



Operating System



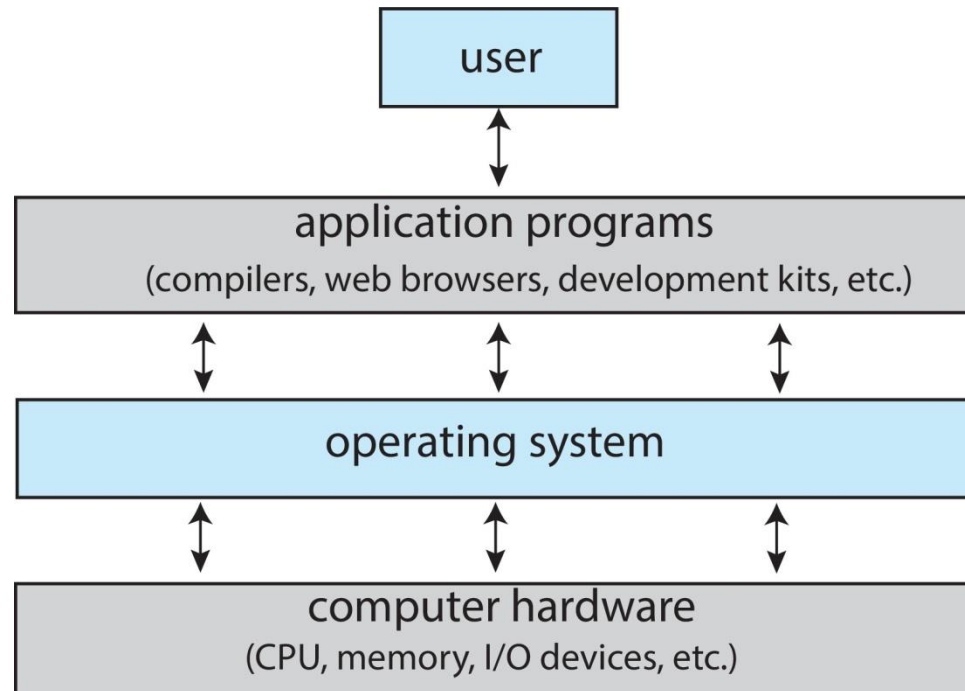
Introduction to OS

by Subhash Chand Agrawal

What is an Operating System?

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

Abstract View of Components of Computer



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

What Operating Systems Do

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

Goals of OS



- Operating Systems are designed to provide a user-friendly and convenient environment.
- Operating systems have a secondary goal of efficiency as well.

Function of OS


□ **Memory Management**

- Keeps tracks of primary memory, i.e., what part of it are in use by whom, what part are not in use.
- In multiprogramming, the OS decides which process will get memory when and how much.
- Allocates the memory when a process requests it to do so.
- De-allocates the memory when a process no longer needs it or has been terminated.

Functions of OS

□ **Process Management**

- OS decides the order in which processes have access to the processor, and how much processing time each process has. This function of OS is called Process Scheduling.
- Allocates the processor (CPU) to a process.
- De-allocates processor when a process is no longer required.

- 
- ❑ **Device Management:** An OS manages device communication via its respective drivers.
 - Keeps tracks of all devices. Program responsible for this task is known as the **I/O controller**.
 - Decides which process gets the device when and for how much time.
 - Allocates the device in the efficient way.
 - De-allocates devices.



□ File Management

- Keeps track of information, location, uses, status etc. The collective facilities are often known as **file system**.
- Decides who gets the resources.
- Allocates the resources.
- De-allocates the resources.

