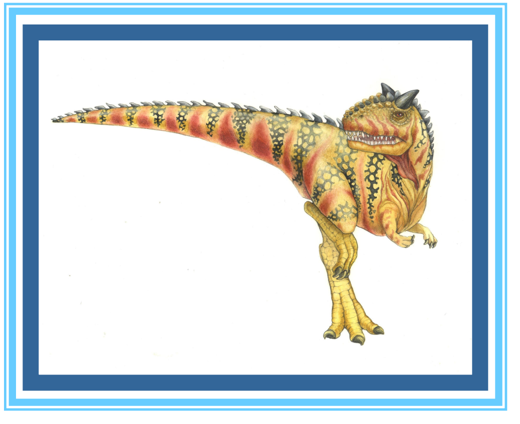
Chapter 1: Introduction



Chapter 1: Introduction

- What is an Operating System?
- Computer-System Organization
- Operating-System Structure
- Operating-System Operations
- Process Management
- Memory Management
- Storage Management
- Protection and Security
- Kernel Data Structures
- Computing Environments
- Open-Source Operating Systems
- Computer-System Architecture

Objectives

- To describe the basic organization of computer systems
- To provide a grand tour of the major components of operating systems
- To give an overview of the many types of computing environments
- To explore several open-source operating systems

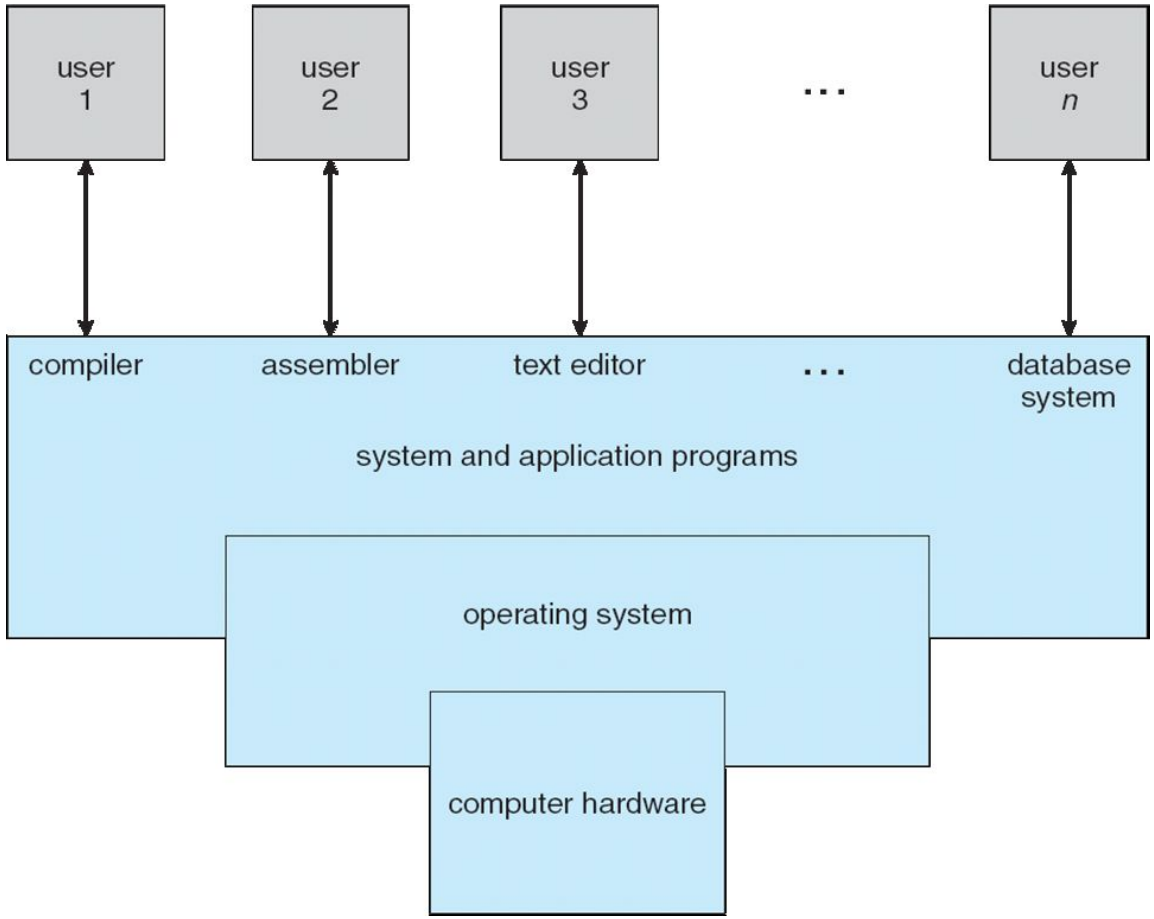
What is an Operating System?

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

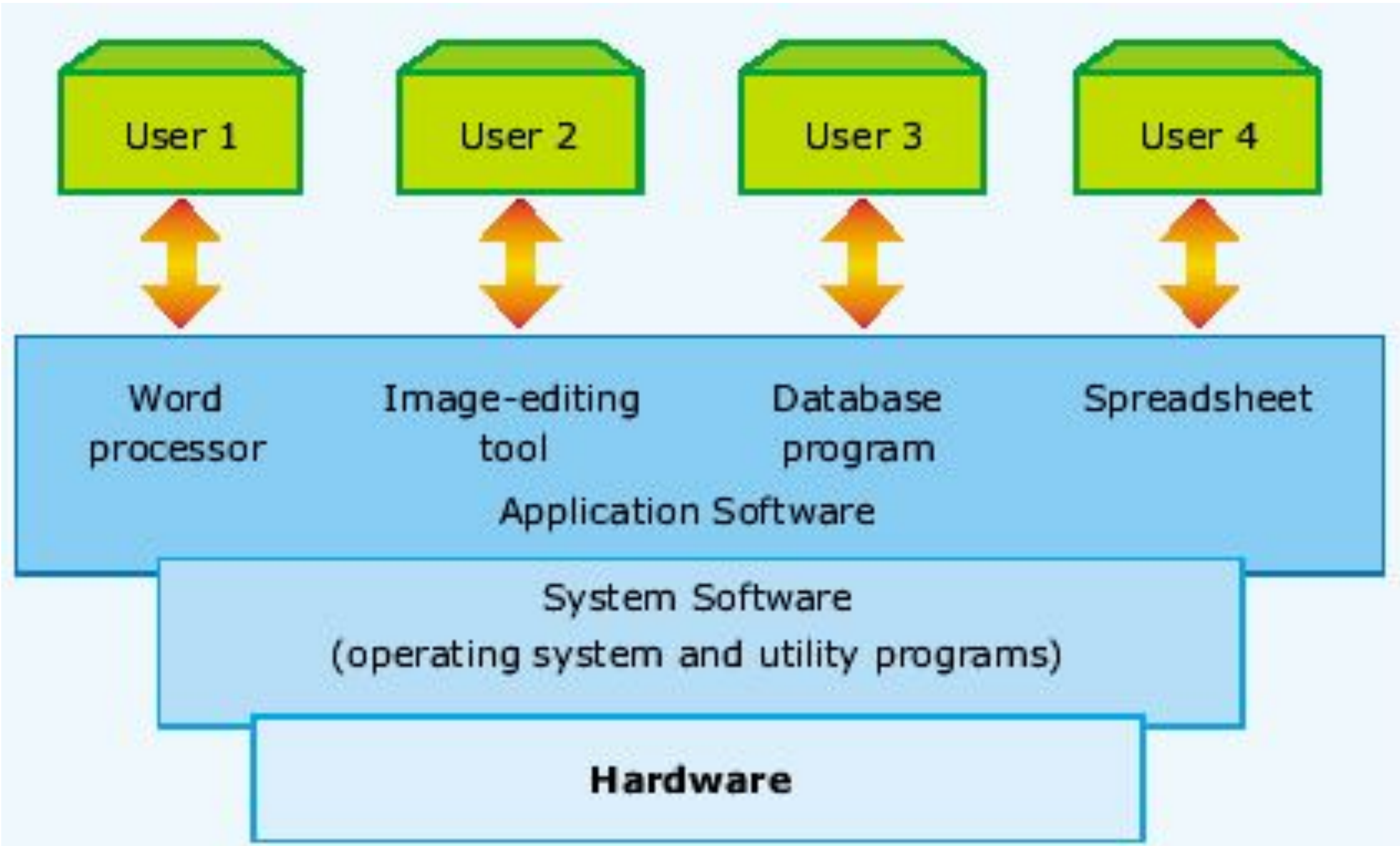
Computer System Structure

- 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

Four Components of a Computer System



Abstract View of system components



Operating system - Controls and manage the hardware and allows the user to run other applications
Utility programs - Perform a specific task usually related to managing system resources.

Utility Program	Uses
Backup utility	To make a duplicate copy of selected files or the entire hard disk
Data-recovery utility	To restore data that has been corrupted and return backed up files to their original form
Antivirus software	To detect and clean viruses from the memory and storage
Program uninstaller utility	To remove an application and all associated system file entries completely
File compression utility	To reduce the size of a file so that less space is required to store the data
Disk scanner program	To detect and correct logical damages on a hard disk or floppy disk
Disk defragmenter utility	To reorganise all the scattered files on the hard disk so that data can be accessed more quickly

What Operating Systems Do

- The operating system controls the hardware and coordinates its use among the various application programs for the various users.
- We can also view a computer system as consisting of hardware, software, and data.
- The operating system provides the means for proper use of these resources in the operation of the computer system.
- An operating system is similar to a government. Like a government, it performs no useful function by itself. **It simply provides an environment within which other programs can do useful work.**
- To understand more fully the operating system's role, we explore operating systems from two viewpoints:
  - The user
  - The system.

