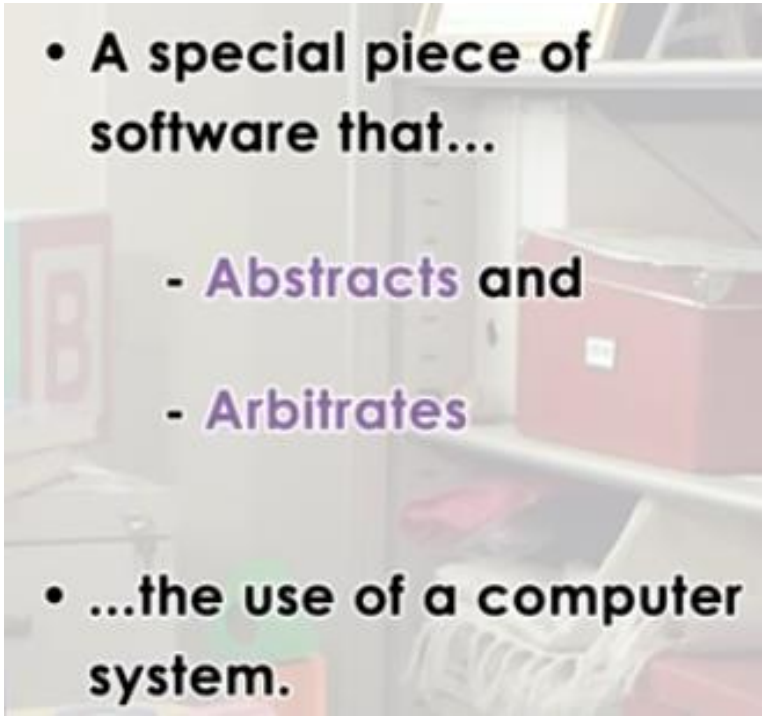
Chapter 1: Introduction

(OS Structure, Modes and Services)

By: Akhil Sohal

OPERATING SYSTEM?



- Abstract: To simplify how hardware actually looks like.
- Arbitrate: To manage , to oversee the hardware use

What is an Operating System?

- It is a layer of system software that:
  - directly has privileged access to underlined hardware
  - hides hardware complexity
  - manages hardware on behalf of one or more applications according to policies.

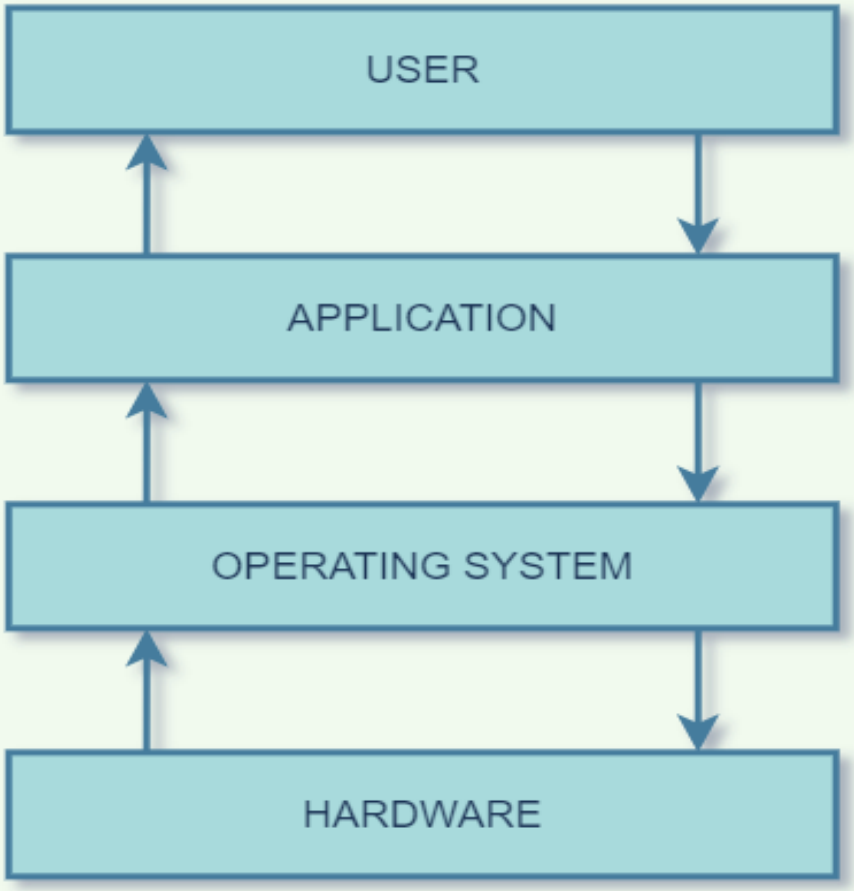
What is an Operating System?

- **What is an Operating system?**
  - **A program that acts as an intermediate/ interface between a user's application program of a computer and the computer hardware.**
  - **Resource allocator (Managing the resources efficiently)**
  - **Control Program**

- **Operating system goals:**
  - **Execute user programs and make problem solving easier.**
  - **Make the computer system convenient to use**
  - **Efficiently use available resources**

- Kernel is a program that (allow) let the hardware to recognize and read the program/process.

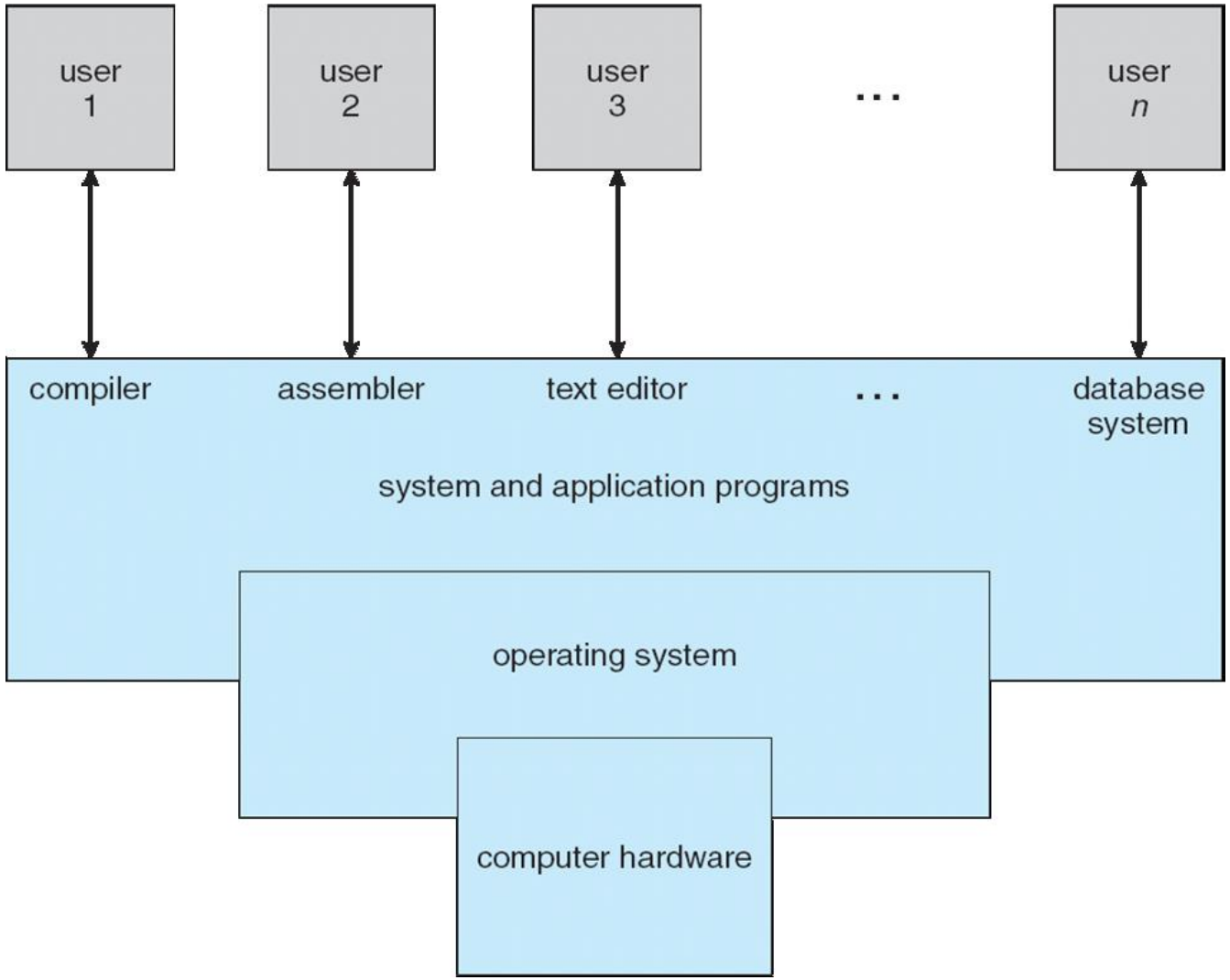
Operating System



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 resources** among various applications and users
  - **System/Application programs** – define the ways in which the system resources are used to solving user problems
    - Word processors, compilers, web browsers, database systems, video games
  - **Users**
    - People, machines, other computers

Four Components of a Computer System



Need of Operating System

Let's suppose you want to add two numbers: c=a+b;

No OS

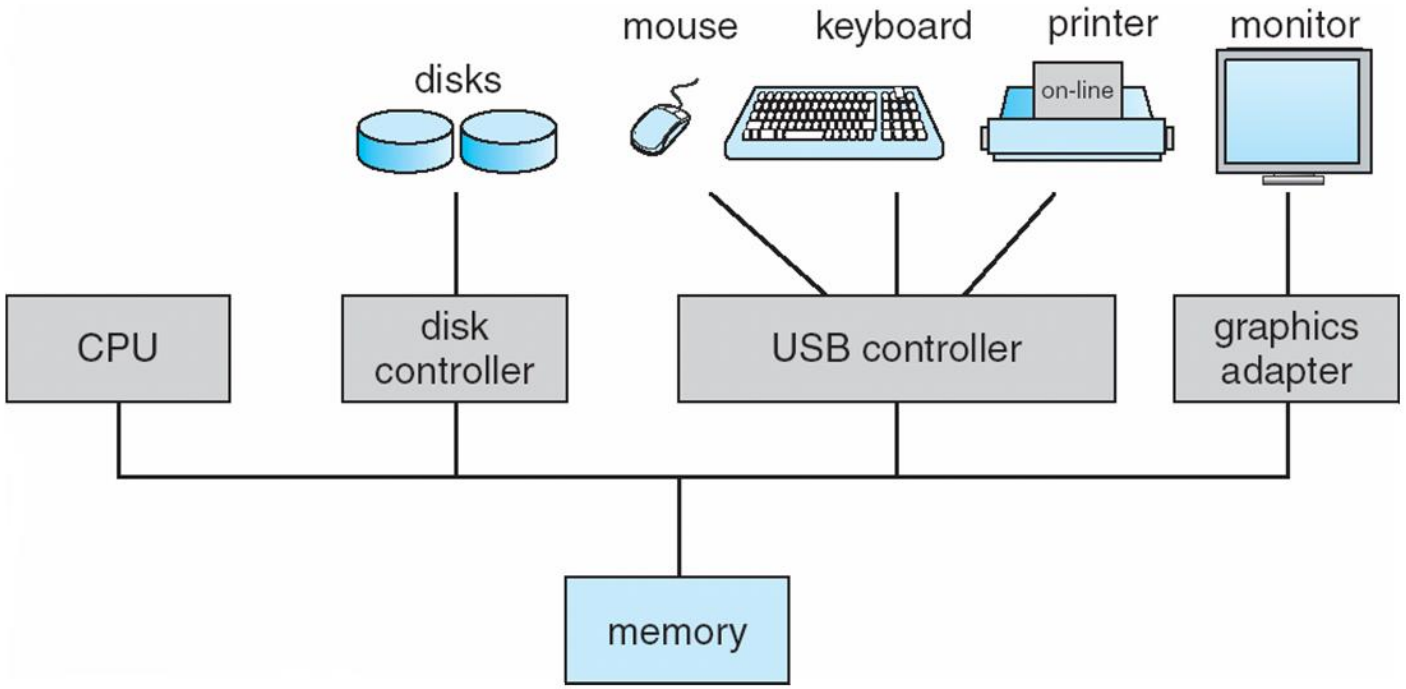
If you are working on MC6800 hardware then the instructions will be:
LDAA $80 – Loading the number at memory location 80
LDAB $81 – Loading the number at memory location 81
ADDB – Adding these two numbers
STAA $55 – Storing the sum to memory location 55
But the moment you hardware changes, for example, say to 8086 or 8088, all the above instructions will change, a problem!

