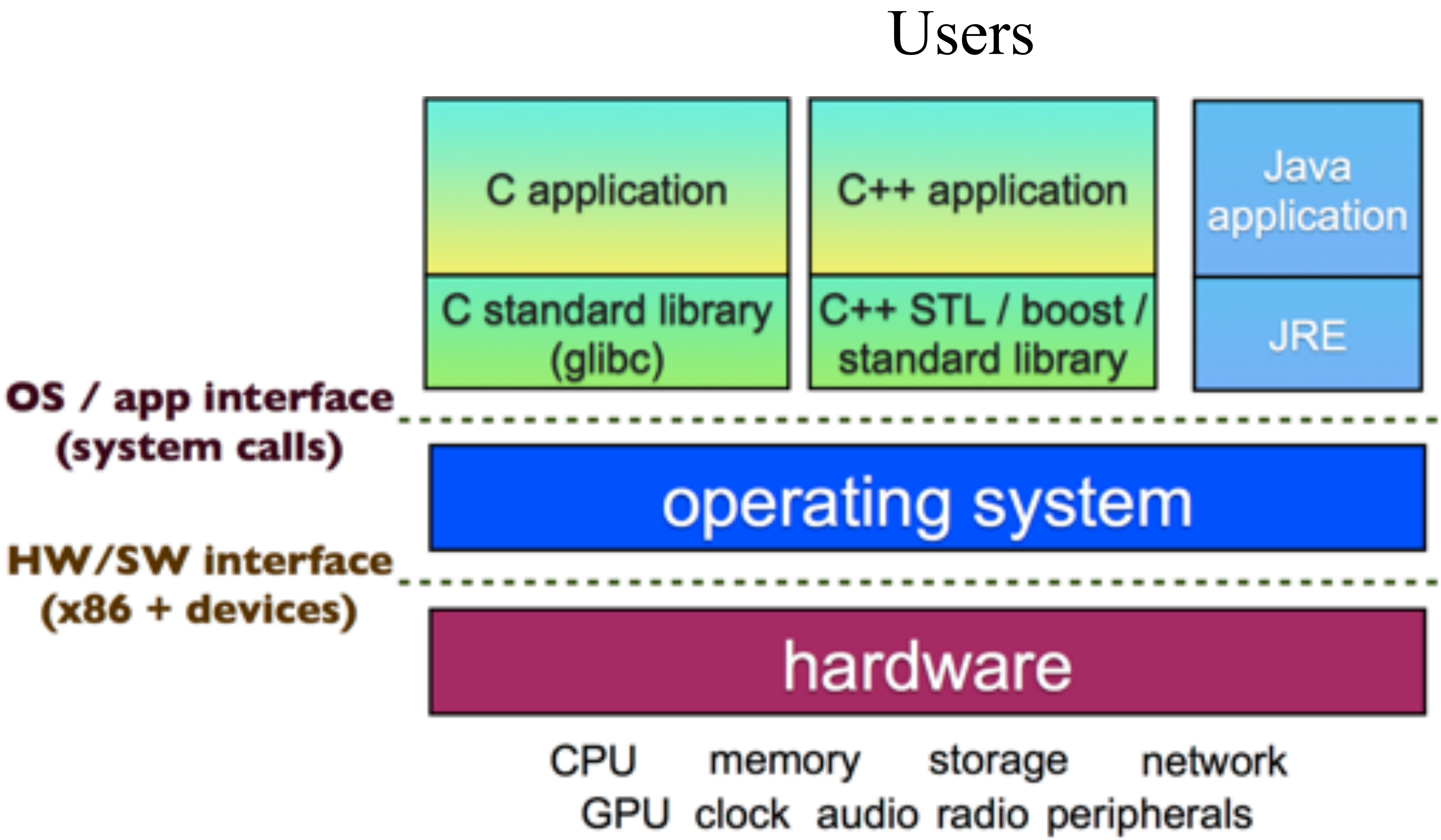
What Operating Systems Do?

- Depends on the point of view
- Users perspective
  - Convenience, ease of use, good performance
  - Dedicated resources
- Server perspective
  - Keep all users happy
  - Shared resources
- Both perspectives are valid
- Different OS requirements for different environments
- Common OS fundamentals at core

Operating System Definition

- OS is a *resource allocator*
  - Manages all resources
  - Decides between conflicting requests
    - efficient and fair resource use
- OS is a *control program*
  - Controls execution of programs to prevent errors and improper use of the computer
- OS provides both resource allocation and control
- Still can have different perspective on scope
  - An OS is everything a vendor ships you when you order an operating system
  - An OS is the one program running at all times (kernel), and everything else is a system program or application

Where does the OS fit in?



What does an OS do again?

- Acts as a resource manager
- Processes
  - Hides programs from one another
- Traffic cop
  - Resource management
  - Who gets to run, when?
- Memory management
  - Protection from other programs' mistakes
- Security
  - Protection from other programs' malice
- System call interface
  - Abstract, simplified interface to services
  - Like a function library, but communicates with the OS
- Portability
  - Programs don't have to take into account details of their environment
- Device management
- Communication
  - Between processes
  - To devices & networks

Operating System History

- 1950s: Simplify operators' job
- 1960s: Structure, concepts, concurrency, theory, …
- 1970s: Small and flexible (UNIX)
- 1980s: Individual user systems (PCs)
- 1990s: Internet, distributed systems
- 2000s: Security, Linux/Windows/Mac OS
- 2010s: Embedded, distributed, mobile, parallel, virtual, …
- Timeline of operating systems (Wikipedia)

  https://en.wikipedia.org/wiki/Timeline_of_operating_systems

Operating Systems (1950s)

- Primitive systems
  - Little memory
  - Programs stored on tape
- Single user
  - Batch processing
  - Computer execute one function at a time
- No overlap of I/O and computation
- OS for IBM 704
  - Atlas Supervisor (University of Manchester)
  - University of Michigan Executive System (UMES)
- You had to wear a suit or dress to use the OS!