User View

The user's view of the computer varies according to the interface being used

- **Single user computers** (e.g., PC, workstations). Such systems are designed for one user to monopolize its resources. The goal is to **maximize the work (**or play) that the user is performing. the operating system is designed mostly for **ease of use** and **good performance.**
- **Multi user computers** (e.g., mainframes, computing servers). These users share resources and may exchange information. The operating system in such cases is designed to maximize resource utilization -- to assure that all available CPU time, memory, and I/O are used efficiently and that no individual users takes more than their fair share.

User View (Cont.)

- **Handheld computers** (e.g., smartphones and tablets). The user interface for mobile computers generally features a **touch screen.** The systems are resource poor, optimized for usability and battery life.
- **Embedded computers** (e.g., computers in home devices and automobiles) The user interface may have numeric keypads and may turn indicator lights on or off to show status. The operating systems are designed primarily to run without user intervention.

System View

From the computer's point of view, the operating system is the program most intimately involved with the hardware. There are two different views:

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

Defining Operating System

No universally accepted definition of **what** an OS:

- Operating systems exist to offer a reasonable way to solve the problem of creating a usable computing system.
- The fundamental goal of computer systems is to execute user programs and to make solving user problems easier.
- Since bare hardware alone is not particularly easy to use, application programs are developed.
  - These programs require certain common operations, such as those controlling the I/O devices.
  - The common functions of controlling and allocating resources are brought together into one piece of software: the **operating system**.

Defining Operating System (Cont.)

No universally accepted definition of what is **part** of the OS:

- A simple viewpoint is that it includes everything a vendor ships when you order the operating system. The features that are included vary greatly across systems:
  - Some systems take up less than a megabyte of space and lack even a full-screen editor,
  - Some systems require gigabytes of space and are based entirely on graphical windowing systems.

Defining Operating System (Cont.)

No universally accepted definition of what is **part** of the OS:

- A more common definition, and the one that we usually follow, is that the operating system is the one program running at all times on the computer -- usually called the **kernel**.
- Along with the kernel, there are two other types of programs:
  - System programs, which are associated with the operating system but are not necessarily part of the kernel.
  - Application programs, which include all programs not associated with the operation of the system.

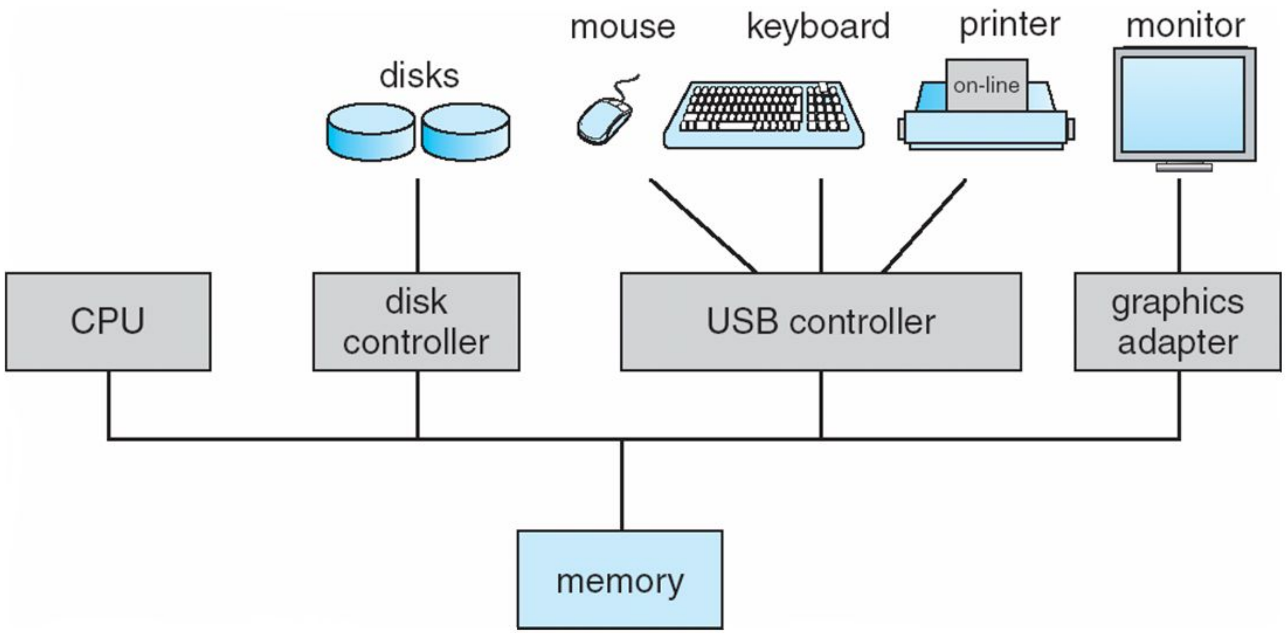
Defining Operating System (Cont.)

- The emergence of mobile devices, have resulted in an increase in the number of features that constituting the operating system.
- Mobile operating systems often include not only a core kernel but also **middleware** -- a set of software frameworks that provide additional services to application developers.
- For example, each of the two most prominent mobile operating systems -- Apple's iOS and Google's Android -- feature a core kernel along with middleware that supports databases, multimedia, and graphics (to name only a few).

Computer-System Organization

- A modern general-purpose computer system consists of one or more CPUs and a number of device controllers connected through a common bus that provides access to shared memory.
- Each device controller is in charge of a specific type of device (for example, disk drives, audio devices, or video displays). Each device controller has a local buffer.
- CPU moves data from/to main memory to/from local buffers.
- The CPU and the device controllers can execute in parallel, competing for memory cycles. To ensure orderly access to the shared memory, a memory controller synchronizes access to the memory.

Modern Computer System



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 Operation

- Once the kernel is loaded and executing, it can start providing services to the system and its users.
- Some services are provided outside of the kernel, by system programs that are loaded into memory at boot time to become **system processes**, or **system daemons** that run the entire time the kernel is running.
- On UNIX, the first system process is **init** and it starts many other daemons. Once this phase is complete, the system is fully booted, and the system waits for some event to occur.
- The occurrence of an event is usually signaled by an **interrupt.**

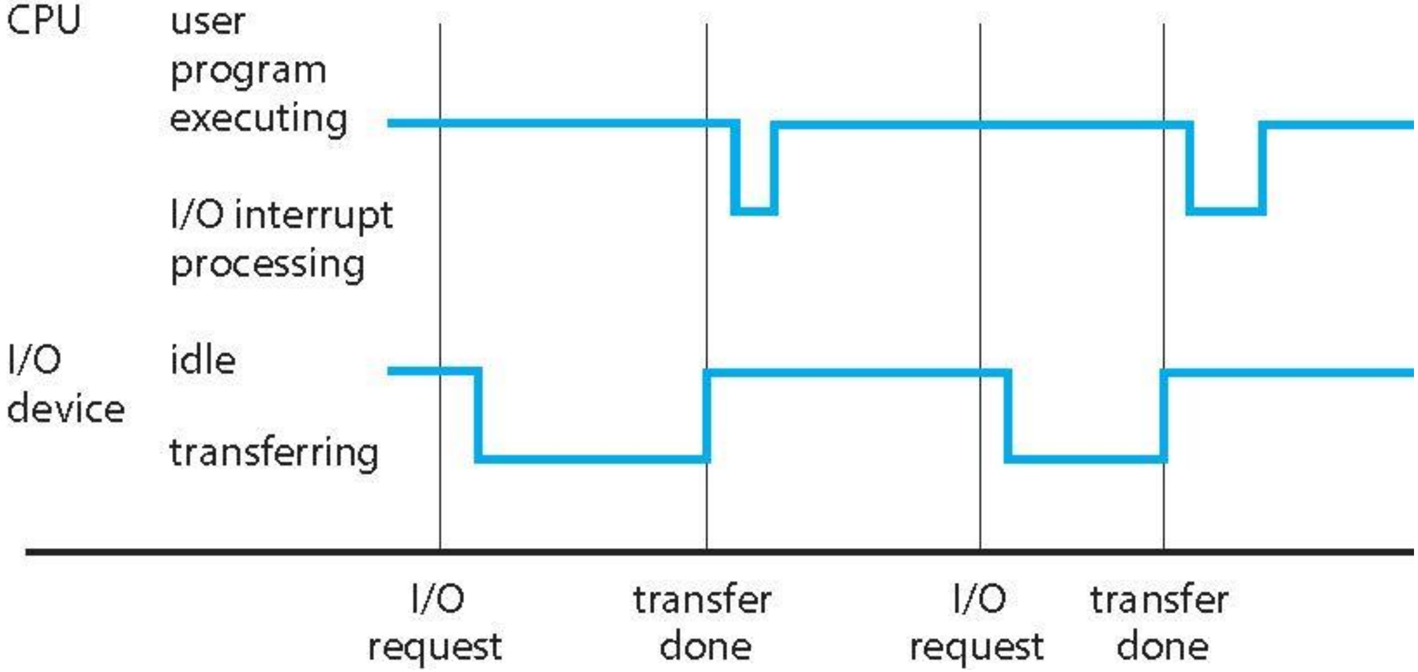
Interrupts

- There are two types of interrupts:
  - **Hardware** -- a device may trigger an interrupt by sending a signal to the CPU, usually by way of the system bus.
  - **Software** -- a program may trigger an interrupt by executing a special operation called a **system call.**
- A software-generated interrupt (sometimes called **trap** or **exception**) is caused either by an error (e.g., divide by zero) or a user request (e.g., an I/O request).
- An operating system is **interrupt driven.**

