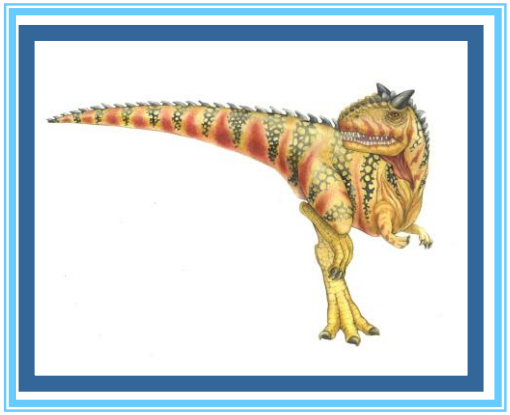
Day1: Introduction to Operating System

Kiran Waghmare

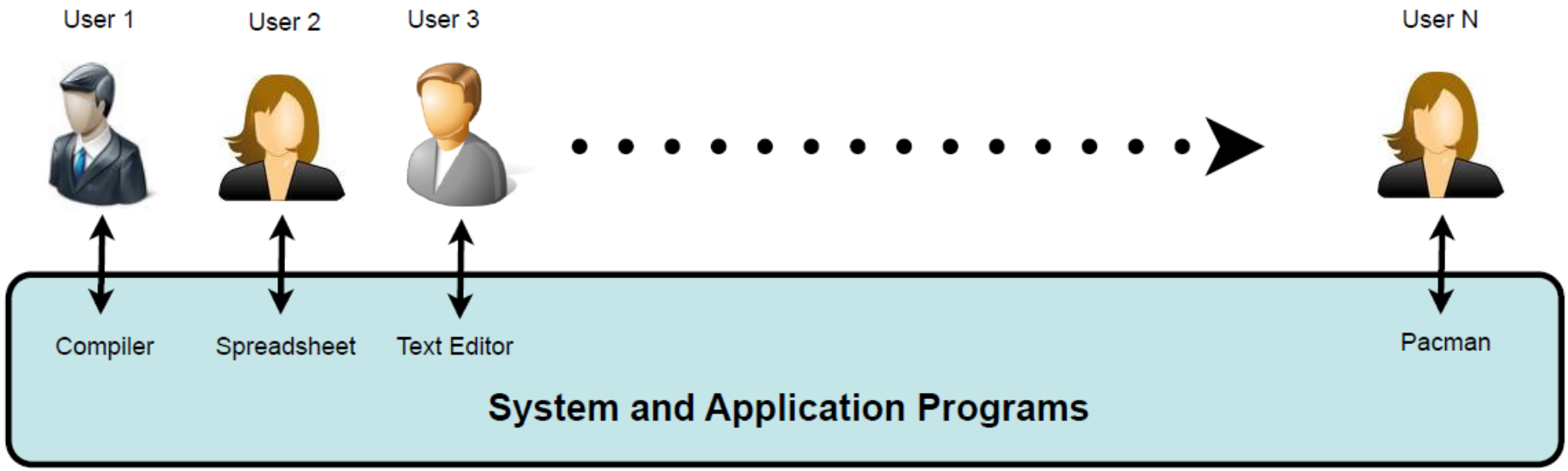


Agenda

- **Introduction to OS**
  - OS
  - Application Software
  - Hardware dependent
  - Components of OS
  - Difference between :
    - Mobile OS, Embedded system OS,
    - Real Time OS,
    - desktop OS server machine os
  - Functions of OS
  - User and Kernel space & model
  - Interrupts & system calls

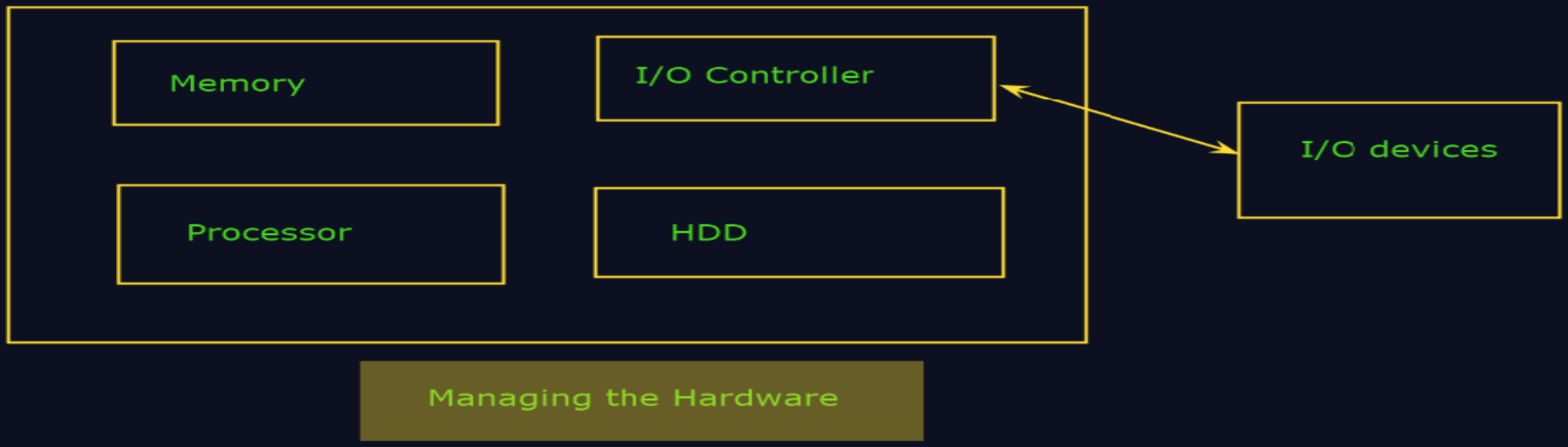
What is an Operating System?

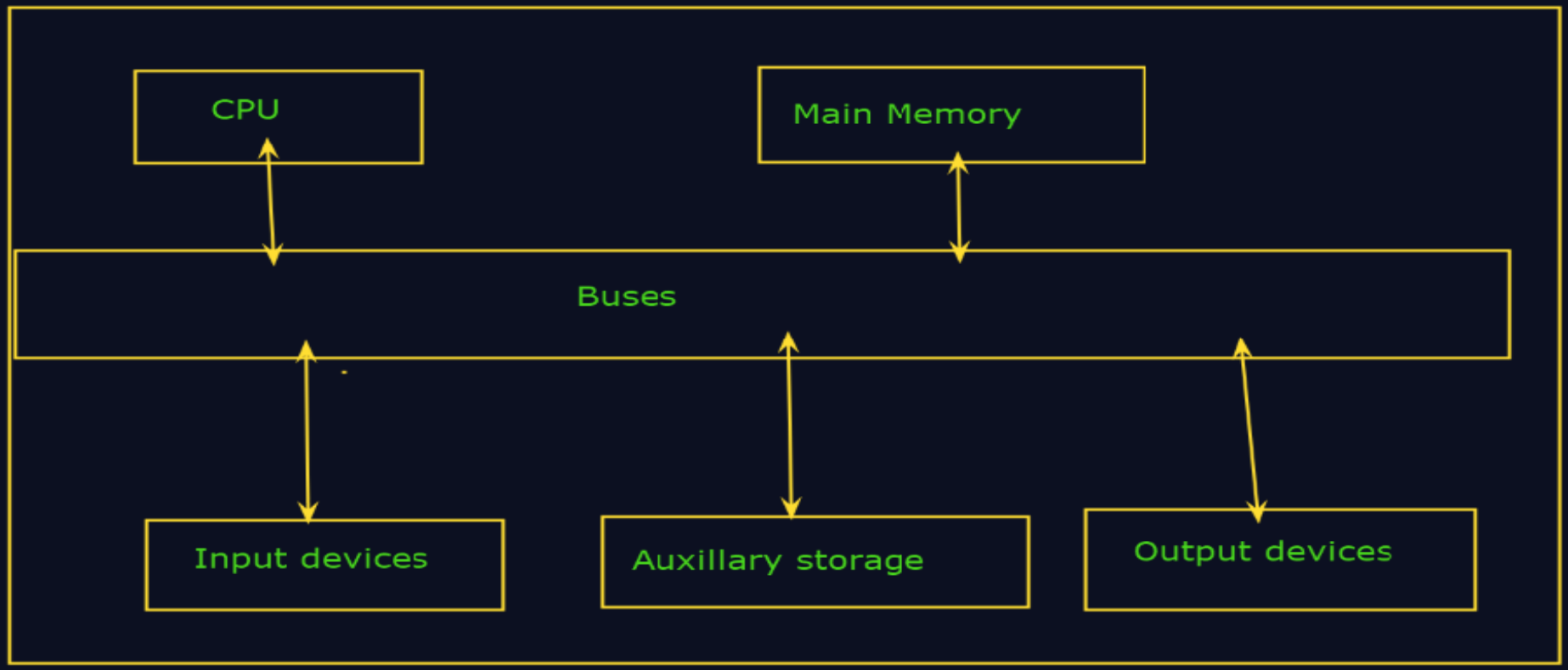
- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and **make solving user problems easier**
  - Make the **computer system convenient** to use
  - Use the computer hardware in an efficient manner



Operating System

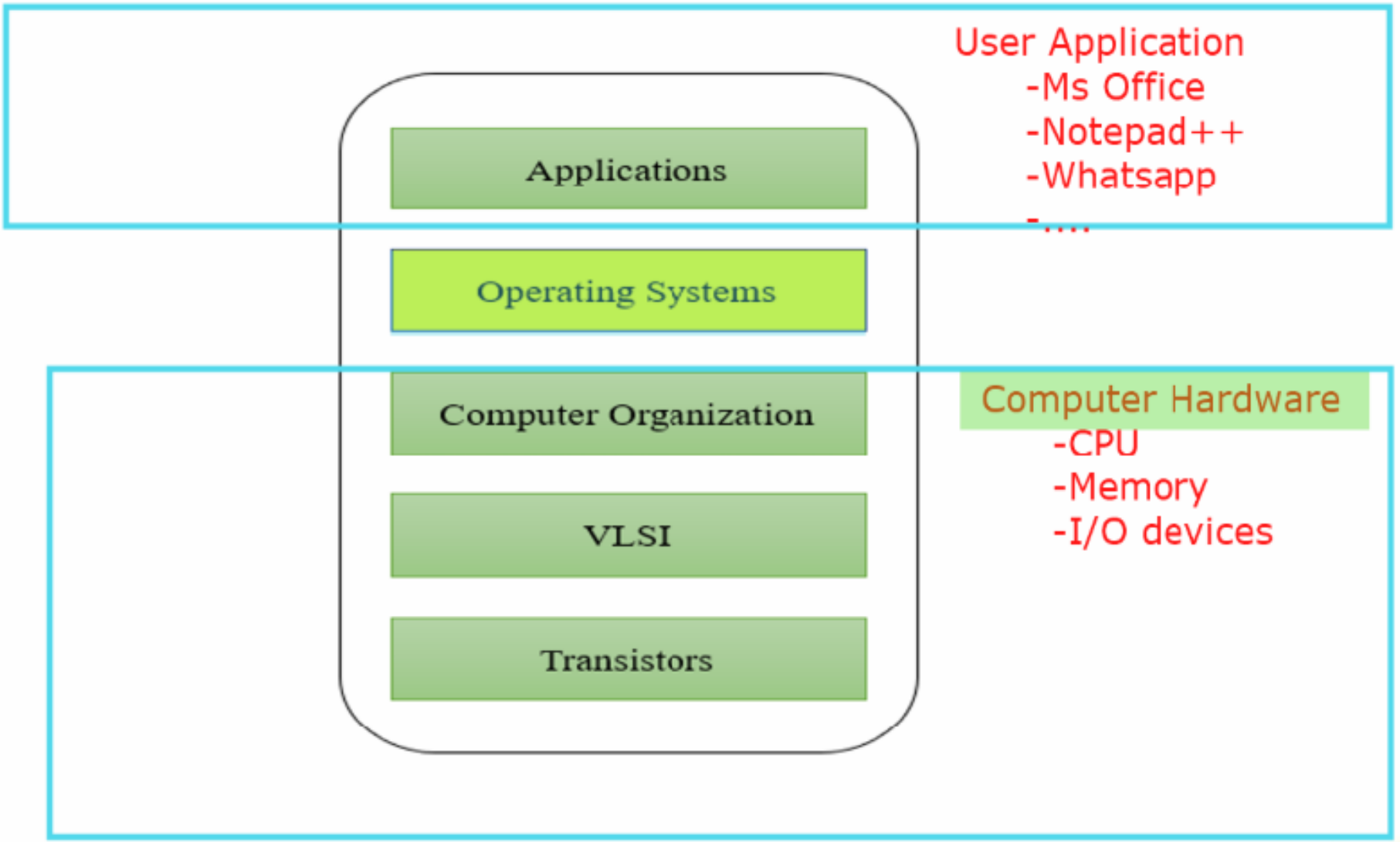
Controls the hardware and coordinates its use among the various application programs for the various users.