IBM 704

Operating Systems (1960s)

- Multiprogramming
  - Timesharing
  - Multiple programs run concurrently
- Many operating systems concepts invented
  - Virtual memory, hierarchical file systems, synchronization, security and many more
- End up with slow, complex systems on limited hardware (Multics)
- Multics was an MIT project which partnered with Bell Labs and GE to try to support multiple users
- You had to have glasses to design the OS!

Edsger W. Dijkstra

Peter J. Denning

Operating Systems (1970s)

- Becoming more available
  - UNIX (Bell Labs)
    - first OS written in a high-level language
- Becoming more flexible
  - Extensible system
  - Community forms beyond developers (precursor to open source movement)
- Performance focus
  - Optimization of algorithms from 1960s
- You had to have beards to program the OS!

Brian W. Kernighan (L)
Dennis M. Ritchie (R)

Operating Systems (1980s)

- Critical mass reached
  - A variety of well-known systems, concepts
  - UNIX fragments (BSD, SysV, …)
  - Open source begins to emerge
- PC Emerges
  - Simple, single user
  - No network
  - Simple OSes: DOS
- Graphical User Interfaces
  - X Windows
  - Apple Macintosh

Bill Gates

Steve Jobs

Operating Systems (1990s)

- Connect to Internet
  - "Real OSes" for PCs
    - NT/2000+, Linux, Mac OS X
  - Web emergence
- Server systems galore
  - Mainframes even re-emerge
- Complex systems and requirements
  - Multiprocessors, memory hierarchy, interconnects, …
  - Parallel
  - Real-time
  - Distributed

Linux B. Torvalds and Penguin

More Operating System Humor

Operating Systems (2000s)

- More dimensions (problems) and range of scales
  - Distributed systems
  - Multicore
  - Security
  - Ubiquitous
  - Virtual Machines
  - Embedded
  - Mobile
- More dominance of OSes (Linux, MacOS, Windows)

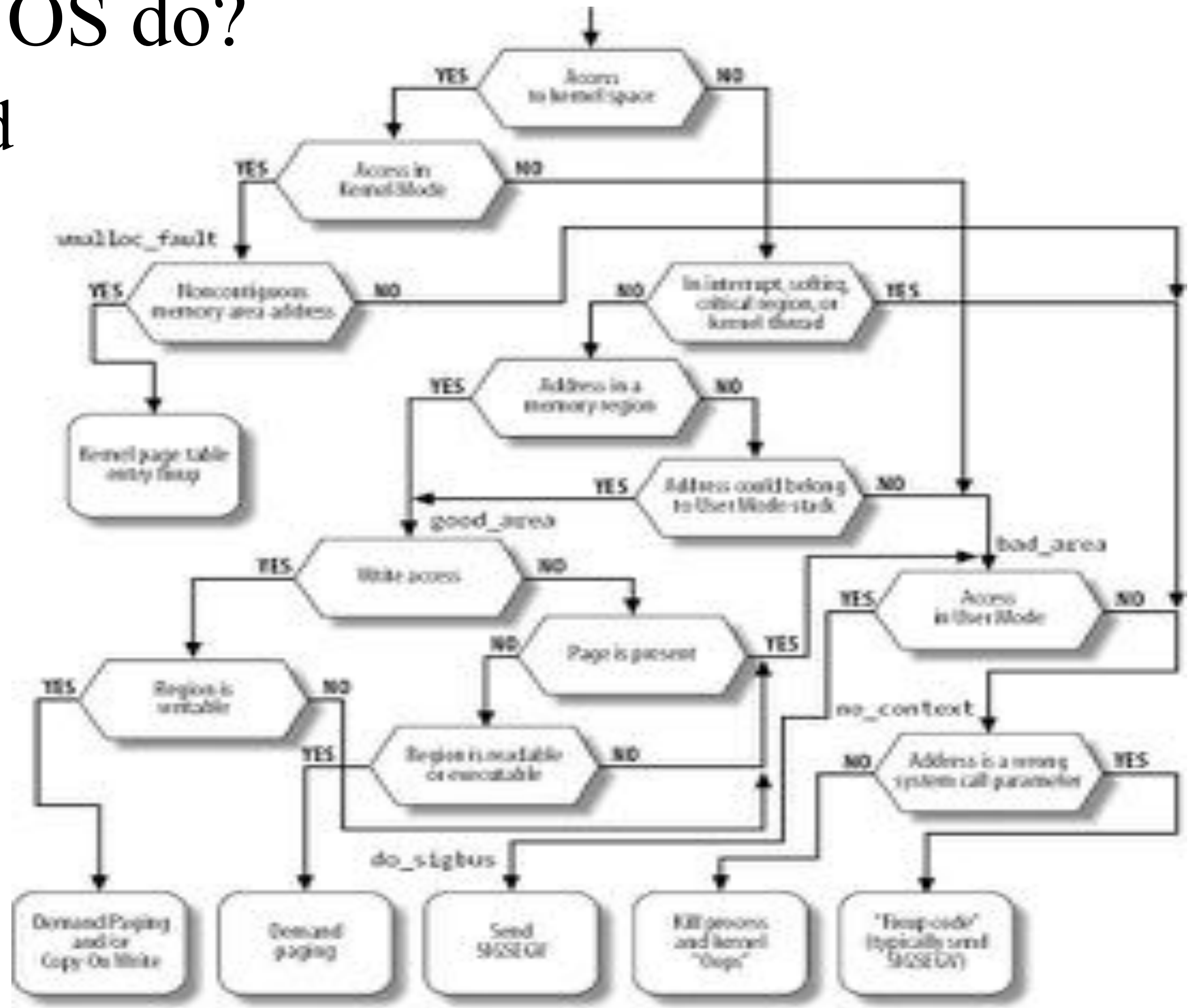
The Holy Trinity

Operating Systems (2010s)

- New, interesting challenges ahead
- Cloud computing is real
- Mobile devices are proliferating
- Internet of things
- Virtual machines
- Parallel computing
- Extreme-scale supercomputing
- OSes of various sorts for various purposes
  - iOS, Android, …
- Good time to learn OS principles and techniques

Under the OS Covers – OS Functions

- What does the OS do?
  - Mostly behind the scenes…
- Consider how page faults are handled
- What are page faults?