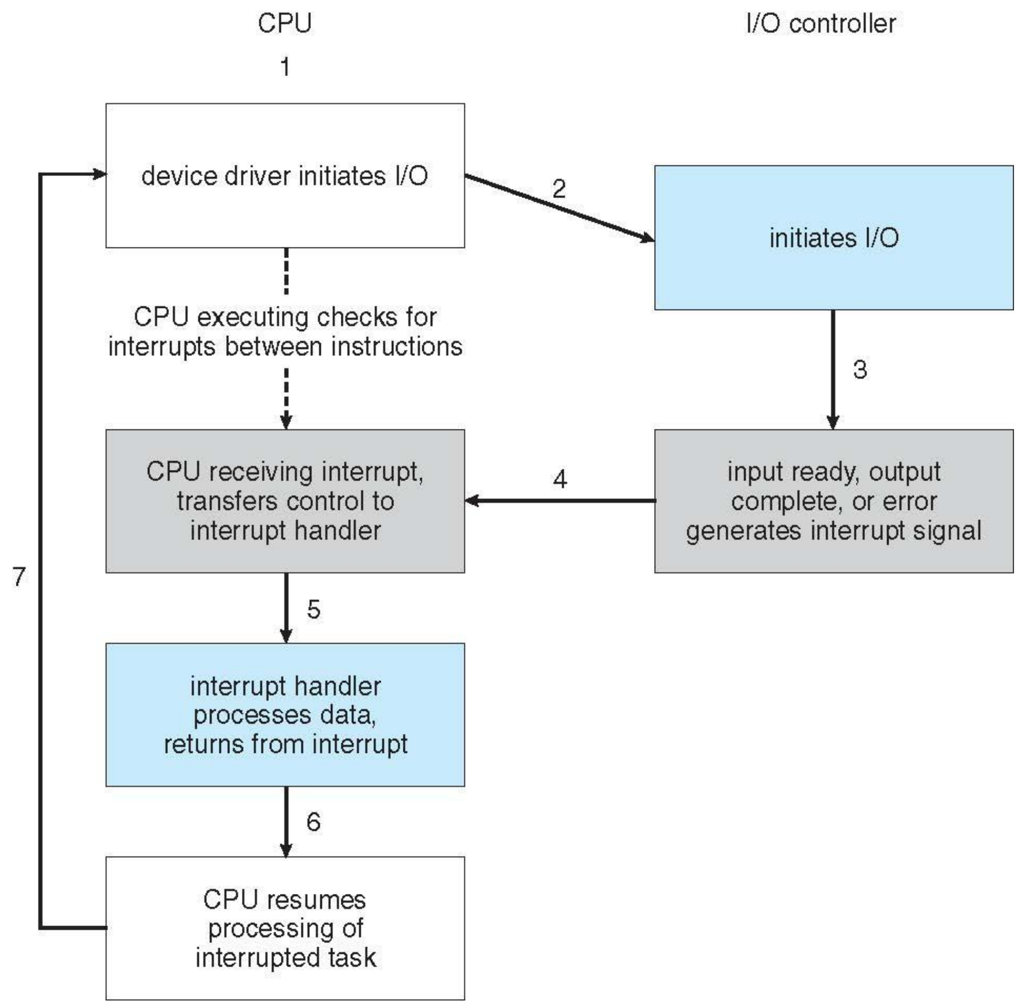
Common Functions of Interrupts

- When an interrupt occurs, the operating system preserves the state of the CPU by storing the registers and the program counter
- Determines which type of interrupt has occurred and transfers control to the interrupt-service routine.
- An interrupt-service routine is a collection of routines (modules), each of which is responsible for handling one particular interrupt (e.g., from a printer, from a disk)
- The transfer is generally through the **interrupt vector**, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction.

Interrupt Timeline



Interrupt-driven I/O cycle.



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

Storage Structure

- Main memory – the only large storage media that the CPU can access directly
  - **Random access**
  - Typically **volatile**
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity
  - Hard 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
  - **Solid-state disks** – faster than hard disks, nonvolatile
    - Various technologies
    - Becoming more popular
- Tertiary storage

Storage Definition

- The basic unit of computer storage is the **bit**. A bit can contain one of two values, 0 and 1. All other storage in a computer is based on collections of bits.
- A **byte** is 8 bits, and on most computers it is the smallest convenient chunk of storage.
- A less common term is **word**, which is a given computer architecture’s native unit of data. A word is made up of one or more bytes.

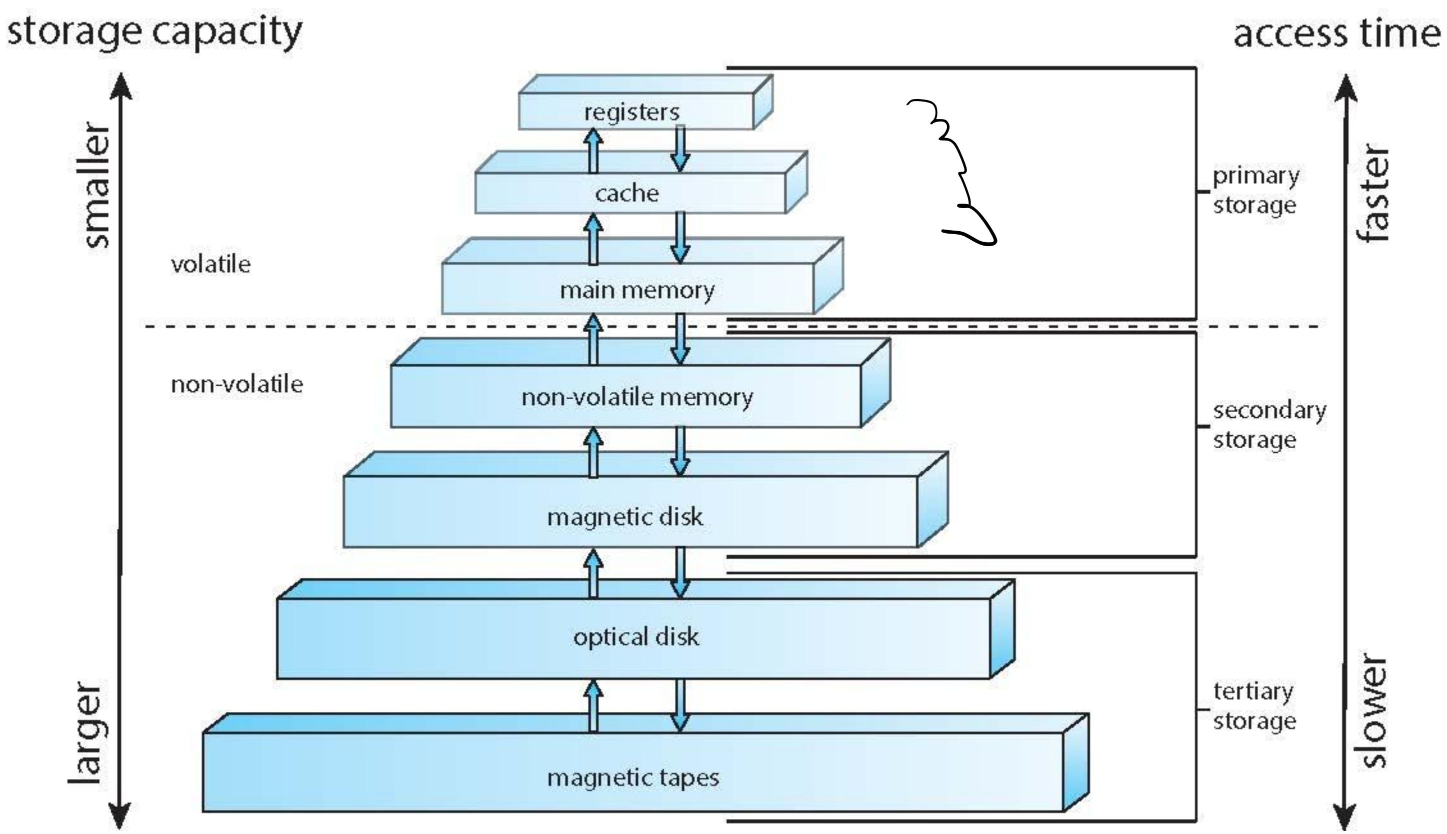
Storage Definition (Cont.)

- Computer storage, along with most computer throughput, is generally measured and manipulated in bytes and collections of bytes.
  - A **kilobyte**, or **KB**, is 1,024 bytes
  - a **megabyte**, or **MB**, is $1{,}024^2$ bytes
  - a **gigabyte**, or **GB**, is $1{,}024^3$ bytes
  - a **terabyte**, or **TB**, is $1{,}024^4$ bytes
  - a **petabyte**, or **PB**, is $1{,}024^5$ bytes
  - exabyte, zettabyte, yottabyte
- Computer manufacturers often round off these numbers and say that a megabyte is 1 million bytes and a gigabyte is 1 billion bytes. Networking measurements are an exception to this general rule; they are given in bits (because networks move data a bit at a time).

Storage Hierarchy

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility
- **Caching** – copying information from "slow" storage into faster storage system;
  - Main memory can be viewed as a cache for secondary storage
- **Device Driver** for each device controller to manage I/O
  - Provides uniform interface between controller and kernel

Storage-device hierarchy



I/O Structure

- A general-purpose computer system consists of CPUs and multiple device controllers that are connected through a common bus.
- Each device controller is in charge of a specific type of device. More than one device may be attached. For instance, seven or more devices can be attached to the **small computer-systems interface** (SCSI) controller.
- A device controller maintains some local buffer storage and a set of special-purpose registers.
- The device controller is responsible for moving the data between the peripheral devices that it controls and its local buffer storage.
- Typically, operating systems have a **device driver** for each device controller. This device driver understands the device controller and provides the rest of the operating system with a uniform interface to the device.

