Nepal College of Information Technology

Affiliated to Pokhara University

Presentation on

:

Operating System Structure and kernel types

OPERATING SYSTEM

Definition of Operating System:

- An operating system (OS) is a collection of software that manages computer hardware resources and provides common services for computer programs.
- Examples: Windows, Linux, Unix and Mac OS, etc.,
- In simple terms, an operating system is an interface between the computer user and the machine.
- Basic function of the OS :-

1. Memory Management
2. Processer Management
3. Device Management
4. File Management
5. Resource Management

OPERATING SYSTEM

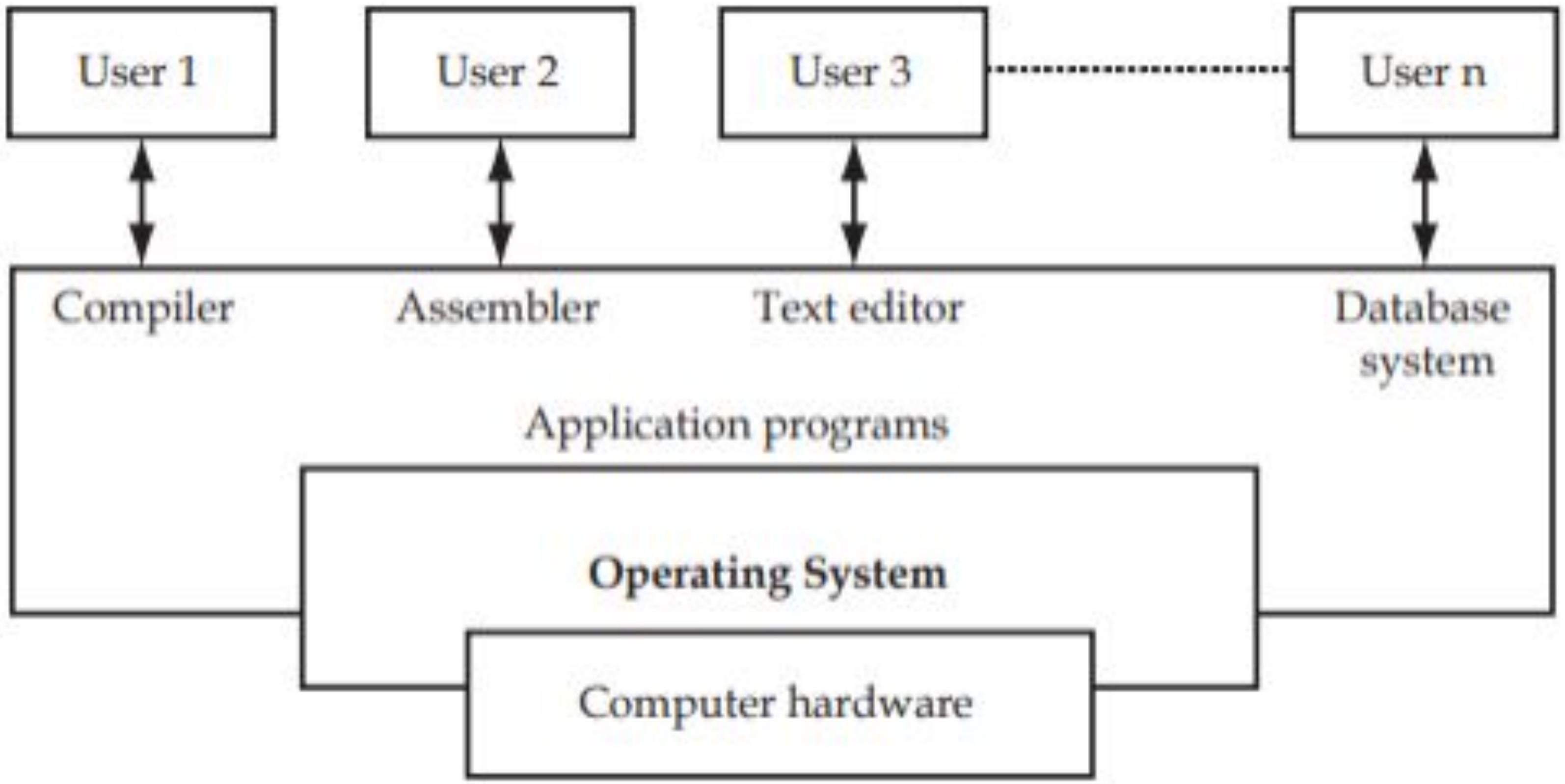


Fig: Abstract View of Components of Computer System

OPERATING SYSTEM STRUCTURES

- An operating system might have many structures.
- According to the structure of the operating system; operating systems can be classified into many categories.
- Some of the main structures used in operating systems are:-

1.Monolithic Architecture

- **Monolithic kernel** is an operating system architecture where the entire operating system is working in kernel space.
- This increases the size of the kernel as well as the operating system.
- All the basic services of OS like **process management, file management, memory management, exception handling, process communication** etc. are all present inside the kernel only.

MONOLITHIC ARCHITECTURE

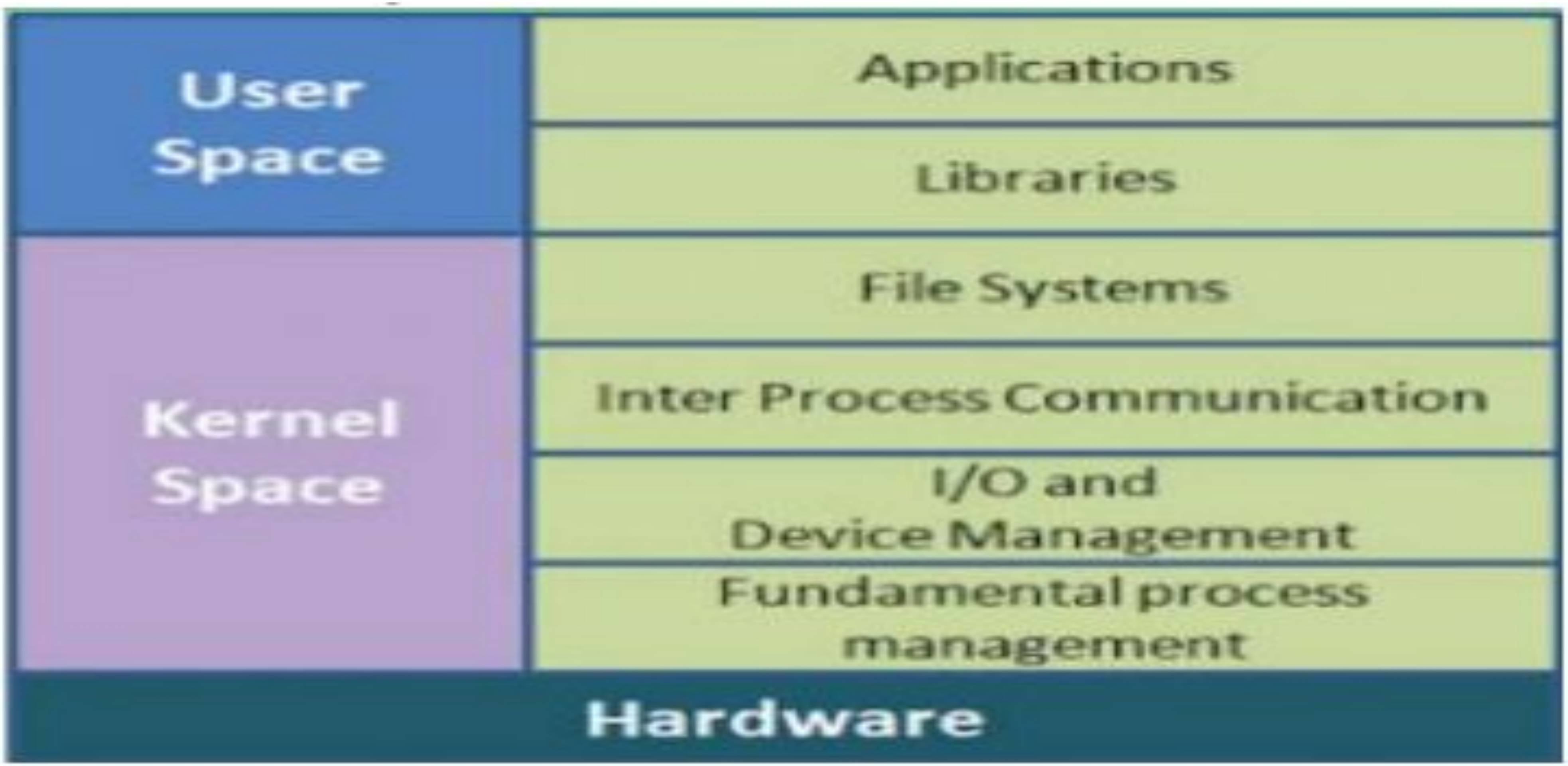
Advantages of Monolithic Kernel:

- The execution of the monolithic kernel is quite fast as the services such as memory management, file management, process scheduling, etc., are implemented under the **same address space.**
- A process runs completely in a single address space in the monolithic kernel.
- All the components can directly communicate with each other and also with the kernel.

Disadvantages of Monolithic Kernel:

- If any **service fa**ils in the monolithic kernel, it leads to the failure of the entire system.
- To add any new service, the entire operating system needs to be **modified** by the user.

MONOLITHIC ARCHITECTURE



OPERATING SYSTEM STRUCTURES

2. Layered System:

- This is an important architecture of operating system which is meant to overcome the disadvantages of early monolithic systems.
- In this approach, OS is split into various layers such that all the layers perform different functionalities.
- Each layer can interact with the one just above it and the one just below it.

layer 5:	user programs
layer 4:	buffering for input and output
layer 3:	Process management
layer 2:	memory management
layer 1:	CPU scheduling
layer 0:	hardware

LAYERED SYSTEM

Advantages of Layered System :

- Easier testing and debugging due to isolation among the layers.
- Adding new functionalities or removing outdated ones is very easy

Disadvantages of Layered System :

- Sometimes, a large no. of functionalities is there and number of layers increase greatly.
- This might lead to degradation in performance of the system.

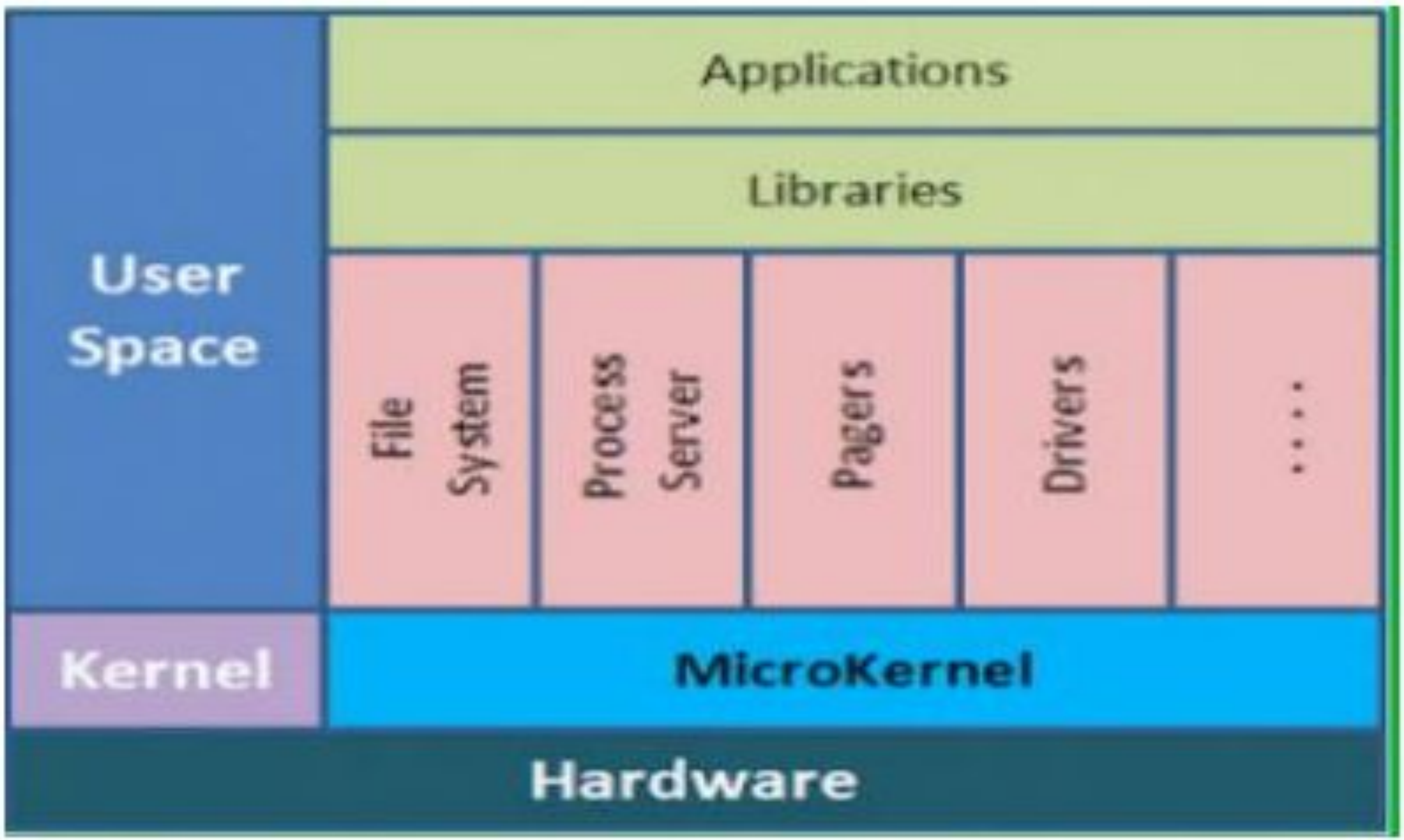
Example of Layered Architecture: Microsoft Windows NT operating System

OPERATING SYSTEM STRUCTURES

3. Microkernels:

- The basic ideology in this architecture is to keep the **kernel as small as possible**.
- We know that kernel is the core part of the operating system and hence it should be meant for handling the most important services only.
- In microkernel architecture, only the most **important services** are put inside the kernel and **rest of the** OS service are present in the system application program
- Microkernel is responsible for the three most important services of operating system namely: **Inter-Process communication, Memory management and CPU scheduling.**
- Microkernel and system applications can **interact** with each other by message passing as and when required.

MICROKERNELS



MICROKERNELS

Advantages:

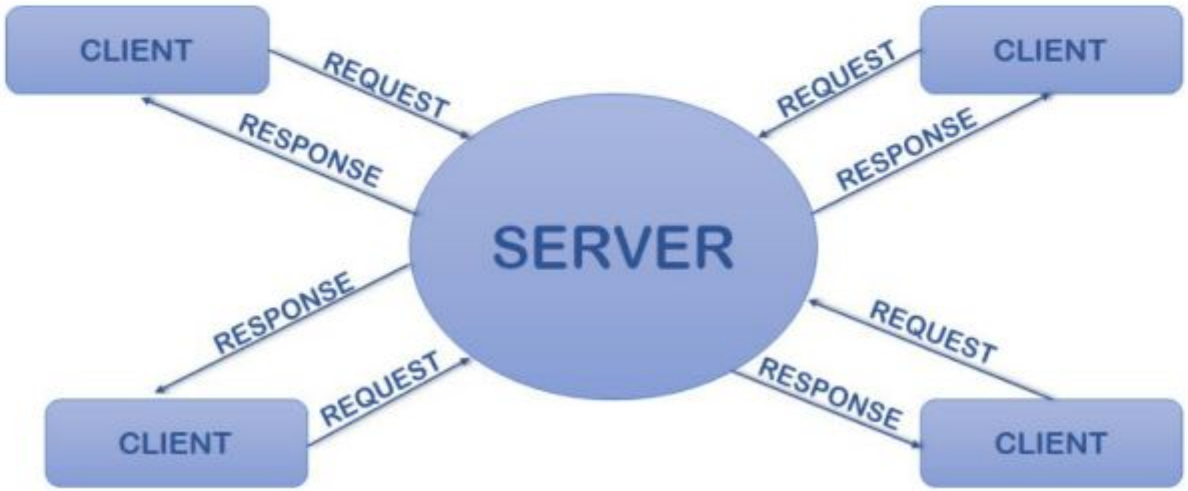
- Kernel is small and isolated and can hence function better
- Expansion of the system is easier, it is simply added in the system application without disturbing the kernel.

Disadvantages:

- Providing services in a microkernel system are expensive compared to the normal monolithic system.
- The performance of a microkernel system can be indifferent and may lead to some problems

4. Client Server Model

- **Two classes of processes -** Server and Clients
- Communication between client and server is via message passing
- Client and server can run on different computers connected by LAN/WAN Servers run as user mode.
- Hence, no system down even if the server crashed
- Well adopted in distributed system • E.g. Windows NT

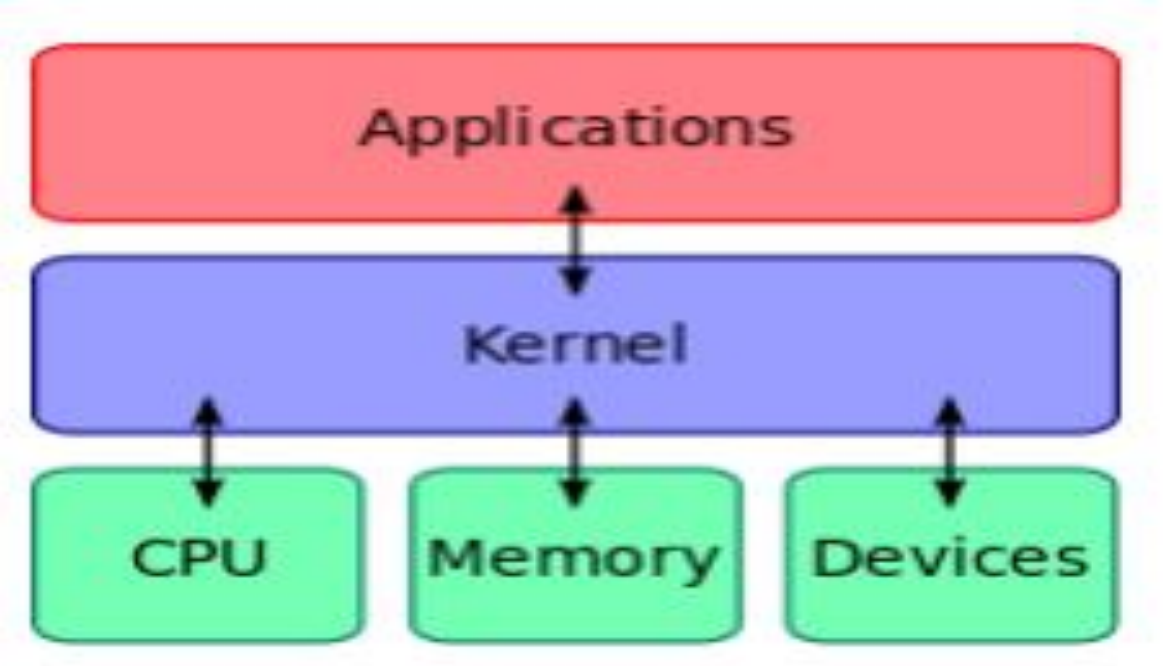


KERNEL AND ITS TYPE

KERNEL

Introduction and Architecture of a Kernel:

- A kernel is a core (central) component of an operating system.
- It acts as an interface between the user applications and the hardware.
- The kernel's **primary function** is to manage the computer's hardware and resources and allow other programs to run and use these resources



KERNEL

- Kernels also usually provide methods for synchronization and communication between processes called inter process communication (IPC).

- **<u>The main tasks of the kernel are:</u>**

1. Process management
2. Device management
3. Memory management
4. interrupt handling
5. I/O communication
6. File system...etc

KERNEL TYPE’S

Types of Kernels

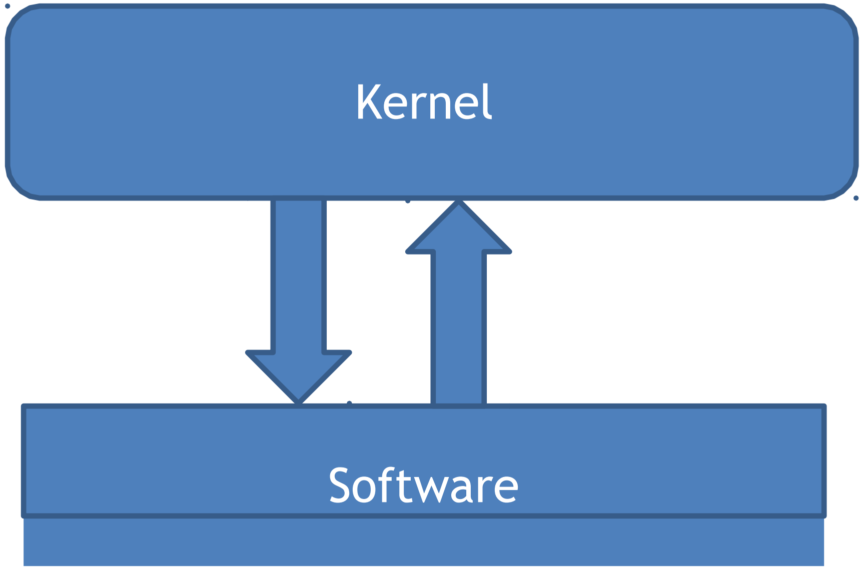
- Kernels may be classified into different categories :-

1) Monolithic Kernel:

- A Monolithic kernel is a single large processes running entirely in a single address space
- This design offers **high performance** but has **less modularity** and can be harder to maintain compared to microkernels.

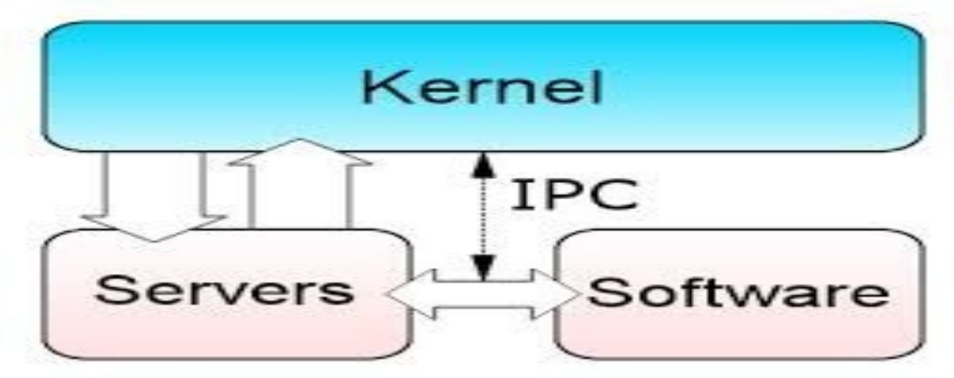
MONOLITHIC KERNEL:

- The kernel can invoke functions directly
- The examples of monolithic kernel based OSs are **Linux, Unix.**



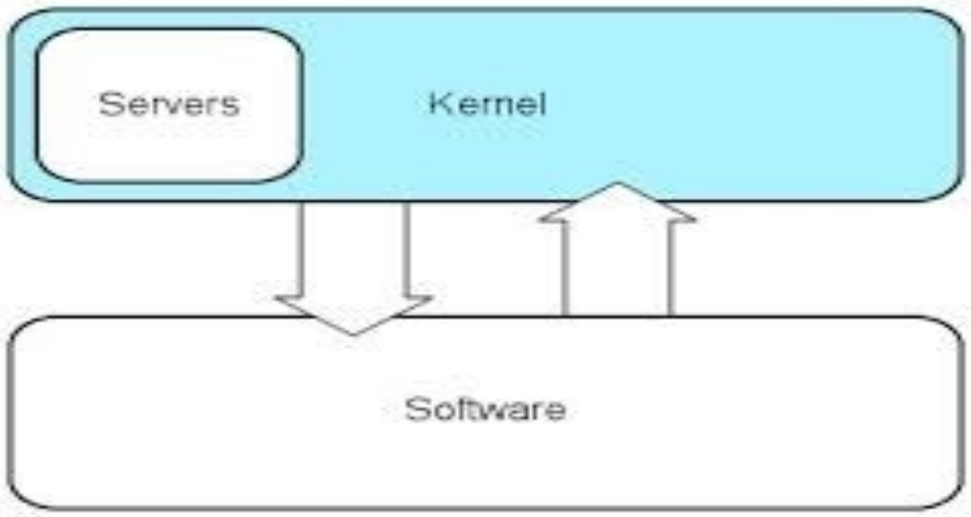
2)MICRO KERNEL:

- In Microkernels, the kernel is **broken down** into separate processes, known as servers.
- Some of the servers run in **kernel space** and some run in **user-space**
- All servers are kept separate and run in different address spaces.
- The communication in microkernels is done via message passing.
- The servers communicate through **IPC** (Interprocess Communication).



3)HYBRID KERNEL :

- Hybrid kernel is a kernel architecture based on a combination of **microkernel and monolithic kernel** architecture used in computer operating systems
- Like monolithic kernels, hybrid kernels execute most tasks in kernel space, enabling **faster performance compared to microkernels**, which involve more overhead due to IPC.
- Like microkernels, hybrid kernels allow for certain services to run in user space, making them more **modular and potentially** easier to manage and maintain.



4)EXO- KERNEL :

- An exokernel is a type of kernel that does not abstract hardware into theoretical models.
- Instead it allocates physical hardware resources
- such as processor time, memory pages, and disk blocks, to different programs.