Page Fault Handling

- A page fault is caused when access to a virtual memory location is not found in physical memory
- What happens?
  - Trap is generated by the hardware
  - Handler in OS determines how to obtain memory
  - If page is still on disk, then handler
    - allocates physical page
    - makes I/O request to disk via file system and driver
  - Driver copies page from disk into new physical page
  - OS restarts the process at the trapped instruction

Page Fault Handling – Considerations

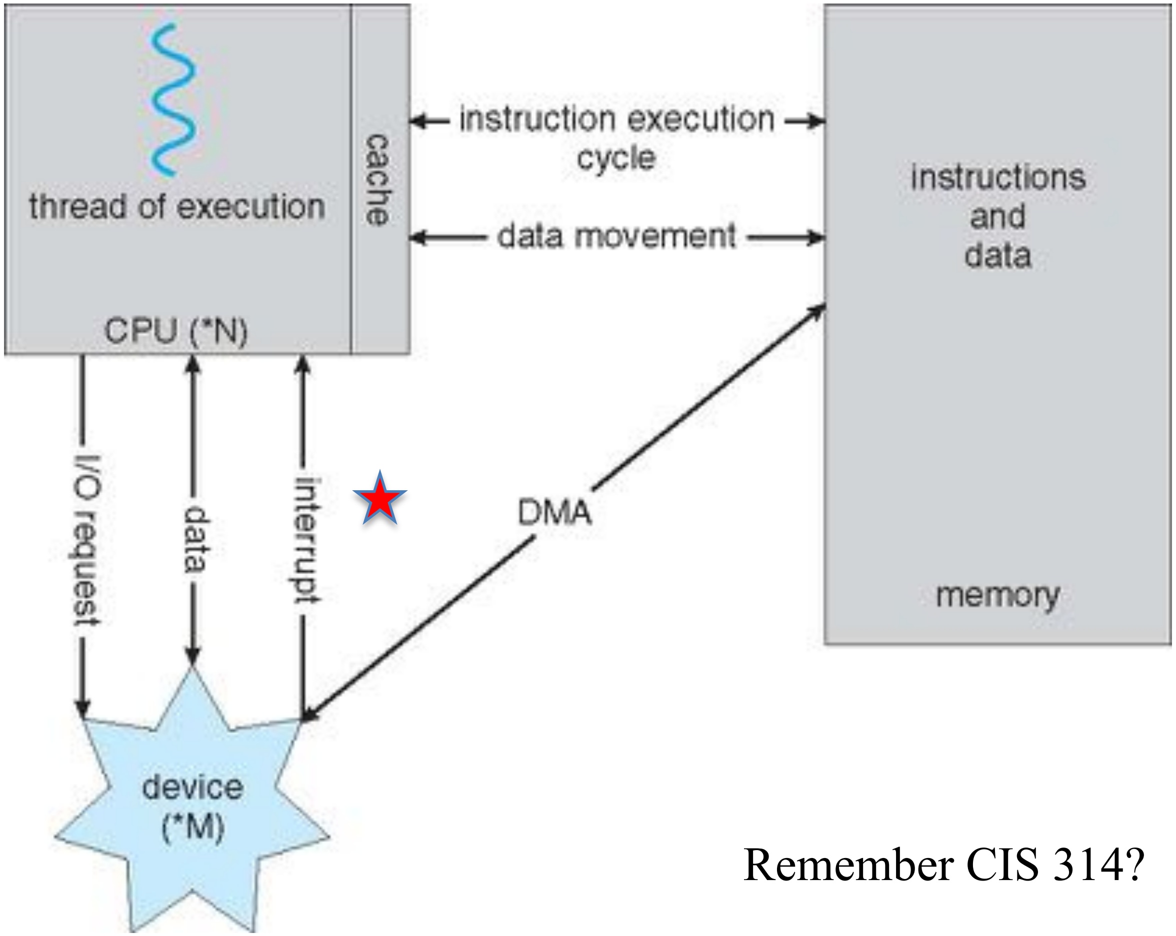
- Suppose there are multiple processes
  - More complex of a situation than with just 1 process
- OS has to make trade-offs
  - What if there are no physical pages available?
  - There may be multiple outstanding disk requests, so what order should they be processed?
  - How does the OS interact with hardware effectively?
  - What process should be run while waiting for the page fault to be handled?
  - Many others…

Learning about Operating Systems

- OS has many protocols like page fault handling
  - You will need to know them
- OS designers add layers of abstraction to simplify programming (e.g., virtual memory)
  - You will need to understand these concepts
- The design of protocols using these concepts involves trade-offs (e.g., disk performance)
  - You will need to understand why OS protocols are written the way that they are

Device I/O – How a modern computer works?

- A *von Neumann* architecture



Remember CIS 314?