I/O Structure (Cont.)

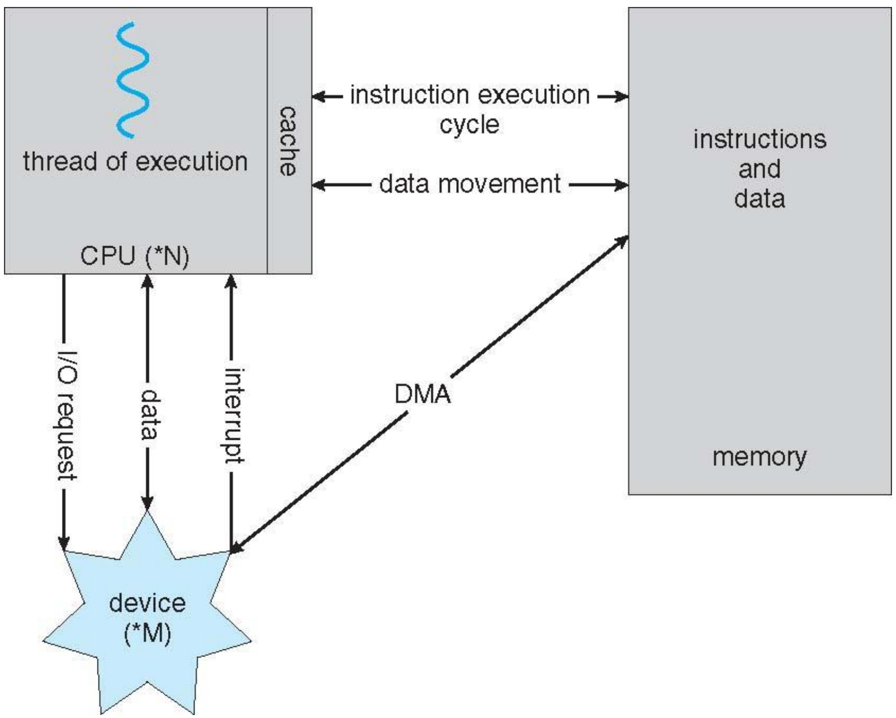
- To start an I/O operation, the device driver loads the appropriate registers within the device controller.
- The device controller, in turn, examines the contents of these registers to determine what action to take (such as “read” a character from the keyboard).
- The controller starts the transfer of data from the device to its local buffer. Once the transfer of data is complete, the device controller informs the device driver via an interrupt that it has finished its operation.
- The device driver then returns control to the operating system, possibly returning the data or a pointer to the data if the operation was a read.
- For other operations, the device driver returns status information.

Direct Memory Access Structure

- Interrupt-driven I/O is fine for moving small amounts of data but can produce high overhead when used for bulk data movement such as disk I/O.
- To solve this problem, **direct memory access** (DMA) is used.
  - After setting up buffers, pointers, and counters for the I/O device, the device controller transfers an entire block of data directly to or from its own buffer storage to memory, with no intervention by the CPU.
  - Only one interrupt is generated per block, to tell the device driver that the operation has completed. While the device controller s performing these operations, the CPU is available to accomplish other work.
- Some high-end systems use switch rather than bus architecture. On these systems, multiple components can talk to other components concurrently, rather than competing for cycles on a shared bus. In this case, DMA is even more effective. The figure in next slide shows the interplay of all components of a computer system.

How a Modern Computer Works

A von Neumann architecture and a depiction of the interplay of all components of a computer system.



Overview and History

- Operating System driven by Cost of resources

- Much of operating system history driven by relative cost factors of hardware and people. In the beginning:
  - Expensive Hardware, Cheap People (Phase I)
    - Goal: maximize hardware utilization.
  - Now: Cheap Hardware, Expensive People (Phase II)
    - Goal: make it easy for people to use computer.

- Relative costs drive the goals of the operating system.

Phase – I, Hardware is costly

- Computers in earlier days of their inception were very bulky, large machines usually run from a console.
- I/O devices consisted of card readers, tape drives and line printers.
- Direct user interaction with the system did not exist.
- Users made a job consisting of programs, data and control information. The job was submitted to an operator who would execute the job on the computer system from the front panel switches(one instruction at a time) from a paper tape or punched cards. Then appropriate buttons would be pushed to set the starting address and to start the execution of the program. The output appeared after minutes, hours or sometimes days. O/P was printed or was punched onto cards or paper tape for later printing. The user collected the output from the operator, which also included a memory dump.

The operating system was very simple and its major task was to transfer control from one job to another. The operating system was resident in memory

Computer-System Architecture

- **Single general-purpose processor**
  - CPU (for executing all instructions) + special-purpose processors like device specific disk, I/O controllers with limited instruction set

- **To speed up processing** - jobs with the same needs were batched together and executed as a group - **batch operating systems.**
- But still the CPU was often idle
  - disparity between operating speeds of electronic devices like the CPU and the mechanical I/O devices. CPU operates in the microsecond / nanosecond ranges whereas I/O devices work in the second / minute range.
  - With improvements in technology, I/O devices became faster but CPU speeds became even faster. So the problem of disparity in operating speeds only widened.

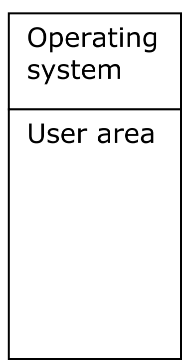


Figure 2.1: Memory layout for simple batch system

Maximizing machine utilization - Spooling

- Solution: Store jobs on a disk **(spooling)**, have computer read them in one at a time and execute them. Build a batch monitor. Big change in computer usage: debugging now done offline from print outs and memory dumps. No more instant feedback.

- **The introduction of disks brought in the concept of spooling - Instead of reading from slow input devices like card readers into the computer memory and then processing the job, the input is first read into the disk. When the job is processed or executed, the input is read directly from the disk. Similarly when a job is executed for printing, the output is written into a buffer on the disk and actually printed later. This form of processing is known as spooling an acronym for Simultaneous Peripheral Operation On Line.**
- Spooling overlaps I/O of one job with the computation of other jobs. For example, spooler may be reading the input of one job while printing the output of another and executing a third job increases the performance of the system by allowing both a faster CPU and slower I/O devices to work at higher operating rates

simultaneous peripheral operations on-line

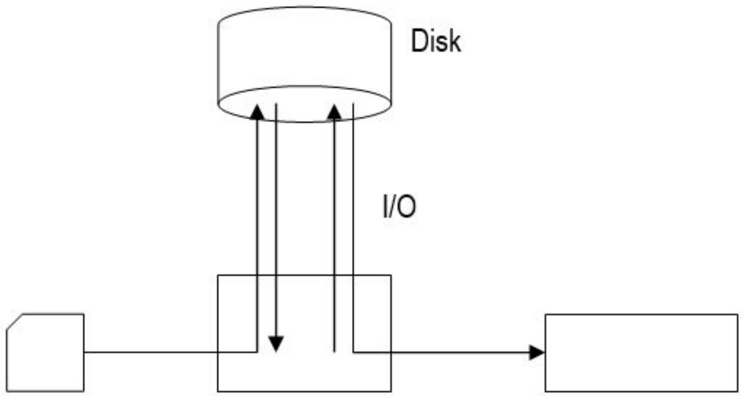


Figure2.2: Spooling

Mainframe Systems

- Reduce setup time by batching similar jobs
- Automatic job sequencing – automatically transfers control from one job to another. First rudimentary operating system.
- Resident monitor
  - initial control in monitor
  - control transfers to job
  - when job completes control transfers back to monitor

Utilization of CPU & I/O

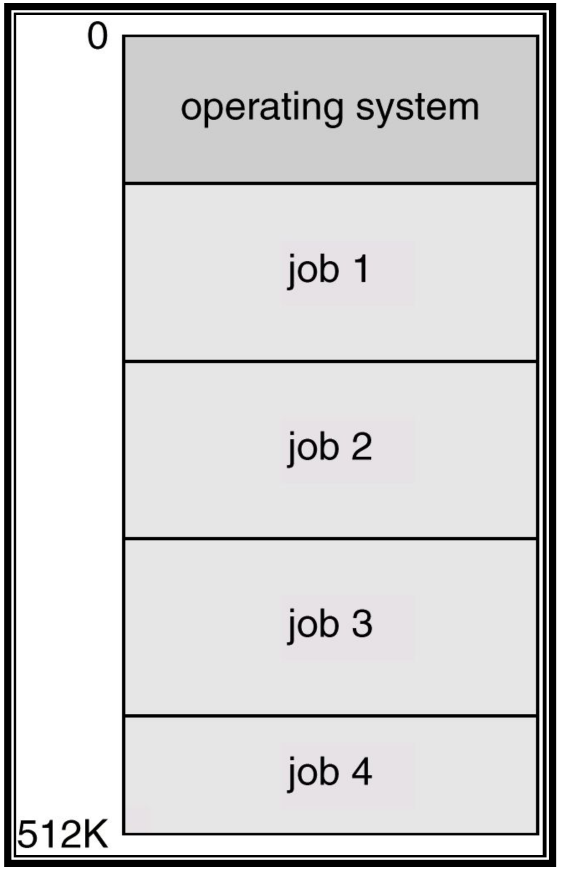
- Problem: one job can't keep both CPU and I/O devices busy. Get poor utilization either of CPU or I/O devices.