Input-Output System : Providing the interface

The Layers in Systems



Os Usage → 1. Hardware Abstraction
2. Resource Management

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

OS usage

- Hardware Abstraction

  turns hardware into something that applications can use

- Resource Management

  manage system's resources

Learning and understanding

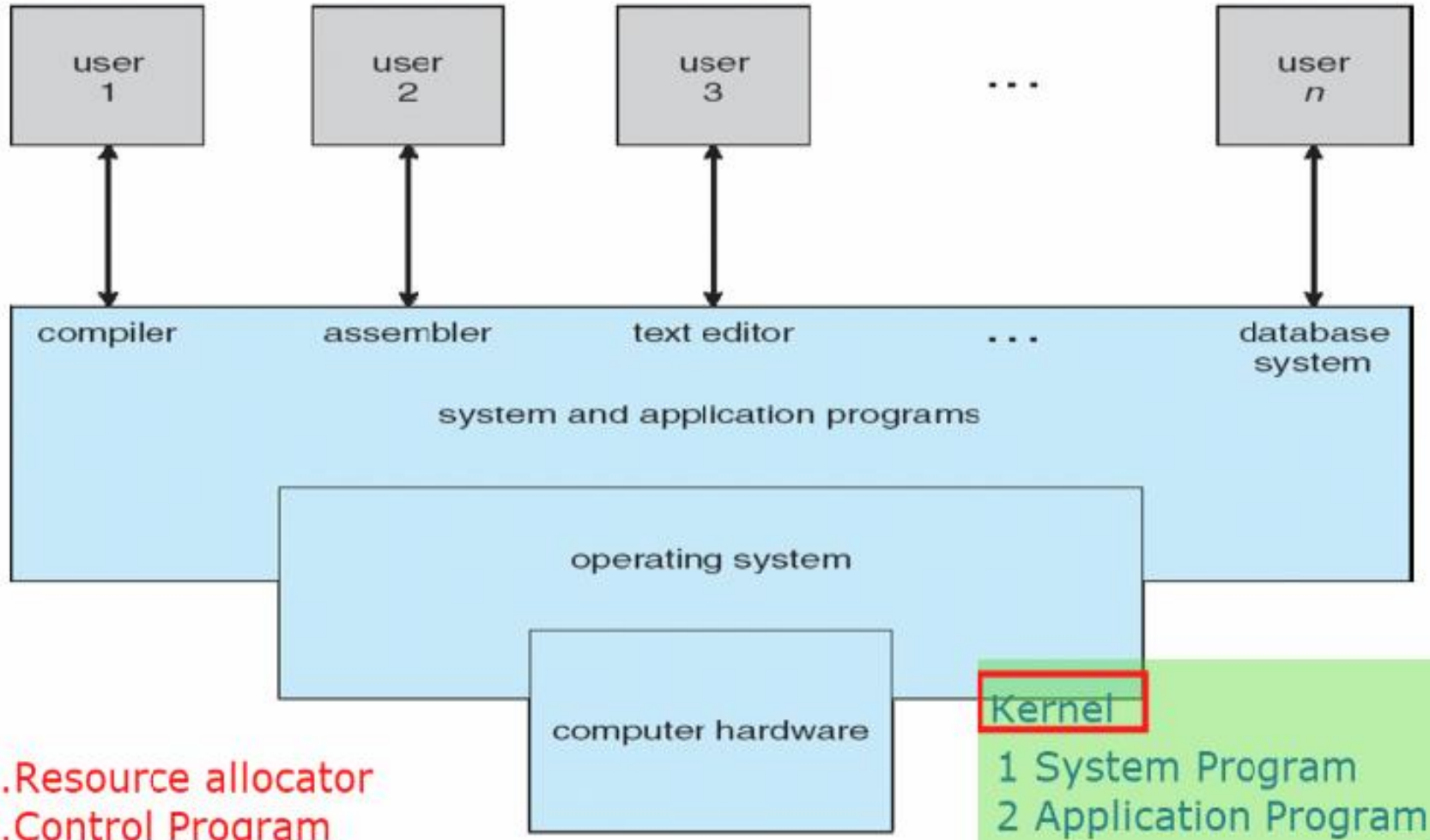
User view

Top down

System view

Bottom up

Four Components of a Computer System



1.Resource allocator
2.Control Program

Kernel
1 System Program
2 Application Program

Kernel : one program which runs all the time in computer

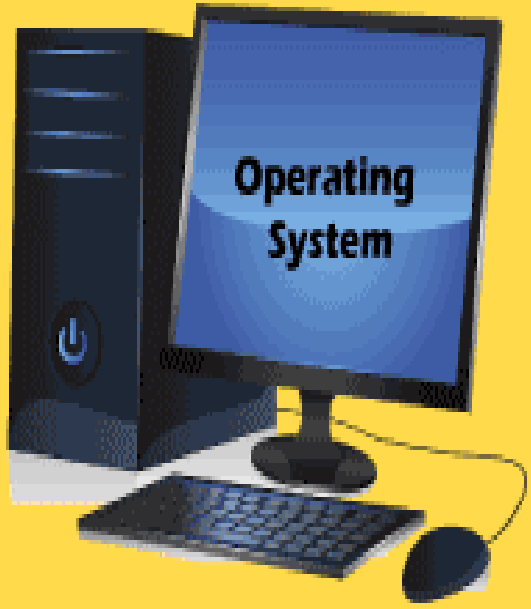
Operating System Definition

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

Difference Between System Software and Application Software

Application Software

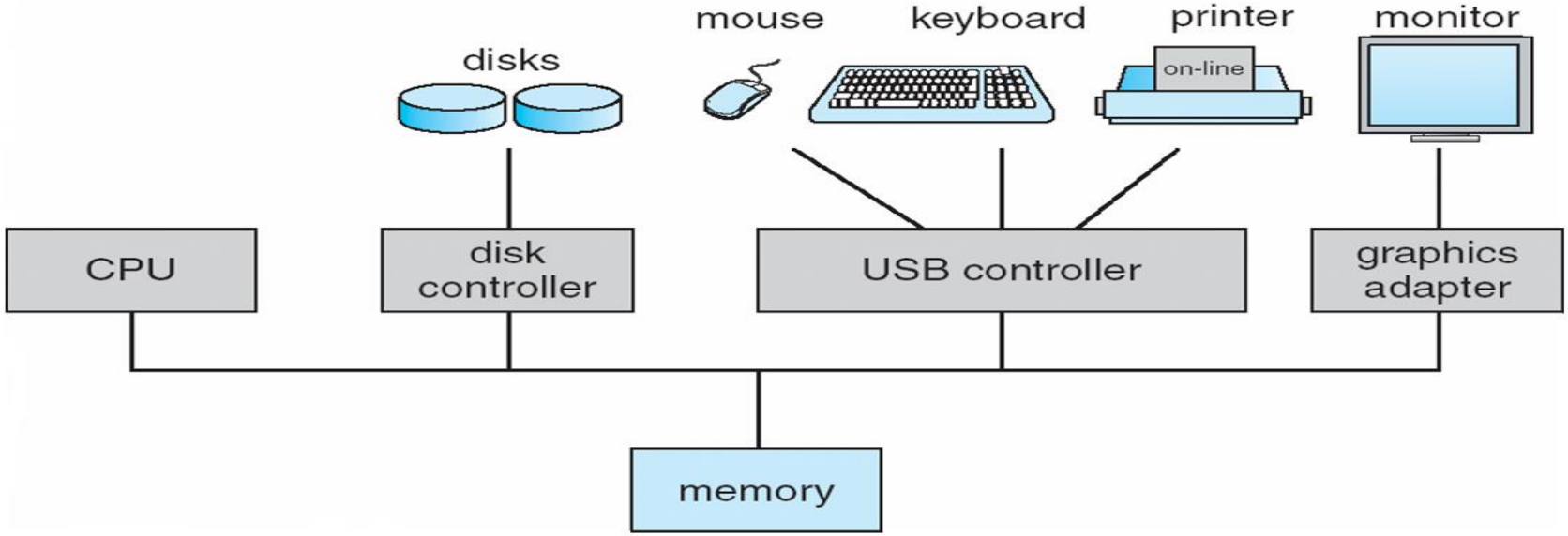
VS



System Software

Computer System Organization

- **Computer-system operation**
  - One or more CPUs, device controllers connect through common bus providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory cycles



Basics of OS:

Kernel:

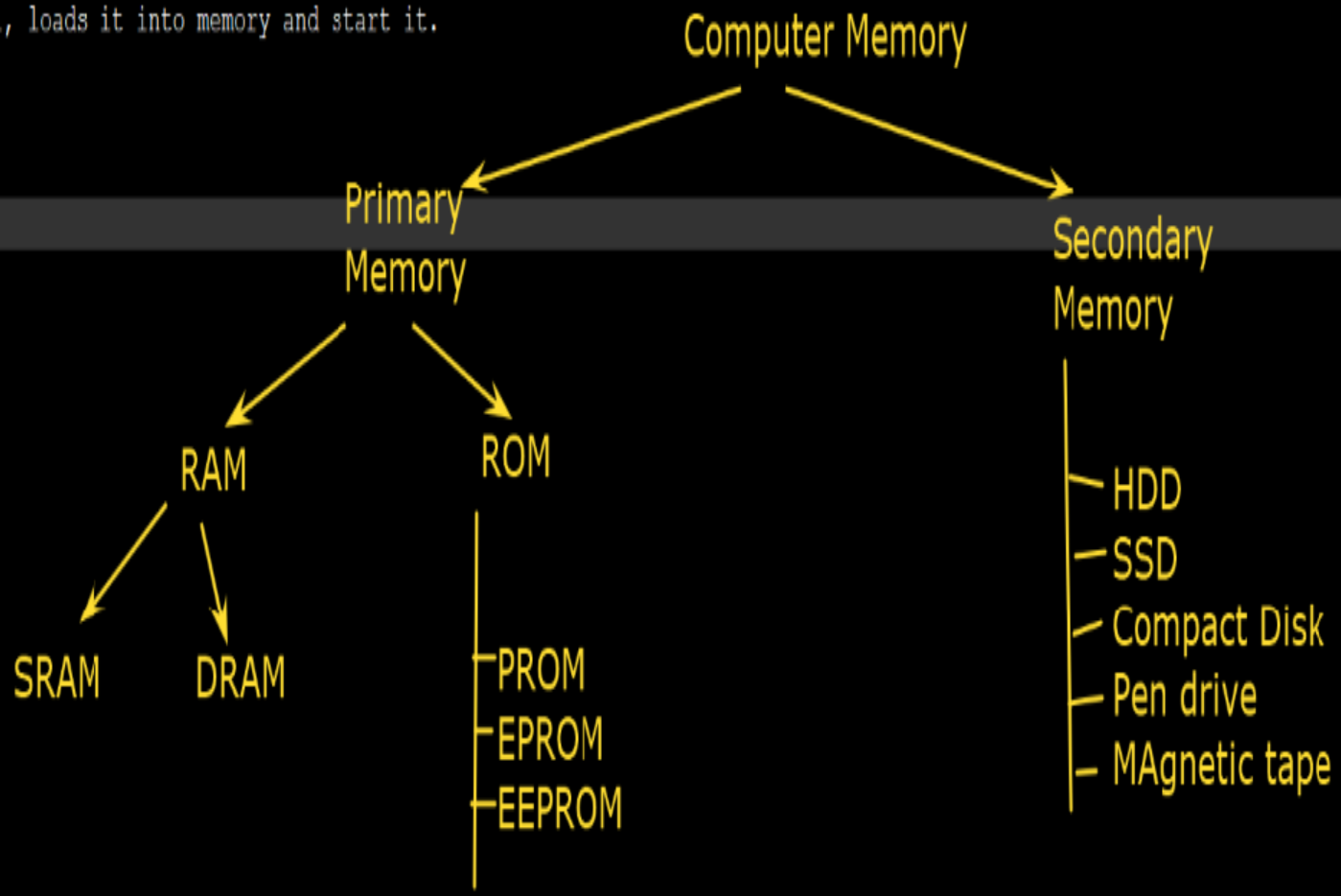
- The kernel is a computer program that is the core of the computer's operating system with complete control over everythig in the system.

System Boot:

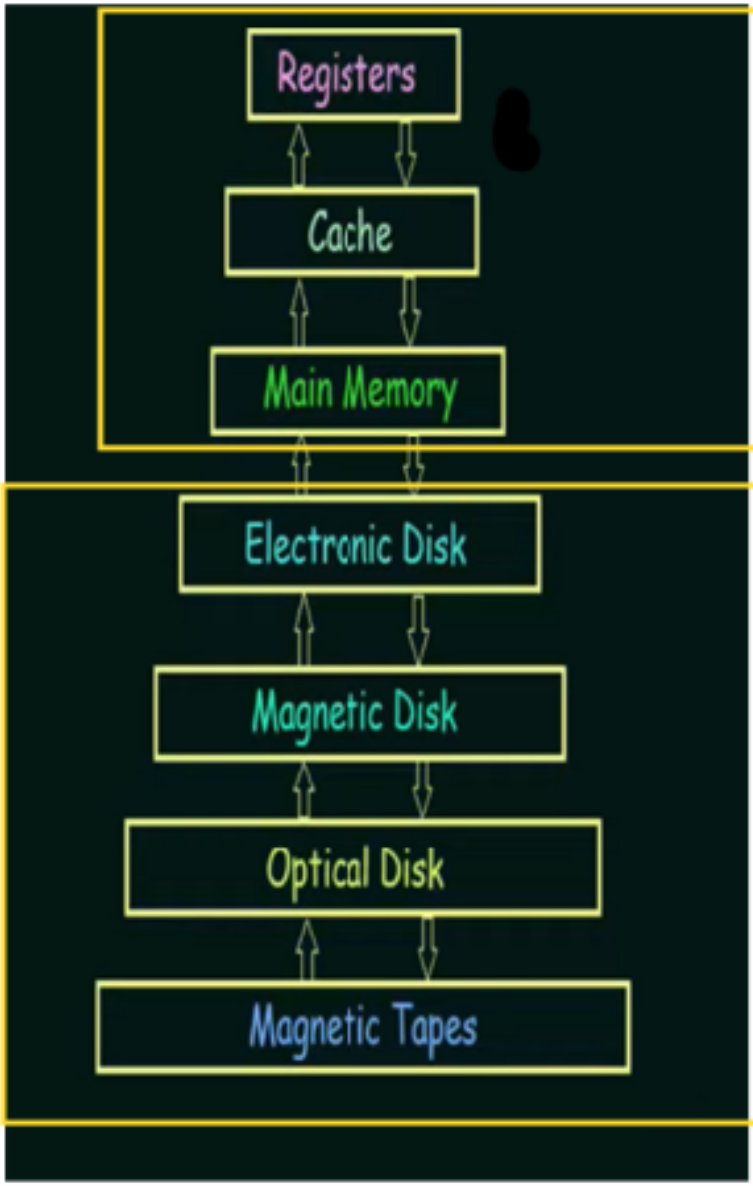
-Boostrap loader: locates the kernel, loads it into memory and start it.

