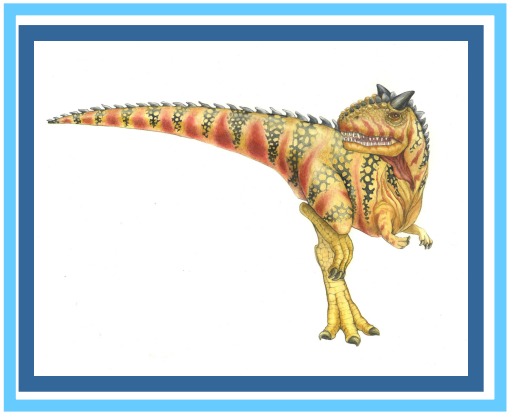
Chapter 1: Introduction



Chapter 1: Introduction

- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Structure
- Operating-System Operations
- Process Management
- Memory Management
- Storage Management
- Protection and Security (time permitting)
- The other chapters are covered in other courses.

Objectives

- To provide a grand tour of the major operating systems components
- To provide coverage of basic computer system organization
- We cover about half the book.
- More emphasis on Linux since it is the environment for the assignments.
- Use the assignments to provide practical examples of how to use the OS features.
- Understand the lowest level of a system

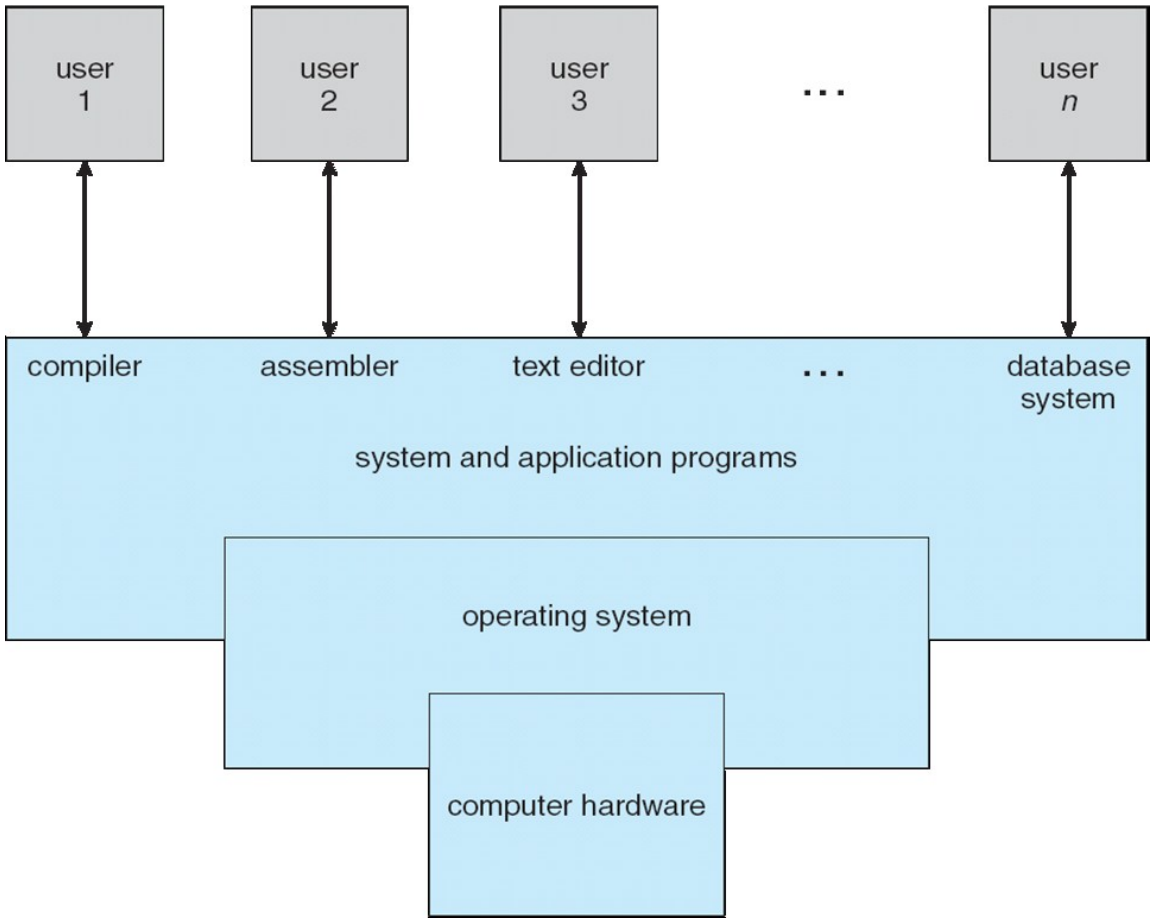
What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner
  - Provide safety, integrity, reliablity.
  - If you are a S/W or H/W vendor, lock the users to your system.

