Introduction to Operating System

Alok Jhaldiyal
Assistant Professor
SoCS, UPES
Dehradun

General Information

- Textbook:
  - Operating System Concepts, 8th or 9th Ed, by Silberschatz, Galvin, and Gagne
- Reference Books:
  - Operating Systems: Principles and Practice by Anderson and Dahlin
  - Modern Operating Systems by Andrew Tanenbaum
- Programming assignments will be covered in associated Lab.

Our Exposure to various OS's

watch OS
TIZEN
fitbit
GARMIN
AMAZFIT
HUAWEI LiteOS
Mac OS
android
BlackBerry
webOS
bada
symbian OS
Windows Phone

Computer System

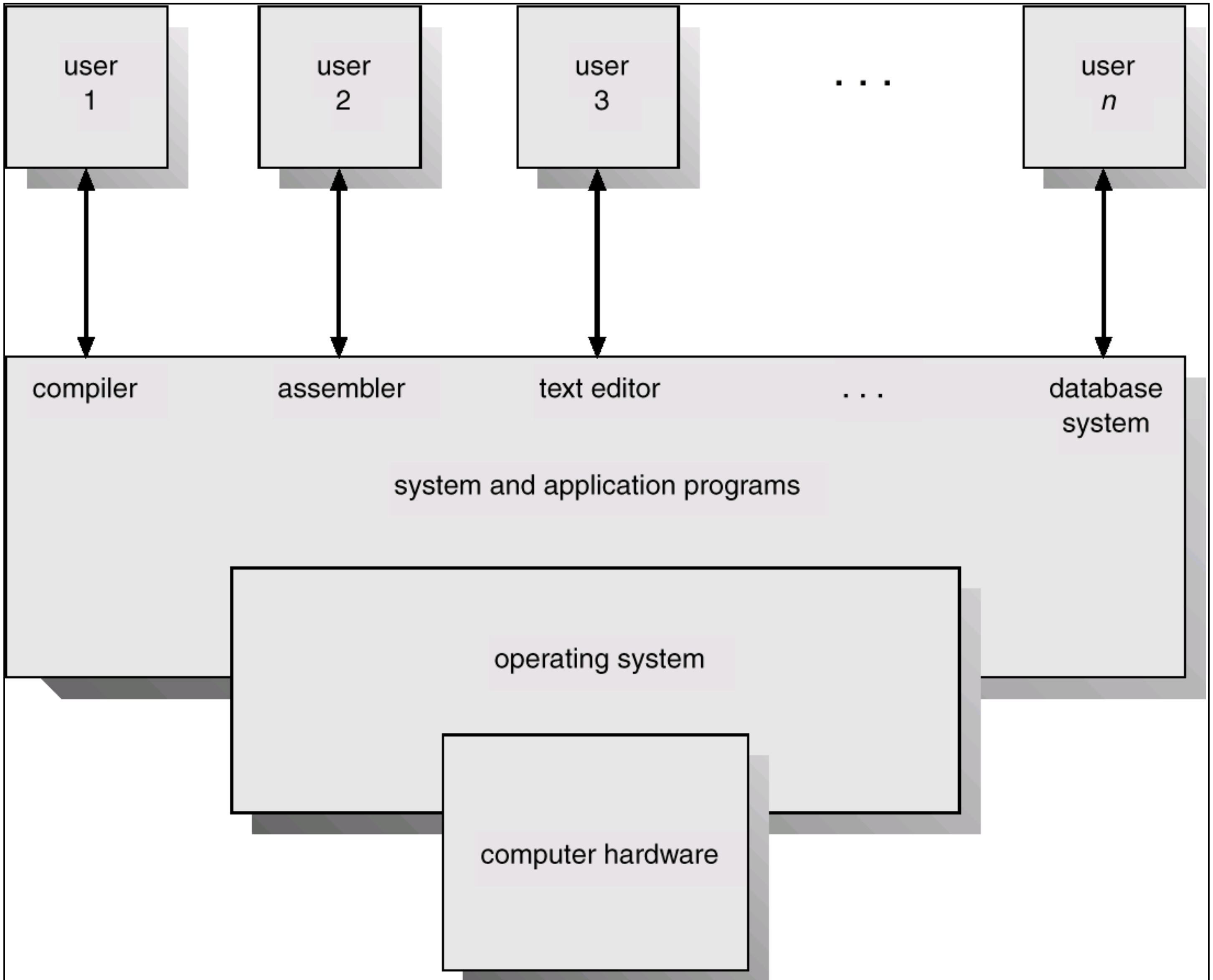
- computer system can be divided roughly into four components:
  - **Hardware**
    - The **central processing unit (CPU)**, the m**emory**, and the **input/output (I/O) devices**
    - Provides the basic computing resources for the system
  - **Operating System**
    - Controls the hardware and coordinates its use among the various application programs for the various users
    - Operating System provides the means for proper use of these resources.
  - **Application Programs**
    - Define the ways in which these resources are used to solve users' computing problems
    - Application Programs: Word Processors, spreadsheets, compilers, browsers……
  - **Users**

What is operating system?