-Booting system, Bootstrap program.

-EPROM or ROM memory



Storage Structure



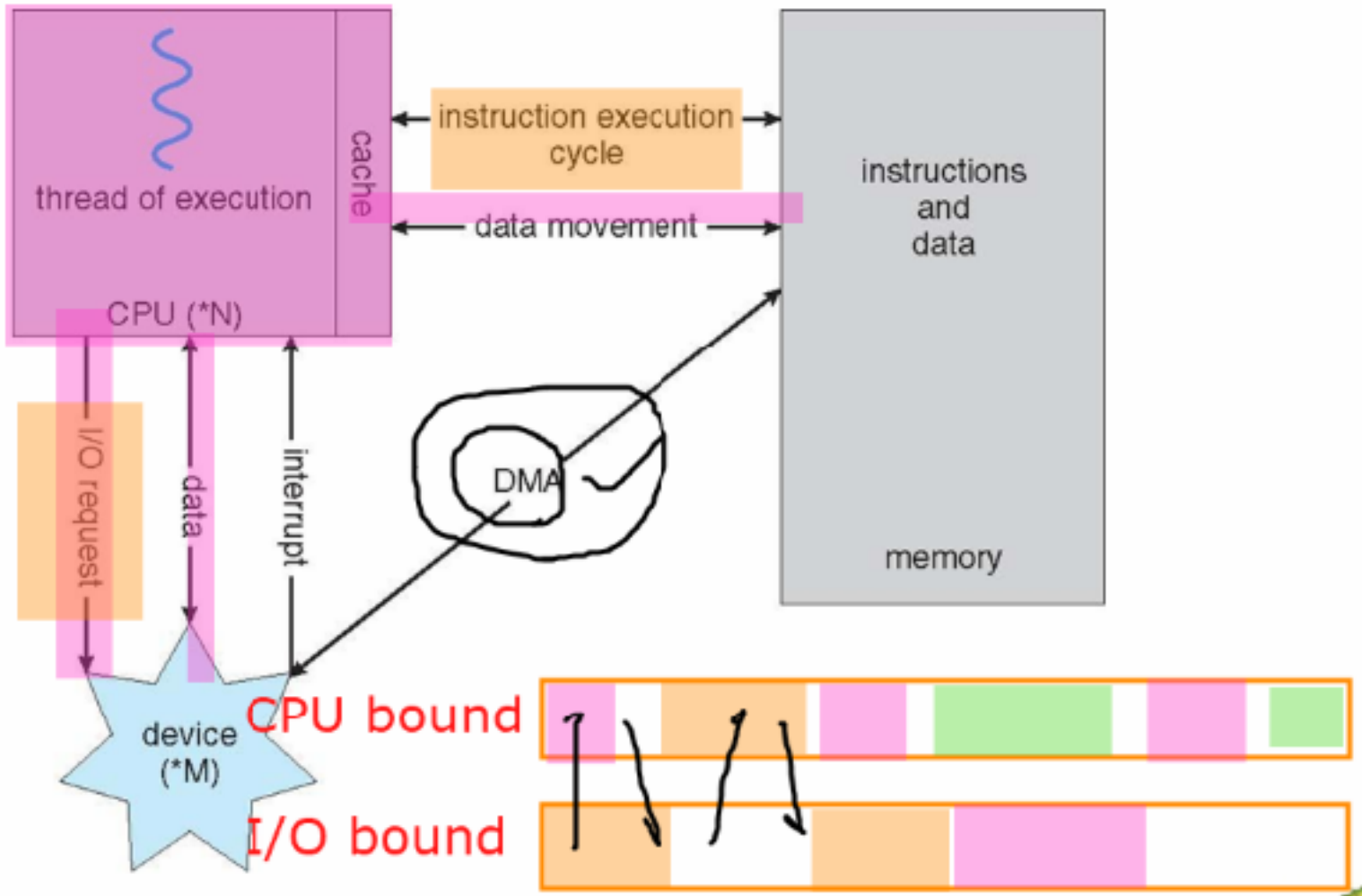
Volatile

Non-volatile

Expensive

smaller s

How a Modern Computer Works



CPU is busy : Efficiency is high

CPU is ideal : Efficiency is low

Direct Memory Access Structure

- Used for high-speed I/O devices able to transmit information at close to memory speeds
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention
- Only one interrupt is generated per block, rather than the one interrupt per byte

Computer-System Operation

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

A Simple Program

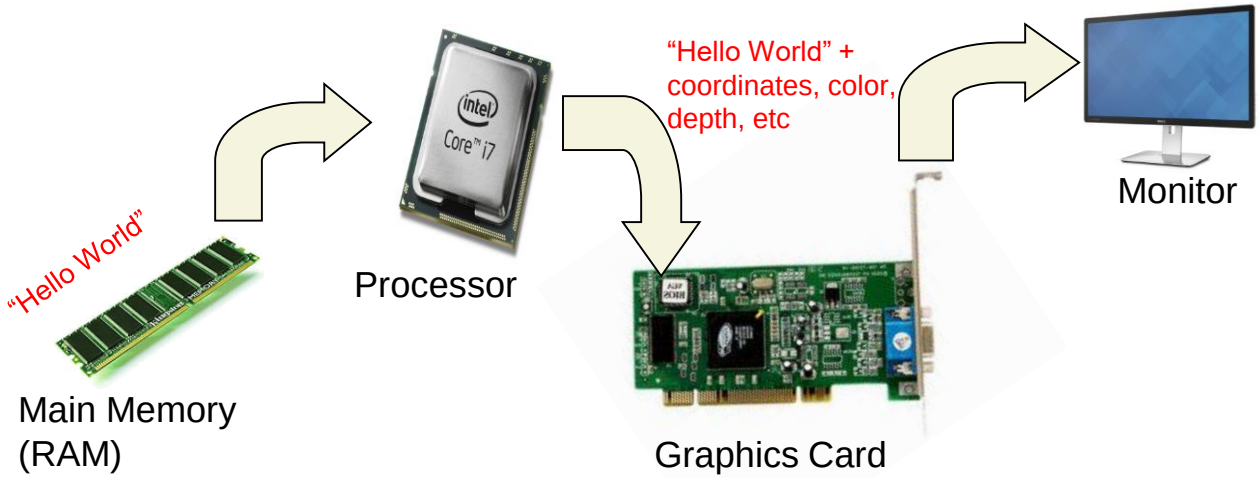
What is the output of the following program?

```c
#include <stdio.h>

int main(){
   char str[] = "Hello World\n";
   printf("%s", str);
}
```

How is the string displayed on the screen?

Displaying on the Screen



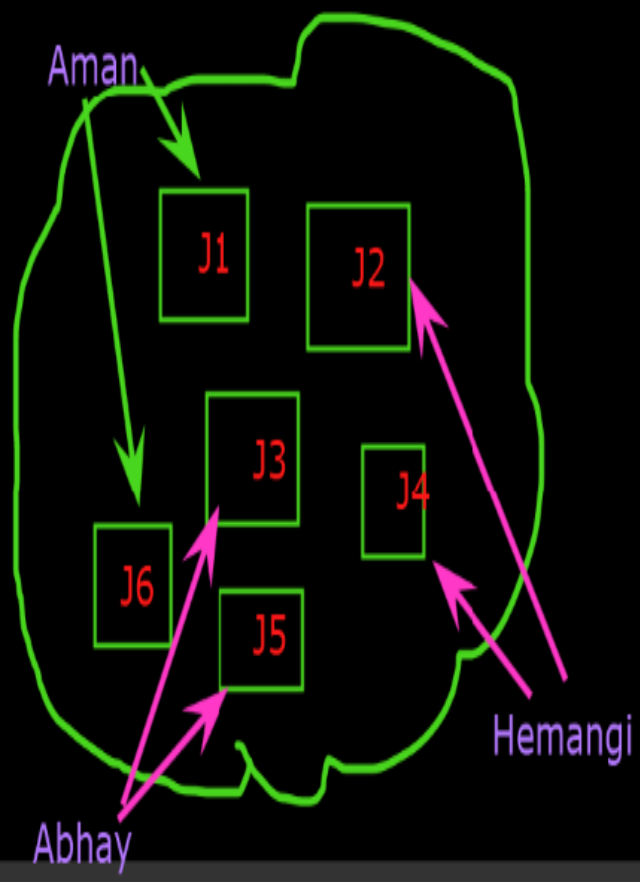
- Can be complex and tedious
- Hardware dependent

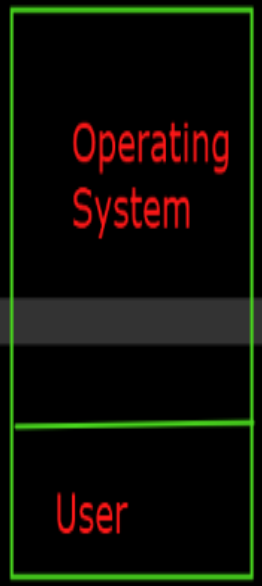
Without an OS, all programs need to take care of every nitty gritty detail

Types of Operating Systems

- Following are some of the most widely used types of Operating system.
  - Simple Batch System
  - Multiprogramming Batch System
  - Multiprocessor System
  - Desktop System
  - Distributed Operating System
  - Clustered System
  - Realtime Operating System
  - Handheld System

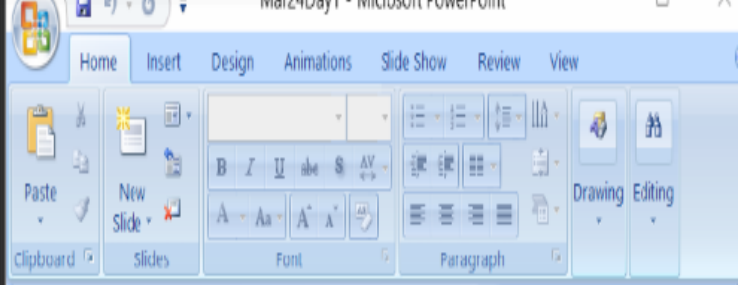
2. Multiprocessor System

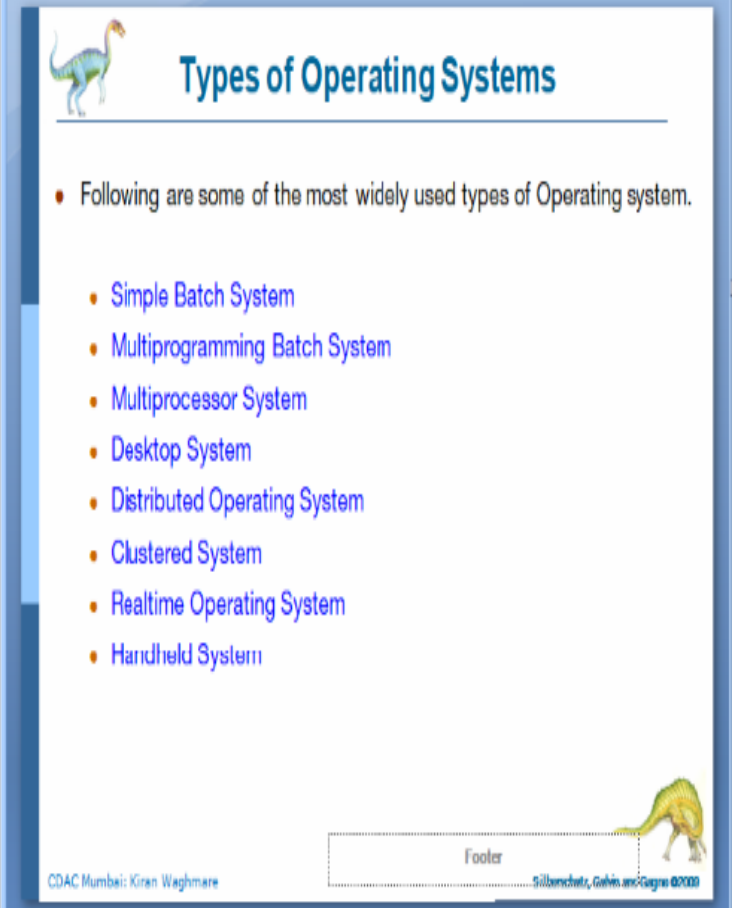




Main Memory (RAM)

