STRUCTURE OF OPERATING SYSTEM

CHAPTER OUTLINE

After comprehensive study of this chapter, you will be able to:

- Introduction
- Layered System
- Kernel, Types of Kernel (Monolithic/Macro Kernel and Micro/ Exo-kernel)
- Client-server Model
- Virtual Machines, Shell

INTRODUCTION

Many commercial operating systems do not have well defined structures. Such systems started as small, simple, and limited systems and then grew beyond their original scope. MS-DOS is an example of such system. In MS-DOS, the interface and the levels of functionality are not well separated. Application programs are able to access the basic I/O routines to write directly to the display and disk drives. Such program leaves MS-DOS vulnerable to malicious programs, causing entire system crashes when user program fails.

SIMPLE STRUCTURE

There are many operating systems that have a rather simple structure. These started as small systems and rapidly expanded much further than their scope. A common example of this is MS-DOS. It was designed simply for a niche amount for people. There was no indication that it would become so popular. An image to illustrate the structure of MS-DOS is as follows:

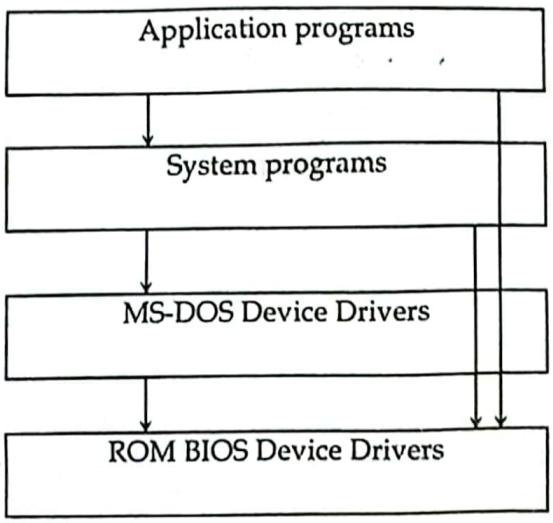


Fig. 2.1: MS-DOS Structure

It is better that operating systems have a modular structure, unlike MS-DOS. That would lead to greater control over the computer system and its various applications. The modular structure would also allow the programmers to hide information as required and implement internal routines as they see fit without changing the outer specifications.

LAYERED STRUCTURE

One way to achieve modularity in the operating system is the layered approach. This approach breaks up the operating system into different layers. In this, the bottom layer is the hardware and the topmost layer is the user interface.

This allows implementers to change the inner workings, and increases modularity. As long as the external interface of the routines doesn't change, developers have more freedom to change the inner

workings of the routines. With the layered approach, the bottom layer is the hardware, while the highest layer is the user interface. The main advantage with layered structure is simplicity of construction and debugging. And the main difficulty is defining the various layers. Also this OS tends to be less efficient than other implementations. An image demonstrating the layered approach is as follows:

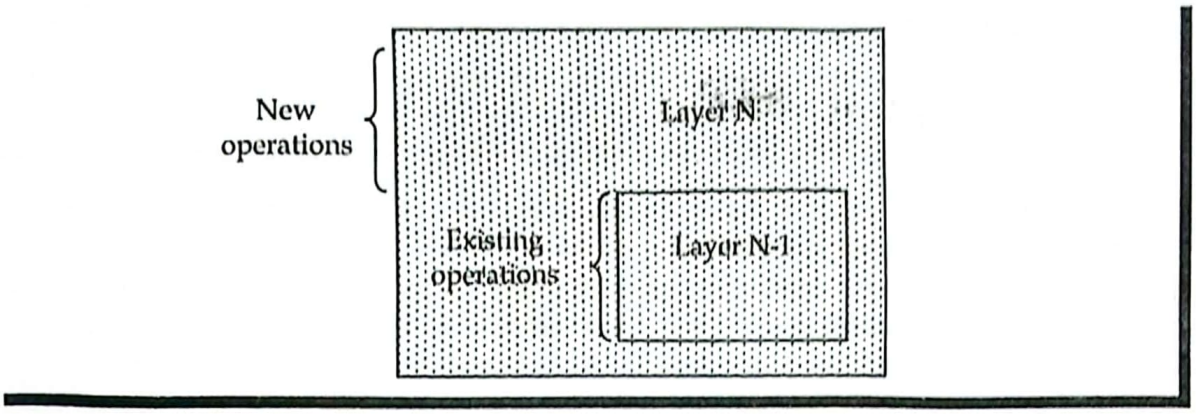


Fig. 2.2: Layered Structure of Operating system

As seen from the image, each upper layer is built on the bottom layer. All the layers hide some structures, operations etc. from their upper layers. One problem with the layered structure is that each layer needs to be carefully defined. This is necessary because the upper layers can only use the functionalities of the layers below them.

This is highly advantageous structure because all the functionalities are on different layers and hence each layer can be tested and debugged separately.

