CSC 139
Operating System Principles

Chapter 1
OS Introduction

Herbert G. Mayer, CSU CSC

8/1/2022

Syllabus

- Components of Computer System
- OS Operations
- Process Management
- Memory Management
- Storage Management
- Protection and Security
- OS Kernel
- Environments
- Open Source OS

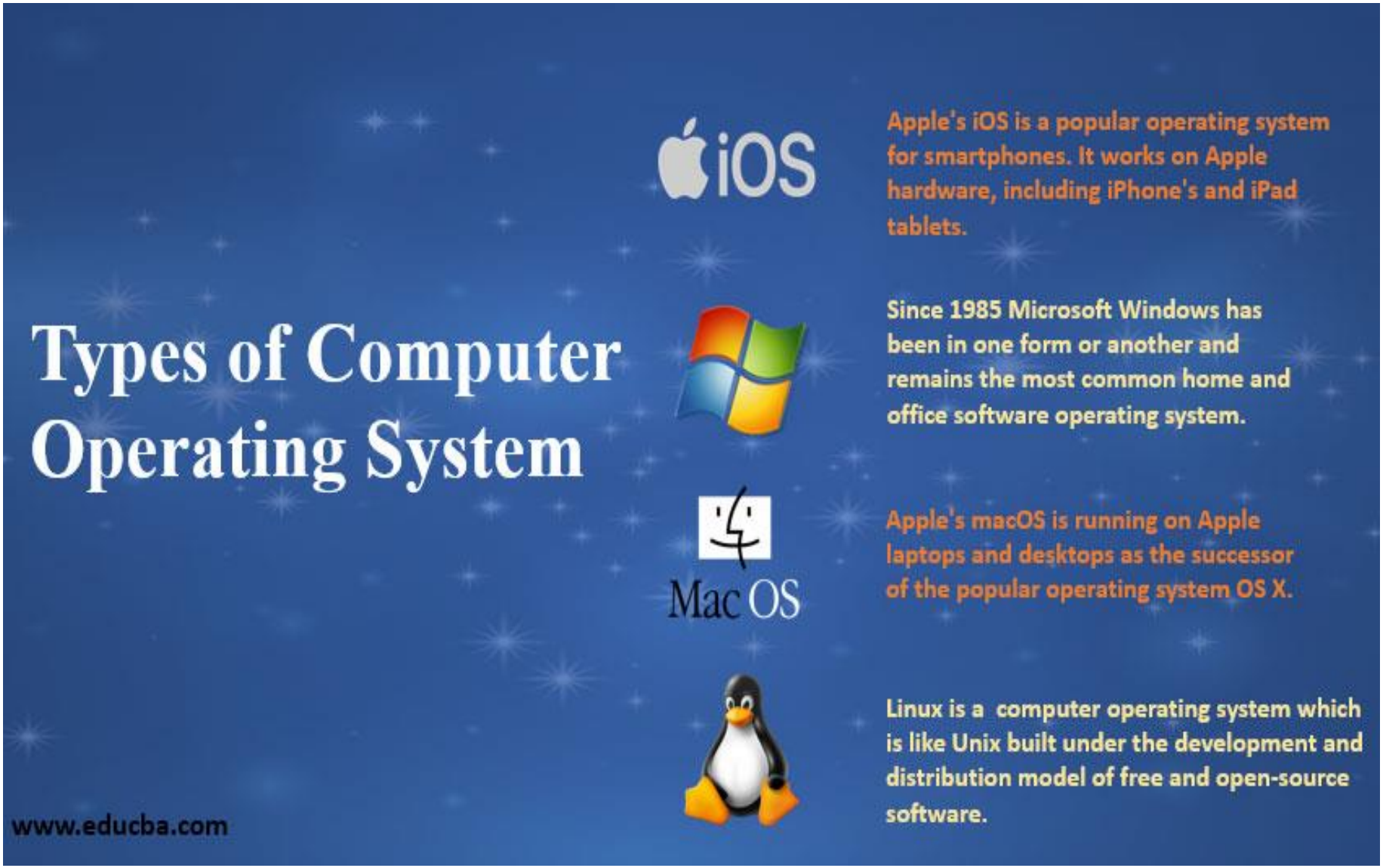
Objectives

- **Clarify, what is an Operating System (OS), and what is its primary use**
- **Describe the basic organization of modern computers and their OSes**
- **Provide a *grand tour* of major components of an operating system**
- **Give an overview of the numerous types of compute environments**
- **And explore some high-level Open Source OSes**

Characterize Operating System

- A computer's Operating Systems is a highly complex piece of system SW
- Acting as intermediary between computer user and the HW plus SW services
- OS goals:
  - Execute user programs; consider priorities of each
  - Efficiently manage compute resources; limited in number
  - And manage them speedily regarding user wait time and time to completion of their use
  - Render computer system convenient and easy to use
  - Manage computer HW effectively and economically
  - Store relevant information long-term

Sample Operating Systems



Computer System Structure

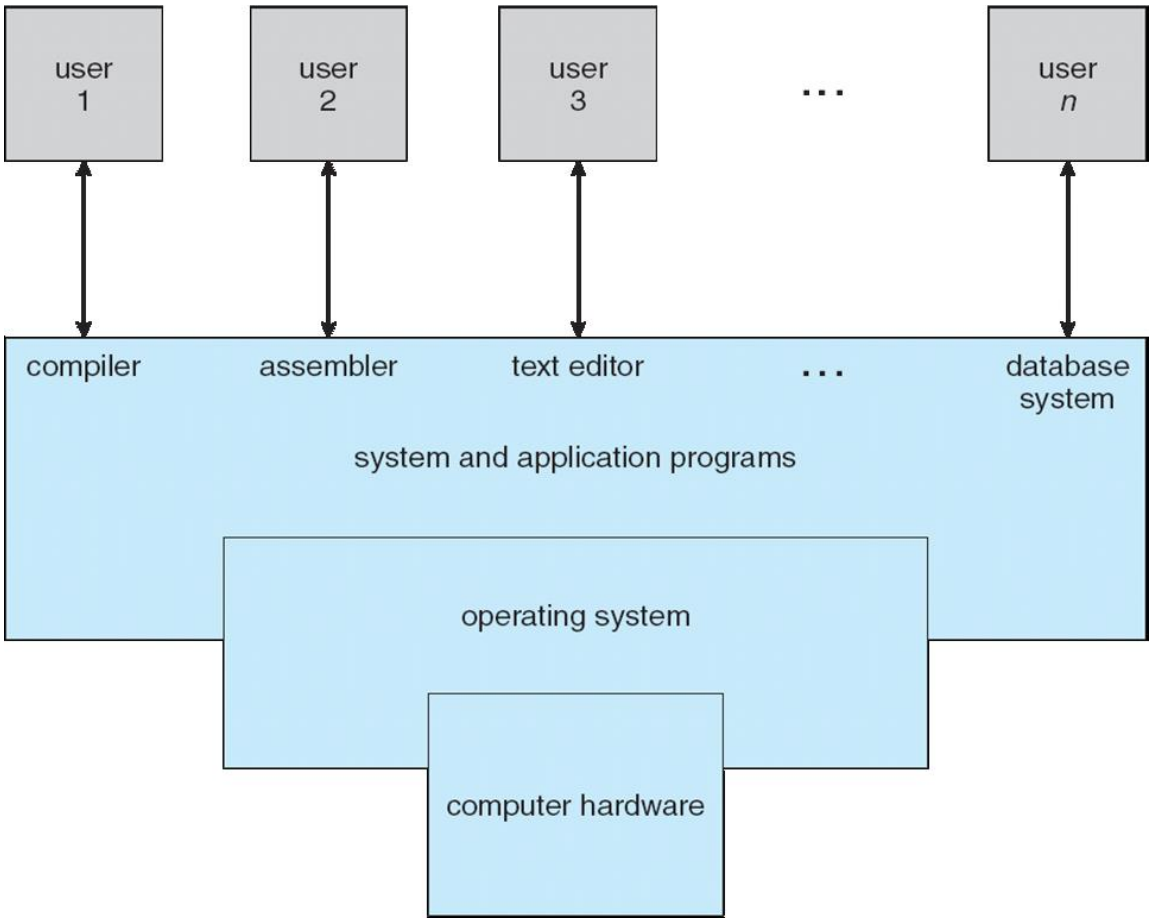
Early Computers

Computer System Structure

Computer system comprises four components:

- **Hardware resources (HW)**
  - **CPU, memory, IO devices e.g. disks, buses**
- **Operating System**
  - **Controls and coordinates use of HW among various applications and competing users**
- **Application programs (Apps)**
  - **Use system resources to solve computing problems**
  - **Word processors, compilers, web browsers, database systems, video games**
- **Users (you)**
  - **Human users**
  - **Other computers, and networks**

