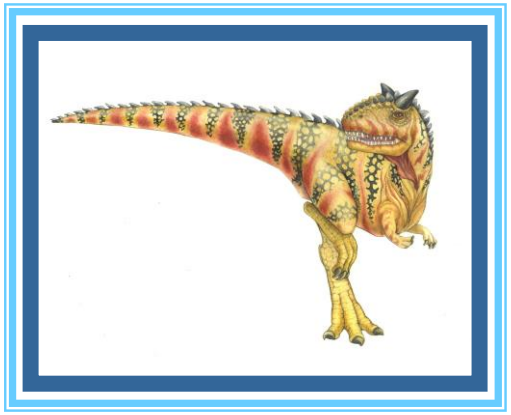
Chapter 1: Introduction

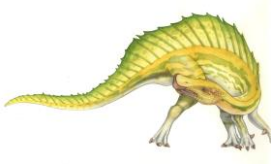


Objectives

- Describe the general organization of a computer system and the role of interrupts
- Describe the components in a modern, multiprocessor computer system
- Illustrate the transition from user mode to kernel mode
- Discuss how operating systems are used in various computing environments
- Provide examples of free and open-source operating systems

What Does the Term Operating System Mean?

- An operating system is “fill in the blanks”
- What about:
  - Car
  - Airplane
  - Printer
  - Washing Machine
  - Toaster
  - Etc.

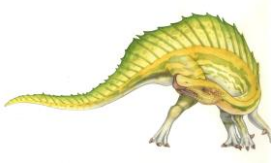


What is an Operating System?

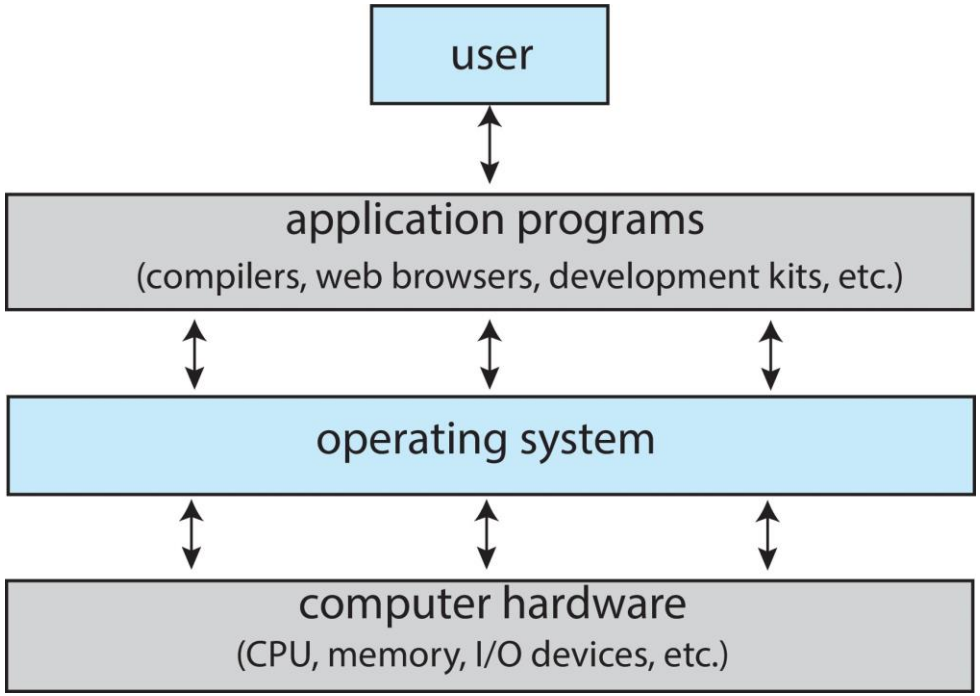
- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an *efficient manner*

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
  - Users
    - People, machines, other computers



Abstract View of Components of Computer

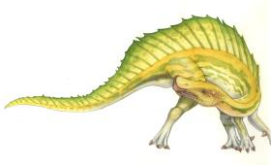


What Operating Systems Do

- Depends on the point of view
- Users want convenience, **ease of use** and **good performance**
  - Don't care about **resource utilization**
- But shared computer such as **mainframe** or **minicomputer** must keep all users happy
  - Operating system is a **resource allocator** and **control program** making efficient use of HW and managing execution of user programs
- Users of dedicate systems such as **workstations** have dedicated resources but frequently use shared resources from **servers**
- Mobile devices like smartphones and tablets are resource poor, optimized for usability and battery life
  - Mobile user interfaces such as touch screens, voice recognition
- Some computers have little or no user interface, such as embedded computers in devices and automobiles
  - Run primarily without user intervention

Defining Operating Systems

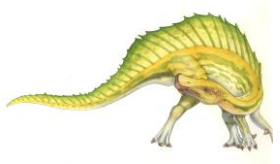
- Term OS covers many roles
  - Because of myriad designs and uses of OSes
  - Present in toasters through ships, spacecraft, game machines, TVs and industrial control systems
  - Born when fixed use computers for military became more general purpose and needed resource management and program control



Operating System Definition

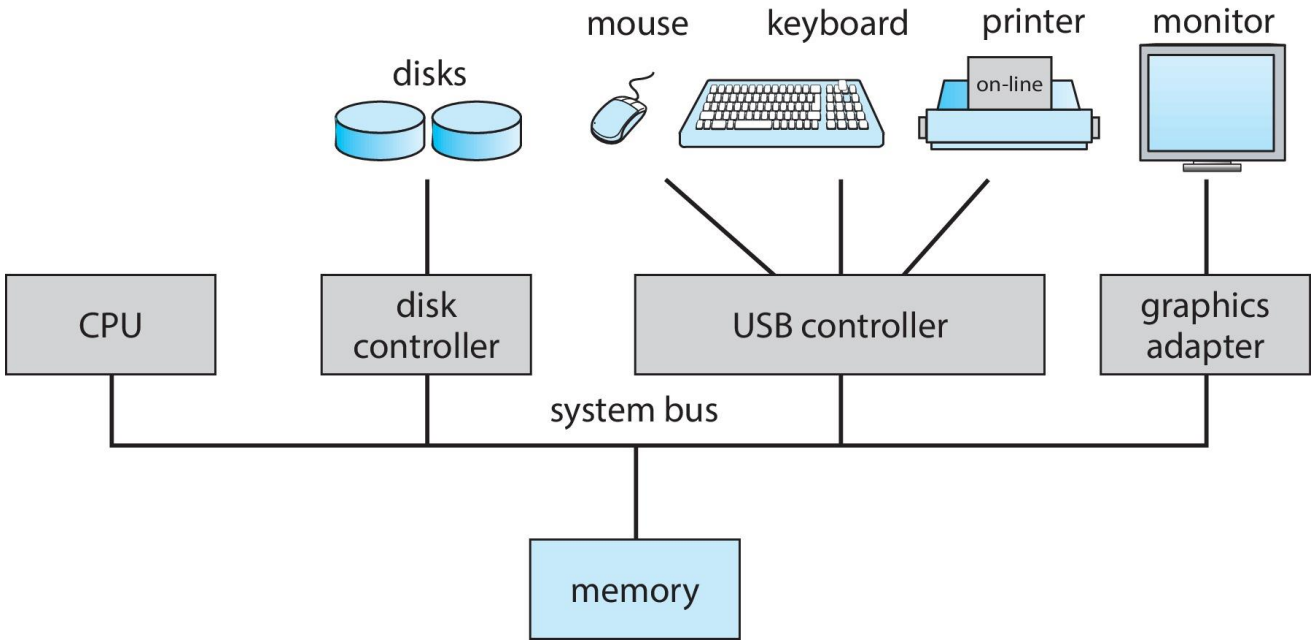
- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is a good approximation
  - But varies wildly
- "The one program running at all times on the computer" is the **kernel,** part of the operating system
- Everything else is either
  - A ***system program*** (ships with the operating system, but not part of the kernel) , or
  - An ***application program***, all programs not associated with the operating system
- Today's OSes for general purpose and mobile computing also include ***middleware*** – a set of software frameworks that provide additional services to application developers such as databases, multimedia, graphics