Computer System Structure

- Computer system can be 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 – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, video games

Four Components of a Computer System (old fashioned)



Operating System Definition

- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer
  - Restricts access to resources

Operating System Definition

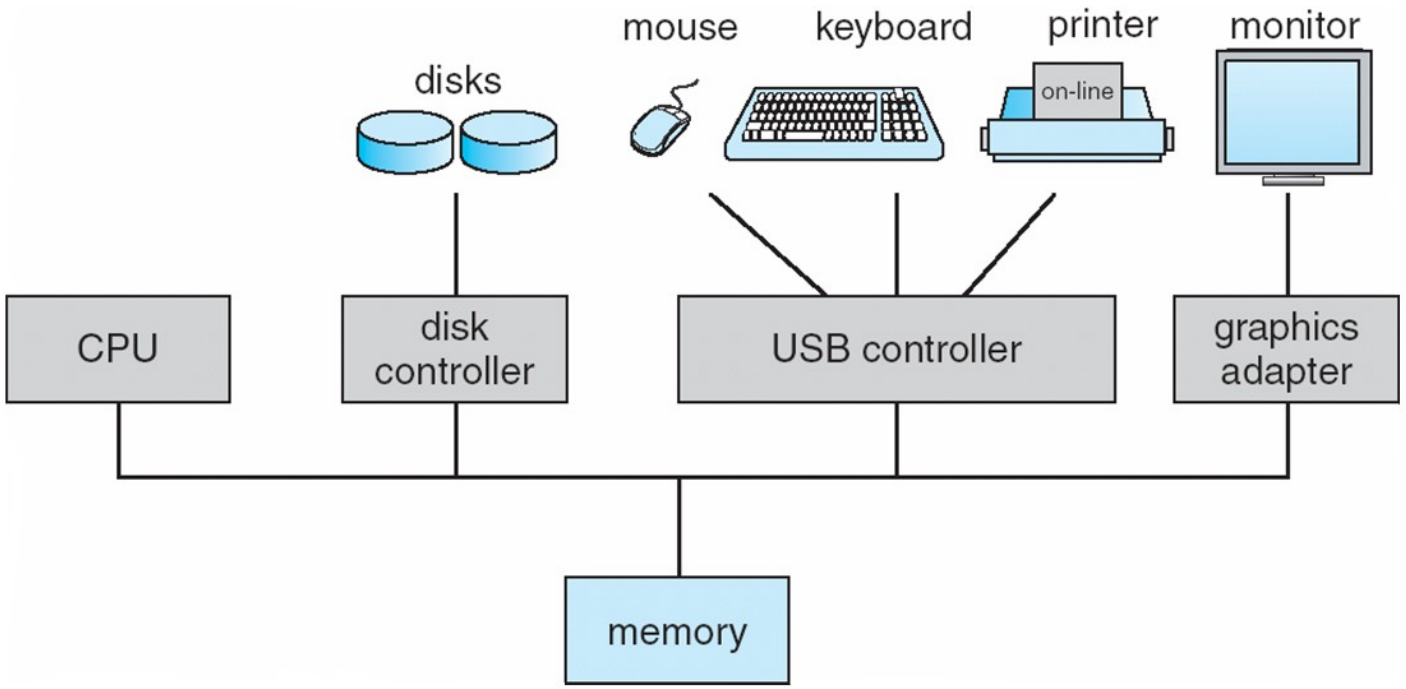
- No universally accepted definition
- “Everything a vendor ships when you order an operating system” is good approximation
  - But varies wildly and includes other things as well.
- “The one program running at all times on the computer” is the **kernel**. Everything else is either a system program (part of the operating system) or an application program