Computer System Structure



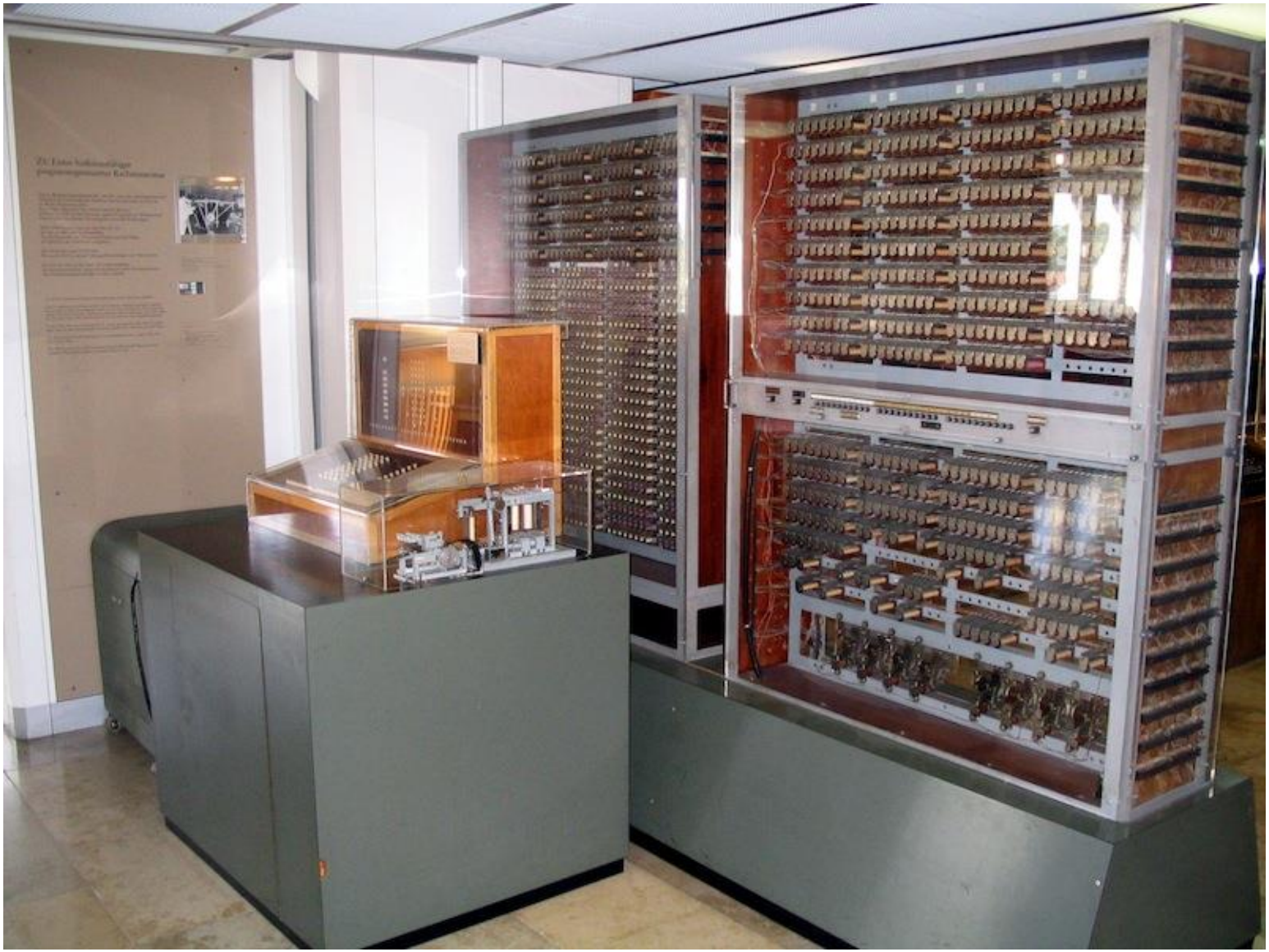
Early Computer System

Blaise Pascal's calculators from 1642

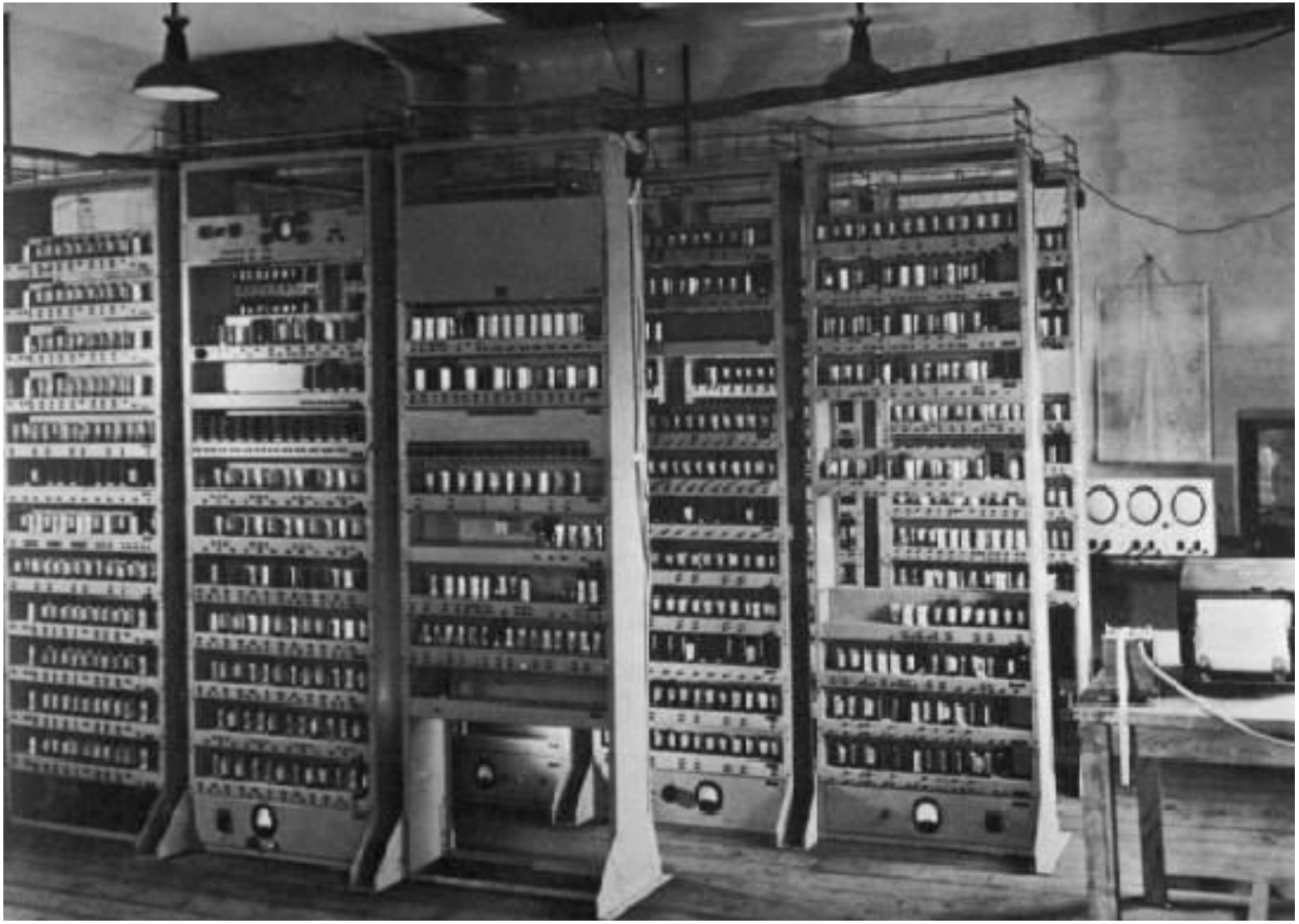
Plus one unit built by Lépine in 1725

Exhibit in the *Musée des Arts et Métiers* in Paris 2020

Early Computer Zuse Z3, 1941

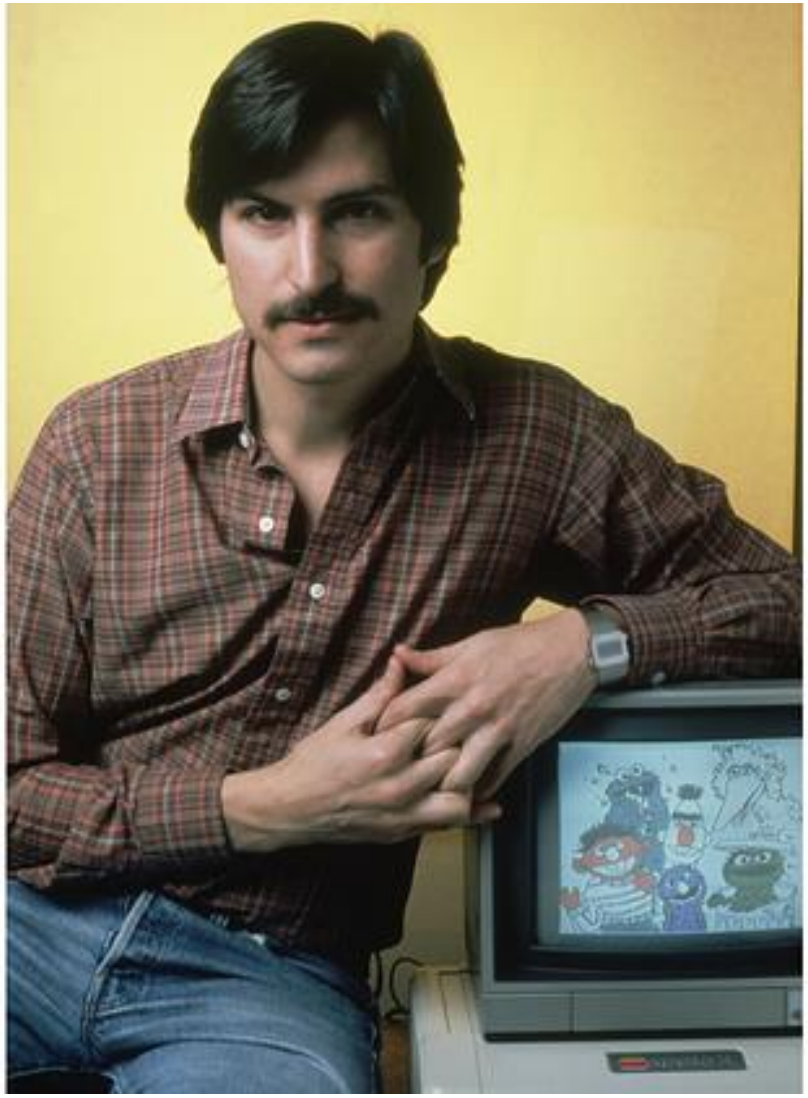


Early Computer EDSAC, 1949



Early Computer Apple, ca. 1976

Early Computer Apple



OS Responsibilities

What Operating Systems Do

- Varies with point of view of OS design
- Users desire for OS: service convenience, ease of use and good performance
  - Don't care primarily about resource utilization
- But shared computers including mainframes or minicomputers must satisfy user needs
- Users of dedicated workstations have committed resources but frequently use additional, shared ones via connected servers
- Handheld computers are resource poor, optimized for portability, and battery life
- Some computers have no user interface, e.g. embedded computers in devices and automobiles

What Operating Systems Do

5 Key Functions of Operating Systems

Operating System Definition

- **OS is a computer resource manager**
  - **Manages all computer HW and SW resources**
  - **Resolves conflicting requests, deciding for efficient and fair resource use**
- **OS is a control program**
  - **Controls execution of programs to prevent errors or improper use of computer**

Operating System Definition

No universally accepted OS definition; but in general:

- **"The System Software a vendor ships when you order a computer" is a plausible, initial definition**
  - **Yet that vendor shipment varies widely**
- **"The one piece of OS SW resident at all times on the computer" is the kernel**
- **Everything else is either**
  - **A *system program* that also ships with OS**
    - **May be loaded transparently upon use request**
  - **Or an *application program***

Software Categories

Not only Operating Systems establish a usable computer platform; also:

- ***Compilers*** **and translators**
  - **Enable programmers to create other software products**
- **Software applications**
  - **Serve as productivity tools to help users solve problems**
- **System software, cooperating with OS**
  - **Provides SW solution: e.g. mail, text editing, store files, . . .**
  - **Coordinates hardware operations**

Language of Computers

- **Computer processes instructions in encoded in binary form: *machine language***
  - **Numeric codes used to represent basic operations, defined by op-codes, e.g.:**
    - **Adding, subtracting numbers**
    - **Comparing values**
    - **Loading and storing bits, bytes, words, strings, . . .**
    - **Moving bit-strings; can be interpreted as numbers, characters**
    - **Repeating instructions, by branching (AKA jumping)**
- **Programmers often use *high-level languages***
  - **C, Fortran, Basic, Cobol, PL/I, Algol, Algol68**
  - **Today largely: Java, C++, SQL, Python, C#**

Consumer Applications

- **Disclaimer −an *end-user license agreement (EULA)*− generally protects SW companies from errors in programs**
- **Protects vendor from consequences of faulty SW**
- **Try to find a SW company that guarantees 0 bugs! ☺**
- ***Licensing:* Users and customers just purchase a software license, not "the" program**
  - **You may only use the program! You do not own that SW**
  - **General model used today!**
- **Distribution by direct sales or download from Web**
  - **Includes *public-domain software* and *shareware***

Web Applications

- **Growing trend toward using applications that run on remote Internet servers instead of local PCs**
- **Making results visible on, and accessible to your site:**
  - **Google Docs**
  - **Webmail programs: Gmail, Hotmail, Yahoo! Mail**
  - **Wikis: Wikipedia**
  - **Online communities: E.g. Facebook**

HW – SW Connection

- ***System software:*** **Class of SW that includes OS and utility programs**
- **Handles low-level details and other tasks behind the scenes, AKA transparently**
- **User does not need to be concerned about all details**
- **Convenient for user, business protection for owner of SW**
- **Interface consistency is convenient for computer user, reduced amount of "re-learning"**

Function of OS

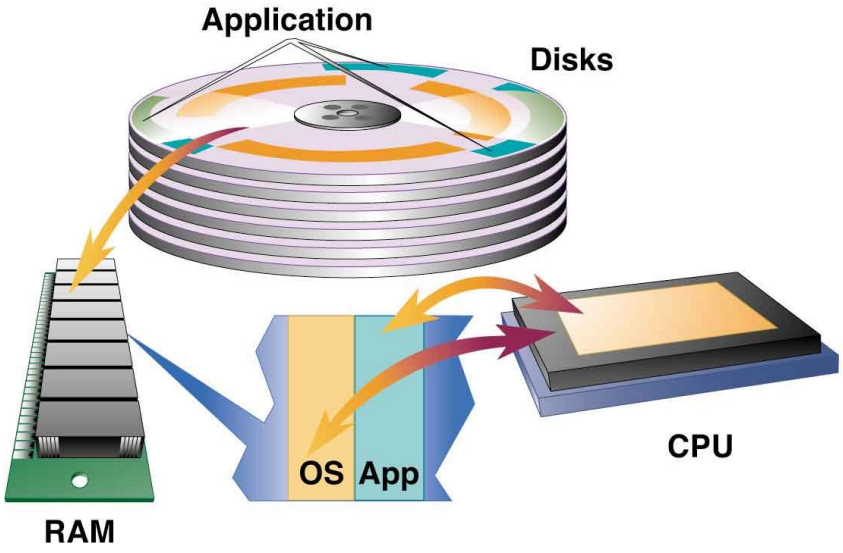
- **Computer depends on the *OS* to:**
  - **Keep HW running**
  - **Use resources efficiently**
  - **Maintain a *file system***
  - **Support *multitasking* even on a UP target**
  - **Manage *virtual memory*** (in recent times again)
- **The OS runs continuously when the computer is turned on**
- **Possibly just executing the “idle Loop”, i.e. that part of the OS code which recognizes: “ahh, no user request needs to be handled now!”**

Utility Programs

- **Serve as tools for doing system (SW, HW, FW) maintenance and repairs not handled by the OS**
- **Utilities make it easier for users to:**
  - **Create, copy, delete, store files on storage devices**
  - **Repair (some, sometimes) damaged data files**
  - **Translate files so different programs can process them**
  - **Guard against viruses and other harmful programs**
  - **Compress files so they take up less space**
  - **Communicate between different computers, locations**

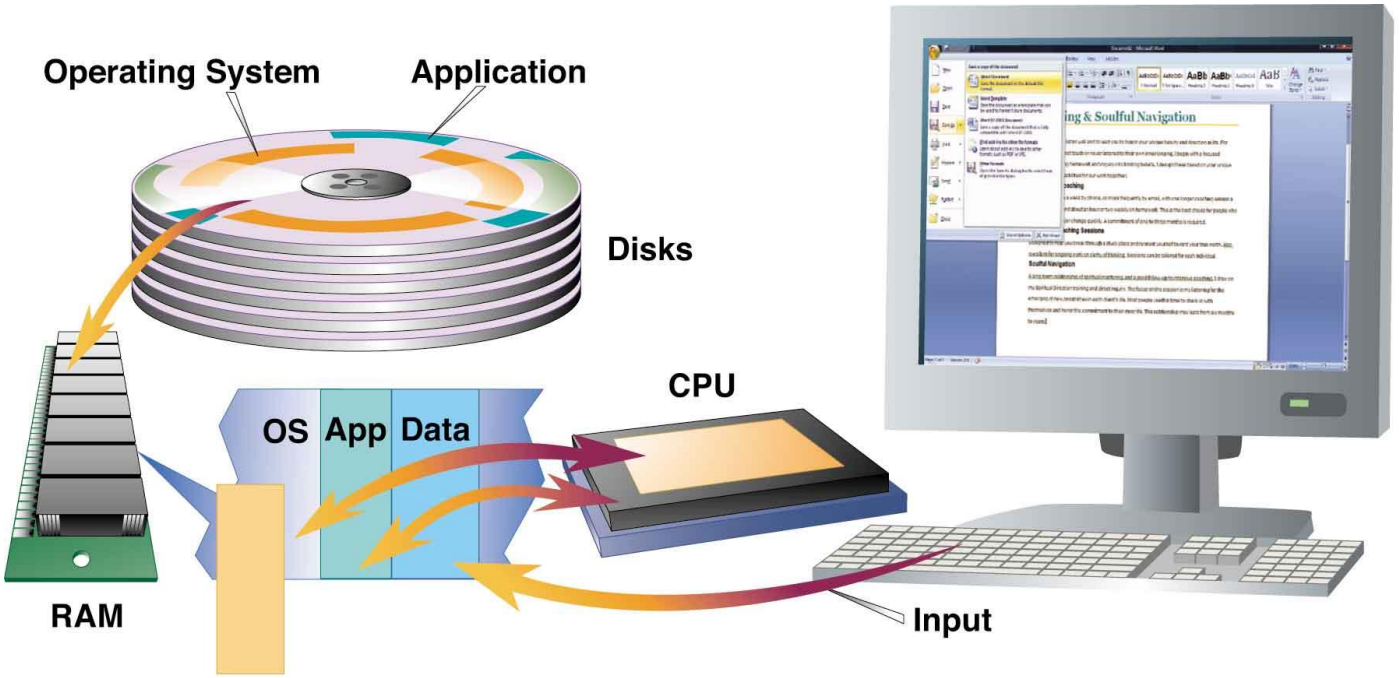
The OS

When you turn on the computer, the CPU automatically begins executing instructions stored in boot ROM, loading an initial OS. That OS then loads more SW modules from disk into system memory



The OS

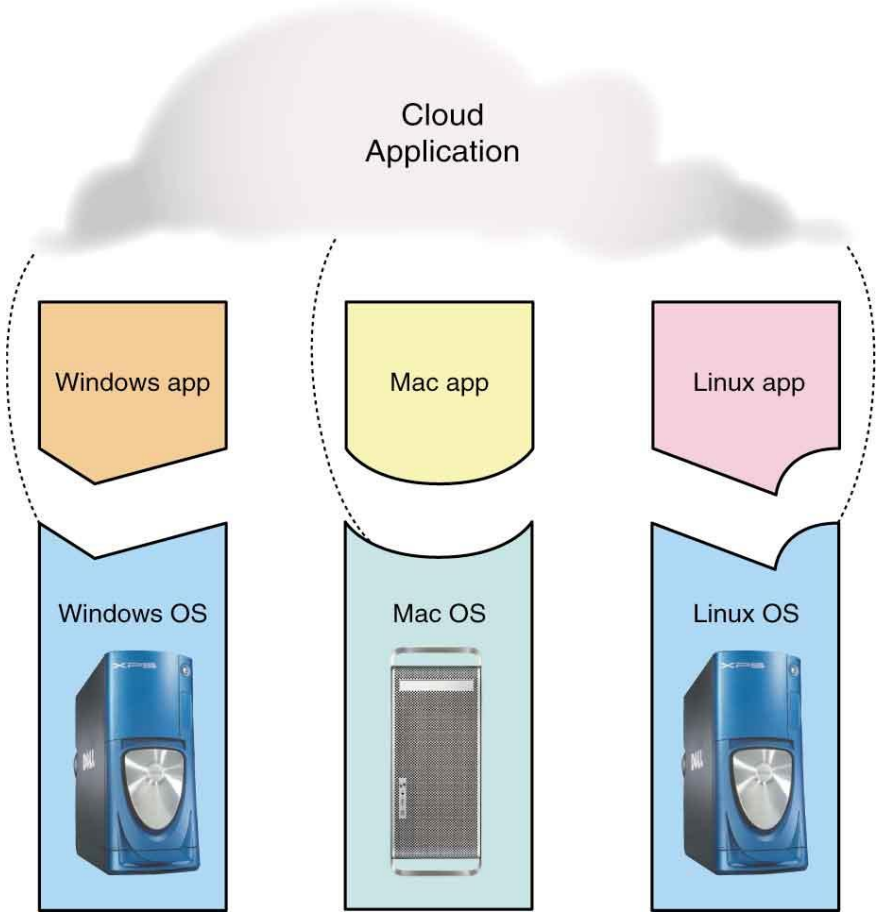
The OS loads application programs into memory; OS remains in memory (in part), to provide services to applications: displays on-screen menus, communicates with printer, performs common services, and more



Unix and Linux

- ***Unix:*** **Originally command-line based, textual input and output based OS**
  - **Internet is populated with computers running Unix**
  - **OS of choice for workstations and mainframes in research and academic settings**
  - **Favored by many who require an industrial-strength, multiuser OS**
  - **Few are open source, e.g. BSD**
- **Linux, a later Unix "cousin", is distributed and supported free, free of license fees**
  - **Source accessible to public**