- User-centric definition
  - A program that acts as an intermediary between a user of a computer and the computer hardware
  - Defines an interface for the user to use services provided by the system
  - Provides a “view” of the system to the user Converts what the hardware gives to what the user wants
  - The view can hide many details of the hardware that the user does not need to know
  - Can even give a very different view of the operating environment to the user than what is actually there

- System-centric definition
  - Efficiently manages and allocates resources to users
  - Controls the execution of user programs and operations of I/O devices
  - Provides isolation/protection between different user programs

Abstract View of System Components



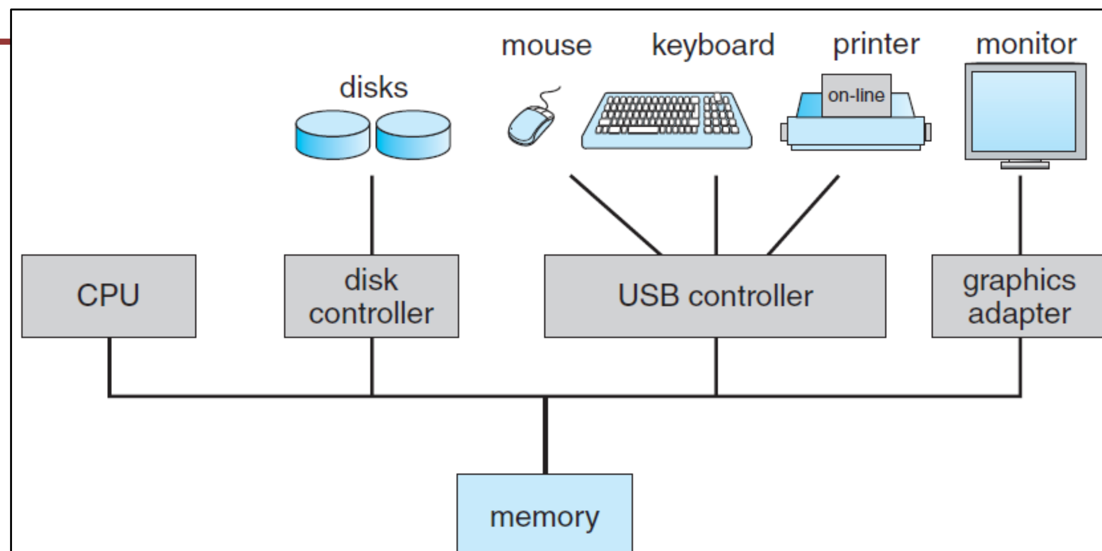
Types of System

- Batch Systems
  - Multiple jobs, but only one job in memory at one time and executed (till completion) before the next one starts
- Multiprogrammed Batch Systems
  - Multiple jobs in memory, CPU is multiplexed between them
  - CPU-bound vs I/O bound jobs
- Time-sharing Systems
  - Multiple jobs in memory and on disk, CPU is multiplexed among jobs in memory, jobs swapped between disk and memory
  - Allows interaction with users

- Personal Computers
  - Dedicated to a single user at one time
- Multiprocessing Systems
  - More than one CPU in a single machine to allocate jobs to
  - Symmetric Multiprocessing, NUMA machines …
  - Multicore
- Other Parallel Systems, Distributed Systems, Clusters…
  - Different types of systems with multiple CPUs/Machines
- Real Time Systems
  - Systems to run jobs with time guarantees
- Many other types
  - Embedded systems, mobiles/smartphones, ….

Computer System Architecture

- Computer system consists of one or more CPUs and a number of device controllers, which execute in parallel.
- Each device controller is in charge of a specific type of device.
- For orderly access to the shared memory, a memory controller synchronizes access to the memory



- Bootstrap program
  - Once computer is powered on, to start running a initial program: Bootstrap Program
  - Initializes all aspects of the system, from CPU registers to device controllers to memory contents.
  - Stored within the computer hardware in read-only memory (**ROM**)
  - Bootstrap program locates the operating-system kernel and loads it into memory
  - A loaded and executing kernel provides services of the system to the user.
- Some services are provided outside of the kernel, by system programs that are loaded into memory at boot time to become **system processes**