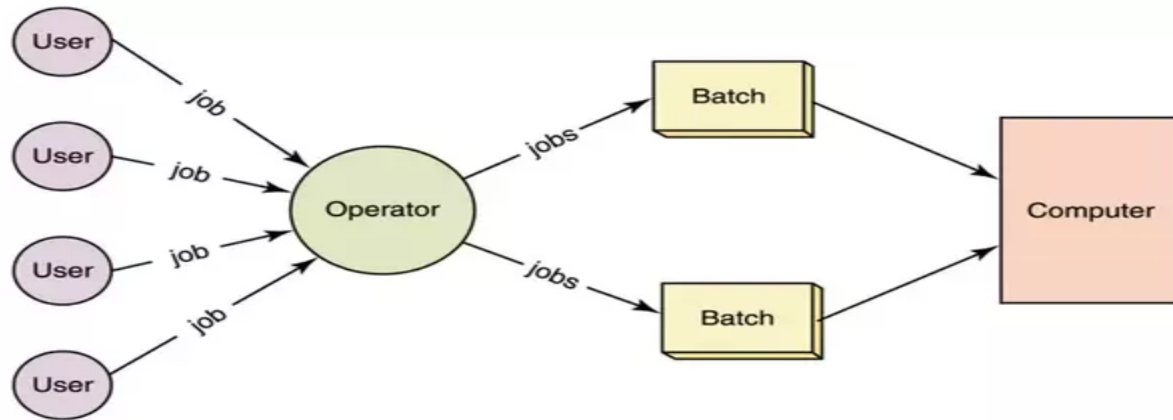
Security

- **Security** – Prevents unauthorized access to programs and data by means of passwords and other similar techniques.
- Protection against unauthorized access through login.
- Protection against intrusion by keeping firewall active.
- Protecting the system memory against malicious access.
- Displaying messages related to system vulnerabilities.

Bootstrapping:

- **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

Simple Batch Systems



- Batch processing was very popular in the **1970s**.
- People used to have a single computer known as a **mainframe**.
- Users using batch operating systems do not interact directly with the computer.
- Each user prepares their job using an offline device like a punch card and submitting it to the computer operator.
- Each job includes program, data, and control instructions.

- Jobs with similar requirements are grouped and executed as a group to speed up processing.
- Once the programmers have left their programs with the operator, they sort the programs with similar needs into batches.
- **Example:** Bank Statements, Payroll system, etc.

Control Cards

□ Problems

1. How does the monitor know about the nature of the job (e.g., Fortran versus Assembly) or which program to execute?
2. How does the monitor distinguish
 - (a) job from job?
 - (b) data from program?

□ Solution

- ▣ Introduce control cards

Resident monitor

- Automatic job sequencing – automatically transfers control from one job to another.
- Special cards called Control Cards that tell the resident monitor which programs to run