With OS

Use any High Level Language like C.
Write c=a+b;
The underlying hardware does not matter because OS takes care of convertion. Infact this is what we always do. Have we ever bothered what hardware are we using. Our focus is just writing the code correctly in C, C++, Python etc.

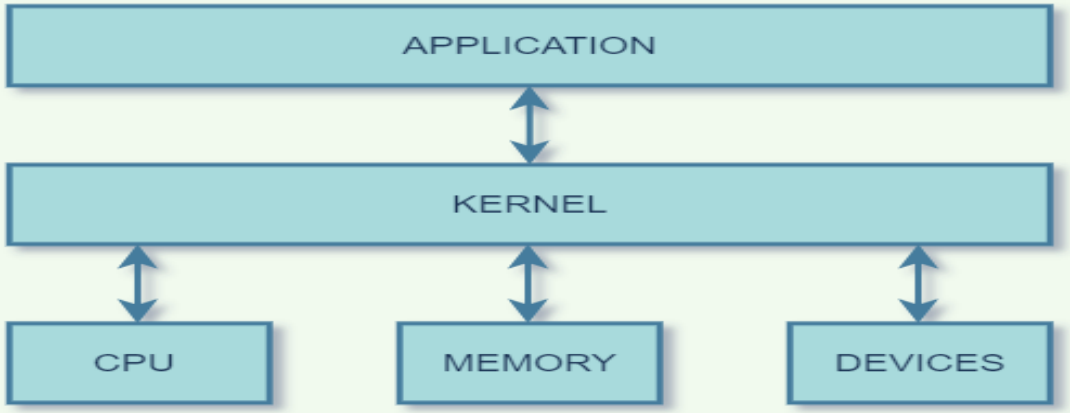
Computer System Organization

- 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



What is Kernel?

- The kernel is the core component of an operating system for a computer (OS).
- All other components of the OS rely on the core to supply them with essential services.
- It serves as the primary interface between the OS and the hardware and aids in the control of devices, networking, file systems, and process and memory management.



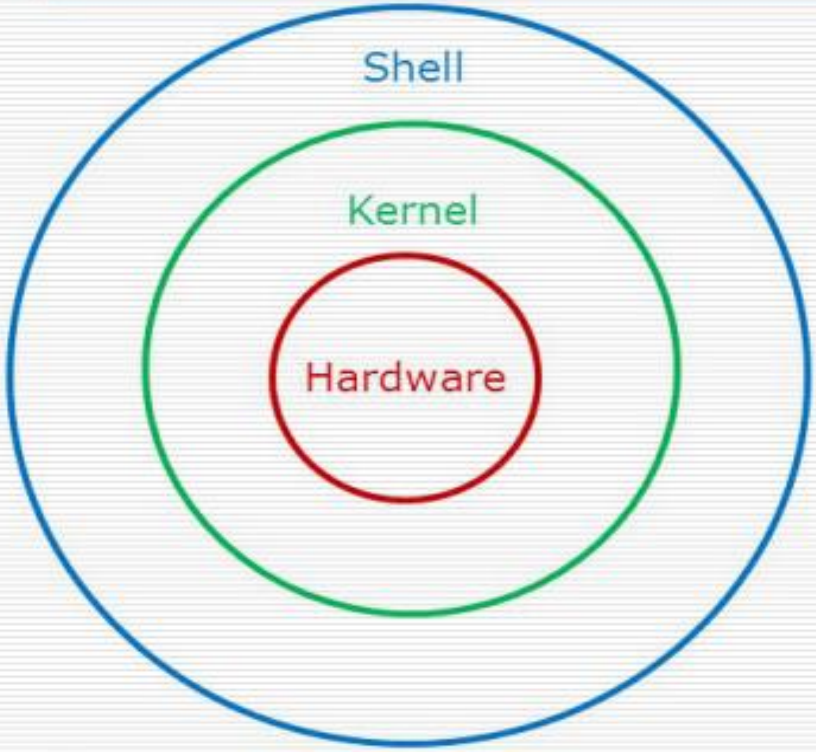
Kernel in Operating System

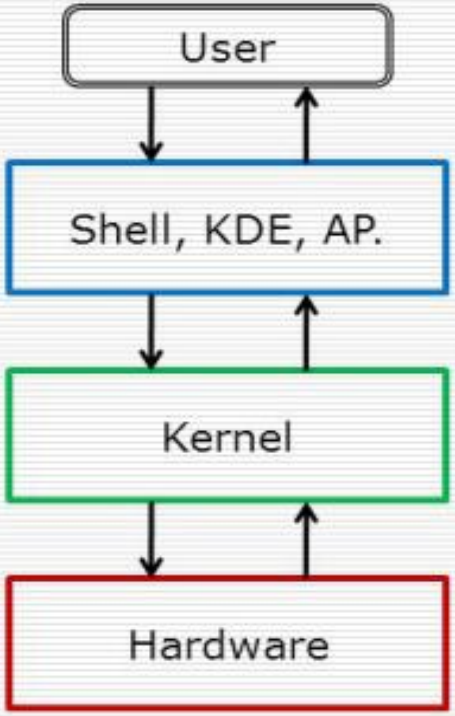
Kernel is central component of an operating system that manages operations of computer and hardware. It basically manages operations of memory and CPU time. It is core component of an operating system. Kernel acts as a bridge between applications and data processing performed at hardware level using inter-process communication and system calls.

Objectives of Kernel

- To establish communication between user level application and hardware.
- To decide state of incoming processes.
- To control disk management.
- To control memory management.
- To control task management.

Where is Kernel ?





System Structure

•**Layer-1: Hardware –**
It consists of all hardware related information.

•**Layer-2: Kernel –**
It interacts with hardware and most of the tasks like memory management, task scheduling, and management are done by the kernel.

•**Layer-3: Shell commands –**
Shell is the utility that processes your requests. When you type in a command at the terminal, the shell interprets the command and calls the program that you want. There are various commands like cp, mv, cat, grep, id, wc, nroff, a.out and more.

•**Layer-4: Application Layer –**
It is the outermost layer that executes the given external applications.

What is shell in Operating System

- Computers understand the language of zeros and ones known as binary language.
- In the early days of computing instructions were provided using binary language, which is difficult for all of us to read and write.
- Therefore, in an operating system there is a special program called the shell.
- The shell accepts human readable commands and
- translates them into something the kernel can read and process.

Supervisor mode and user mode

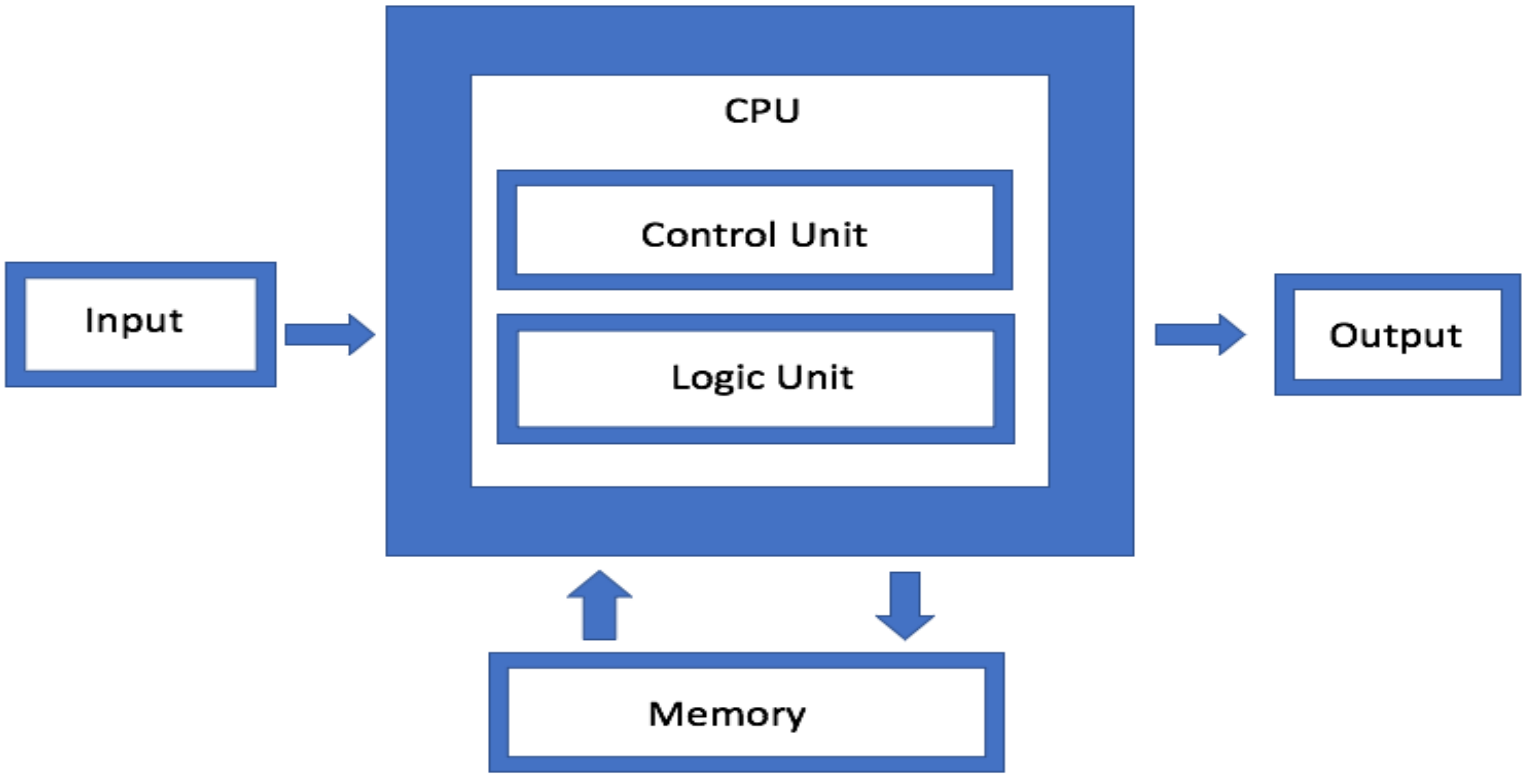
The unrestricted mode is called **supervisor mode or kernel mode**, and the restricted one is called **user mode**.

•The operating system runs in kernel mode or supervisor mode whereas compiler and editors run in user mode.

•If a user wants to write a new compiler and replace the provided one she can do it, but she is not free to write her own clock interrupt handler, which is part of the operating system and is normally protected by hardware against attempts by users to modify it.
•User modes does not allow operations like writes to random memory, to protect programs from one another whereas the supervisor mode allow such operation as the operating system needs such kind of stuff.

Review of Computer Organization

- I von Neumann Architecture



Bootstrap Loader

- A bootstrap loader begins loading the operating system and the control is eventually passed on to the operating system kernel.
- A bootstrap loader might alternatively load a more comprehensive loader that accomplishes loading the operating system kernel.
- A bootstrap loader typically resides in the ROM (Read Only Memory)