Overview of Computer System Structure



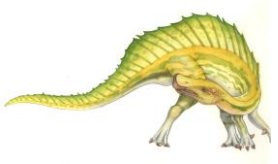
Computer System Organization

- 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

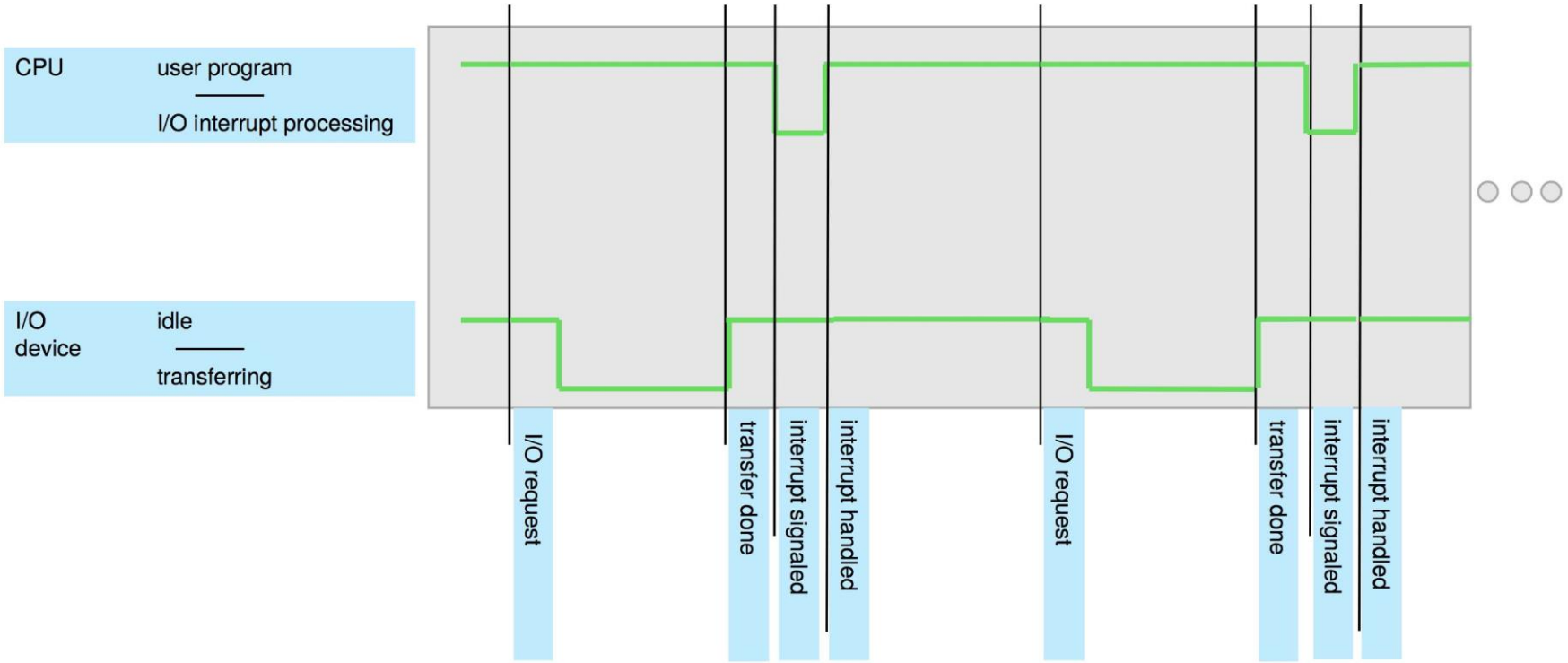
- I/O devices and the CPU can execute concurrently
- Each device controller is in charge of a particular device type
- Each device controller has a *local buffer*
- Each device controller type has an operating system **device driver** to manage it
- CPU moves data from/to main memory to/from local buffers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



Common Functions of Interrupts

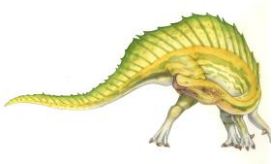
- An operating system is **interrupt driven**
- Interrupt transfers control to the **interrupt service routine (ISR)** generally, through the **interrupt vector**, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction
-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

