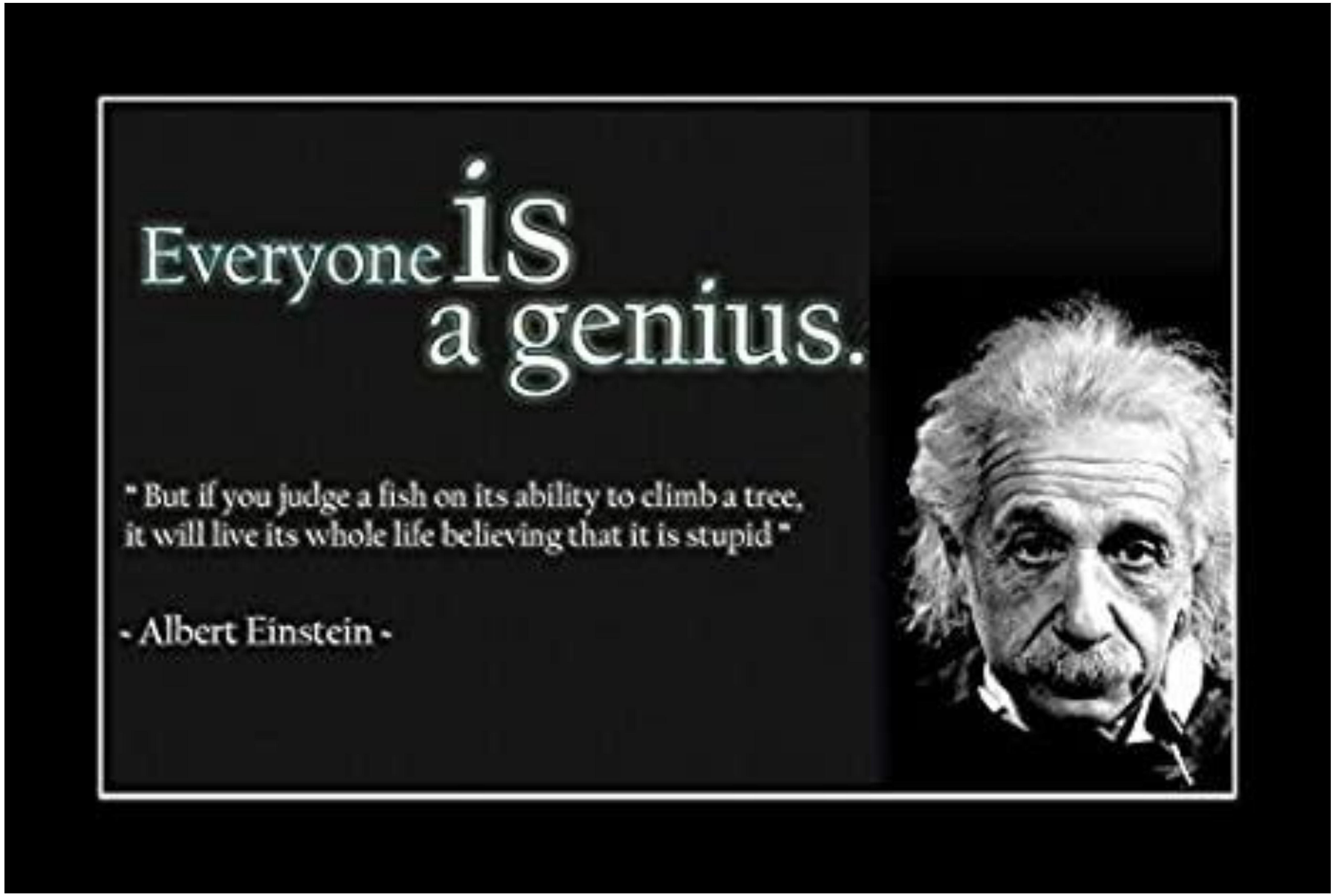
Operating System

BIM

Er. Santosh Bhandari,
(Master Computer Science)

Motivation


Everyone is a genius.
" But if you judge a fish on its ability to climb a tree, it will live its whole life believing that it is stupid "
- Albert Einstein -

1. Operating system principles, components, and usage

Vocabulary

Throughput:
Context Switching:
Time Quantum:
Modularity:
Abstraction:
Layring:
response time:
Efficiency:
Overhead:

Introduction

Computer System Components

1. Hardware: Provides basic computing resources (CPU, memory, I/O devices).

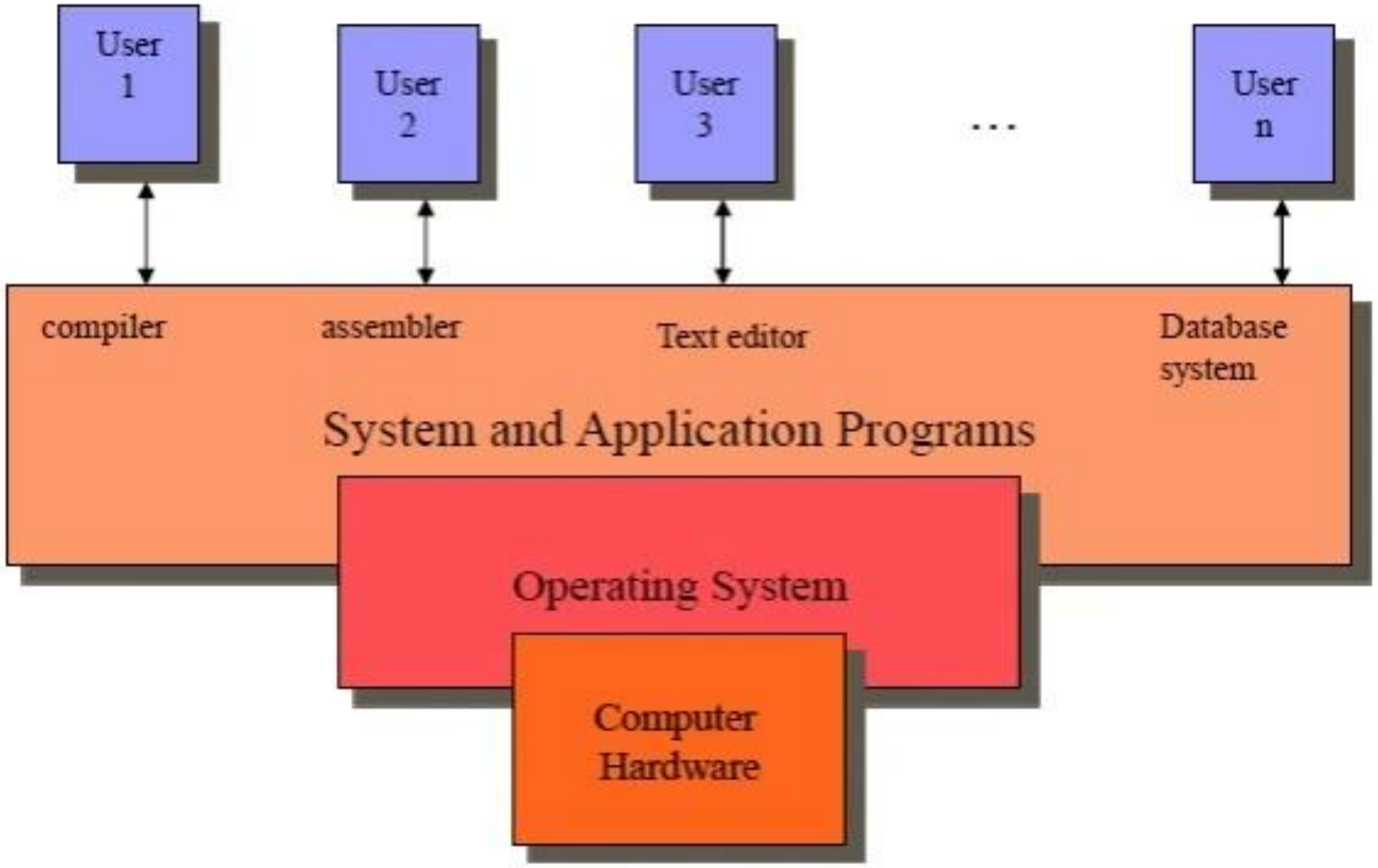
2. Software:

a. System software: Operating System: Controls and coordinates the use of hardware among application programs.

b. Application software: Application Programs: Solve computing problems of users (database systems, video games, banking software).

3. Users: People, machines, other computers

Abstract View of System



What is OS?

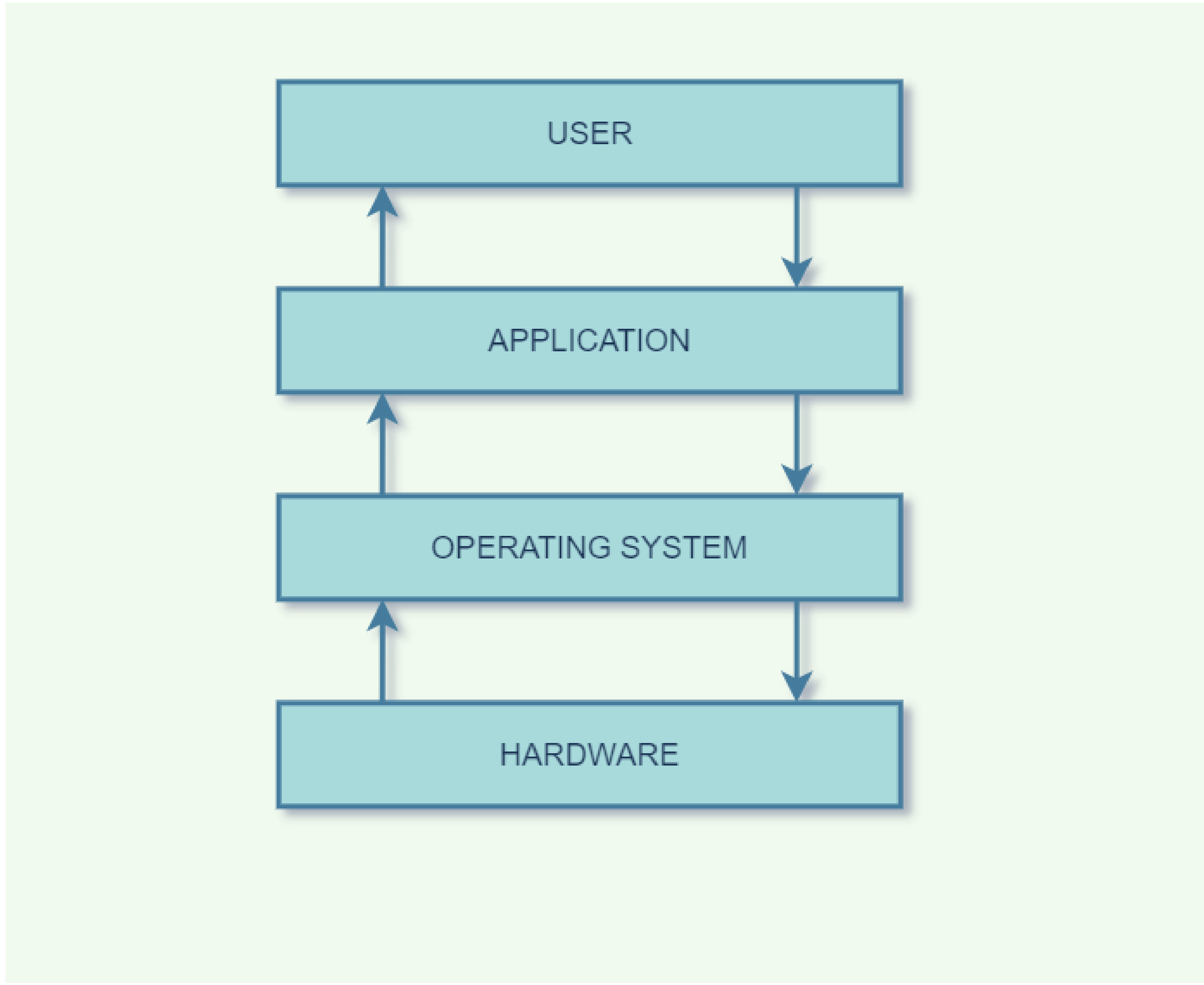
-It is a program that acts as an **interface** between the user and the computer hardware.

-It **controls the execution** of all kinds of programs.

-Manages computer hardware resources

-Operating System lies in the category of **system software**.

-Regulates hardware and software programs



OS Goals

Primary: Convenience, Easy to use

Secondary: Efficiency, Reliability, Management

Summarized:

- Use computer hardware efficiently.
- Allow sharing of hardware and software resources.
- Make application software portable and versatile.
- Provide isolation, security, and protection among user programs.

OS Functions

-Booting: Start computer

-Memory Management: Allocates and manages primary memory or RAM

-Process Management (CPU Scheduling): Manages the process lifecycle, including creation, execution, and termination.

-**Error Detection:** Finds the program errors.

-File System Management: Controls how data is stored and retrieved.

-Security: Protects system data and resources from unauthorized access.

-I/O device management: Monitors and controls peripheral devices.

OS Roles

How can an operating system run multiple applications?

To achieve this, operating systems need to play three roles:

1. Referee: OS manages resources shared between applications running on the same machine, OS isolates applications from each other.

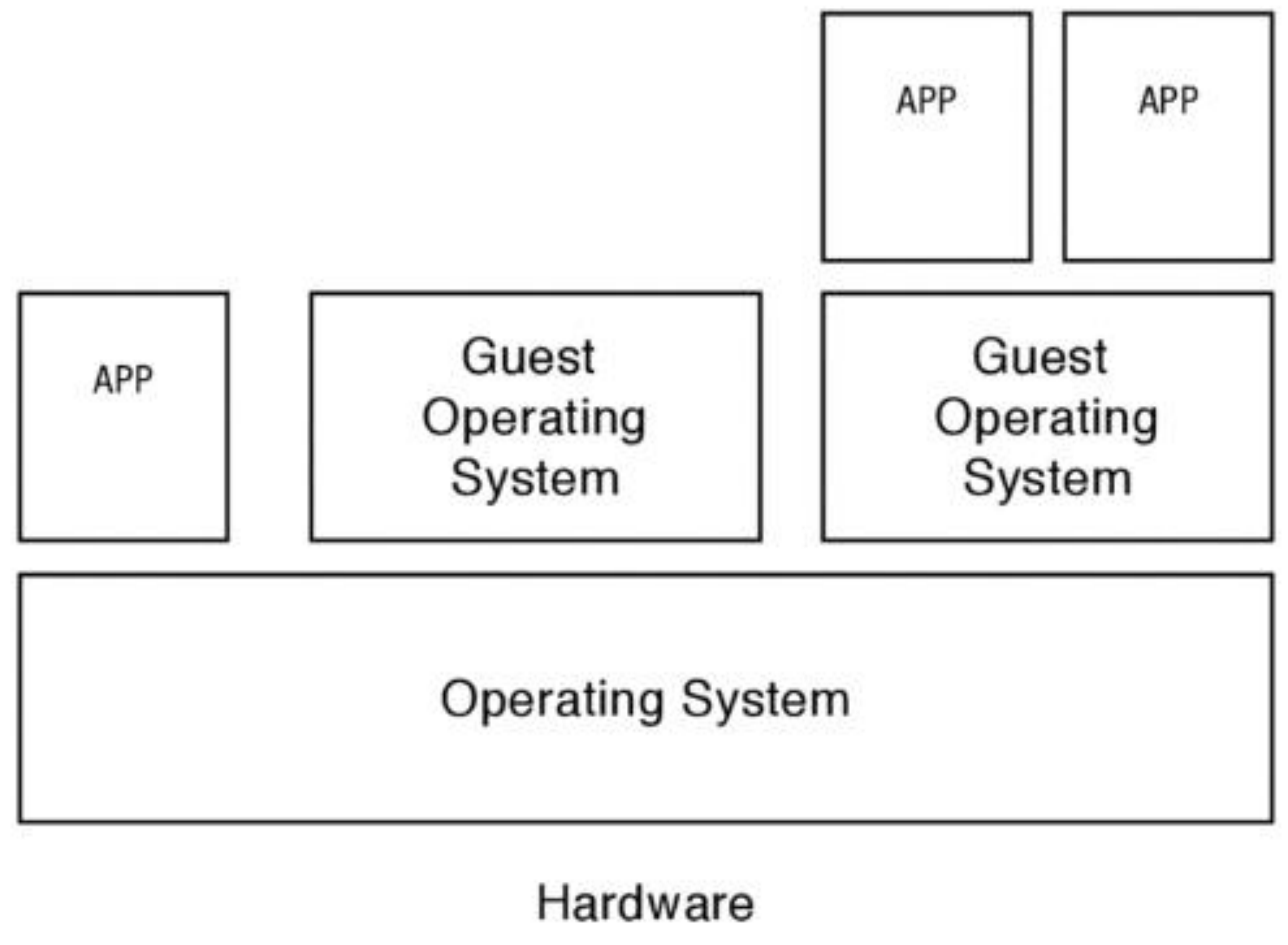
Challenges: An error in one application should not disrupt other applications, or even the OS itself. This is called **fault isolation**

Example: One user should not be allowed to monopolize system resources or to access or corrupt another user's files without permission; a buggy application should not be able to crash the operating system.

OS Roles

2. Illusionist: Operating systems provide an abstraction of physical hardware to simplify application design.

-Virtualization provides an application with the illusion of resources that are not physically present.



Example: The operating system running in the virtual machine, called the guest operating system, thinks it is running on a real, physical machine, but this is an illusion.

OS Roles

3. Glue. Operating systems provide a set of common services that facilitate sharing among applications.

-Libraries, Widgets

Example: a file written by one application can be read by another. Many operating systems provide common user interface so applications can have the same "look and feel."

OS Components

1. Shell:

-Shell handles user interactions.

-It manages the interaction between user and kernal system.

-Shell acts as an interpreter to convert high level language provided

by user to convert in to low level language to understand by kernel.

OS Components

2. Kernel:

-All hardware are under the control of the kernel.

-The user can’t access the hardware directly, the user need kernel to access the hardware.

- 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.

OS Components

3. System Utilities: Programs that perform specialized, individual tasks.

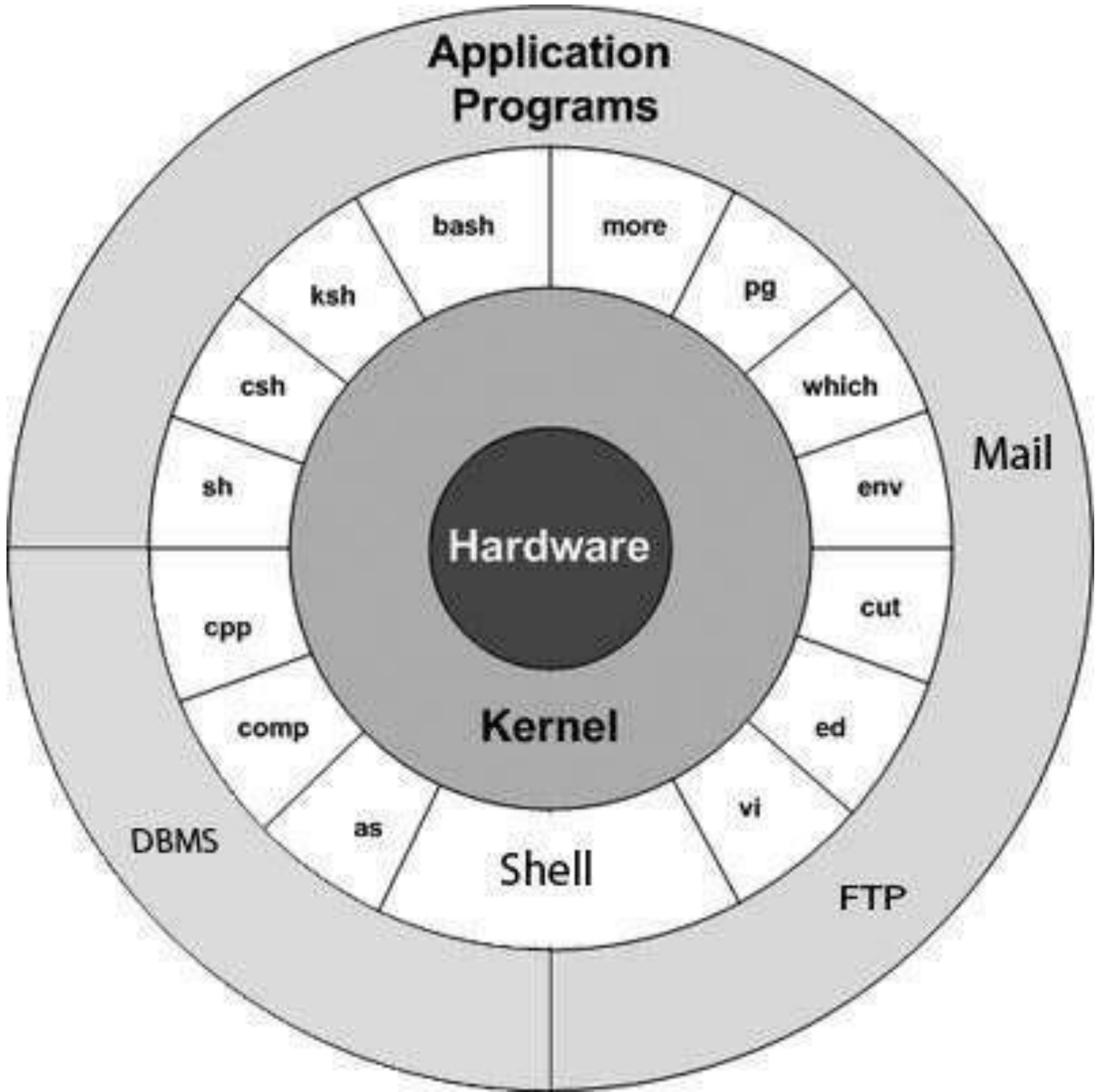
Examples: disk cleanup, Windows Defender, Backup and restore, task manager, etc.

OS Components

-User types command in terminal

-**Shell** converts from high-level language provided by users to low-level and pass instruction to the kernel

-**Kernel** interacts with hardware to execute the instruction.



User mode vs kernel mode

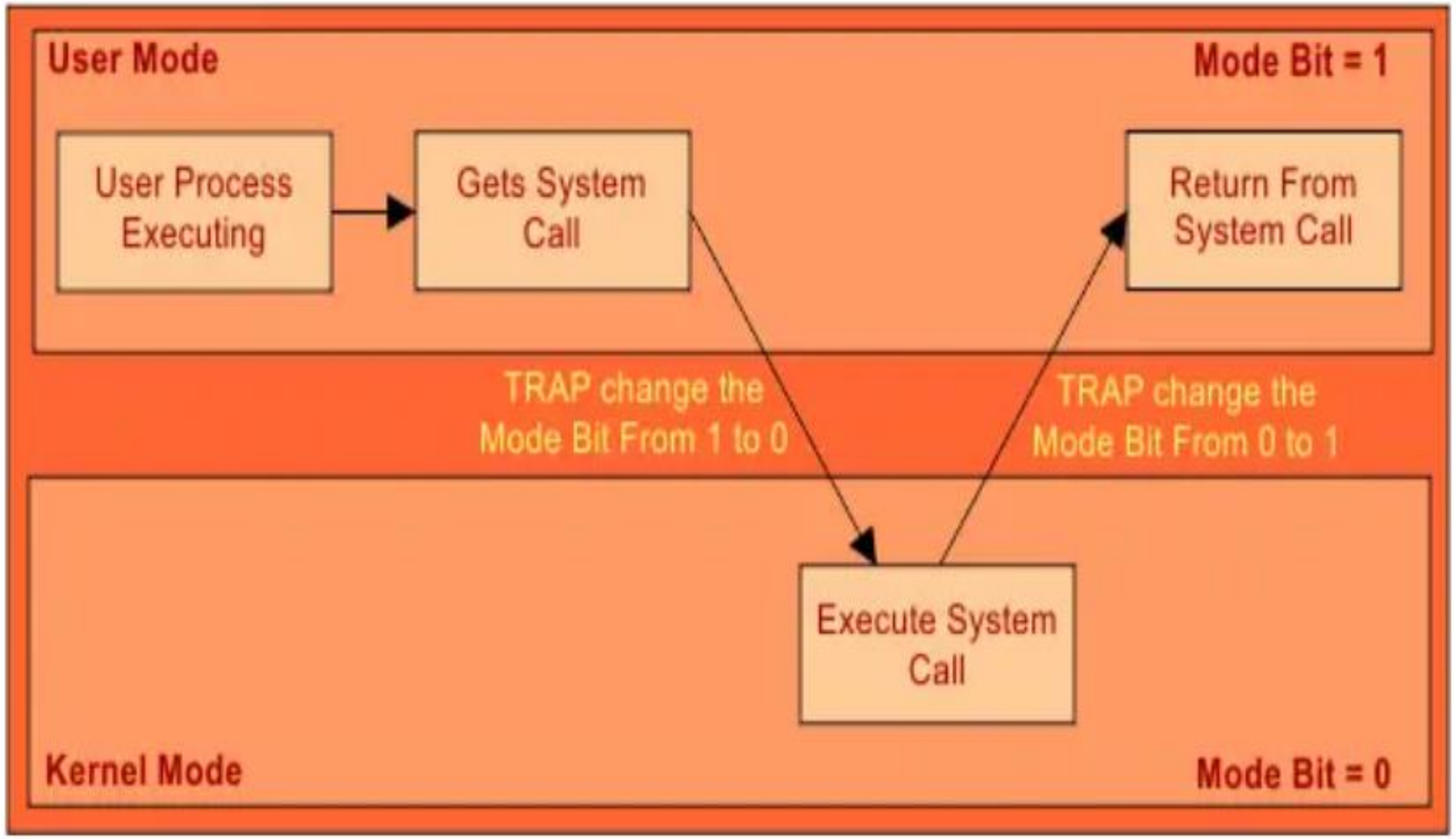
-Executing code in user mode has no ability to directly access some resources like Hard Disk, memory, Printer, and other I/O devices. Because, To access these resources we have to use kernel mode through System Call.

User mode vs kernel mode

-Input is performed in user mode

-User access the hardware through a system call

-Change of mode (user to kernel) called TRAP.



OS Types

1. Batch Operating System
2. Multi-Programming System
3. Multi-Processing System
4. Multi-Tasking Operating System
5. Real-Time Operating System
6. Distributed Operating System
7. Network Operating System
8. Mobile OS
9. Cluster OS
10. Embedded OS

OS Types

1. **Batch Operating System**

-In the 1970s, Batch processing was very popular.

-In this technique, similar types of jobs are batched together and executes in time.

-The system put all of the jobs in a queue on the basis of first come first serve and then executes the jobs one by one.

-The users collect their respective output when all the jobs get executed.

OS Types

-User can't interact with OS (non-interactive)

-Operator interacts with the system

-User-formed job in the form of a punch card and submit to the operator

-A job contains a **program, input data, and control instruction.**

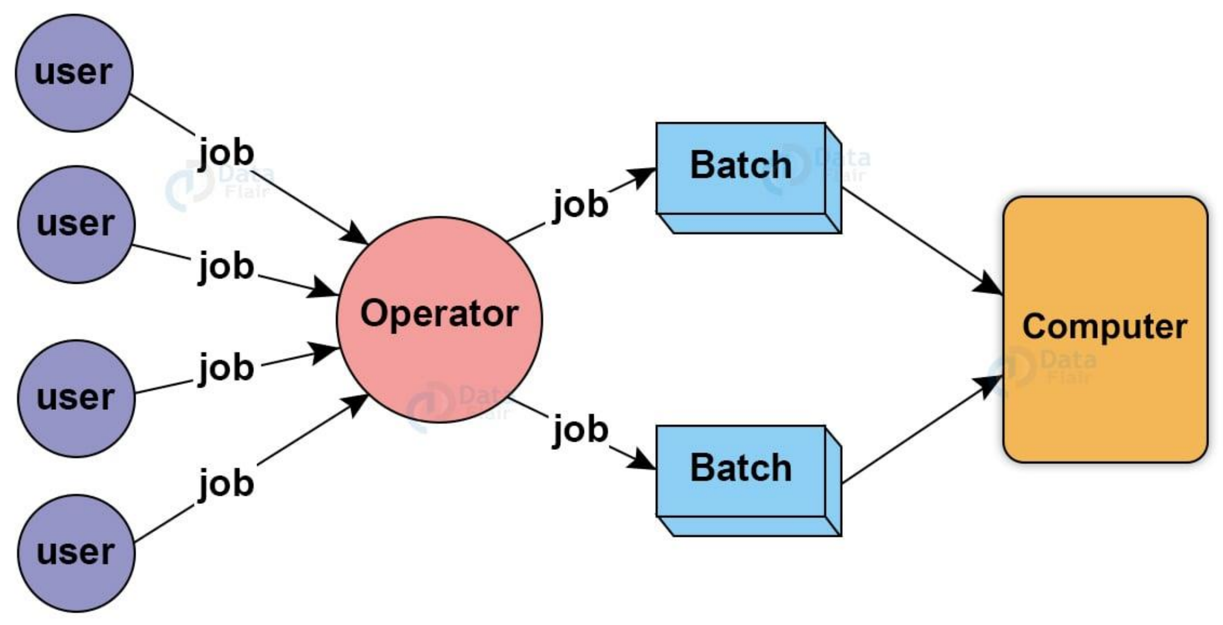
The design Consideration:

-Maximizing job throughput (work/time),

-Resource utilization,

not on minimizing response time.

Batch Operating System



OS Types

Advantages of Batch Operating System

1. Multiple users can share the batch systems.
2. It is easy to manage large work repeatedly in batch systems.

OS Types

Disadvantages of Batch Operating System

1. No prioritization
2. Batch systems are hard to debug.
3. It is sometimes costly.
4. The other jobs will have to wait for an unknown time if any job fails.

Examples of Batch OS: Payroll Systems, Bank Statements, etc.

OS Types

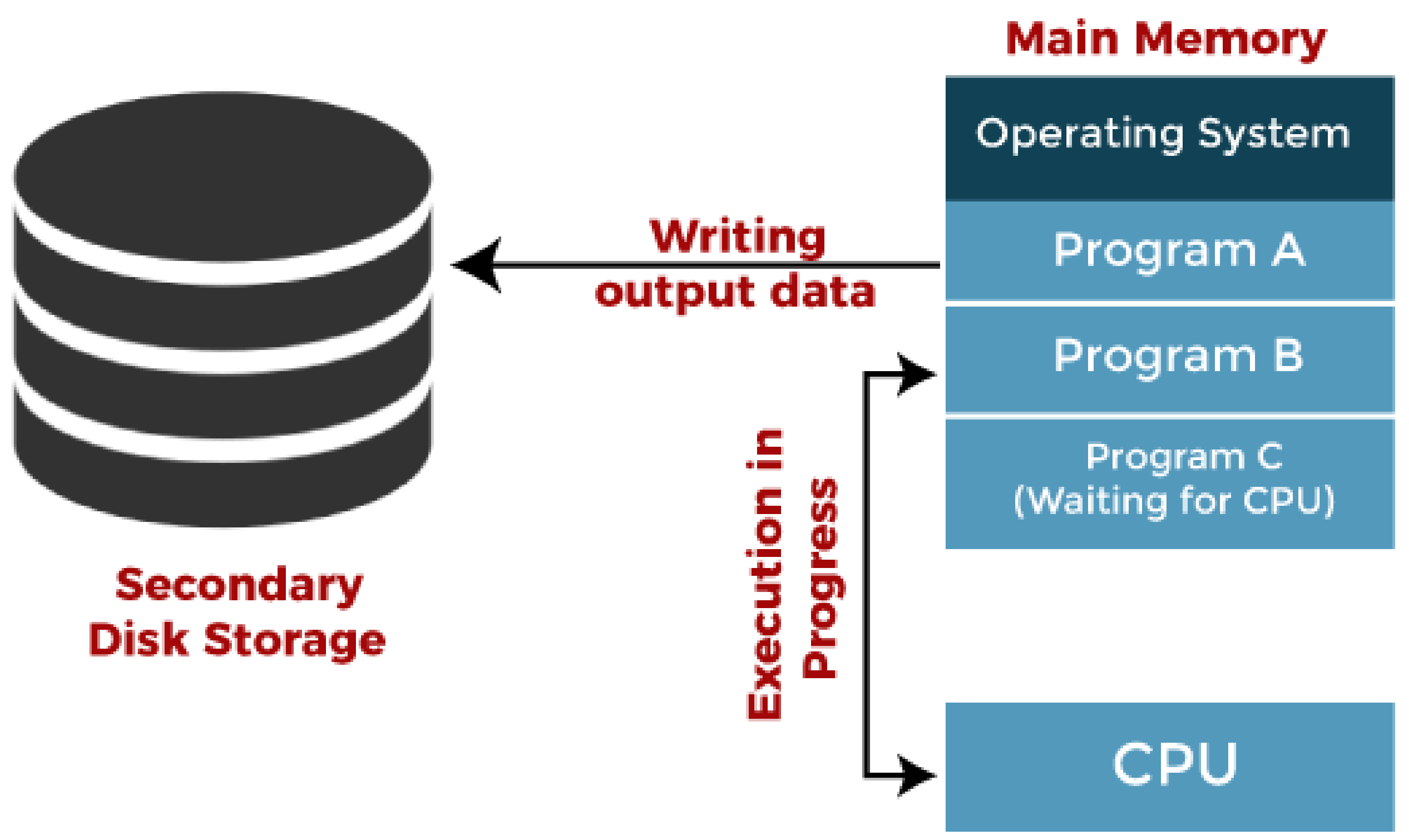
2. Multi-Programming OS

-In Multiprogramming Operating Systems more than one program is present in the main memory and any one of them can be kept in execution.

-It is non-preemptive.

(If one program is executing another program has to wait until the previous program completes execution.)

-program C has to wait until program B completes it's execution.



Jobs in multiprogramming system

Example: Early mainframe systems, like IBM OS/360, were designed to be multi-programming, Traditional UNIX systems etc.

OS Types

Advantages of Multi-Programming Operating System

-Multi Programming increases the Throughput of the System.

-It helps in reducing the response time.

Disadvantages of Multi-Programming Operating System

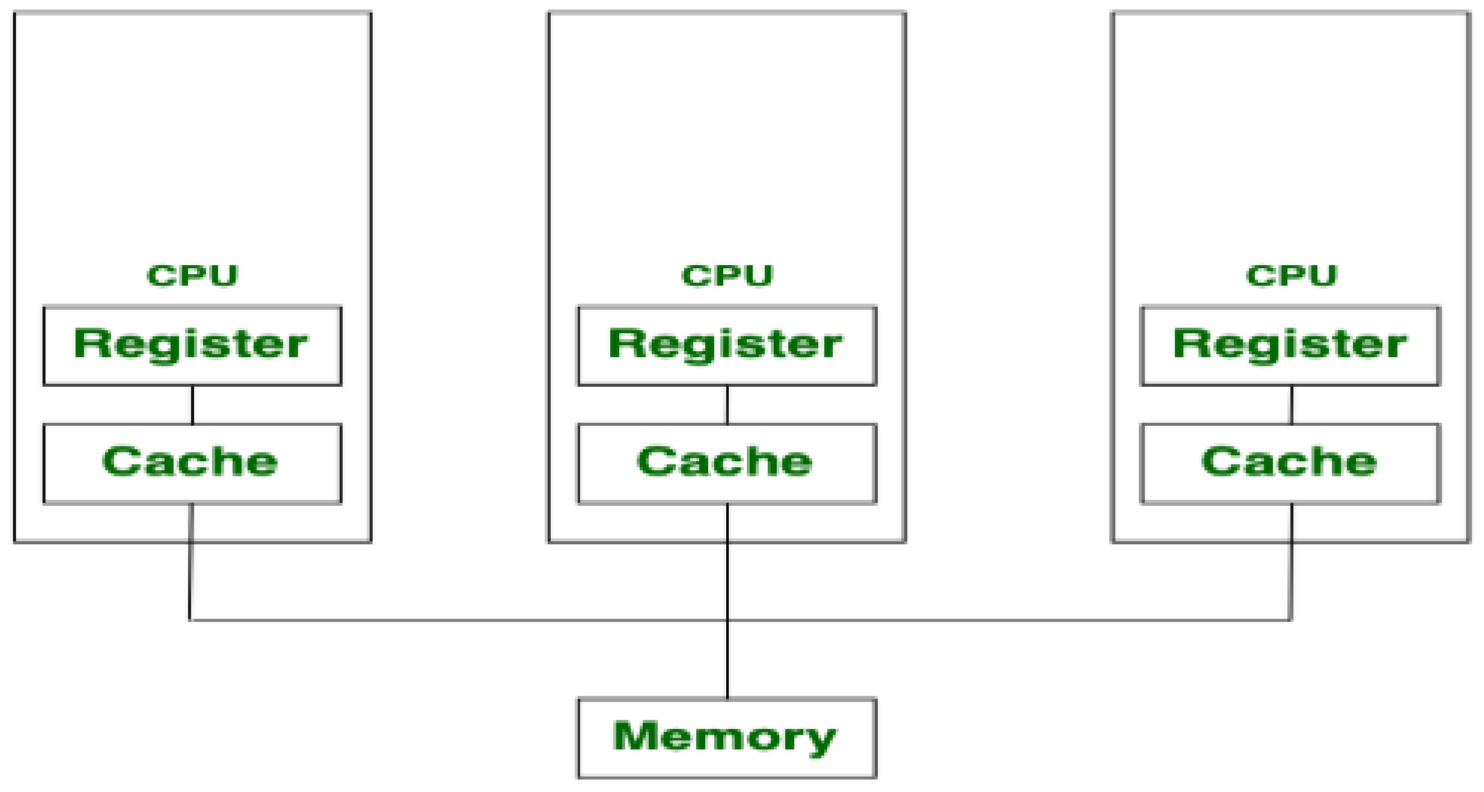
- If one process is executing another program has to wait until the previous program completes execution.

OS Types

3. Multiprocessing Operating System

-In Multiprocessing, Parallel computing is achieved.

-There are multiple CPUs are present in the system which can execute more than one process.



Multiprocessing

OS Types

Advantages of Multi-Processing Operating System

-It increases the throughput of the system.

-As it has several processors, so, if one processor fails, we can proceed with another processor.

Disadvantages of Multi-Processing Operating System

-Due to the multiple CPU, it can be more complex and somehow difficult to understand.

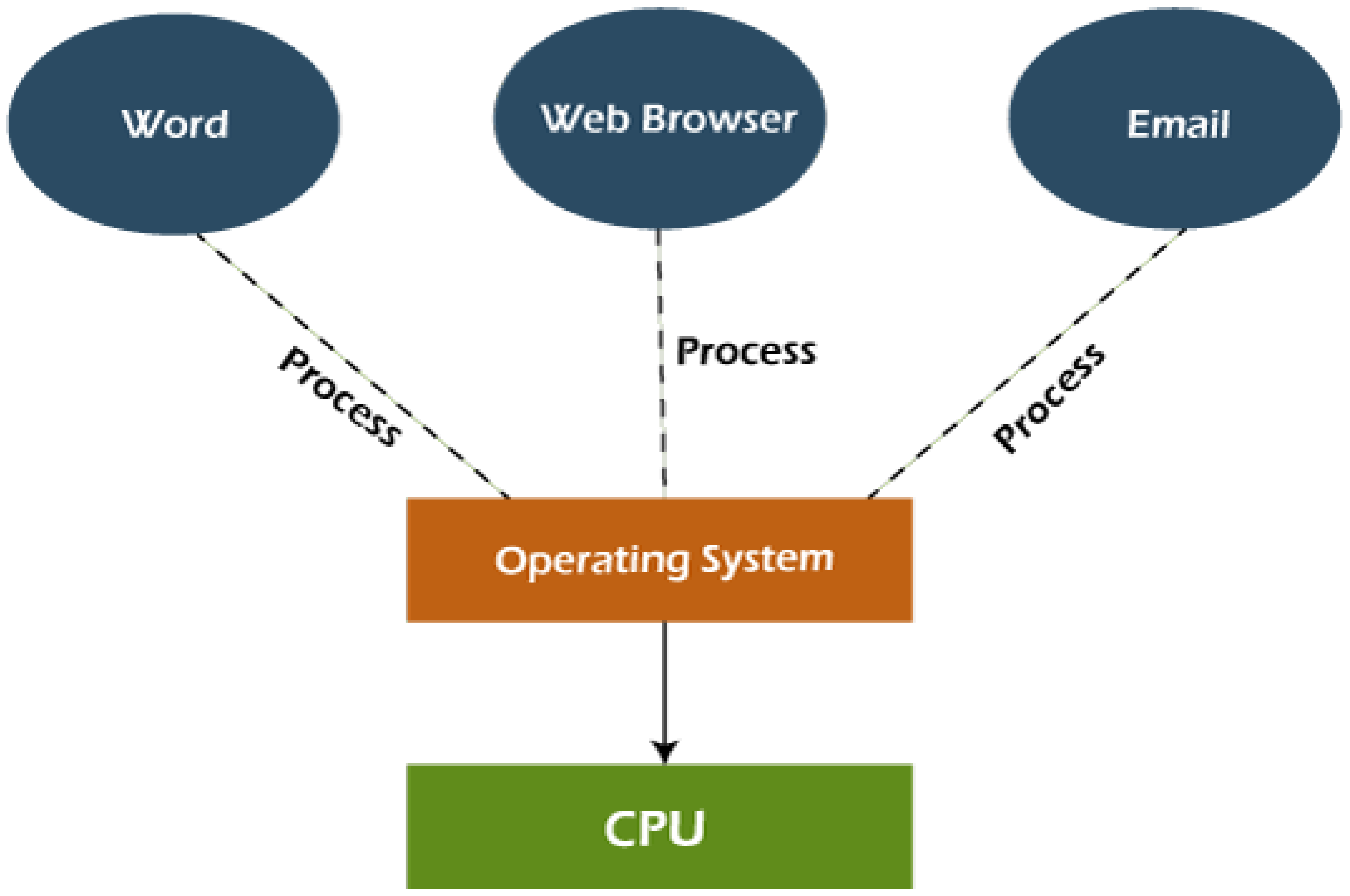
OS Types

4. Multi-Tasking/Time sharing/fare share Operating System

-A Multitasking Operating System is simply a multiprogramming Operating System with having facility of a Round-Robin Scheduling Algorithm.

-It can run multiple programs simultaneously.

-if one task is executing and it can forcefully remove and process for another task based on time quantum.



OS Types

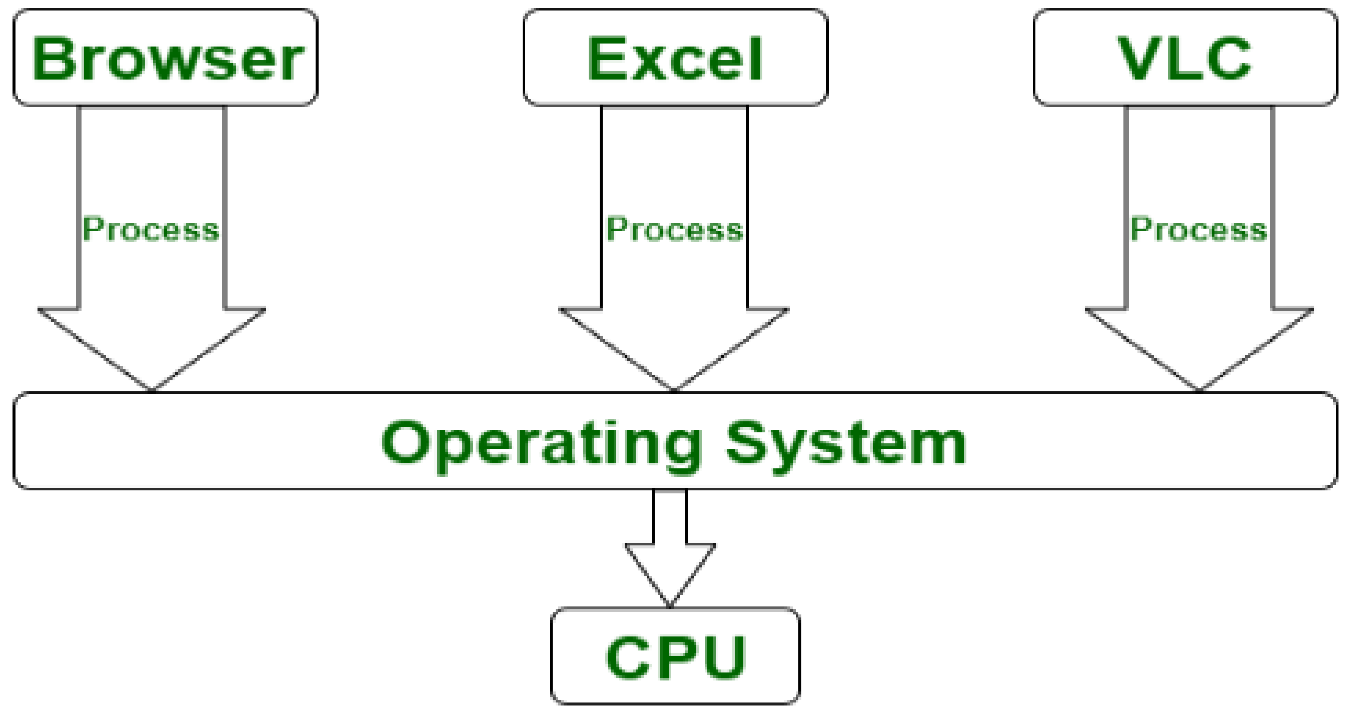
4. Multi-Tasking/Time sharing/fare share

-It seems there is multi tasking at a time, but this is not true.

-It is based on time quantum

-Process switching is fast

-Process execute in FIFO order.



Multitasking

The design considerations:

-**R**esponse time

-**R**eliability

-**U**ser interface capabilities.

OS Types

Advantages of Multitasking operating system

- This operating system is more suited to supporting multiple users simultaneously.
- The multitasking operating systems have well-defined memory management.

Disadvantages of Multitasking operating system

- The multiple processors are busier at the same time to complete any task in a multitasking environment, so the CPU generates more heat.

Multi-tasking vs Multiprocessing

Multi-tasking	Multiprocessing
The execution of more than one task simultaneously is known as multitasking.	The availability of more than one processor per system, that can execute several set of instructions in parallel is known as multiprocessing.
The number of CPU is one.	The number of CPUs is more than one.
It takes moderate amount of time.	It takes less time for job processing.
In this, one by one job is being executed at a time.	In this, more than one process can be executed at a time.
The number of users is more than one.	The number of users is can be one or more than one.
Throughput is moderate.	Throughput is maximum.
Its efficiency is moderate.	Its efficiency is maximum.
Windows, Mac OS, Linux: supports a preemptive multitasking approach	Linux supports multiprocessing, windows server Designed for powerful server applications supporting high-load server tasks by distributing loads across multiple CPUs.

OS Types

5. Real-Time Operating System (RTOS)

-These types of OSs serve real-time systems. This system is time-bound and has a fixed deadline

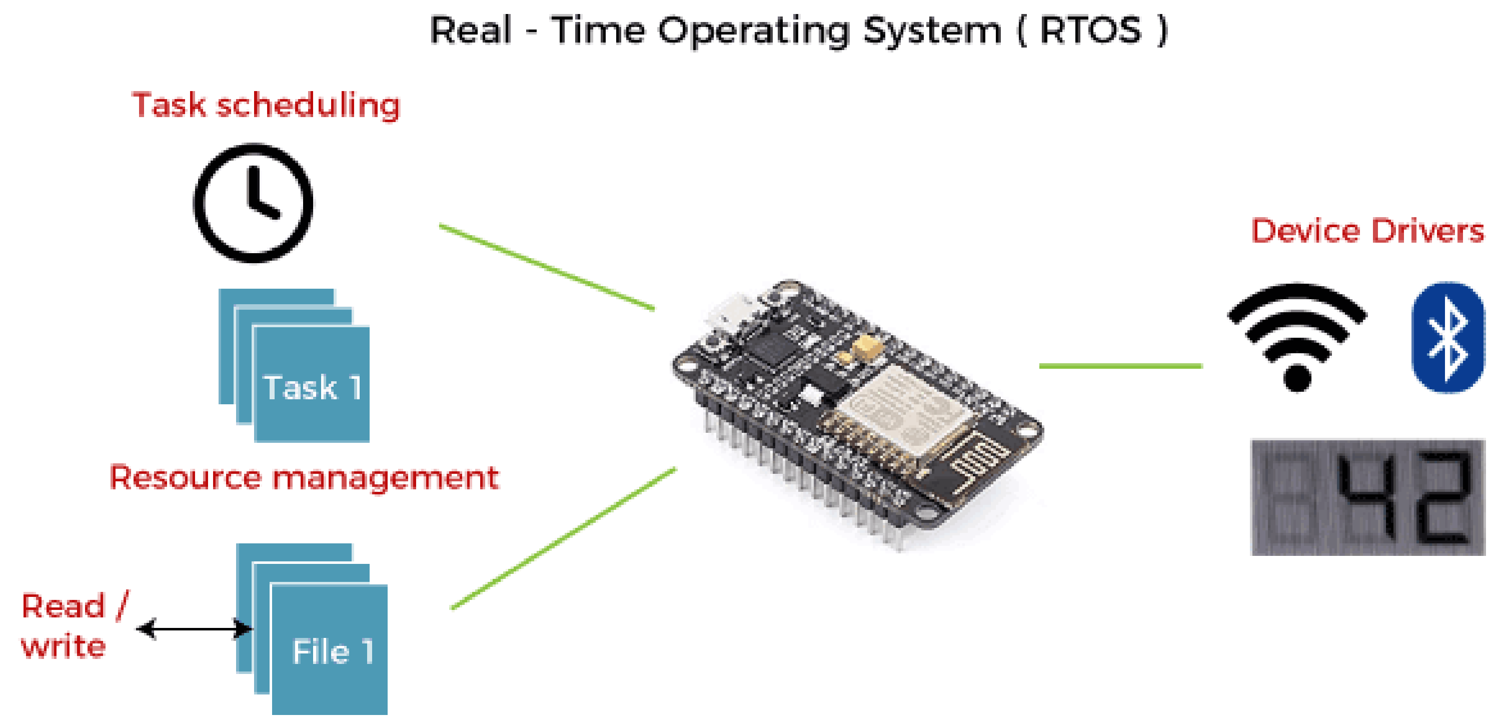
Two types:

1. Hard real-time: the operation should be completed in the exact time

Example: Air Traffic Control, Medical Systems, Time bomb, Missile Lunch

2. Soft real-time: the process will be completed within time bound.

Example: Multimedia Transmission



The design considerations:

-Predictability

-Dependability

-Quick context switching (Response).

Not High Throughput

OS Types

Advantages of Real-time operating system:

- Easy to layout, develop, and execute real-time applications under the real-time operating system.

Disadvantages of Real-time operating system:

- Real-time operating systems are very costly to develop.
- Real-time operating systems are very complex and can consume critical CPU cycles.

Regular OS Vs. RTOS

Regular OS	Real-Time OS (RTOS)
Complex	Simple
Best effort	Guaranteed response
Fairness	Strict Timing constraints
Average Bandwidth	Minimum and maximum limits
Unpredictable behavior	Predictable behavior

OS Types

6. Distributed Operating System

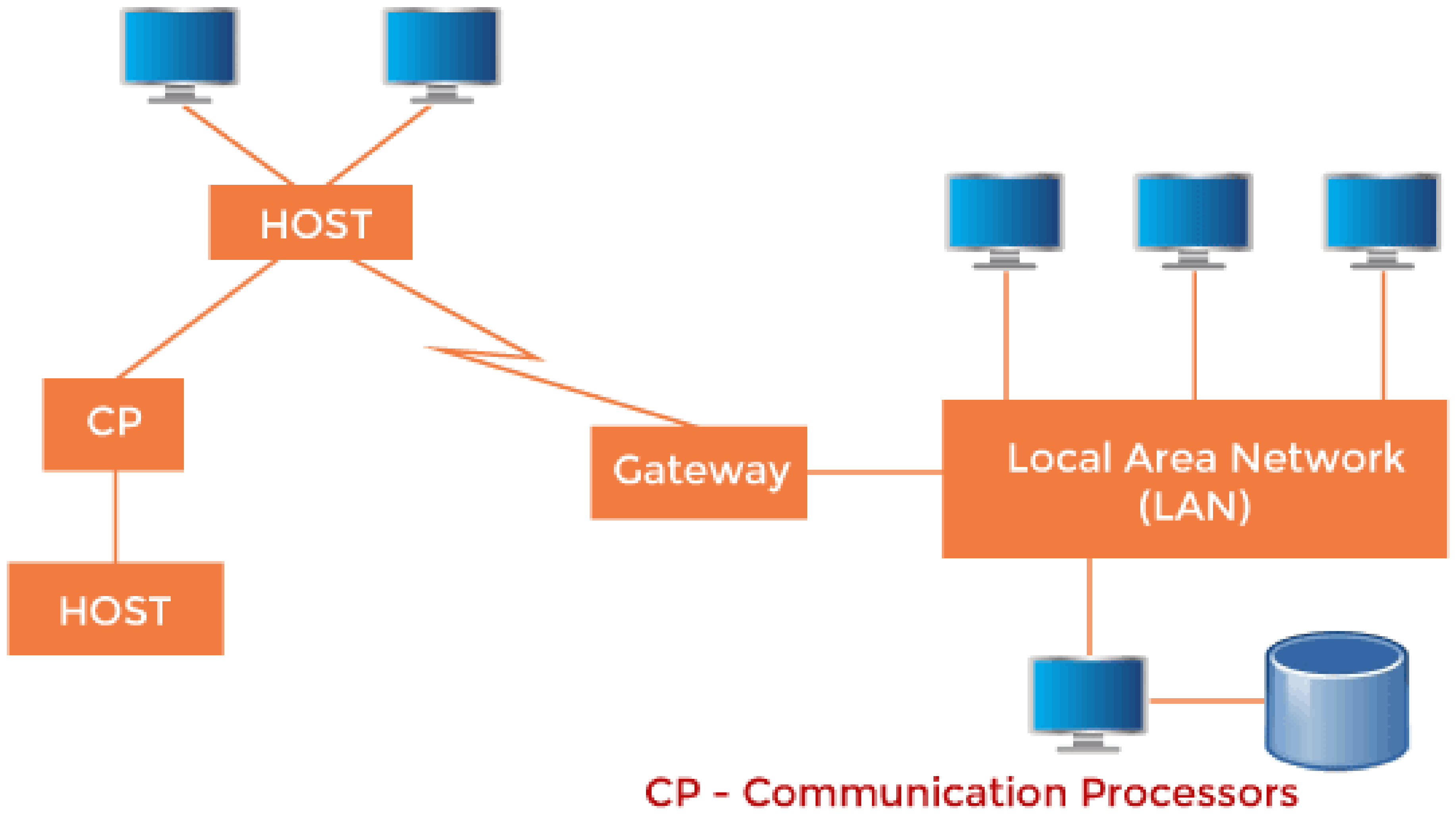
-The Distributed Operating system is not installed on a single machine, it is divided into parts, and these parts are loaded on different machines.

-A part of the distributed Operating system is installed on each machine to make their communication possible.

-Distributed Operating systems are much more complex, large, and sophisticated than Network operating systems because they also have to take care of varying networking protocols.

OS Types

It is always possible that one user can access the files or software which are not actually present on his system but some other system connected within this network
i.e., remote access is enabled within the devices connected in that network.
Examples: Telephone and cellular networks



A Typical View of a Distributed System

OS Types

Advantages of Distributed Operating System

- Failure of one will not affect the other network communication, as all systems are independent of each other.
- Load on host computer reduces.
- Delay in data processing reduces.

OS Types

Disadvantages of Distributed Operating System

- Failure of the main network will stop the entire communication.
- To establish distributed systems the language is used not well-defined yet.
- These types of systems are not readily available as they are very expensive.

OS Types

7. Network Operating System

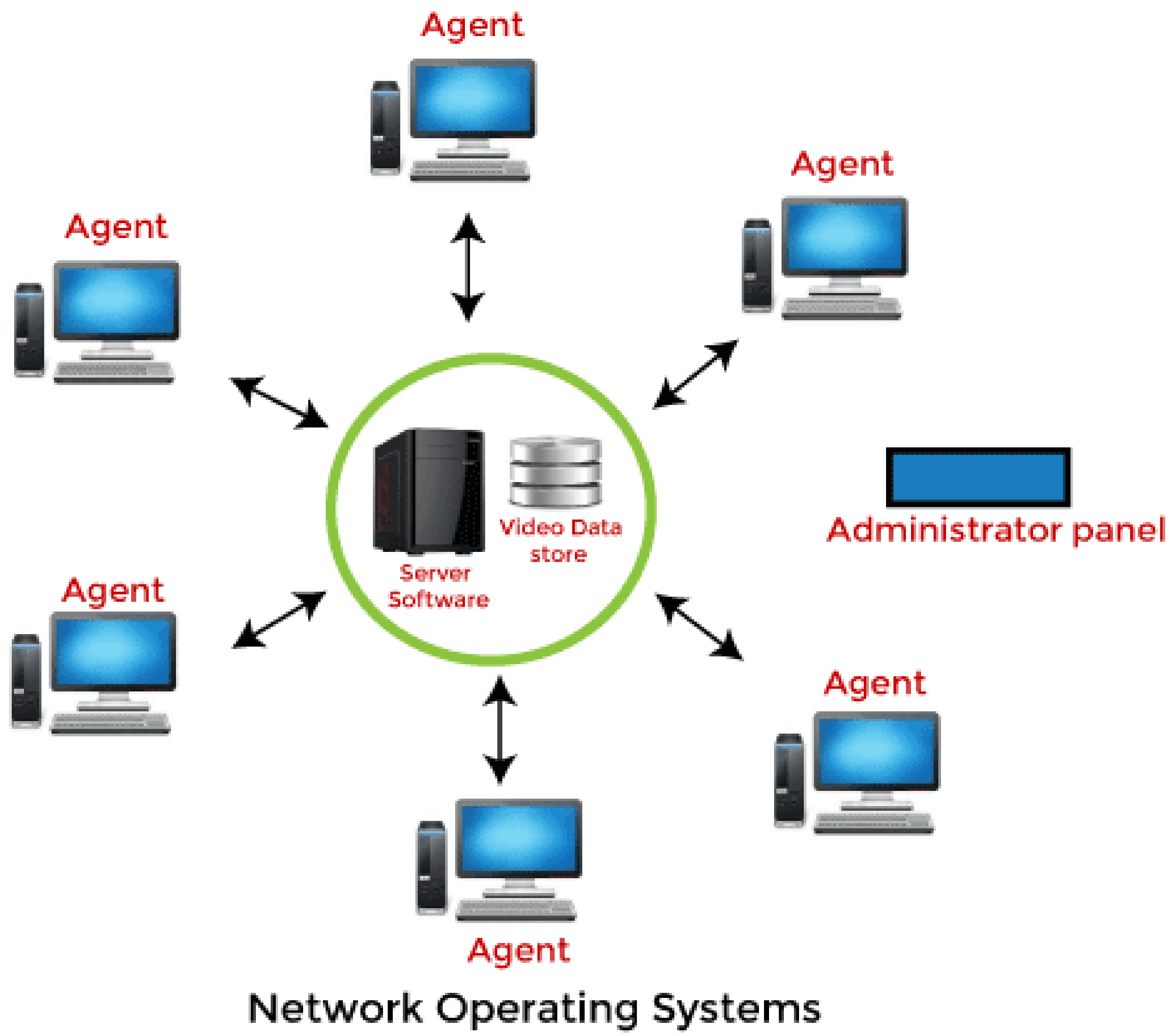
-These systems run on a server and provide the capability to manage data, users, groups, security, applications, and other networking functions.

-These types of operating systems allow shared access to files, printers, security, applications, and other networking functions over a small private network.

OS Types

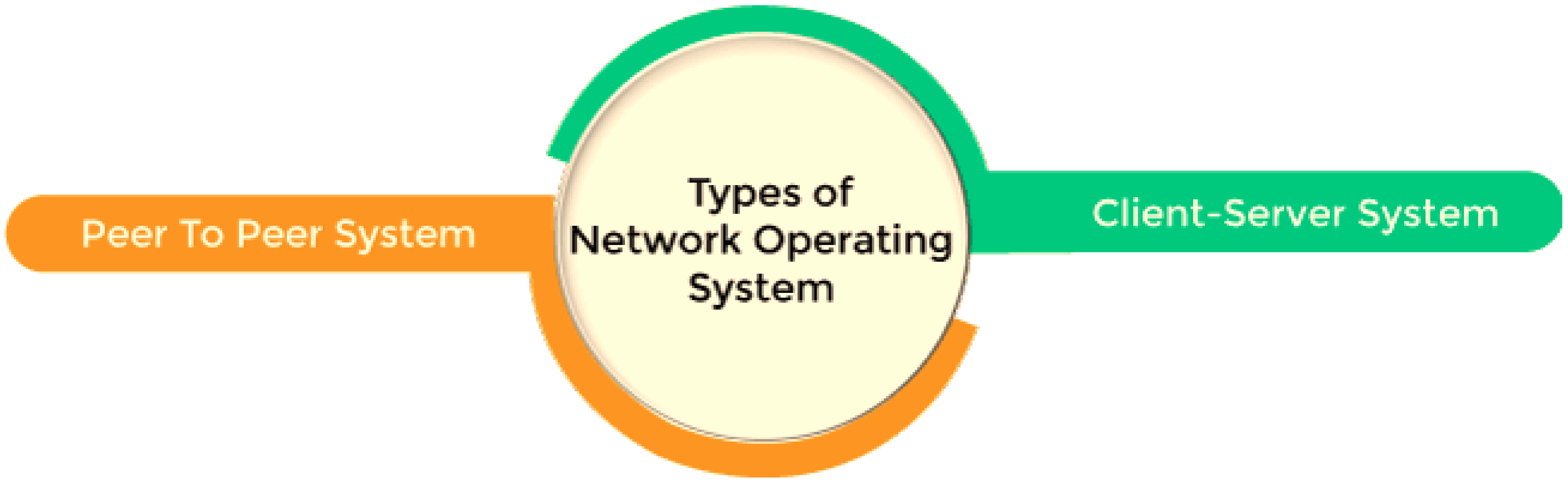
An Operating system, which includes software and associated protocols to communicate with other computers via a network conveniently and cost-effectively, is called Network Operating System.

Examples: Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, BSD, etc.



OS Types

Types of Network OS



OS Types

8. Mobile Operating System:

-Used in mobile phones, smartwatch, TV

Example:

-Android, IOS, Blackberry, Solaris, etc.

OS Types

9. Cluster Operating System:

-similar to parallel systems because both systems use multiple CPUs.

-clustered systems are made up of two or more independent systems linked together

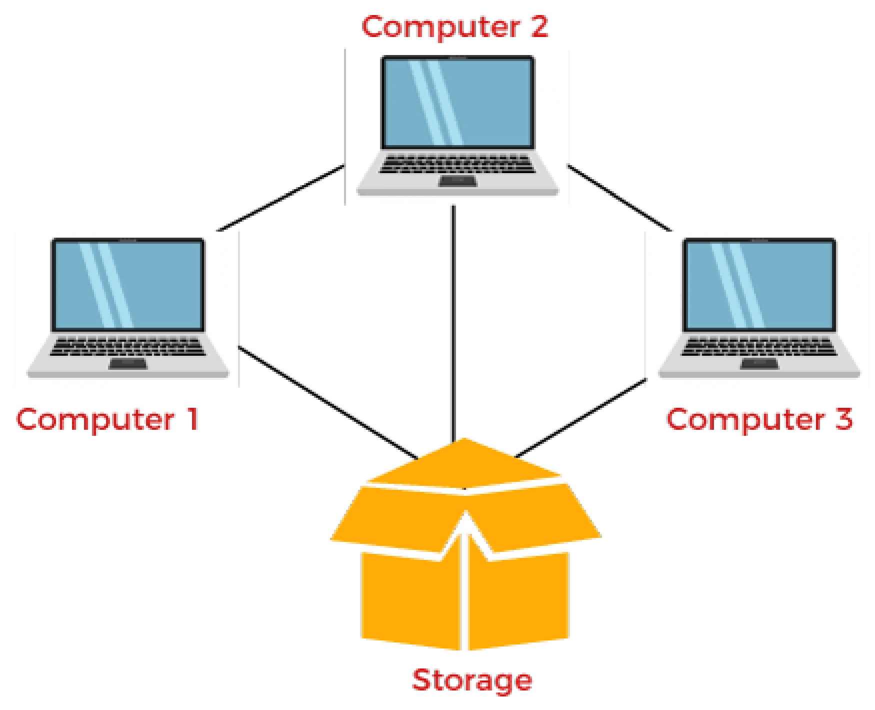
-All cluster nodes use two different approaches to interact with one another, like message passing interface (MPI) and parallel virtual machine (PVM).

(PVM enables the integration of different computing architectures and operating systems into one virtual parallel computer. Used in Complex Simulations, data analysis).

(message passing between various components (or nodes) of the system)

OS Types

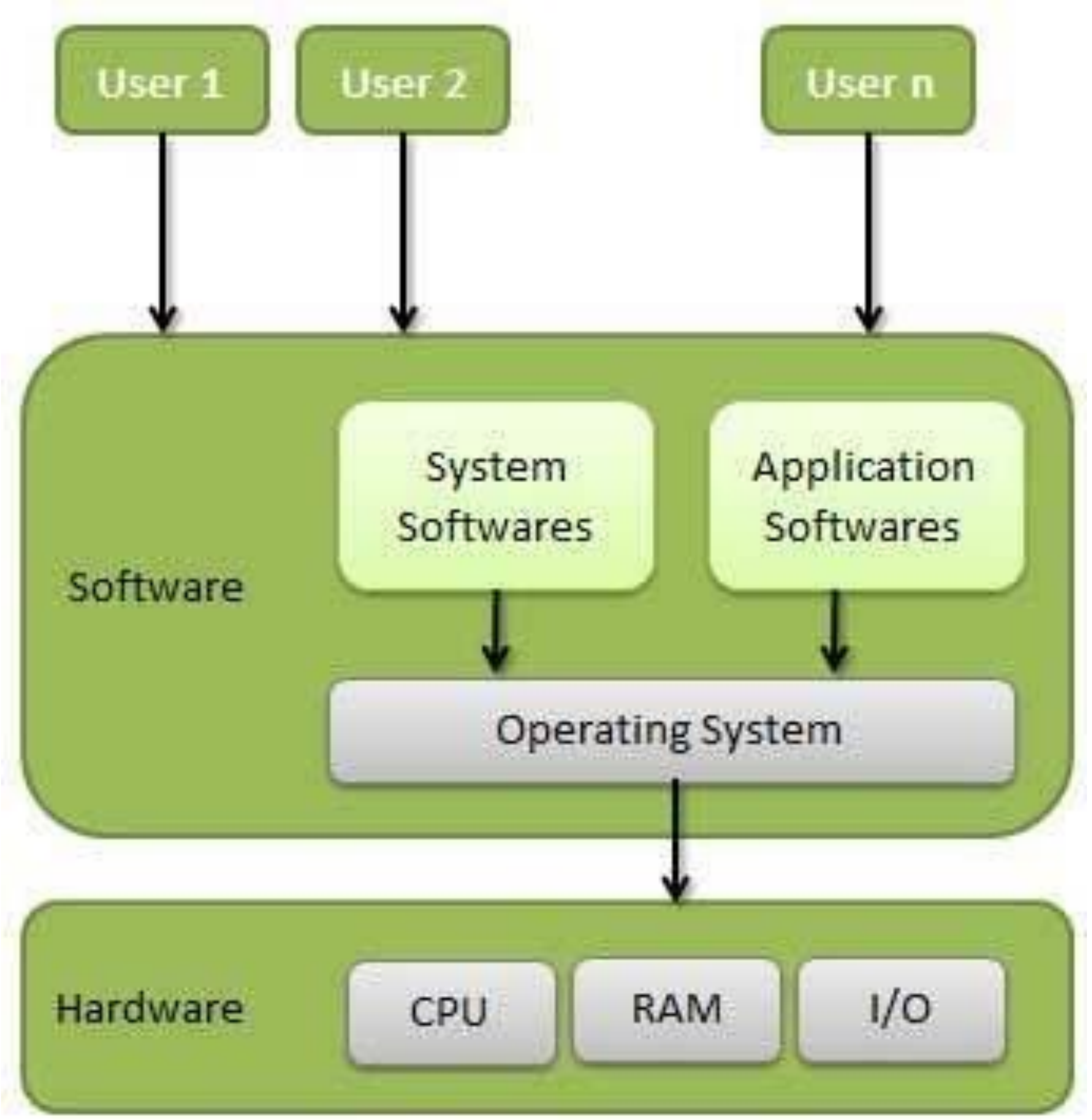
Cluster operating systems are a combination of software and hardware clusters. Hardware clusters aid in the sharing of high-performance disks among all computer systems, while software clusters give a better environment for all systems to operate.



Clustered System

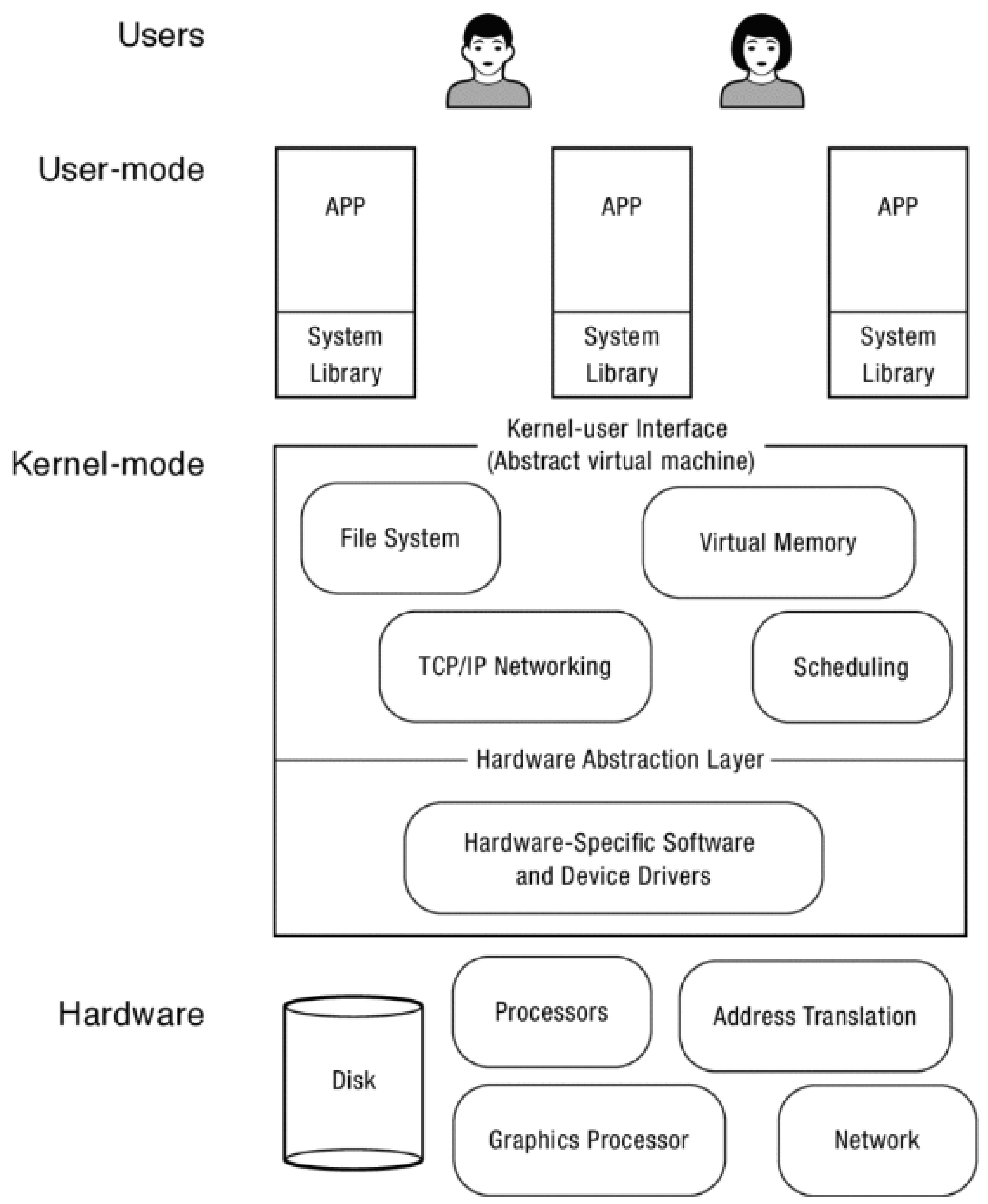
OS Architecture

Some popular Operating Systems include Linux Operating System, Windows Operating System, MAC Os, VMS, Android etc.



OS Architecture

structure of a general-purpose operating system.



OS: Development Landscape

First Generation (1940s-1950s):

-No OS, Users directly interacted with the hardware, programming in machine code.

Second Generation (1950s-1960s):

-Batch processing system, allowing users to submit jobs and have them processed in a batch.

Third Generation (1960s-1970s):

-Time-sharing systems were developed for interactive user access.

Example: UNIX

OS: Development Landscape

Fourth Generation (1970s-1980s):

-File systems, networking, and device management.

Examples: MS-DOS, Mac OS, and early versions of Windows.

Fifth Generation (1980s-Present):

-Modern OS: multitasking, multiuser, advanced GUIs, networking.

Examples: Windows, macOS, and Linux.

Sixth Generation (Present and Future):

-Virtualization, cloud computing technologies. Emphasis on security and scalability. Advanced AI.

OS: System Structure

An operating system is a design that enables user application programs to communicate with the hardware of the machine.

Types of Structure:

1. Simple Structure
2. Monolithic Structure
3. Micro-Kernel Structure
4. Hybrid-Kernel Structure
5. Exo-Kernel Structure
6. Layered Structure
7. Modular Structure
8. Virtual Machines

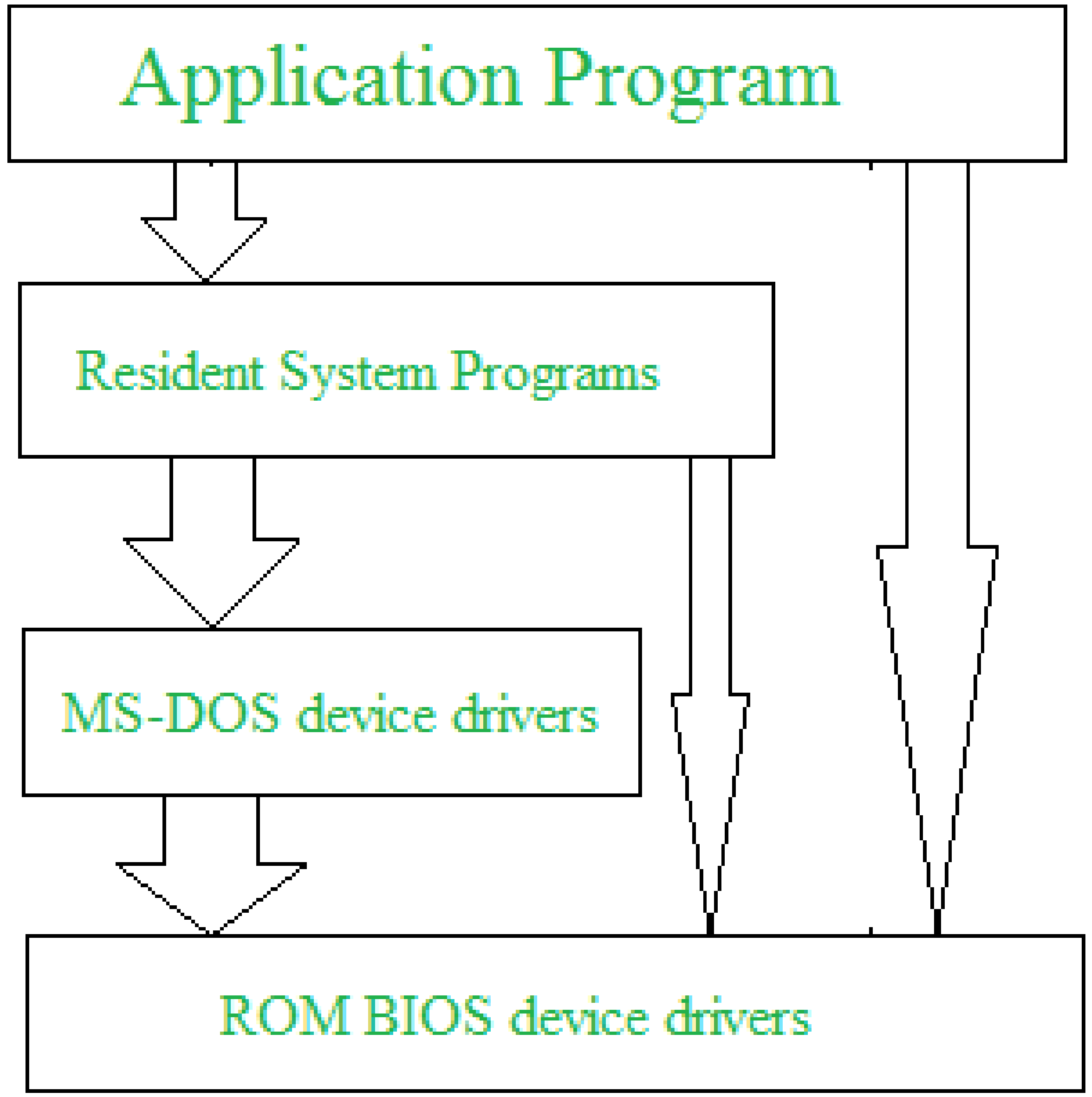
OS: System Structure

1. Simple Structure:

-The Simple structured operating systems do not have a well-defined structure.

-These systems will be simple, small and limited systems.

Example: MS-DOS.



OS: System Structure

Advantages of Simple Structure:

1. It is simple to develop.
2. It offers superior performance.

Disadvantages of Simple Structure:

1. The entire operating system breaks if just one user program malfunctions.
2. Since the layers are interconnected, and in communication with one another, there is no abstraction or data hiding.

OS: System Structure

2. Monolithic structure

-In a monolithic kernel, all system services run in kernel space in a centralized approach.

-Services and device drivers are integrated in one large block of code that runs entirely in kernel mode

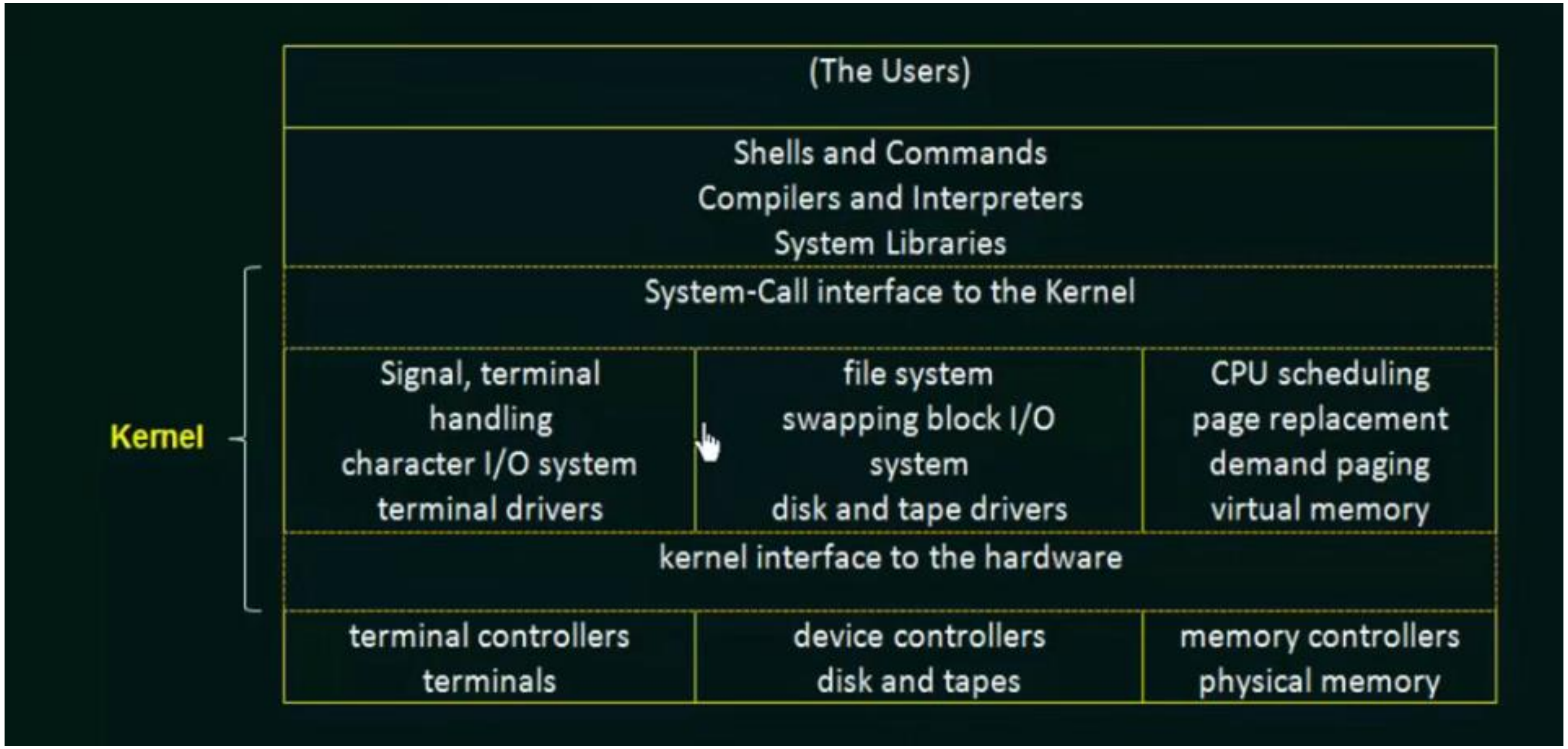
-The monolithic operating system controls all aspects of the operating system's operation, including file management, memory management, device management.

In monolithic architectures, system calls directly invoke the necessary kernel services

Example: Old version of UNIX

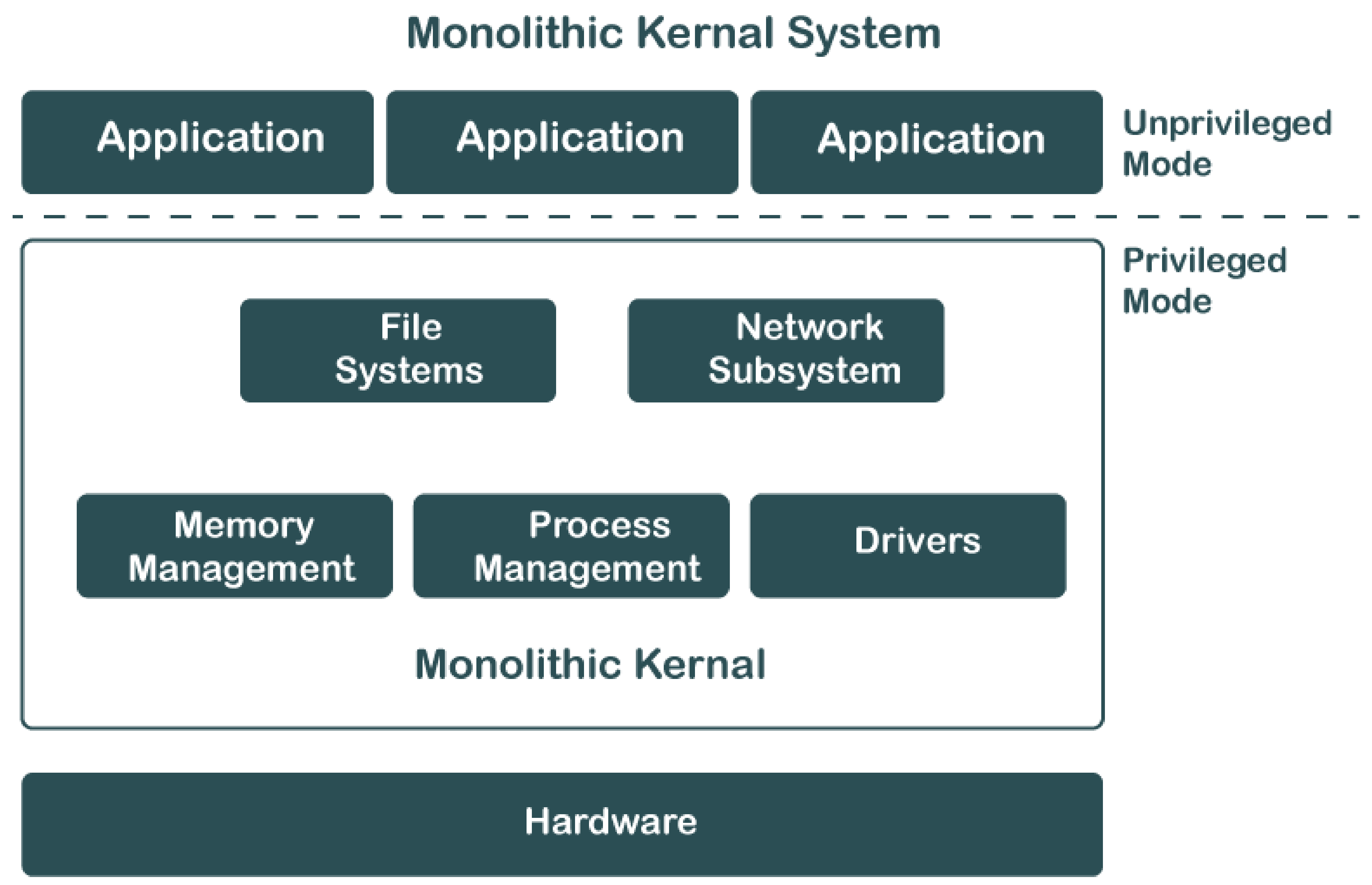
OS: System Structure

2. Monolithic structure



OS: System Structure

This is an old operating system that was used in banks to carry out simple tasks like batch processing and time-sharing, which allows numerous users at different terminals to access the Operating System.



OS: System Structure

Advantages of Monolithic Structure:
-It is easy to design and execute.
-The monolithic kernel runs quickly compared to other systems.
Disadvantages of Monolithic Structure:
-The monolithic kernel's services are interconnected in address space and have an impact on one another, so if any of them malfunctions, the entire system does as well.
-It is not adaptable. Therefore, launching a new service is difficult.
-Everything is in same layer so it is complex to modify.

OS: System Structure

3. Layered structure

-Modularity

-The OS is separated into layers or levels in this kind of arrangement.

-Layer 0 (the lowest layer) contains the hardware,

-layer 1 (the highest layer) contains the user interface

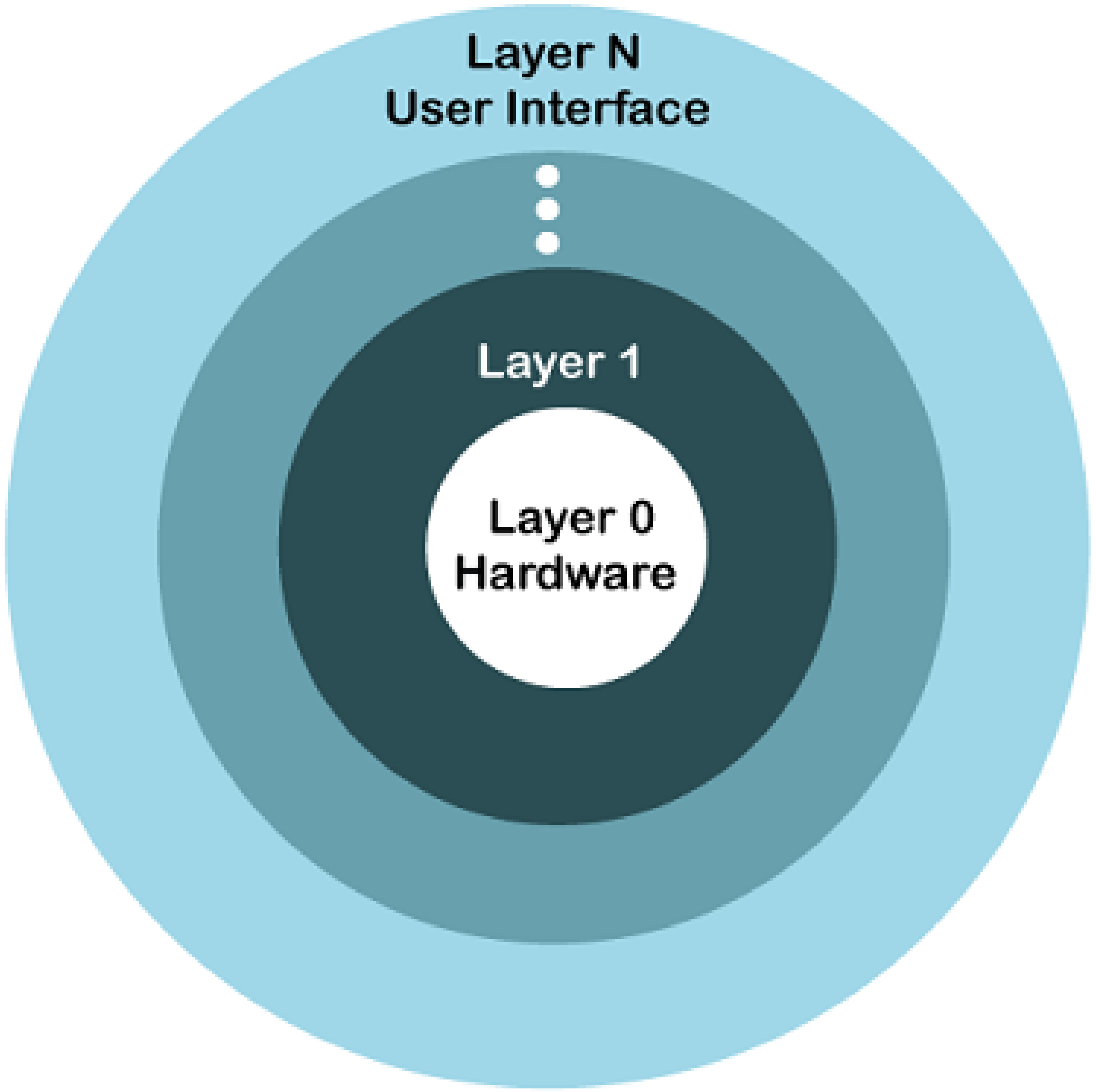
-(layer N). These layers are organized hierarchically, with the top-level layers making use of the capabilities of the lower-level ones.