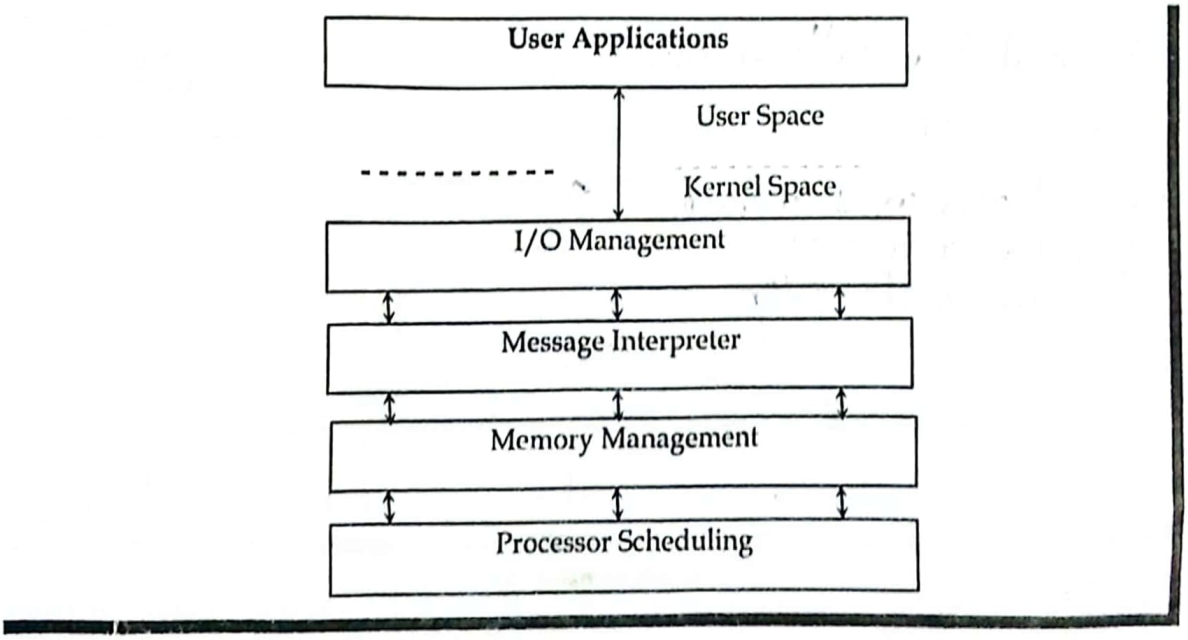


Fig: 2.3: Layered architecture

Advantages of Layered architecture

- Dysfunction of one layer will not affect the entire operating system
- Easier testing and debugging due to isolation among the layers
- Adding new functionalities or removing the obsolete ones is very easy

Disadvantages of Layered architecture

- It is not always possible to divide the functionalities, many a times they are in related and can't be separated.
- Sometimes, a large no. of functionalities is there and number of layers increase grea This might lead to degradation in performance of the system.
- No communication betwee non-adjacent layers.

KERNEL

A Kernel is a computer program that is the heart and core of an Operating System. Since th Operating System has control over the system so, the Kernel also has control over everything in th system. It is the most important part of an Operating System. Whenever a system starts, the Kernel i the first program that is loaded after the boot loader because the Kernel has to handle the rest of th thing of the system for the Operating System. The Kernel remains in the memory until the Operatin System is shut-down.

The Kernel is responsible for low-level tasks such as disk management, memory management, tas management, etc. It provides an interface between the user and the hardware components of th system. When a process makes a request to the Kernel, then it is called System Call.

A Kernel is provided with a protected Kernel Space which is a separate area of memory and this are is not accessible by other application programs. So, the code of the Kernel is loaded into thi protected Kernel Space. Apart from this, the memory used by other applications is called the Use Space. As these are two different spaces in the memory, so communication between them is a bi slower.