Interrupt Timeline

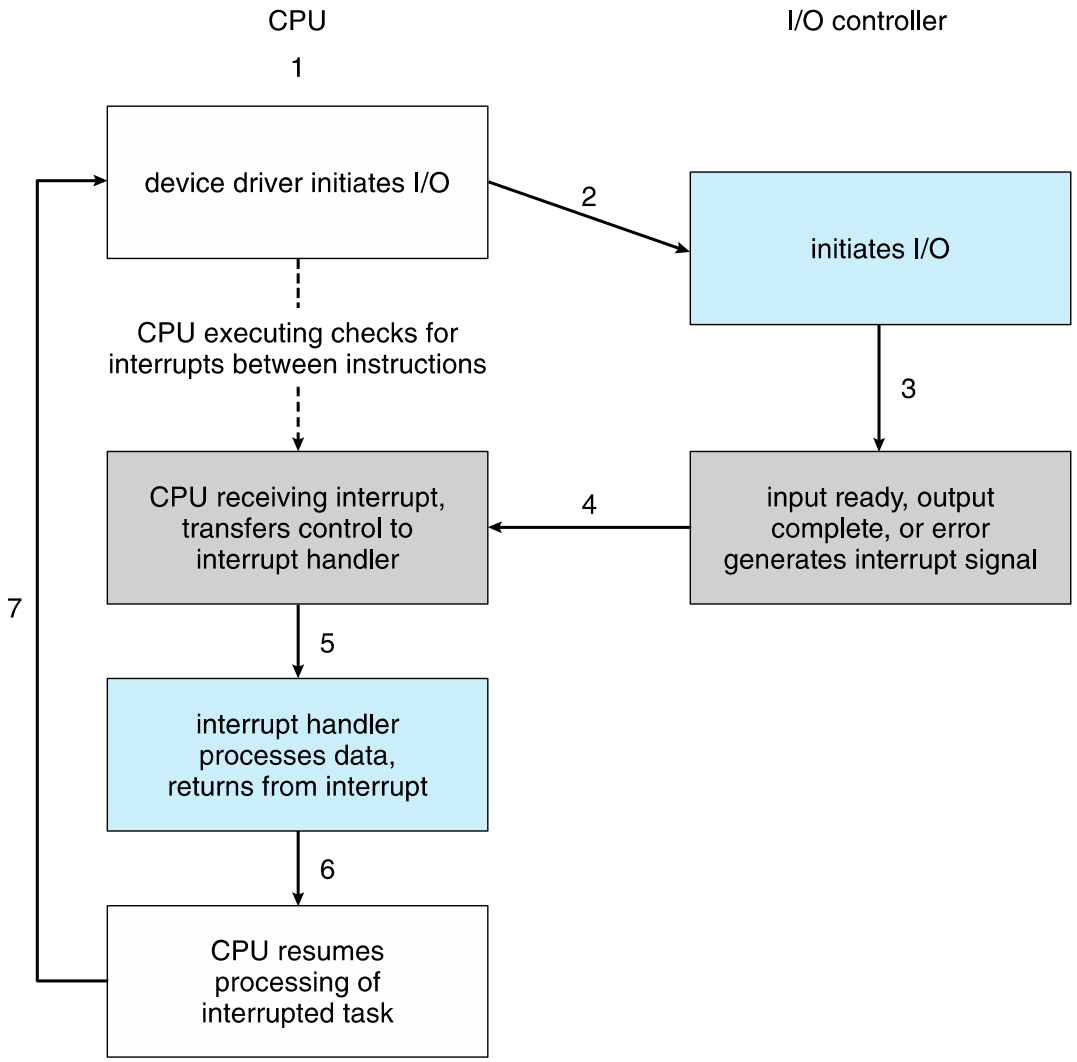


Interrupt Handling

- The operating system preserves the state of the CPU by storing the registers and the program counter(PC)
- Determines which type of interrupt has occurred:
- Separate segments of code determine what action should be taken for each type of interrupt



Interrupt-drive I/O Cycle



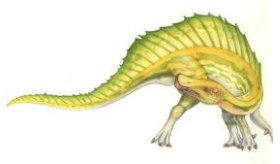
I/O Structure

- Two methods for handling I/O
  - After I/O starts, control returns to user program only upon I/O completion
  - After I/O starts, control returns to user program without waiting for I/O completion

I/O Structure (Cont.)

- 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, no simultaneous I/O processing
- After I/O starts, control returns to user program 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

Storage Structure

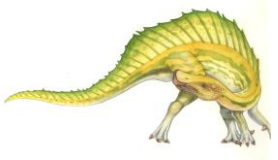


Storage Structure

- Main memory – only large storage media that the CPU can access directly
  - **Random access**
  - Typically **volatile**
  - Typically **random-access memory** in the form of **Dynamic Random-access Memory (DRAM)**
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity

Storage Structure (Cont.)

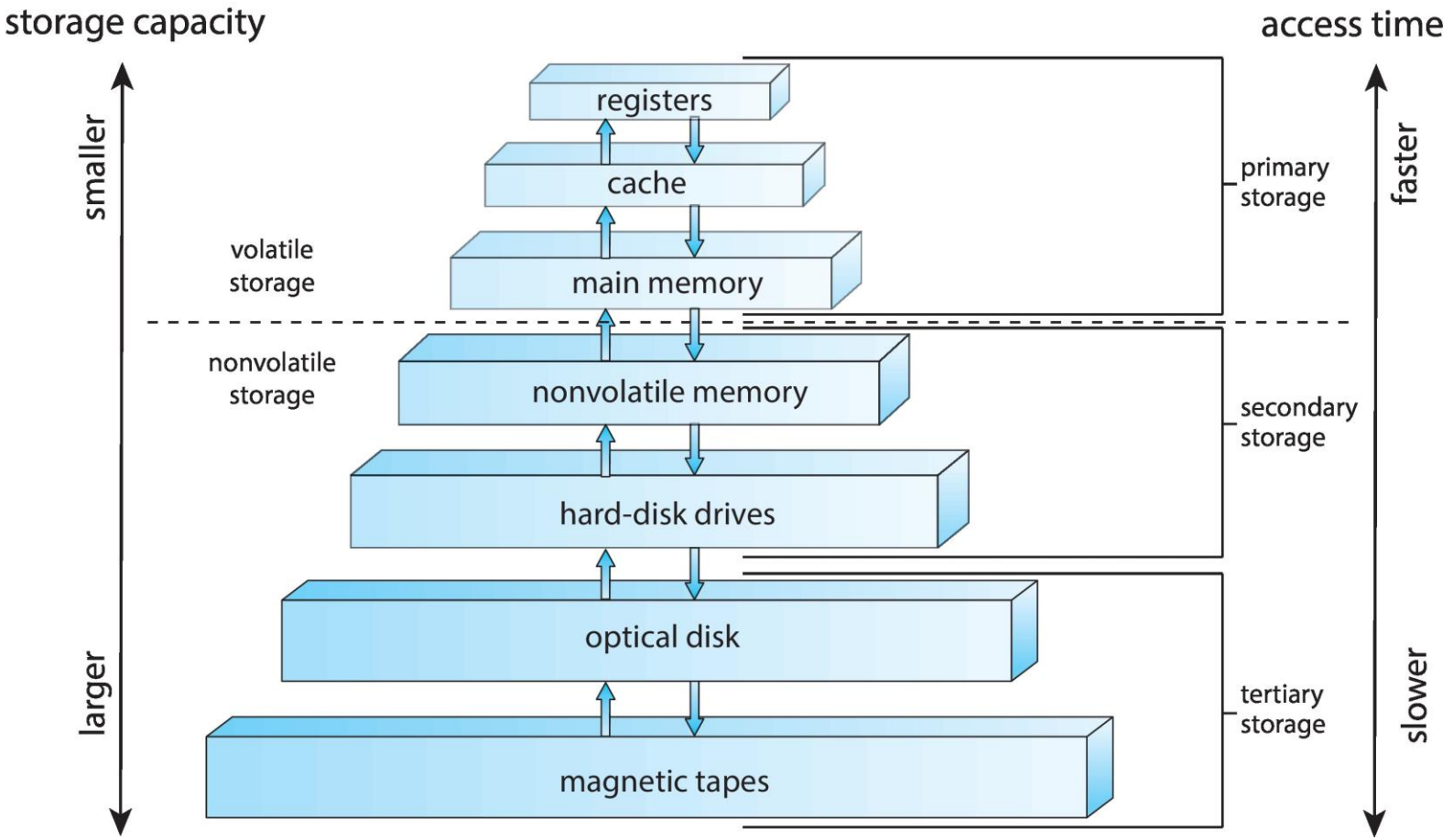
- **Hard Disk Drives** (**HDD**) – rigid metal or glass platters covered with magnetic recording material
  - Disk surface is logically divided into **tracks**, which are subdivided into **sectors**
  - The **disk controller** determines the logical interaction between the device and the computer
- **Non-volatile memory** (**NVM**) devices– faster than hard disks, nonvolatile
  - Becoming more popular as capacity and performance increases, price drops