MULTICORE

VERSUS

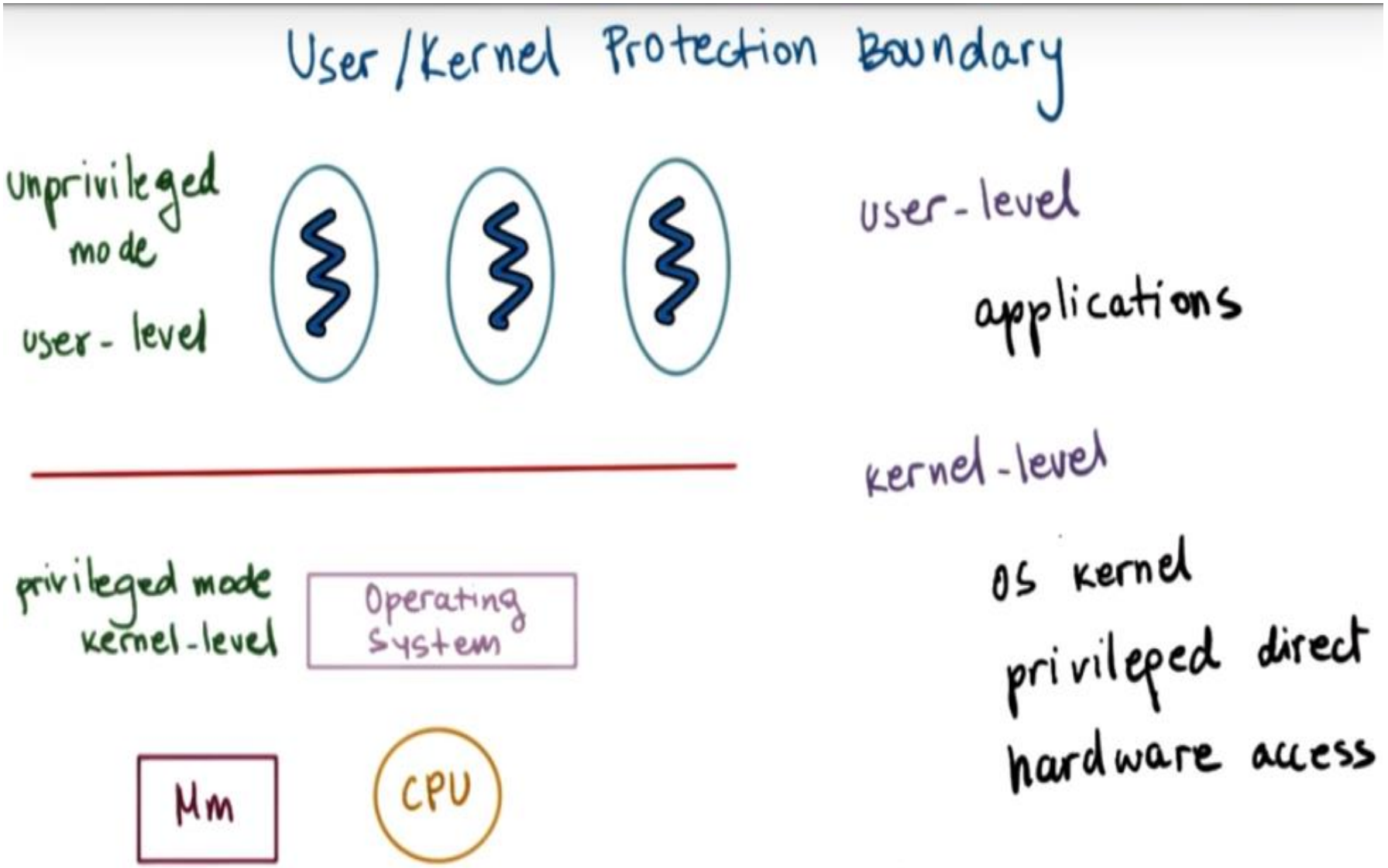
MULTIPROCESSOR

MULTICORE	MULTIPROCESSOR
A single CPU or processor with two or more independent processing units called cores that are capable of reading and executing program instructions	A system with two or more CPUs that allows simultaneous processing of programs
Executes a single program faster	Executes multiple programs faster
Not as reliable as a multiprocessor	More reliable since failure in one CPU will not affect the other
Have less traffic	Have more traffic

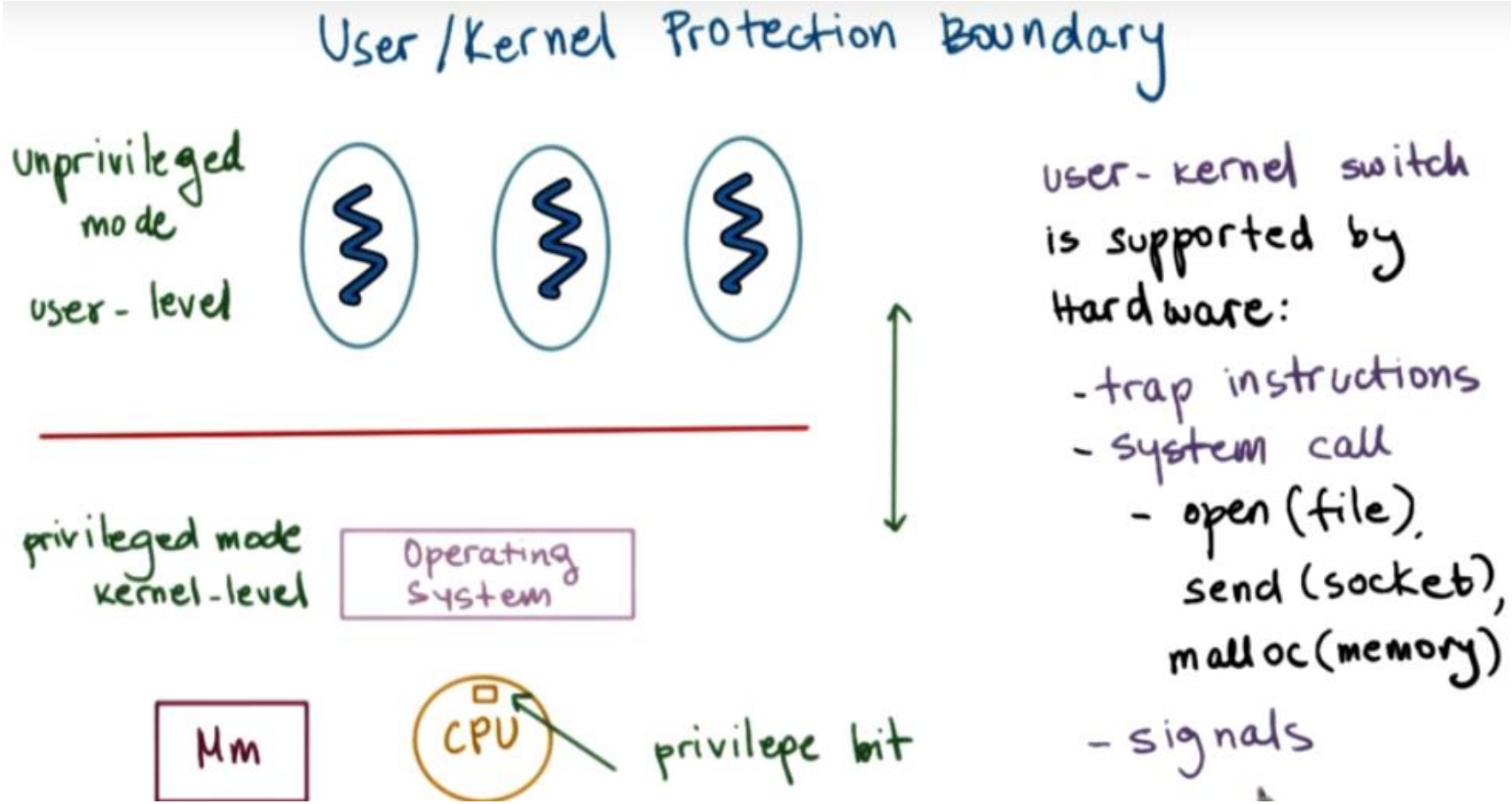
Operating-System Operations

- **To protect OS, Dual-mode** operations exist:
  - **User mode (1)** and **kernel mode (0)**
  - A **Mode bit is added to** hardware to indicate mode
    - Provides ability to distinguish when system is running user mode or kernel mode
    - System call changes mode to kernel, return from call resets it to user

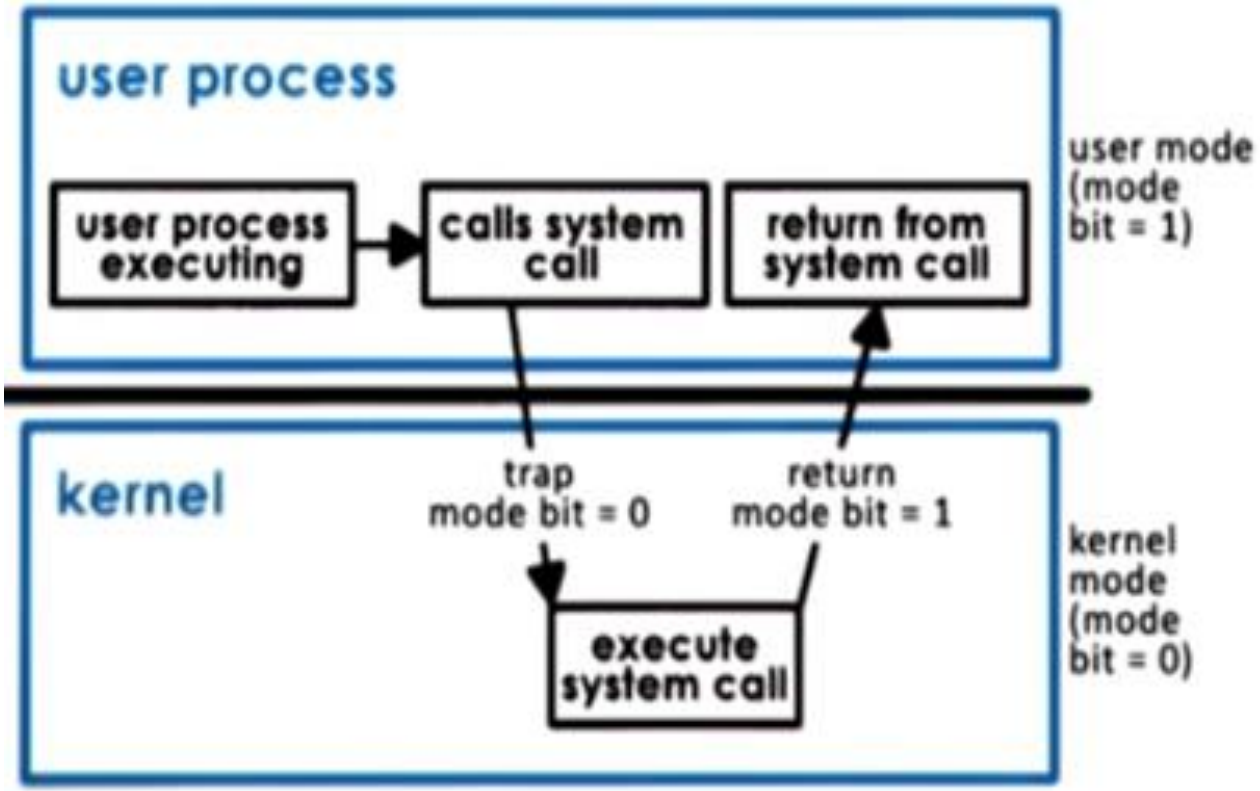
User/Kernel Protection Boundary



User/Kernel Protection Boundary



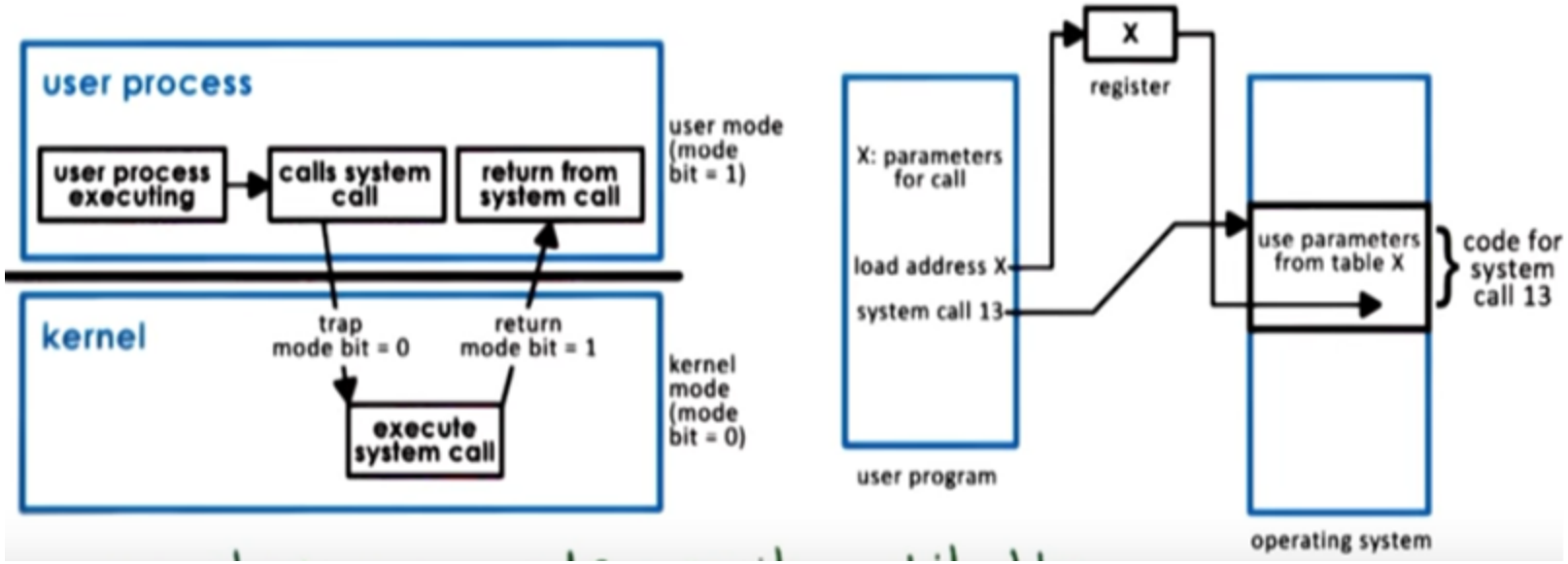
Transition from User to Kernel Mode



To make a system call an application must

- write argoments
- save relevant data at well-defined location
- make system call

