CHAPTER 3
OPERATING SYSTEM SUPPORT

BY: ANKU JAISWAL
ASST. PROFF. IOE PULCHOWK CAMPUS

CONTENT

3.1 The operating system layer

3.2 Protection

3.3 Process and threads

3.4 Communication and invocation

3.5 Operating system architecture

OPERATING SYSTEM

An Operating System (OS) is an **interface between a computer user and computer hardware.**

An operating system is a software that performs all the basic tasks like :

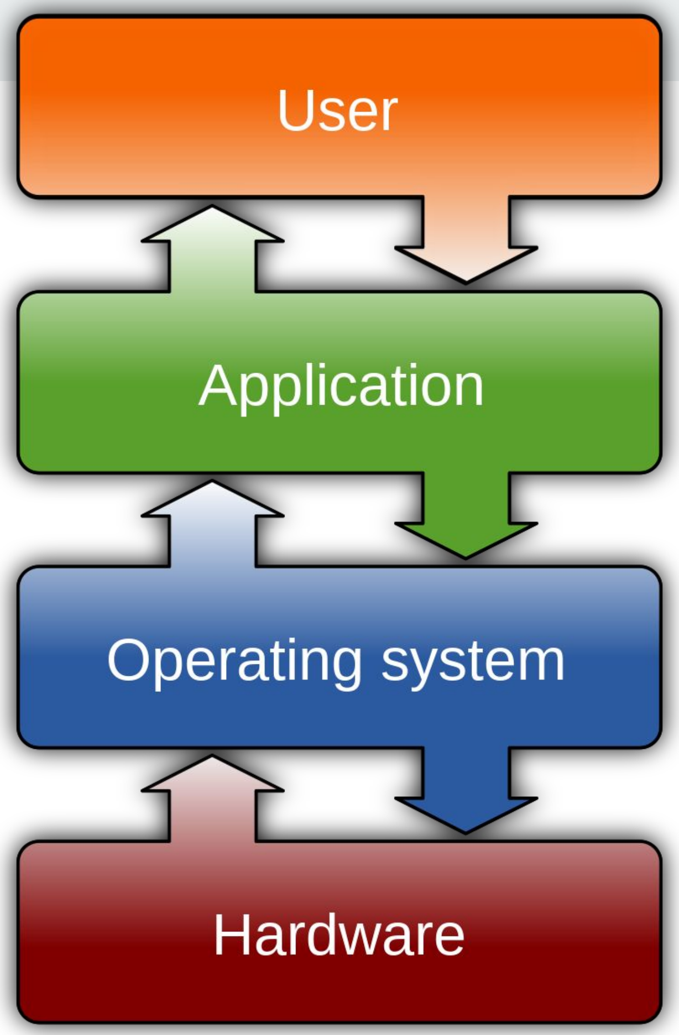
file management,

memory management,

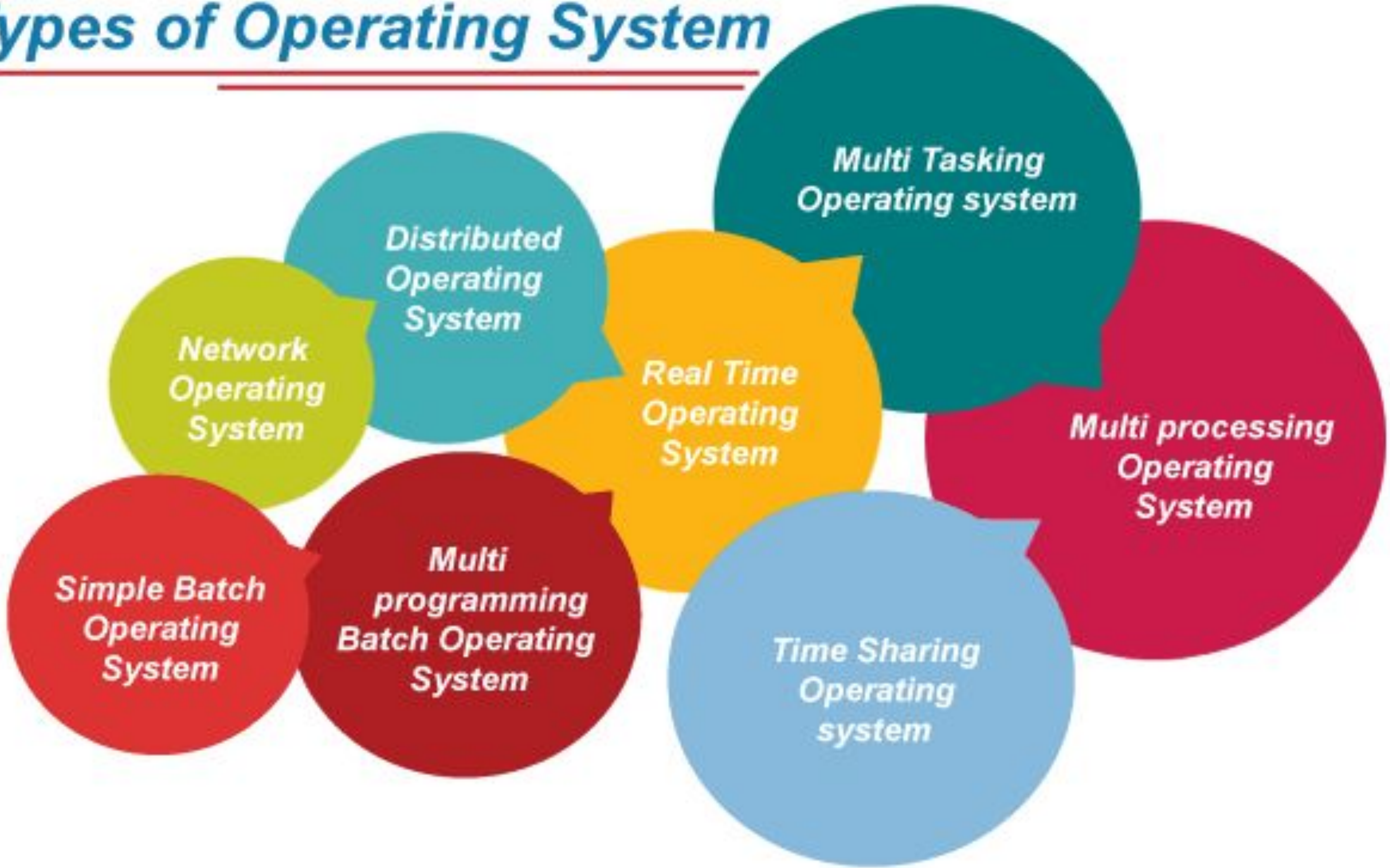
process management,

handling input and output, and

controlling peripheral devices such as disk drives and printers.


User
Application
Operating system
Hardware

Types of Operating System



Multi Tasking Operating system

Distributed Operating System

Network Operating System

Real Time Operating System

Multi processing Operating System

Simple Batch Operating System

Multi programming Batch Operating System

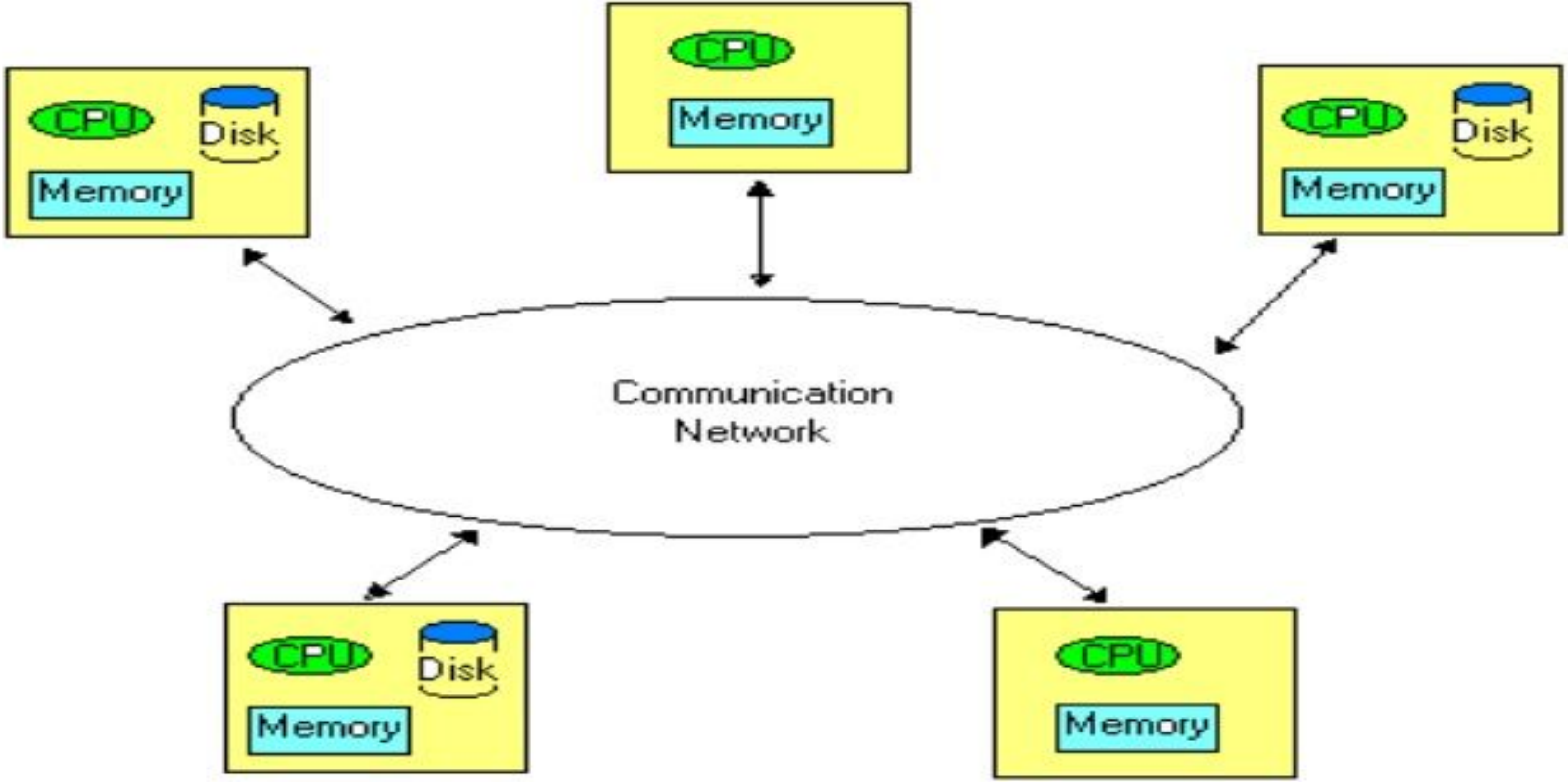
Time Sharing Operating system

DISTRIBUTED OS(DOS)

Distributed operating system is an OS which is distributed on number of computational nodes which are connected with each other through network.

This operating system consists of numerous computers, nodes, and sites joined together via **LAN/WAN** lines.

Architecture of Distributed OS



TYPES OF DOS

1. **Client-Server Systems**
2. **Peer-to-Peer Systems**
3. **Middleware**
4. **Three-tier**
5. **N-tier**

EXAMPLE OF DOS

Solaris

It is designed for the SUN multiprocessor workstations

Micros

The MICROS operating system ensures a balanced data load while allocating jobs to all nodes in the system.

DYNIX

It is developed for the Symmetry multiprocessor computers.

Locus

It may be accessed local and remote files at the same time without any location hindrance.

Mach

It allows the multithreading and multitasking features.

APPLICATION OF DOS

Network Applications

Telecommunication Networks

Parallel Computation

Real-Time Process Control

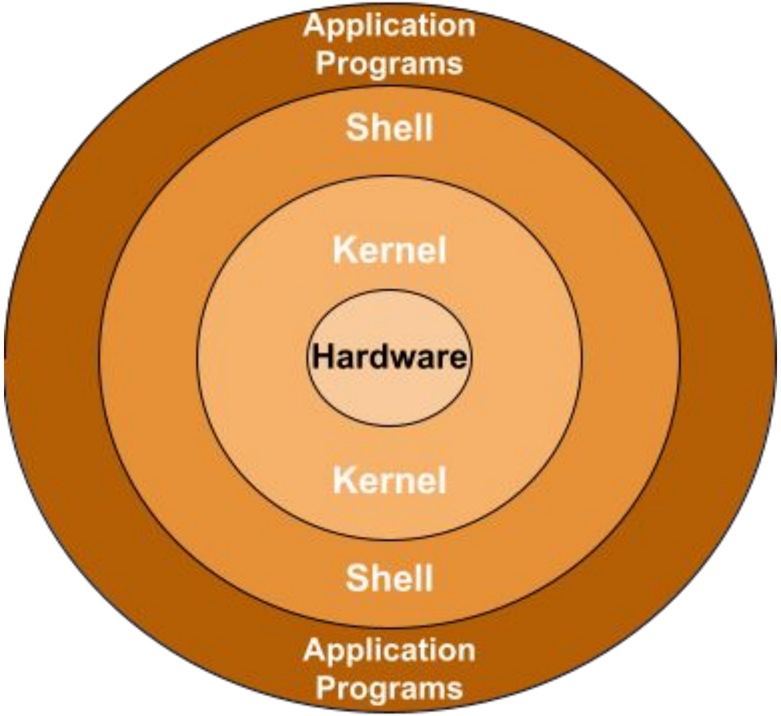
ADVANTAGES

1. It may share all resources (CPU, disk, network interface, nodes, computers, and so on) from one site to another, **increasing data availability** across the entire system.
2. It reduces the probability of data corruption because all **data is replicated** across all sites; if one site fails, the user can access data from another operational site.
3. The **entire system operates independently** of one another, and as a result, if one site crashes, the entire system does not halt.

DISADVANTAGES

1. It is **hard to implement adequate security** in DOS since the nodes and connections must be secured.
2. The **database connected to a DOS is relatively complicated** and hard to manage in contrast to a single-user system.
3. The underlying **software is extremely complex** and is not understood very well compared to other systems.

3.1. OPERATING SYSTEM LAYER



Hardware Layer:

This is the lowest layer, directly interacting with the physical hardware components such as CPU, memory, storage devices, and peripheral devices like keyboards, mice, and printers.

Kernel Layer:

The kernel is the core component of the operating system. It provides essential services, such as process management, memory management, device management, and system calls.

The kernel manages the communication between hardware and software components, ensuring that programs can run efficiently and access hardware resources securely.

Operating System Services Layer:

This layer builds on top of the kernel and provides higher-level services to applications and users.

These services may include file management, networking, security, user interface management, and system utilities.

Application Layer:

At the topmost layer are the user applications.

These are the programs that users interact with directly to perform various tasks like word processing, web browsing, email, gaming, etc.

Applications rely on the services provided by the lower layers of the operating system to access hardware resources and perform their functions.

ADVANTAGES

1. **Modularity**
2. **Easy debugging**
3. **Easy update**
4. **No direct access to hardware**
5. **Abstraction**

3.2. PROTECTION IN OS

Protection is a method that limits the access of programs, processes, or users to the resources defined by a computer system.

Protection can be used to allow several users to safely share a common logical namespace, such as a directory or files, in multi-programming operating systems.

NEED OF PROTECTION

1. There may be security risks like unauthorized reading, writing, modification, or preventing the system from working effectively for authorized users.
2. It helps to ensure data security, process security, and program security against unauthorized user access or program access.
3. It is important to ensure no access rights' breaches, no viruses, no unauthorized access to the existing data.
4. Its purpose is to ensure that only the systems' policies access programs, resources, and data.

GOALS OF PROTECTION

1. The policies define how processes access the computer system's resources, such as the CPU, memory, software, and even the operating system.
2. Protection is a technique for protecting data and processes from harmful or intentional infiltration.
3. It contains **protection policies** either established by itself, **set by management** or imposed individually by programmers to ensure that their programs are protected to the greatest extent possible.

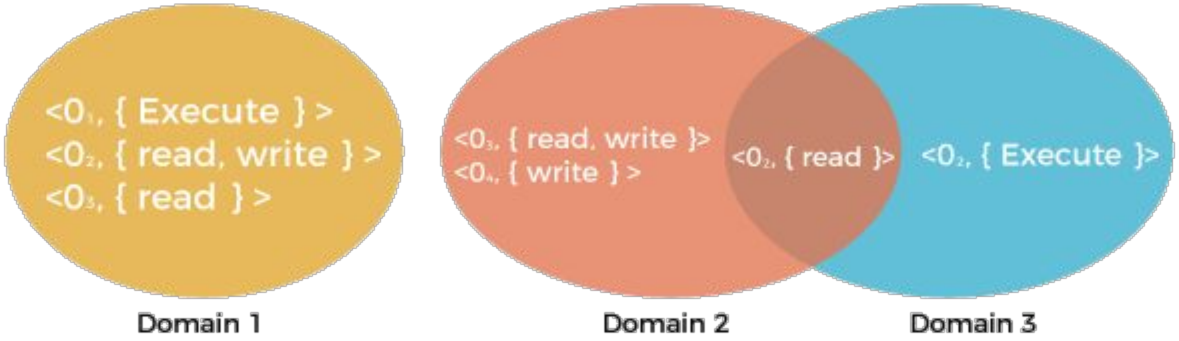
DOMAIN OF PROTECTION

1. The protection policies restrict each process's access to its resource handling.
2. A process is obligated to use only the resources necessary to fulfil its task **within the time constraints and in the mode in which it is required.**
3. It is a **process's protected domain.**

Processes and objects are abstract data types in a computer system, and these objects have operations that are unique to them.

A domain component is defined as **<object, {set of operations on object}>**.

Domain of Protection



3.3. PROCESS AND THREADS

PROCESS

A Program does nothing unless its instructions are executed by a CPU.

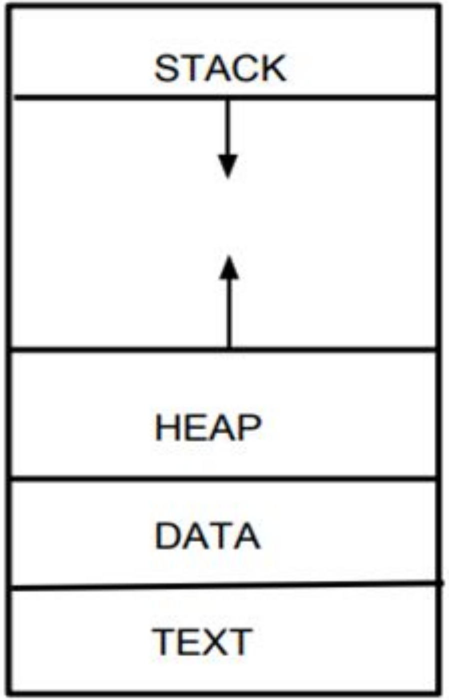
A program in execution is called a process.