-Each layer has its task.

Example: UNIX

OS: System Structure

Layered structure



OS: System Structure

Advantages of Layered Structure:

-Work duties are separated since each layer has its own functionality, and there is some amount of abstraction.

-Debugging is simpler because the lower layers are examined first, followed by the top layers.

Disadvantages of Layered Structure:

-Performance is compromised in layered structures due to layering.

-Construction of the layers requires careful design because upper layers only make use of lower layers' capabilities.

OS: System Structure

4. Micro-kernel Structure

-In a microkernel, only the most basic services (such as memory management and process scheduling) run in kernel space, with other services running in user space.

-This results in a smaller kernel called the micro-kernel

-Improves system security and stability because it limits the amount of code running in kernel mode.

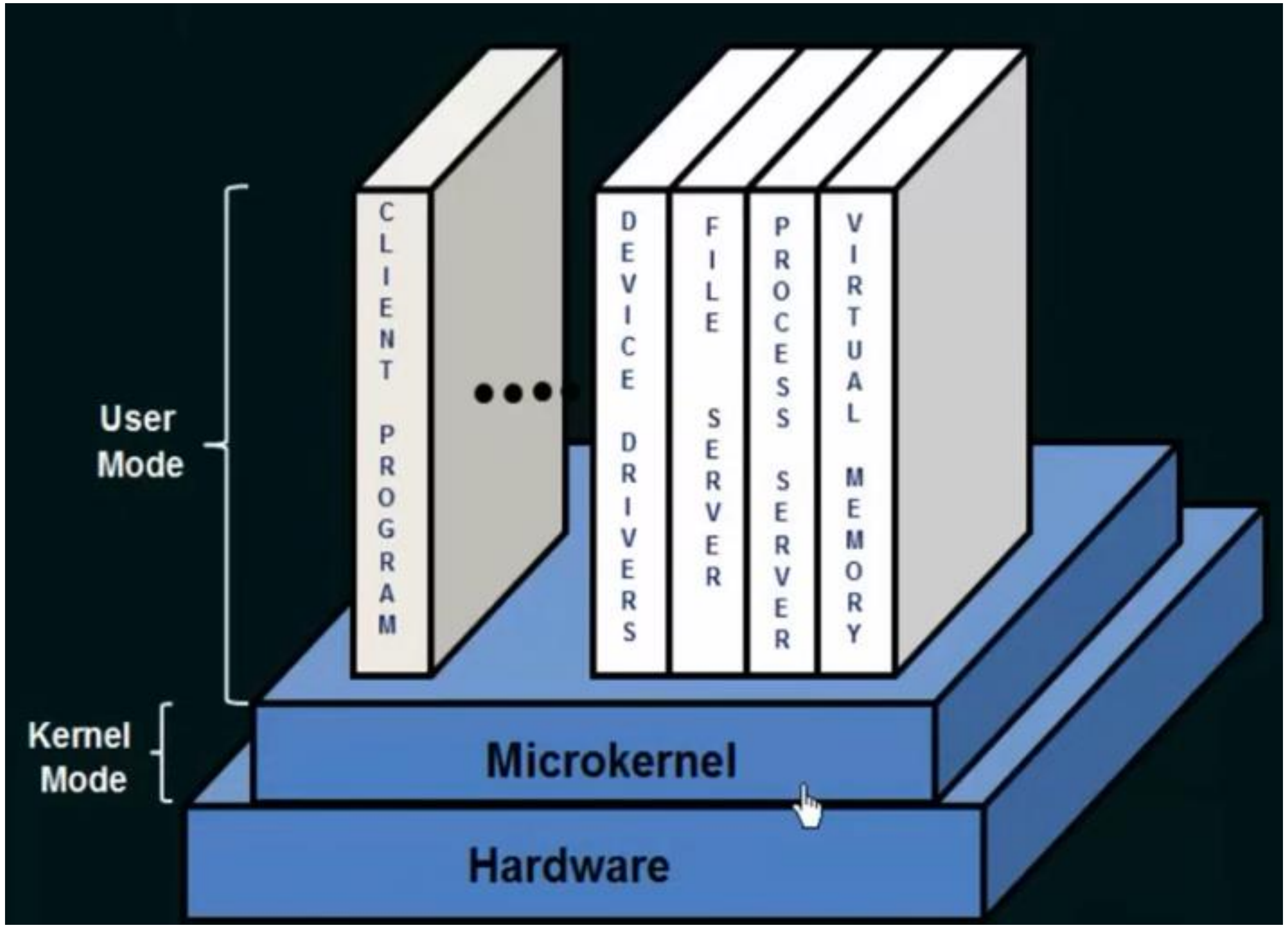
Example: MAC OS

OS: System Structure

4. Micro-kernel Structure

-The main function of microkernel is to provide a communication interface between the client program and system programs.

-The communication is takes place through message passing.



OS: System Structure

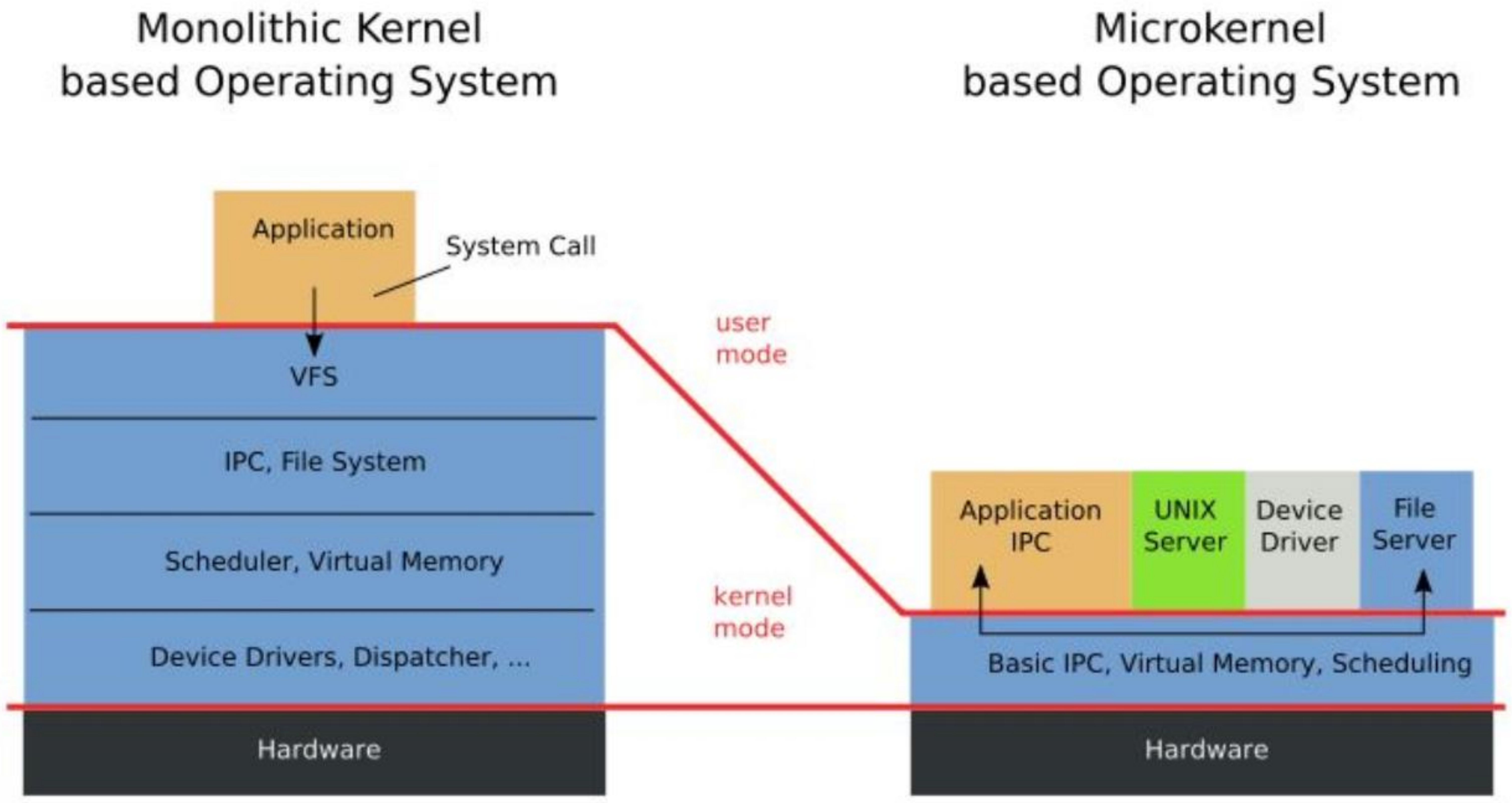
Advantages of Micro-kernel structure

- Since the program is executing in user mode, if any program crashes entire system will not crash.
- It makes the operating system portable to various platforms.
- As microkernels are small so these can be tested effectively.

Disadvantages of Micro-kernel structure

- Increased level of inter-module communication.
- Performance decreases due to an increase in system function overhead.
- It is complex because it exists [kernel + user interface]

Monolithic vs Microkernel



OS: System Structure

5. Hybrid-Kernel Structure

Hybrid-kernel structure is nothing but just a combination of both monolithic-kernel structure and micro-kernel structure.

Example: UNIX OS supports hybrid structure.

OS: System Structure

Advantages of Hybrid-Kernel Structure

- It offers good performance as it implements the advantages of both structure in it.
- It provides better isolation and security by implementing micro-kernel approach.
- It enhances overall system reliability by separating critical functions into micro-kernel for debugging and maintenance.

OS: System Structure

Disadvantages of Hybrid-Kernel Structure

- It increases overall complexity of system by implementing both structure (monolithic and micro) and making the system difficult to understand.
- The layer of communication between micro-kernel and other component increases time complexity and decreases performance compared to monolithic kernel.

OS: System Structure

6. Exo-Kernel Structure

-An Exokernel OS design pushes the boundaries of minimalism.
(i.e. reduce the abstraction and provide direct access to hardware)
-The exokernel architecture's goal is to enable application-specific customization by separating resource management.
-Applications are given direct access to hardware resources,
-Used in distributed systems.

OS: System Structure

Advantages of Exo-kernel

- Support for improved application control.
- It improves the performance of the application.
- Each user-space program is allowed to use a custom memory management system.

OS: System Structure

Disadvantages of Exo-kernel

- A decline in consistency
- Exokernel interfaces have a complex architecture.

OS: System Structure

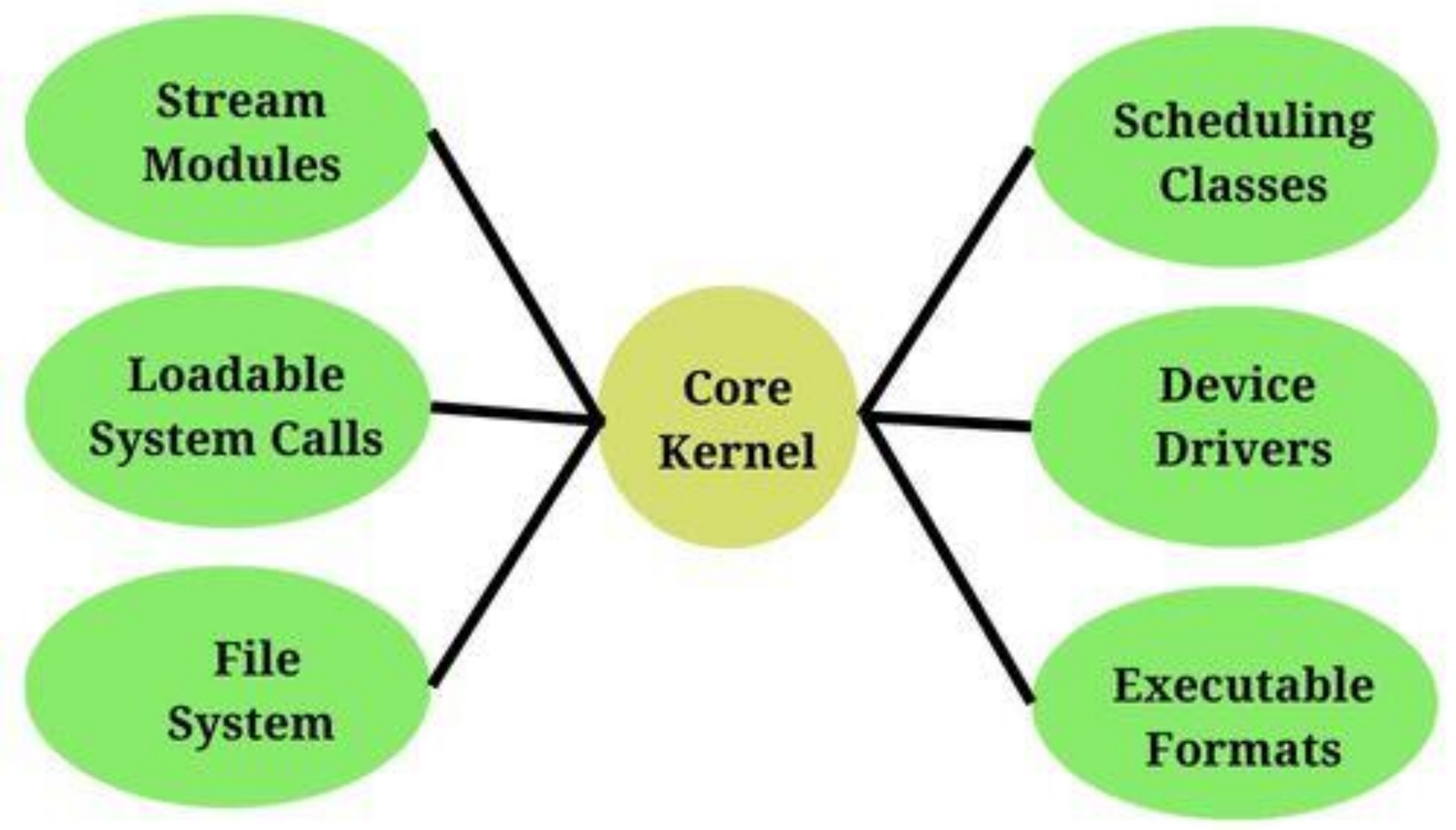
7. Modular structure or approach

A modular OS is based on the idea of dividing the system into smaller and independent units, called modules, that can be loaded and unloaded as needed.