Compatibility Issues



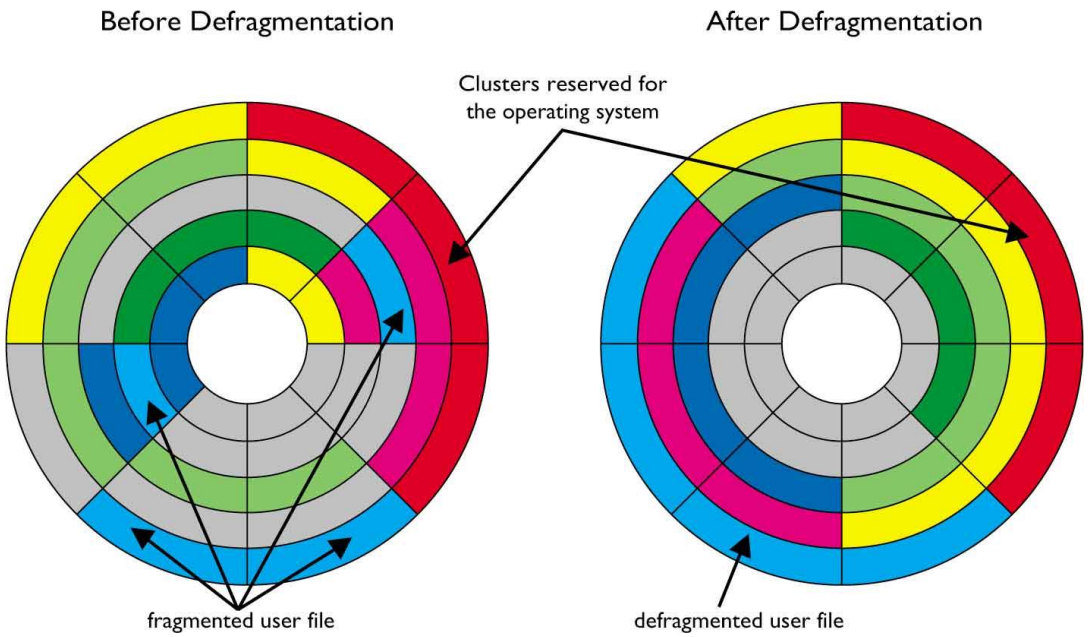
- Operating systems are designed to run on particular HW platforms
- Applications are designed to run under a particular operating systems
- Most cloud applications are designed to run on multiple platforms

Disk Formatting

- **Hard disks (old rotating disks) are formatted by manufacturer before installing operating system:**
  - **Electronic marks are written onto disk**
  - **Disk is divided into series of concentric tracks**
  - ***Tracks* are divided into sectors**
  - ***Sectors* are bundled *clusters* AKA *blocks***
- **File system provides way to link multiple clusters to store single, large files**
- **And to store numerous, distinct files**
- **Usually organized as a hierarchy of directories**
  - **Exceptions: early CDC computers with “flat” file structure**

Defragmentation

- **Contents of file may become scattered over diverse clusters**
- **File access is faster if file is assigned to contiguous clusters**



- **A *defragmentation utility* eliminates fragmentation**
- **AKA defragger**

Intellectual Property

Intellectual Property Laws

- **Software piracy: Illegal duplication of copyrighted software; is rampant across the world**
- **Do respect the ©. This little sybols is not solely a little symbol ☺**
- **Few SW companies use physical copy protection methods; that makes copying easy**
- **Some SW users, even developers, are unaware of copyright laws**
- **Others simply *"look the other way"***
- **And yes, hard to believe: *Some users do steal outright!***

Piracy Problem

- SW industry loses billions of dollars annually to software pirates
- Some countries and cultures are particularly good at and experienced in pirating ☺
- Business Software Alliance (BSA) estimates that possibly one-third of software in use has been illegally copied
- Piracy is particularly hard on small SW companies
- Industry organizations work with law enforcement agencies to crack down on piracy

Computer Startup

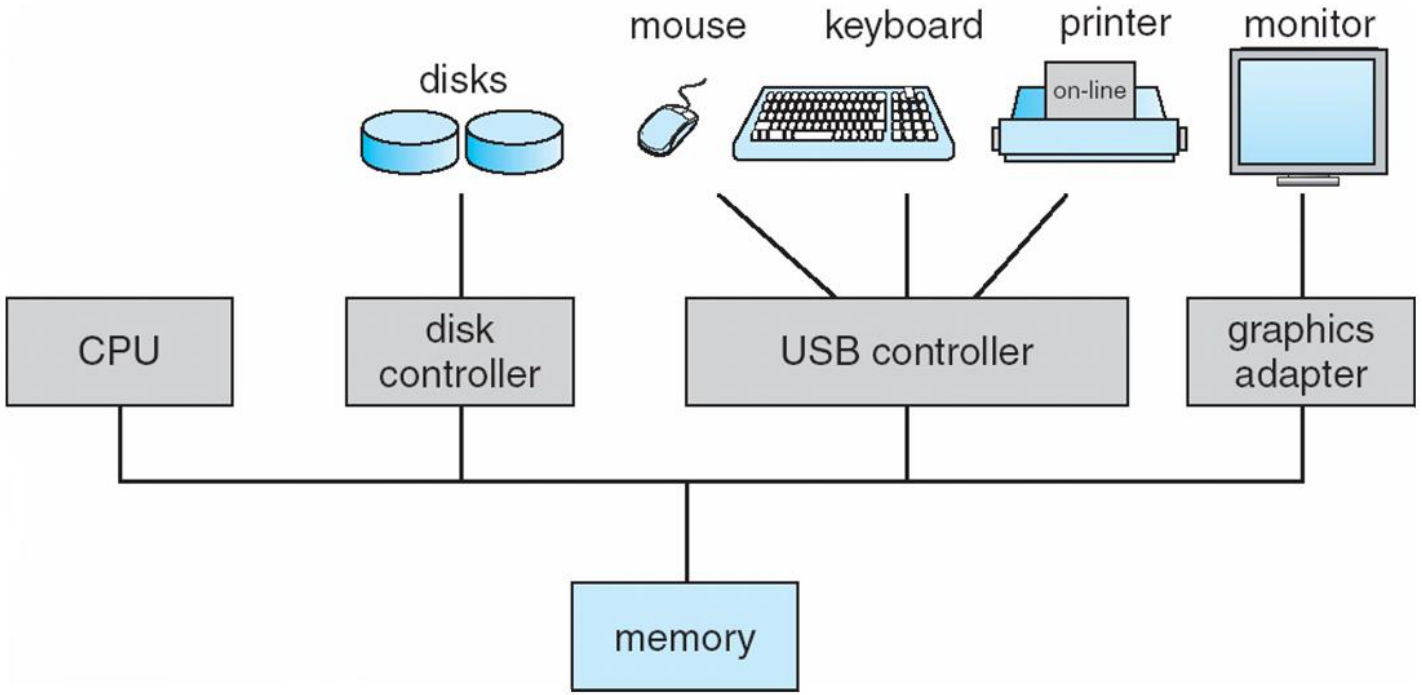
Bootstrap program is loaded at power-up, AKA at reboot time

- **Loads operating system kernel and starts execution**
- **Typically stored in ROM or EPROM, AKA firmware**
- **Initializes all aspects of system**
- **See bootstraps at side of boots? ☺**

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 access



Interrupts

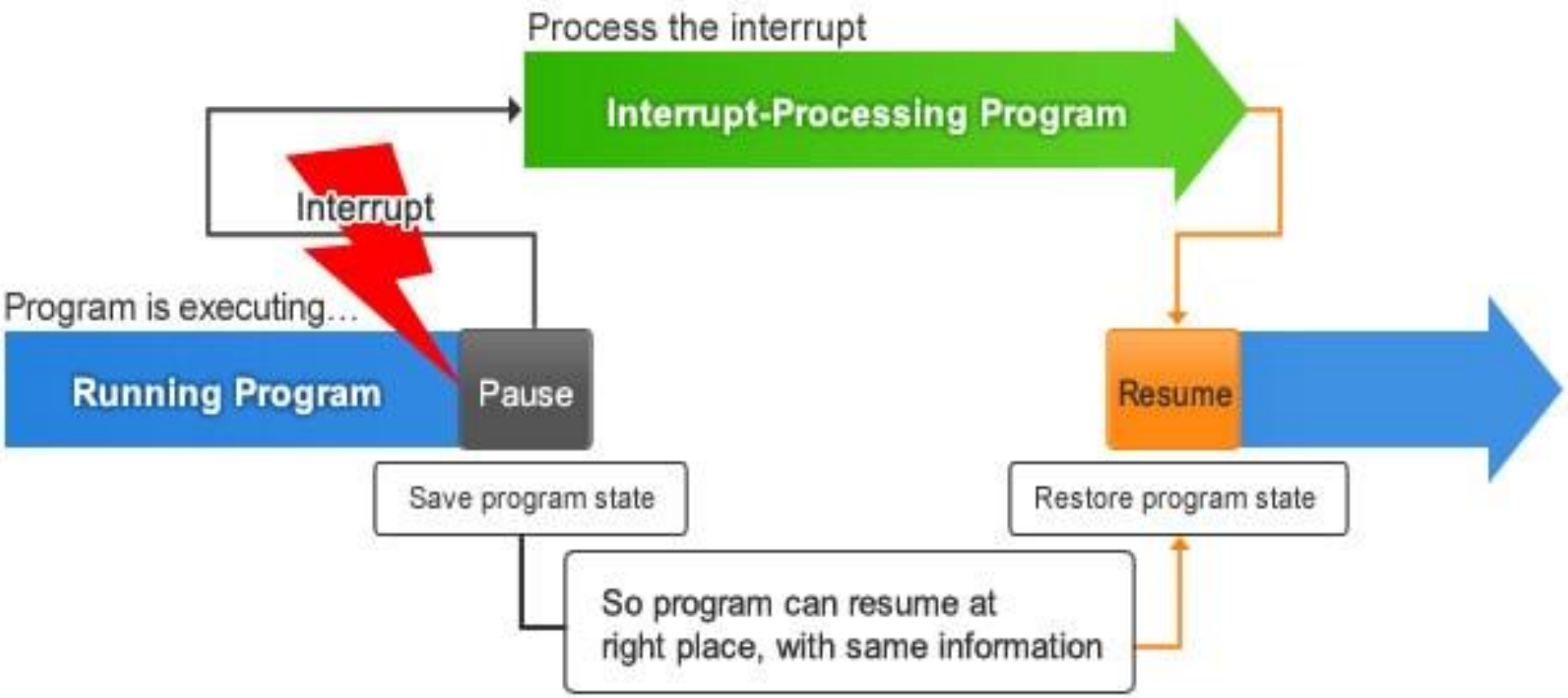
Computer System Operation

- IO devices and CPU can at times execute in parallel; AKA simultaneously
- Each device type requires a particular device controller
- Device controller has local memory for buffer space
- CPU moves data from/to main memory, and to/from local buffers
- IO is transfer of information between main memory and outside world; outside world means peripheral devices; e.g. keyboard, printer, mouse, disk, CRT, …
- Device controller informs CPU that it has finished its operation by causing an interrupt

Common Functions of Interrupts

- Typically an OS is interrupt driven
- An interrupt "interrupts" ☺ regular flow of execution; transfers control to the computer's interrupt service routine, through so called interrupt vector, which contains the address of each distinct service routine
- Interrupt architecture must save the address of the interrupted instruction and the complete state of the interrupted SW
- A trap or exception is a software generated interrupt, caused by an error in a user program
- Lest you forget: an OS is interrupt driven

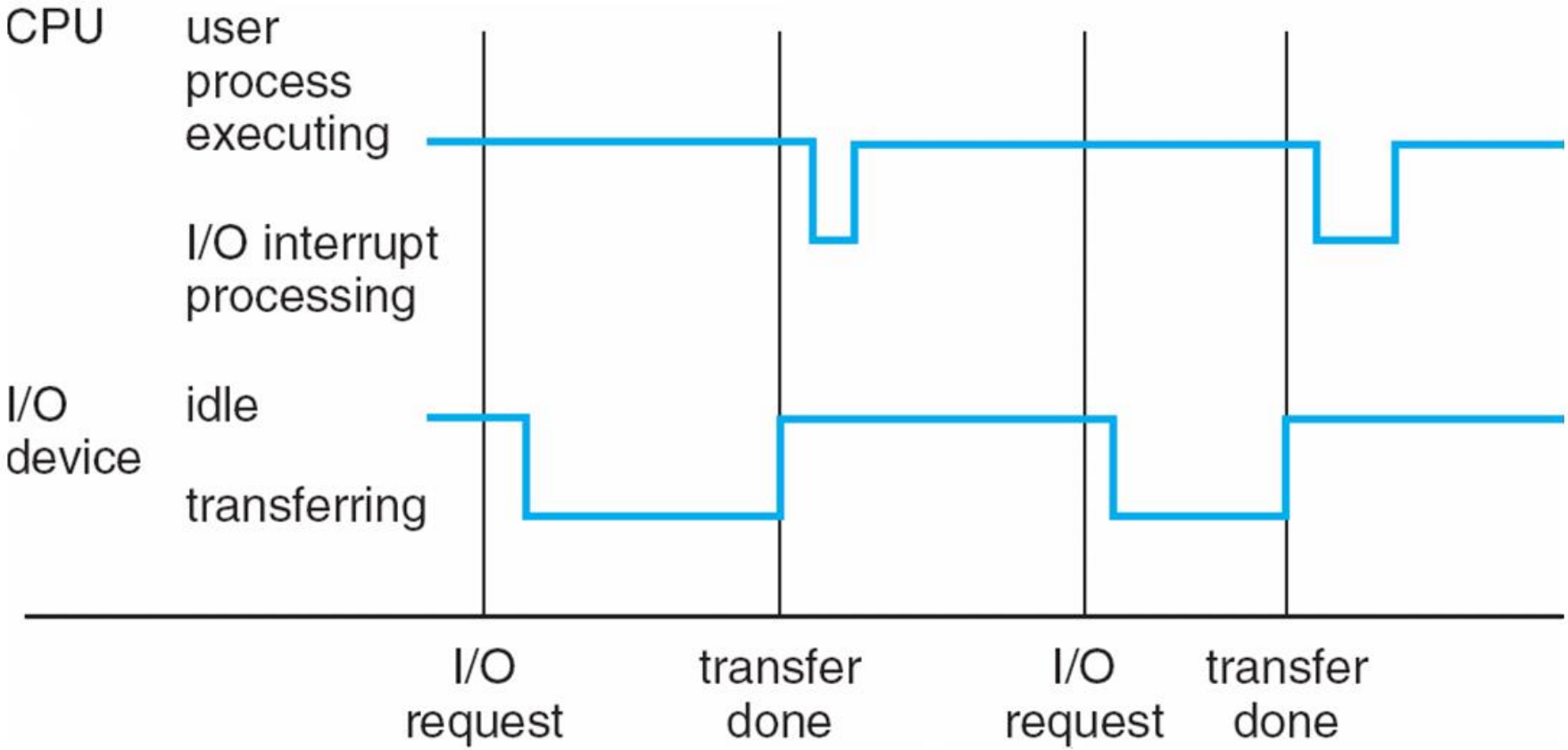
Interrupt


Process the interrupt
Interrupt-Processing Program
Interrupt
Program is executing…
Running Program
Pause
Resume
Save program state
Restore program state
So program can resume at
right place, with same information

Interrupt Handling

- At interrupt, OS preserves the CPU state by storing registers, program counter, stack etc.
- OS determines, which type of interrupt can occur:
  - polling
  - vectored interrupt system
- Separate modules of code determine what action should be taken to handle each type of interrupt
- AKA interrupt handler!
- Including catastrophic events, such as unexpected, and highly undesirable loss of power

Interrupt Timeline



Storage

IO Structure

- Case 1: Once IO starts, control only returns to user program upon IO completion (e.g. file input)
  - Wait instruction idles the CPU until further interrupt
  - Wait loop
  - Wastes resources, in a planned fashion ☺
  - At most one such IO request is outstanding for any process at any one time
- Case 2: Once IO starts, control returns to user program without waiting for IO completion (e.g. screen output)
  - Device-status table contains entry for each IO device indicating its type, address, and state
  - OS indexes into IO device-status table to determine device status and to modify table entry to include interrupt
  - Compute continuation works only for some IO operations

Mass Storage

Storage Definitions

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. Given enough bits, it is amazing how many things a computer can represent: numbers, letters, images, movies, sounds, documents, and programs, to name a few. A **byte** is a unit of 8 bits that are addressable. On most computers it is the smallest convenient unit of storage.