\$JOB

\$FTN

\$RUN

\$DATA

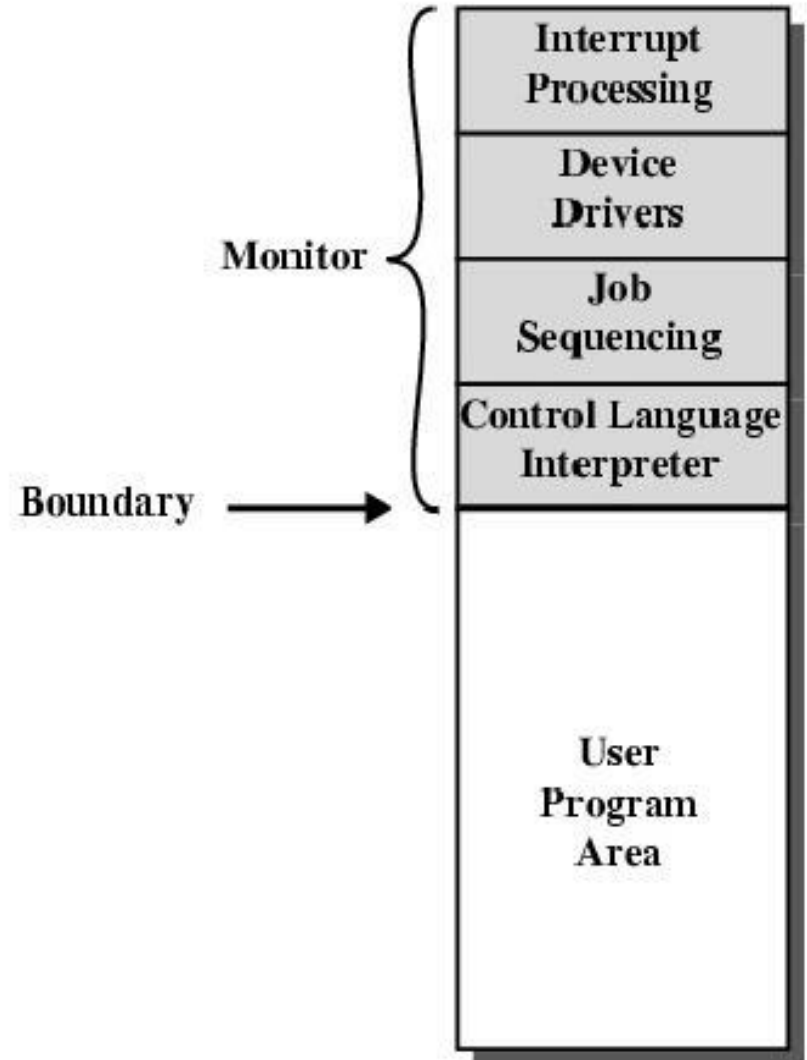
\$END

Resident Monitor

The resident monitor would automatically sequence from one program to another and from job to another.

Control card interpreter was responsible for reading and carrying out the instruction on the cards at the point of execution.

The delay between job submission and job completion is called **TURNAROUND** time, may result the amount of computing needed or from delays before the OS starts to process the job.



Control Cards (Cont.)

- Parts of resident monitor
 - ▣ Control card interpreter – responsible for reading and carrying out instructions on the cards.
 - ▣ Loader – loads systems programs and applications programs into memory.
 - ▣ Device drivers – know special characteristics and properties for each of the system's I/O devices.
- Problem: Slow Performance – I/O and CPU could not overlap ; card reader very slow.

Disadvantages

❑ Starvation

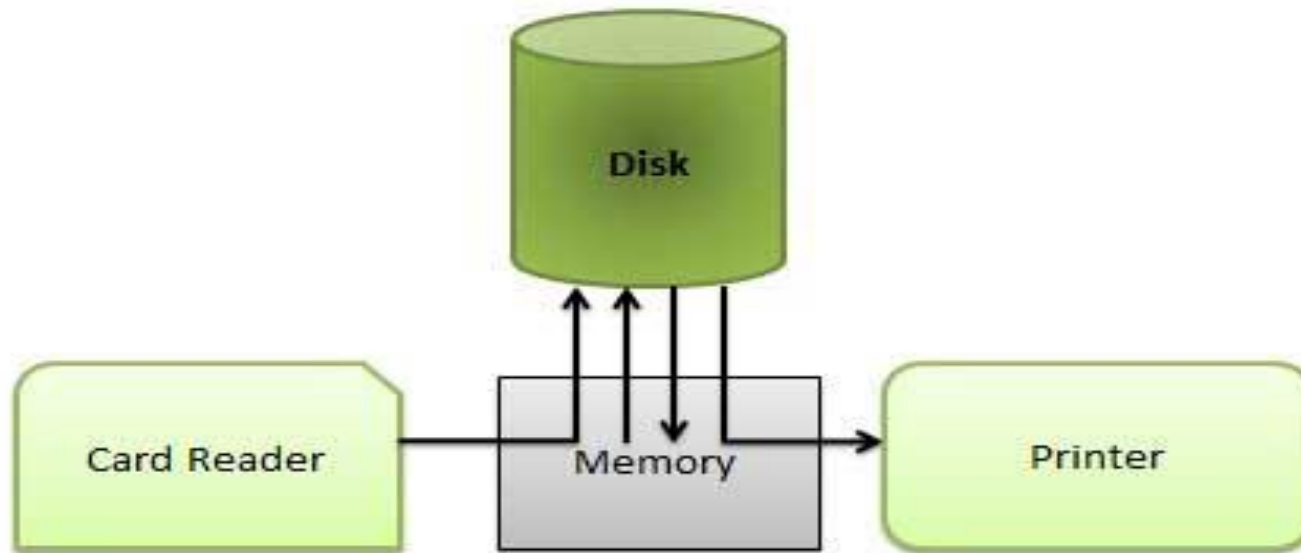
Batch processing suffers from starvation. If there are five jobs J1, J2, J3, J4, J4 and J5 present in the batch. If the execution time of J1 is very high then other four jobs will never be going to get executed or they will have to wait for a very high time. Hence the other processes get starved.

❑ Not Interactive

Batch Processing is not suitable for the jobs which are dependent on the user's input. If a job requires the input of two numbers from the console then it will never be going to get it in the batch processing scenario since the user is not present at the time of execution.