Transition from User to Kernel Mode

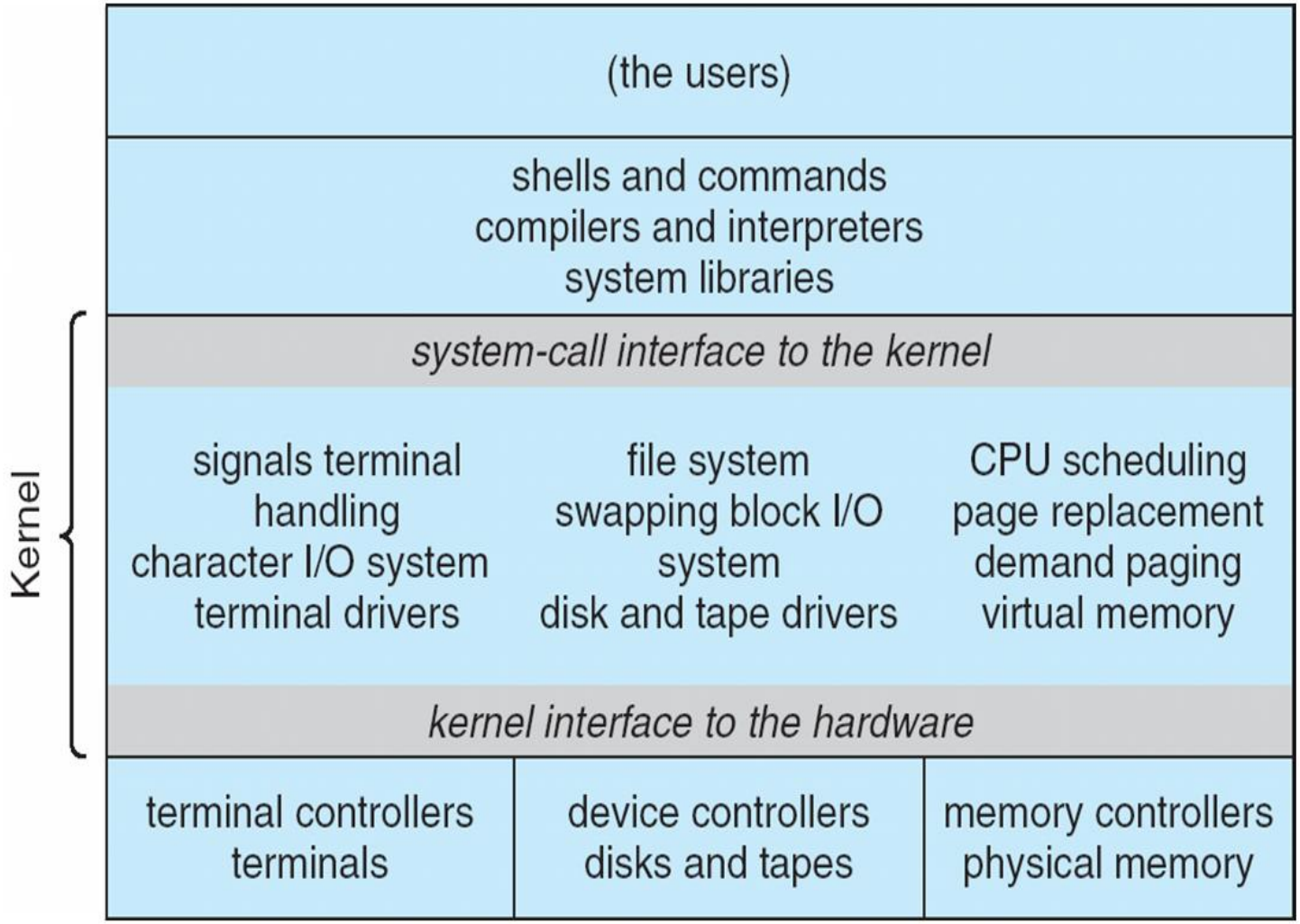


Non Simple Structure -- UNIX

The UNIX OS consists of two parts:

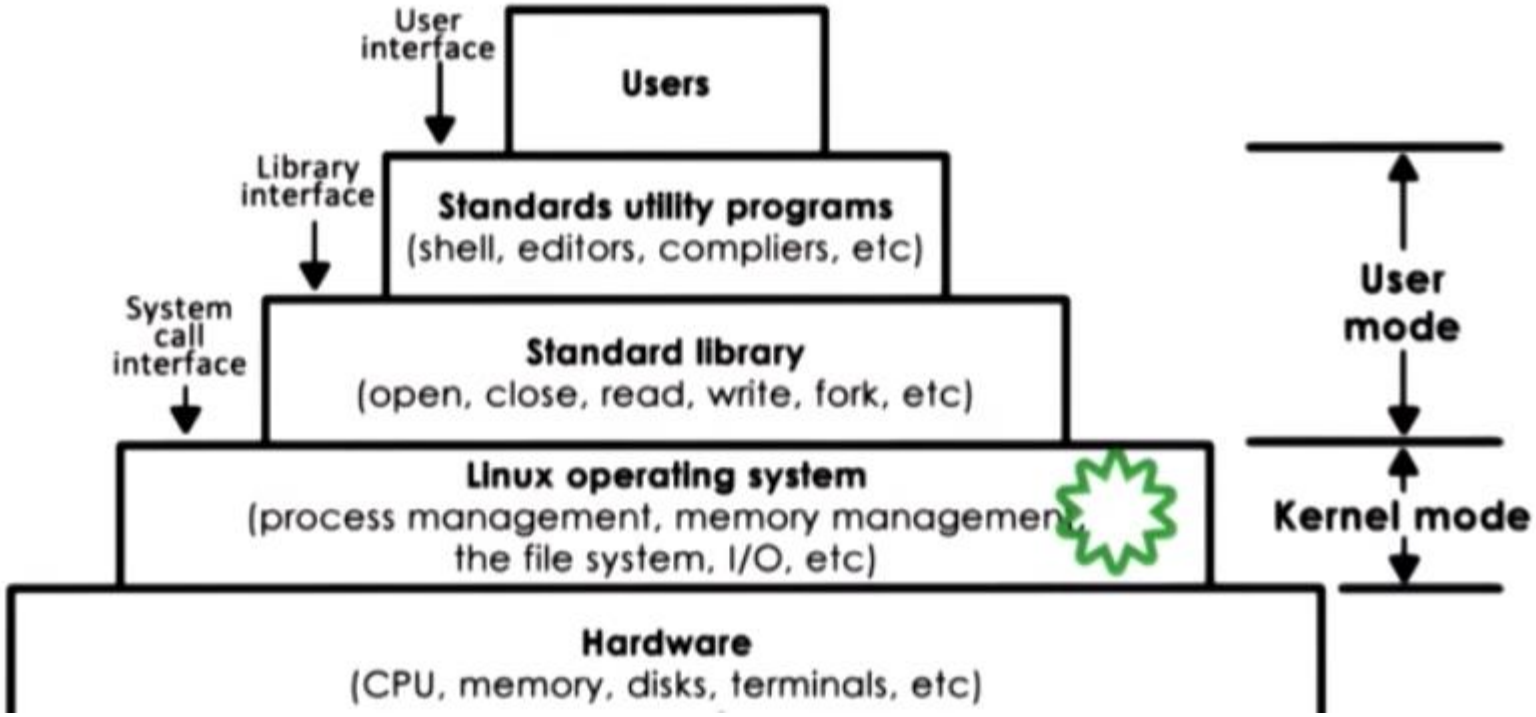
- System programs
- The kernel
  - Consists of everything below the system-call interface and above the physical hardware
  - Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level

Traditional UNIX System Structure

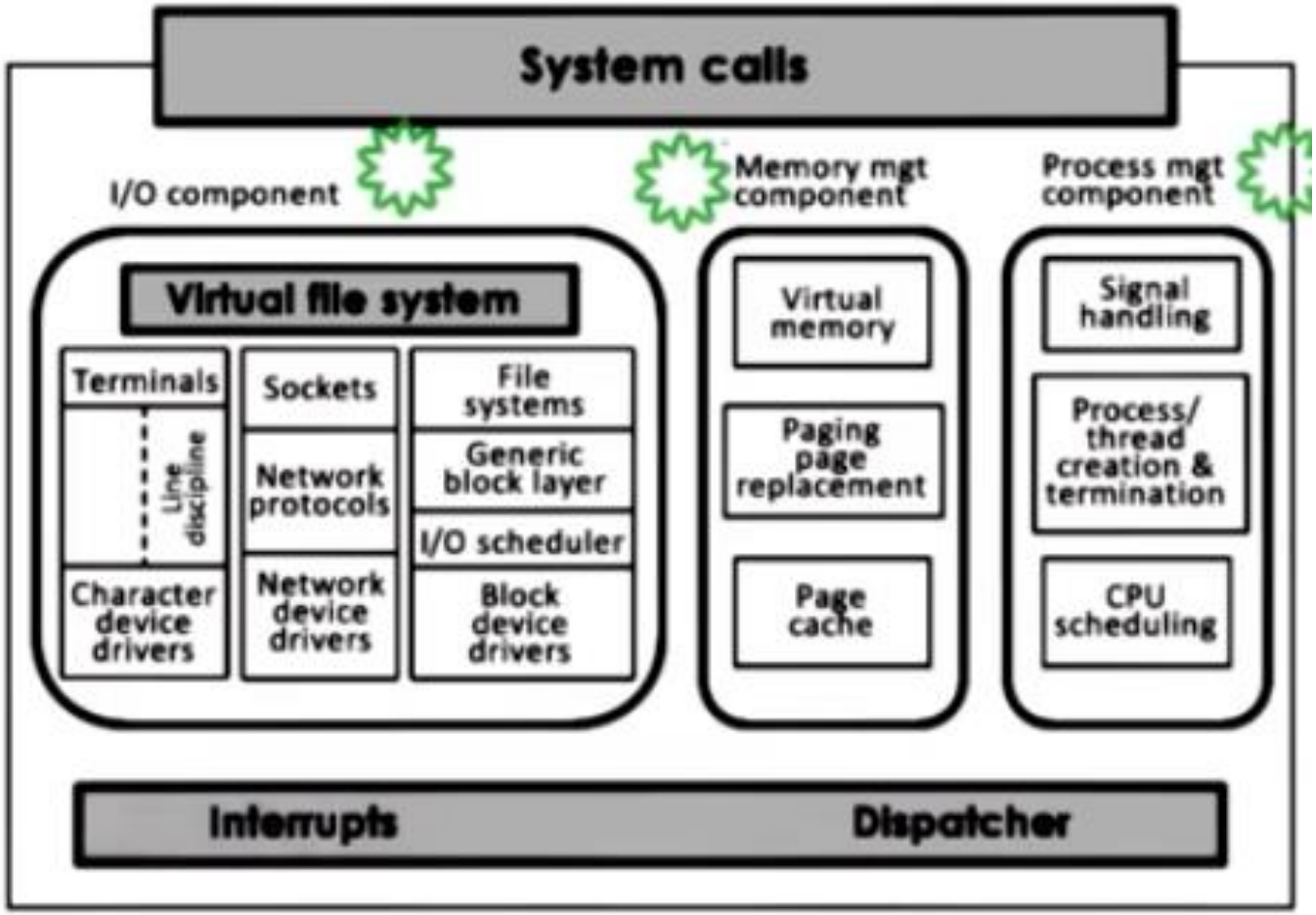


Traditional LINUX System Structure

Linux Architecture

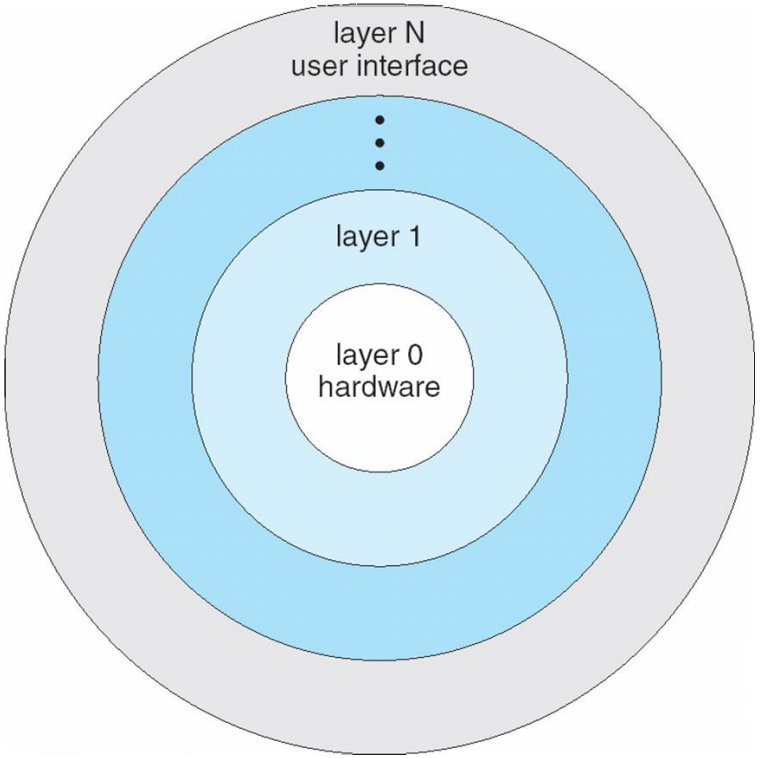


Kernel has many inbuilt modules



Layered Approach

- The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers



Kernel

- A kernel is a central component of an operating system.
- It acts as an interface between the user applications and the hardware.
- The sole aim of the kernel is to manage the communication between the software (user level applications) and the hardware (CPU, disk memory etc).
- The main tasks of the kernel are :
  - Process management
  - Device management
  - Memory management
  - Interrupt handling
  - I/O communication
  - File system...etc..