- Solution: multiprogramming - several jobs share system. Dynamically switch from one job to another when the running job does I/O.

- Big issue: protection. Don't want one job to affect the results of another. Memory protection and relocation added to hardware, OS must manage new hardware functionality. OS starts to become a significant software system.

Multiprogrammed Batch Systems

Several jobs are kept in main memory at the same time, and the CPU is multiplexed among them.



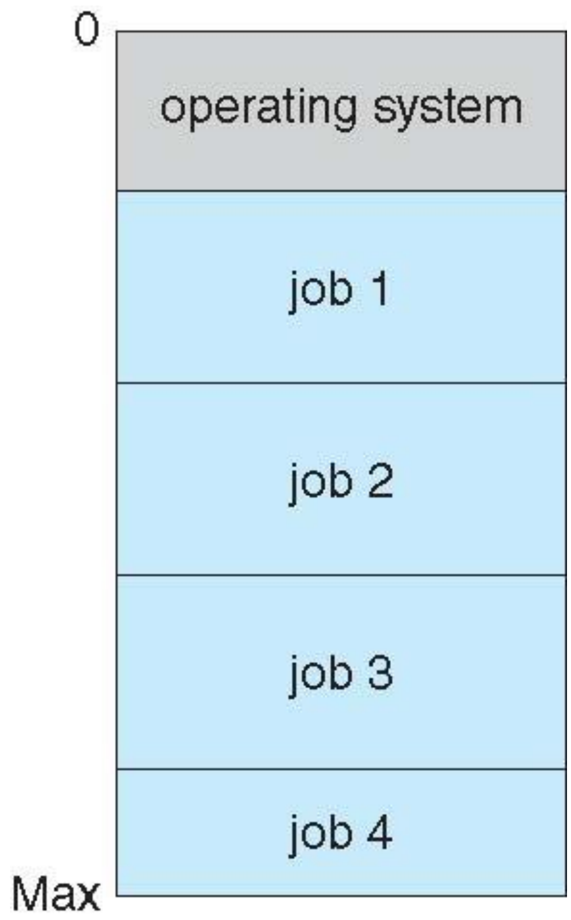
Multiprogrammed System

- **Multiprogramming** organizes jobs (code and data) so CPU always has one to execute
- A subset of total jobs in system is kept in memory
- Batch systems:
  - One job selected and run via **job scheduling**
  - When it has to wait (for I/O for example), OS switches to another job
- Interactive systems:
  - Logical extension of batch systems -- CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing

Interactive Systems

- **Response time** should be < 1 second
- Each user has at least one program executing in memory. Such a program is referred to as a **process**
- If several processes are ready to run at the same time, we need to have **CPU scheduling.**
- If processes do not 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



Computer-System Architecture

Multiprocessors systems

- Also known as **parallel systems**, **tightly-coupled systems**
- **Share computer bus, timer and peripherals**
- **Started with servers but also in laptop/desktops and smart phones also.**
- Advantages include:
  - **Increased throughput –** The speed up ratio is **not N** due to overhead of perfect communication/cooperation + contention for shared resources lowers the expected gain
  - **Economy of scale –** Multiple processors working on certain data has lesser cost due to sharing of peripherals, mass storage
  - **Increased reliability** – failure does not halt the system. Graceful-degradation/fault-tolerance
- Two types:
  - **Symmetric Multiprocessing** – each processor are peers having all their own resources – register, cache and performs all tasks AIX - commercial Unix IBM machine. N CPU => N processes
  - **Asymmetric Multiprocessing** – each processor is assigned a specific task by the Boss processor.

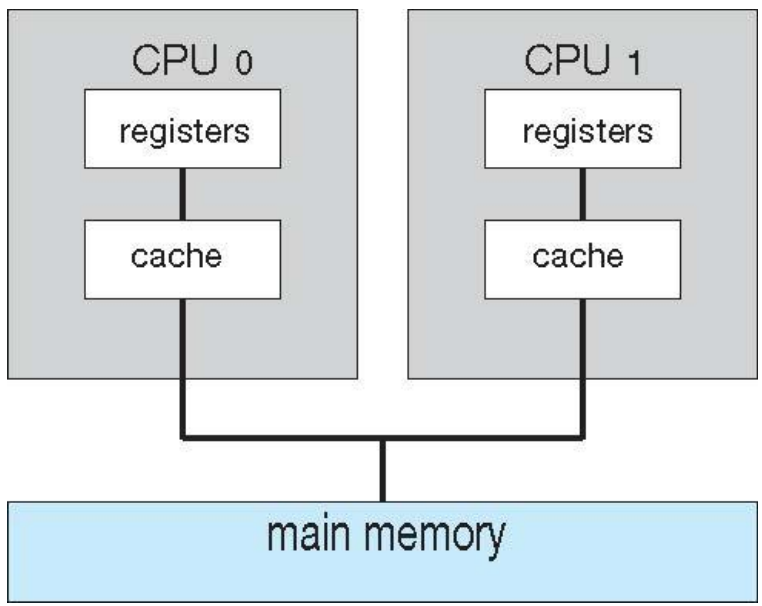
Symmetric Multiprocessing Architecture

CPU has an integrated memory controller => adding CPUs increase the amount of memory addressable causing a system to change its memory access model from **uniform memory access (UMA) to non-uniform memory access (NUMA).**

UMA when access to any RAM from any CPU takes the same amount of time.

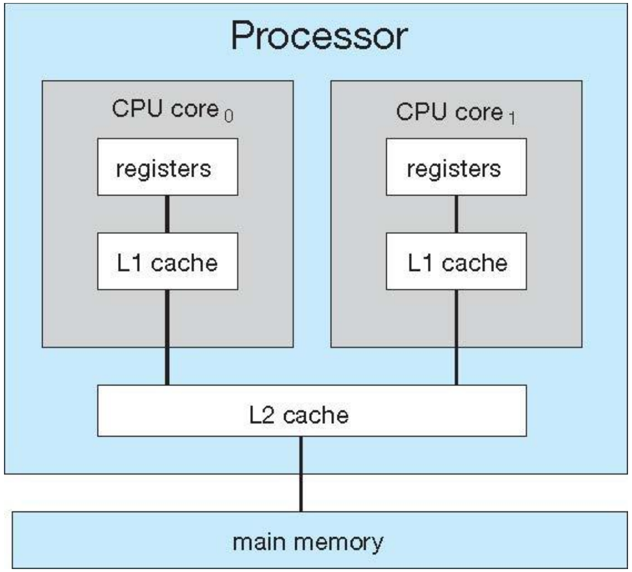
With NUMA, some parts of memory may take longer to access than other parts, creating a performance penalty.

NUMA penalty is minimized by resource management



Multicore Systems

- Most CPU design now includes multiple computing cores on a single chip. Such multiprocessor systems are termed **multicore**.
- Multicore systems can be more efficient than multiple chips with single cores because:
  - On-chip communication is faster than between-chip communication.
  - One chip with multiple cores uses significantly less power than multiple single-core chips, an important issue for laptops as well as mobile devices.
- Note -- while multicore systems are multiprocessor systems, not all multiprocessor systems are multicore.



Blade Servers

Finally, **blade servers** - multiple processor boards, I/O boards, and networking boards are placed in the same chassis.

Difference between these and traditional multiprocessor systems

Each blade-processor board boots independently and runs its own operating system.

Some blade-server boards are multiprocessor as well, which blurs the lines between types of computers.

In essence, these servers consist of multiple independent multiprocessor systems

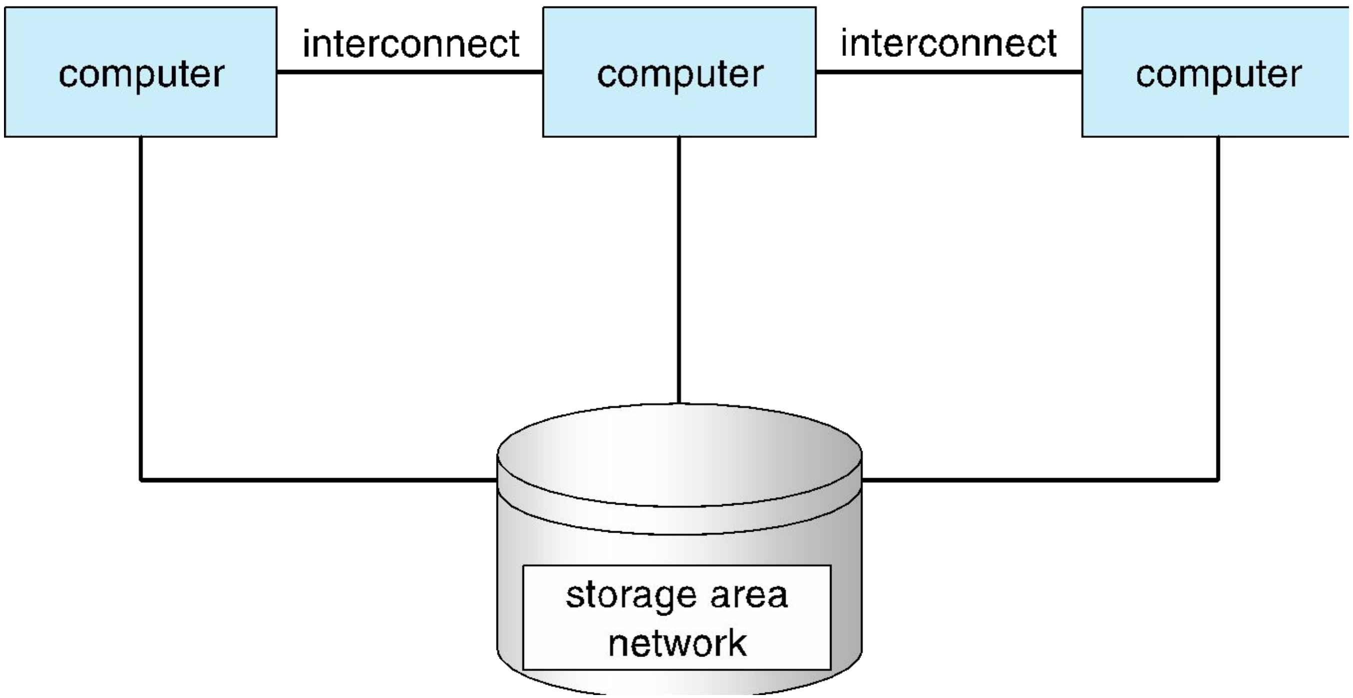
Clustered Systems

- Like multiprocessor systems, but multiple systems working together – composed of two or more individual systems/nodes joined together – loosely coupled. Each node being a single processor system or a multicore system.