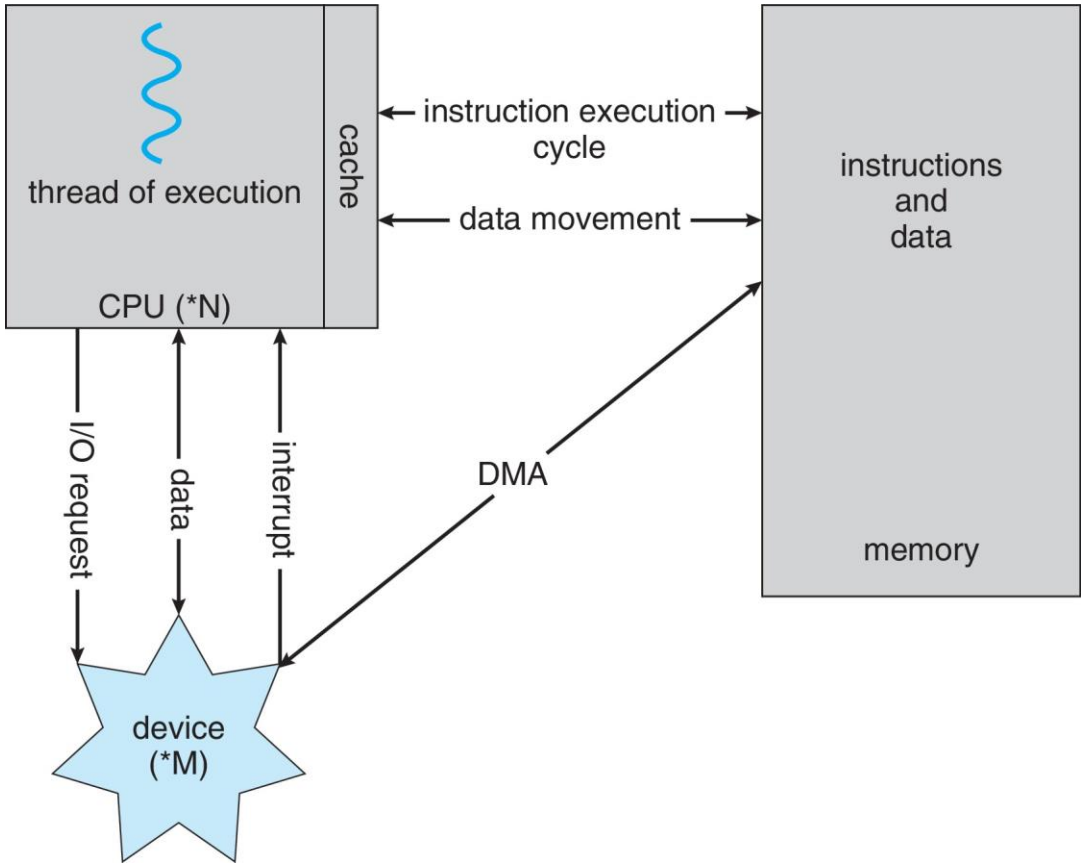
Storage Hierarchy

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility
- **Caching** – copying information into faster storage system; main memory can be viewed as a cache for secondary storage

Storage-Device Hierarchy



How a Modern Computer Works



A von Neumann architecture

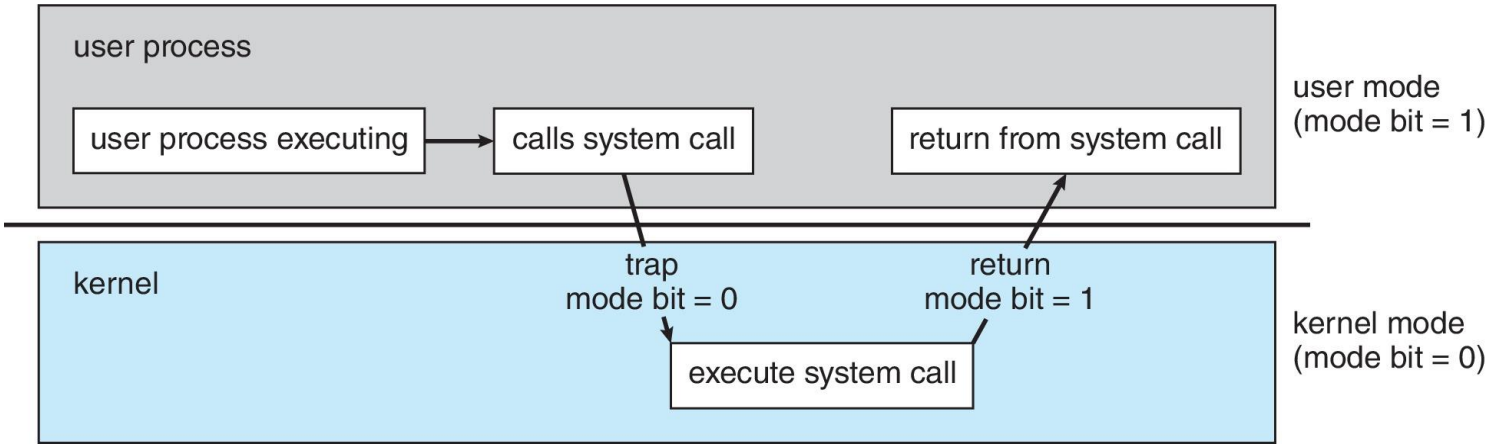
Direct Memory Access Structure

- 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

Dual-mode Operation

- **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.
  - When a user is running ⇨ mode bit is “user”
  - When kernel code is executing ⇨ mode bit is “kernel”
- How do we guarantee that user does not explicitly set the mode bit to “kernel”?
  - System call changes mode to kernel, return from call resets it to user
- Some instructions designated as **privileged**, only executable in kernel mode

Transition from User to Kernel Mode

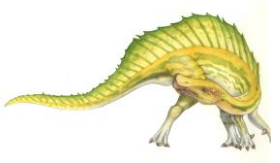


Computer Startup

- **Bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EEPROM, generally known as **firmware**
  - Initializes all aspects of system (CPU registers, Device Controllers, etc.)
  - Loads operating system kernel and starts execution

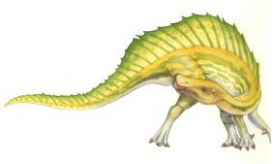
Operating-System Operations

- Bootstrap program – simple code to initialize the system, load the kernel
- Kernel loads
- Starts **system daemons** (services provided outside of the kernel)
- Kernel **interrupt driven** (hardware and software)
  - Hardware interrupt by one of the devices
  - Software interrupt (**exception** or **trap**):
    - Software error (e.g., division by zero)
    - Request for operating system service – **system call**
    - Other process problems include infinite loop, *processes modifying each other or the operating system*



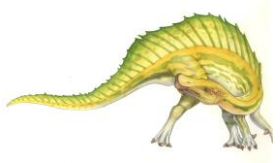
Multiprogramming (Batch system)

- Single user cannot always keep CPU and I/O devices busy
- 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 job has to wait (for I/O for example), OS switches to another job
- Completion time is not very critical in batch processing.
  - Objective is to maximize the CPU utilization.

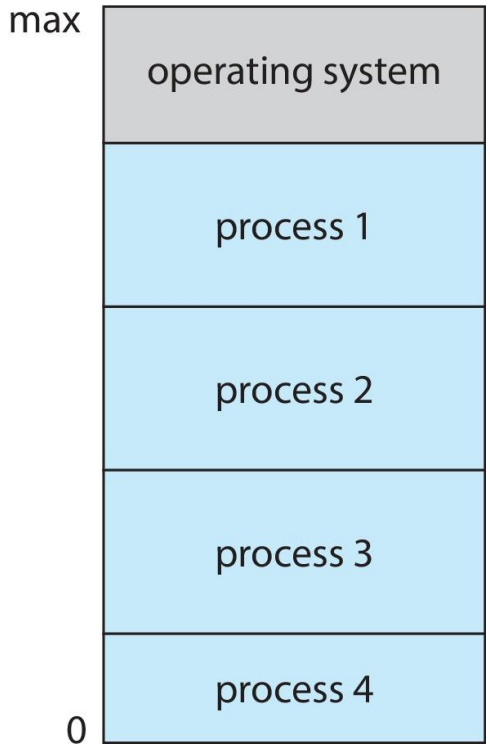


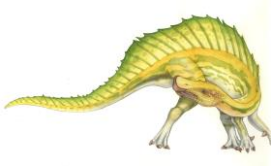
Multitasking (Timesharing)