3. Clustered System





Multiprocessor Systems

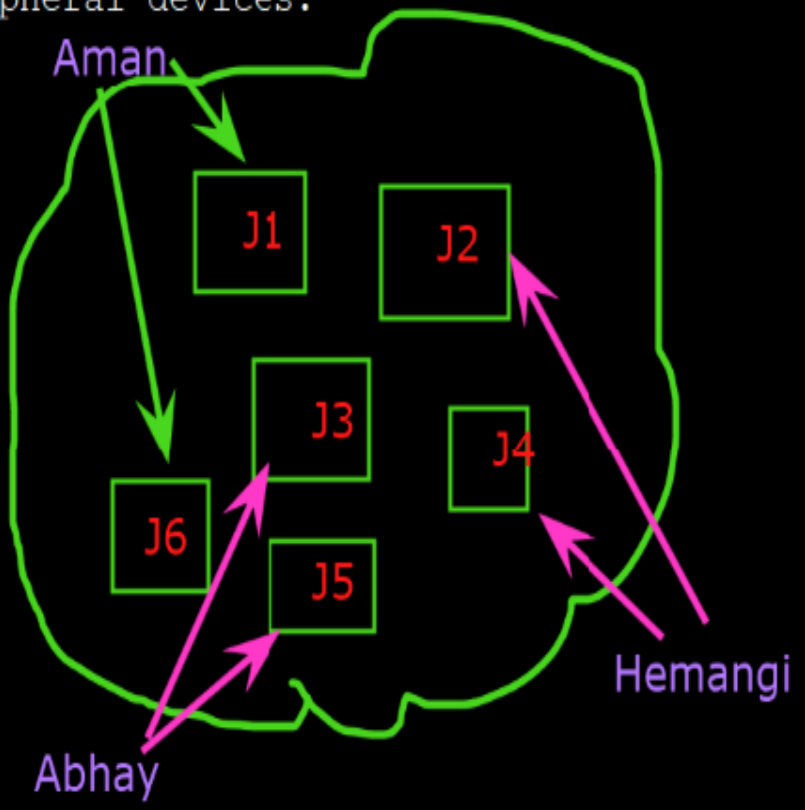
- A Multiprocessor system consists of **several processors** that share a common physical memory.
- Multiprocessor system provides **higher computing power and speed**.
- In multiprocessor system all processors **operate under single operating system**.
- **Multiplicity of the processors and how they do act together** are transparent to the others.

Advantages of Multiprocessor Systems

1. Enhanced performance
2. Execution of several tasks by different processors concurrently, increases the system's throughput without speeding up the execution of a single task.
3. If possible, system divides task into many subtasks and then these subtasks can be executed in parallel in different processors. Thereby speeding up the execution of single tasks.

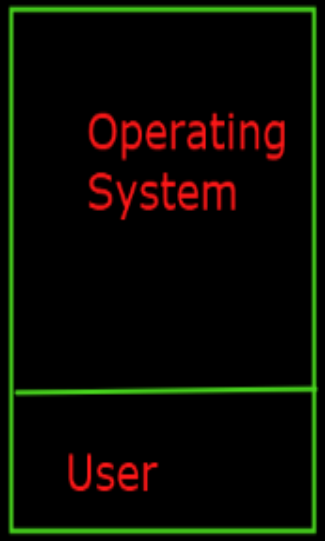
2. Multiprocessor System

-prallel system or tightly coupled system
-2/more processors in close communiucation, sharing the computer bus and somtimes the clock peripheral devices.



CPU Utilizatio is incresed:

Efficiency = Throghput

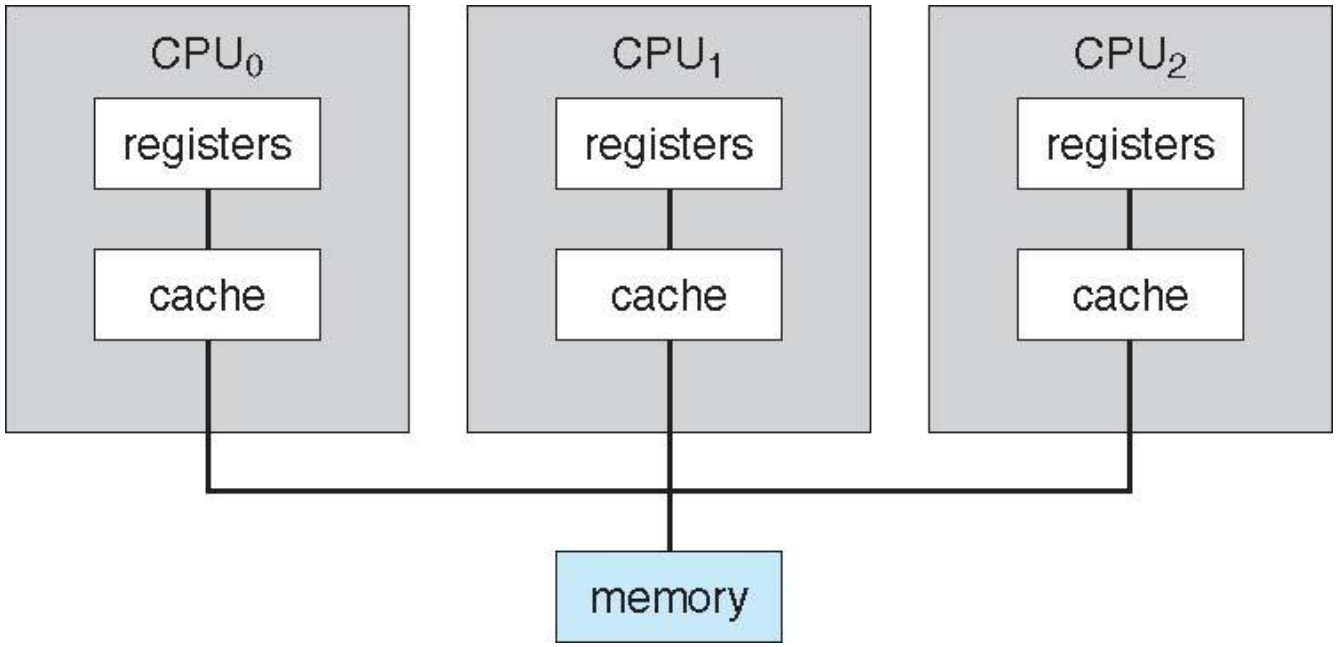


Main Memory (RAM)

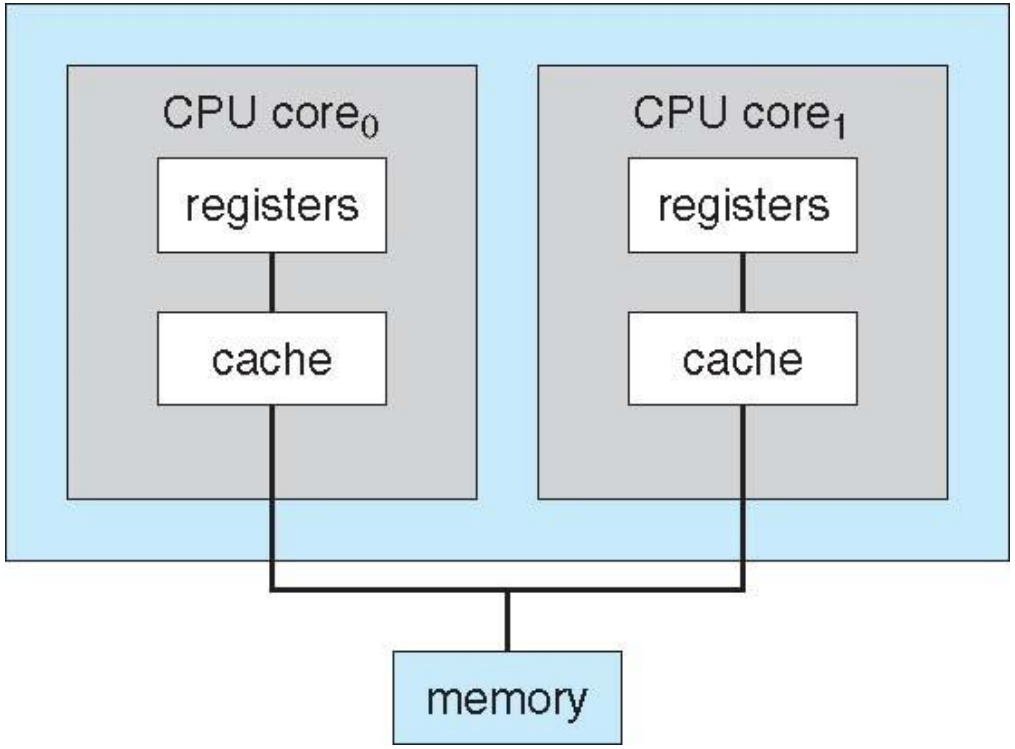
Computer-System Architecture

- Most systems use a single general-purpose processor (PDAs through mainframes)
  - Most systems have special-purpose processors as well
- Multiprocessors systems growing in use and importance
  - Also known as parallel systems, tightly-coupled systems
  - Advantages include
    1. Increased throughput
    2. Economy of scale
    3. Increased reliability – graceful degradation or fault tolerance
  - Two types
    1. Asymmetric Multiprocessing
    2. Symmetric Multiprocessing

Symmetric Multiprocessing Architecture

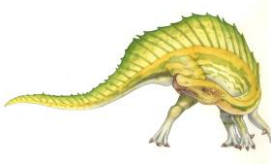


A Dual-Core Design



Operating System Structure

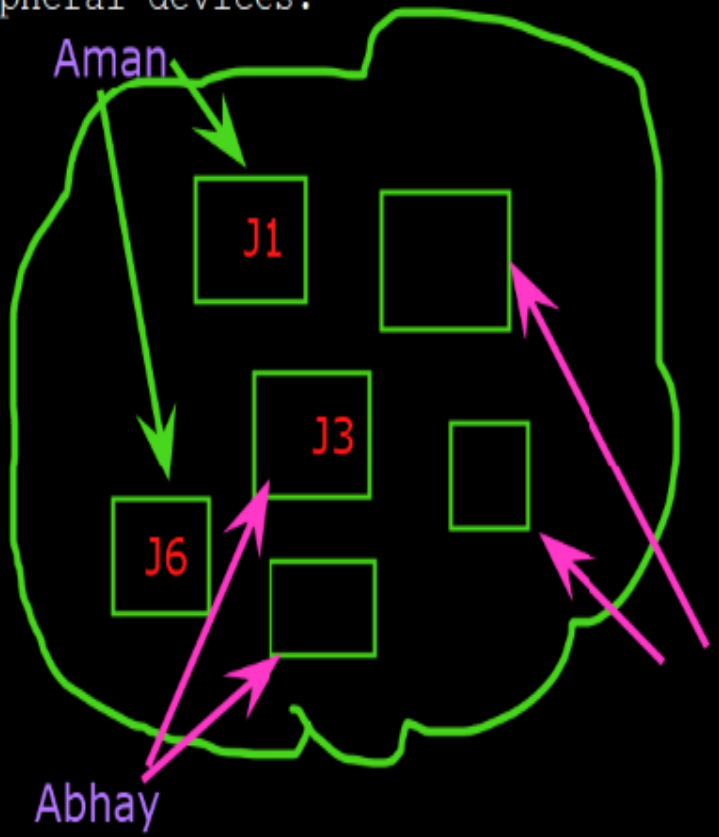
- Multiprogramming needed for efficiency
  - Single user cannot keep CPU and I/O devices busy at all times
  - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job selected and run via job scheduling
  - When it has to wait (for I/O for example), OS switches to another job
- Timesharing (multitasking) is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating interactive computing
  - Response time should be < 1 second
  - Each user has at least one program executing in memory □process
  - If several jobs ready to run at the same time □ CPU scheduling
  - If processes don't fit in memory, swapping moves them in and out to run
  - Virtual memory allows execution of processes not completely in memory



2. Multiprocessor System

-prallel system or tightly coupled system

-2/more processors in close communiucation, sharing the computer bus and somtimes the cloc peripheral devices.