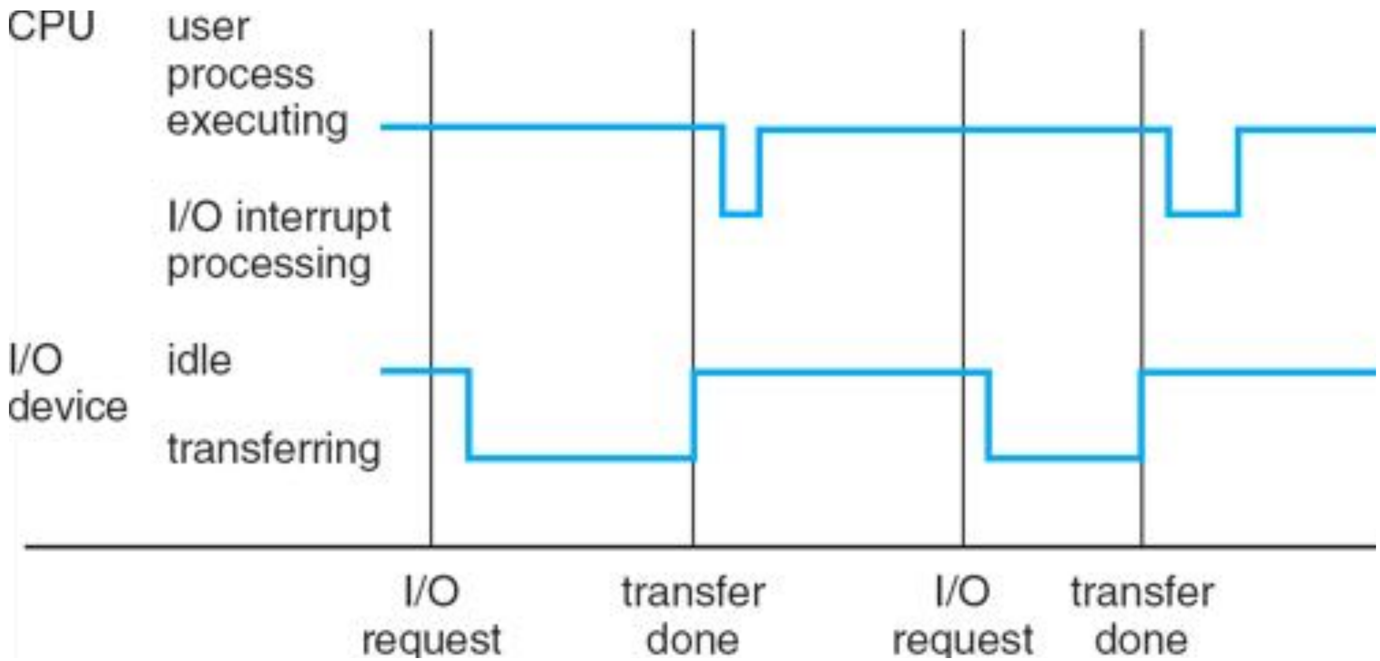
Common Functions of Interrupts

- Interrupt transfers control to the interrupt service routine, generally through an interrupt vector
- The operating system preserves the state of the CPU by storing registers and the program counter
- Determines which type of interrupt has occurred:
  - Polling
  - Vectored interrupt system
- Interrupt architecture must save the address of the interrupted instruction
- A trap or exception is a software-generated interrupt caused either by an error or a user request
- An operating system is interrupt driven

I/O Structure

- After I/O starts, control returns to user program …
  - Only upon I/O completion
    - wait instruction idles the CPU until the next interrupt
    - wait loop (contention for memory access)
    - at most one I/O request is outstanding at a time
  - Without waiting for I/O completion
    - system call – request to the OS to allow user to wait for I/O completion
    - device-status table contains entry for each I/O device indicating its type, address, and state
    - OS indexes into I/O device table to determine device status and to modify table entry to include interrupt