Computer Startup

- **bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution

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



Computer-System Operation

- I/O devices and the CPU can execute concurrently
- Each device controller is in charge of a particular device type
- Each device controller has local buffers
- CPU moves data from/to main memory to/from local buffers
- 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*

Common Functions of Interrupts

- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines
- Interrupt architecture must ensure the integrity of the system
- A *trap* is a software-generated interrupt caused either by an error or a user request
- An operating system is **interrupt driven**

Interrupt Handling

- 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
- Separate segments of code determine what action should be taken for each type of interrupt

I/O Structure

- After I/O starts, control returns to some user program only upon I/O completion (primitive)
  - Wait instruction idles the CPU until the next interrupt
  - At most one I/O request is outstanding at a time, no simultaneous I/O processing
- After I/O starts, control returns to a user program without waiting for I/O completion (modern)
  - **System call** – request to the operating system to allow process to wait for I/O completion (other process goes in)
  - **Device-status table** contains entry for each I/O device indicating its type, address, and state
  - Operating system indexes into I/O device table to determine device status and to modify table entry to include interrupt

Direct Memory Access Structure

- Used for high-speed I/O devices able to transmit information at close to memory speeds
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention
- Only one interrupt is generated per block, rather than the one interrupt per byte
- Without DMA performance is much slower and very erratic

Important

- DMA is a very important concept

Storage Structure

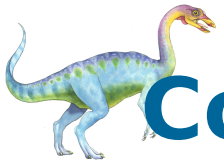
- Main memory – the only large storage media that the CPU can access directly
- Secondary storage – extension of main memory that provides large nonvolatile storage capacity
  - Magnetic disks – rigid metal or glass platters covered with magnetic recording material
    - Disk surface is logically divided into **tracks**, which are subdivided into **sectors**
    - The **disk controller** determines the logical interaction between the device and the computer
  - Flash memory – emulates a magnetic disk file system

Storage Hierarchy

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility
- **Caching** – copying information into faster storage system; Normally hidden from the programmer

Caching

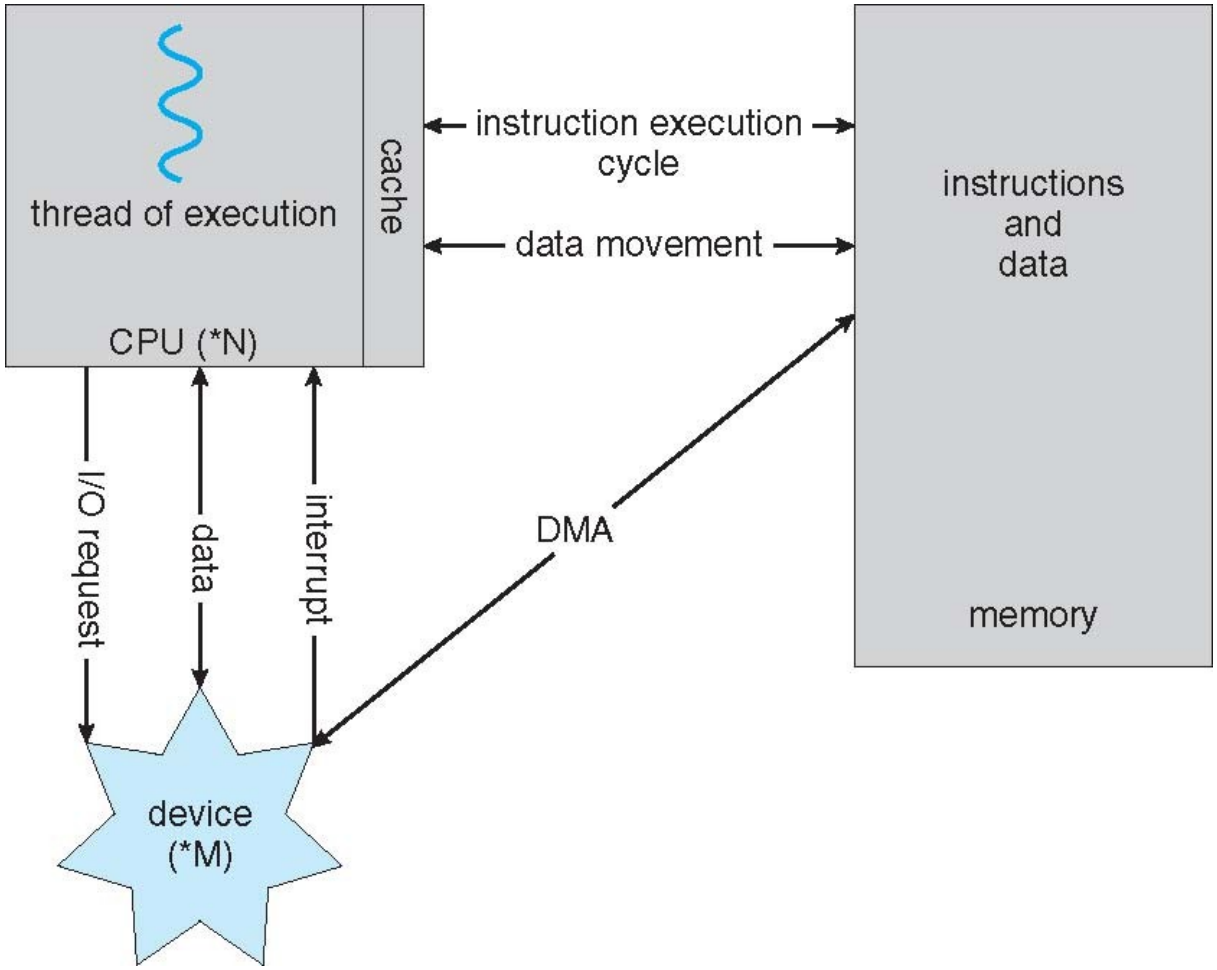
- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
  - If it is, information used directly from the cache (fast)
  - If not, data copied to cache and used there
- Cache smaller than storage being cached
  - Cache management important design problem
  - Cache size and replacement policy