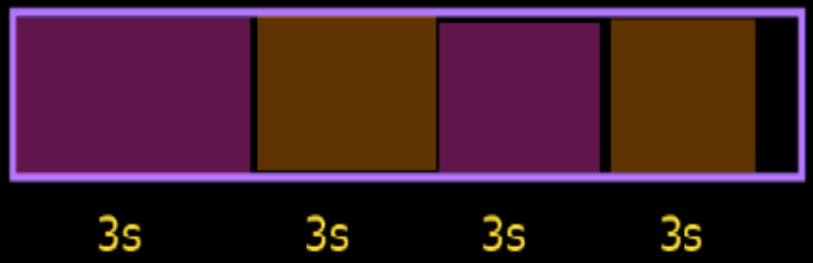


1. Multiprogramming

-Multiple user -time slice = 3s

2. Multitasking

-Multiple task



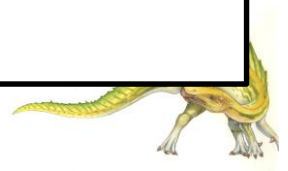
Operating-System Operations

- Interrupt driven by hardware
- Software error or request creates exception or trap
  - Division by zero, request for operating system service
- Other process problems include infinite loop, processes modifying each other or the operating system
- Dual-mode operation allows OS to protect itself and other system components
  - User mode and kernel mode
  - Mode bit provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as privileged, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user

Desktop Systems

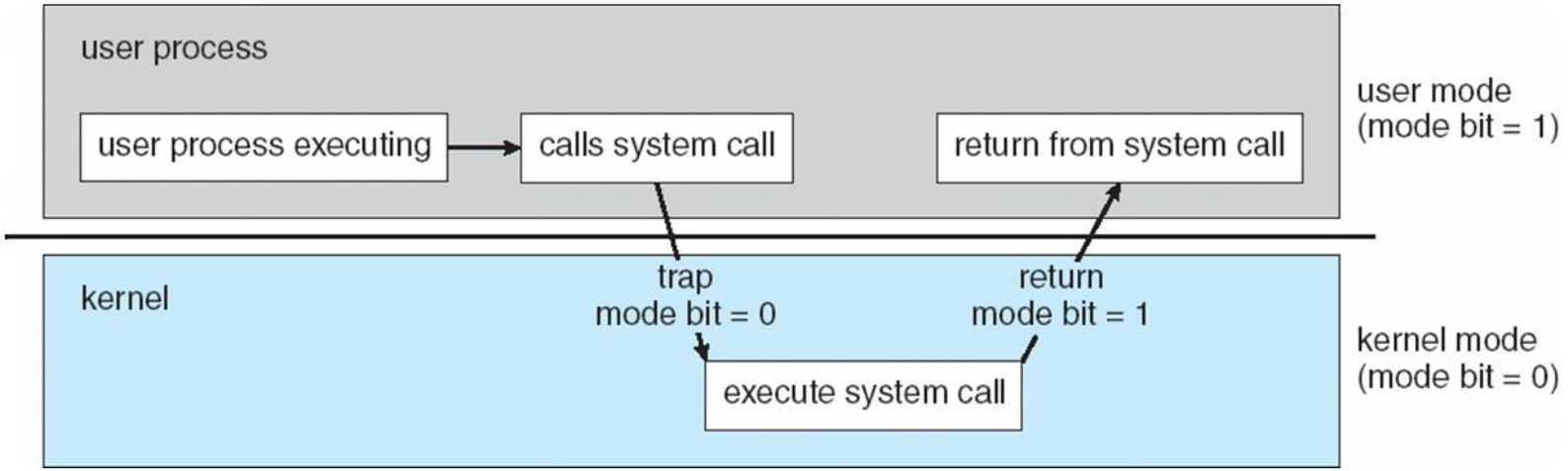
- Earlier, CPUs and PCs lacked the features needed to protect an operating system from user programs.
- PC operating systems therefore were neither multiuser nor multitasking.
- However, the goals of these operating systems have changed with time; instead of maximizing CPU and peripheral utilization, the systems opt for maximizing user convenience and responsiveness.
- These systems are called **Desktop Systems** and include PCs running Microsoft Windows and the Apple Macintosh.
- Operating systems for these computers have benefited in several ways from the development of operating systems for **mainframes.**

- **Microcomputers** were **immediately able to adopt some of the technology developed for larger operating systems**.
- On the other hand, the hardware costs for microcomputers are sufficiently **low** that individuals have sole use of the computer, and CPU utilization is no longer a prime concern.
- Thus, some of the design decisions made in operating systems for mainframes may not be appropriate for smaller systems.



Transition from User to Kernel Mode

- Timer to prevent infinite loop / process hogging resources
  - Set interrupt after specific period
  - Operating system decrements counter
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time



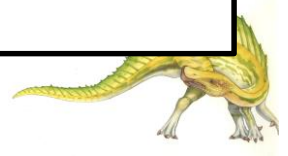
Distributed Operating System

- The motivation behind developing distributed operating systems is the availability of powerful and inexpensive microprocessors and advances in communication technology.

- These advancements in technology have made it possible to design and develop distributed systems comprising of many computers that are inter connected by communication networks. The main benefit of distributed systems is its low price/performance ratio.

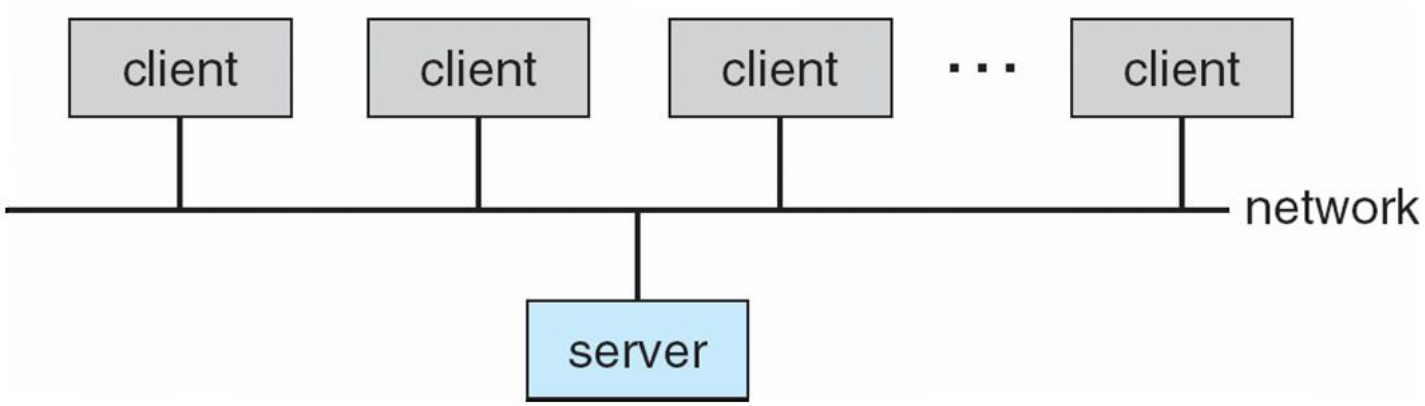
Advantages Distributed Operating System

- As there are multiple systems involved, user at one site can utilize the resources of systems at other sites for resource-intensive tasks.
- Fast processing.
- Less load on the Host Machine.



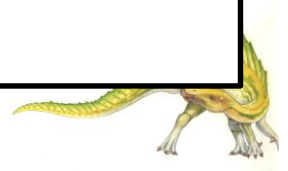
Computing Environments (Cont)

- 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)
    - File-server provides interface for clients to store and retrieve files



Types of Distributed Operating Systems

- Following are the two types of distributed operating systems used:
  - Client-Server Systems
  - Peer-to-Peer Systems
- **Client-Server Systems**
- Centralized systems today act as server systems to satisfy requests generated by client systems. The general structure of a client-server system is depicted in the figure below:
- Server Systems can be broadly categorized as: Compute Servers and File Servers.
- Compute Server systems, provide an interface to which clients can send requests to perform an action, in response to which they execute the action and send back results to the client.
- File Server systems, provide a file-system interface where clients can create, update, read, and delete files.



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 *Gnutella*

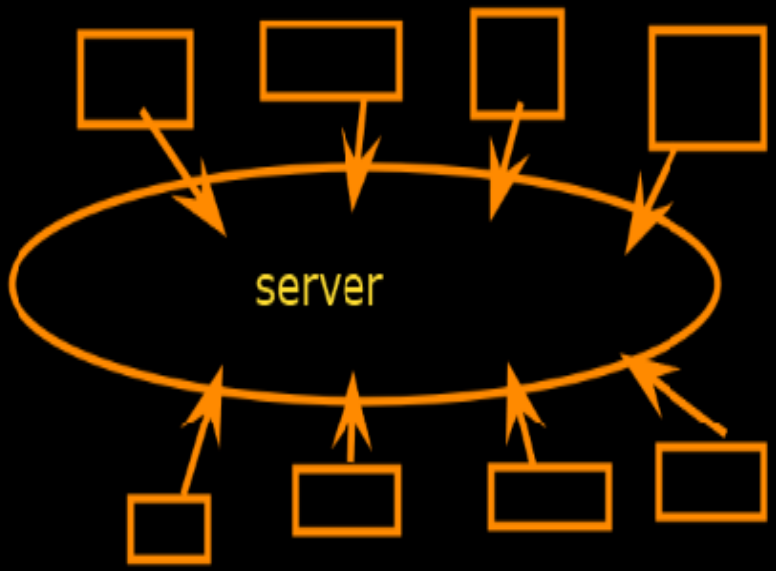
Clustered Systems

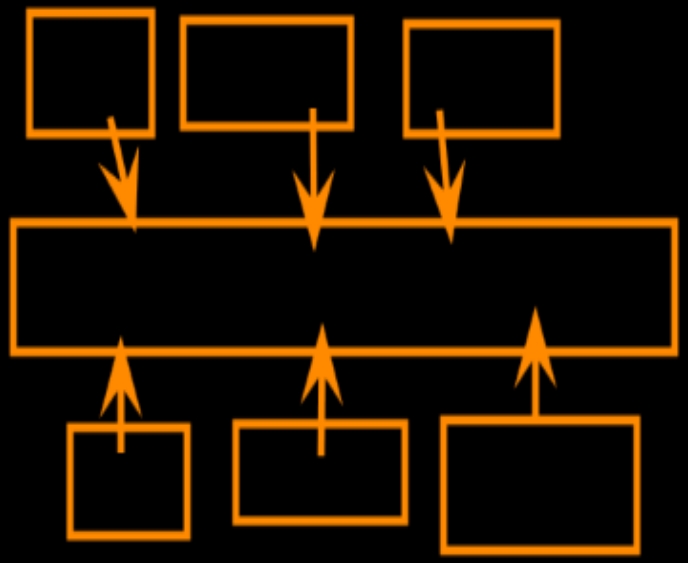
- Like multiprocessor systems, but multiple systems working together
  - Usually sharing storage via a storage-area network (SAN)
  - Provides a high-availability service which survives failures
    - Asymmetric clustering has one machine in hot-standby mode
    - Symmetric clustering has multiple nodes running applications, monitoring each other
  - Some clusters are for high-performance computing (HPC)
    - Applications must be written to use parallelization

3. Clustered System

Client-Server system

Peer to peer system



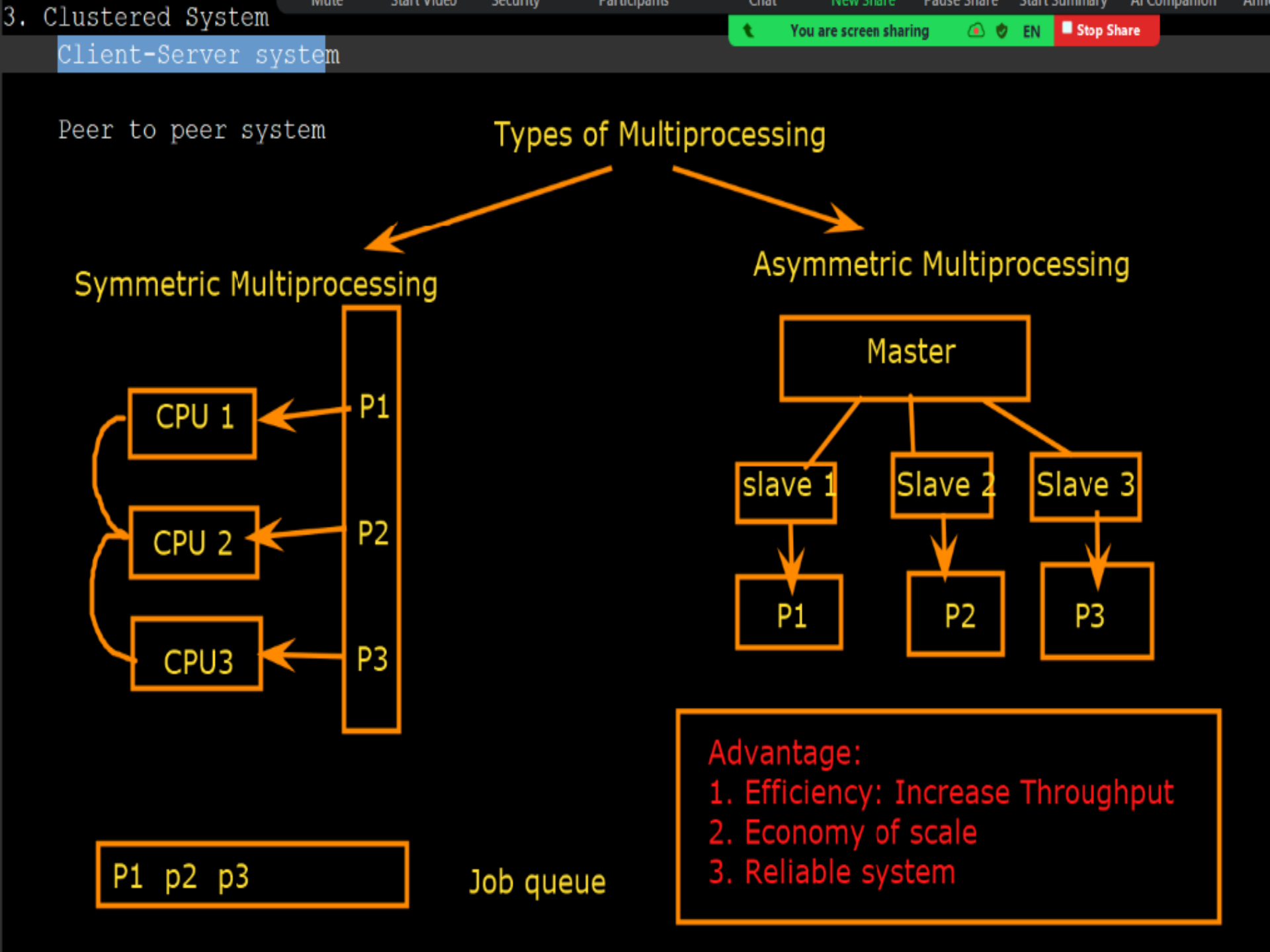


Clusters

3. Clustered System

Client-Server system

Peer to peer system



Real Time Operating System

- It is defined as an operating system **known to give maximum time for each of the critical operations** that it performs, like OS calls and interrupt handling.
- The Real-Time Operating system which guarantees the maximum time for critical operations and complete them on time are referred to as Hard Real-Time Operating Systems.
- While the real-time operating systems that can only guarantee a maximum of the time, i.e. the critical task will get priority over other tasks, but no assurity of completing it in a defined time. These systems are referred to as Soft Real-Time Operating Systems.