Spooling

- Simultaneous peripheral operations online
- Spooling refers to putting data of various I/O jobs in a buffer.
- This buffer is a special area in memory or hard disk which is accessible to I/O devices.
- The example of Spooling is ***printing***. The documents which are to be printed are stored in the SPOOL and then added to the queue for printing. During this time, many processes can perform their operations and use the CPU without waiting while the printer executes the printing process on the documents one-by-one.



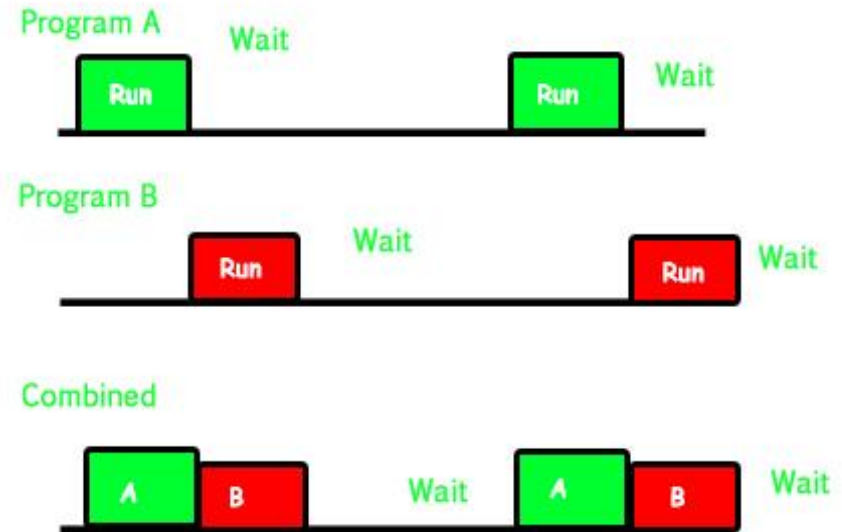
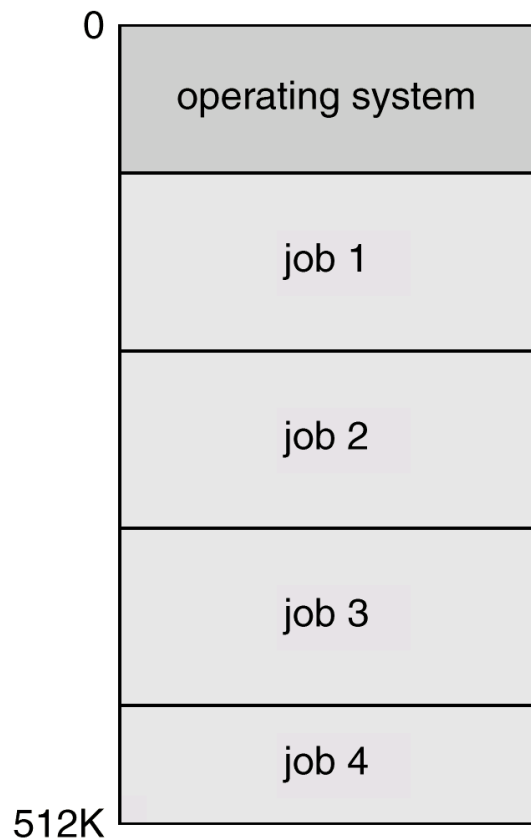
- Spooling overlaps the I/O of one job with the computation of other jobs.
- Spooling has a direct beneficial effect on the performance of the system
- Spooling can keep the CPU and the I/O devices working at same time at much higher rates.

Multiprogramming Batch Systems

- ❑ Multiprogramming is an extension to the batch processing where the CPU is kept always busy.
- ❑ Each process needs two types of system time: CPU time and IO time.
- ❑ In multiprogramming environment, for the time a process does its I/O, The CPU can start the execution of other processes.
- ❑ Therefore, multiprogramming improves the efficiency of the system.