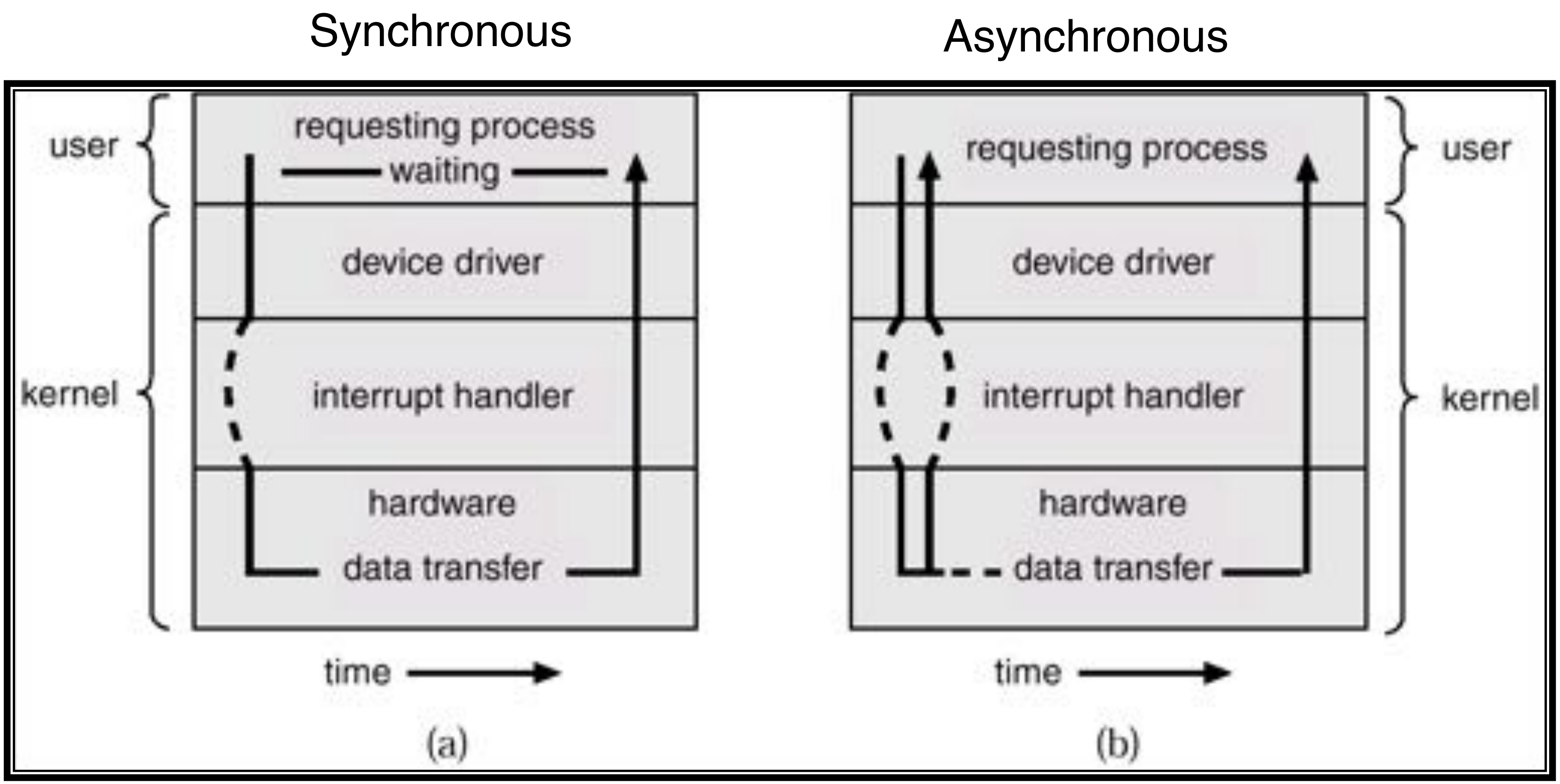
Interrupt Timeline



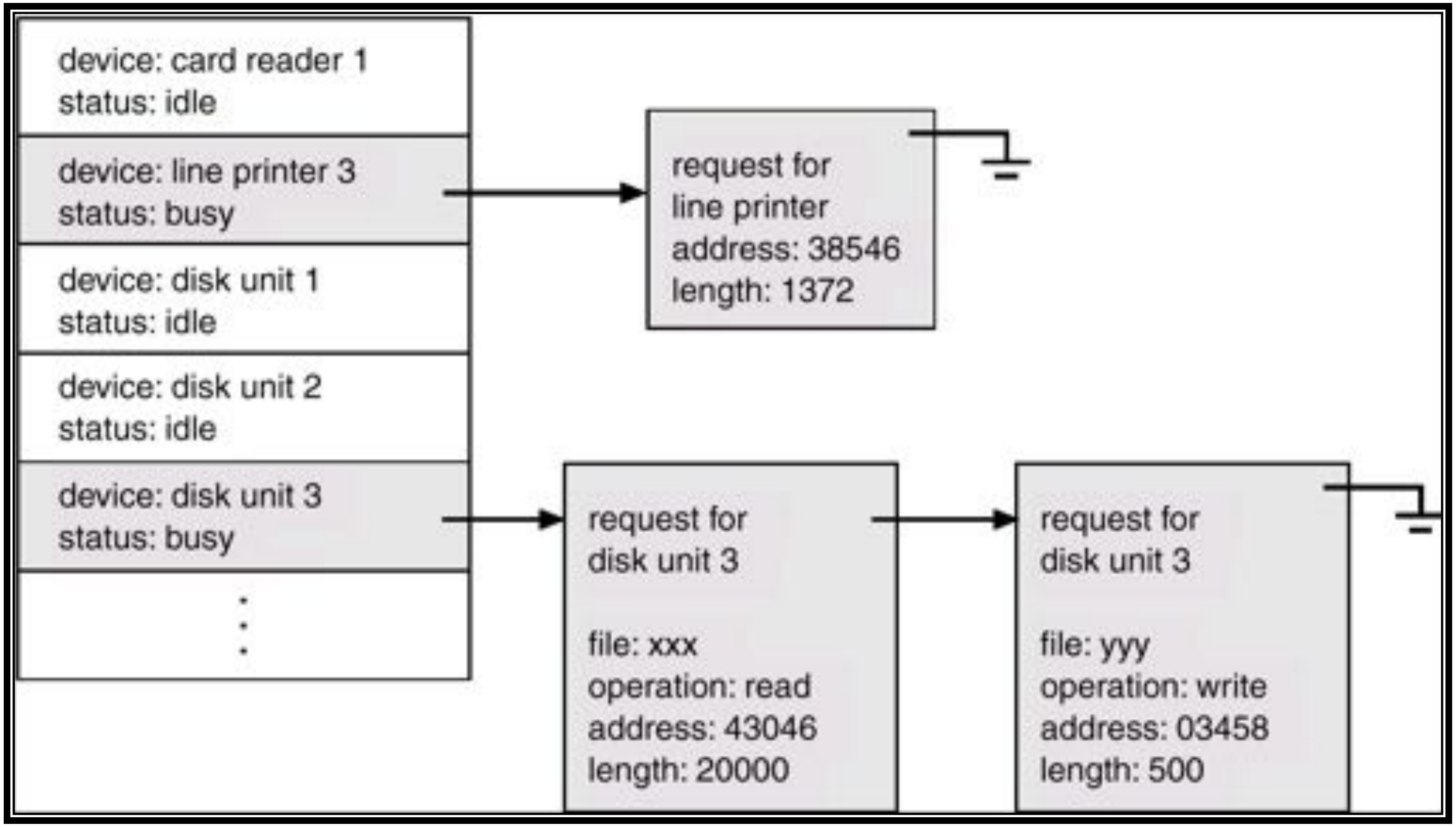
I/O Structure – Two Methods (1)