Kernel

- A kernel is the lowest level of software that interfaces with the hardware in your computer.

- It is responsible for interfacing all applications that are running in “user mode” down to the physical hardware, and allowing processes, to get information from each other using inter-process communication (IPC).

Kernel Types

kernels fall into one of three types:

- Monolithic
- Microkernel
- Hybrid

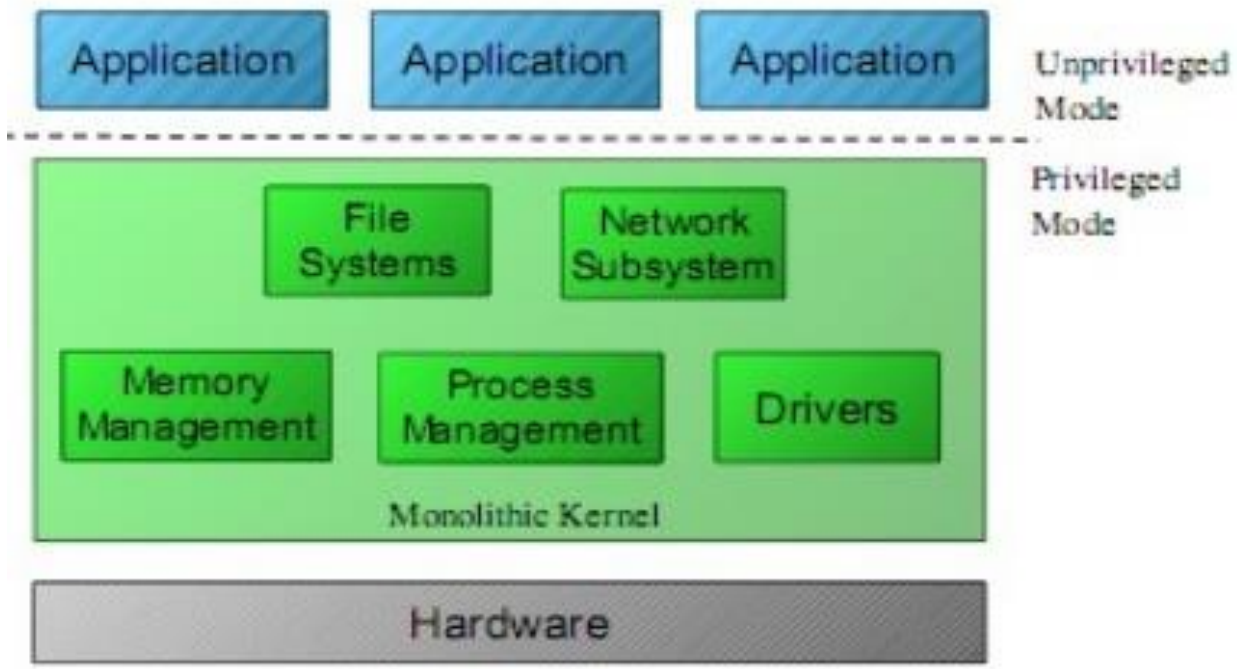
Monolithic Kernel

- A **monolithic kernel** is an operating system architecture where the entire operating system (which includes the device drivers, file system, and the application IPC etc.) is working in kernel space, in supervisor mode.

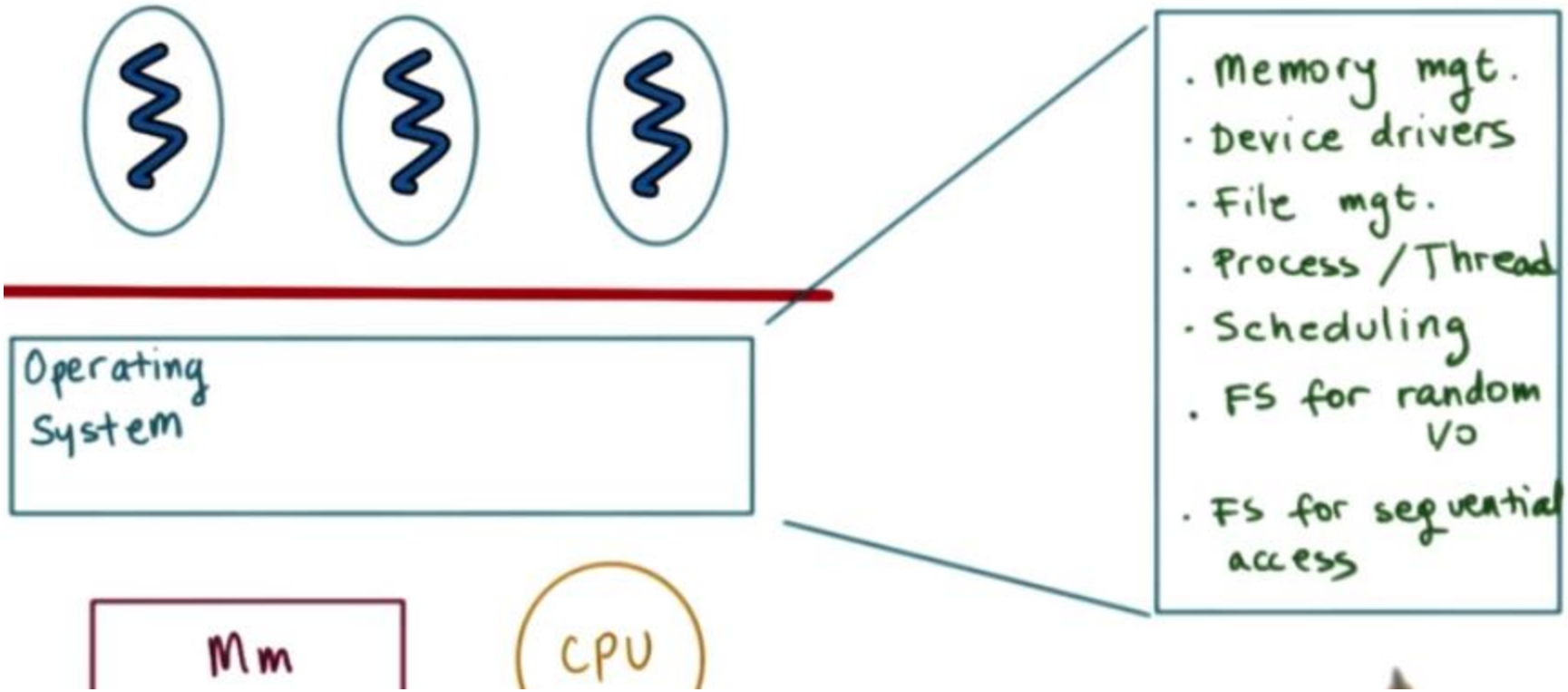
- Monolithic kernels are able to dynamically load (and unload) executable modules at runtime.

- Examples of operating systems that use a monolithic kernel are - Linux, Solaris, OS-9, DOS, Microsoft Windows (95,98,Me) etc.

Monolithic Kernel



Monolithic Kernel



Monolithic Kernel

Pros:

- More direct access to hardware for programs
- Easier for processes to communicate between each other
- If your device is supported, it should work with no additional installations
- Processes react faster because there isn't a queue for processor time.

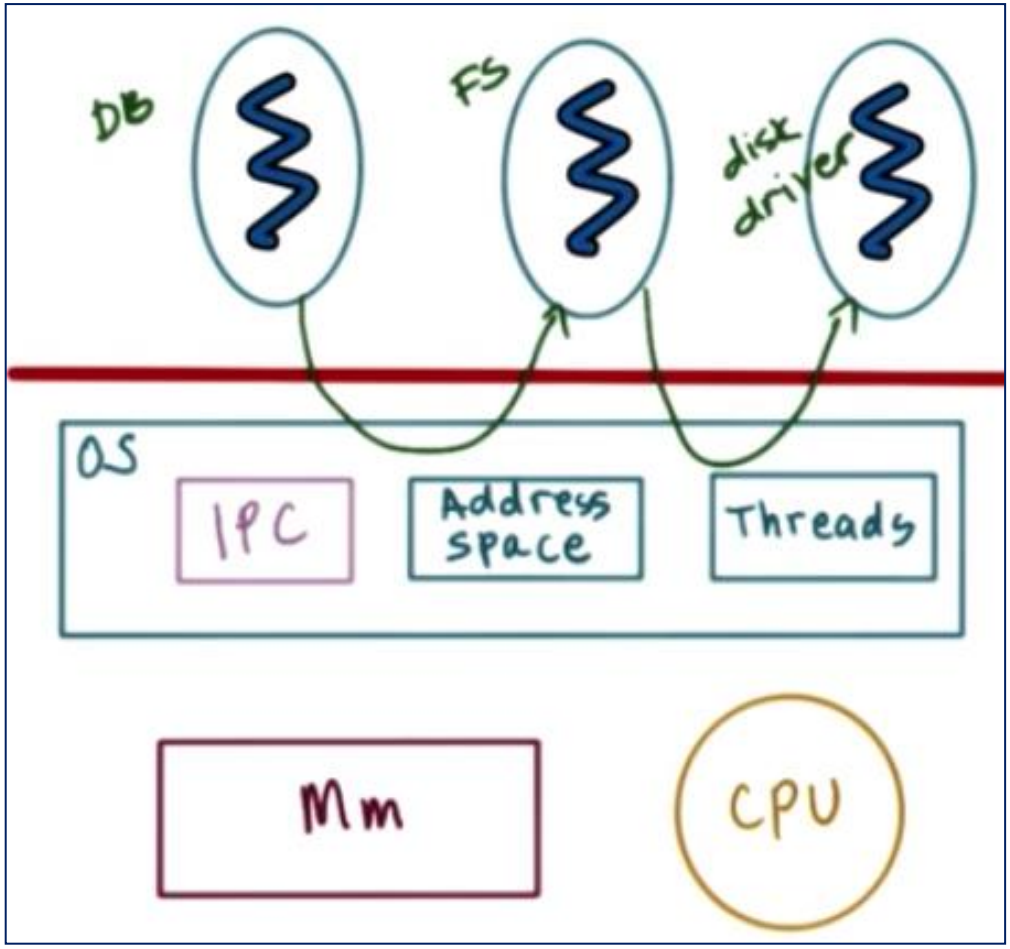
Cons:

- Large memory is needed
- Less secure because everything runs in supervisor mode