- A logical extension of Batch systems– the CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing
  - Objective is to minimize the response time.
  - **Response time** should be < 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
  - **Virtual memory** allows execution of processes not completely in memory



Memory Layout for Multiprogrammed System



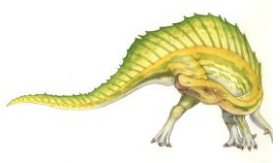


Timer

- Timer to prevent infinite loop (or 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. 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
- 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

The operating system is responsible for the following activities in connection with process management:

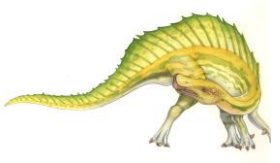
- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

Memory Management

- To execute a program all (or part) of the instructions must be in (primary) 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

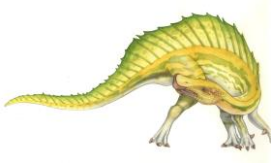
File-system Management

- OS provides uniform, logical view of information storage
  - Abstracts physical properties of storage devices to logical storage unit - **file**
  - Maps files onto physical media (magnetic disk, optical disk, etc)
- File-System management
  - Files usually organized into directories for ease of use.
  - Access control on most systems to determine 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



Mass-Storage Management

- Usually disks used to store data that does not fit in main memory or data that must be kept for a “long” period of time
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
  - Mounting and unmounting
  - Free-space management
  - Storage allocation
  - Disk scheduling
  - Partitioning
  - Protection



Caching

- 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

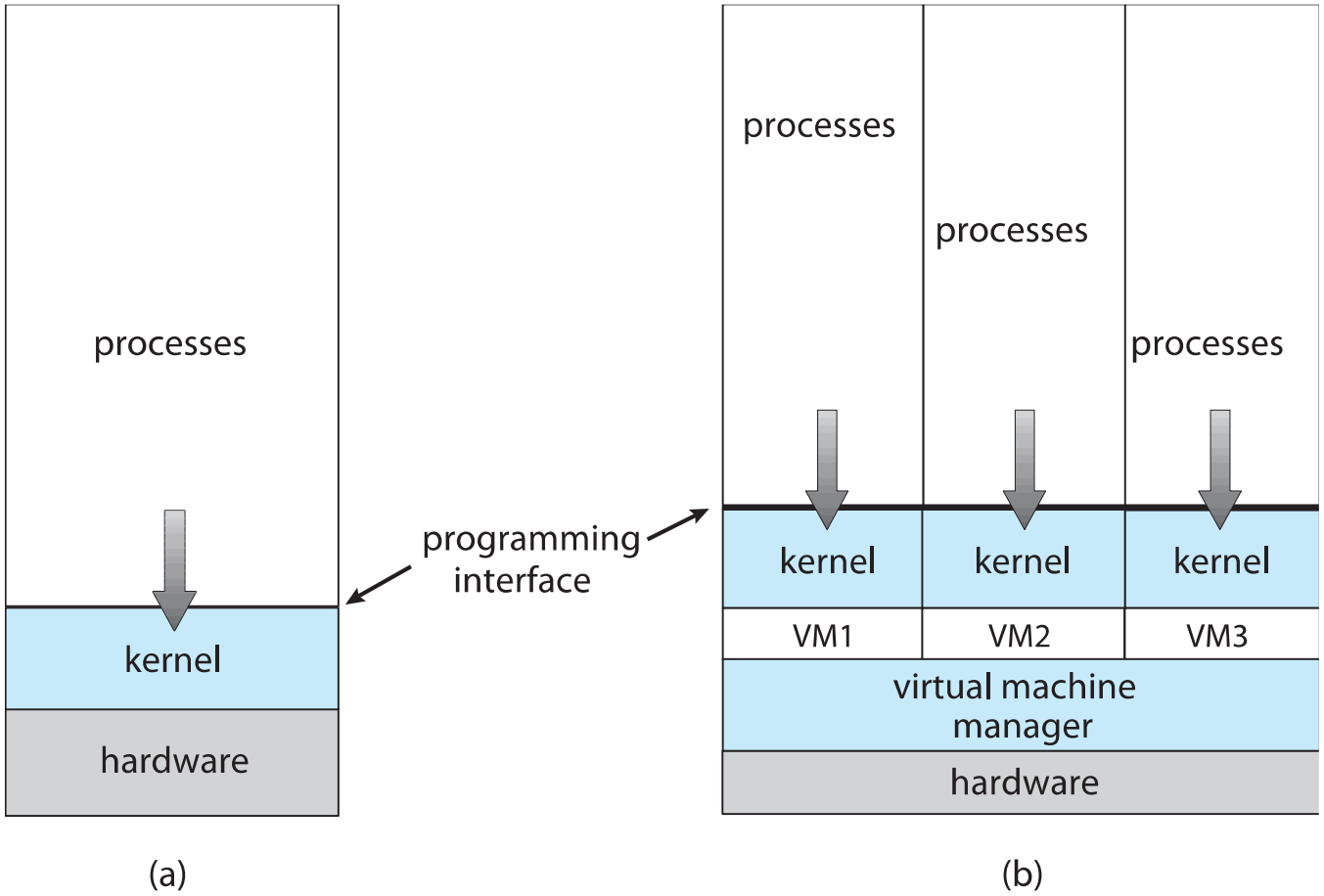
Protection and Security

- **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

Virtualization

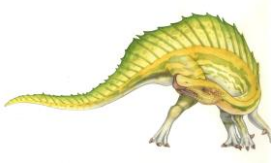
- **Emulation is** simulating computer hardware in software, is typically used when the source CPU type is different from the target CPU type.
  - Apple switched from the IBM Power CPU to the Intel x86 CPU for its desktop and laptop computers, it included an emulator **Rosetta**
- **Virtualization** is a technology that allows us to abstract the hardware of a single computer (the CPU, memory, disk drives, network interface cards etc) into several different execution environments.
- Creates the illusion that each separate environment (virtual machine) is running on its own private computer.
- Virtualization allows operating systems to run as applications within other operating systems.
- **Virtualization** – OS natively compiled for CPU, running **guest** OSes also natively compiled
  - **VMM** (virtual machine Manager) provides virtualization services e.g., Vmware, Virtual Box