- Synchronous
  - After I/O starts, control returns to user program only upon I/O completion
  - A wait instruction idles the CPU until next interrupt
  - Or the CPU runs a wait loop (problems?)
  - No simultaneous I/O processing (only 1 request)
- Asynchronous
  - After I/O starts, control returns to user program without waiting for I/O completion
  - User program then waits for I/O complete
  - I/O interrupts upon completion

I/O Structure – Two Methods (2)

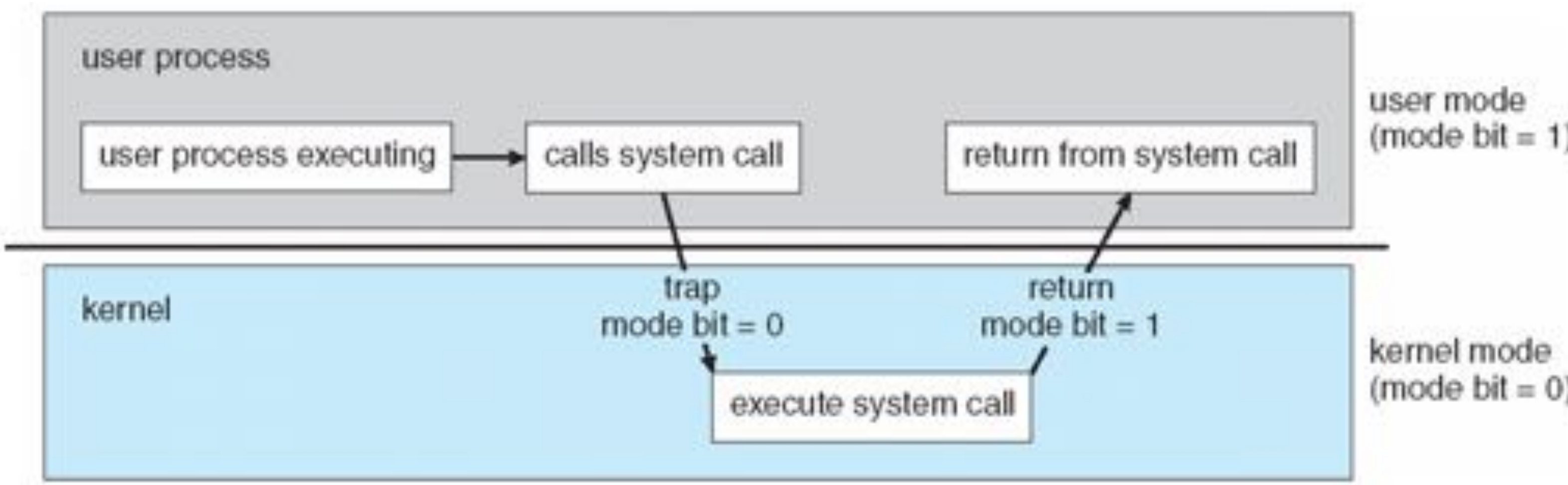


Device Status Table



Timer Interrupts for Kernel Control

- Timer to prevent infinite loop / process hogging resources
  - Timer is set to interrupt the computer after some time period
  - Keep a counter that is decremented by the physical clock
  - Operating system set the counter (privileged instruction)
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time



Process Management

- A process is a program in execution
  - A program is a passive entity
  - A process is an active entity (unit of work in the system)
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data
- Process termination requires reclaim of any reusable resources
- Single-threaded process has one program counter specifying location of next instruction to execute
  - Process executes instructions sequentially
  - One at a time, until completion
- Multi-threaded process has one program counter per thread
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
  - Concurrency by multiplexing the CPUs among the processes / threads

Process Management Activities

- OS is responsible for process management
- Activities include:
  - Creating and deleting both user and system processes
  - Suspending and resuming processes
  - Providing mechanisms for process synchronization
  - Providing mechanisms for process communication
  - Providing mechanisms for deadlock handling