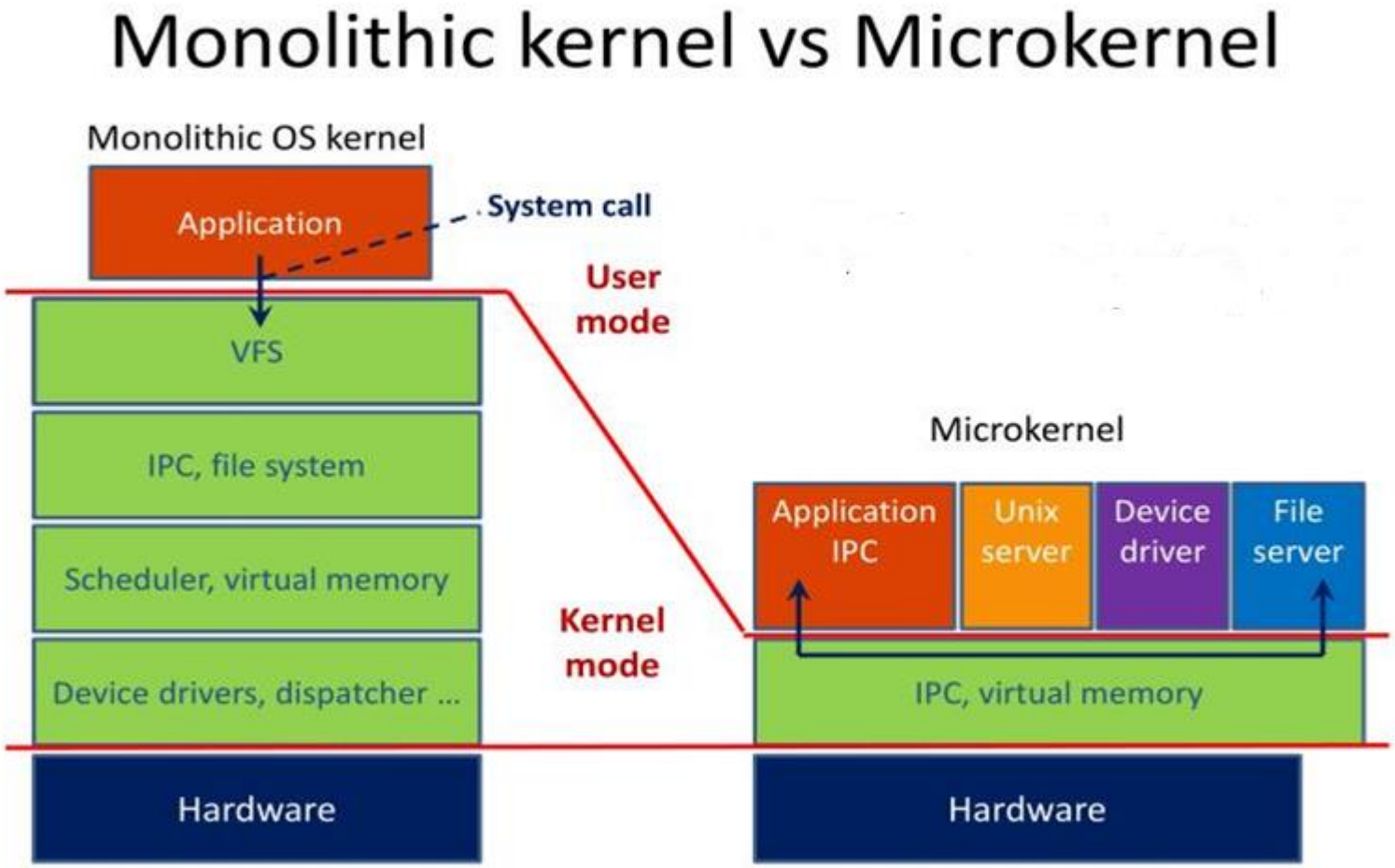
MicroKernel

- In a Microkernel architecture, the core functionality is isolated from system services and device drivers.
- This architecture allows some basic services like device driver management, file system etc. to run in user space.
- This reduces the kernel code size and also increases the security and stability of OS as we have minimum code running in kernel.
- Examples of operating systems that use a microkernel are - QNX, Integrity, PikeOS, Symbian, L4Linux, Singularity, K42, Mac OS X, HURD, Minix, and Coyotos.

MicroKernel



Types of Kernel



MicroKernel

Pros

- Portability
- Small install footprint
- Small memory
- Security

Cons

- Hardware may react slower because drivers are in user mode
- Processes have to wait in a queue to get information
- Processes can’t get access to other processes without waiting

HybridKernel

- Hybrid kernels have the ability to pick and choose what they want to run in user mode and what they want to run in supervisor mode.
- Device drivers and file system I/O will be run in user mode while IPC and server calls will be kept in the supervisor mode.
- This require more work of the hardware manufacturer because all of the driver responsibility is up to them.

Hybrid Kernel

Pros

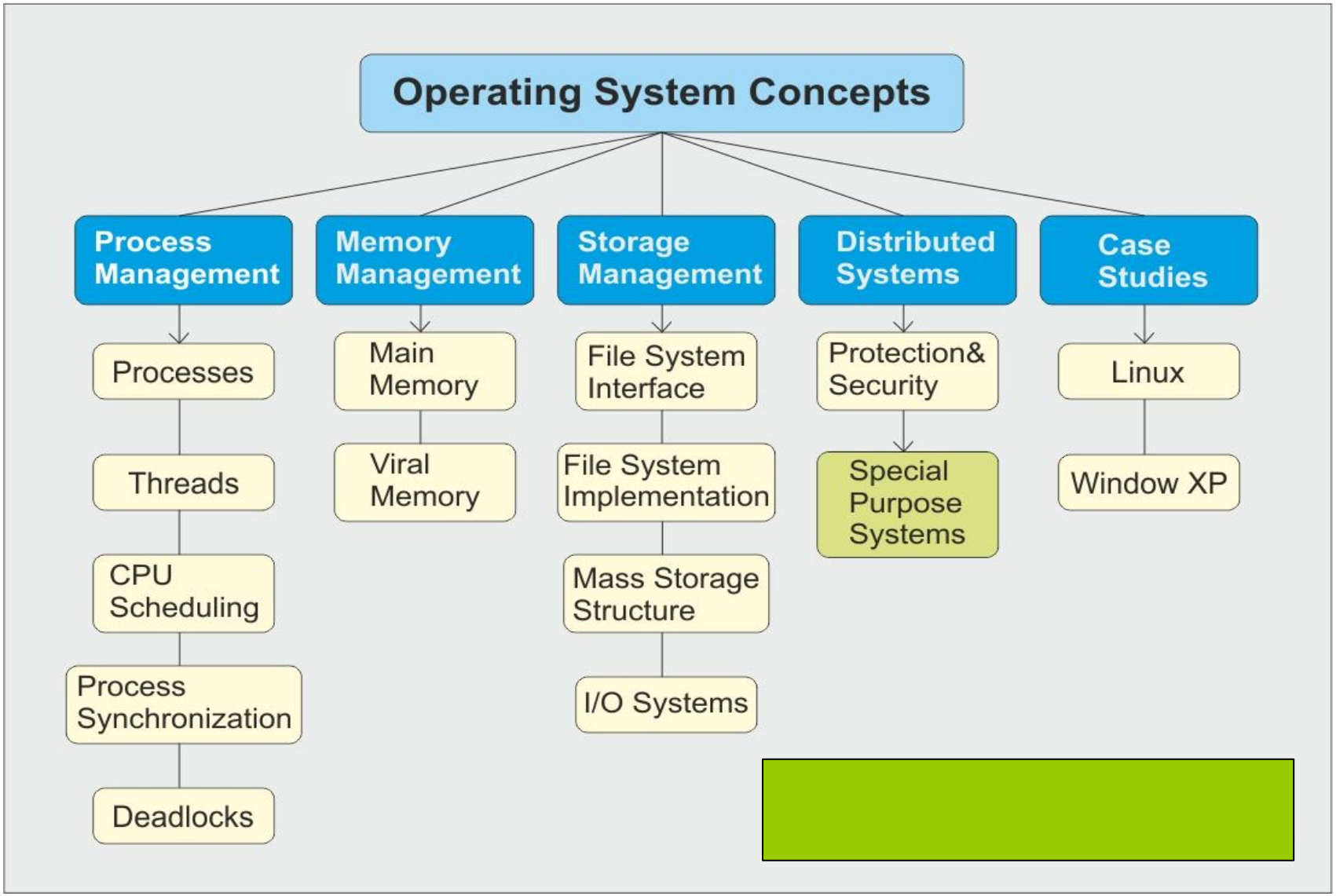
- Developer can pick and choose what runs in user mode and what runs in supervisor mode
- More flexible than other models

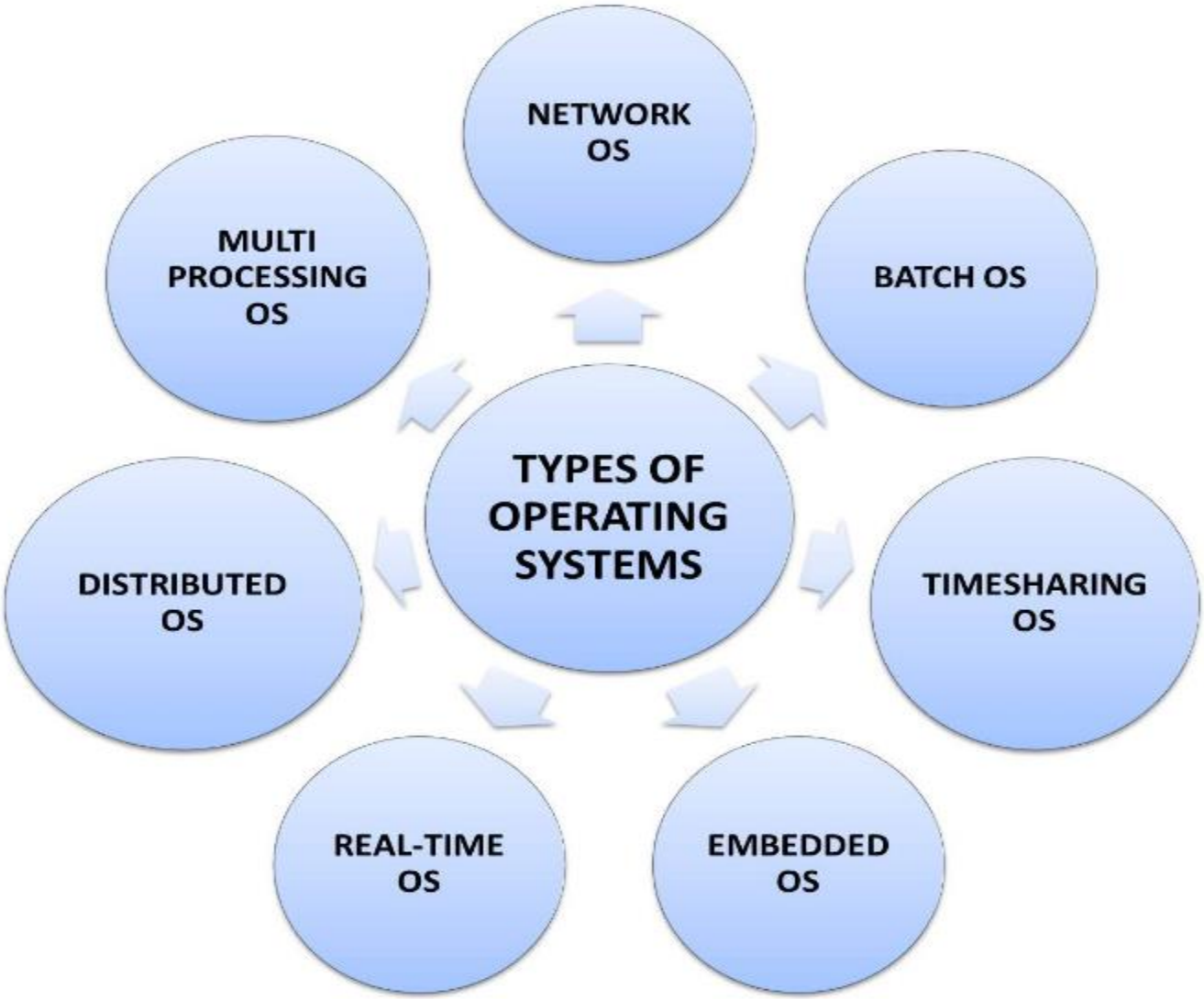
Cons

- Processes have to wait in a queue to get information
- Processes can’t get access to other processes without waiting
- Device drivers need to be managed by user

History of Operating system

Generation	Year	Electronic device used	Types of OS Devices
First	1945-55	Vacuum Tubes	Plug Boards
Second	1955-65	Transistors	Batch Systems
Third	1965-80	Integrated Circuits(IC)	Multiprogramming
Fourth	Since 1980	Large Scale Integration	PC


Operating System Concepts
Process Management
Memory Management
Storage Management
Distributed Systems
Case Studies
Processes
Threads
CPU Scheduling
Process Synchronization
Deadlocks
Main Memory
Viral Memory
File System Interface
File System Implementation
Mass Storage Structure
I/O Systems
Protection& Security
Special Purpose Systems
Linux
Window XP


TYPES OF OPERATING SYSTEMS
NETWORK OS
BATCH OS
TIMESHARING OS
EMBEDDED OS
REAL-TIME OS
DISTRIBUTED OS
MULTI PROCESSING OS

Interrupts

- An interrupt is a signal from a device attached to a computer or from a program within the computer that causes the main program that operates the computer (the operating system) to stop and figure out what to do next.