Computing Environments - Virtualization

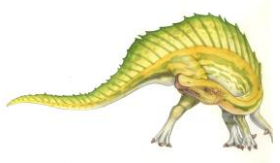


Distributed Systems

- 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



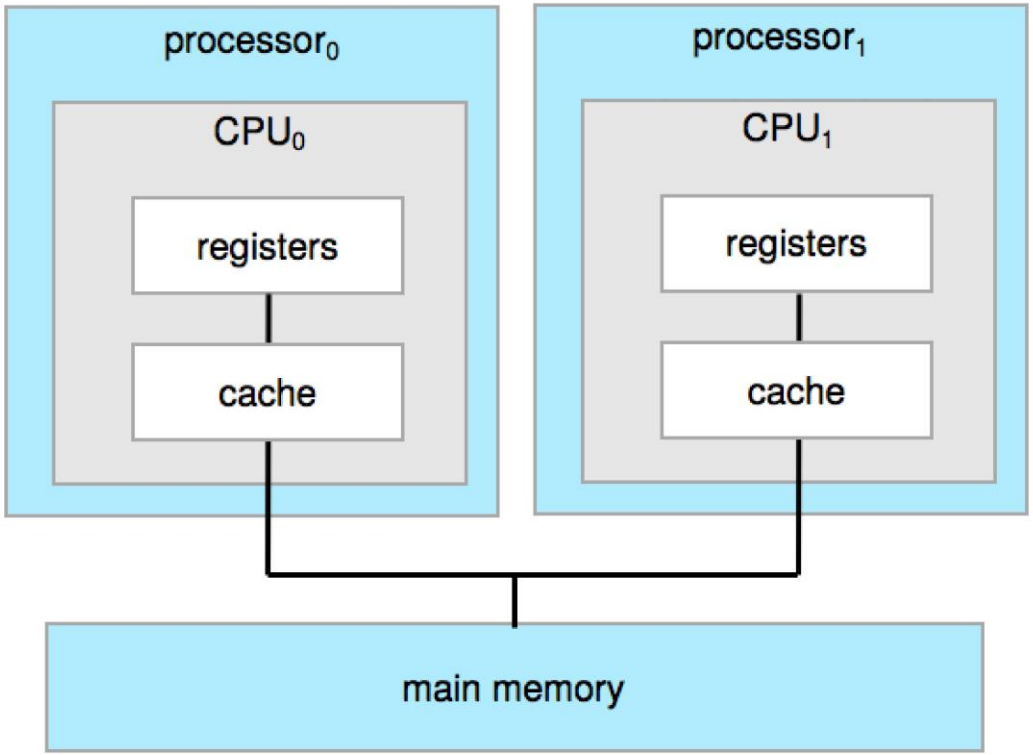
Computer System Architecture

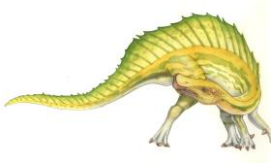


Computer-System Architecture

- Most systems use a single general-purpose processor
  - 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:
    1. **Increased throughput**
    2. **Economy of scale**
    3. **Increased reliability** – graceful degradation or fault tolerance
  - Two types:
    1. **Asymmetric Multiprocessing** – each processor is assigned a specie task.
    2. **Symmetric Multiprocessing** – each processor performs all tasks

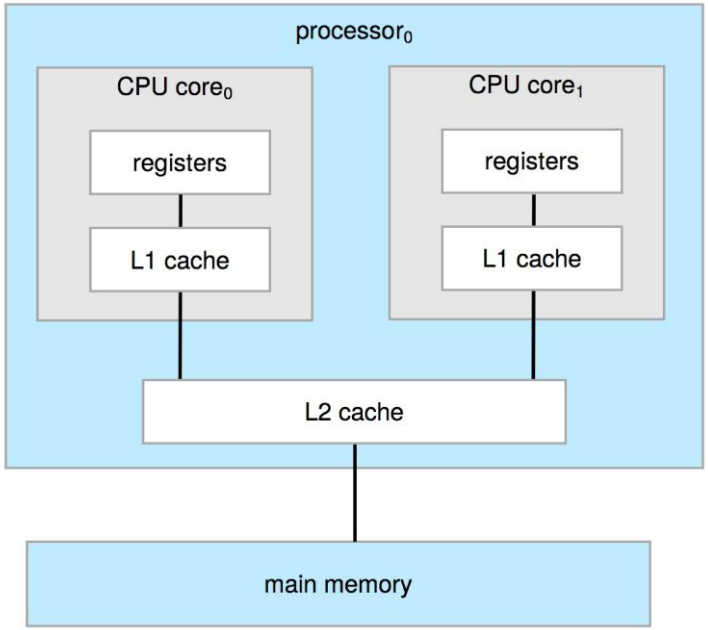
Symmetric Multiprocessing Architecture

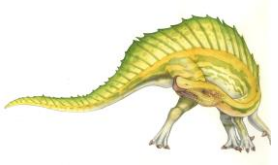




Dual-Core Design

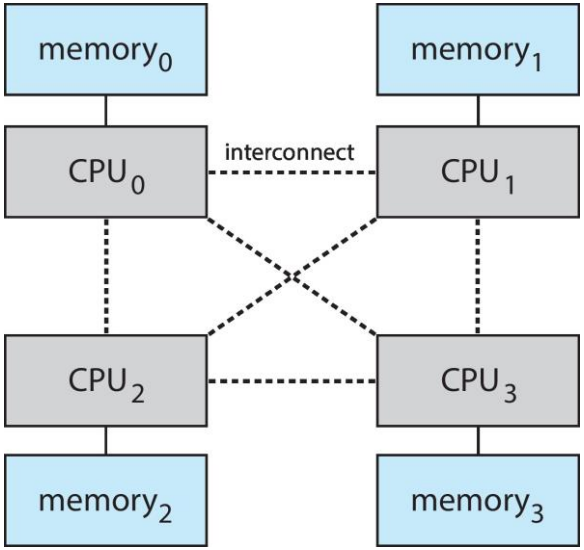
- The definition of multiprocessor has evolved over time and now **includes multicore systems**, in which multiple computing cores reside on a single chip i.e., Multi-chip and **multicore**
- more efficient than multiple chips with single cores because of faster on-chip communication
- multiple cores uses significantly less power than multiple single-core chips, hence suitable for mobile devices

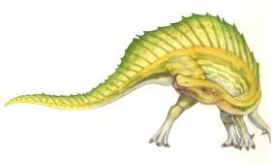




Non-Uniform Memory Access System

- Adding additional CPUs to a multiprocessor system will increase computing; however, the concept does not scale very well
  - contention for the system bus becomes a bottleneck
  - provide each CPU (or group of CPUs) with its own local memory
- Non-uniform memory access
  - The CPUs are connected by a shared system interconnect, so that all CPUs share one physical address space

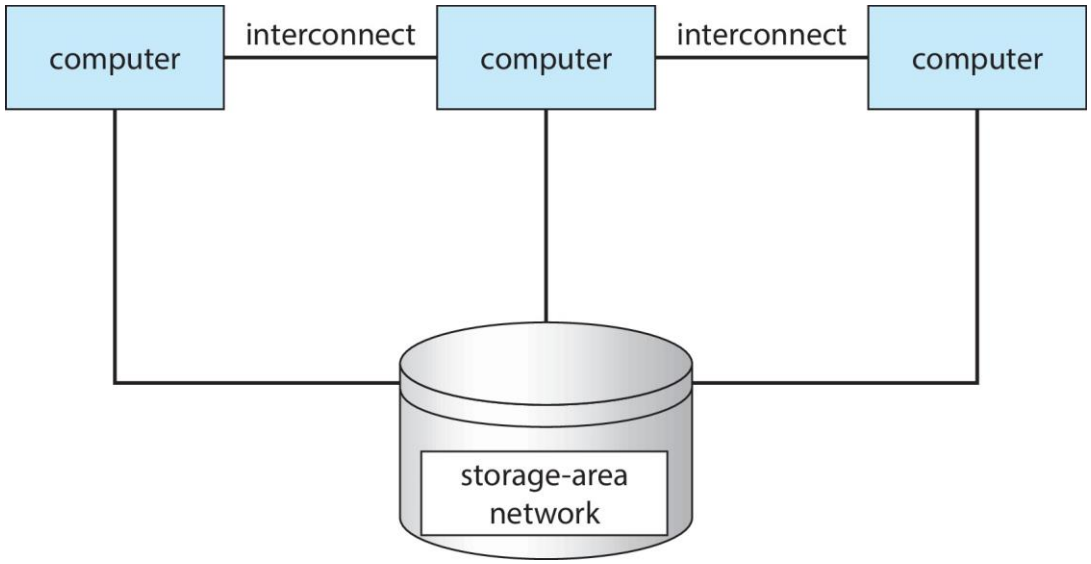


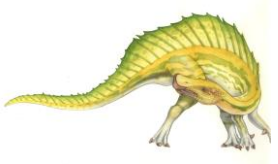


Clustered Systems

- Like multiprocessor systems, but multiple systems working together
- Clustered systems differ from the multiprocessor systems as they are composed of two or more individual (multicore) systems.
  - 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 to monitor the active server
    - **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**
  - Some have **distributed lock manager** (**DLM**) to avoid conflicting operations