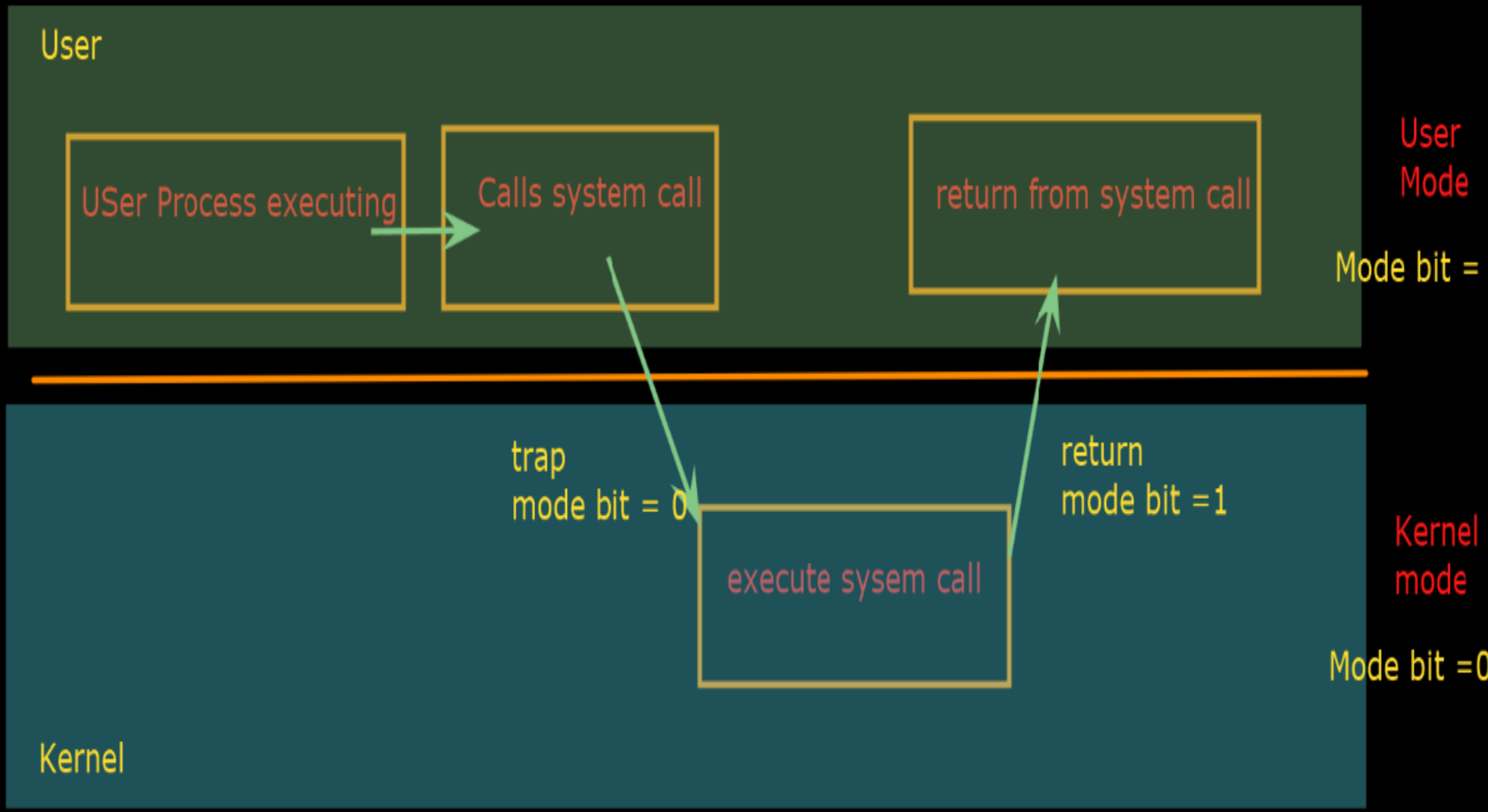
errupt

nel and User mode : Dual mode

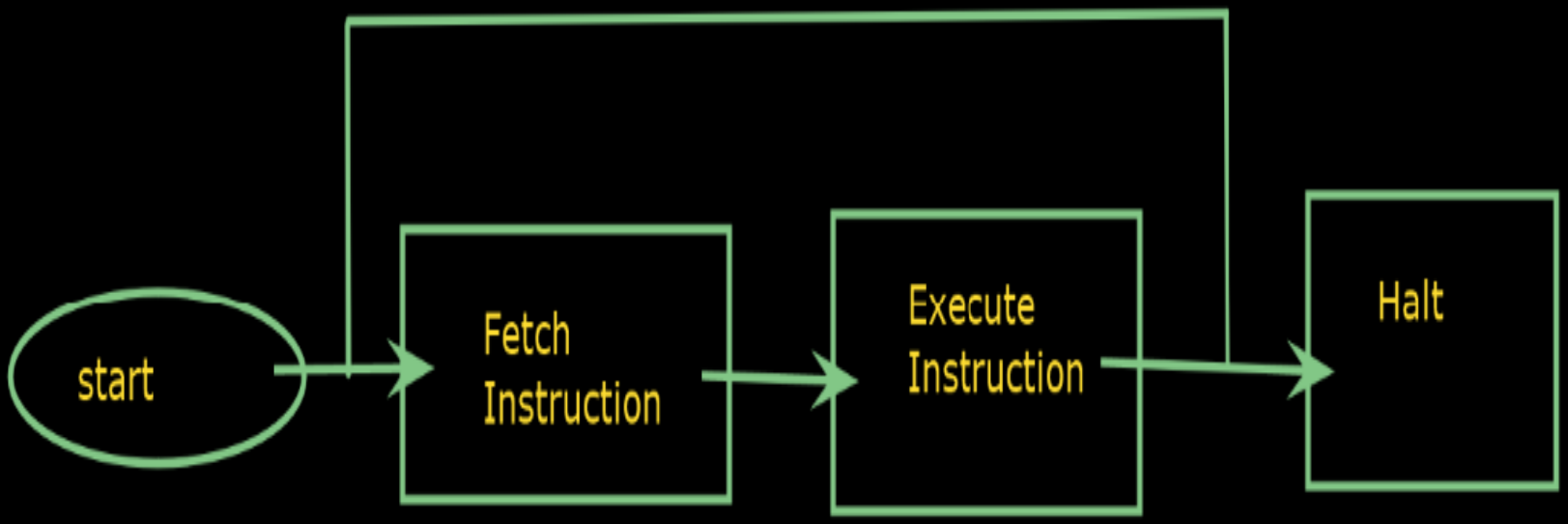
-Kernel mode: handle hardware opration

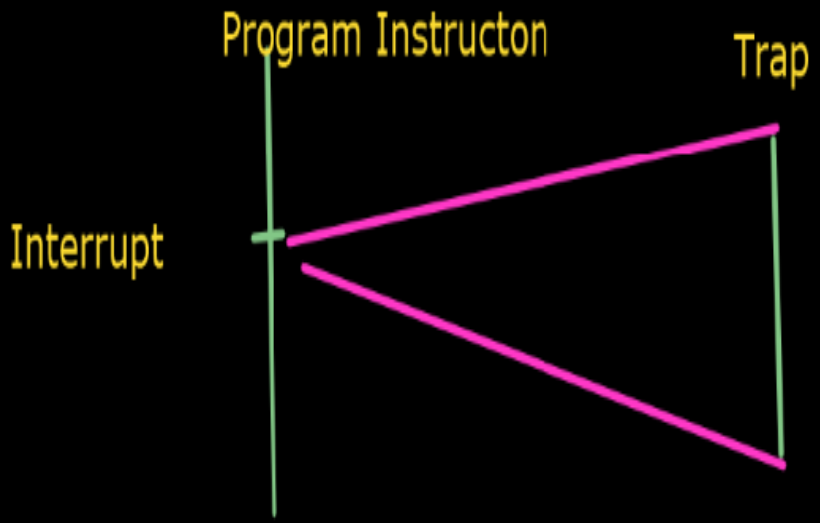
-USer mode: handle application software

-Mode bit: undersatand by Kernel mode



Interrupt





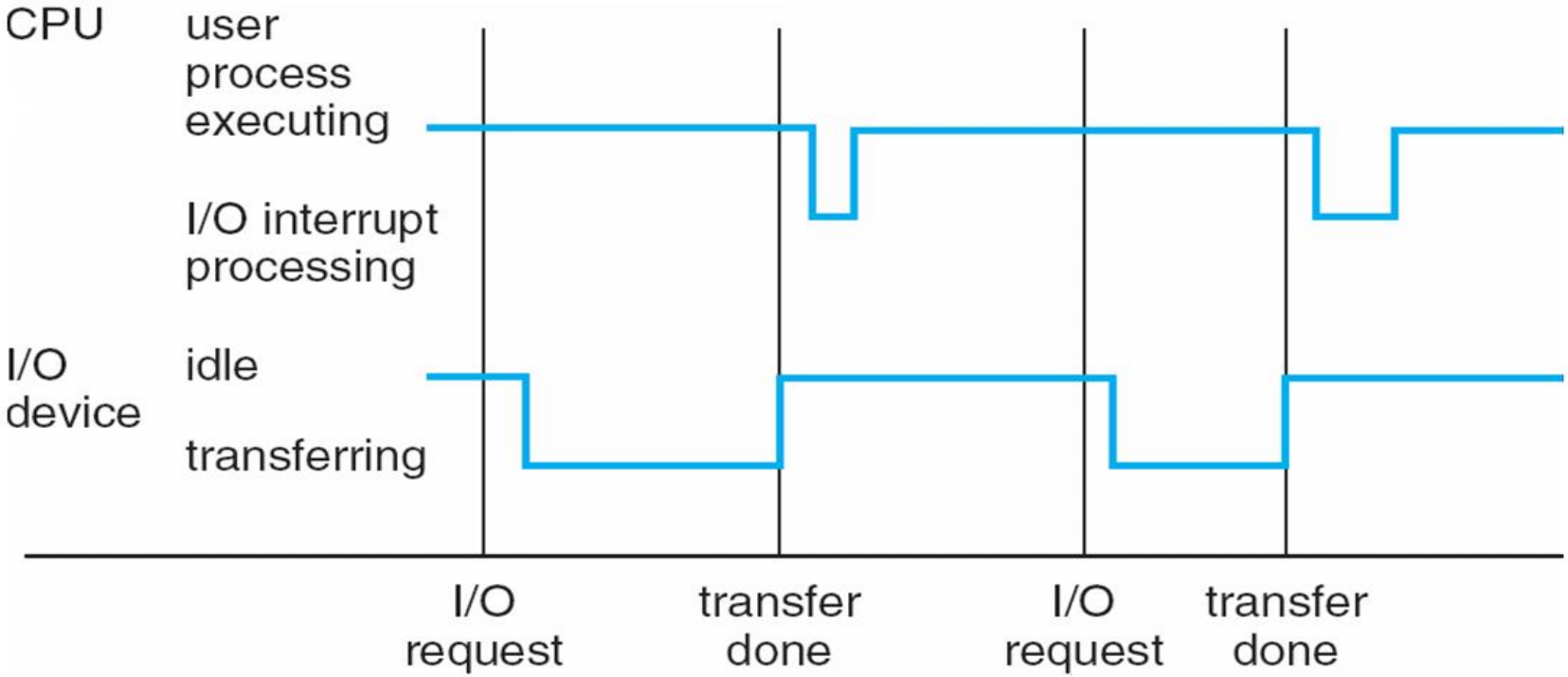
Interrupt Handling:

---

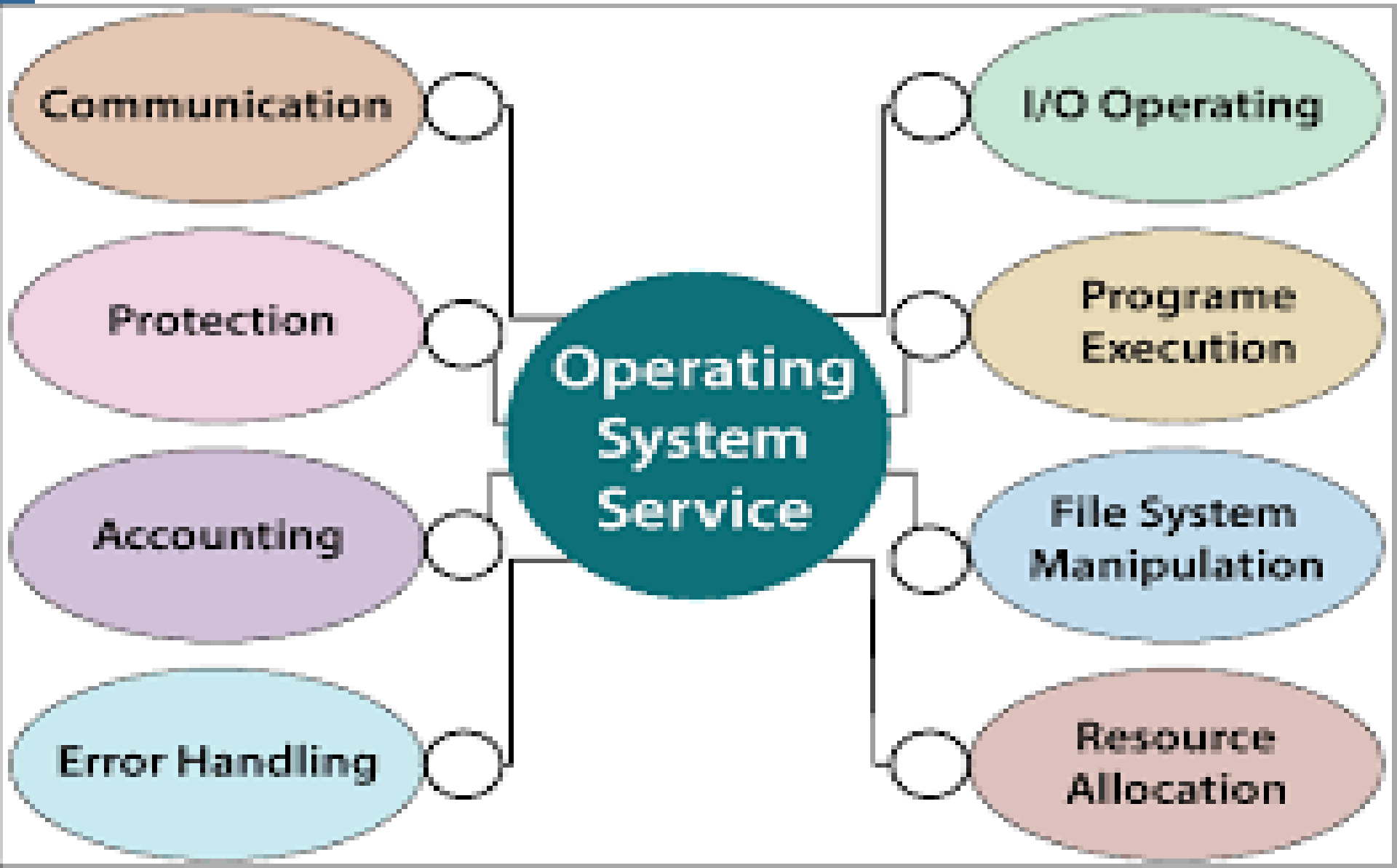
1. Polling
2. Vectored Interrup

System.

Interrupt Timeline



Operating System Services



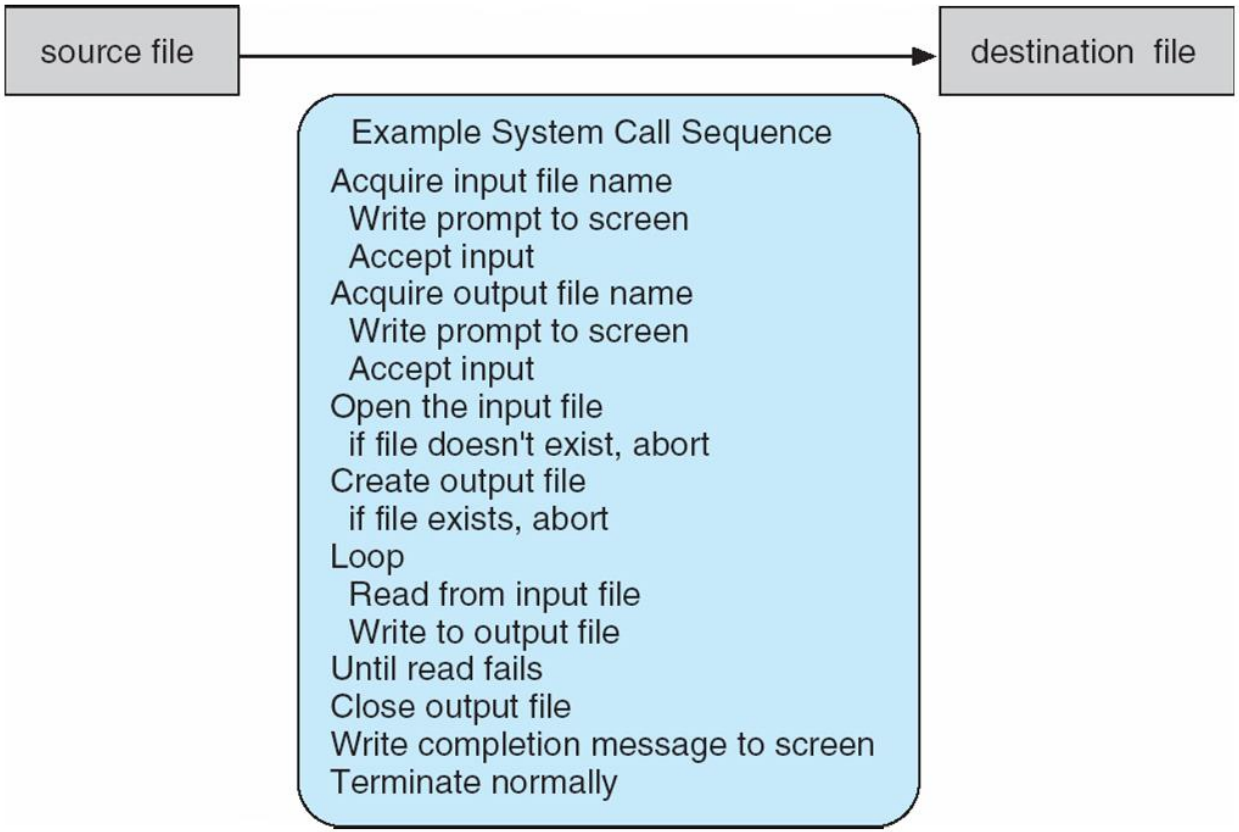
System Calls

- Programming interface to the services provided by the OS
- Typically written in a high-level language (C or C++)
- Mostly accessed by programs via a high-level Application Program Interface (API) rather than direct system call use
- Three most common APIs are Win32 API for Windows, POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X), and Java API for the Java virtual machine (JVM)
- Why use APIs rather than system calls?

(Note that the system-call names used throughout this text are generic)

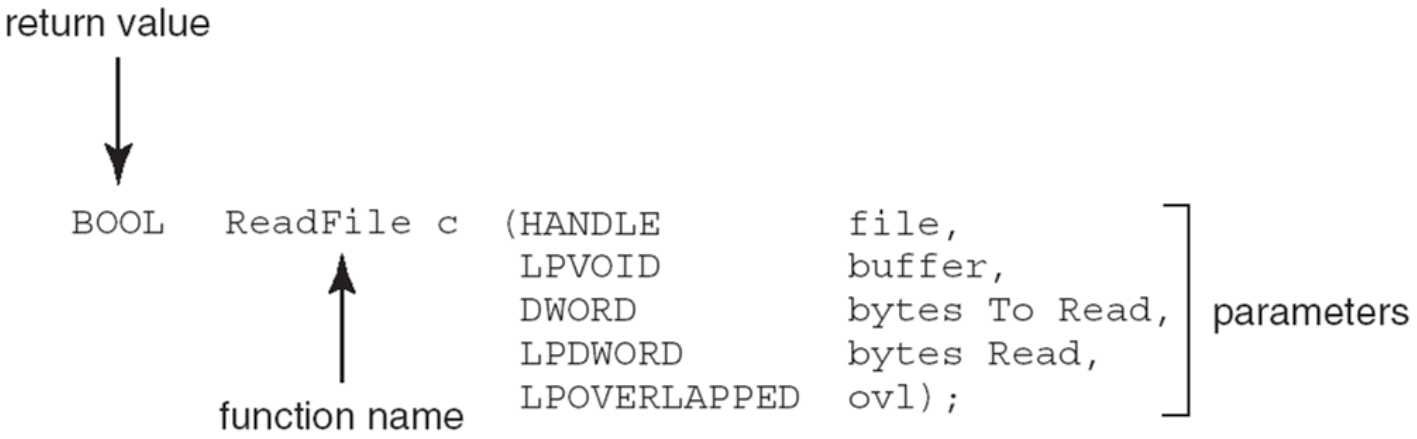
Example of System Calls

- System call sequence to copy the contents of one file to another file



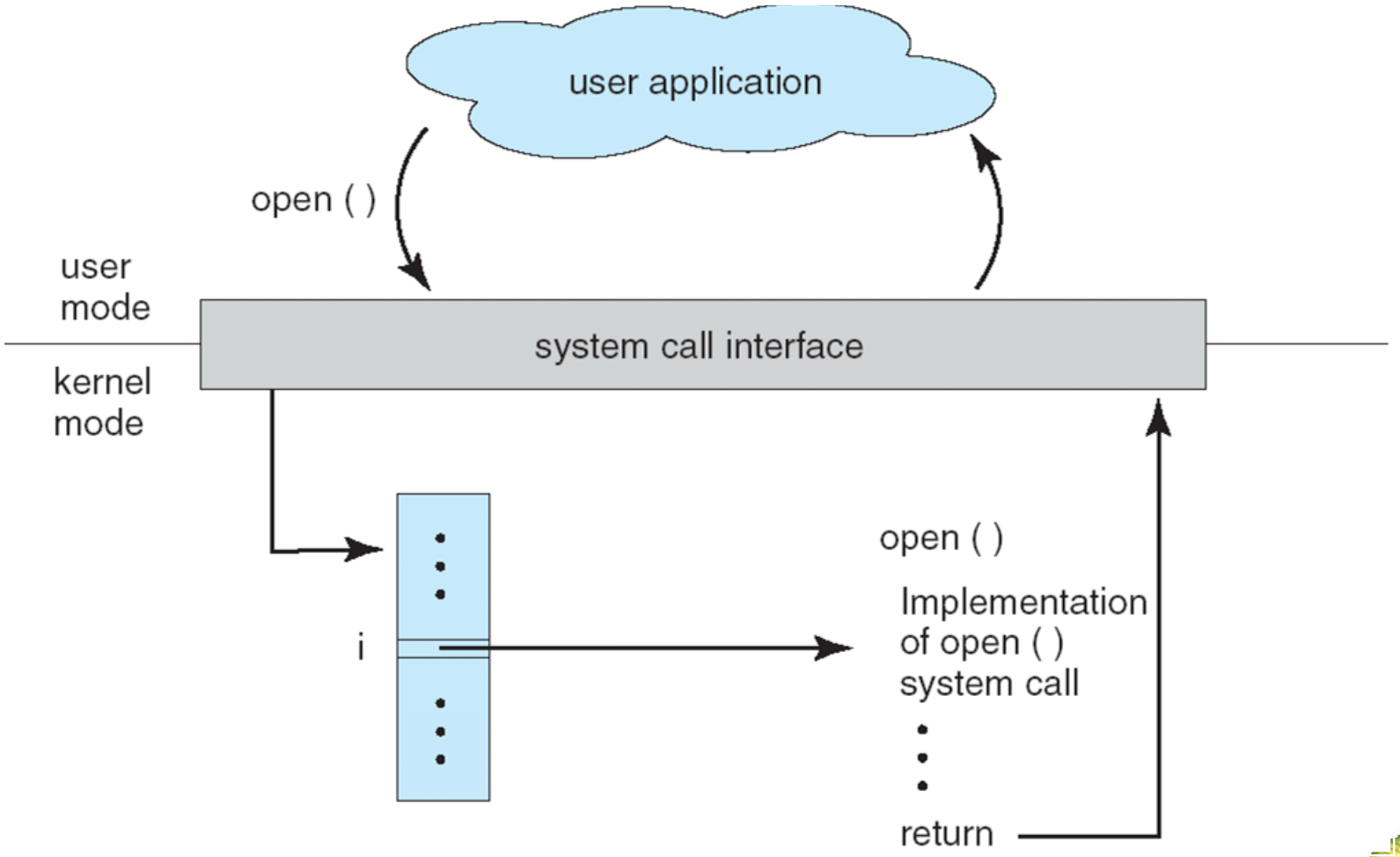
Example of Standard API

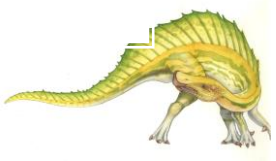
- Consider the ReadFile() function in the
- Win32 API—a function for reading from a file



- A description of the parameters passed to ReadFile()
  - HANDLE file—the file to be read
  - LPVOID buffer—a buffer where the data will be read into and written from
  - DWORD bytesToRead—the number of bytes to be read into the buffer
  - LPDWORD bytesRead—the number of bytes read during the last read
  - LPOVERLAPPED ovl—indicates if overlapped I/O is being used