5)NANOKERNEL :

- A nanokernel is a very minimal operating system kernel. that provides basic hardware abstraction, such as managing the CPU, handling interrupts, and interacting with the MMU(Memory Management Unit).
- without providing higher-level system services.
- By interfacing the CPU, managing interrupts and interacting with the MMU.
- The interrupt management and MMU interface are not necessarily part of a nanokernel.
- Most architecture these components are directly connected to the CPU, therefore,
- it often makes sense to integrate these interfaces into the kernel. It lack system services.

1.4 KERNEL DESIGN AND IMPLEMENTION STRATEGIES

KERNEL DESIGN AND IMPLEMENTION STRATEGIES

- Kernel design and implementation involve creating the fundamental **layer** of an operating system that manages hardware and system resources.
- It ensures that software can **interact** with hardware efficiently and securely.
- The kernel acts as an **intermediary** between the user applications and hardware.
- Below is an overview of kernel design principles and implementation steps:

Step-1 Kernel Design Principles:

a) **Abstraction Layer**: The kernel provides an abstraction layer to hide hardware complexities from the user space.

b) **Efficiency**: A good kernel design minimizes overhead to ensure that resources (e.g., CPU, memory) are used optimally.

c) **Modularity**: A kernel should be modular, allowing easy updates and changes without affecting the entire system.

d) **Security and Isolation**: The kernel should ensure that different processes and applications cannot interfere with each other, offering protection and security.

e) **Concurrency and Scheduling**: It must efficiently handle multitasking and resource sharing, managing multiple processes and threads.

Step-2 Kernel Types :

- **Monolithic Kernel**: All services (device drivers, file systems, network protocols) run in kernel space.

  Implementation: Requires careful coordination to avoid conflicts between components, since all run in privileged mode.

- **Microkernel**: Only the essential services (e.g., IPC, scheduling) run in kernel space, and others (e.g., device drivers) run in user space.

  Implementation: Involves inter-process communication (IPC) to handle services in user space, adding some overhead.

- **HYBRID KERNEL**
- **EXOKERNEL**

Step-3 Key Components in Kernel Implementation:-

1) Process Management:

Scheduler: Manages the execution of processes, deciding which process gets CPU time.

Context Switching: Saves the state of a process and restores the state of another when switching between processes.

IPC (Inter-Process Communication): Allows processes to communicate and synchronize, typically using message passing or shared memory.

2) Memory Management:

Virtual Memory: Abstracts physical memory and provides processes with their own address space.

Page Tables and MMU: Maps virtual addresses to physical memory locations, often using paging or segmentation.

Memory Allocation: Manages the allocation and deallocation of memory, often through algorithms like first-fit or best-fit

3)File System Management:

File System Abstraction: Provides a uniform interface for file operations (open, read, write, close).

Disk I/O: Manages how data is read from and written to storage devices

Metadata Management: Keeps track of file attributes such as name, size, and permissions

4)Interrupt Handling:

Interrupt Service Routines (ISR): Handle hardware interrupts and ensure that high-priority tasks are given CPU time.

Interrupt Vector Table: Maps interrupt numbers to their corresponding ISRs.

5) Security and Protection:

Access Control: Ensures that only authorized users and processes can access certain resources.

user and Kernel Mode: Keeps user processes isolated from critical kernel processes to prevent crashes or unauthorized access.

STEP-4 IMPLEMENTATION KERNEL:

- ❖**Choose the Kernel Type**: Decide whether a monolithic, microkernel, hybrid, or exokernel design is best suited for your system's needs.
- ❖**Set Up Boot Process**: Implement a boot loader that loads the kernel into memory after the system powers on.
- ❖**Develop System Calls**: Implement basic system calls that allow user programs to interact with the kernel (e.g., file operations, process control)
- ❖**Set Up Process Management**: Implement the process scheduler, context switching, and IPC mechanisms.
- ❖**Memory Management**: Implement virtual memory, paging, and memory allocation mechanisms.

•

- **Device Drivers**: Write or configure drivers for handling various hardware components (disks, network interfaces, etc.).

- **Interrupt Handling**: Implement interrupt handling routines and an interrupt vector table.

- **Security Mechanisms**: Add access control, user authentication, and other security features.

- **Testing and Debugging**: Thoroughly test the kernel for performance, security, and stability before release.

THANK YOU