For example, most computers don't have an instruction to move a bit but do have one to move a byte. A less common term is **word**, which is a given computer architecture's native unit of data. A word is made up of multiple bytes. For example, a computer that has 64-bit registers and 64-bit memory addressing typically has 64-bit (8-byte) words. A computer executes many operations in its native word size rather than a byte at a time.

Computer storage, along with most computer throughput, is generally 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

. . . Etc.

Computer manufacturers often round off these numbers and say that a megabyte is 1 million bytes or 10^6 bytes; and a gigabyte is 1 billion bytes or 10^9 bytes. Networking measurements are an exception to this general rule; they are given in bits, since some networks move data a bit at a time.

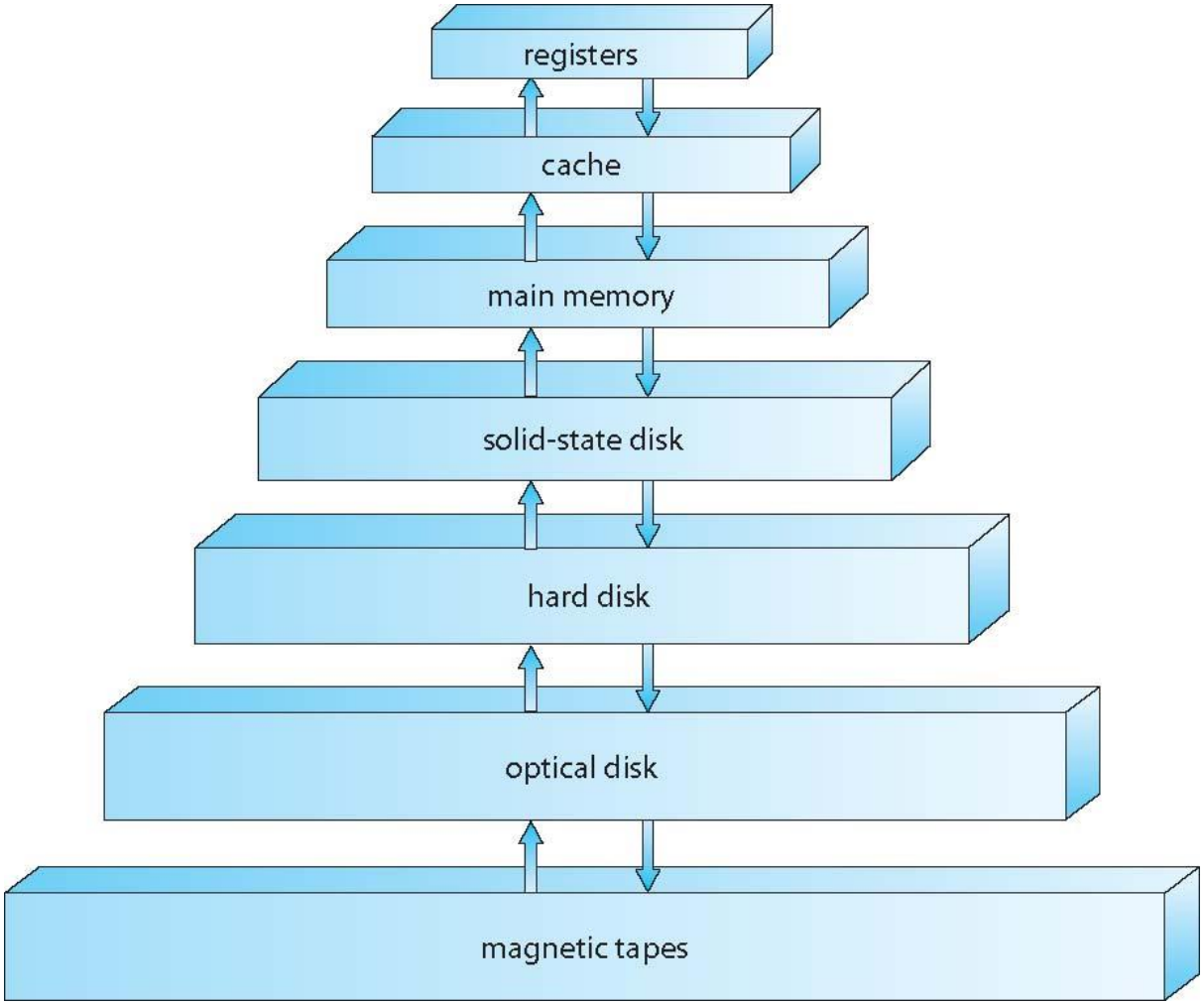
Storage Structure

- **Main memory: large 2**64, addressable medium to store information the CPU accesses via load & store**
  - **Random access memory, acronym: RAM**
  - **Typically volatile, bus fast compared to peripheral devices**
- **Secondary storage – extension of main memory that provides large nonvolatile storage capacity**
- **Hard disk AKA Hard drive (HD) – rigid metal or glass platters covered with magnetic recording material**
  - **Disk surface is logically divided into tracks, further subdivided into sectors**
  - **Disk controller determines logical interaction between device and computer**
- **Solid State Disk (SSD) – faster than HD, nonvolatile**
  - **Various technologies, more expensive than conventional, rotating disk**
  - **Becoming more popular, despite being more expensive!**

Storage Hierarchy

- Storage systems organized in hierarchy, grouped by
  - Speed of access (get to the first bit)
  - Speed of transmission (transmit bytes per second)
  - Cost
  - Volatility
- Caching: copying information into faster storage system; main memory can be viewed as a cache for secondary storage; term typically *not used* this way!
- Device Driver (SW) for each HW device, manages IO performed by any one *individual* controller
  - Provides uniform interface between controller and kernel
  - Enables same IO device to be used on different OSes

Storage Device Hierarchy



Caching

- Important principle, performed at various levels in computer; implemented purely in HW
- Information in use copied from slower (larger) to faster (smaller) storage temporarily; caveat: DUPs ☺
- Faster storage (cache or first-level cache) is checked first to determine if information is already present
  - If it is, AKA hit, information used directly from cache (fast)
  - If not, data are copied from slower device to cache and used there (first time slow, from then on fast)
- Cache is way, way smaller than storage being cached
  - Cache management poses critical design problem
  - Cache size and replacement policy
  - Multiple “levels” of caches, known as L1, L2, L3 cache etc.
  - Bigger L-number means: larger size, slower access

TLB

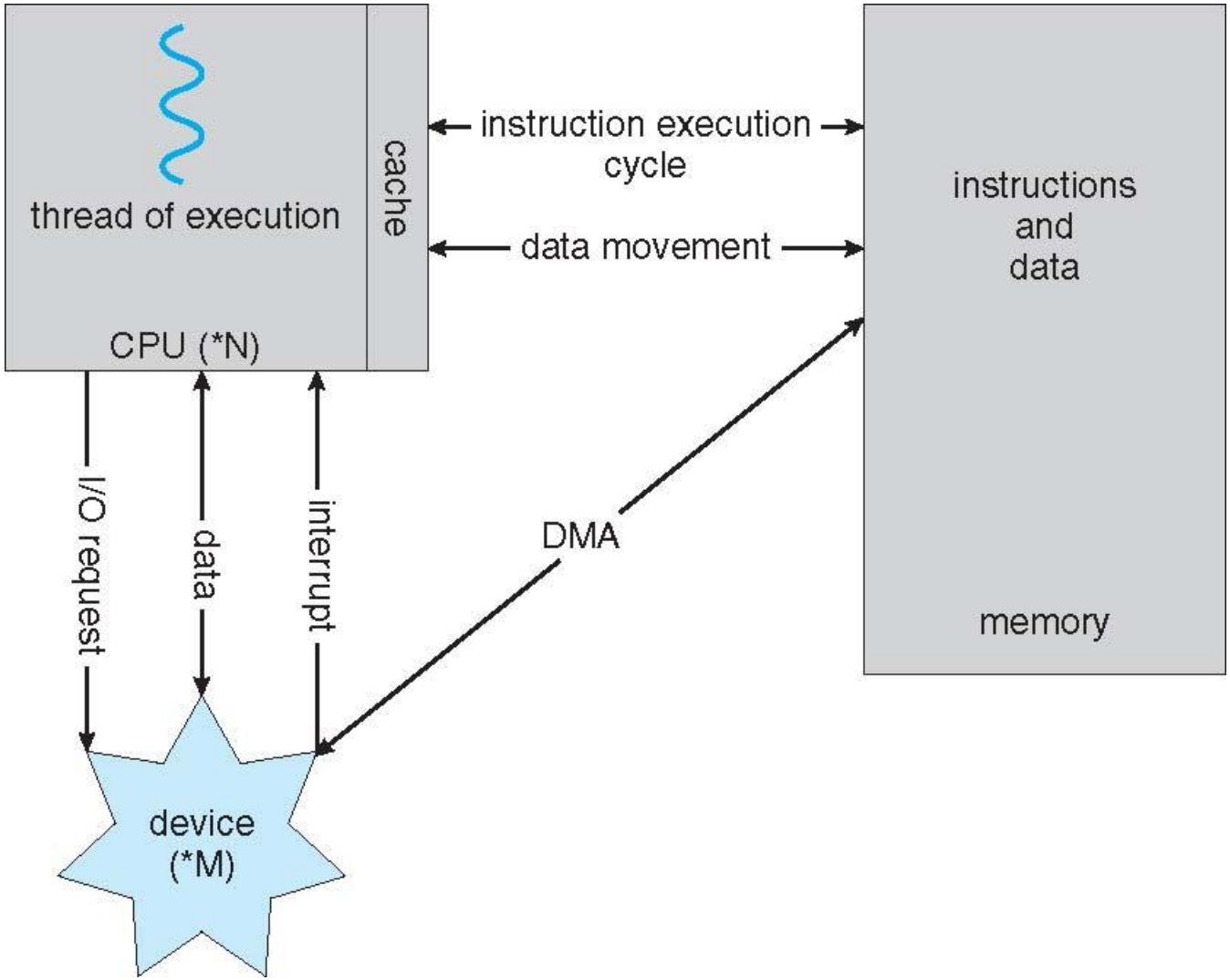
- A paged VMM system typically uses a special-purpose cache, named translation look-aside buffer (TLB)
- Generally very small, highly effective, aside from –in addition to– the regular data and instruction cache
- TLB reduces memory access time by bypassing steps of VMM mapping; speeds up VMM mapping drastically!
- Successful TLB access saves the complete result of multiple mapping steps
- Is part of memory management unit (MMU)
- TLB stores recent translations of virtual memory address to physical memory; can fairly be labeled as an address-translation cache

Direct Memory Access

Commonly referred to as DMA:

- **DMA is IO managed by OS, minimal CPU support**
- **Used for high-speed IO devices able to transmit information fast**
- **Device controller transfers blocks of data from buffer storage directly to or from main memory without CPU intervention; so CPU does "real" work**
- **Caveat: such data cannot be referenced by instructions before DMA completion!**
- **One interrupt is generated per block, rather than one interrupt per datum (e.g. word) to be transferred; i.e. *few* interrupts**