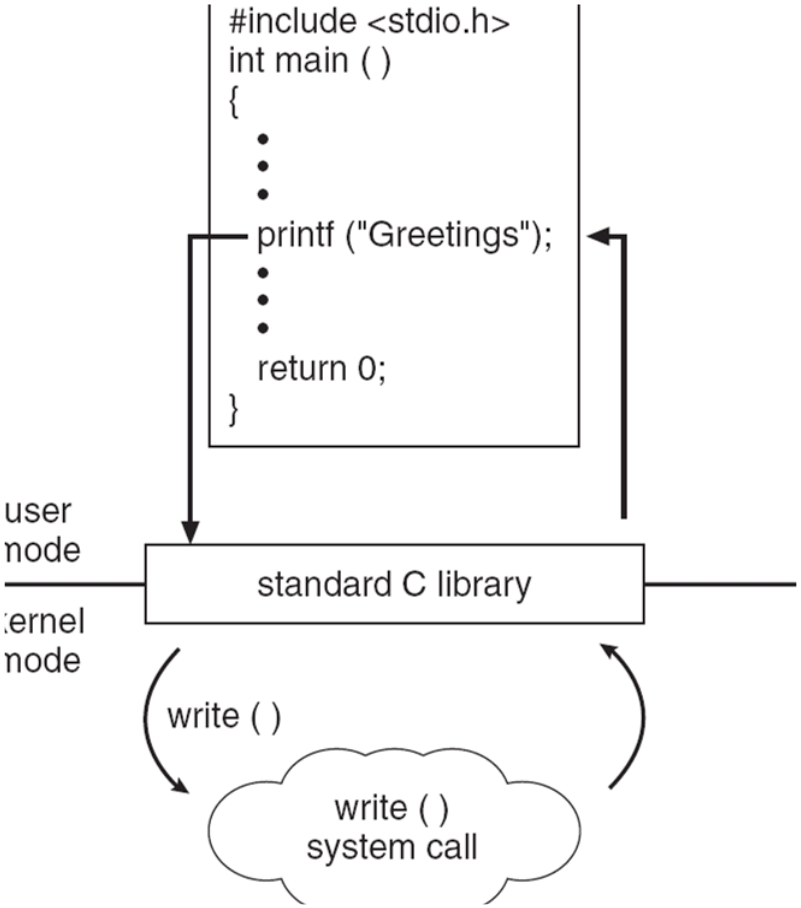
API – System Call – OS Relationship



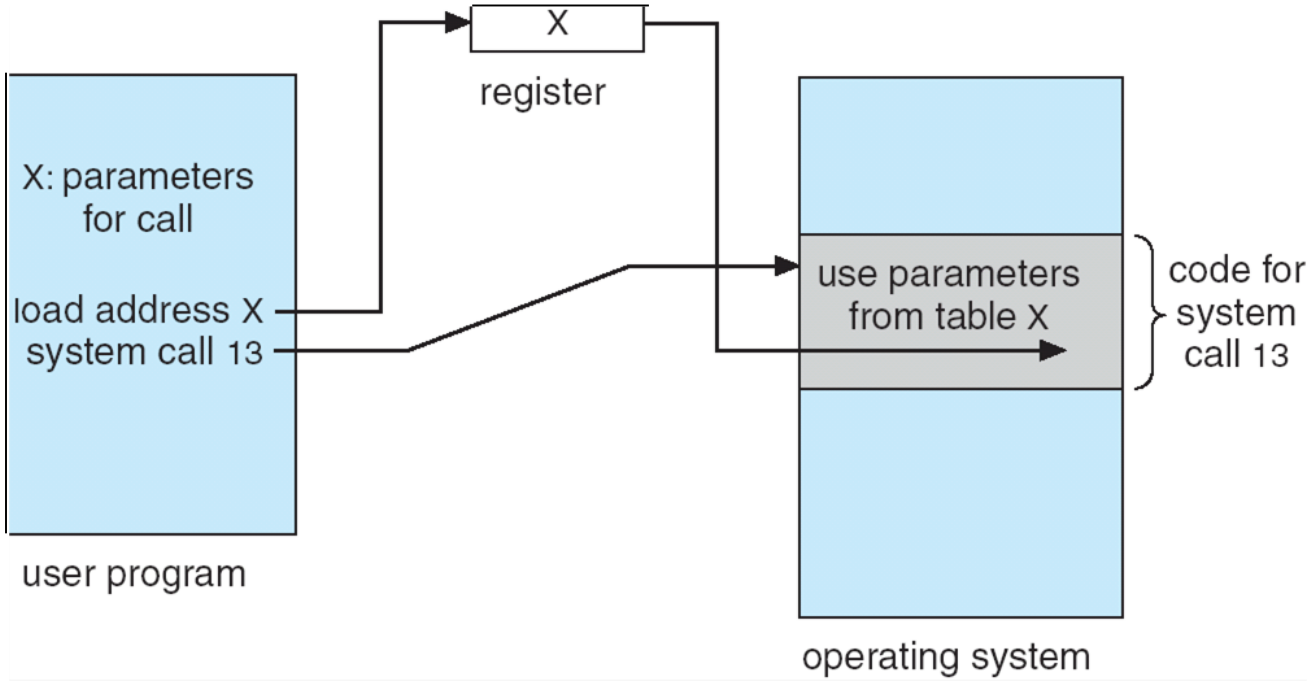


Standard C Library Example

- C program invoking printf() library call, which calls write() system call



Parameter Passing via Table



Types of System Calls

- Process control
- File management
- Device management
- Information maintenance
- Communications
- Protection

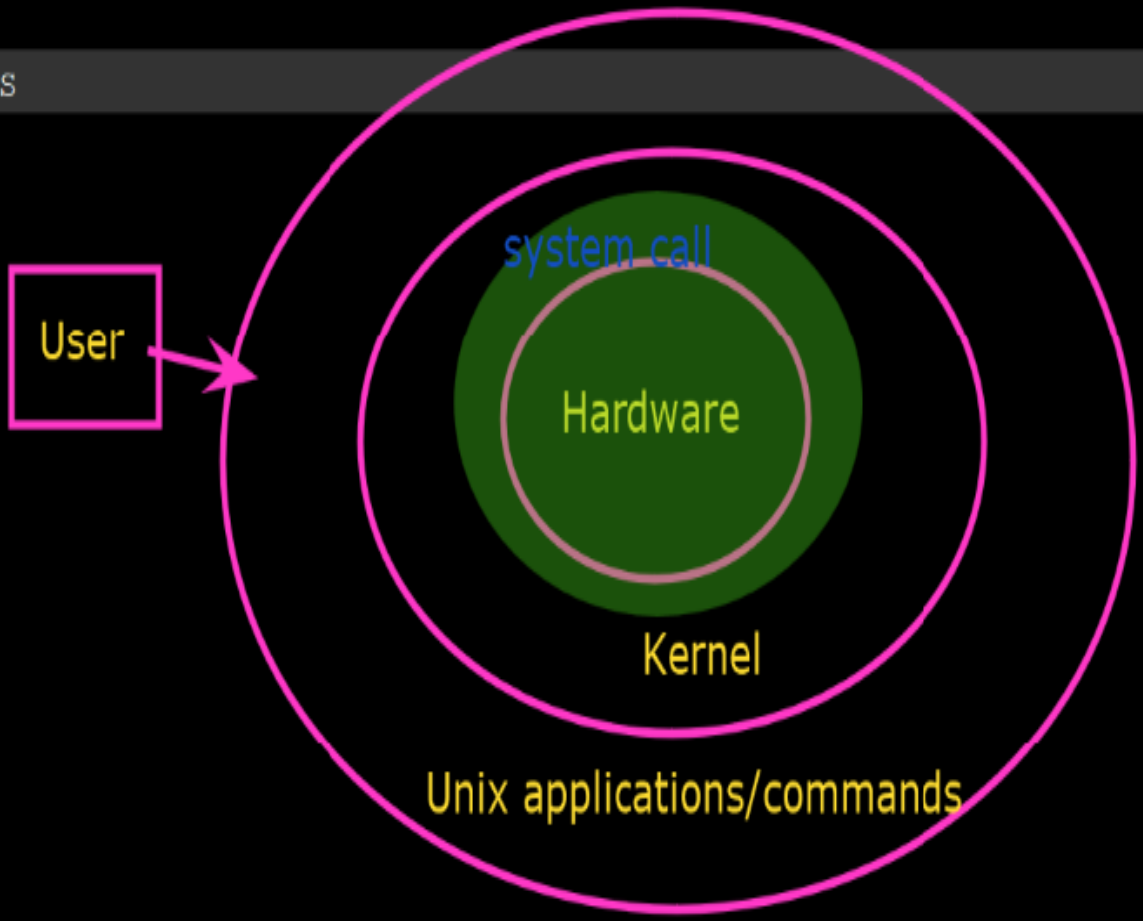
Examples of Windows and Unix System Calls

	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() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

Powerful features of Unix:

-Security: private and shared files
-Multi-user support
-Inter process communication
-Extensive network support
-data transmission to I/O devices
-STABILITY:
-Example: Critical applications

Layered UNIX architecture



Kernel:

-heart of OS

-the low level core of the system that is the interface between user application and hardwre.

-Function:

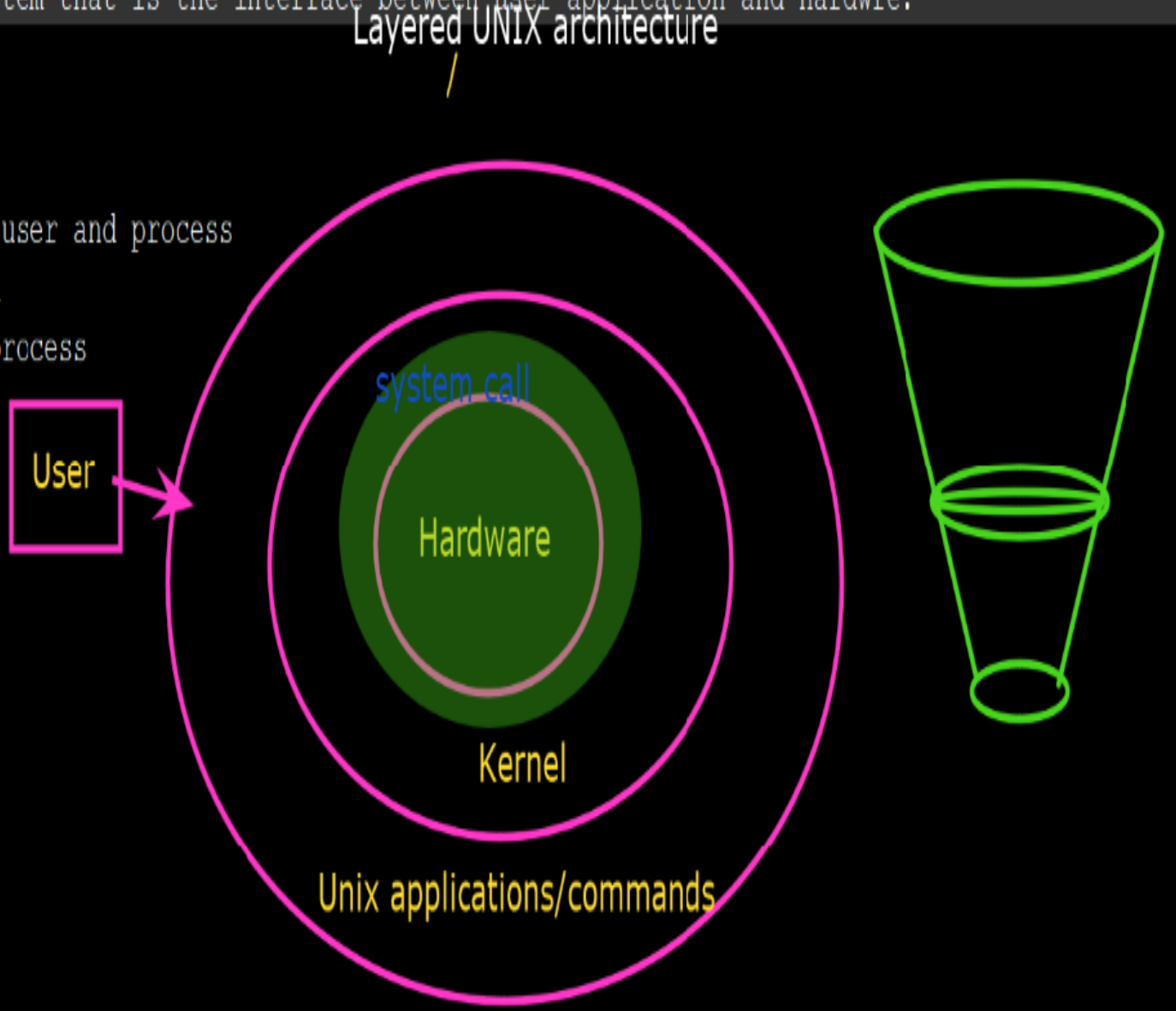
Mange memory

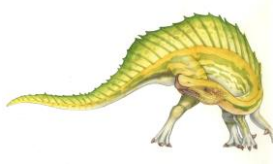
Manage I/O devices

Allocate the time between user and process

Interprocess communication

Setting the priority for process





Command [option] [arguments]

pwd: present working directory

ls:list of files

File system: