# 2

# Operating System Models

## 2.1 Introduction

Over the past several years, a number of trends affecting operating system design are witnessed and foremost among them is a move towards modularity. Operating systems such as Microsoft's Windows, IBM's OS/2, C-DAC's PARAS and others are splintered into discrete components, each having a small, well defined interface, and each communicating with others via inter-task message interface. The lowest level is the microkernel, which provides only essential OS services, such as context switching. Windows NT, for example, also includes a hardware abstraction layer (HAL) beneath its microkernel which enables the rest of the OS to perform irrespective of the processor underneath. This high level of OS portability is a primary driving force behind the modular, microkernel-based push.

## 2.2 What is New in OS Trends ?

Today's operating systems provide two fundamental services for users. First, they make the computer hardware easier to use. They create a virtual machine that differs markedly from the real machine. Indeed, the computer revolution of the last two decades is due, in part, to the success that operating systems have achieved in shielding users from the obscurities of computer hardware.

Second, an operating system shares hardware resources among users. One of the most important resource is processor. A multitasking operating systems, such as UNIX, C-DAC PARAS, Windows NT, divides the work that needs to be done among processes, giving each process memory, system resources, at-least one *thread of execution*, an executable unit within a process. The operating system runs one thread for a short time and then switches to another, running each thread in turn. Even on single-user system, multitasking is extremely helpful because it enables the computer to perform multiple tasks at once. For example, a user can edit a document while another document is printing in the background or while a compiler compiles a large program. Each process gets its work done, and to the user all the programs appear to run simultaneously.

Apart from the above benefits, the new concept in operating system services is supporting multiple threads of control in a process itself. This concept has opened the eye of Parallel Processing, the parallelism within a process, instead of across the programs. The task level parallelism (discussed above) is often not suitable for high performance parallel programming. Hence, in next-generation operating system kernels, address space and threads are decoupled so that a single address space can have multiple execution threads. Programming a process having multiple threads of control is known as multithreading. POSIX threads interface is a programming environment for multithreading, the parallelism within a process.

The structure of an operating system is dictated by the model employed in building them. An operating system model is a broad framework that unifies the many features and services the operating system provides and tasks it performs. Operating systems are broadly classified into three categories, based on the their structuring mechanism as follows:

1. Monolithic Operating System,
2. Layered Operating System, and
3. Client-Server or Microkernel Operating System.

## 2.4 Monolithic Operating System

The components of monolithic operating system are organized haphazardly and any module can call any other module without any reservation. Similar to the other operating systems, applications in monolithic OS are separated from the operating system itself. That is, the operating system code runs in a privileged processor mode (referred to as kernel mode), with access to system data and to the hardware; applications run in a non-privileged processor mode (called the user mode), with a limited set of interfaces available and with limited access to system data. The monolithic operating system structure with separate user and kernel processor mode is shown in Figure 2.1.

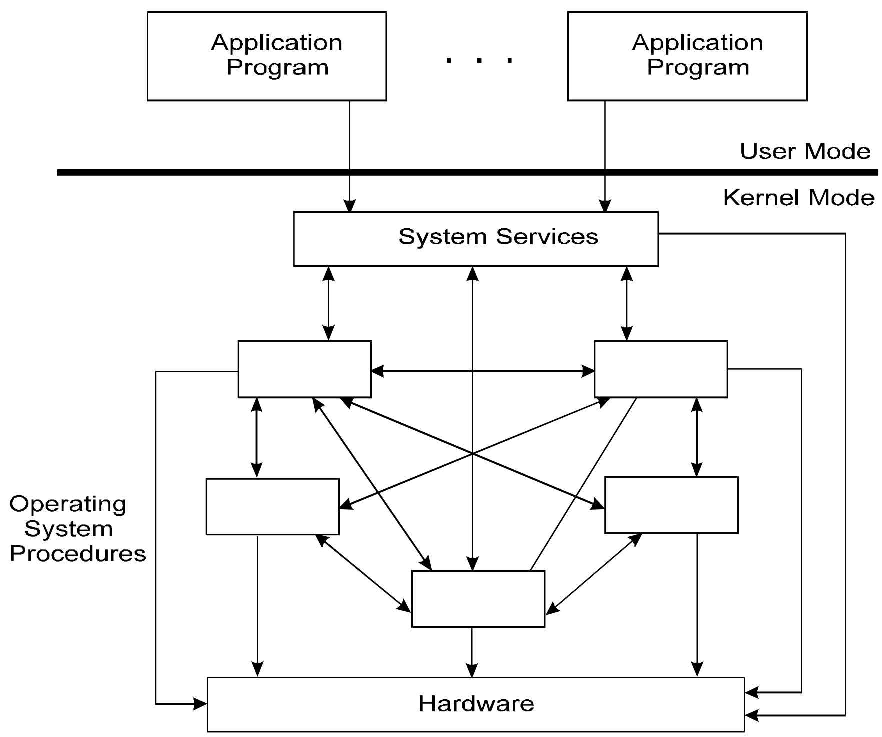


**Figure 2.1: Monolithic Operating System**

When a user-mode program calls a system service, the processor traps the call and then switches the calling thread to kernel mode. Completion of system service, switches the thread back to the user mode, by the operating system and allows the caller to continue.

The monolithic structure does not enforce data hiding in the operating system. It delivers better application performance, but extending such a system can be difficult work because modifying a procedure can introduce bugs in seemingly unrelated parts of the system.

**Example Systems:** CP/M and MS-DOS

## 2.5 Layered Operating System

The components of layered operating system are organized into modules and layers them one on top of the other. Each module provide a set of functions that other module can call. Interface functions at any particular level can invoke services provided by lower layers but not the other way around. The layered operating system structure with hierarchical organization of modules is shown in Figure 2.2.

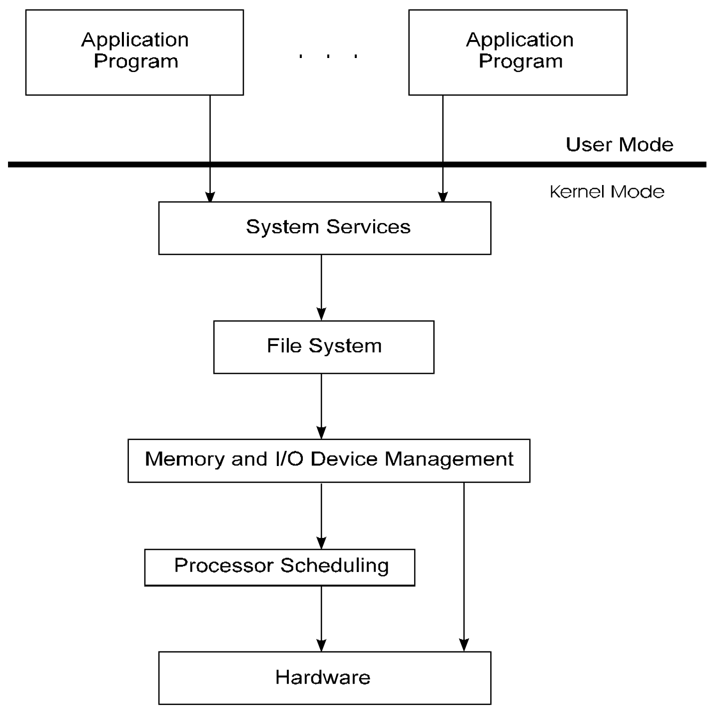


**Figure 2.2: Layered Operating System**

One advantage of a layered operating system structure is that each layer of code is given access to only the lower-level interfaces (and data structures) it requires, thus limiting the amount of code that wields unlimited power. That is in this approach, the Nth layer can access services provided by the (N-1)th layer and provide services to the (N+1)th layer. This structure also allows the operating system to be debugged starting at the lowest layer, adding one layer at a time until the whole system works correctly. Layering also makes it easier to enhance the operating system; one entire layer can be replaced without affecting other parts of the system. Layered operating system delivers low application performance in comparison to monolithic operating system.