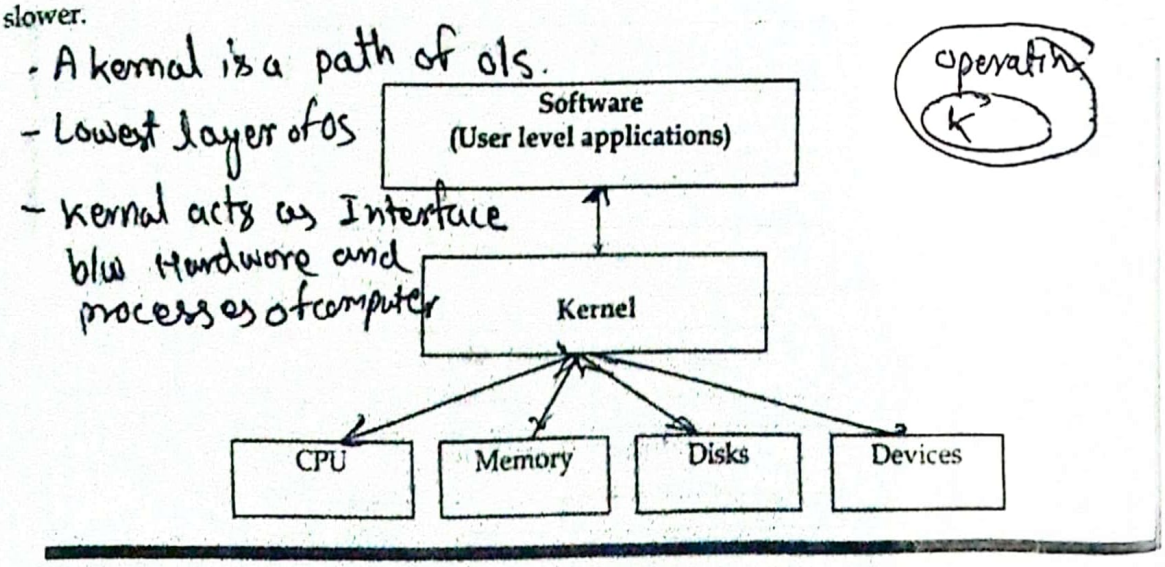


Fig 2.4: Kernel in OS

Functions of a Kernel

Following are the functions of a Kernel:

- Access Computer resource: A Kernel can access various computer resources like th CPU, I/O devices and other resources. It acts as a bridge between the user and th resources of the system.

- **Resource Management:** It is the duty of a Kernel to share the resources between various processes in such a way that there is uniform access to the resources by every process.
- **Memory Management:** Every process needs some memory space. So, memory must be allocated and de-allocated for its execution. All these memory management is done by a Kernel.
- **Device Management:** The peripheral devices connected in the system are used by the processes. So, the allocation of these devices is managed by the Kernel.
- **Interrupt Handling:** While executing the processes, there are conditions where tasks with more priority need to be handled first. In these cases, the kernel has to interrupt in-between the execution of the current process and handle tasks with more priority which has arrived in between.

Kernel Mode and User Mode

There are certain instructions that need to be executed by Kernel only. So, the CPU executes these instructions in the Kernel Mode only. For example, memory management should be done in Kernel-Mode only. While in the User Mode, the CPU executes the processes that are given by the user in the User Space.

Types of Kernel

In general, there are five types of Kernel. They are:

a. **Monolithic Kernels**

Monolithic Kernels are those Kernels where the user services and the kernel services are implemented in the same memory space i.e. different memory for user services and kernel services are not used in this case. By doing so, the size of the Kernel is increased and this, in turn, increases the size of the Operating System. As there is no separate User Space and Kernel Space, so the execution of the process will be faster in Monolithic Kernels.