Multiprogrammed Batch Systems

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



- In a multi-programmed system, as soon as one job goes for an I/O task, the Operating System interrupts that job, chooses another job from the job pool (waiting queue), gives CPU to this new job and starts its execution.
- In this way, no CPU time is wasted by the system waiting for the I/O task to be completed.
- Therefore, the ultimate goal of multi programming is to keep the CPU busy as long as there are processes ready to execute.
- This way, multiple programs can be executed on a single processor by executing a part of a program at one time, a part of another program after this, then a part of another program and so on, hence executing multiple programs. Hence, the CPU never remains idle.

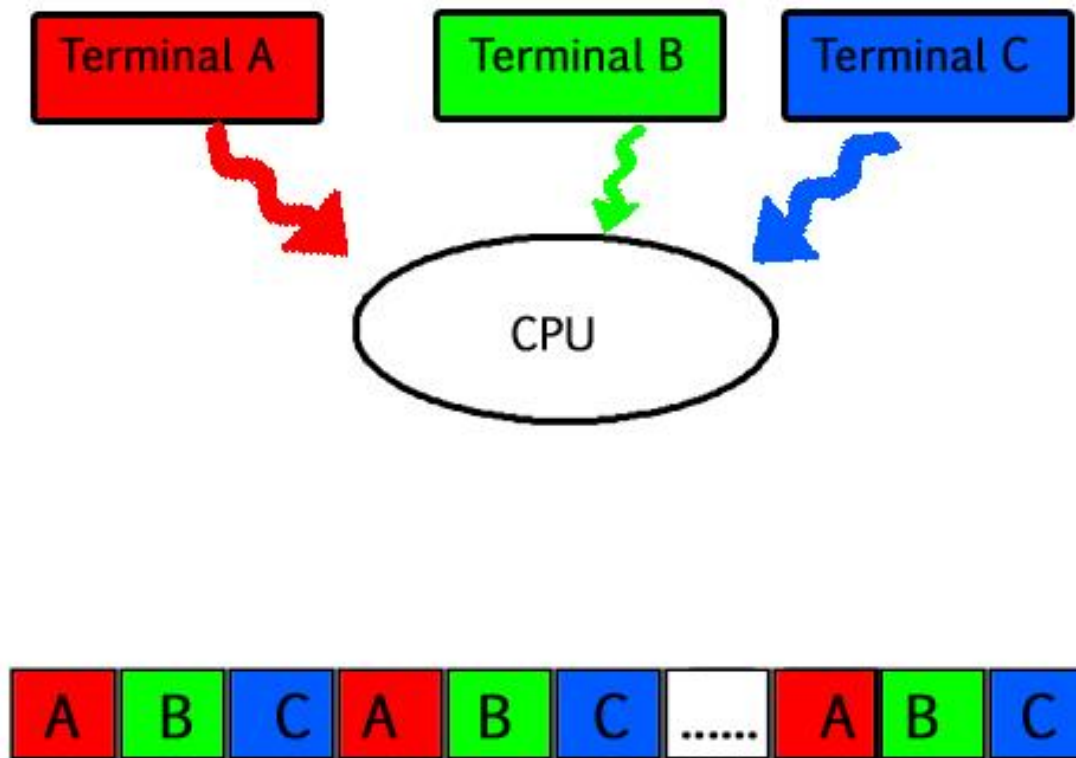
OS Features Needed for Multiprogramming

- ❑ Memory management – the system must allocate the memory to several jobs.
- ❑ CPU scheduling – the system must choose among several jobs ready to run.
- ❑ Allocation of devices.

Time-Sharing Systems—Interactive Computing

- Time sharing system is where each process is allocated a particular time span and the process has to finish its completion within that time span. If it is failed to complete its execution then CPU control goes to the immediate next process.
- A job is swapped in and out of memory to the disk.
- It is considered to be a **logical** extension of multiprogramming because both does simultaneous execution but differ in their prime objectives.
- The main objective of time-sharing systems is to **minimize response time** but not maximizing the processor use(which is the objective of multiprogramming systems).
- Eg: Multics, Unix, etc.

At any time the CPU is executing only one task while other tasks are waiting for their turn.



Parallel Systems

- Multiprocessor systems with more than one CPU in close communication.
- *Tightly coupled system* – processors share memory, bus, a clock, peripheral devices and; communication usually takes place through the shared memory.
- Advantages of parallel system:
 - ▣ Increased *throughput*
 - ▣ Economical
 - ▣ Increased reliability
 - graceful degradation
 - fail-soft systems

Parallel Systems (Cont.)