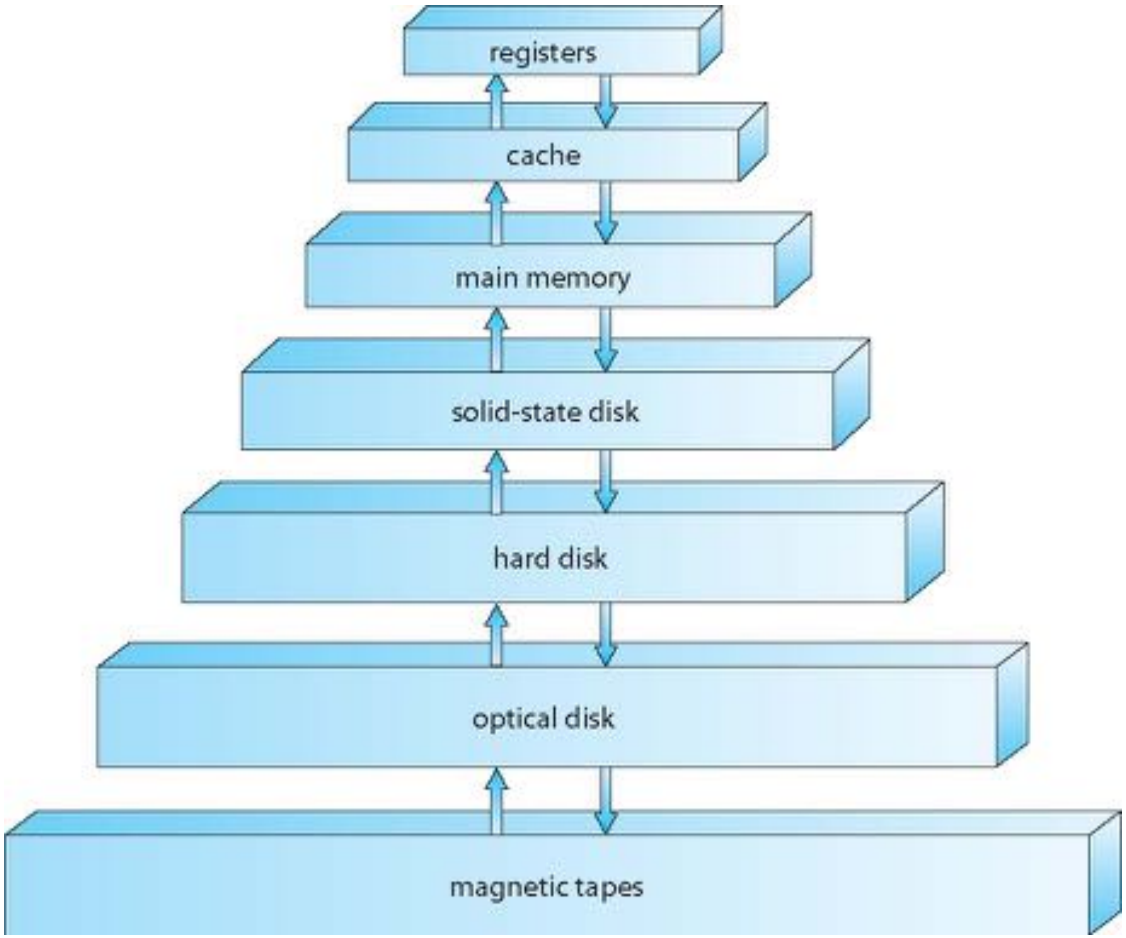
Scheduling

- Determine which task to perform given:
  - Multiple user processes
  - Multiple hardware components
- Provide effective performance
  - Responsive to users, CPU utilization
- Provide fairness
  - Do not starve low priority processes

Memory Management

- To execute a program all (or part) of the instructions must be in memory
- All (or part) of the data that is needed by the program must be in memory.
- Memory management determines what is in memory and when
  - Optimizing CPU utilization and computer response to users
- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

Storage Hierarchy



Fast Epensive Small

volatile

non-volatile

Slow Cheap Large

Storage Management

- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - file
  - Each medium is controlled by device
    - varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
  - Files usually organized into directories
  - Access control on most systems determines who can access what
  - OS activities include
    - Creating and deleting files and directories
    - Primitives to manipulate files and directories
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media

Protection and Security

- Protection – any mechanism for controlling access of processes or users to resources defined by the OS
- Security – defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish among users, to determine who can do what
  - User identities (user IDs, security IDs) include name and associated number, one per user
  - User ID then associated with all files, processes of that user to determine access control
  - Group identifier (group ID) allows set of users to be defined and controls managed, then also associated with each process, file
  - Privilege escalation allows user to change to effective ID with more rights

OS Structures

- Multiprogramming (batch system)
  - Single user cannot keep CPU and I/O devices busy at all times
  - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job selected and run via job scheduling
  - When it has to wait (for I/O for example), OS switches to another job
- Timesharing (multitasking)
  - CPU switches jobs so frequently
  - Response time should be < 1 second
  - Each user has at least one program executing in memory
  - If several jobs ready to run at the same time
  - Virtual memory allows processes not completely in memory

Computing Environments – Traditional

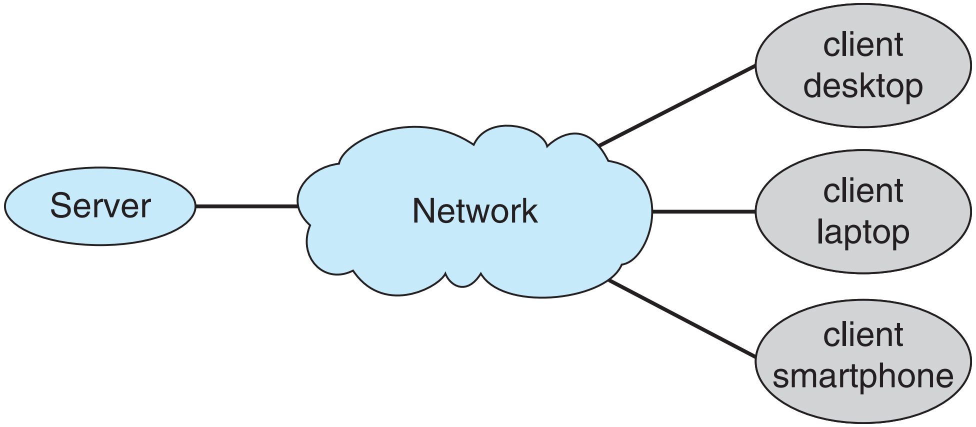
- Stand-alone general purpose machines
- But blurred as most systems interconnect with others (i.e., the Internet)
- Portals provide web access to internal systems
- Network computers (thin clients) are like Web terminals
- Mobile computers interconnect via wireless networks
- Networking becoming ubiquitous – even home systems use firewalls to protect home computers from Internet attacks

Computing Environments – Distributed

- Distributed computing
  - Collection of separate, possibly heterogeneous, systems networked together
    - network is a communications path, TCP/IP most common
      - Local Area Network (LAN)
      - Wide Area Network (WAN)
      - Metropolitan Area Network (MAN)
      - Personal Area Network (PAN)
  - Network Operating System provides features between systems across network
    - communication scheme allows systems to exchange messages
    - illusion of a single system