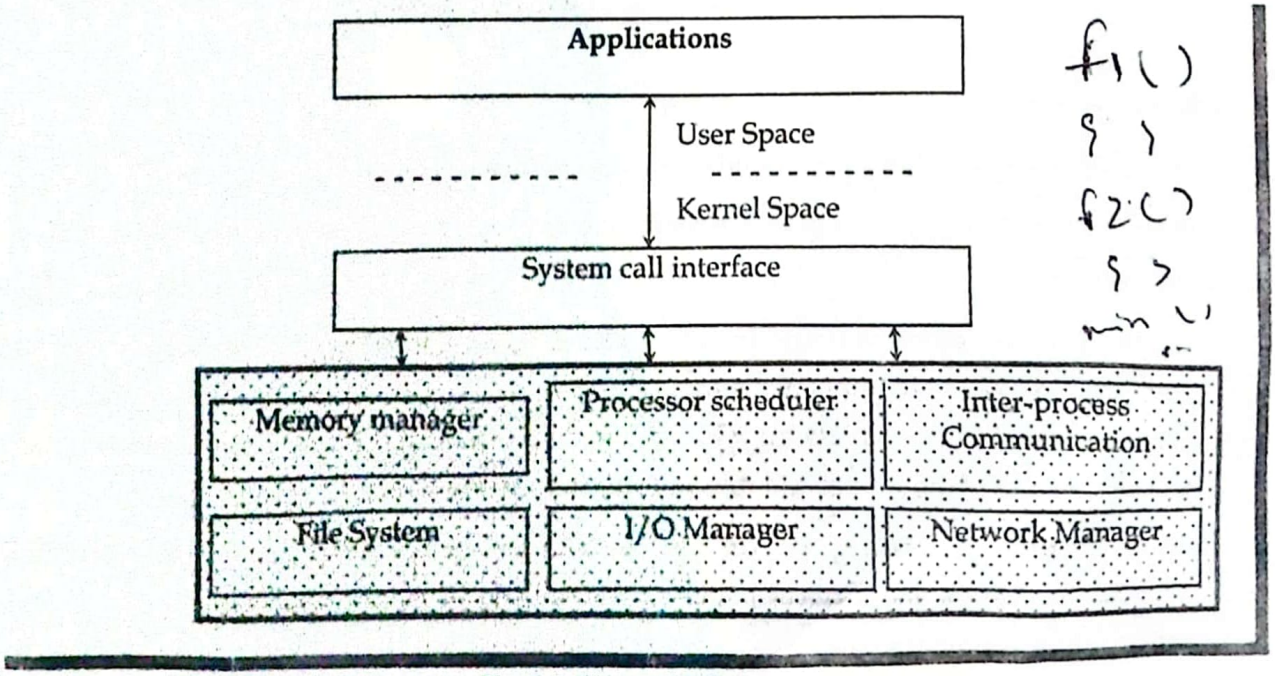


Fig: 2.5: Monolithic kernel

In this model, for each system call there is one service procedure that takes care of it. The utility procedures do things that are needed by several service procedures, such as fetching data from user programs. This division of the procedures into three layers is shown in Fig. below:

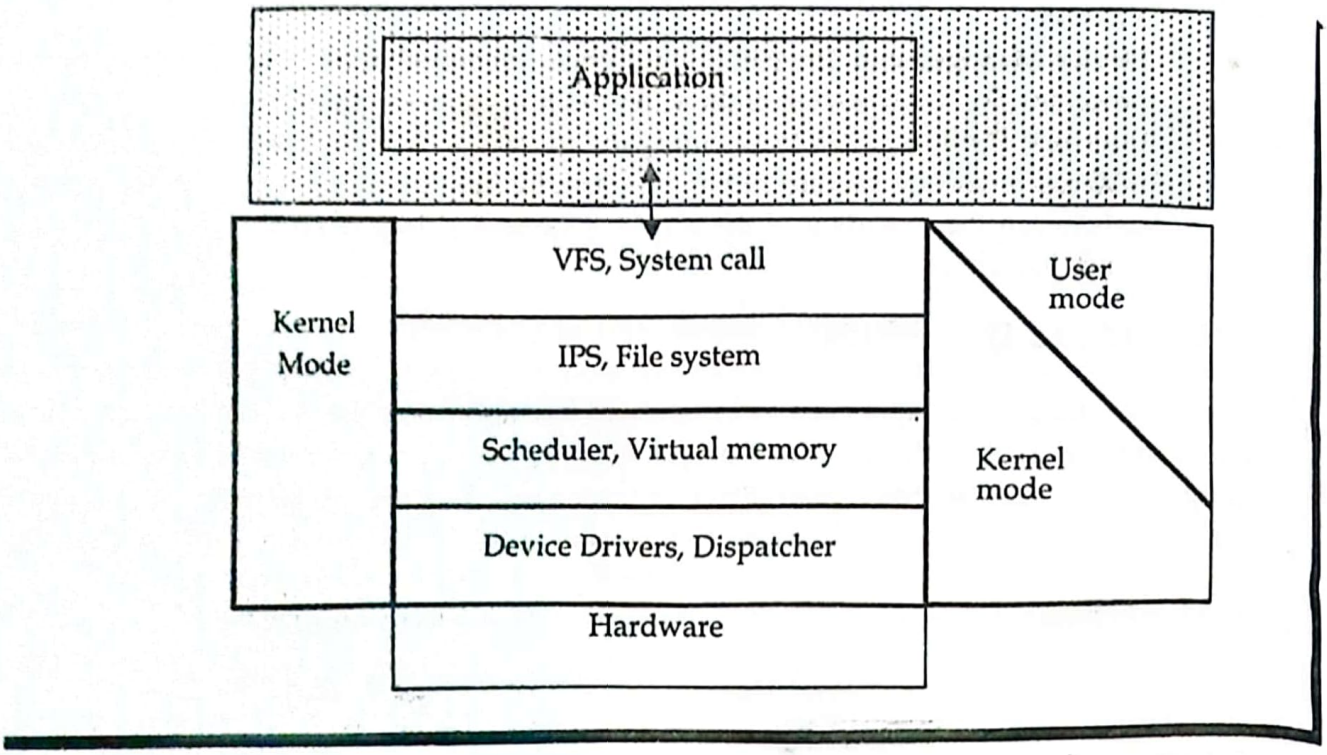


Fig 2.6: Monolithic system structure

Advantages

- It provides CPU scheduling, memory scheduling, file management through System calls only.
- Execution of the process is fast because there is no separate memory space for user and kernel.

Disadvantages

- If any service fails, then it leads to system failure.
- If new services are to be added then the entire Operating System needs to be modified.

b. Microkernel

A Microkernel is different from Monolithic kernel because in a Microkernel, the user services and kernel services are implemented into different spaces i.e. we use User Space and Kernel Space in case of Micro-kernels. As we are using User Space and Kernel Space separately, so it reduces the size of the Kernel and this, in turn, reduces the size of Operating System.

As we are using different spaces for user services and kernel service, so the communication between application and services is done with the help of message parsing and this, in turn, reduces the speed of execution.

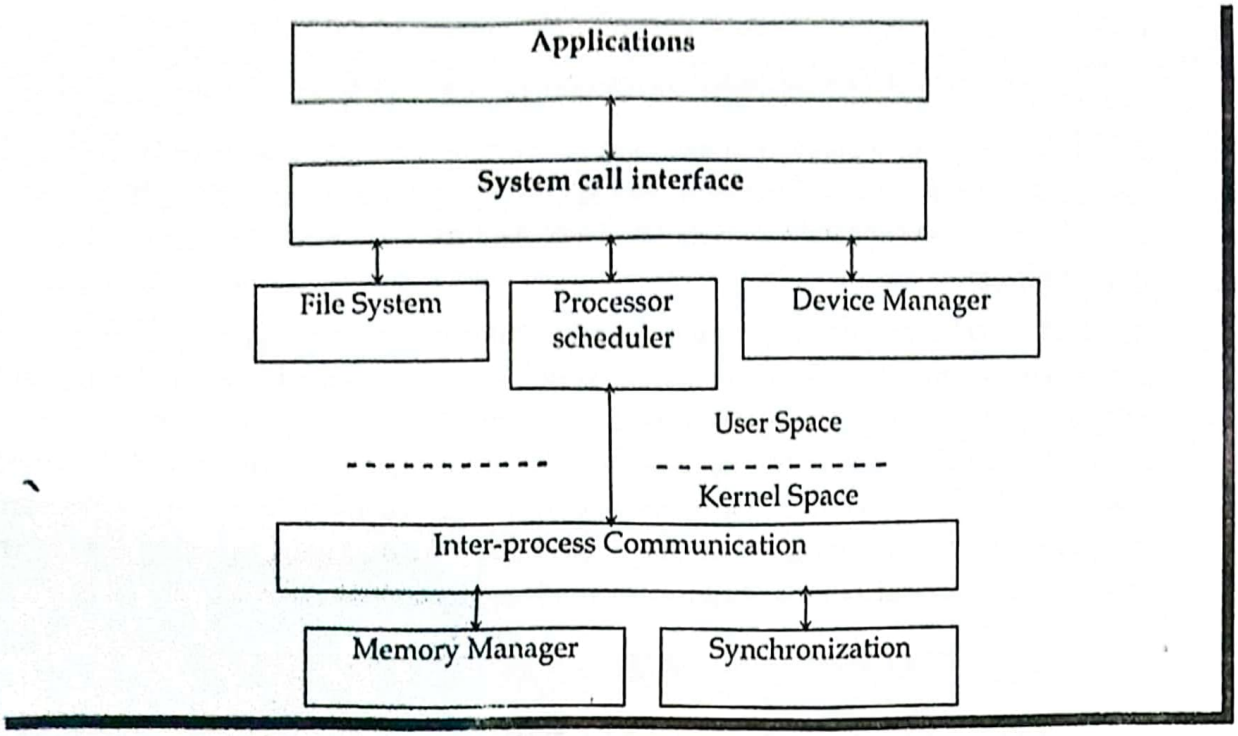


Fig 2.7: Microkernel structure

This structures the operating system by removing all nonessential portions of the kernel and implementing them as system and user level programs. Generally, they provide minimal process and memory management, and a communications facility and communication between components of the OS is provided by message passing. With microkernel, Extending the operating system becomes much easier as well as any changes to the kernel tend to be fewer, since the kernel is smaller and the microkernel also provides more security and reliability. The main disadvantage of microkernel is the poor performance due to increased system overhead from message passing.

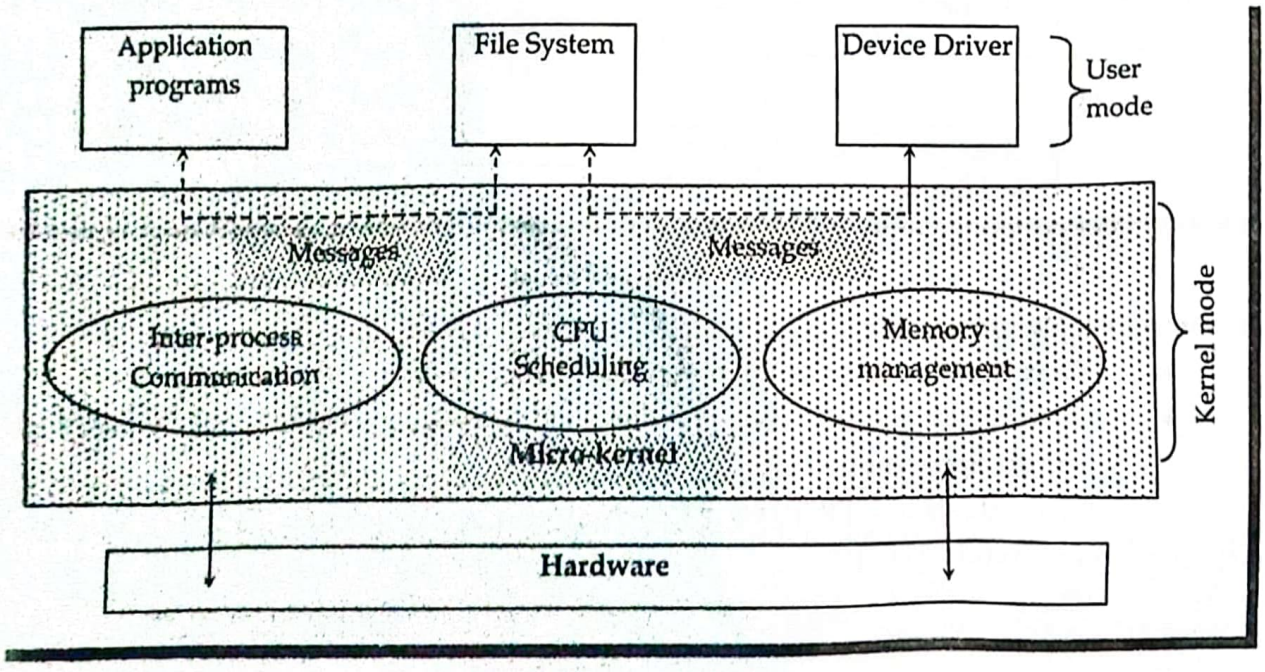


Fig 2.8: Microkernel structure

Advantages

- If new services are to be added then it can be easily added.

Disadvantages

- Since we are using User Space and Kernel Space separately, so the communication between these can reduce the overall execution time.

c. **Hybrid Kernel**

A hybrid kernel is an operating system kernel architecture that attempts to combine aspects and benefits of microkernel and monolithic kernel architectures used in computer operating systems. It makes the use of the speed of Monolithic Kernel and the modularity of Microkernel.Hybrid kernels are micro kernels that have some non-essential code in kernel-space in order for the code to run more quickly than it would be in user-space. So, some services such as network stack or file system are run in Kernel space to reduce the performance overhead, but still, it runs kernel code as servers in the user-space.

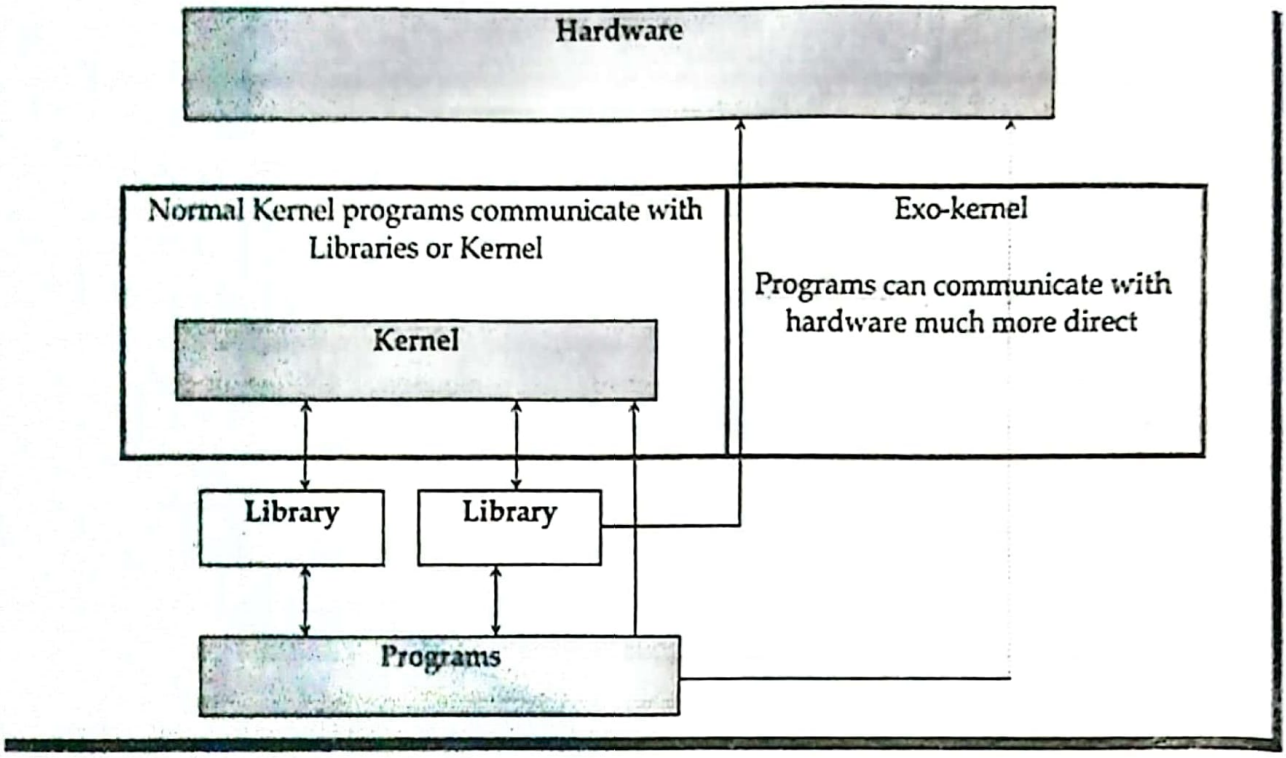


Fig 2.9: Hybrid kernel

Advantages of Hybrid Kernels

- Faster development time for drivers that can operate from within modules.
- On demand capability versus spending time recompiling a whole kernel for things like new drivers or subsystems.
- Faster integration of third party technology

Disadvantages of Hybrid Kernels

- With more interfaces to pass through, the possibility of increased bugs exists
- Maintaining modules can be confusing for some administrators when dealing with problems like symbol differences.

Call

SYSTEM CALLS

The interface between a process and an operating system is provided by system calls. In general, system calls are available as assembly language instructions. They are also included in the manuals used by the assembly level programmers. 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. A figure representing the execution of the system call is given as follows:

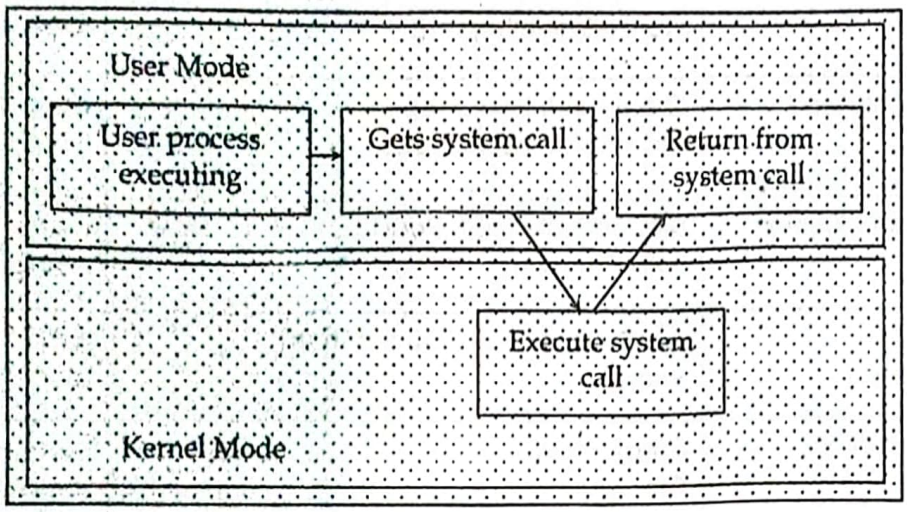


Fig 2.10: System calls

As can be seen from this diagram, the processes execute normally in the user mode until a system call interrupts this. Then the system call is executed on a priority basis in the kernel mode. After the execution of the system call, the control returns to the user mode and execution of user processes can be resumed. In general, system calls are required in the following situations:

- If a file system requires the creation or deletion of files. Reading and writing from files also require a system call.
- Creation and management of new processes.
- Network connections also require system calls. This includes sending and receiving packets.
- Access to a hardware device such as a printer, scanner etc. requires a system call.

Types of System Calls

There are mainly five types of system calls. These are explained in detail as follows:

- **Process Control:** These system calls deal with processes such as process creation, process termination etc.,
- **File Management:** These system calls are responsible for file manipulation such as creating a file, reading a file, writing into a file etc.

- **Device Management:** These system calls are responsible for device manipulation such as reading from device buffers, writing into device buffers etc.
- **Information Maintenance:** These system calls handle information and its transfer between the operating system and the user program.
- **Communication:** These system calls are useful for inter process communication. They also deal with creating and deleting a communication connection.

Examples of Windows and UNIX System Calls

	Windows	Unix
Process Control	CreateProcess(),ExitProcess() WaitForSingleObject()	fork() exit(),wait()
File Manipulation	CreateFile(),ReadFile() WriteFile(),CloseHandle()	open(),read() write(),close()
Device Manipulation	SetConsoleMode(),ReadConsole() WriteConsole()	ioctl() read(),write()
Information Maintenance	GetCurrentProcessID(),SetTimer() Sleep()	getpid() alarm(),sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

SYSTEM PROGRAMS

System programs provide an environment where programs can be developed and executed. In the simplest sense, system programs also provide a bridge between the user interface and system calls. In reality, they are much more complex. For example, a compiler is a complex system program. The system programs are used to program the operating system software. While application programs provide software that is used directly by the user, system programs provide software that are used by other systems such as SaaS applications, computational science applications etc. Most system programs are created to have a low runtime overhead. These programs may have small runtime library. Some parts of the system programs may be directly written in assembly language by the programmers. A debugger cannot be used on system programs mostly. This problem can be solved by running the programs in a simulated environment. Some examples of system programs are operating system, networking system, web site server, data backup server etc.

System Programs Purpose

The system program serves as a part of the operating system. It traditionally lies between the user interface and the system calls. The user view of the system is actually defined by system programs and not system calls because that is what they interact with and system programs are closer to the user interface.