Clustered computers share storage and are closely linked via a local-area network LAN or a faster interconnect, such as InfiniBand.

- Usually sharing storage via a **storage-area network (SAN)**
- Provides a **high-availability of** services by adding a level of redundancy which even survives failures
  - **Asymmetric clustering** has one machine in hot-standby mode
  - **Symmetric clustering** has multiple nodes running applications, monitoring each other using all possible hardware.
- Some clusters are for **high-performance computing (HPC) providing** greater computational power
  - Applications must be written to use **parallelization**
- **Parallel Clusters allows** multiple hosts to access the same data on shared storage + require the use of special versions of software and special releases of applications.
  - **Access control and locking system to avoid conflicting accesses.**
- **Clusters over WAN**
- Some have **distributed lock manager** (**DLM**) to avoid conflicting operations

Clustered Systems



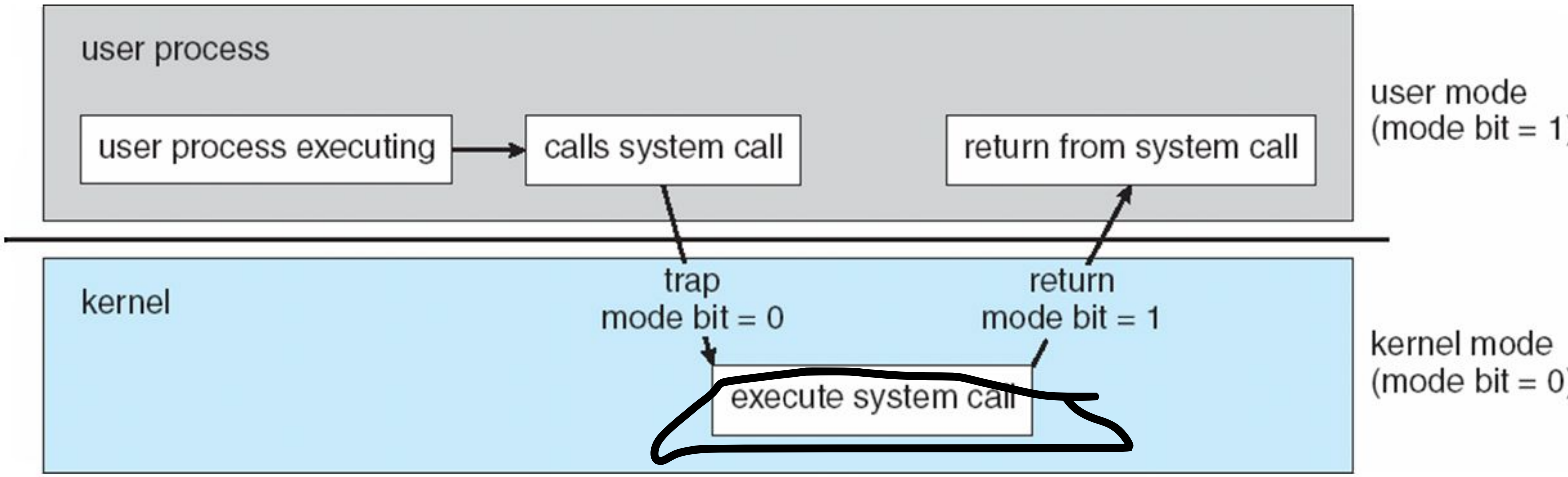
Computing Environments – Real-Time Systems

- Real-time embedded systems most prevalent form of computers
  - Vary considerable, special purpose, limited purpose OS, **real-time OS**
  - Use expanding
- Many other special computing environments as well
  - Some have OSes, some perform tasks without an OS
- Real-time OS has well-defined fixed time constraints
  - Processing ***must*** be done within constraint
  - Correct operation only if constraints met

Modes of Operation

- A mechanism that allows the OS to protect itself and other system components
- Two modes:
  - **User mode**
  - **Kernel mode**
- Mode bit (0 or 1) 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
  - Systems call by a user asking the OS to perform some function changes from user mode to kernel mode.
  - Return from a system call resets the mode to user mode.

Transition from User to Kernel Mode



Timer

To prevent process to be in infinite loop (process hogging resources), a **timer** is used, which is a hardware device.

- Timer is a counter that is decremented by the physical clock.
- Timer is set to interrupt the computer after some time period
- Operating system sets the counter (privileged instruction)
- When counter reaches the value zero, and interrupt is generated.
- The OS sets up the value of the counter before scheduling a 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, etc.
  - Initialization data
- Process termination requires reclaim of any reusable resources
- A thread is a basic unit of CPU utilization within a process.
  - Single-threaded process. Instructions are executed sequentially, one at a time, until completion
  - Process has one **program counter** specifying location of next instruction to execute
- Multi-threaded process has one program counter per thread
- Typically, a system has many processes, some user, some operating system running concurrently on one or more CPUs
  - Concurrency by multiplexing the CPUs among the 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 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 Management

- OS provides uniform, logical view of information storage
- Abstracts physical properties to logical storage unit - **file**
- Files are stored in a number of different storage medium.
  - Disk
  - Flash Memory
  - Tape
- Each medium is controlled by device drivers (i.e., disk drive, tape drive)
  - 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 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

Secondary-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
  - Free-space management
  - Storage allocation
  - Disk scheduling
- Some storage need not be fast
  - Tertiary storage includes optical storage, magnetic tape
  - Still must be managed – by OS or applications

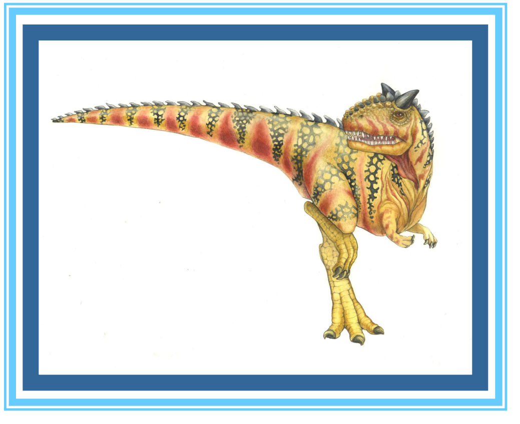
Protection and Security

- **Protection** – A mechanism for controlling access of processes (or users) to resources defined by the OS
- **Security** – A 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 is associated with all files and 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

- Allows operating systems to run applications within other OSes
  - Vast and growing industry
- **Emulation** used when the source CPU type is different from the 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

End of Chapter 1



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 are smaller (size-wise) than storage being cached
  - Cache management important design problem
  - Cache size and replacement policy

Performance of Various Levels of Storage

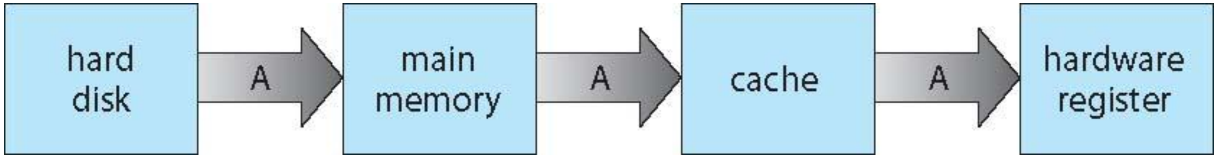
V. imp

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

Movement between levels of storage hierarchy can be explicit or implicit

Migration of data "A" from Disk to Register

- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy



- Multiprocessor environment must provide **cache coherency** in hardware such that all CPUs have the most recent value in their cache
- Distributed environment situation even more complex
  - Several copies of a datum can exist
  - Various solutions covered in Chapter 17