Bootstrap Steps

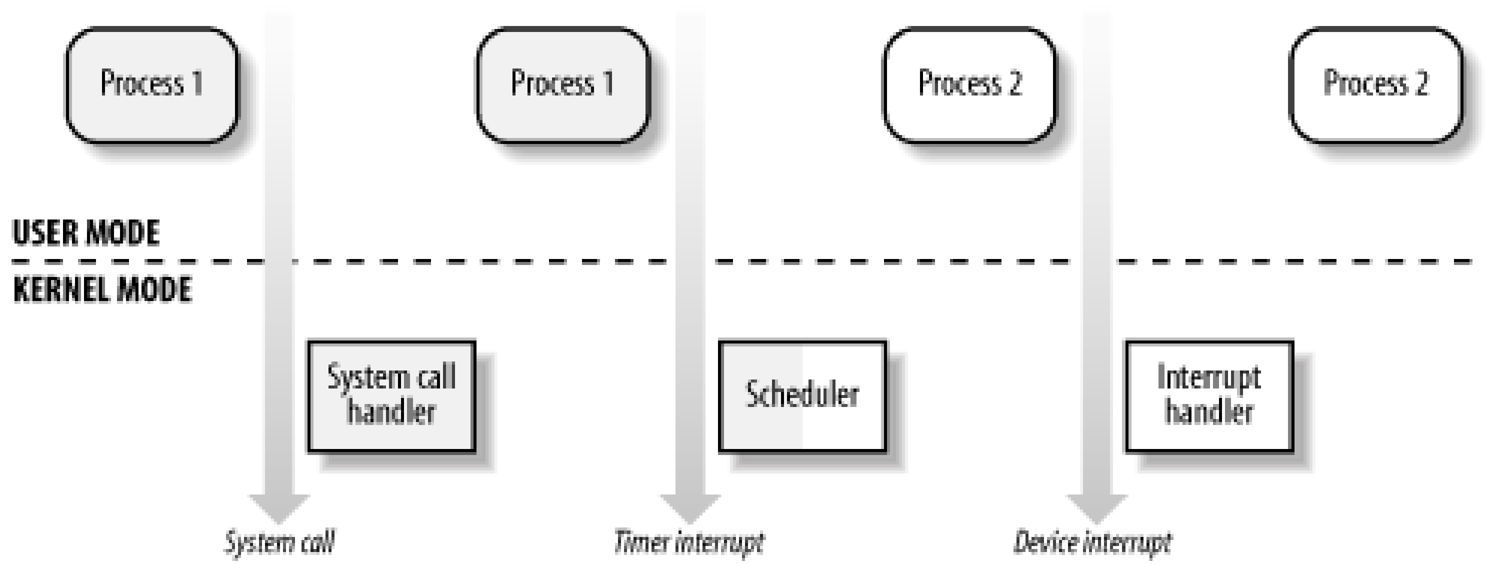
- Loading the code and initializing the kernel
- Detecting the Devices and configuring them
- Creating spontaneous system processes
- Operator intervention (manual boot only)
- Execution of system startup scripts
- Multiuser operation

Kernel

- Core component of operating system
- Unix kernels provide an execution environment in which applications may run
- Interfaces between the three major computer hardware components
  - Application/user interface
  - The CPU
  - Memory
  - I/O devices.
- Kernel also sets up memory address space for applications, loads files with application code into memory

Transitions Between User and Kernel Mode

- Process 1 in User Mode issues a system call, after which the process switches to Kernel Mode and the system call is serviced.
- Process 1 then resumes execution in User Mode until a timer interrupt occurs and the scheduler is activated in Kernel Mode
- A process switch takes place and Process 2 starts its execution in User Mode until a hardware device raises an interrupt.
- As a consequence of the interrupt, Process 2 switches to Kernel Mode and services the interrupt.



Homework

Case Study 1: Study and document the Unix bootstrap process.