Direct Memory Access



Saves Intervention by CPU Except at Start + End

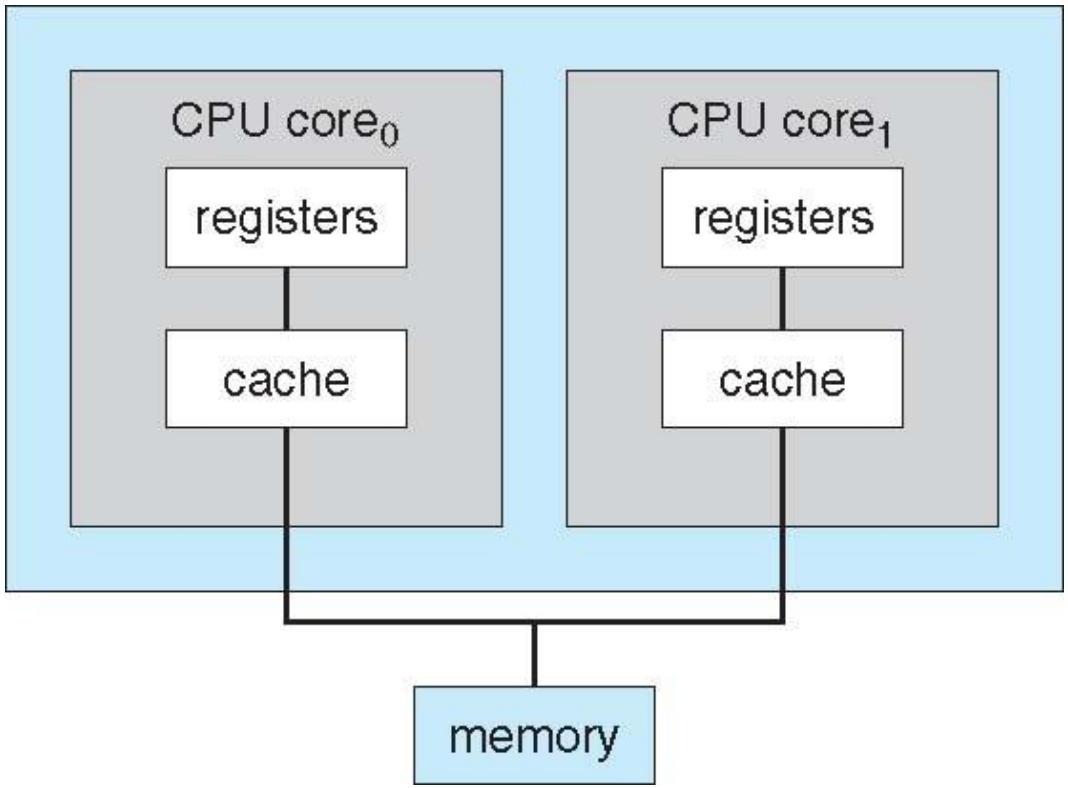
Architecture

Computer System Architecture

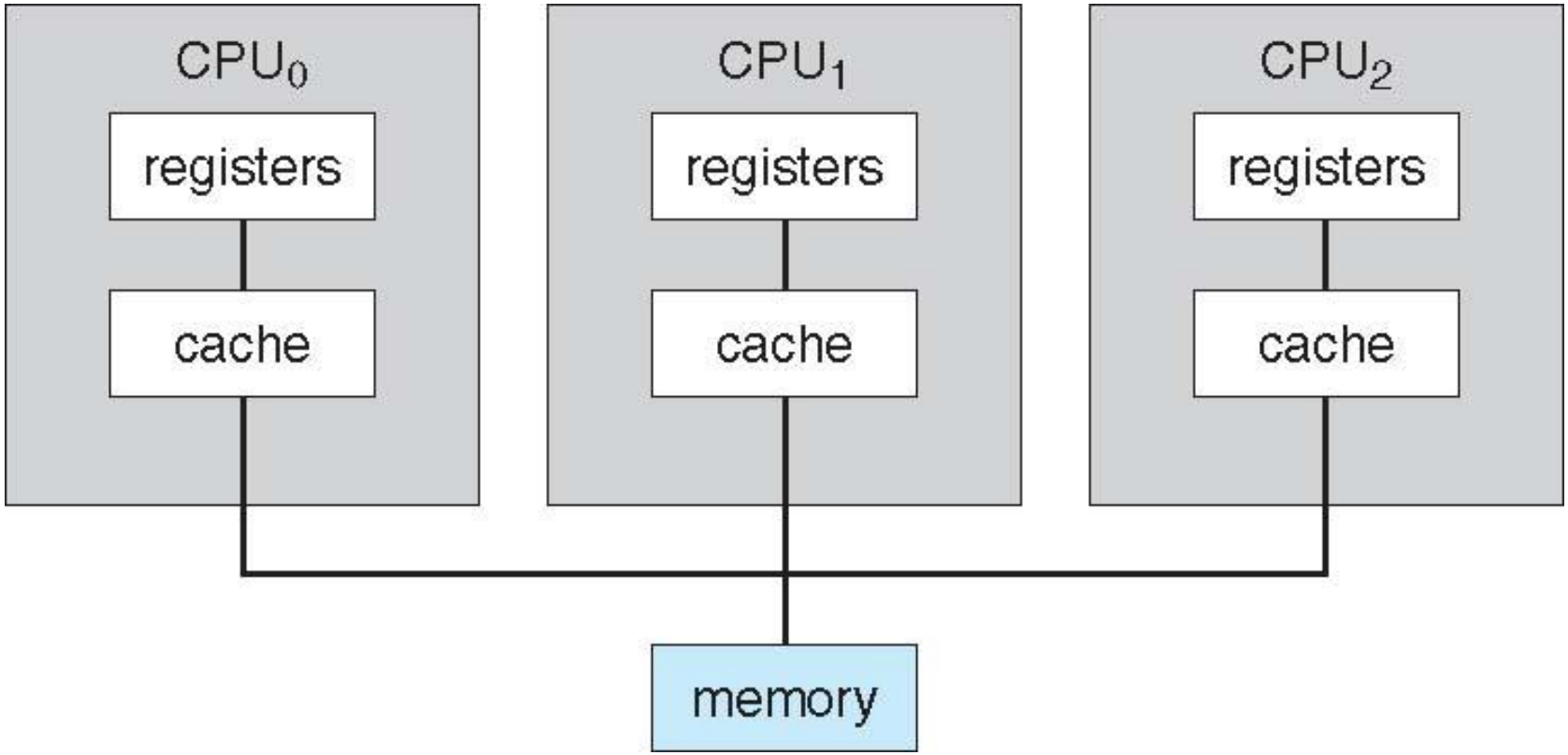
- Older systems use single general-purpose processor
  - Modern systems have special-purpose processors as well
- Multiprocessors systems growing in use and importance; i.e. system having multiple CPUs
  - Also known as parallel systems, MP systems; note: parallel processing is generally a considerable challenge
  - Advantages include:
    1. Increased throughput of computation
    2. Economy of scale
    3. Increased reliability – graceful degradation or fault tolerance
  - Two types:
    1. Asymmetric Multiprocessing – each CPU assigned specific task
    2. Symmetric Multiprocessing – each CPU may perform any task

Dual-Core Design

- **Multi-processor** (multi = 2 here)
- **Synonym: multicore**
- **Any one silicon die holds multiple complete CPUs**



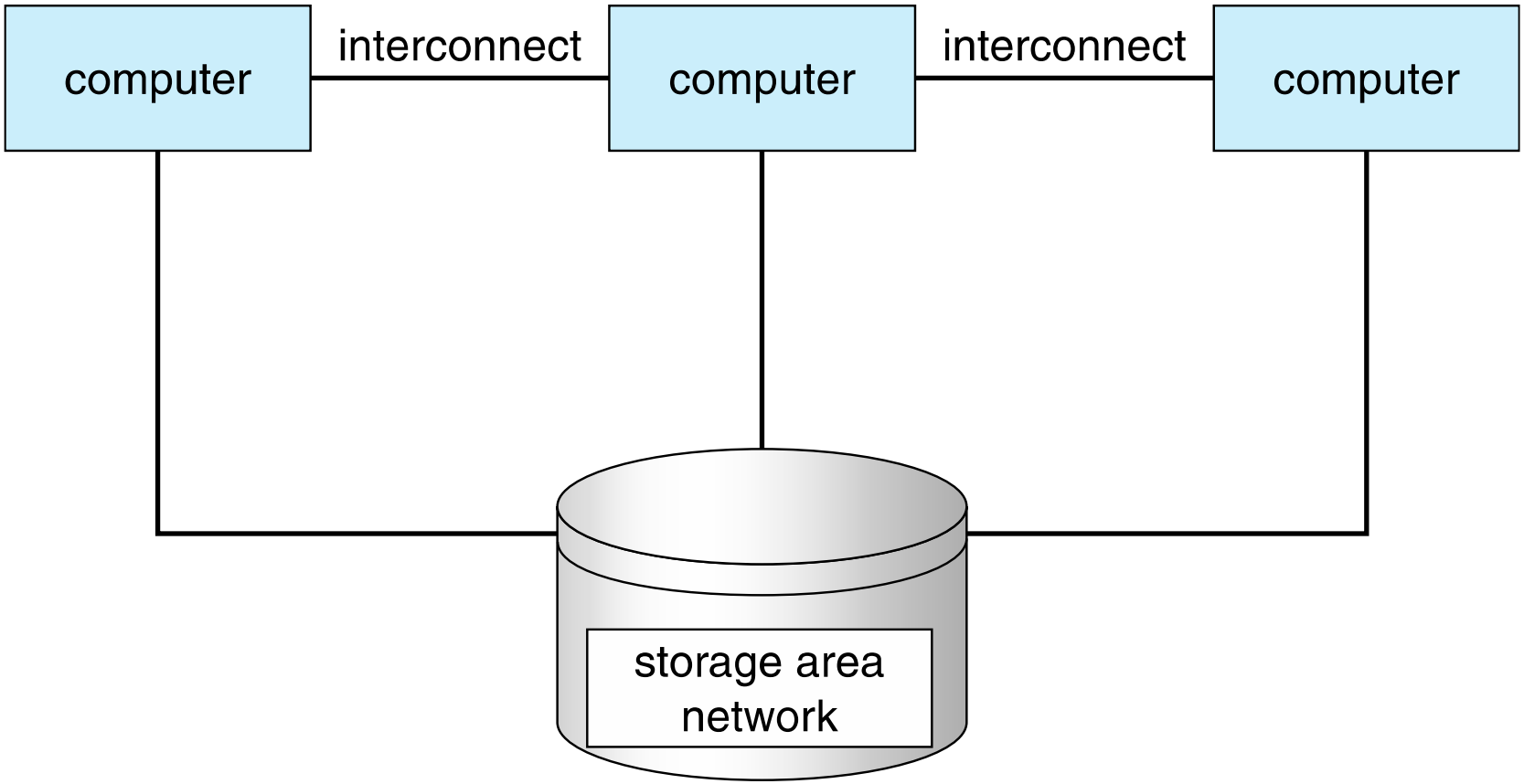
Symmetric MP Architecture



Clustered Systems

- Some similarities with multiprocessor, but uses multiple systems working cooperatively
- Main similarity: Wish to increase overall performance of large compute needs by having multiple CPUs
  - Cluster shares storage via storage-area network (SAN)
  - Provides high-availability service which survives some failures
    - Asymmetric clustering has one machine in hot-standby mode
    - Symmetric clustering has multiple nodes running applications, monitoring each other (note: overhead)
  - Some clusters are for high-performance computing (HPC)
    - Applications must be written to use parallelism (NOT easy!)
    - Many programming languages have UP execution in mind

Clustered Systems



Operating System Structure

Multiprogramming Batch System:

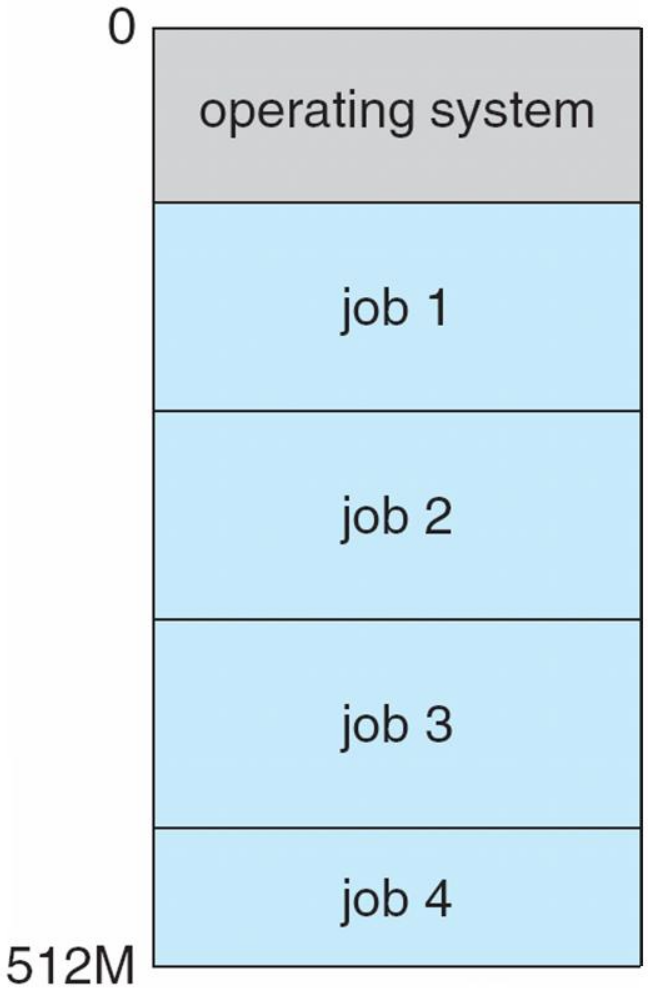
- **Low overhead, efficient**
  - **Single user generally would not keep CPU and IO devices busy**
  - **Multiprogramming organizes multiple jobs so CPU has almost always some job to execute**
  - **Subset of jobs in system is kept in memory; AKA as kept resident!**
  - **One job is selected and run by the job scheduler (scheduler is integral part of OS)**
  - **When process itself causes wait due to IO, OS switches to another process, in order to not waste CPU resources**

Operating System Structure

Timeshared Multitasking:

- **CPU switches jobs so frequently that users can interact with their jobs despite sharing a single CPU, creating interactive computing**
  - **Response time should be < 1 second**
  - **Each user has one or more programs executing in memory ⇨ processes**
  - **If several jobs are ready to run at the same time ⇨ CPU manages schedule**
  - **If processes do not all fit in memory, swapping moves them in/out, i.e. to/from disk**
  - **Virtual memory allows execution of processes using more memory than physically available; with 64-bit addressing: standard MO again, as almost no system has all 2^{64} bytes of physical memory; hence virtualize!**

Multiprogramming Memory Layout

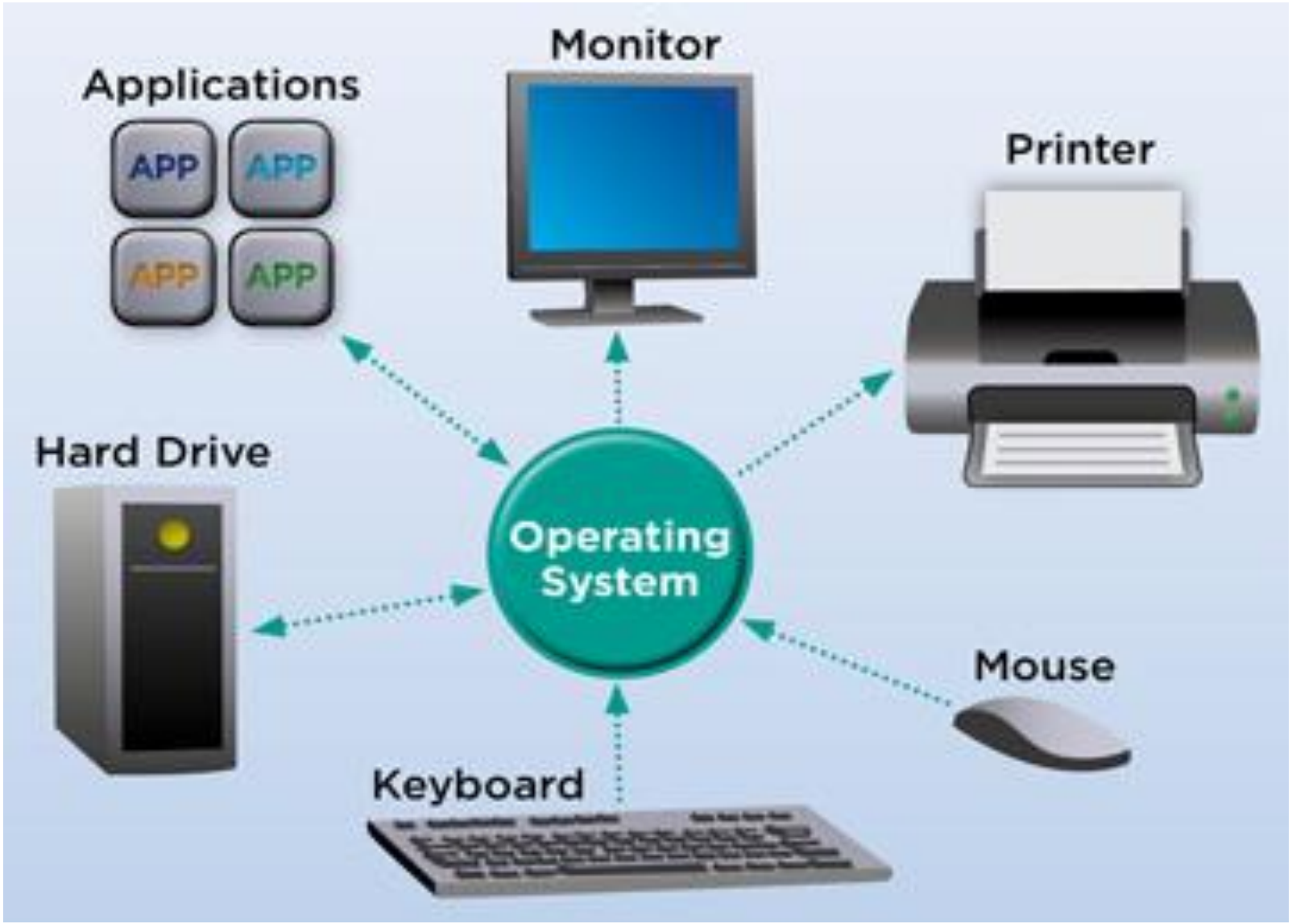


- **Efficient: to keep all "jobs" resident in memory**
- **At times insufficient space to do so**
- **In such cases some remain resident, others are "swapped out"; later "swapped back in", but at high cost of cycles**
- **This solves problem of concurrent execution of multiple, competing "jobs"**

Operating System Functions

- **Interrupt driven (HW as well as SW)**
  - **Condition for interrupt caused by numerous HW devices, or by SW**
  - **Software interrupt (exception or trap):**
    1. **Request for system service e.g. IO**
    2. **Software error (e.g., FP division by zero)**
    3. **Request for other OS service; e.g. mount a tape drive**
    4. **Also could be: process problems, including infinite loops, illegal memory access, FP zero-divide, etc.**