Clustered Systems

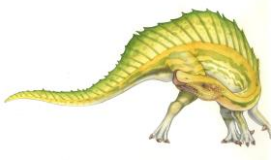




Computer System Environments

Computing Environments

- Traditional
- Mobile
- Client Server
- Peer-to-Peer
- Cloud computing
- Real-time Embedded



Traditional

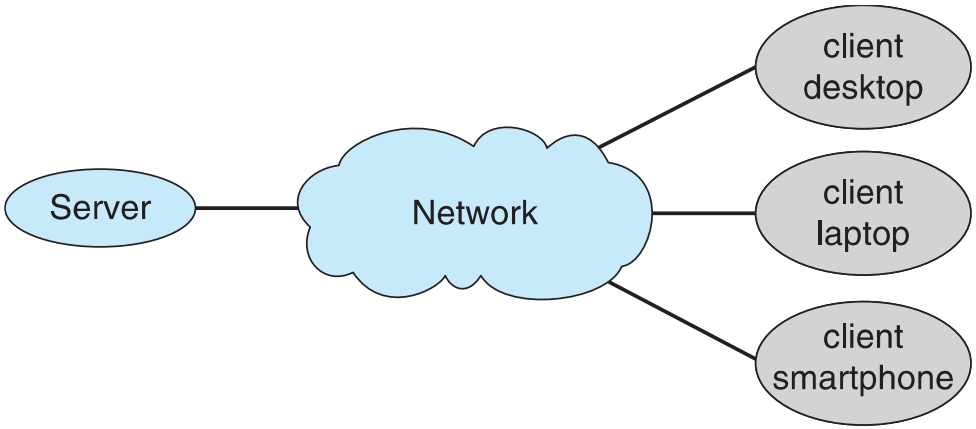
- Stand-alone general-purpose machines
- But blurred as most systems interconnect with others (i.e., the Internet)
- **Portals** provide web access to internal systems
- **Network computers** (**clients**) are like Web terminals
- Mobile computers interconnect via **wireless networks**
- Networking becoming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

Mobile

- Handheld smartphones, tablets, etc.
- 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**

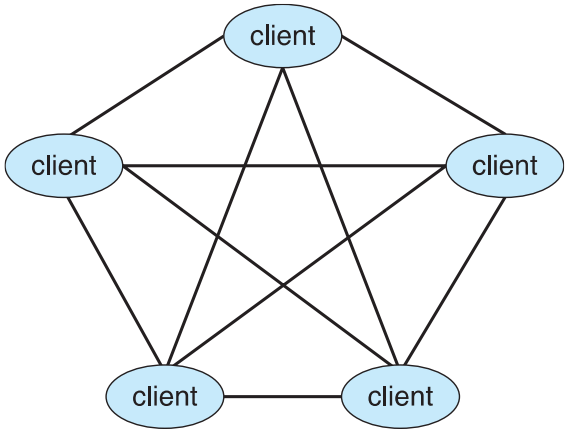
Client Server

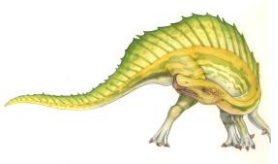
- Client-Server Computing
  - Dumb terminals supplanted by smart PCs
  - Many systems now **servers**, responding to requests generated by **clients**
    - **Compute-server system** provides an interface to client to request services (i.e., database)
    - **File-server system** provides interface for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



Peer-to-Peer

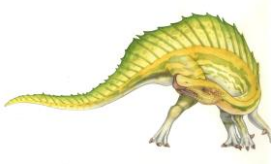
- Another model of distributed system
- P2P does not distinguish clients and servers
  - Instead all nodes are considered peers
  - May each act as client, server or both
  - Node must join P2P network
    - Registers its service with central lookup service on network, or
    - Broadcast request for service and respond to requests for service via ***discovery protocol***
  - Examples include Torrent, Skype





Cloud Computing

- Delivers computing, storage, even apps as a service across a network
- Logical extension of virtualization because it uses virtualization as the base for it functionality.
  - Amazon **EC2** has thousands of servers, millions of virtual machines, petabytes of storage available across the Internet, pay based on usage

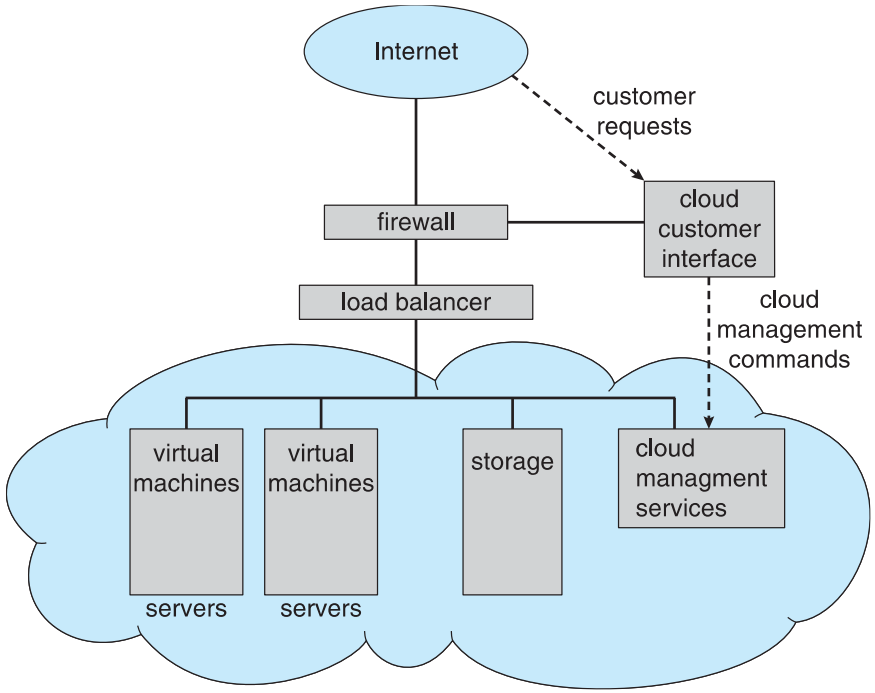


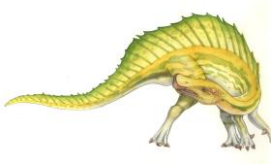
Cloud Computing (Cont.)

- Many types
  - **Public cloud** – available via Internet to anyone willing to pay
  - **Private cloud** – run by a company for the company's own use
  - **Hybrid cloud** – includes both public and private cloud components
  - Software as a Service (**SaaS**) – one or more applications available via the Internet (i.e., word processor)
  - Platform as a Service (**PaaS**) – software stack ready for application use via the Internet (i.e., a database server)
  - Infrastructure as a Service (**IaaS**) – servers or storage available over Internet (i.e., storage available for backup use)

Cloud Computing (cont.)