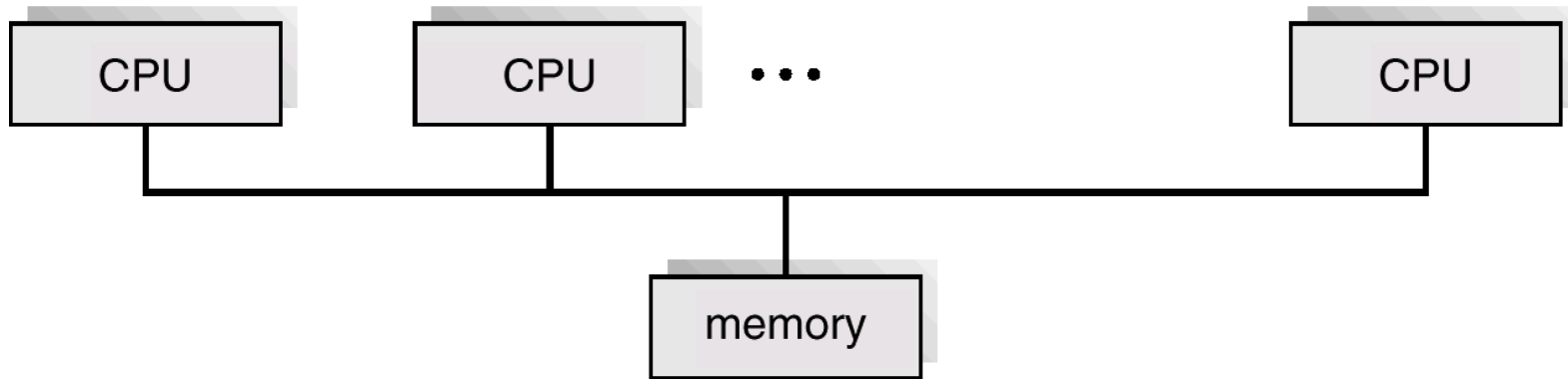
- *Asymmetric multiprocessing*

- ▣ Each processor is assigned a specific task; master processor schedules and allocates work to slave processors.
- ▣ More common in extremely large systems

- *Symmetric multiprocessing (SMP)*

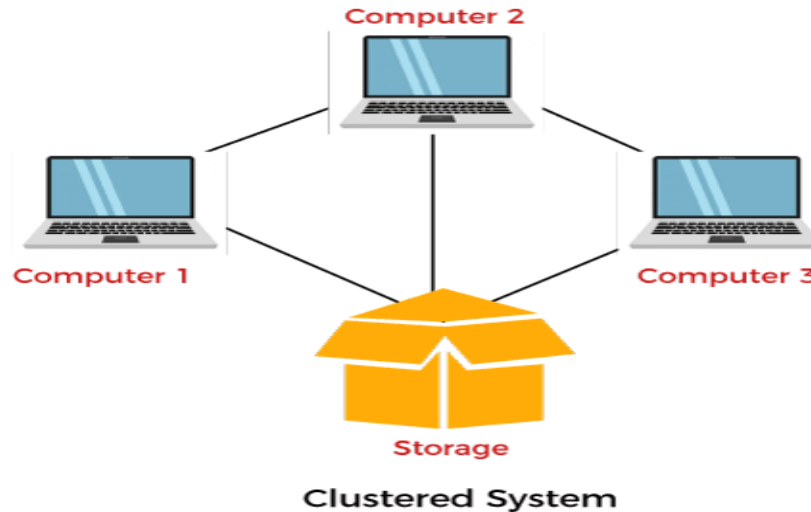
- ▣ Each processor runs an identical copy of the operating system.
- ▣ Many processes can run at once without performance deterioration.
- ▣ Most modern operating systems support SMP

Symmetric Multiprocessing Architecture




But with multiprocessing, each process can be assigned to a different processor for its execution. If its a dual-core processor (2 processors), two processes can be executed simultaneously and thus will be two times faster, similarly a quad core processor will be four times as fast as a single processor.

Clustered System



Cluster systems are similar to parallel systems because both systems use multiple CPUs.