Operating System Functions



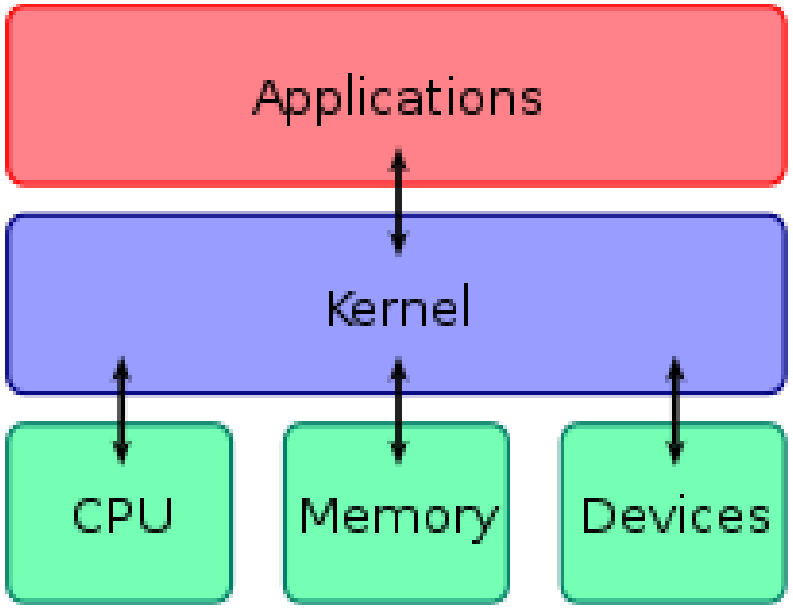
Kernel

Operating System Kernel

- Common today, Dual-mode operation!
- Dual-mode allows OS to protect itself from user error, and other system component errata
  - 1. User mode and 2. Kernel mode
  - Mode bit part of processor architecture, supported by HW
  - 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 system call resets it to user mode
- Increasingly CPUs also support multi processor (MP) operation

Operating System Kernel

- **Kernel: name for that part of an OS that has complete control over all system resources**
- **Is also that "portion of the operating system code that is always resident" in memory**
- **Kernel facilitates interactions between HW and SW components**



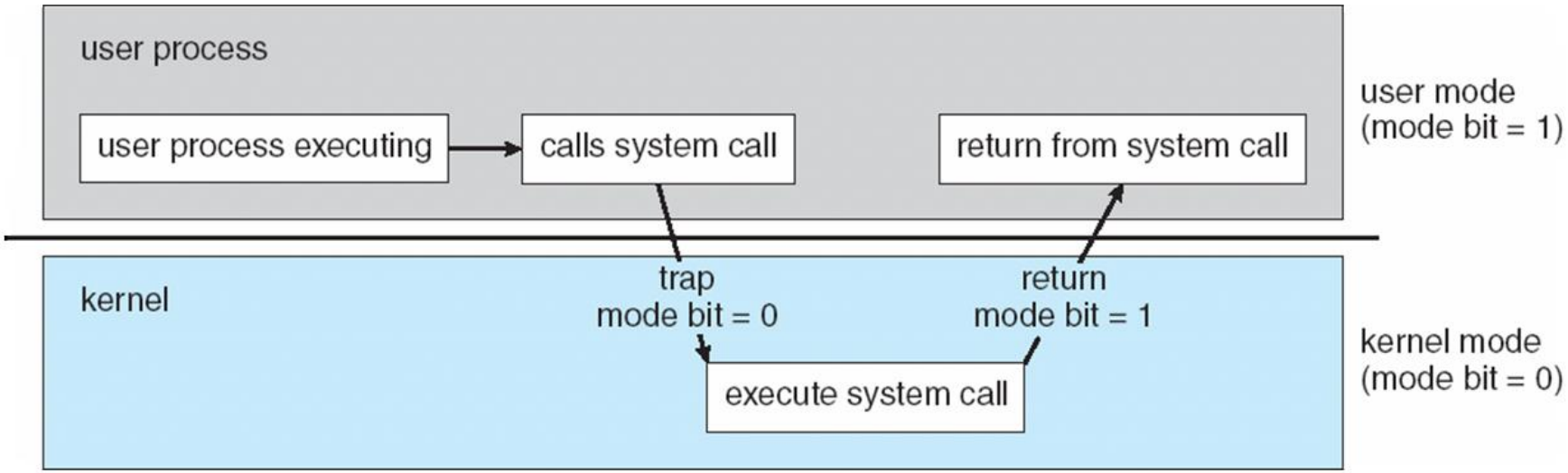
Operating System Kernel

- Generally, kernel is first loaded, right after booting, through special HW device: boot loader
- Critical kernel code is loaded into a separate memory area protected from access by apps, i.e. by users
- Known as kernel space
- Kernel performs tasks, such as: run processes, manage HW devices, handle interrupts
- In contrast, apps use a separate area of memory, known as user space
- This separation prevents/reduces user data and kernel data from interfering with each other

Time Slicing

So called time quantum prevents infinite loop from monopolizing resources, specifically CPU

- **Timer set to interrupt process after set time period**
- **Keep counter decremented by physical clock**
- **Operating system sets the counter initially: privileged!**
- **When counter reaches zero, generates timer interrupt**



Process Management

- *Process* is a program in execution; is the essential unit of work for the OS!
- Process needs resources to accomplish its task
  - CPU, memory, IO, files, bus; above all: The CPU!
  - Data, placement, and initialization of some data (static data)
- Process termination returns reserved resources
- Single-threaded process has one program counter (pc) specifying location of next instruction to execute
  - Process executes instructions sequentially, one at a time, until completion; may: branch, call, return, conditional jump
- Multi-threaded process has one pc per thread
- System has many processes, users, tasks
  - Concurrency by multiplexing CPUs among processes and threads on UP target

Process Management Activities

OS is responsible for the following activities in connection with process management:

- **Creating and deleting processes, both user and system processes**
- **Suspending and resuming processes**
- **Providing mechanisms for process synchronization**
- **Providing mechanisms for process communication**
- **Providing mechanisms for deadlock handling, specifically deadlock prevention!**

Memory Management

- To execute a program, instructions must be resident
- Data needed by program must also be in memory; AKA must be resident
- Memory manager (part of OS) determines what is in memory: when, where, length, priority, which process
  - Optimizing CPU utilization and computer response to users
- Memory management activities
  - Keeping track of which parts of memory are currently used by which process; i.e. who owns which part?
  - Deciding which processes and data move in, out of memory
  - Allocating and de-allocating memory space as needed
  - Increase, decrease physical memory for a process

Storage Management

- OS provides uniform, logical view of storage, i.e. information on disc drives –rotating or SSD or . . .
  - Abstracts physical properties to logical storage unit, e.g. file
  - Each medium controlled by device driver: disk, tape, other
    - Varying, numerous properties include access speed, capacity, data-transfer rate, life-time, access method: sequential-random
- File-System management
  - Files usually organized via a hierarchy of directories
  - Access control is managed: who may access what and how
  - OS activities include
    - Creating and deleting files and directories
    - Primitives to manipulate files and directories, r/w/x
    - Mapping files onto secondary storage
    - Backup data onto stable (non-volatile) storage media

Storage Management

- Disks store data that do not fit in main memory, or data that are to be preserved *long term*
- Speed of computer operation hinges significantly on disk subsystem, its speed, parallelism, algorithms
- OS activities
  - Free-space management of disks
  - Storage allocation and ownership
  - Disk scheduling: which sector is fasted to access?
- Not all storage needs to be fast:
  - Tertiary storage includes optical storage, magnetic tape
  - Still must be managed – by OS or applications
  - Varies between WORM (write-once, read-many-times) and RW (read-write); parameters: speed, cost, volatility

Performance of Storage Media

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

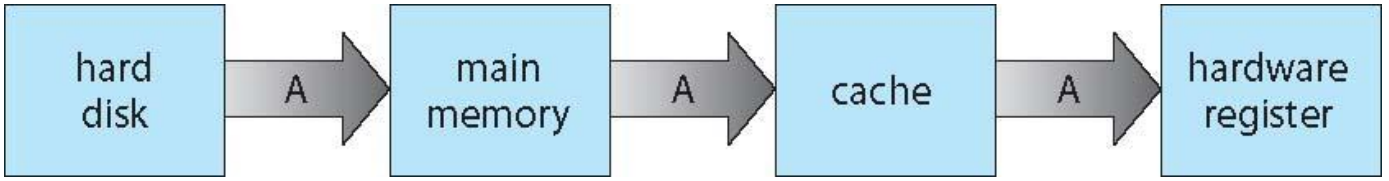
It would be unusual to name registers: storage media

Movement between levels of storage hierarchy can be explicit (by user) or implicit (by OS)

CMOS: Complementary Metal Oxide Semiconductor

Data Migration Disk to Register

- Compute environment must be careful to use most recent value, no matter where it is stored in hierarchy



- UP system has coherence challenge: regs, L1, L2, mem.
- MP environment even harder: cache coherence in HW such that all CPUs have (or use) the most recent value from their cache; copy in memory can be stale
- Distributed environment situation even more complex
  - Several more copies of any one datum can exist
  - Creates so called cache coherence challenge! Even on UP!
  - Complex on MP, where each CPU has multiple levels

IO Subsystem

- **One purpose of OS: Hide peculiarities of HW devices from user; referred to as Device Abstraction!**
- **IO subsystem responsible for**
  - **Memory management of IO including buffering (storing data temporarily local disk memory during transfer)**
  - **(Generic) Caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs)**
  - **SPOOL Simultaneous Peripheral Operations On-Line**
  - **Disk drives have their own cache (distinct from L1, L2 etc. of CPU) for faster IO – cache here: faster than disc access!**
  - **General device-driver interface**
  - **Drivers for specific HW devices**

Protection and Security

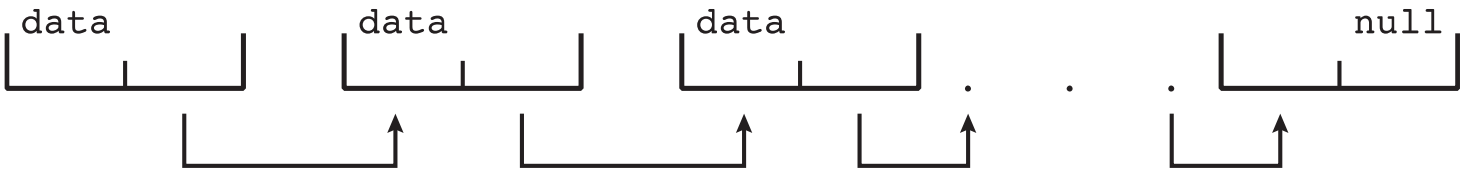
Protection and Security

- **Protection:** Is a SW or HW mechanism for controlling access to system resources, including processes!
- **Security:** Designed defense of system against attacks
  - Huge range of attacks practiced: worms, viruses, identity theft, password-theft, theft of service
  - Systems distinguish among users, to determine who has permission to do what
  - User identities (user IDs, security IDs) include name and associated number; one ID per user!
  - User ID then is associated with own files, and with processes of that user to control legal access
  - Group identifier (group ID) allows multiple users to be defined, each of which may manage processes, files, etc.
  - Privilege escalation allows user to gain more rights, in a managed way: the OS "knows" or tracks such escalation

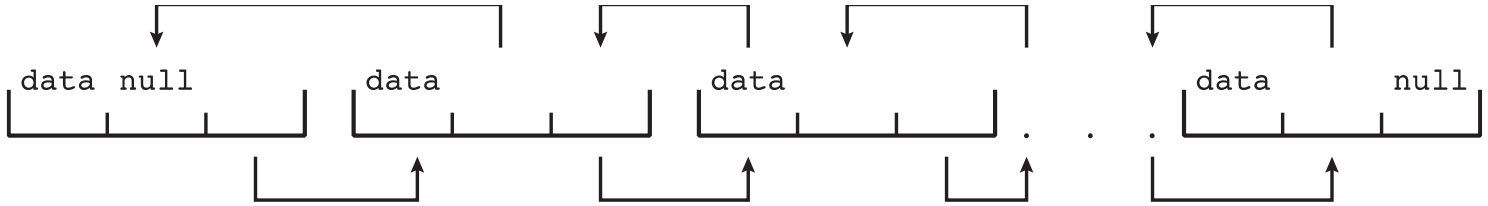
OS Data Structures

Similar to standard programming data structures, e.g.:

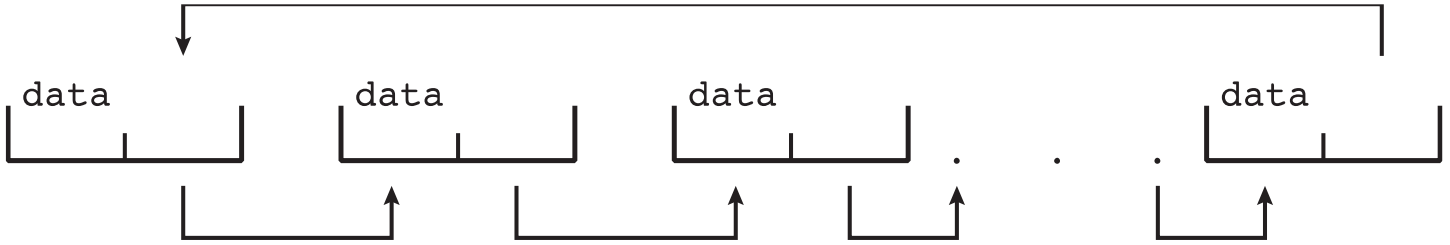
- ***Singly linked list***



- ***Doubly linked list***



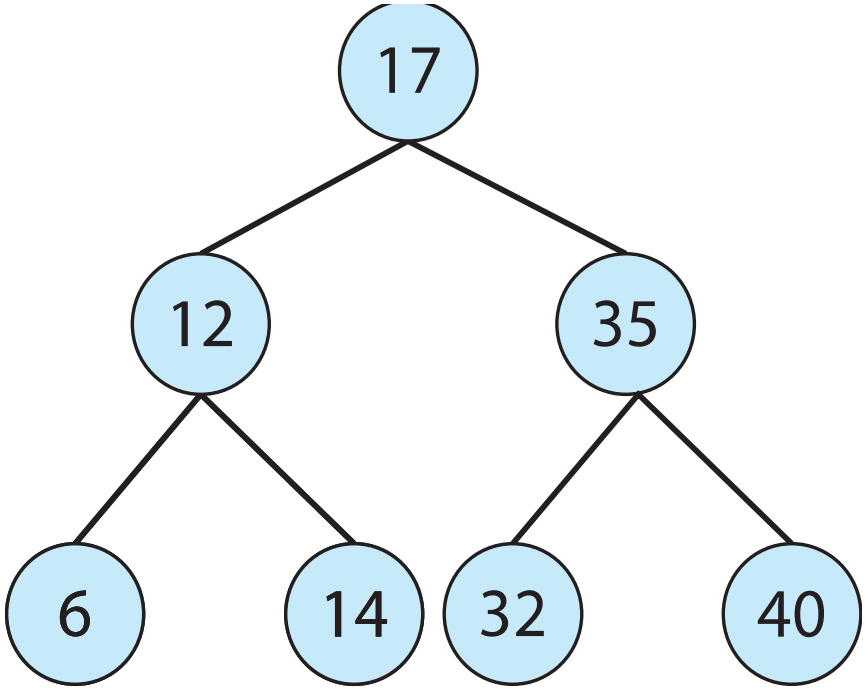
- ***Circular linked list***



OS Data Structures

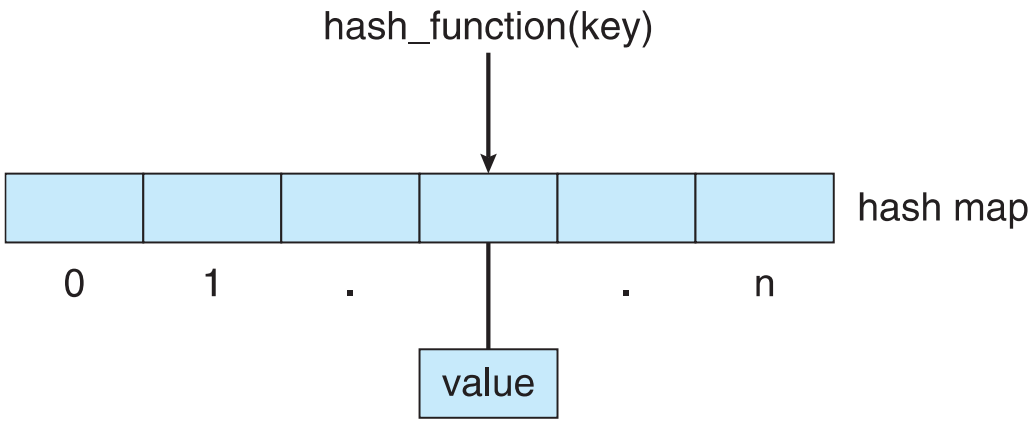
Binary search tree, root and left and right subtree