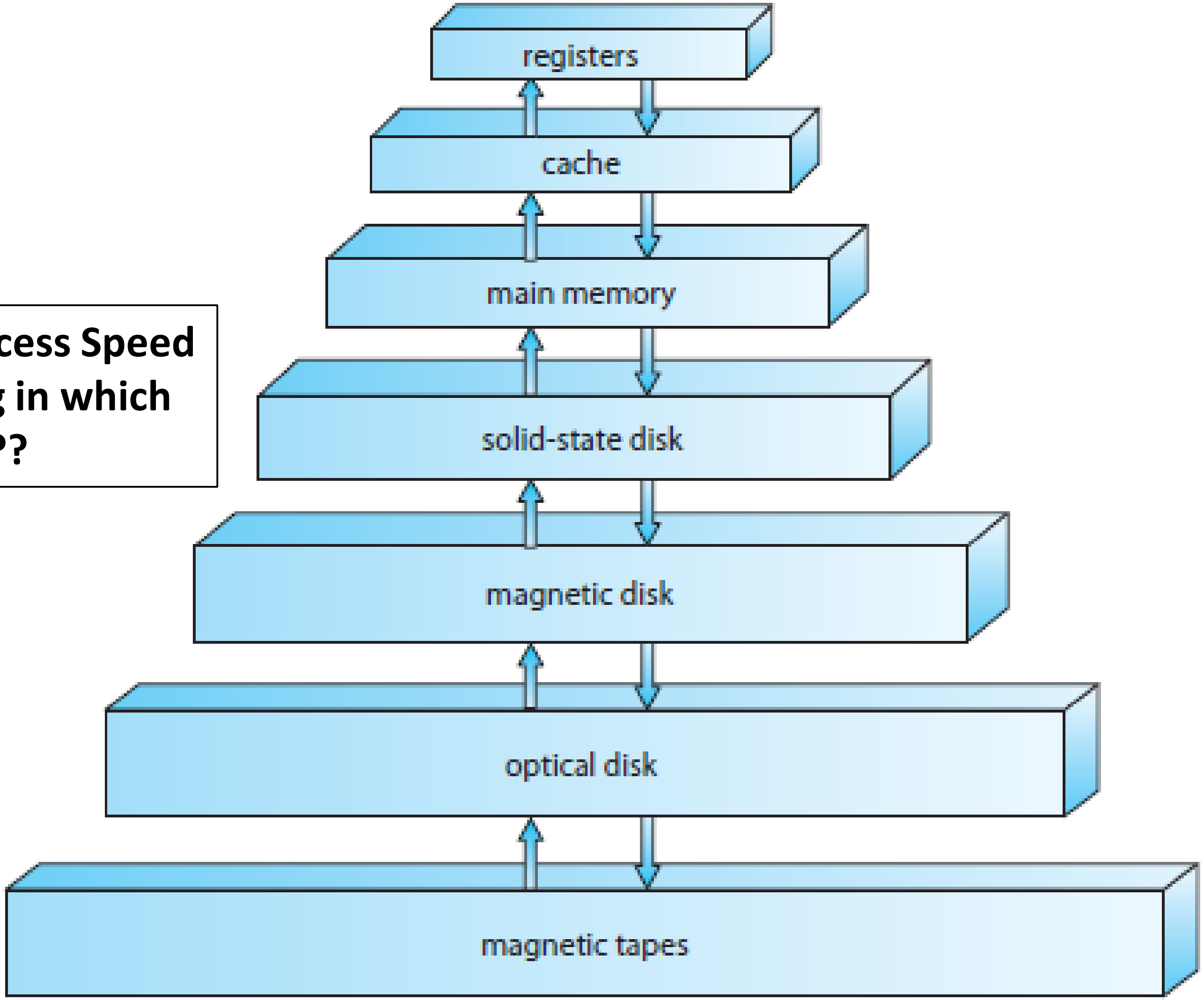
Submission Date: 18/08/2023

- Interrupts
  - Occurrence of an event is usually signalled by an **interrupt** from either the hardware or the software.
  - When the CPU is interrupted, it stops what it is doing and immediately transfers execution to a fixed location.
  - Fixed location usually contains the starting address where the service routine for the interrupt is located.
  - On completion of service routine the CPU resumes the interrupted computation.

Storage Structure

- CPU can load instructions only from memory, so any programs to run must be stored there.
- Mostly computer access there programs from Random Access Memory (RAM)
- Static programs such as Bootstrap are stored in ROM.
- All forms of memory provide an array of bytes, each byte having its own address.
- Sequence of load or store instructions maintains the execution.
- Load instruction moves a byte or word from main memory to an internal register within the CPU.
- Store instruction moves the content of a register to main memory

Memory Access Speed is increasing in which direction????



Storage-device hierarchy

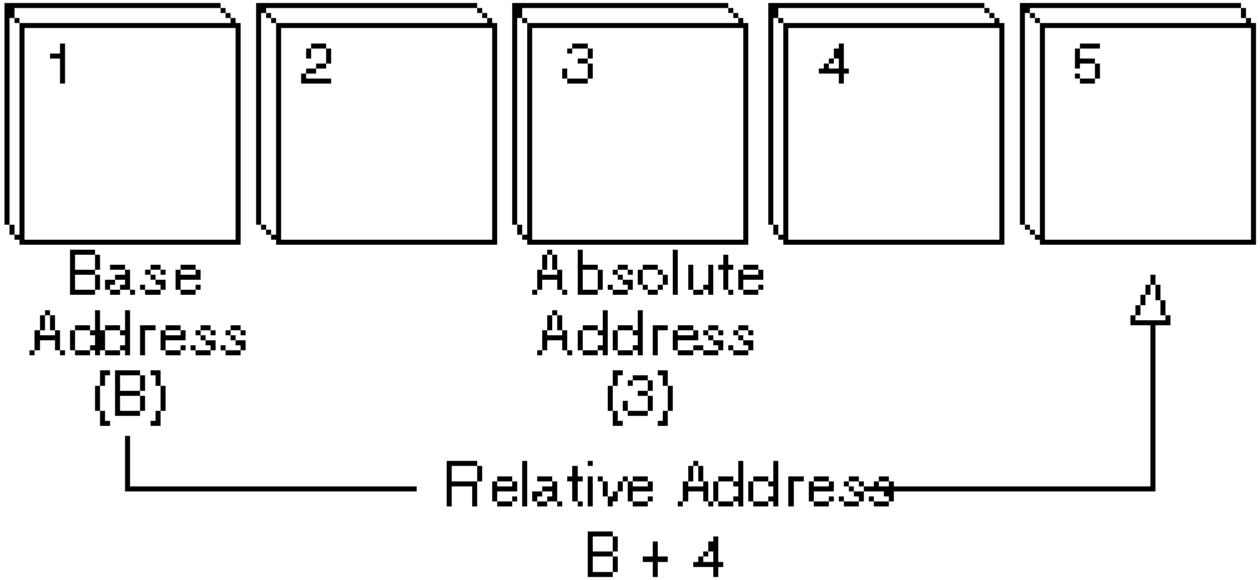
I/O Structure

- A large portion of operating system code is dedicated to managing I/O.
- Device controller who is in charge of a specific type of device, maintains some local buffer storage and a set of special-purpose registers.
- To start an I/O operation, the device driver loads the appropriate registers within the device controller.
- Device driver then informs the operating system that the data is loaded and is ready.

Operating System Operations

- Process Management
  - A program in execution, as mentioned, is a process.
  - A word-processing program being run by an individual user on a PC is a process.
  - A process needs certain resources—CPU time, memory, files, and I/O devices.
  - Activities in connection with process management:
    - Scheduling processes and threads on the CPUs
    - Creating and deleting both user and system processes
    - Suspending and resuming processes
    - Providing mechanisms for process synchronization

- Memory Management
  - Main memory is central to the operation of a modern computer system
  - Main memory is a repository of quickly accessible data shared by the CPU and I/O devices.
  - For a program to be executed, it must be mapped to absolute addresses and loaded into memory.
  - Program executes by accessing program instructions from memory by their absolute addresses.
  - Program terminates and its memory space is declared available
  - Activities in connection with memory management:
    - Keeping track of which parts of memory are currently being used and who is using them
    - Deciding which processes (or parts of processes) and data to move into and out of memory.
    - Allocating and deallocating memory space as needed.



- Absolute Address:
  - A fixed address in memory
  - Is not a relative address
  - Also called real addresses

- Storage Management
  - Operating system provides a uniform, logical view of information storage, a logical unit, the **file**
  - OS maps files onto physical media and accesses these files via the storage devices.
  - Activities in connection with storage management:
    - File System Management
    - Mass storage management
    - Caching
    - I/O Systems
  - File system management:
    - Creating and deleting files
    - Creating and deleting directories to organize files
    - Supporting primitives for manipulating files and directories
    - Mapping files onto secondary storage
    - Backing up files on stable (nonvolatile) storage media

- Storage Management
  - Mass-Storage Management
    - Free-space management
    - Storage allocation
    - Disk scheduling
  - Caching
  - I/O Systems
    - A memory-management component that includes buffering, caching, and spooling
    - A general device-driver interface
    - Drivers for specific hardware devices

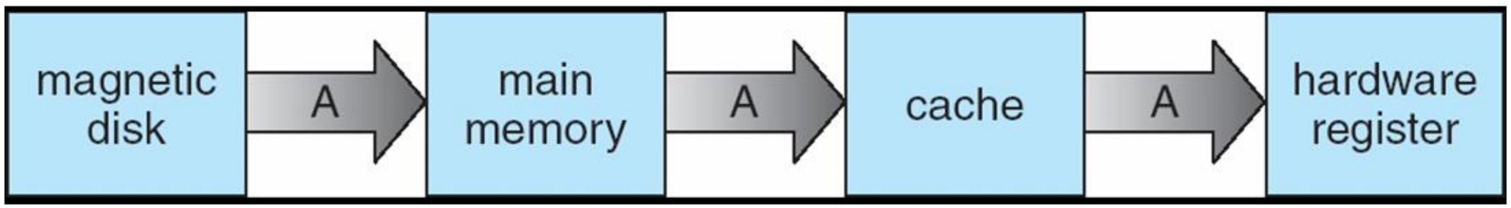
Performance of Various Levels of Storage

- Movement between levels of storage hierarchy can be explicit or implicit

Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000.000
Bandwidth (MB/sec)	20,000 – 100,000	5000 – 10,000	1000 – 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape

Migration of Integer 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

- Protection and Security
  - Protection, then, is any mechanism for controlling the access of processes or users to the resources defined by a computer system.
  - Job of **security** is to defend a system from external and internal attacks which may include viruses and worms, denial-of service attacks, identity theft and theft of service.