I/O Subsystem

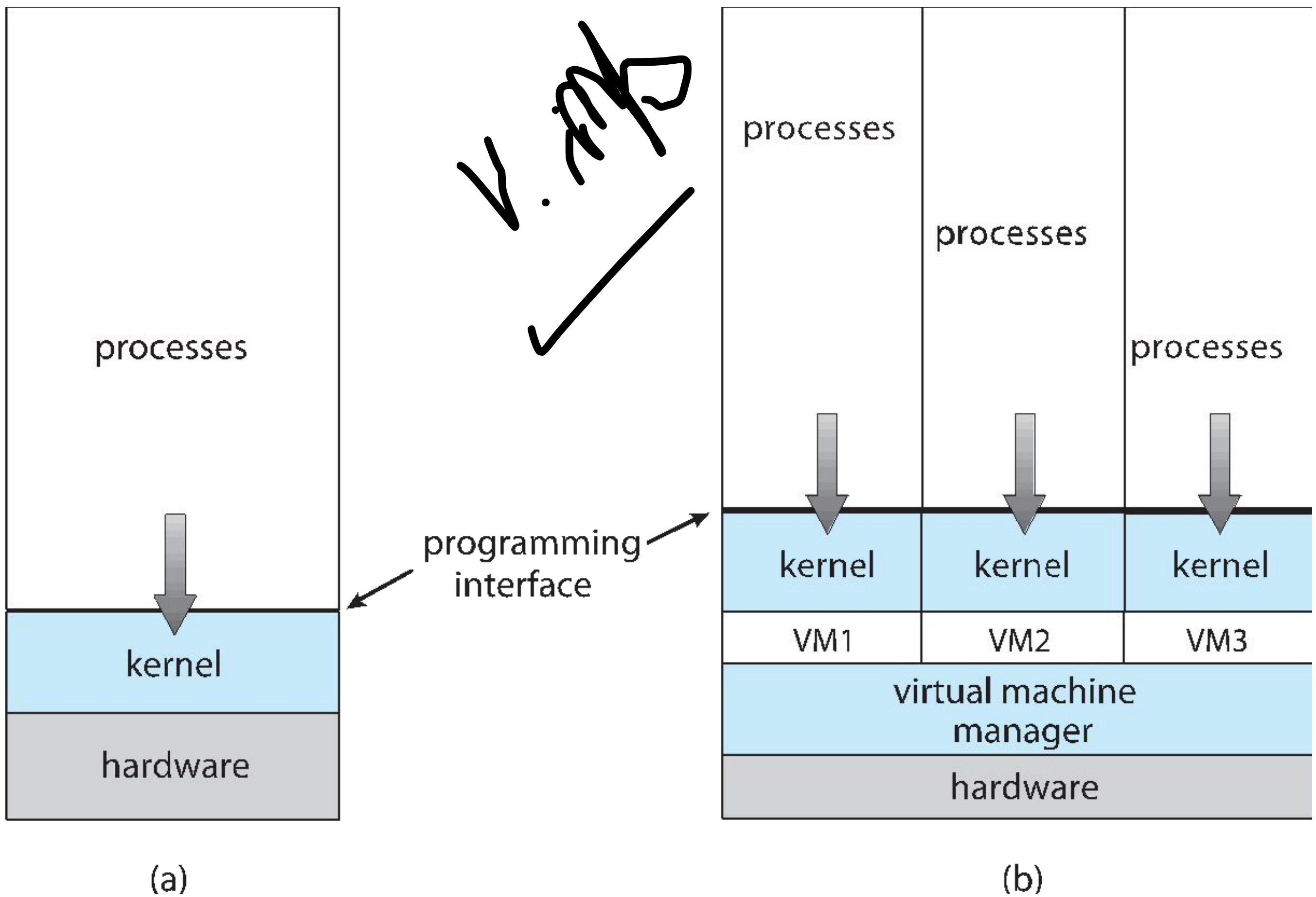
- One purpose of an operating system is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
  - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance),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

Virtualization on Laptops and Destops

- A VMM allow the user to install multiple operating systems to run application written for operating systems other than the native host.
  - Apple laptop running Mac OS X host Windows as a guest
  - Developing apps for multiple OSes without having multiple systems
  - Testing applications without having multiple systems
  - Executing and managing compute environments within data centers

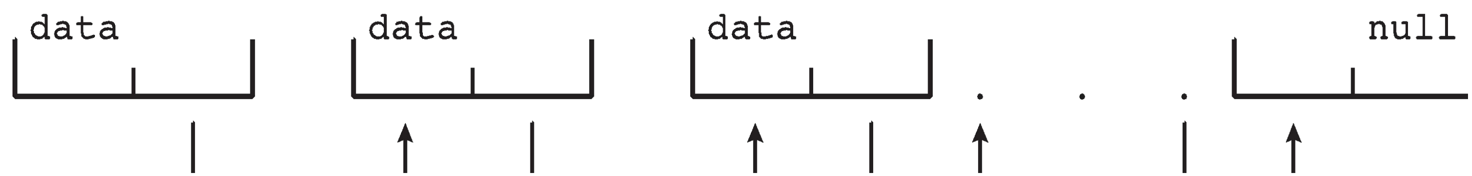
Virtualization Architecture Structure

V. imp

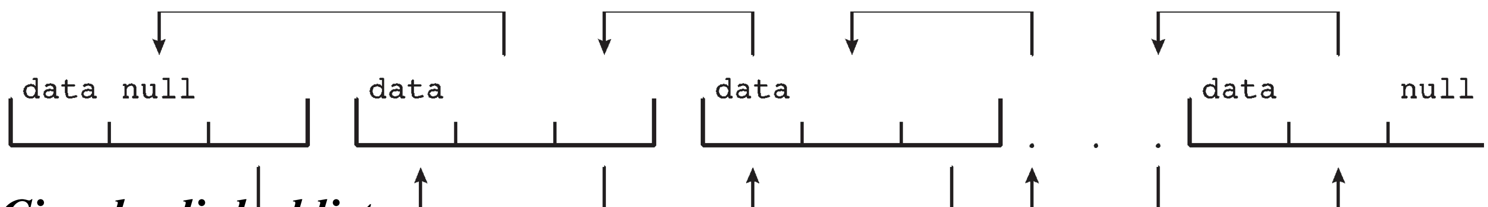


Kernel Data Structures

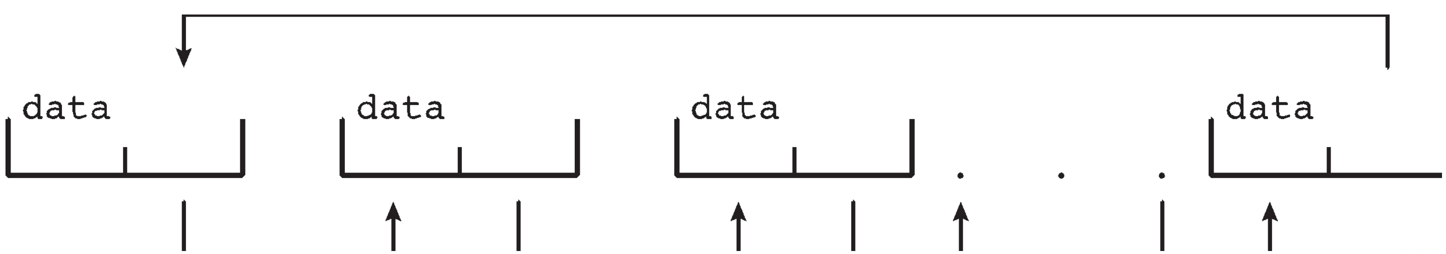
- Many -- similar to standard programming data structures
- ***Singly linked list***