- Interrupts can be of following type:
  - Generated by Hardware (Hardware Interrupt)
  - Generated by Software (Software Interrupt)

Hardware Interrupt

1. Hardware interrupts are used by **devices** to communicate that they **require attention from the operating system**.
2. Hardware interrupts by sending signal to CPU **via system bus.**
3. Hardware interrupts are referenced by an ***interrupt number*.**
4. These numbers are mapped with hardware that created the interrupt. This enables the system to monitor/understand **which device created the interrupt and when it occurred**.

Software Interrupt/ Trap

- Interrupt generated by executing a instruction.
- Software interrupts by a special operation called a System Call or Monitor Call.

Exp: 1. cout in C++ is a kind of interrupt because it would make a system to print something.

2. division by zero

Basic steps when interrupt occurs:

1. Interrupt Occurred?
2. H/w Transfers control to OS
3. OS preserves current state of process by using Registers and Program counter
4. Determine which kind of interrupt has occurred and Provides resources
5. When interrupt is executed, address is loaded to program counter and interrupted services are resumed.

What is a System Call?

- A system call is a method for a computer program to request a service from the kernel of the operating system on which it is running.
- A system call is a method of interacting with the operating system via programs.
- A system call is a request from computer software to an operating system's kernel.

System Calls

- Allow user-level processes to request services of the operating system.
- It provides a way in which program talks to the operating system.

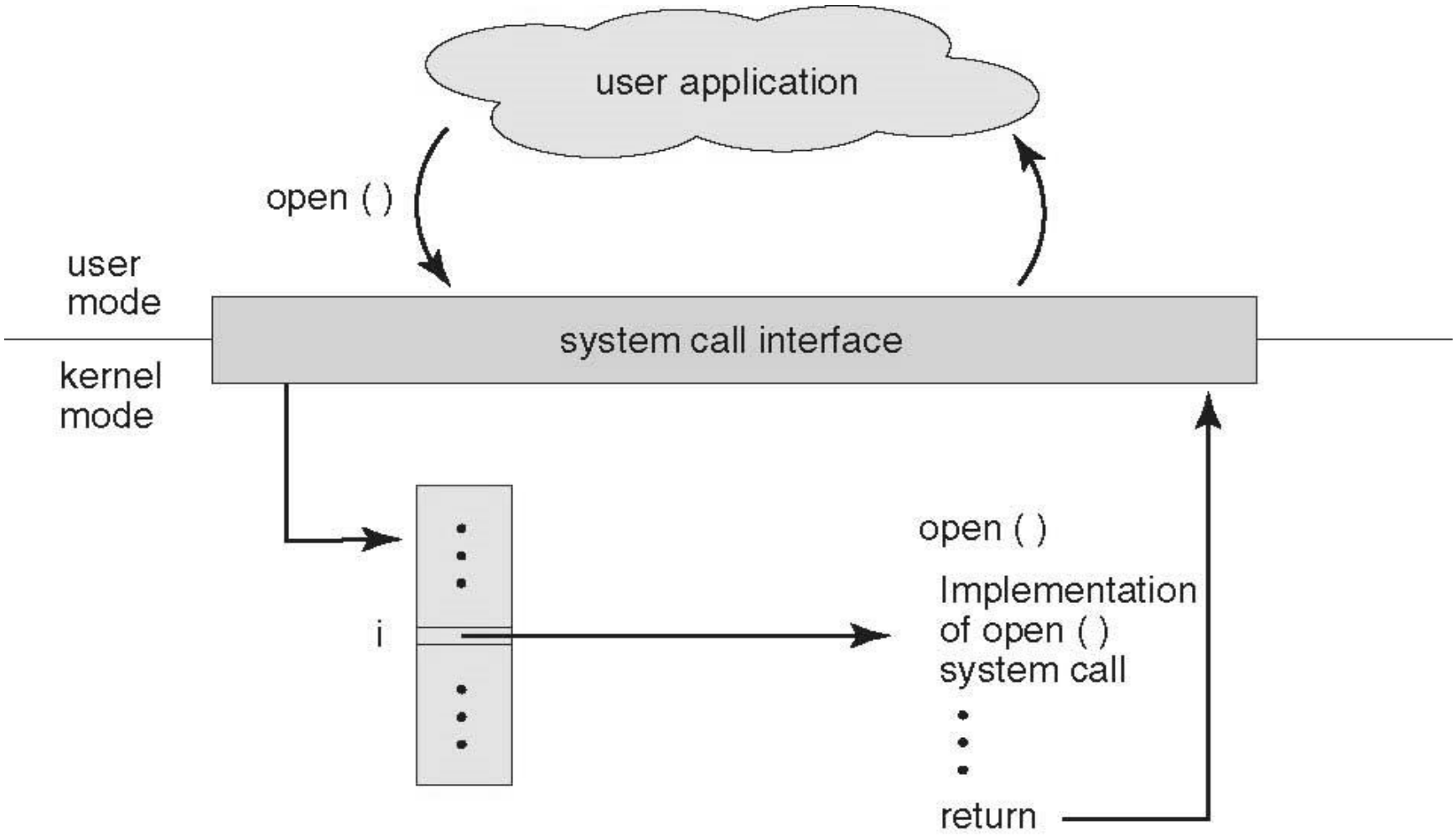
Why system calls are required?

- **It is a request to the operating system to perform some activity**.
- It is a **call to the kernel** in order to **execute a specific function** that controls a device or executes a instruction.
- A system call looks like a procedure call

System Call Implementation

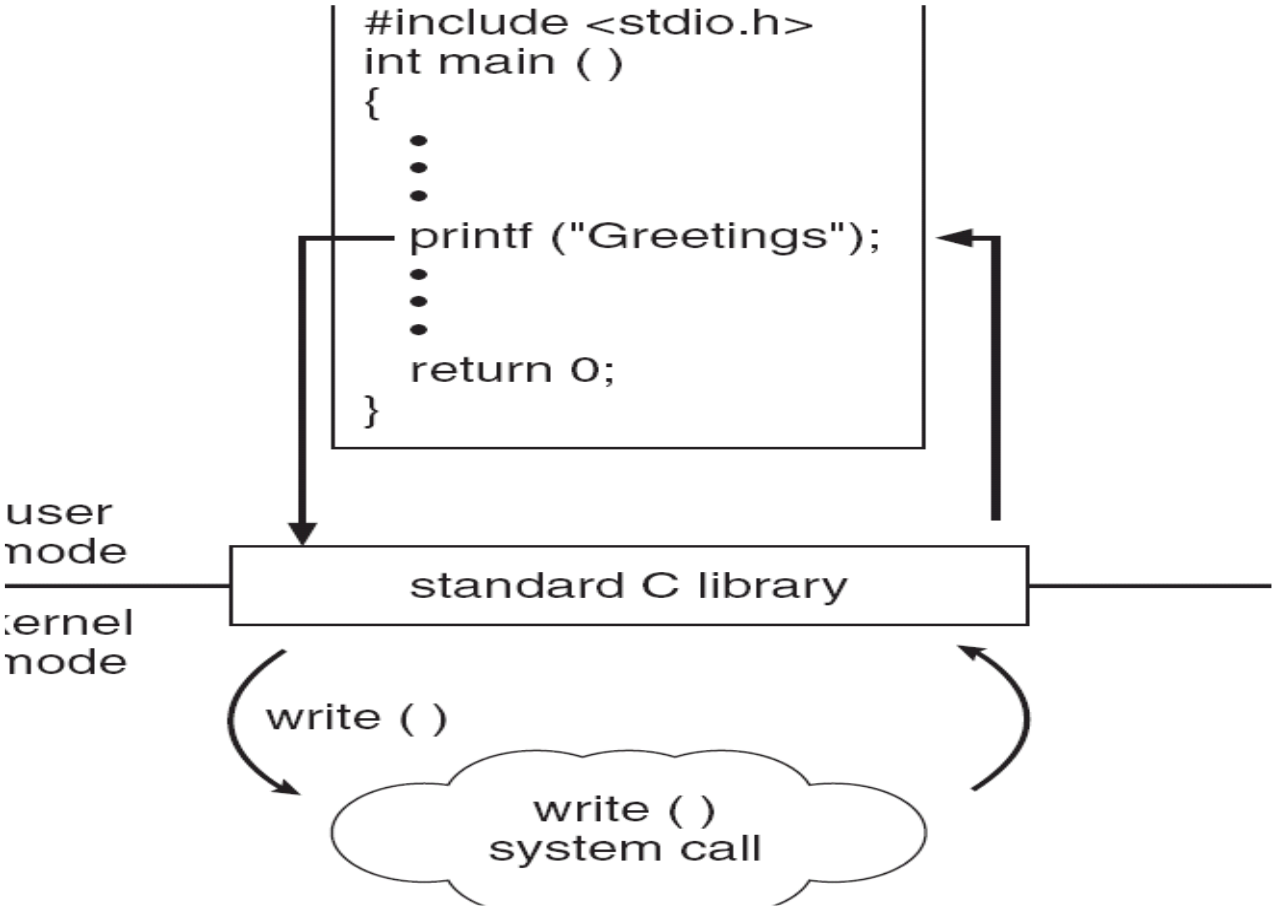
- A number is associated with each system call
  - System-call interface maintains a table indexed according to these numbers
- The system call interface invokes intended system call in OS kernel and returns status of the system call with a return value.
- The caller needs to know nothing about how the system call is implemented
  - Just needs to obey API and understand what OS will do as a result call
  - Most details of OS interface hidden from programmer by API

API – System Call – OS Relationship



Standard C Library Example

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



System Call Dispatch

1. Kernel assigns system call number
2. Kernel initializes system call table, mapping system call number to functions implementing the system call
3. User process sets up system call number and arguments
4. User process runs

System Call Dispatch

5. Hardware switches to kernel mode and calls kernel's interrupt handler for user process (interrupt dispatch)
6. Kernel looks up syscall table using system call number
7. Kernel calls/invokes the corresponding function
8. Kernel returns by running interrupt return and processing the actual program.

System Call Parameter Passing

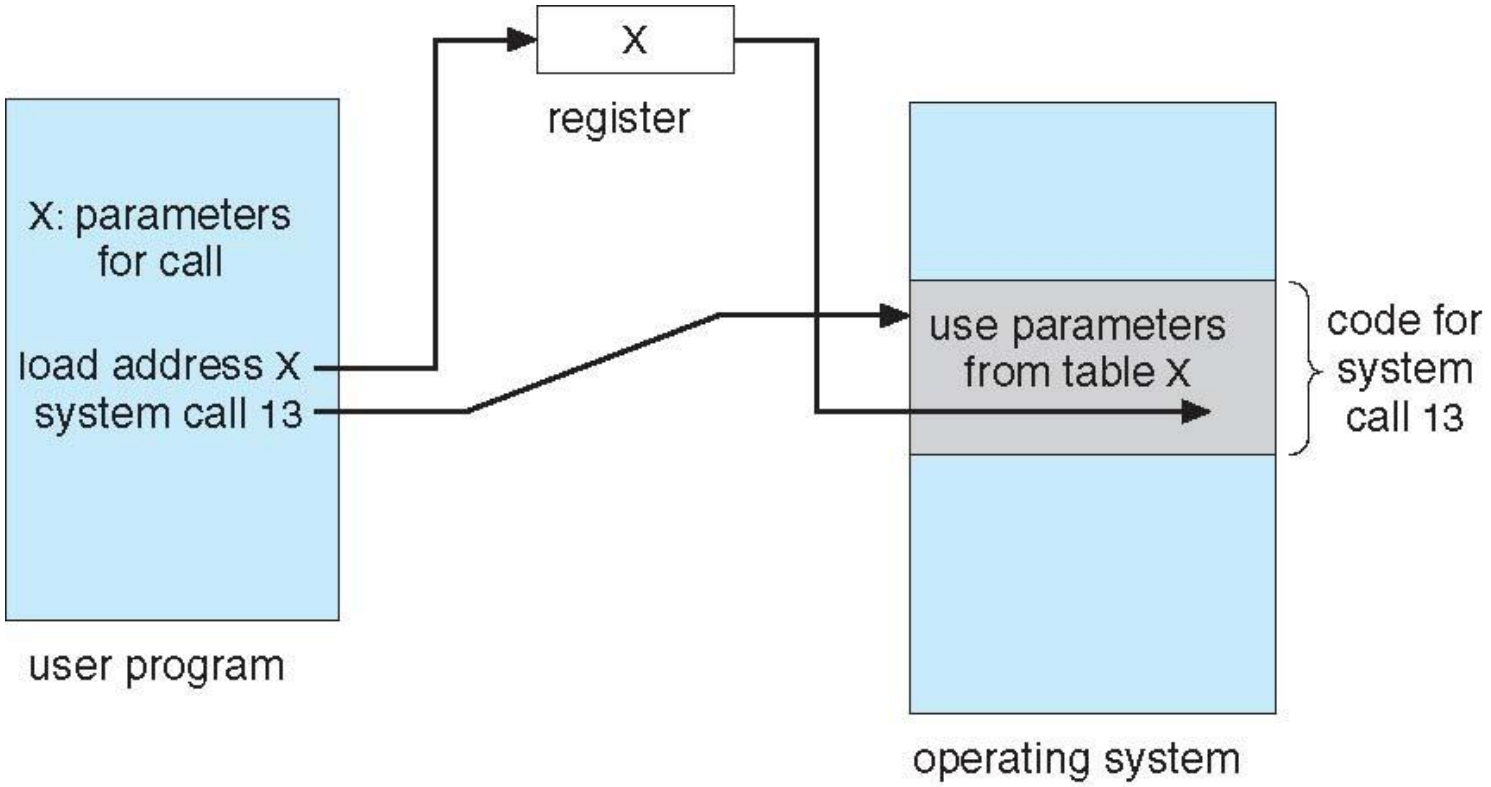
Passing Parameters to System Calls:

- Information required for a system call vary according to OS and call.

- **Three general methods used to pass parameters to the OS**

  1. **Pass the parameters in *registers***
     - When parameters are < 6.
  2. **Parameters stored in a *block,* or table**, in memory, and address of block passed as a parameter in a register. (6 or more)
     - This approach taken by Linux and Solaris
  3. **Parameters placed**, or *pushed,* **onto the *stack*** by the program and *popped* off the stack by the operating system.

Parameter Passing via Table



Types of System Calls

5 Categories

- Process Control
  - end, abort
  - load, execute
  - create process, terminate process
  - Fork system call
  - get process attributes, set process attributes
  - wait for time
  - wait event, signal event
  - allocate and free memory

- File Management
  - create file, delete file
  - open, close file
  - read, write, reposition
  - get and set file attributes

Types of System Calls (Cont.)

- Device Management
  - request device, release device
  - read, write, reposition
  - get device attributes, set device attributes
  - logically attach or detach devices
- Information Maintenance
  - get time or date, set time or date
  - get system data, set system data
  - get and set process, file, or device attributes

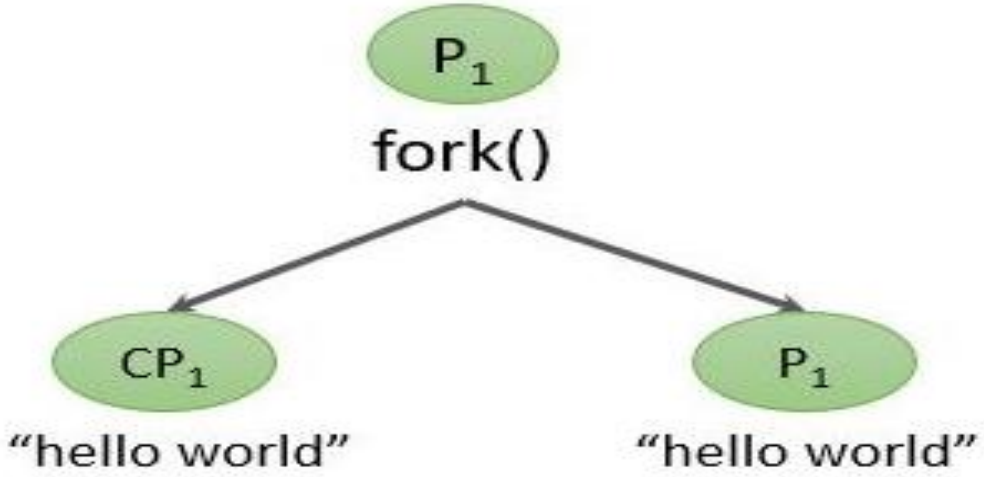
- Communications
  - create, delete communication connection
  - send, receive messages
  - transfer status information
  - attach and detach remote devices

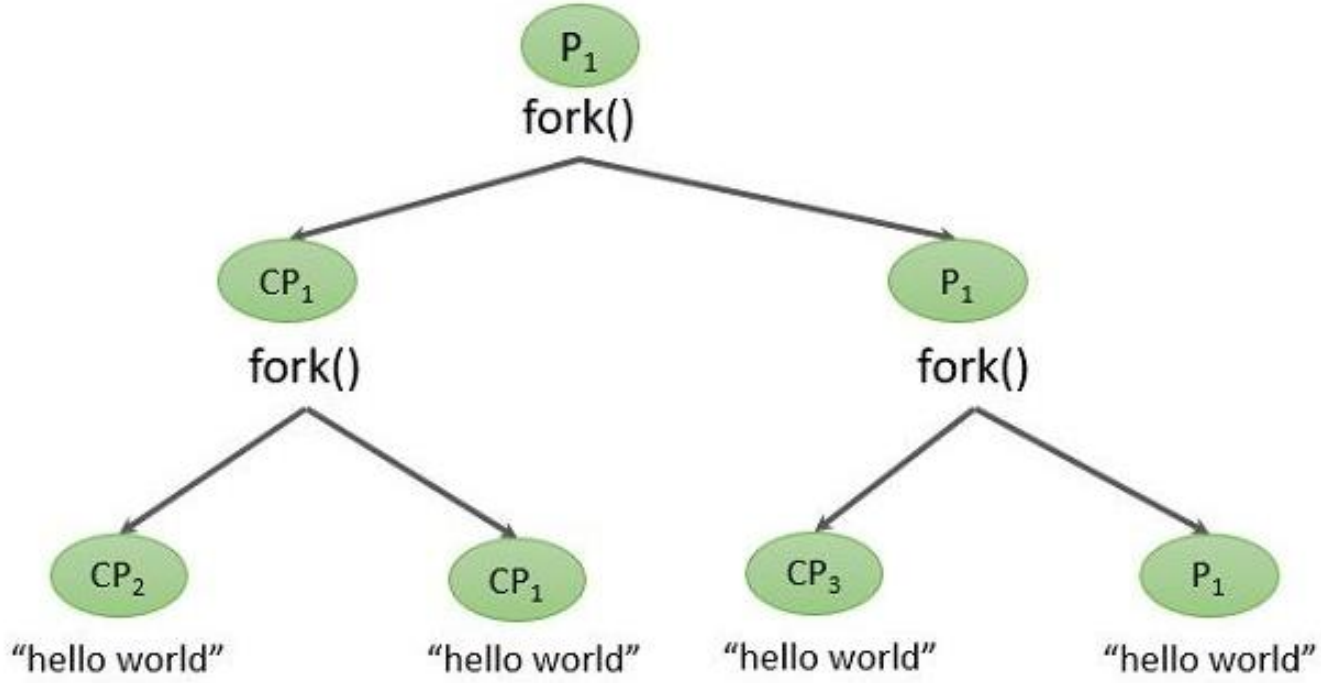
(Imp)Fork system call

- Fork system call is used for creating a new process
- which is called ***child process***
- which runs concurrently with the process that makes the fork() call (parent process)

Main Aim –

- To achieve multiprocessing




P1
fork()
CP1
P1
fork()
fork()
CP2
CP1
CP3
P1
"hello world"
"hello world"
"hello world"
"hello world"

Q1. Who controls the execution of programs to prevent errors and improper use of computer?

a) Resource allocator

b) Control Program

c) Hardware

d) None of the above

Q2. The operating system switches from user mode to kernel mode so the mode bit will change from?

a) 0 to 1

b) 1 to 0

c) Remain constant

d) None

Q3. In which type of operating system users do not interact directly with the computer system?

a) Multiprogramming operating systems

b) Multiprocessing operating systems

c) Batch operating systems

d) Distributed operating systems

Ans 3) c

Q4. What is the objective of multiprogramming operating systems?

a) Maximize CPU utilization

b) Switch the CPU among processes

c) Achieve multitasking

d) None of the above

Ans4. a)

Q5. Who signalled for the occurrence of an event either from the hardware or the software?

a) Bootstrap program

b) Interrupt

c) Disk Controller

d) CPU

Ans5: b

Q6. In which type of I/O interrupts the control return to the user program after the completion of I/O operation?

a) Synchronous I/O interrupts

b) Asynchronous I/O interrupts

c) System Call

d) Hardware

Ans 6) a

Q7: The device-status table contains

a) each I/O device type

b) each I/O device state

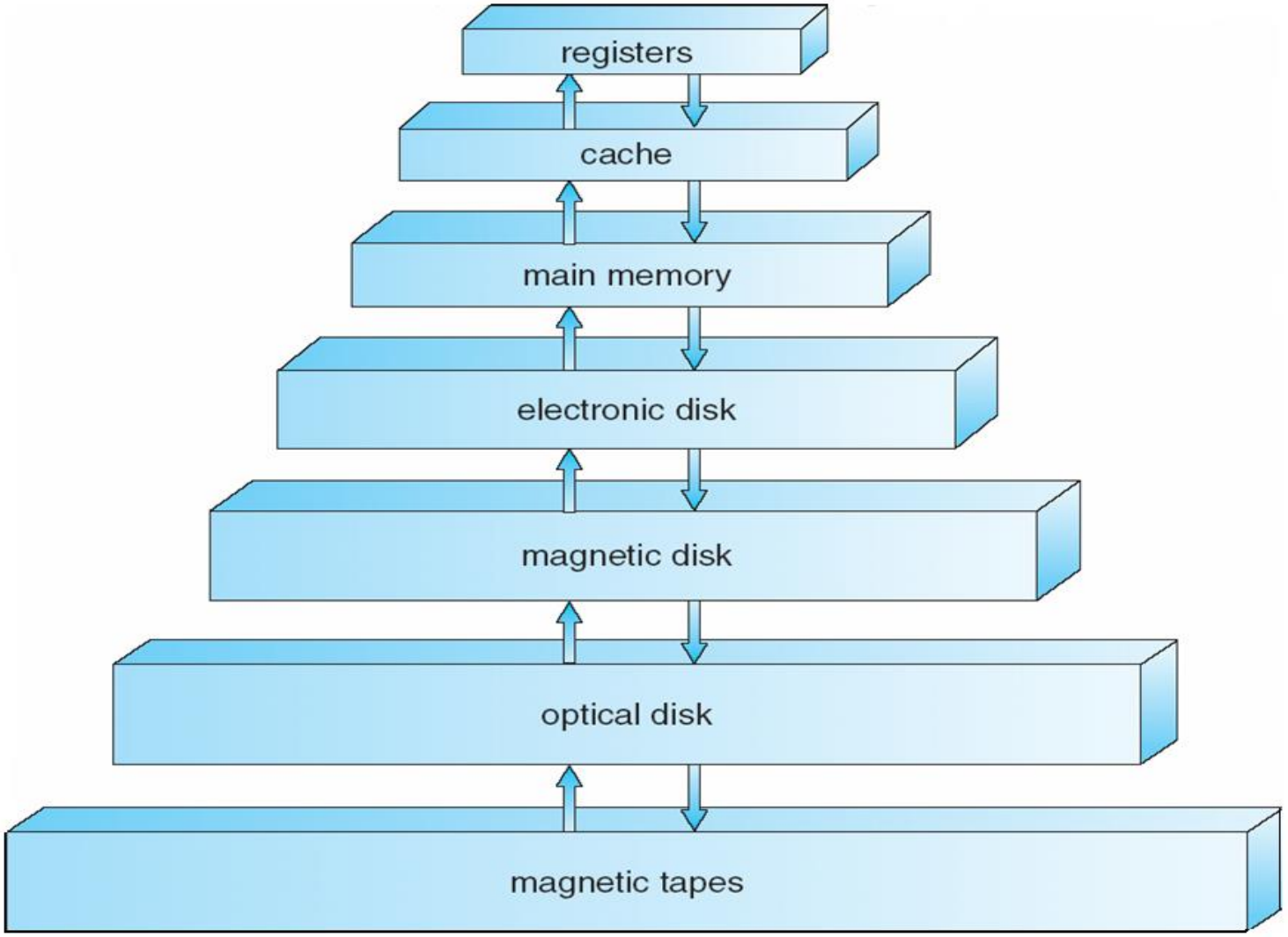
c) each I/O device address

d) all of the above

(d)

Ans 7 (d)

Storage Structure and Hierarchy



Magnetic Tape

Optical Tape

Main Memory (RAM)

Magnetic Disks

- A read/write head travels across a spinning magnetic disk, retrieving or recording data
- Each disk surface is divided into sectors and tracks
- Example of disk addressing scheme: surface 3, sector 5, track 4