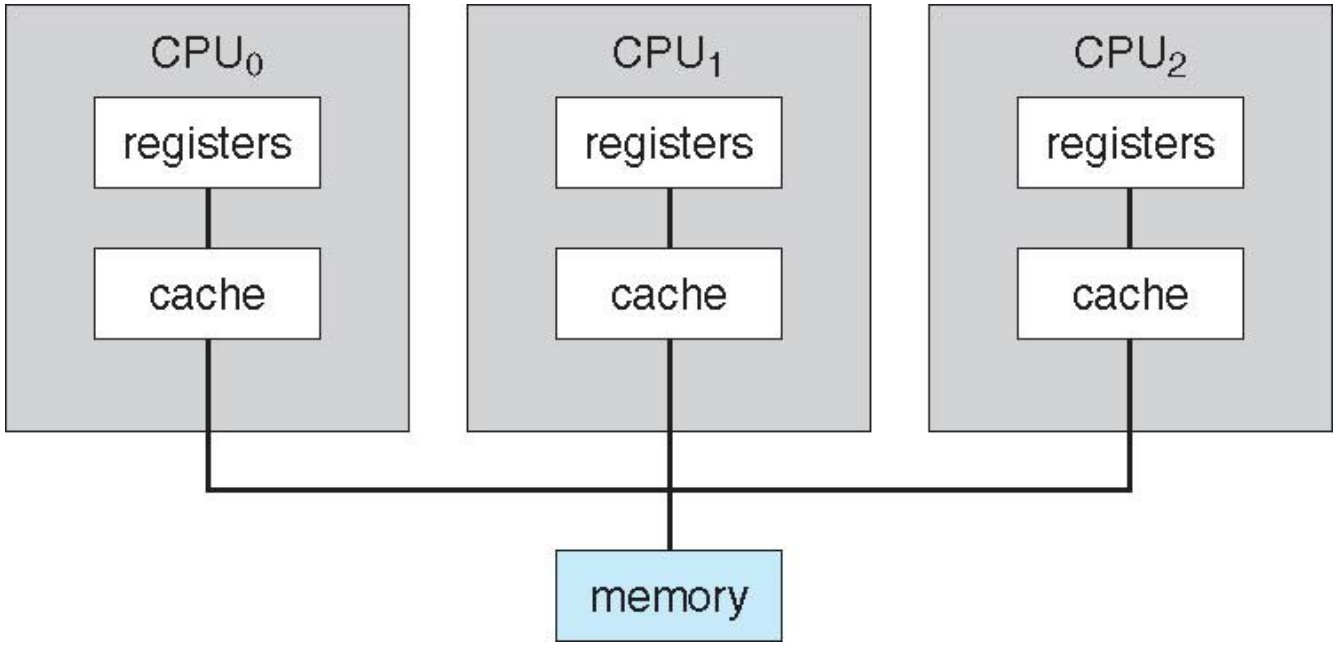
Computer-System Architecture

- Tiny systems use a single general-purpose processor (PDAs and cellphones)
  - Most systems have special-purpose processors as well
- Multiprocessors systems growing in use and importance
  - Also known as parallel systems, tightly-coupled systems
  - Advantages include
    - Increased throughput
    - Economy of scale
    - Increased reliability – graceful degradation or fault tolerance
  - Two types
    1. Asymmetric Multiprocessing
    2. Symmetric Multiprocessing

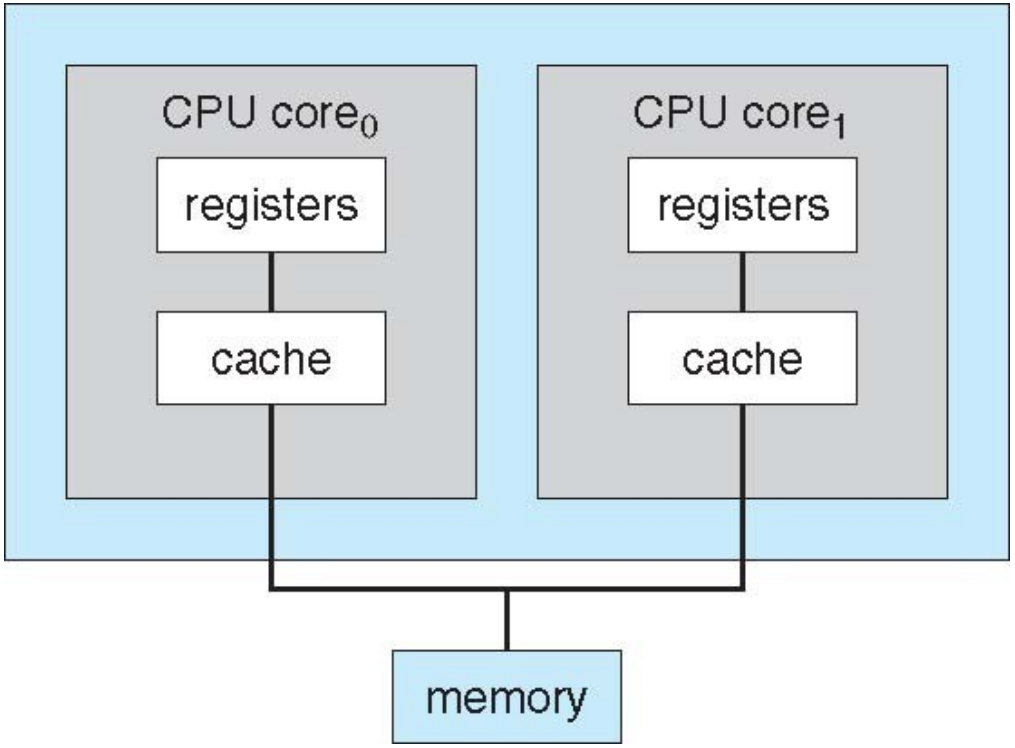
How a Modern Computer Works



Symmetric Multiprocessing Architecture



A Dual-Core Design



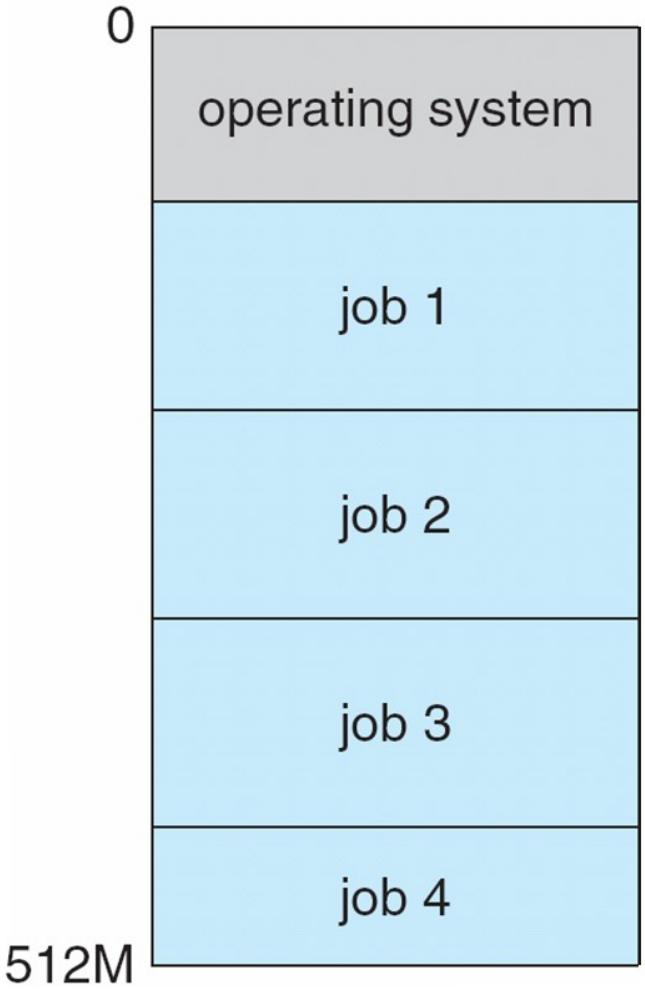
Clustered Systems

- Like multiprocessor systems, but multiple systems working together
  - Usually sharing storage via a storage-area network (SAN)
  - Provides a high-availability service which survives failures
    - Asymmetric clustering has one machine in hot-standby mode
    - Symmetric clustering has multiple nodes running applications, monitoring each other
  - Some clusters are for high-performance computing (HPC)
    - Applications must be written to use parallelization

Operating System Structure

- **Multiprogramming** long time ago was needed for efficiency
  - Single user could not 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)** is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing
  - **Response time** should be much less than 1 second
  - Each user has at least one program executing in memory **process**
  - If several jobs ready to run at the same time **CPU scheduling**
  - If processes don't fit in memory, **swapping** moves them in and out to run

Memory Layout for Multiprogrammed System (vintage)



Operating-System Operations

- Interrupts should be handled by the system
- Software errors (**exceptions** or **traps**) too
- User program requests (like open file, talk to the net) too
- Processes should not be allowed to access other processes space
- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as **privileged**, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user

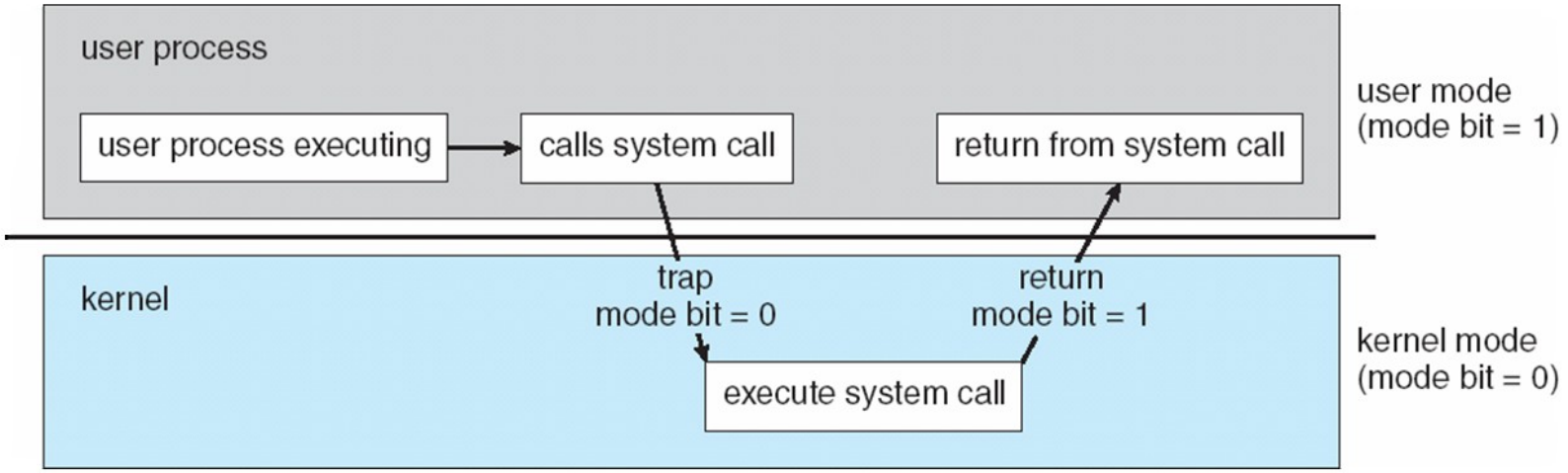
Important!