Modes of Operating System

- Operating System is interface between applications and the underlying hardware.
- At the same time to maintain systems integrity, it needs to prevent system integrity from application accessing hardware directly.
- Mode Bit is added to the hardware to indicate current mode.
- To enforce this protection, CPU provides two modes of operation:
  - Kernel Mode
  - User Mode

- User Mode
  - Direct access to the hardware is prohibited, and so is any arbitrary switching to kernel mode.
  - For a user-mode application, Windows creates a process for the application.
  - Process provides the application with a private virtual address space and a private handle table.
  - As virtual address space is private, one application cannot alter data that belongs to another application.
  - If an application crashes, the crash is limited to that one application

- Kernel Mode
  - Known as supervisor mode or privileged mode.
  - Has complete access to all of the computer's hardware and can control the switching between the CPU modes.
  - Code that runs in kernel mode shares a single virtual address space
  - If a kernel-mode driver accidentally writes to the wrong virtual address, data that belongs to the operating system or another driver could be compromised
  - If a kernel-mode driver crashes, the entire operating system crashes

Operating system classification

Operating Systems can be classified as:

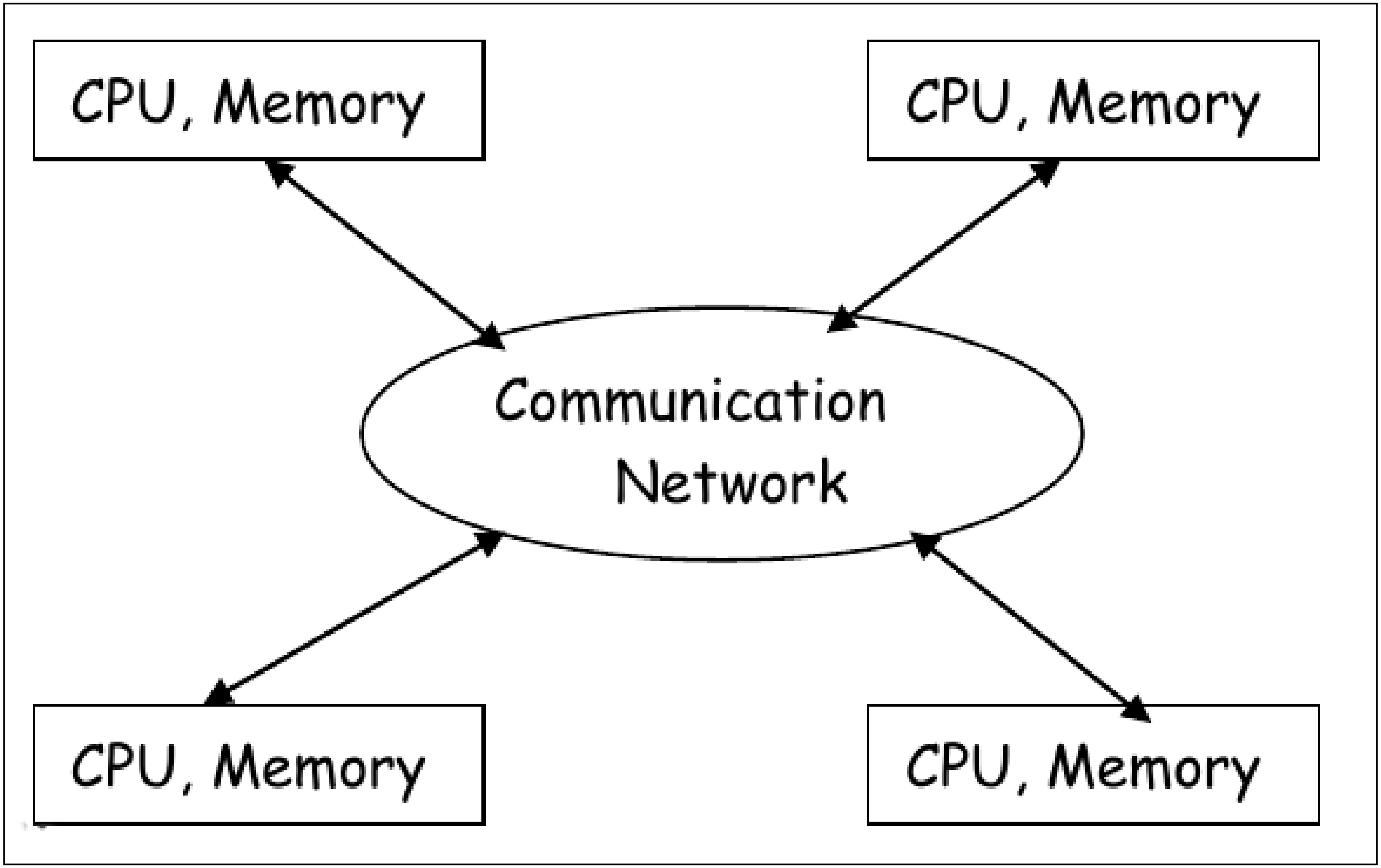
- *GUI:* Graphical User Interface operating systems are operating systems that have the capability of using a mouse and are graphical
- *Multi user:* allows multiple users to utilize the computer and run programs at the same time
- *Multi processing*: allows multiple processors to be utilized
- *Multi tasking*: allows multiple software processes to be run at the same time
- *Multi threading:* allows different parts of a software program to run concurrently