**Example Systems:** VAX/VMS, Multics, UNIX

## 2.6 Operating System for MPP Systems

The MPP (Massively Parallel Processing) systems are built using thousands of processors. Every CPU needs operating system to manage its resources and hide details of machine architecture from the programmers. The use of heavy monolithic operating systems such as those discussed above (monolithic or layered), will unnecessarily consume lots of system resources; traditional operating systems offer a wide variety of services most of which are accessed once in a while. Loading of monolithic kernel of all CPUs will consume abundant amount of memory resource and at the same time increases complexity of a system. For instance, UNIX's filesystem offers a wide variety of services which are rarely used by a compute-intensive (parallel) applications. Such services can be *moved out of kernel* and make available in the form of user level subsystems/servers (see Figure 2.3). The remaining portion of the kernel is known as *microkernel*.

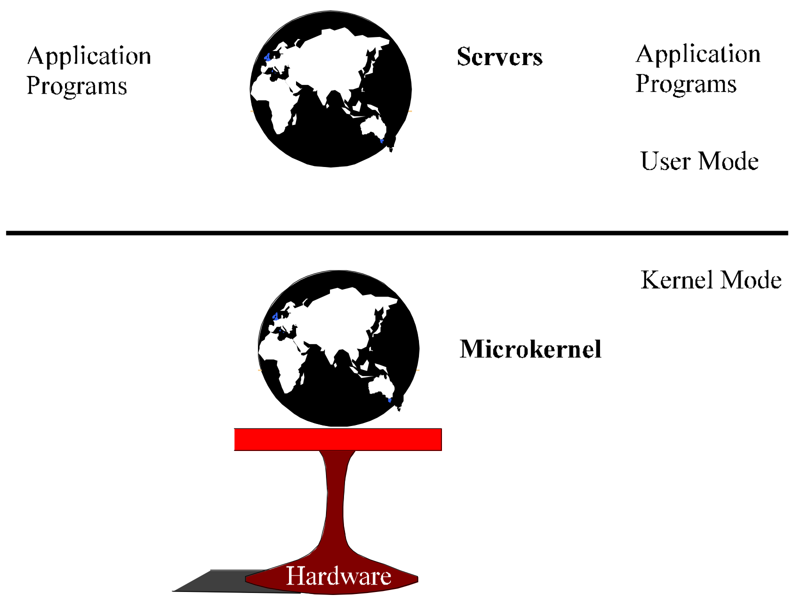


**Figure 2.3: Microkernel - New trend in OS design**

A microkernel is a tiny operating system core that provides the foundation for modular and portable extensions. Every next-generation OS will have one. In theory, a microkernel with a small privileged core surrounded by user-mode services, would deliver unprecedented modularity, flexibility, and portability. Thus OS designed using microkernel technology has the following relation:

**OS = Microkernel + User Subsystems (Servers)**

Microkernel does not necessarily mean small system. The appendage 'micro' suggests that kernel providing only a minimal function that allows user-level system processes to perform OS services efficiently. The microkernel implements essential core-operating system functions. The functions typically encompasses process management, inter-process communication, address space management, and hardware abstraction.

## 2.7 Client-Server or Microkernel Operating System

The advent of new concepts in operating system design, microkernel, is aimed at migrating traditional services of an operating system out of the monolithic kernel into the user-level process. The idea is to divide the operating system into several processes, each of which implements a single set of services - for example, I/O servers, memory server, process server, threads interface system. Each server runs in user mode, provides services to the requested client. The client, which can be either another operating system component or application program, requests a service by sending a message to the server. An OS kernel (or microkernel) running in kernel mode delivers the message to the appropriate server; the server performs the operation; and microkernel delivers the results to the client in another message, as illustrated in Figure 2.4.

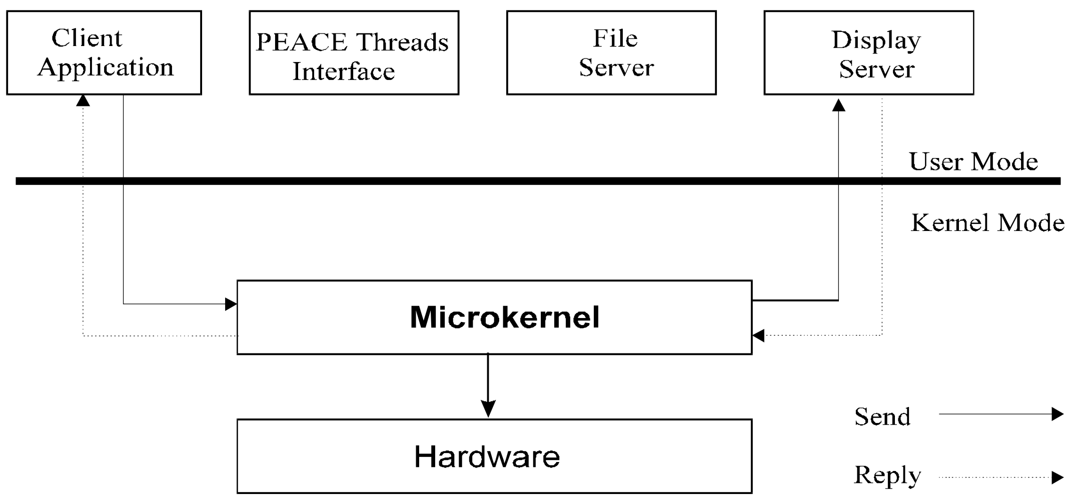


**Figure 2.4: Microkernel Operating System**

In general, services that were traditionally an integral part of the file systems, windowing systems, security services, etc. are becoming peripheral modules that interact with the kernel and each other known as subsystem. The microkernel approach replaces the vertical stratification of operating system functions with a horizontal one. Components above the microkernel communicate directly with one another, although using messages that pass through the microkernel itself. The microkernel plays a traffic cop. It validates messages, passes them between the components, and grants access to hardware.

One such implementation of microkernel based operating system is PARAS microkernel by C-DAC, India. PARAS is a operating environment for the parallel super computer developed by them. Microkernel based operating systems are gaining popularity in distributed and multiprocessor systems. When a microkernel receives a message from a process, it may handle it directly or pass the message to another process. Because microkernel need to know whether the message comes from a local or remote processes, the message passing scheme offers an elegant foundation for Remote Procedure Calls (RPC). The communication between the processes can be achieved through shared memory model or message passing model.

The subsystems running on top of the microkernel implement their own kernels which can manage the resources better than the application ignorant general-purpose conventional OS mechanism. The dominant representatives of this new generation OS technology are:

- C-DAC PARAS
- Mach - Carnegie Mellon University
- Windows NT - Microsoft Corporation
- Chorus

### Portability, Extensibility, and Reliability

With all the processor-specific code is isolated into the microkernel, changes needed to run on a new processor are fewer and grouped logically together. It makes possible for running an operating system on more than one processor. With increasingly complex monolithic systems, its extension becomes difficult, if not impossible, to ensure reliability. The microkernel's limited set of well-defined interfaces enables orderly growth and evolution.

The subsystems running on top of microkernel can be customized to meet the needs of the new platform which enhances the portability of the system. The PEACE threads interface is implemented as a run-time subsystem on top of the C-DAC PARAS microkernel operating system.

The debatable issue in the microkernel OS is in the proper division of functionality between the microkernel and its surrounding modules (subsystem).

It would be a wrong conclusion if microkernels are considered as small in terms of code complexity. The microkernels appears on the (research/commercial) market the complexity of which is significantly greater than the old-fashioned, monolithic system.

**Example Systems:** C-DAC PARAS, Windows NT/95, Mach, QNX, Chorus

### Characteristics of Microkernel OS

The most important characteristics of microkernel are listed below:

- Simplified base OS
- Traditional services of OS have become peripheral
- Improved Reliability
- Vertical style access instead of Horizontal
- Message Passing Facilities
- Leads to a Distributed Computing Model (Transparent Local or Remote Services)
- Subsystems - POSIX, Database, File, Network Server, etc.
- Monolithic application performance competence
- Foundations for Modular and Portable Extensions