The primary difference is that clustered systems are made up of two or more independent systems linked together and a shared storage media, and all systems work together to complete all tasks

- 
- Cluster components are generally linked via fast area networks such as LAN, and each node executing its instance of an operating system.
 - The primary purpose of using a cluster system is to assist with weather forecasting, scientific computing, and supercomputing systems.

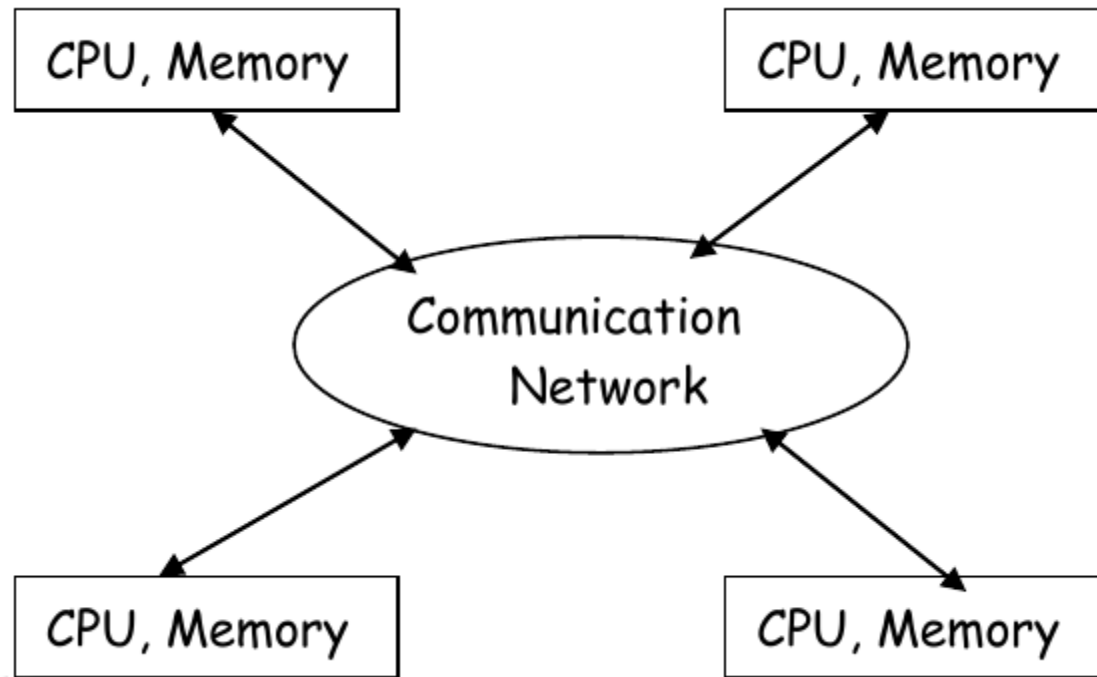
Real-Time Systems

- Often used as a control device in a dedicated application such as controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems.
- Well-defined fixed-time constraints.
- *Hard real-time system.*
 - ▣ Secondary storage limited or absent, data stored in short-term memory, or read-only memory (ROM)
 - ▣ Missile systems, air traffic control systems, Autopilot System In Plane, Pacemakers
- *Soft real-time system*
 - ▣ Limited utility in industrial control or robotics
 - ▣ Useful in applications (multimedia, virtual reality) requiring advanced operating-system features.

Distributed Systems

- Distribute the computation among several physical processors.
- *Loosely coupled system* – each processor has its own local memory; processors communicate with one another through various communications lines, such as high-speed buses or telephone lines.
- Advantages of distributed systems.
 - ▣ Resources Sharing
 - ▣ Computation speed up – load sharing
 - ▣ Reliability
 - ▣ Communications

Examples of Distributed Operating System are- LOCUS, etc.



What is Kernel in Operating System?