- 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





Real-Time Embedded Systems

- Real-time embedded systems most prevalent form of computers
  - Vary considerable, special purpose, limited purpose OS, **real-time OS**
  - Found everywhere, from car engines and manufacturing robots to optical drives and microwave ovens.
  - Some have OSes, some perform tasks without an OS
    - application-specific integrated circuits (**ASICs**)
- Real-time OS has well-defined fixed time constraints
  - Processing ***must*** be done within constraint
  - Correct operation only if constraints met

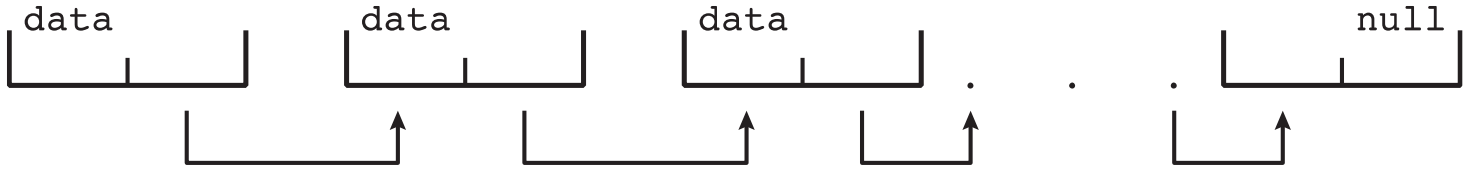
Free and Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary **closed-source** and **proprietary**
- Counter to the **copy protection** and **Digital Rights Management** (**DRM**) movement
- Started by **Free Software Foundation** (**FSF**), which has "copyleft" **GNU Public License** (**GPL**)
  - Free software and open-source software are two different ideas championed by different groups of people
    - **https://www.gnu.org/philosophy/open-source-misses-the-point.en.html**
- 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 on many platforms - http://www.virtualbox.com)
  - Use to run guest operating systems for exploration

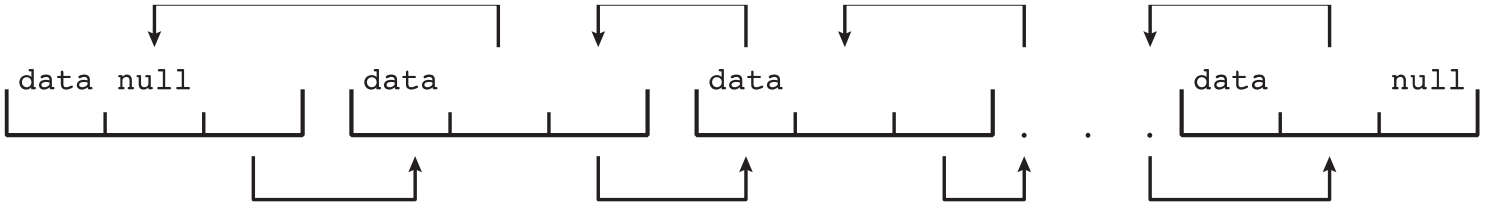
Kernel Data Structure

Kernel Data Structures

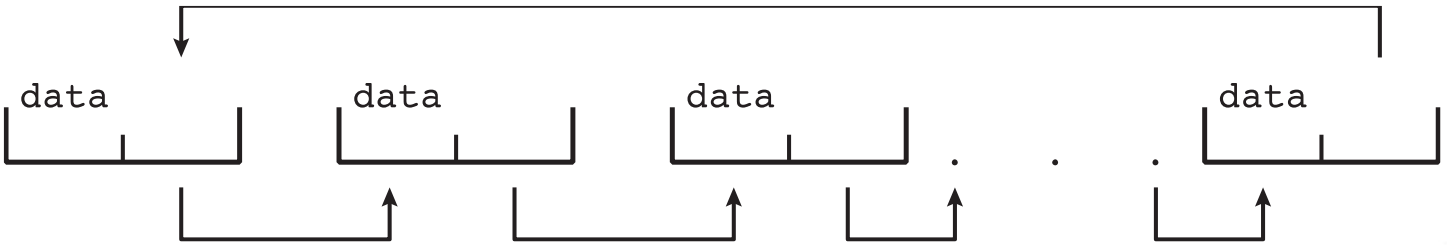
- Many similar to standard programming data structures
- ***Singly linked list***



- ***Doubly linked list***

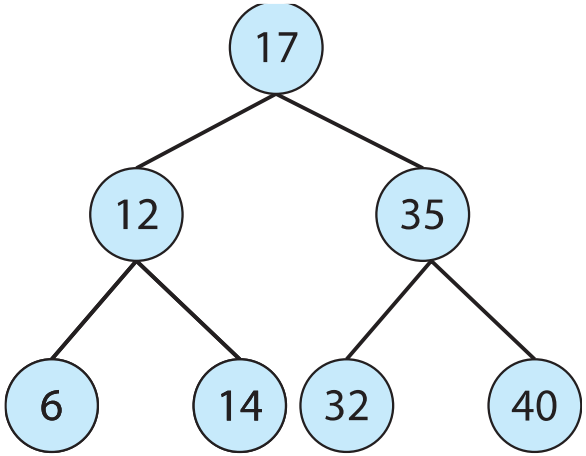


- ***Circular linked list***



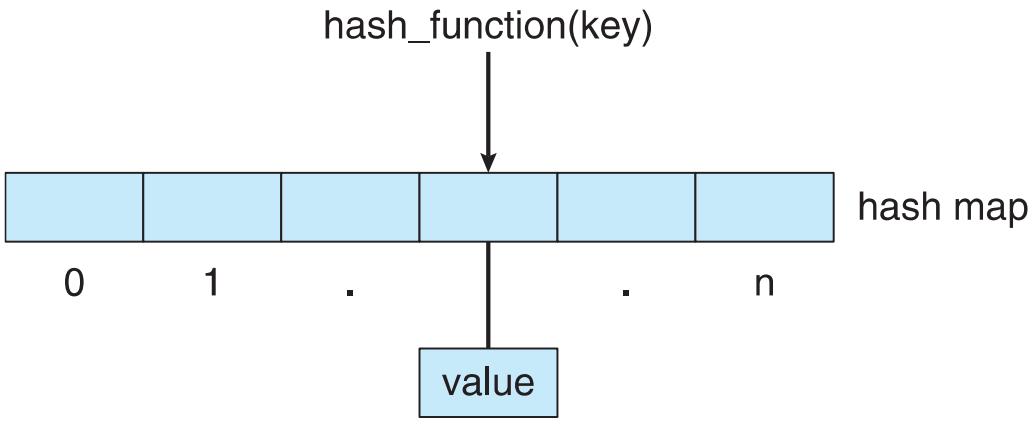
Kernel Data Structures

- **Binary search tree**
  left <= right
  - Search performance is *O(n)*
  - **Balanced binary search tree** is *O(lg n)*

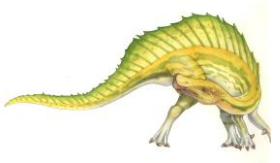


Kernel Data Structures

- **Hash function** can create a **hash map**



- **Bitmap** – string of *n* binary digits representing the status of *n* items
- Linux data structures defined in ***include*** files `<linux/list.h>`, `<linux/kfifo.h>`, `<linux/rbtree.h>`



End of Chapter 1