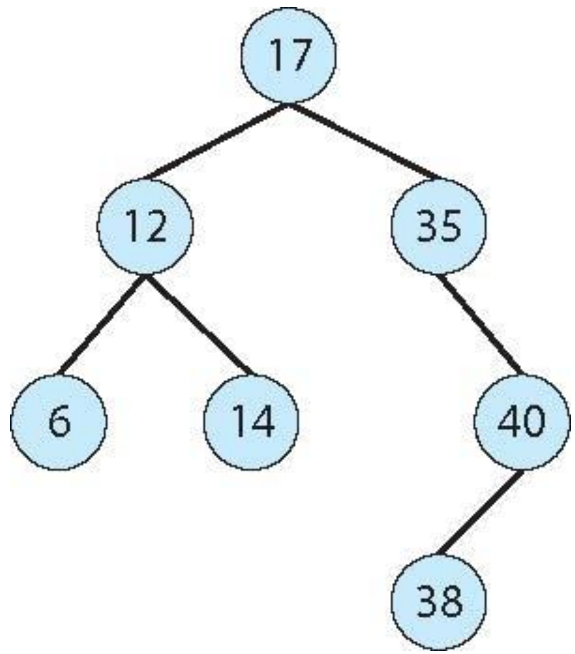
- ***Doubly linked list***



- ***Circular linked list***

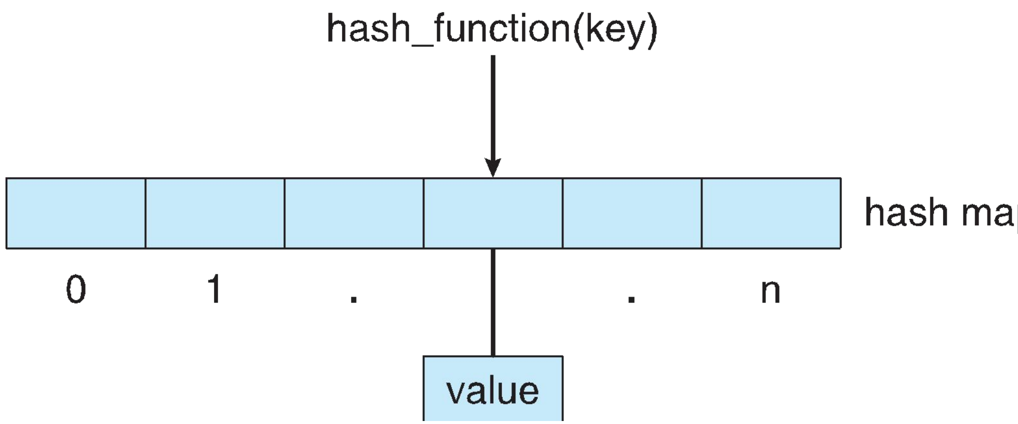


Binary search tree



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>`

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 - Mobile

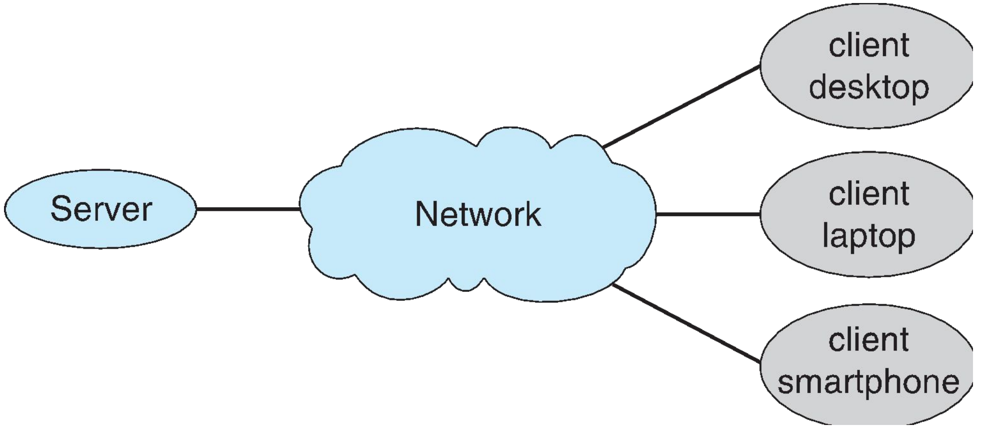
- Handheld smartphones, tablets, etc
- What is the functional difference between them and a “traditional” laptop?
- Extra features – more OS features (GPS -- Waze)
- Allows new types of apps like ***augmented reality***
- Use IEEE 802.11 wireless, or cellular data networks for connectivity
- Leaders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

Computing Environments – Distributed

- 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 to allow sharing of data between systems across a network.
  - Communication scheme allows systems to exchange messages
  - Illusion of a single system

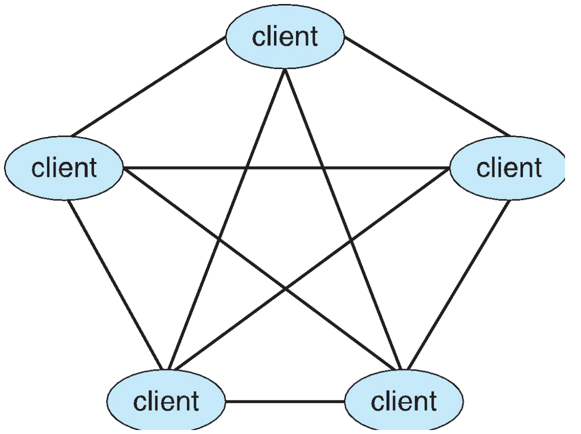
Computing Environments – Client-Server

- 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 - Peer-to-Peer

- Another model of distributed system. P2P does not distinguish clients and servers
  - Instead all nodes are considered peers
  - Each node may 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

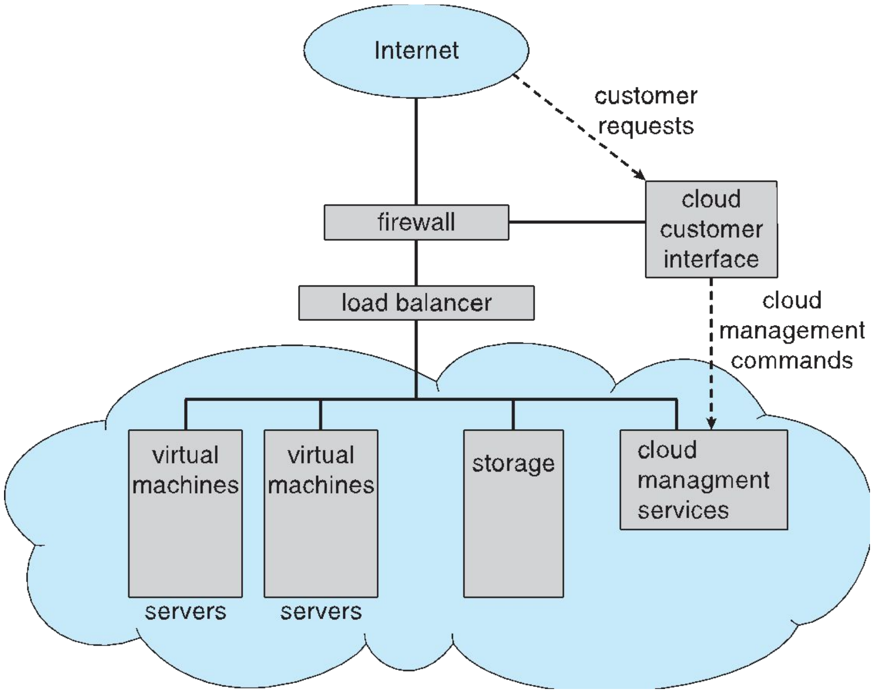


Computing Environments – Cloud Computing

- 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
- 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)

Computing Environments – Cloud Computing

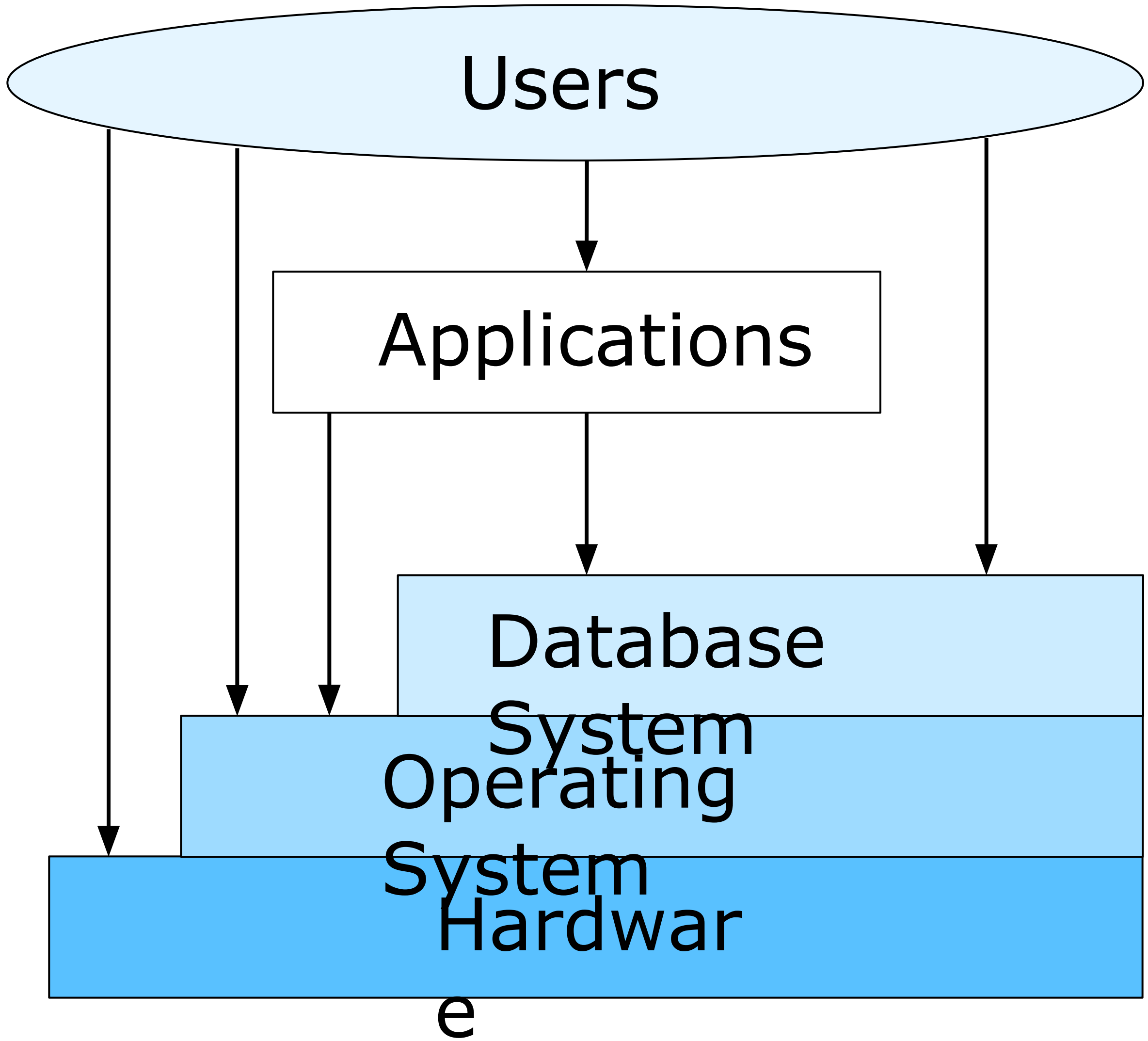
- Cloud computing environments composed of traditional OSes, plus VMMs, plus cloud management tools
  - Internet connectivity requires security like firewalls
  - Load balancers spread traffic across multiple applications



Computing Environments – Real-Time Systems

- Real-time embedded systems most prevalent form of computers
  - Vary considerable, special purpose, limited purpose OS, **real-time OS**
  - Use expanding
- Many other special computing environments as well
  - Some have OSes, some perform tasks without an OS
- Real-time OS has well-defined fixed time constraints
  - Processing ***must*** be done within constraint
  - Correct operation only if constraints met

Evolution of Computer Systems



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 on many platforms - http://www.virtualbox.com)
  - Use to run guest operating systems for exploration

End of Chapter 1