Previous slide was very important

(that's what allows modern OSes to work)

Example: Infinite loops

- Timer to prevent infinite loop / process hogging resources
  - Set interrupt after specific period
  - Operating system decrements counter
  - When counter zero generate an interrupt
- Also used to perform system calls:



Process Management

- A process is a program in execution. It is a unit of work within the system. Program is a *passive entity*, process is an *active entity*.
- 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

Memory Management

- All data in memory before and after processing (kind of)
- All instructions in memory in order to execute (kind of)
- Memory management determines what is in memory 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 Management

- OS provides uniform, logical view of information storage
  - We cover this near the end of the course
- File-System management
  - and this too.

I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
  - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs)
  - General device-driver interface
  - Drivers for specific hardware devices

Protection and Security

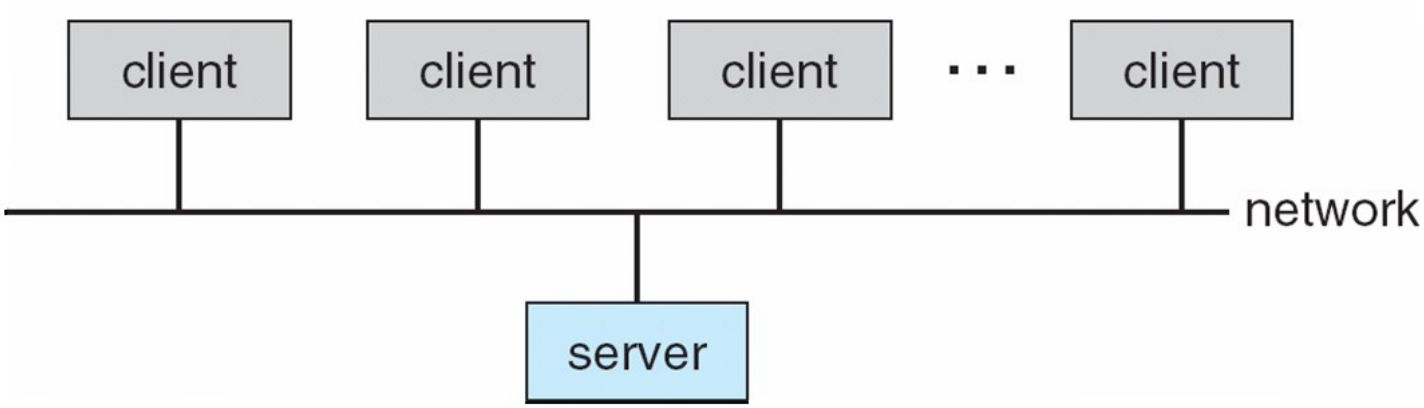
- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service

Computing Environments

- Traditional computer
  - Blurring over time
  - Office environment
    - PCs connected to a network, terminals attached to mainframe or minicomputers (!) providing batch and timesharing
    - Now portals allowing networked and remote systems access to same resources
  - Home networks use “home” routers with
    - WiFi
    - NAT, port forwarding, VPN, DDNS
    - Firewall

Computing Environments (Cont)

- Client-Server Computing
  - PCs without home directories are the clients
  - **Servers**, respond to requests generated by **clients**
    - **Compute-server** provides an interface to client to request services (i.e. database)
    - **File-server** provides interface for clients to store and retrieve files



Peer-to-Peer Computing

- 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*

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, BSD UNIX (including core of Mac OS X), and Sun Solaris

Open Source

- Many devices/appliances run on Linux (routers, parkmeters)
- Many new multimedia devices run Linux (Archos, Cowon, Nokia N8x0)
- Since Vista grew in size out of control, many new mini laptops run Linux (Asus, Acer, HP, OLPC)
- Microsoft supports interconnectivity with Linux
- About half the web servers run open source OS and/or server

End of Chapter 1