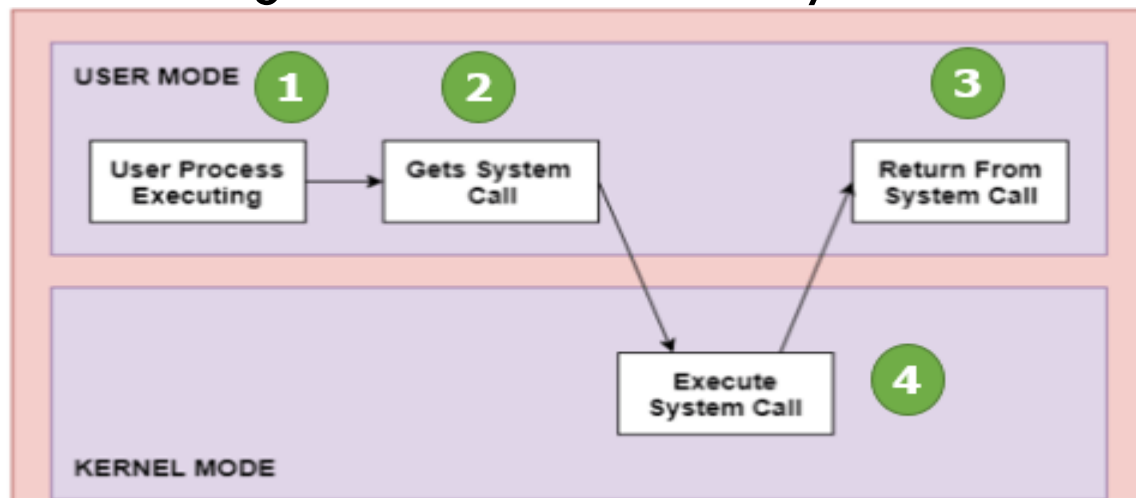
- **Kernel** is central component of an operating system that manages operations of computer and hardware. It basically manages operations of memory and CPU time.
- Kernel loads first into memory when an operating system is loaded and remains into memory until operating system is shut down again.
- It is responsible for various tasks such as disk management, task management, and memory management.

Operating-System Operations

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

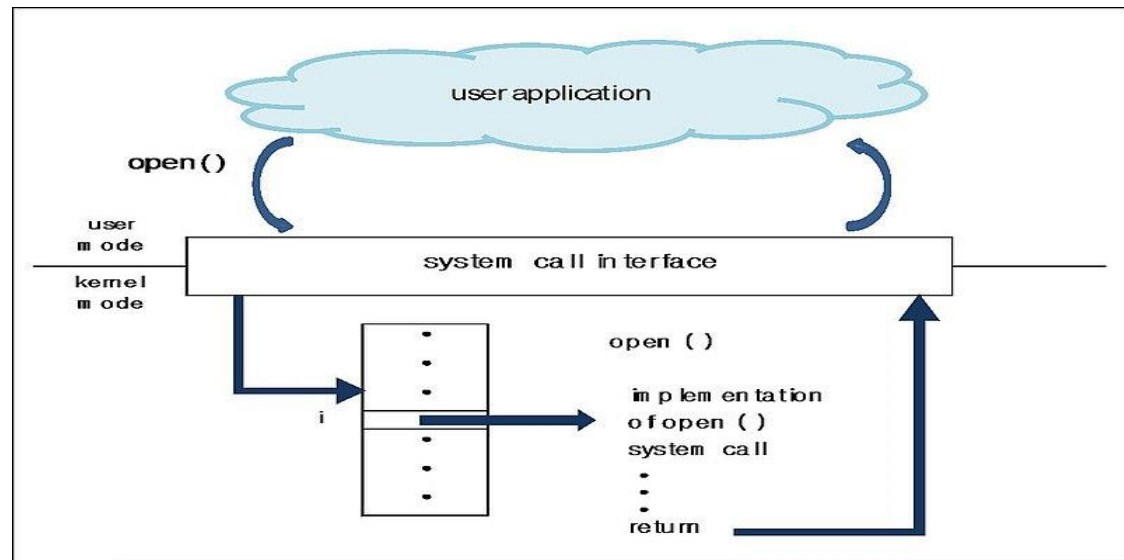
System Call

- A system call is a way for programs to **interact with the operating system**.
- A computer program makes a system call when it makes a request to the operating system's kernel.
- System call **provides** the services of the operating system to the user programs via Application Program Interface(API).
- All programs needing resources must use system calls.



□ Services Provided by System Calls :

1. Process creation and management
2. Main memory management
3. File Access, Directory and File system management
4. Device handling(I/O)
5. Protection
6. Networking, etc.



Types of System Calls :

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()



- Thank You