Distributed Operating Systems

- Distributed systems are loosely coupled systems
- A Distributed computer system is a collection of autonomous computer systems
- Distributed systems communicate with one another through various communication lines like high speed buses or telephone lines
- The processors in a distributed system may vary in size and function
- Example: small microprocessors, workstations , minicomputer and large general purpose computers

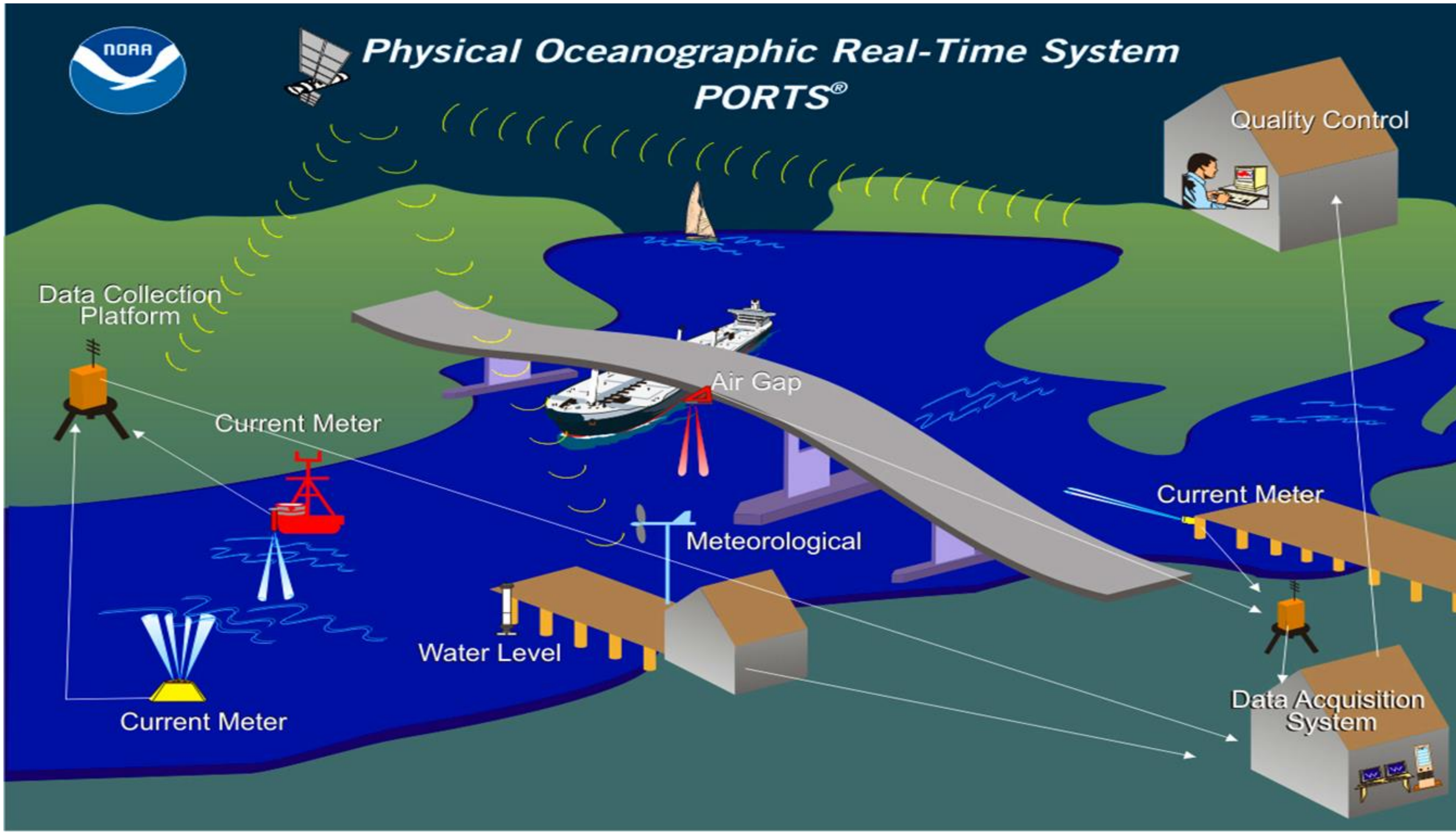
Distributed Operating Systems

- Processors in distributed systems are referred by no. of different names like, sites, nodes, computers, etc.
- Important reasons for building distributed systems are
  - Resource sharing
  - Computation speedup
  - Reliability
  - Communication
- The key objective of a distributed operating system is transparency
- Ideally, component and resource distribution should be hidden from users and applications programs unless they explicitly demand