In order to accomplish its task, process needs the computer resources.

There may exist more than one process in the system which may require the same resource at the same time.

Some resources may need to be executed by one process at one time to maintain the consistency otherwise the system can become inconsistent and deadlock may occur.

Layout of process inside main memory



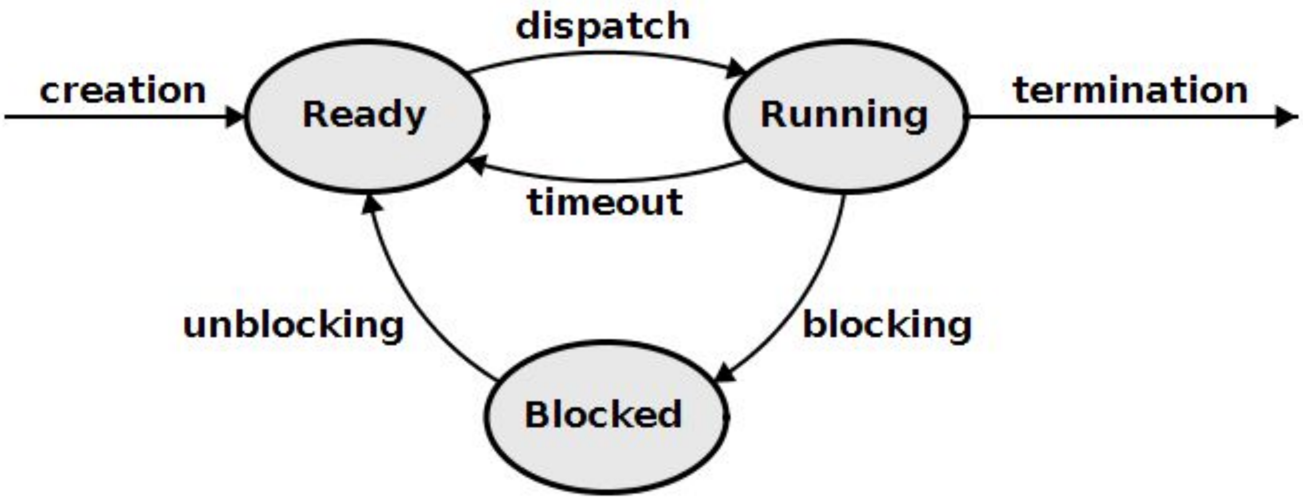
Stack: This section contains local variable, function and returns address.

Heap: This Section is used to provide dynamic memory whenever memory is required by the program during runtime It is provided form heap section.

Text: This section contains the executable instruction, constants etc

Data: This Section contains the global variables and static local variables.

PROCESS LIFE CYCLE



The life-cycle of a process can be described by a state diagram which has states representing the execution status of the process at various times .

A process is created .

Whenever a process is created, it directly enters in the ready state, in which, it waits for the CPU to be assigned.

One of the processes from the ready state will be chosen by the OS depending upon the scheduling algorithm.

From the Running state, a process can make the transition to the block or wait state depending upon the scheduling algorithm or the intrinsic behavior of the process.

A process terminates or ends.

THREADS

A thread is the smallest unit of processing that can be performed in an OS.

Think of MS Word application, which is a process that runs on computer.

But an application can do more than one thing at a time, which means that a given process in an operating system can have one or more **threads**.

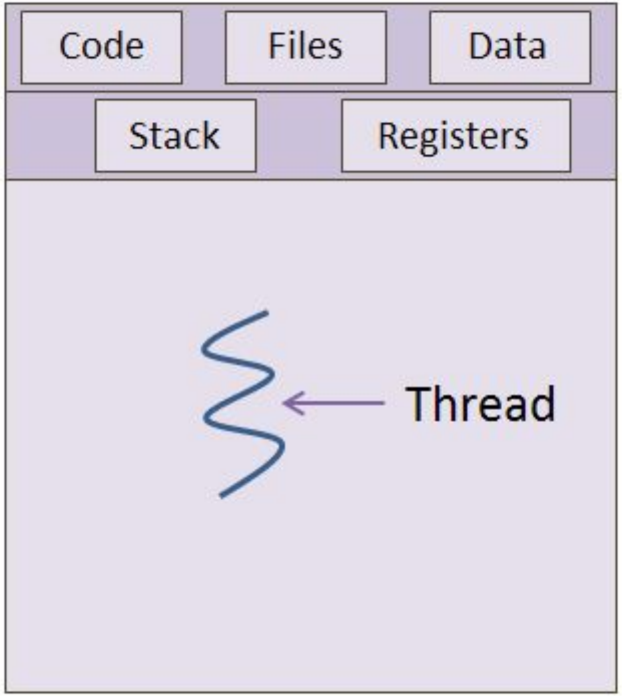
For example, in a browser, many tabs can be viewed as threads.

ADVANTAGE OF THREAD OVER PROCESS

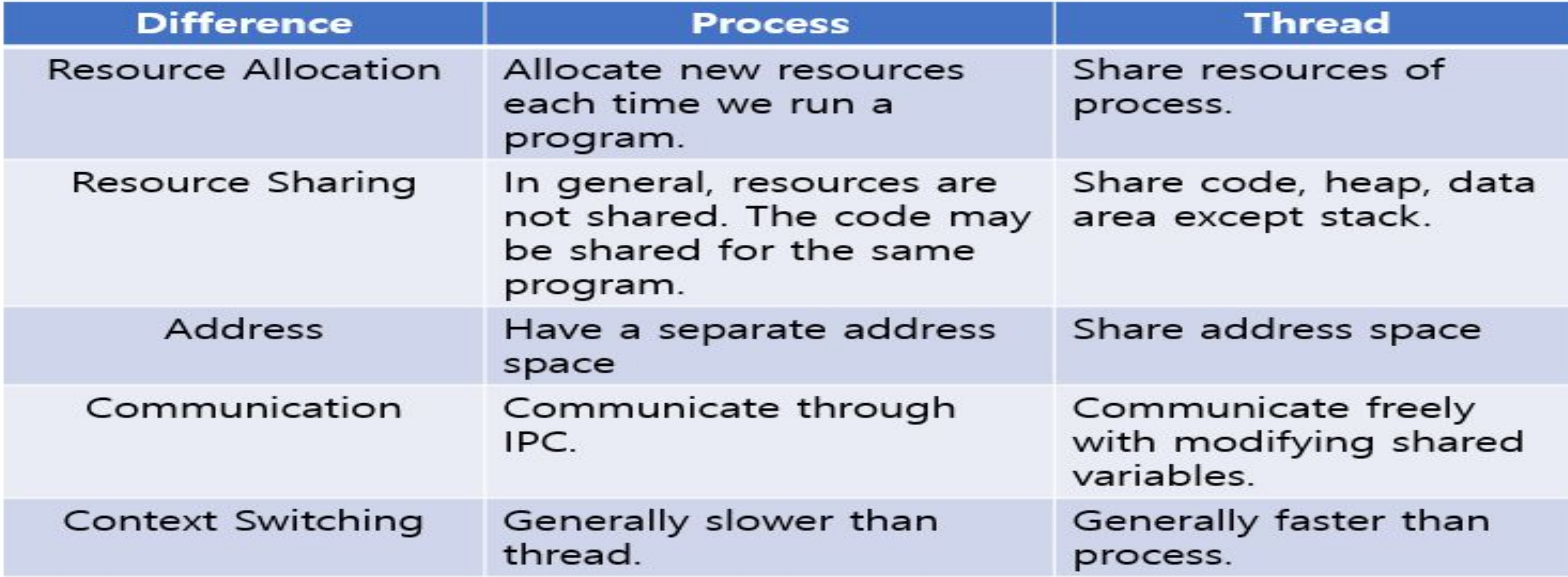
1. *Responsiveness:*
2. *Faster context switch:*

3. *Resource sharing:*

4. *Communication:*


Code
Files
Data
Stack
Registers
Thread

PROCESS VS THREADS



Difference	Process	Thread
Resource Allocation	Allocate new resources each time we run a program.	Share resources of process.
Resource Sharing	In general, resources are not shared. The code may be shared for the same program.	Share code, heap, data area except stack.
Address	Have a separate address space	Share address space
Communication	Communicate through IPC.	Communicate freely with modifying shared variables.
Context Switching	Generally slower than thread.	Generally faster than process.

3.4. COMMUNICATION AND INVOCATION

-RPC

-RMI

3.5. OPERATING SYSTEM ARCHITECTURE

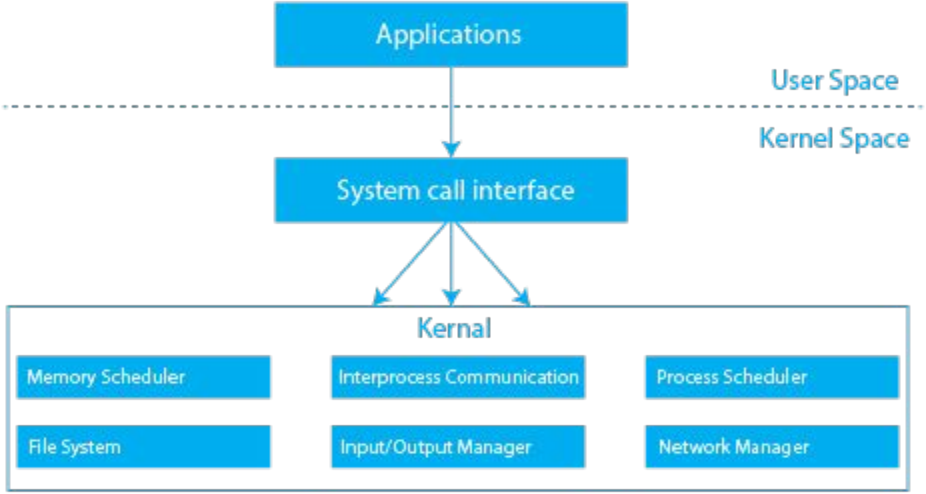
- Monolithic Architecture
- Layered Architecture
- Microkernel Architecture
- Hybrid Architecture

Monolithic Architecture

Monolithic Architecture is the oldest and the simplest type of Operating System Architecture.

In this architecture, each and every component is contained in a single kernel only.

The various components in this OS Architecture communicate with each other via function calls.


Applications
User Space
Kernel Space
System call interface
Kernal
Memory Scheduler
Interprocess Communication
Process Scheduler
File System
Input/Output Manager
Network Manager

Advantages of Monolithic Architecture

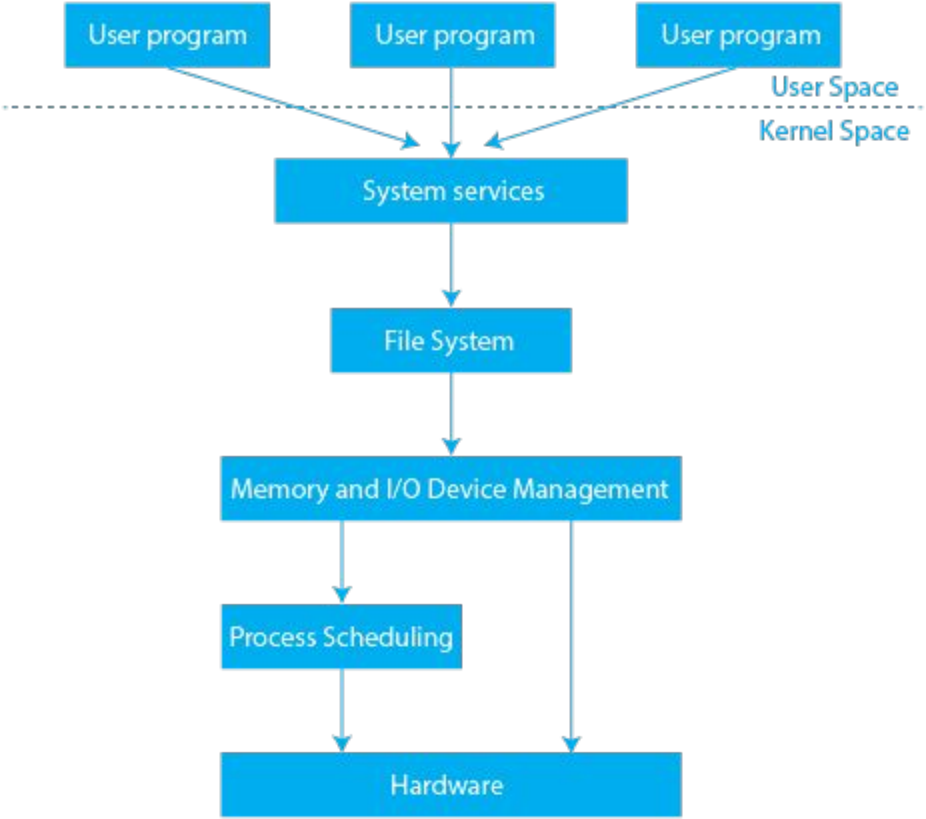
- This type of architecture is easier to develop.
- Easy to maintain.
- It shows efficient performance because of the direct communication between the components.
- It is easy to secure as all the components are contained in a single file.

Layered Architecture

In a layered architecture, the operating system is divided into layers, with each layer performing a specific set of functions.

The layers are organized in a hierarchical order, with each layer depending on the layer below it.

The layering approach makes the system easier to maintain and modify, as each layer can be modified independently without affecting the other layers.


User program
User program
User program
User Space
Kernel Space
System services
File System
Memory and I/O Device Management
Process Scheduling
Hardware

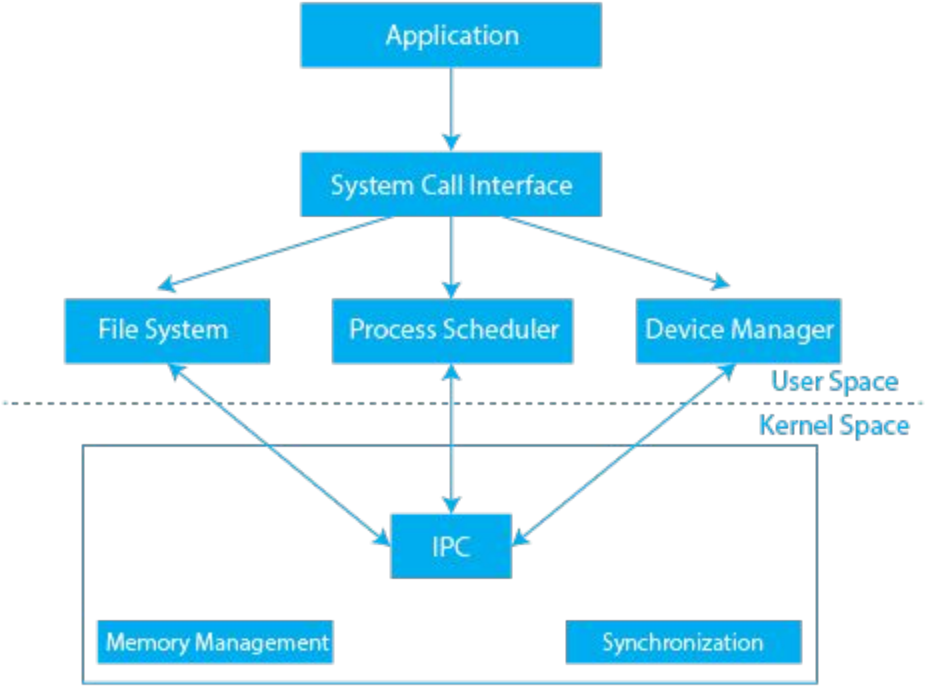
Advantages of Layered Architecture

- Separation of concerns makes it easier to develop and maintain individual layers
- Components within a layer can be swapped out without affecting other layers
- Scalability is improved because layers can be scaled independently

Microkernel Architecture

Process management, networking, file system interaction, and device management are executed outside the kernel in this architecture, while memory management and synchronization are executed inside the kernel.

The processes inside the kernel have a relatively high priority, and the components are highly modular, so even if one or more components fail, the operating system continues to function.

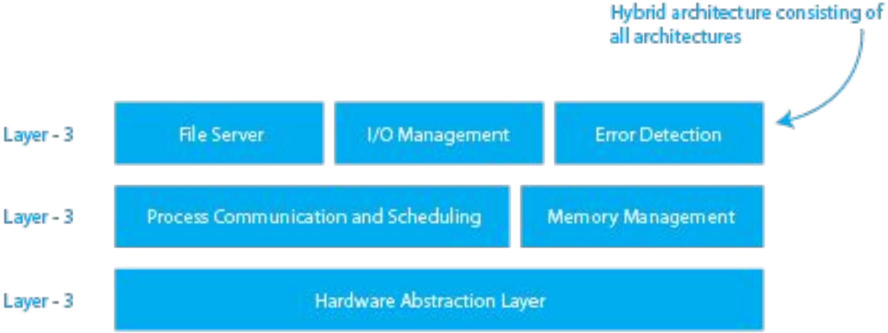

Application
System Call Interface
File System
Process Scheduler
Device Manager
User Space
Kernel Space
IPC
Memory Management
Synchronization

Advantages of Microkernel Architecture

- A highly modular design makes it easier to develop and maintain individual components
- Can support a wide range of operating systems and hardware platforms
- Enables customization and flexibility, allowing for the creation of tailored systems

Hybrid Architecture

As the name implies, hybrid architecture is a hybrid of all the architectures discussed thus far, and therefore it contains characteristics from all of those architectures, which makes it highly valuable in modern operating systems.



Hybrid architecture consisting of all architectures

Layer - 3

File Server

I/O Management

Error Detection

Layer - 3

Process Communication and Scheduling

Memory Management

Layer - 3

Hardware Abstraction Layer

Advantages of Hybrid Architecture

- Combines the benefits of multiple architectures, such as microkernel and monolithic kernel, allowing for better performance, scalability, and flexibility.
- Offers a higher level of security by isolating critical components in separate modules and reducing the attack surface.
- Allows for easier integration of different software components, as it supports multiple programming paradigms and facilitates communication between them.