- **Search performance in tree can be as bad as *O(n)***
- ***Balanced binary tree* search is *O(lg n)***



OS Data Structures

- **Hash function used datum to compute (hash) its entry; can "collide"**



- **Bitmap or hash of value: An n-bit index computed from the searched object itself, index being supposed location of that searched –or to be stored– object; may be wrong**
- **As it is non-unique, thus causes conflicts, AKA "collisions"**
- **After collision, another entry is computed; can become slow**
- **Parameterized by fill factor; should be << threshold; typical < 80%**
- **Linux data structures defined in**

  ***include* files** `<linux/list.h>, <linux/kfifo.h>, <linux/rbtree.h>`

Compute Environments

Compute Environments – Traditional

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

Compute Environments – Large Scale

Compute Environments – Mobile

- **Handheld smartphones, tablets, etc.**
- **What is functional difference between them and a laptop?**
- **Extra OS features on Mobiles: GPS, gyroscope**
- **Use IEEE 802.11 wireless, or cellular data networks for connectivity**
- **Allows new types of apps like *augmented reality***
- **Current technology leaders are Apple iOS and Google Android**

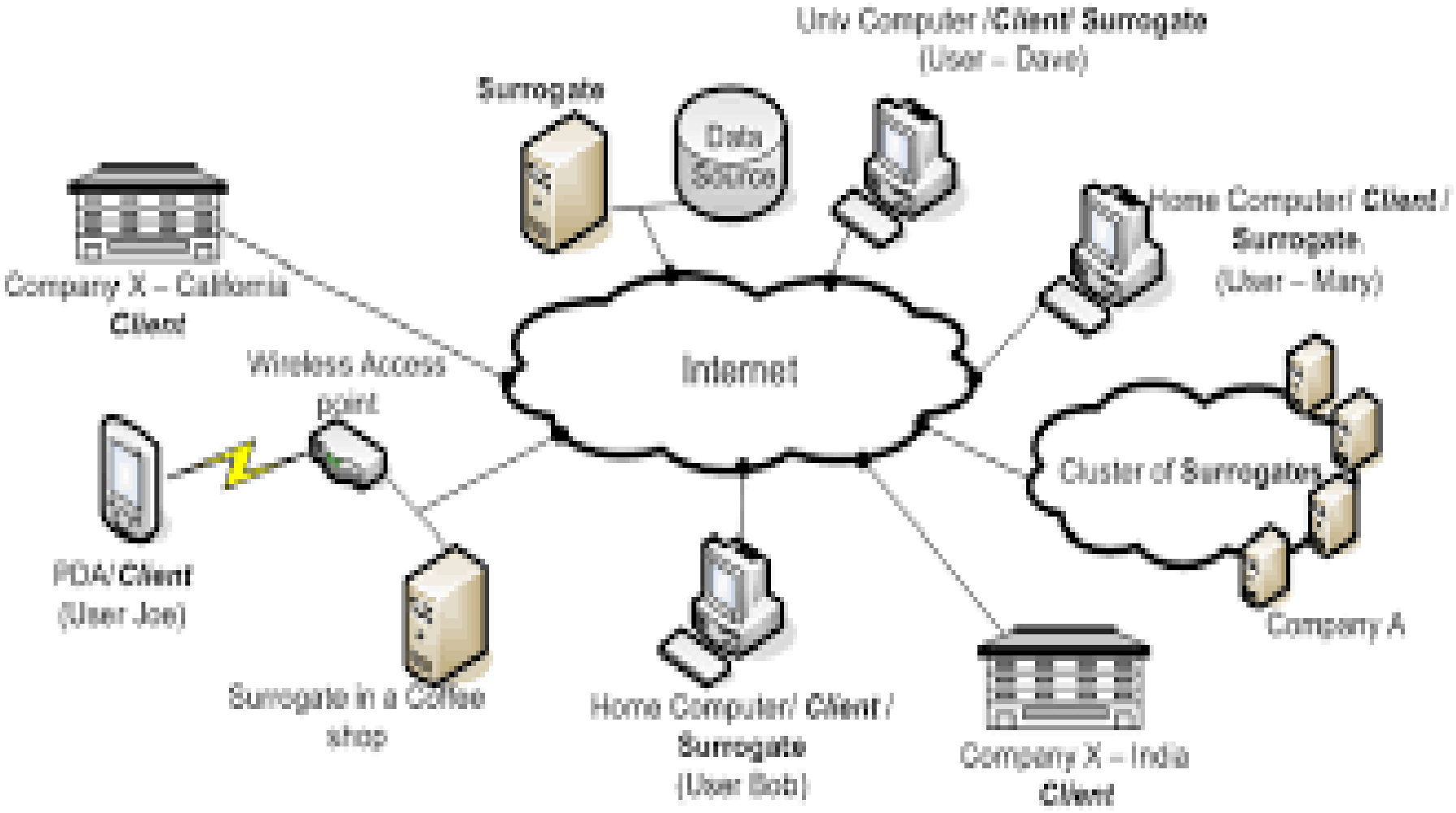
Compute Environments – Mobile

Compute Environments – Distributed

Distributed computing:

- Collection of separate, possibly heterogeneous, systems networked together
  - Network is a communications path, TCP/IP most common: Transmission Control Protocol, Internet Protocol; see [5]
  - Local Area Network (LAN)
  - Wide Area Network (WAN)
  - Metropolitan Area Network (MAN)
  - Personal Area Network (PAN)
- Network Operating System provides communication between systems across network
  - Communication scheme allows systems to exchange messages, commands, data; thus to cooperate!
  - Illusion of single system for certain applications

Compute Environments – Distributed



Compute Environments – Client-Server

Meaning of Client-Server Computing

- Dumb terminals of old: supplanted by smart PCs
- In client-server computing, a server receives requests from connected client computers and shares server resources, with client computers on the network
- A client computer initiates contact with a server to use shareable resources that client lacks, but needs
- Many systems now are servers, responding to requests generated by clients
- Compute-server system provides an interface to client who requests services, such as database access and queries, or compute-intensive tasks
- File-server system provides interface for clients to store and retrieve files; file file access rights are to be observed

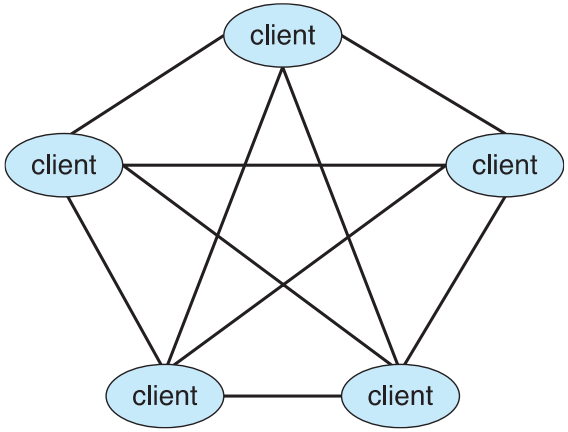
Client-Server Computing

Webopedia:
https://www.webopedia.com/Computer_Science/Client_Server_Computing

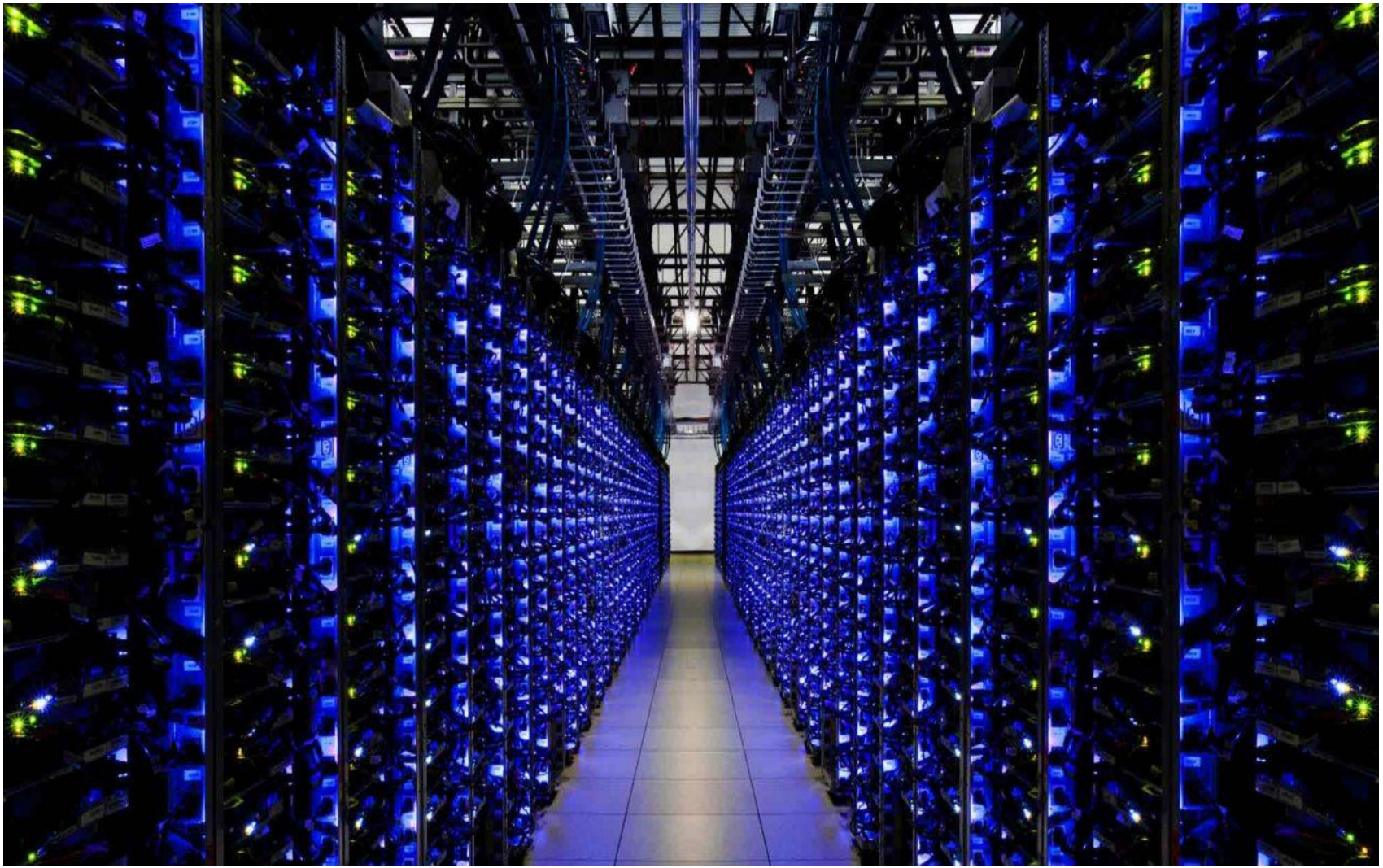
In client/server computing, a server takes requests from client computers and shares its resources, applications and/or data with one or more client computers on the network, and a client is a computing device that initiates contact with a server in order to make use of a shareable resource

Compute Environments – 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 or server or both
  - Node must join P2P network!
    - Registers its service with central lookup service on network; or
    - Broadcasts request for service and responds to requests for service via *discovery protocol*
  - Examples include Napster and Gnutella, Voice over IP (VoIP) such as Skype!!
  - Great tool when working!!



Compute Environments – Virtualization



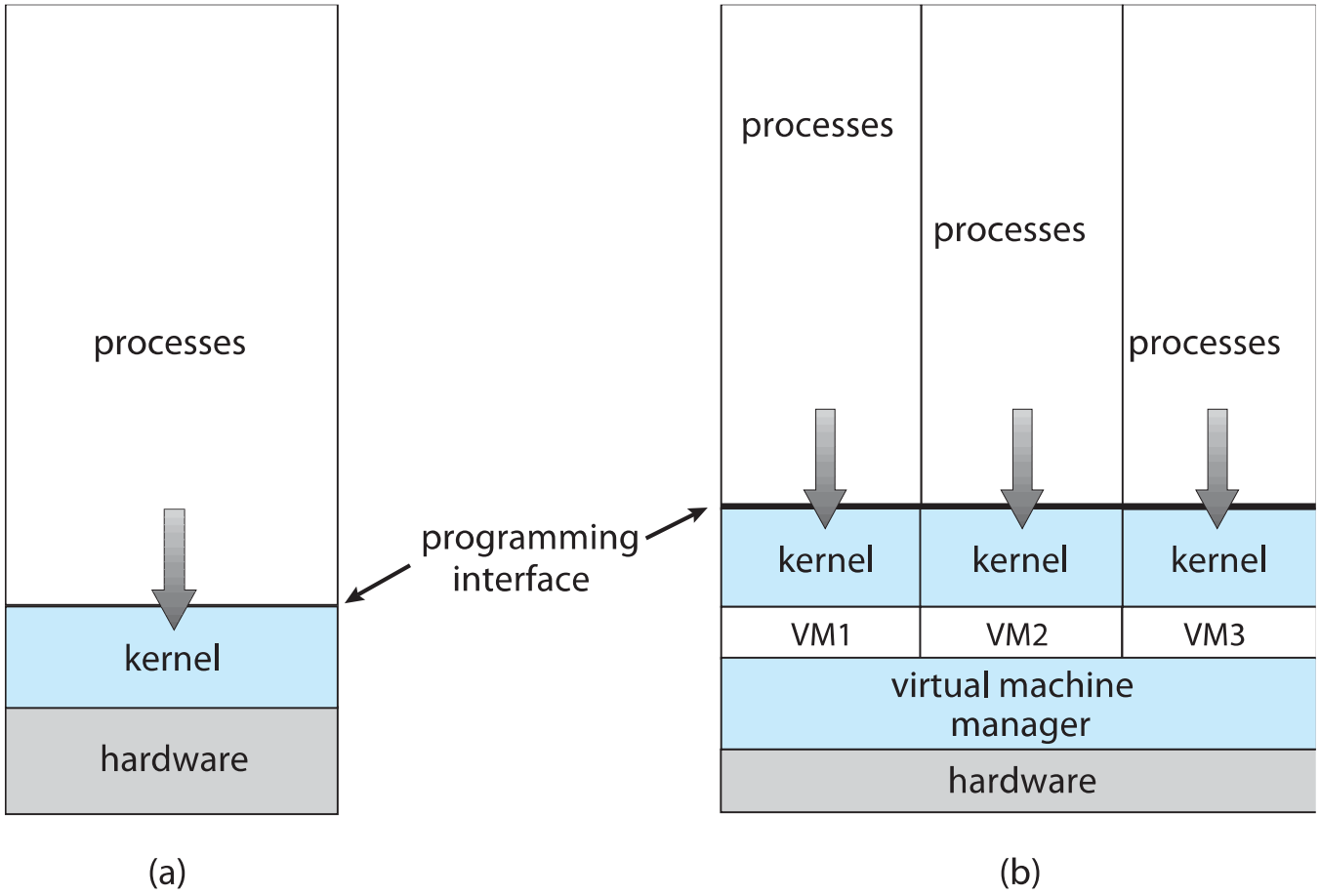
Compute Environments – Virtualization

- Allows one OS to run applications within another OS
  - Vast and growing industry
- Emulation used when source CPU type different from target type (i.e. PowerPC to Intel x86); can be costly!
  - Generally slowest method dictates performance: Bottleneck
- When source program is not compiled to native code: Interpretation
- Virtualization: host OS natively compiled for CPU, running guest OS, also natively compiled
  - Consider VMware running WinXP guests, each running applications, all on native WinXP host OS
  - VMM (Virtual Machine Manager) provides virtualization services; VMM ref, see [4]