CPU, Memory
CPU, Memory
Communication
Network
CPU, Memory
CPU, Memory

Real Time Operating System

- Real-time systems has well defined, fixed time constraints
- Processing must be done with in the defined constraints, or the system will fail
- For example:
  - Weather Forecasting, Self driving Cars, Earthquake warning systems, wireless sensor networks


Physical Oceanographic Real-Time System
PORTS®
NOAA
Quality Control
Data Collection
Platform
Current Meter
Air Gap
Current Meter
Meteorological
Water Level
Current Meter
Data Acquisition
System

Real Time Operating System

- Real-time system is used as a control device in a dedicated application
- Sensors bring data to computers
- Computer analyze data and adjust controls to modify the sensor inputs
- Example:
  - Scientific experiments, medical imaging systems, industrial control systems etc..
- Real-time system functions correctly only if it returns the correct result within its time constraints

Real Time Operating System

- A primary objective of real-time systems is to provide quick event - response times, thus meet the scheduling dead lines
- User convenience and resource utilization are of secondary concern to real- time system designers
- Real-time operating systems usually rely on some specific policies and techniques for doing their job

System Calls

- The mechanism used by an application program to request service from the operating system.
- System calls often use a special machine code instruction which causes the processor to change mode (Protected or Supervisor mode)
- This allows the OS to perform restricted actions such as accessing hardware devices or the memory management unit

Types of System Call

Process management

Call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

Types of System Call

Directory and file system management

Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

Miscellaneous

Call	Description
s = chdir(dirname)	Change the working directory
s = chmod(name, mode)	Change a file's protection bits
s = kill(pid, signal)	Send a signal to a process
seconds = time(&seconds)	Get the elapsed time since Jan. 1, 1970

System Calls Vs API Call

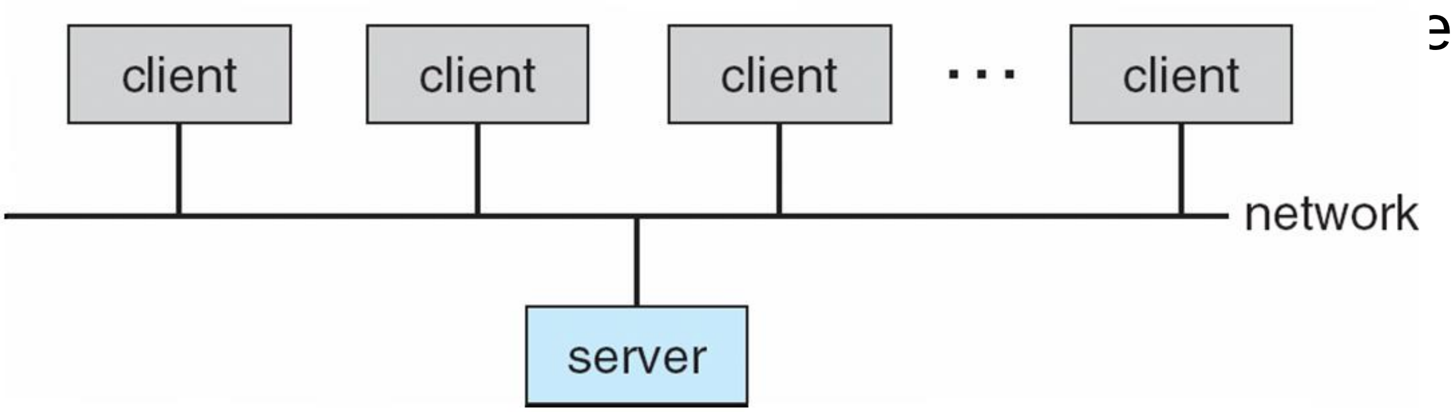
- Processes in a system runs in different modes, process running in user mode have no access to the privileged instructions.
- If process want perform any privileged instruction or need of any services they request kernel for that service through **System Calls**.
- API is generic term used to identify the functions exposed by any libraries.
- These functions are implemented as part of libraries, or SDK.
- System call is when you call the kernel, whereas a System API are used to invoke system call.

Computing Environments

- Traditional computer
  - Office environment
    - PCs connected to a network, terminals attached to mainframe or minicomputers providing batch and timesharing
    - Now portals allowing networked and remote systems access to same resources
  - Home networks
    - Used to be single system, then modems
    - Now firewalled, networked

Computing Environments

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



Peer to Peer Computing

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

Web Based Computing

- Web has become ubiquitous
- PCs most prevalent devices
- More devices becoming networked to allow web access
- New category of devices to manage web traffic among similar servers: **load balancers**
- Use of operating systems like Windows 95, client-side, have evolved into Linux and Windows XP, which can be clients and servers

Thank You