KERNEL ARCHITECTURE (MONOLITHIC, MICROKERNEL)

Kernel:

The kernel is the core component of operating system, that manages the computer resources. It acts as a bridge between the hardware and the software that ensure everything runs smoothly. It enables applications to interact with the computer's resources like memory, CPU, and input/output devices.

When we switch on the computer, Kernel is the first part of the OS to load into the computer's main memory (RAM) by the bootloader. Kernel remains active from the system start up to until its is shutdown. Kernel is the only program that is **always active** while the system is running.

Operating system can be implemented with the help of various structures. The structure of the OS depends mainly on how the various common components of the operating system are interconnected and melded into the kernel

The different structure of the operating system are

1. Simple Structure
2. **Monolithic Structure**
3. Layered Approach
4. **Microkernels**
5. Modules
6. Hybrid Systems

1. **Monolithic Structure:**

The kernel performs all function such as file management, CPU scheduling, memory management, I/O management and other operating system functions through system calls.

All the functional component are included into one single kernel. Example : linux, unix, windows and macOS

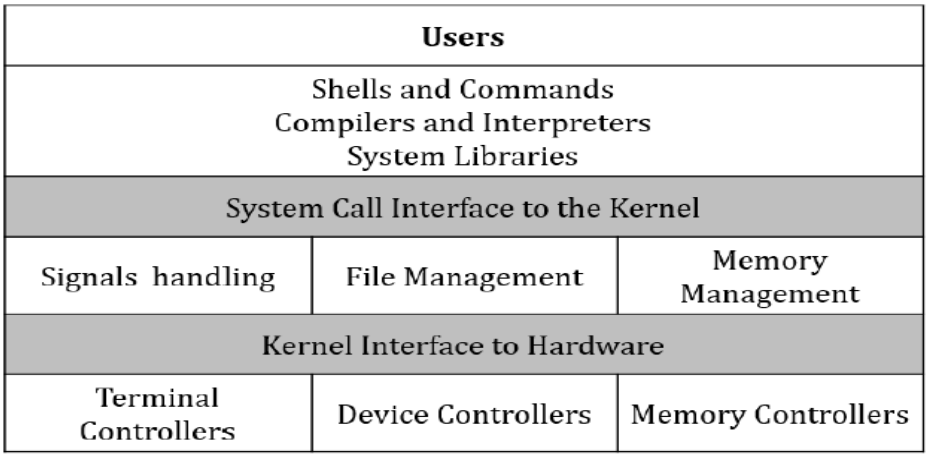


Fig: Unix - Monolithic Structure

.

The Linux operating system is based on UNIX and is structured similarly. The Linux kernel is monolithic in that it runs entirely in kernel mode in a single address space. It have a modular design that allows the kernel to be modified during run time.

Advantage:

- Simple and easy to implement.

- Execution is fast due to direct access to all services

Disadvantage:

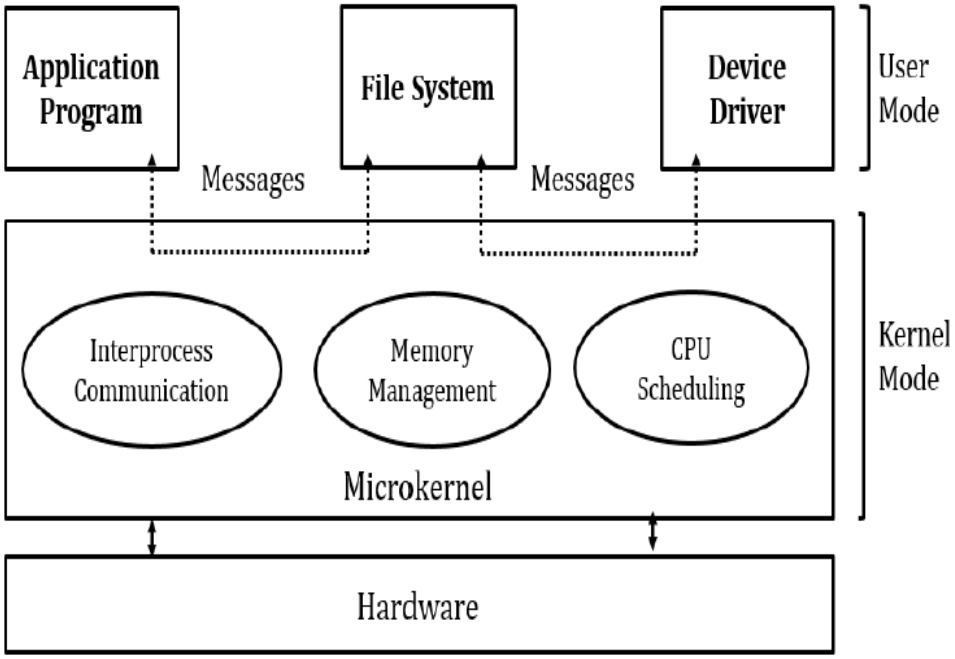
- **Reliability: If** the user program fails, the entire system may crash.
- **Complexity:** Hard to maintain and debug
- **Limited Extensibility:** Adding new functional component is difficult.
- Less protection.

2. Micro-kernel:

In this structural designs only the essential components are kept in the kernel and all non-essential components are implemented as system and user programs. This result in a smaller kernel called the micro-kernel. Example: QNX, Minix, and Symbian

Essential functions/components such as basic process and memory management, inter-process communication and other services such as file systems, device drivers

Here communication between programs is done with the help of message passing.



In the mid-1980s, researchers at Carnegie Mellon University developed an operating system called Mach that modularized the kernel using the microkernel approach.

The best-known illustration of a microkernel operating system is ***Darwin***, the kernel component of the macOS and iOS operating systems. Darwin, consists of two kernels, one of which is the Mach microkernel

Advantage:

- Small memory space is needed for the kernel.
- **Modularity**: Easy to maintain and debugging
- Easy update
- No direct access to the hardware, which decreases the system crash.
- Abstract (layers don't have the idea about other layers ie.) layers implementation detail is hidden)
- **Extensibility:** Adding new functional component is easy.

- **Flexibility :** Easy to add or remove functional components without affecting the entire system.
- More **secure and reliable**.
- **Portability**: More portable than monolithic because most of the operating system services run outside the kernel space.

Disadvantage:

- **Performance Overhead:** One user program communicates with another application/user program through IPC with the help of message passing which is an overhead or burden for kernel.
- **Complexity:** Designing and implementing the communication mechanisms between user-space processes and the kernel can be more complex.
- **Development Difficulty: This** architecture can be more difficult than developing monolithic kernels because it requires more attention in designing the communication and synchronization mechanisms.
- **Higher Resource Usage:** Microkernel architecture can use more system resources, such as memory and CPU, than monolithic kernels because it requires more communication and synchronization mechanisms