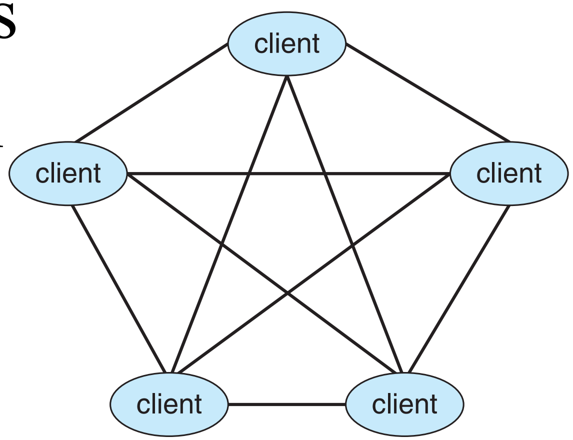
Computing Environments – Client-Server

- Client-Server Computing
  - Dumb terminals supplanted by smart PCs
  - Many systems now servers, responding to requests generated by clients
    - compute-server system provides an interface to client to request services (i.e., database)
    - file-server system provides interface for clients to store and retrieve files



Computing Environments – Client-Server

- Another model of distributed system
- P2P does not distinguish clients and servers
  - Instead all nodes are considered peers
  - May each act as client, server or both
  - Node must join P2P network
    - registers its service with central lookup service on network, or
    - broadcast request for service and respond to requests for service via discovery protocol
  - Examples include Napster and Gnutella, Voice over IP (VoIP) such as Skype



Computing Environments – Mobile

- Handheld smartphones, tablets, ….
- What is the functional difference between them and a "traditional" laptop?
- Extra feature – more OS features (GPS, gyroscope)
- Allows new types of apps like augmented reality
- Use IEEE 802.11 wireless, or cellular data networks for connectivity
- Leaders are Apple iOS and Google Android
- "Internet of things"

Computing Environments – Virtualization (1)

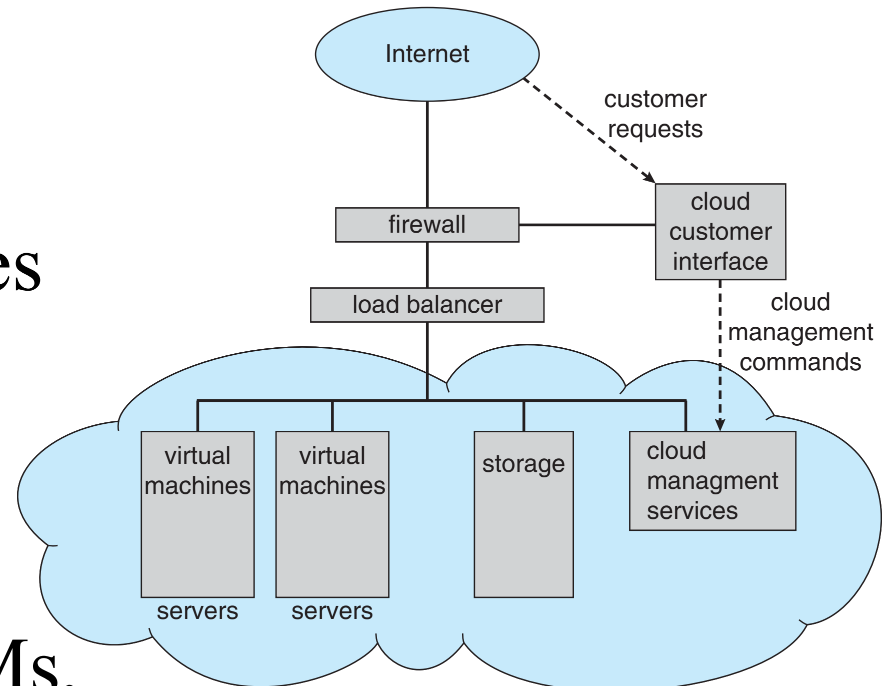
- Allows Oses to run applications within other Oses
- Emulation used when source CPU type different from target type (i.e. PowerPC to Intel x86)
  - Generally slowest method
  - When computer language not compiled to native code – Interpretation
- Virtualization – OS natively compiled for CPU, running guest OSes also natively compiled
  - Consider VMware running WinXP guests, each running applications, all on native WinXP host OS
  - VMM (virtual machine Manager) provides virtualization services

Computing Environments – Virtualization (2)

- Use cases involve laptops and desktops running multiple OSes for exploration or compatibility
  - Apple laptop running Mac OS X host, Windows as a guest
  - Developing apps for multiple OSes without having multiple systems
  - QA testing applications without having multiple systems
  - Executing and managing compute environments within data centers
- VMM can run natively, in which case they are also the host
  - There is no general purpose host then (VMware ESX and Citrix XenServer)

Computing Environments – Cloud Computing

- Delivers computing, storage, even applications as a service across a network
  - IaaS, PaaS, SaaS
- Logical extension of virtualization because it uses virtualization as the base for it functionality.
- Cloud computing environments composed of traditional OSes, plus VMMs, plus cloud management tools
  - Internet connectivity requires security like firewalls
  - Load balancers spread traffic across multiple applications



Open Source Operating Systems

- Operating systems made available in source-code format rather than just binary closed-source
- Counter to the copy protection and Digital Rights Management (DRM) movement
- Started by Free Software Foundation (FSF), which has "copyleft" GNU Public License (GPL)
- Examples include GNU/Linux and BSD UNIX (including core of Mac OS X), and many more
- Can use VMM like VMware Player (Free on Windows), VirtualBox (open source and free)
  - Use to run guest operating systems for exploration

Lab 1 Assignment

- Find Lab 1 on the CIS 415 web site
- Gets you familiar with the lab environment
- Programming assignments will be using a virtual machine image that will be run inside VirtualBox
  - Practice using VirtualBox and the VM
- Canvas will be used for managing and submitting your assignments and projects
- There is a survey as part of Lab 1 assignment
- The first lab meeting will focus on making sure everyone completes the Lab 1 assignment