Compute Environments – Virtualization

- Use cases involve laptops and desktops running multiple OSes for exploration or compatibility
  - Apple laptop running Mac OS X host, Windows as a guest
  - Developing apps for multiple OSes without having multiple systems
  - QA testing applications without having multiple systems
  - Executing and managing compute environments within data centers
- VMM can also run natively, can be host OS
  - Then there is no (and also no need for) a general purpose host OS

Compute Environments – Virtualization



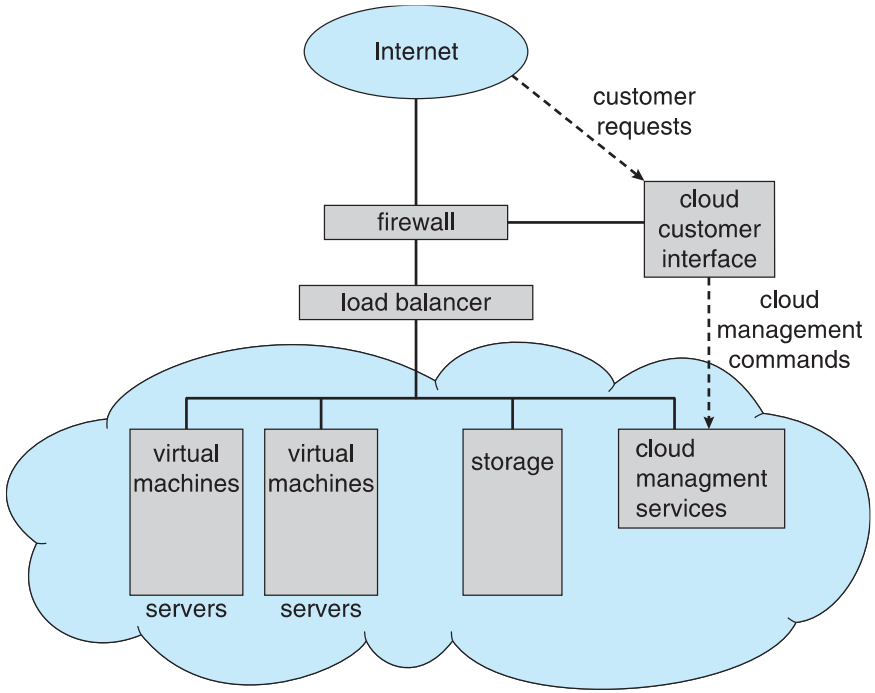
Compute Environments – Cloud

- Delivers computing, storage, apps as service across networks
- Logical extension of virtualization: use virtualization as base for function
  - 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; good for massive data to be physically separate from developing organization, in case of disaster!

Compute Environments – Cloud

Cloud computing environments: traditional OSes, plus VMs, plus cloud management tools

- **Internet connectivity requires security like firewalls**
- **Load balancers spread traffic across multiple applications**



Computing Environments – RT Embedded

- Real-time (RT) often used in embedded systems
- 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
- RT OS has well-defined fixed time constraints!
  - Processing *must* be done within that time constraint
  - Correct operation only if constraints are met
  - If time constraint not met, generally catastrophic consequences, such as data loss; even lives lost

Open Source OS

Open Source Operating Systems

- Operating systems made available in source-code format rather than just binary closed-source
- Counter to Copy Protection © and Digital Rights Management (DRM) movement
- Open Source started by Free Software Foundation (FSF): note "copyleft" GNU Public License (GPL)
- See: Richard Stallman licenses, ca. 1985
- Examples include GNU/Linux and BSD Unix
- Can use VM like VMware Player (Free on Windows), Virtualbox; Open Source and free on many platforms
  - http://www.virtualbox.com
  - Used to run guest operating systems for exploration

Unix Origin

- ~1970, Kenneth Thompson, Dennis Ritchie, and others at AT&T, Bell Labs began developing a small operating system (also: Brian Kernighan)
- For a little used ☺ minicomputer PDP-7 at Bell Labs
- Operating system was soon christened Unix
- As a pun on earlier operating system, a gigantic project called MULTICS, developed by GE
- MULTICS had numerous features, great features but had awful performance due to excessive SW complexity
- Incurred high development cost

Unix Creators

Ken Thompson & Dennis Richie

Unix Origin

- **Unix originally was a one man project led by Ken Thompson of Bell Labs, and has since grown to become a widely used OS**
- **Unix has gone through numerous, different generations and even mutations**
  - **Some differ substantially from the original version, like Berkeley Software Distribution (BSD)**
  - **. . . Or like Linux**
  - **Others, still contain major portions that are based on the original source code**

Characteristics of Unix

- **Multi-user & Multi-tasking: Unix capable of allowing multiple users to log onto the system, each running multiple tasks; standard for modern OSes**
  - **Matter of course today, was novelty during inception**
- **Unix is ancient, antique, yet it's popularity is still high**
- **Many variations have spawned off, some died off, but most modern Unix systems can be traced back to the original version**
- **There is an grand amount of applications available for Unix OSes**
- **These range from commercial applications such as CAD, Maya, WordPerfect, to many free applications**

Characteristics of Unix

- Of the many applications available under Unix, most are free; free of monetary license requirements
- Moderately resource Intensive: Generally, Unix installations tend to be *tolerably demanding* on system resources
- The “old and dusty family computer” that barely runs Windows is often sufficient to run the latest version of Linux
- Internet Development: Much of the backbone of the Internet is run by Unix servers
- Many web servers run Unix with the Apache web server - another free application
- Apache now ~27 years old; see [9]

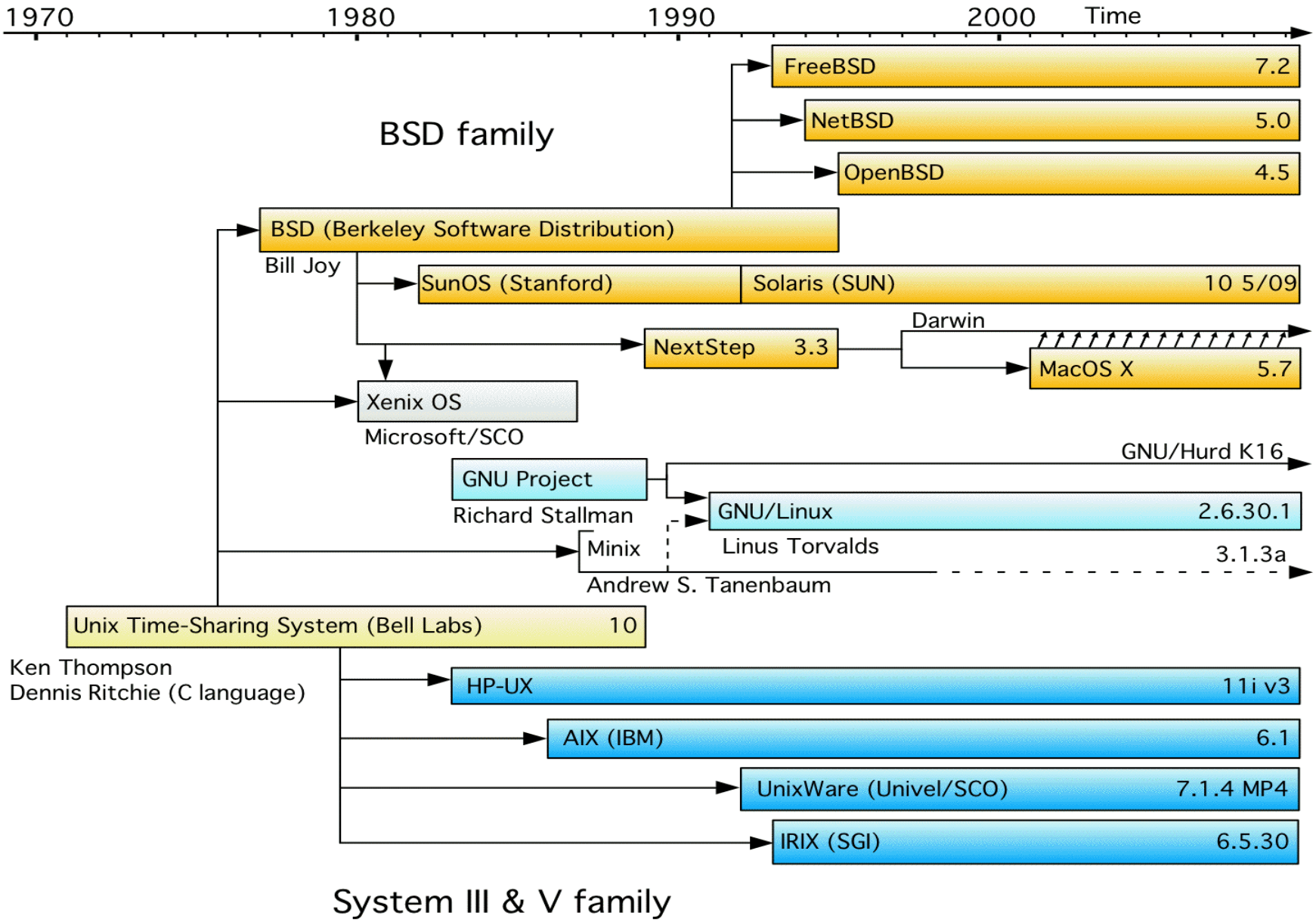
Unix Modules

- Built-in System Utilities allow users to perform tasks which involve complex actions
- Utilities provide user interface functions, basic to an OS, but too complex to be built into the shell
- Examples of utilities are programs that let us view the contents of a directory, move & copy files, remove files, etc...
- Application Software & Utilities are not part of OS
- They are additional programs, often bundled with the OS distribution, or available separately
- Can range from basic utilities, to full scale commercial applications, some available for a fee

Unix Modules

- **Kernel handles memory management, IO, and process scheduling**
- **In a sense, the Unix kernel *is* the OS:**
- **It provides SW connection to the HW**
- **Kernel is complex piece of system SW, even sometimes named a micro-kernel ☺**
- **Shell and Graphical User Interfaces (GUIs): Unix shells provides a "command line" interface which allows the user to type in commands**
- **Such commands are translated by the shell into something the kernel can comprehend**
- **These commands are executed by the kernel**

Unix Evolution [6]



Connect to Server

- **Unix server:**
  - **One can logon from virtually any computer that has internet access, be it Windows, Mac, or Unix itself**
  - **Log on to a remote server**
- **In this case, communication proceeds via a local terminal to one of these remote servers**
  - **All of the commands actually execute on the remote server**
  - **It is also possible to open up graphical applications through this window, but that requires a good bit more setup and extra SW**

Linux vs. Windows

- **OS does not have to use a graphical interface**
  - **The OS itself (the kernel) is relatively small**
  - **The GUI just another set of applications that can be installed and run on top the existing text-based OS**
- **File system differences**
  - **Windows typically uses FAT32 or NTFS file systems**
  - **Linux typically uses the ext2 or ext3 file systems**
  - **In large research and university environments, where file access is necessary across the network, something like Network File System (NFS) or the Andrew File System (AFS) is used**
  - **Windows lists all drives separately (A:,C:,D:, etc…), with "My Computer" at the highest level**
  - **Unix starts its highest level at "/" and drives can be mounted anywhere underneath it**

X Window

- X Window originated around 1984 at MIT [8]
- X Window draws windows on the screen under most GUI-based versions of Unix
- X windows language is different from that of Microsoft Windows or Mac OS X
- X window system consists of 2 distinct parts - the X server and some X clients:
  - The server controls the display directly, and is responsible for all input/output via the keyboard, mouse or display
  - Clients do not access the screen directly - they communicate with server, handling all IO
  - Clients do "real" computing, running apps. Etc.
  - Clients communicate with the server, causing the server to open one or more windows to handle IO for client

Desktop Manager

- Gnome and KDE are examples of desktop managers. Both of these show similarities with MS Windows
  - They have the equivalent of a Start Menu, have an equivalent of Windows Explorer, and have some sort of control panel
- KDE [7]: SW developer community with goal to share Unix based SW tools & products
- Desktop Manager provides ability to manage details of your system that would otherwise require specific commands in a terminal window
  - These details include managing your files, launching programs, configuring various aspects of your system, etc.
- Desktop manager is optional
- Some older systems lack desktop manger that connected X server and Window manager

Window Manager

- **The Window Manager manages the placement of Windows on your system**
- **Makes it possible to move, resize, and minimize the various programs running on your computer**
- **KDE handles this functionality as well, whereas Gnome does not directly provide this functionality, but rather relies on an independent window manager to do it**

Window Manager

- **Think of the Window Manager as the framing around the windows as well as all of the associated functionality that they provide**
  - **For example, most all window managers can close, minimize, maximize & resize**
  - **However most Unix window managers add so much more in the way of functionality**
  - **“Decoration” and customization of these windows under Unix tends to be much more flexible**
  - **Many Window Managers also provide shading, sticky/nonsticky, window history, and desktop and workspace manipulations**

Notes on X window

- Most Unix systems can be installed without the GUI
- The GUI is just another application that runs on top of the operating system
- There exist many implementations of all three of these components
  - It is possible to mix and match implementation and versions of these
  - They need not be alike and need not be all by the same organization
- Quite a paradigm shift from Microsoft and Apple

Linux Creator

Linus Torvalds, born 1969, in Helsinki

Tools under Linux

- **Text Editors**
  - **Xemacs**
  - **Emacs**
  - **Pico**
  - **vi (before there was ed)**
- **Compilers**
  - **C compiler – gcc or cc**
  - **C++ compiler – g++**
  - **Java compiler & Java Virtual Machine – javac & java**
- **Debuggers**
  - **C / C++ debugger - gdb**
- **Interpreters**
  - **Perl – perl**
  - **Tcl/Tk – tcl & wish**
- **Miscellaneous**
  - **Web Browsers - Mozilla, Netscape, Firefox, and Lynx**
  - **Instant Messengers – Gaim**
  - **Email – Netscape, Pine**

Summary

- OS is manager of computing resources: SW and HW
- Introduced very high-level, abstract computer architecture
- Reviewed OS structure and some OS operations
- Defined simple view of process and memory management
- Key of protection and security: grant access to compute resources only to authorized entities, and to define which form of access be allowed
- Introduced virtualization, compute environment and process definition

Bibliography

1. History: https://en.wikipedia.org/wiki/History_of_operating_systems
2. Origin of Unix: https://en.wikipedia.org/wiki/History_of_Unix
3. Multics: https://en.wikipedia.org/wiki/Multics
4. Virtual Machine Manager: https://www.infoblox.com/glossary/virtual-machine-manager-vmm/
5. TCP/IP: https://en.wikipedia.org/wiki/Internet_protocol_suite
6. https://en.wikipedia.org/wiki/Unix
7. KDE: https://en.wikipedia.org/wiki/KDE
8. X Window: https://en.wikipedia.org/wiki/X_Window_System
9. Apache web server: https://httpd.apache.org/