-It is considered as the best approach for an OS.

-It involves designing of a modular kernel.

For example, Solaris OS is organized



The kernel contains the set of core components and links to other services of the operating system.

OS: System Structure

8. VIRTUAL MACHINES (VMs)

-The hardware of our personal computer, including the CPU, disc drives, RAM, and NIC (Network Interface Card), is abstracted by a virtual machine.

-An operating system enables us to run multiple processes concurrently.

-A virtual box is an example of it.

OS: System Structure

Advantages of Virtual Machines:

- Due to total isolation between each virtual machine and every other virtual machine, there are no issues with security.
- A virtual machine may offer an architecture for the instruction set that is different from that of actual computers.
- Simple availability, accessibility, and recovery convenience.

OS: System Structure

Disadvantages of Virtual Machines:

- Depending on the workload, operating numerous virtual machines simultaneously on a host computer may have an adverse effect on one of them.
- When it comes to hardware access, virtual computers are less effective than physical ones.

OS: Design Strategies

Process Management: Design considerations for managing processes and threads effectively.

Memory Management: Techniques such as paging, segmentation, and virtual memory.

I/O and File System Design: Strategies to manage input/output operations and file system organization.

OS: Implementation Techniques

Programming Languages: Role of languages like C and assembly in the development of operating systems.
System Calls: Interface between the OS and the applications.
Concurrency: Mechanisms like semaphores, mutexes, and monitors used to handle concurrency.

Compare and contrast between following.
Parallelism:
Concurrency
Sequential:

OS: System Calls

-The way to access the kernel mode from user mode is called system call.
-The interface between a process and an operating system is provided by system calls.
-In general, system calls are available as assembly language instructions.
-System calls are usually made when a process in user mode requires access to a resource. Then it requests the kernel to provide the resource via a system call.

OS: System Calls

Why do you need system calls in Operating System?

situations where you must require system calls.

- It is must require when a file system wants to create or delete a file.
- Network connections require the system calls to sending and receiving data packets.
- If you want to read or write a file, you need to system calls.
- If you want to access hardware devices, including a printer, scanner, you need a system call.
- System calls are used to create and manage new processes.

OS: System Calls

What is the purpose of System Calls?

Ans: The purpose of the system calls is to allow user-level applications access of the services provided by the kernel. The user apps do not have the privilege to perform operations, so they make system calls which further request a kernel to provide a specific service.

What happens when a System Call is executed?

Ans: When a system call is executed, a context switch occurs and the computer system switches from user mode to kernel mode and now the kernel performs the desired operation.

System Calls Types

Types of System Calls

1. Process Control
2. File Management
3. Device Management
4. Inter-Process Communication

System Calls Types

Process Control
Process control is the system call that is used to direct the processes.
Examples: creating, loading, aborting, ending, executing, processing, terminating the process, etc.

fork(): Creates a new process (child) by duplicating the current process (parent).
exec(): Loads and runs a new program in the current process and replaces the current process with a new process.

System Calls Types

Process Control

wait(): The primary purpose of this call is to ensure that the parent process doesn’t proceed further with its execution until all its child processes have finished their execution.

exit(): It simply terminates the current process.

kill(): This call sends a signal to a specific process and has various purpose including – requesting it to quit voluntarily, or force quit, or reload configuration.

System Calls Types

File Management

File management is a system call that is used to handle the files.

Examples: create files, delete files, open, close, read, write, etc.

open(): Opens a file for reading or writing.

read(): Reads data from a file

write(): Writes data to a file.

close(): Closes a previously opened file.

seek(): Moves the file pointer within a file.

System Calls Types

Memory Management

brk(): Changes the data segment size for a process in HEAP Memory.

sbrk(): To increase or decrease the HEAP size.

mlock() and **unlock():** memory lock defines a mechanism through which certain pages stay in memory and are not swapped out to the swap space in the disk.

System Calls Types

Device Management

Device management is a system call that is used to deal with devices. Examples: read device, write, get device attributes, release device, etc.

SetConsoleMode(): This call is made to set the mode of console (input or output).

WriteConsole(): It allows us to write data on console screen.

ReadConsole(): It allows us to read data from console screen.

open(): This call is made whenever a device or a file is opened.

close(): This call is made when the system closes the file or device.

System Calls Types

Inter-Process Communication:

To communicate between two or more processes.

pipe(): Creates a unidirectional communication channel between processes.

socket(): Creates a network socket for communication.

shmget(): It is short for – ‘shared-memory-get’. It allows one or more processes to share a portion of memory.

semget(): It is short for – ‘semaphore-get’. This call typically manages the coordination of multiple processes.

System Calls examples

Process	Windows	Unix

Process Control	CreateProcess()
	ExitProcess()
	WaitForSingleObject()
		Fork()
		Exit()
		Wait()

File Manipulation	CreateFile()
	ReadFile()
	WriteFile()
	CloseHandle()
		Open()
		Read()
		Write()
		Close()

Device Management	SetConsoleMode()
	ReadConsole()
	WriteConsole()
		Ioctl()
		Read()
		Write()

Information Maintenance	GetCurrentProcessID()
	SetTimer()
	Sleep()
		Getpid()
		Alarm()
		Sleep()

System Calls examples

Process	Windows	Unix

Communication	CreatePipe()
	CreateFileMapping()
	MapViewOfFile()
		Pipe()
		Shmget()
		Mmap()

Protection	SetFileSecurity()
	InitializeSecurityDescriptor()
	SetSecurityDescriptorgroup()
		Chmod()
		Umask()
		Chown()

OS Challenges

Reliability

-Does the system do what it was designed to do?

Availability

-What portion of the time is the system working?

-Mean time to failure (MTTF), Mean time to Repair (MTTR)

Security

-Can the system be compromised by an attacker?

Privacy

-Data is accessible only to authorized users

OS Challenges

Performance

Latency/response time: How long does an operation take to complete?

Throughput: How many operations can be done per unit of time?

Overhead: How much extra work is done by the OS?

Fairness: How equal is the performance received by different users?

Predictability: How consistent is the performance over time?

Open source

The term "open source" refers to computer software or applications where the owners or copyright holders enable the users or third parties to use, see, and edit the product's source code.

Example: Linux, Open Solaris, Free RTOS, Open BDS, Free BSD, Minix, etc.

Open source: Characteristics

Community Collaboration: Development and maintenance are often carried out by a community of volunteers and developers from around the world.

Licensing: Open-source operating systems are typically released under licenses that allow users to use, modify, and distribute the software freely. Common open-source licenses include the GNU General Public License (GPL), Apache License, and MIT License.

Open source

The term "open source" refers to computer software or applications where the owners or copyright holders enable the users or third parties to use, see, and edit the product's source code.

Example: Linux, Open Solaris, Free RTOS, Open BDS, Free BSD, Minix, etc.

Open source: Characteristics

Community Collaboration: Development and maintenance are often carried out by a community of volunteers and developers from around the world.

Licensing: Open-source operating systems are typically released under licenses that allow users to use, modify, and distribute the software freely. Common open-source licenses include the GNU General Public License (GPL), Apache License, and MIT License.

Open source: Characteristics

Distribution: Users are free to distribute the operating system and its modifications.

Security: Open-source software often benefits from a large community of users who scrutinize the code for security vulnerabilities.

Open source: Characteristics

Flexibility and Customization: Users and developers have the freedom to customize the operating system according to their requirements.

Open source: Development

1960s - 1970s: Early Days of Computing:
The advent of time-sharing systems and the development of operating systems like UNIX at Bell Labs in the 1970s marked a shift toward more modular and shareable software.

Open source: Development

1980s: Emergence of Free Software:

In the early 1980s, the Free Software Foundation (FSF) was founded by Richard Stallman. Stallman advocated for the concept of "free software," emphasizing users' freedom to run, modify, and distribute software.

Stallman introduced the GNU (GNU's Not Unix) project to develop a free and open-source Unix-like operating system.

-Key components such as the GNU Compiler Collection (GCC) and the GNU General Public License (GPL) were established.

Open source: Development

Late 1990s: Open Source Initiative (OSI) Formation:

-In 1998, the term "open source" was coined, and the Open Source Initiative (OSI) was established.
-The OSI aimed to promote and protect the open-source software development model.
-The OSI introduced the Open Source Definition, a set of criteria to determine whether a software license qualifies as open source.

Open source: Development

1991: Birth of Linux:

Linus Torvalds, a Finnish student, released the first version of the Linux kernel in 1991. Linux was developed as a Unix-like operating system kernel and was released under the GPL.

Open source: Development

2010s - Present: Mainstream Acceptance and Growth:

-Open-source software became widely accepted in both the business and consumer realms.
-Many companies adopted open-source solutions for their infrastructure and development needs.
-Platforms like GitHub provided a collaborative environment for developers to contribute to open-source projects.

Open source: Development

Graphical User Interfaces: The development of GUIs for Linux made more user-friendly and accessible to a wider audience.

Mobile and Embedded Linux: Linux found its way into mobile devices, with the Android operating system for smartphones and tablets.

Linux on the Desktop: Linux-based desktop OS, such as Ubuntu, gained popularity among end users, offering free and open-source alternatives to commercial operating systems like Windows and macOS.

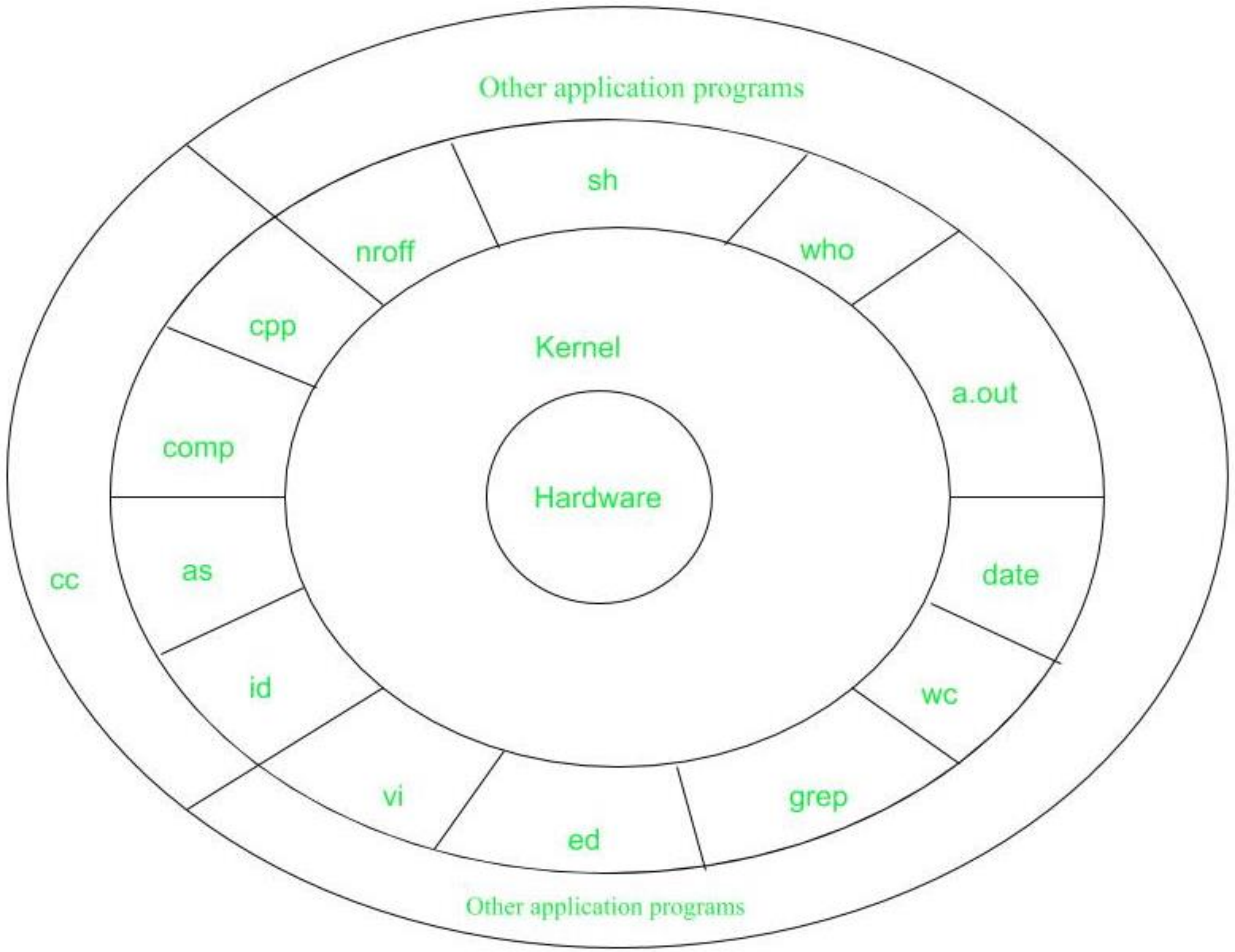
Open source: Development

Cloud: Linux became a fundamental part of cloud computing platforms and container technologies like Docker and Kubernetes.

Community and Collaboration: The open-source development model and the Linux community have continued to grow, with countless developers and organizations contributing to the kernel and related projects.

UNIX structure

1. Hardware
2. Kernel
3. Shell
4. Application



Advantages of UNIX

Stability: It can run for long periods of time without requiring a reboot, which makes it ideal for critical systems that need to run continuously.
Security: UNIX has a robust security model that includes file permissions, user accounts, and network security features.
Scalability: UNIX can be scaled up to handle large workloads and can be used on a variety of hardware platforms.
Flexibility: UNIX is highly customizable and can be configured to suit a wide range of needs. It can be used for everything from simple desktop systems to complex server environments.
Command-line interface: UNIX's command-line interface allows for powerful and efficient interaction with the system.

Windows

Characteristics:

1. Graphical User Interface (GUI)
2. Multitasking Capability
3. Virtual Memory Management
4. Security and Privacy Features
5. Interoperability (interoperability is a characteristic of a product or system to work with other products or systems)
6. Device Management
7. Cloud Integration
8. Software management
9. Customization

Open source

Advantages

1. Reliable and efficient
2. Cost-efficient
3. Flexibility
4. Community support

Important about Linux

1. GUI text editors:
-There are various text editors depending on the desktop environment you're using.
a. Gedit: This is the default text editor for the GNOME desktop environment. You can open it by running command gedit
b. Kate: If you're using the KDE desktop environment, you can use the Kate text editor:
c. Pluma: If you're using the MATE desktop environment, you can use the Pluma text editor:
d. Mousepad: For the Xfce desktop environment, you can use the Mousepad text editor:

Important about Linux

2. Text editor within Linux command line:
-If you want to open a text editor within a terminal window in Linux,
you can use command-line text editors.
Nano: Nano is a simple and easy-to-use text editor.
Example: nano myfile

Important about Linux

3. Linux compiler and debugger:

a. GCC (GNU Compiler Collection) is a standard compiler in Linux

-To check version: gcc –version

-To compile program

gcc –g –o output filename.c

./output

b. GDB (Gnu DeBugger) is a Standard debugger program in Linux

Find me

9851083215

Santosh.it288@mail.com

www.phtechno.com

Kathmandu