An image that describes system programs in the operating system hierarchy is as follows:

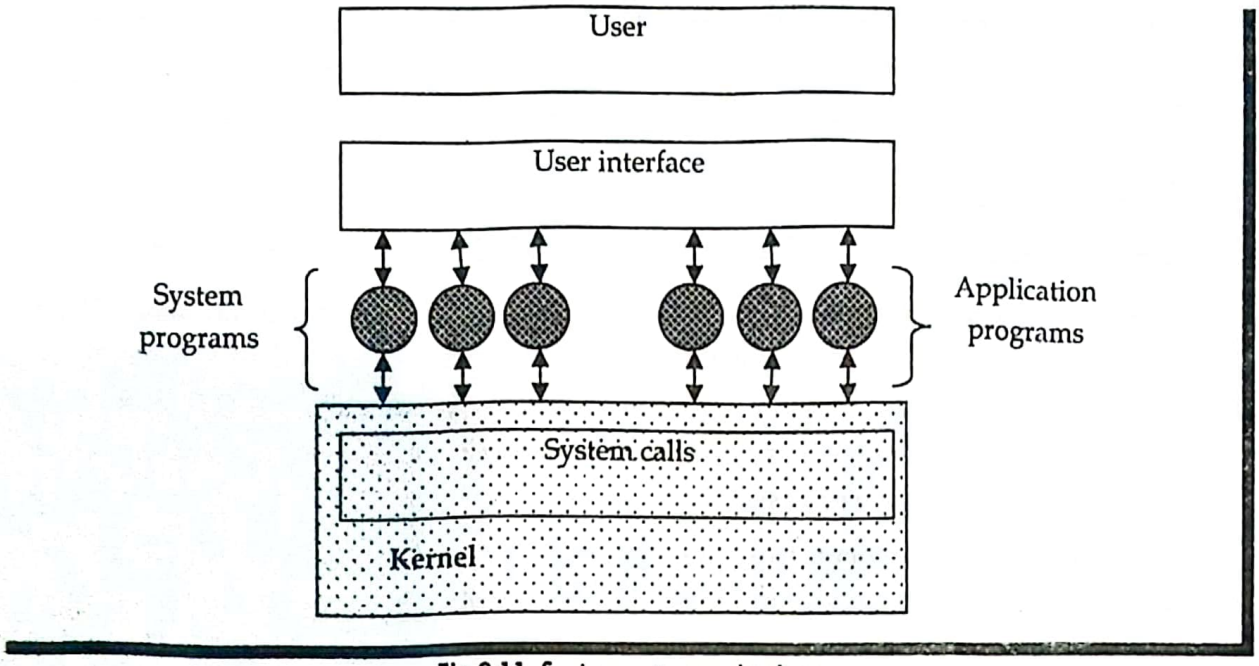


Fig 2.11: System program structure

In the above image, system programs as well as application programs form a bridge between the user interface and the system calls. So, from the user view the operating system observed is actually the system programs and not the system calls.

Types of System Programs

System programs can be divided into seven parts. These are given as follows:

- **Status Information:** The status information system programs provide required data on the current or past status of the system. This may include the system date, system time, and available memory in system, disk space, logged in users etc.
- **Communications:** These system programs are needed for system communications such as web browsers. Web browsers allow systems to communicate and access information from the network as required.
- **File Manipulation:** These system programs are used to manipulate system files. This can be done using various commands like create, delete, copy, rename, print etc. These commands can create files, delete files, copy the contents of one file into another, rename files, print them etc.
- **Program Loading and Execution:** The system programs that deal with program loading and execution make sure that programs can be loaded into memory and executed correctly. Loaders and Linkers are a prime example of this type of system programs.

- **File Modification:** System programs that are used for file modification basically change the data in the file or modify it in some other way. Text editors are a big example of file modification system programs.
- **Application Programs:** Application programs can perform a wide range of services as per the needs of the users. These include programs for database systems, word processors, plotting tools, spreadsheets, games, scientific applications etc.
- **Programming Language Support:** These system programs provide additional support features for different programming languages. Some examples of these are compilers, debuggers etc. These compile a program and make sure it is error free respectively.

CLIENT-SERVER MODEL

Client-server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. A server host runs one or more server programs, which share their resources with clients. They are usually multiple clients in communication with a single server. The clients send requests to the server and the server responds to the client requests.

The client/server model introduces two roles that can be assumed by processes: the role of service user (client) and the role of service provider (server). The distribution of roles implies an asymmetry in the distributed execution of an application. The server offers a service to which one or more clients have access as shown in Figure below. Here processes act as natural units in the distribution. In the context of distributed systems, the communication between client and server can be based on one of the mechanisms mentioned in the previous section. The client/server model only introduces roles that can be assumed by a process. At a given point in time, a process can assume the role of both client and server. This scenario occurs, for example, when the server is carrying out a task and delegates a subtask to a subordinate server.

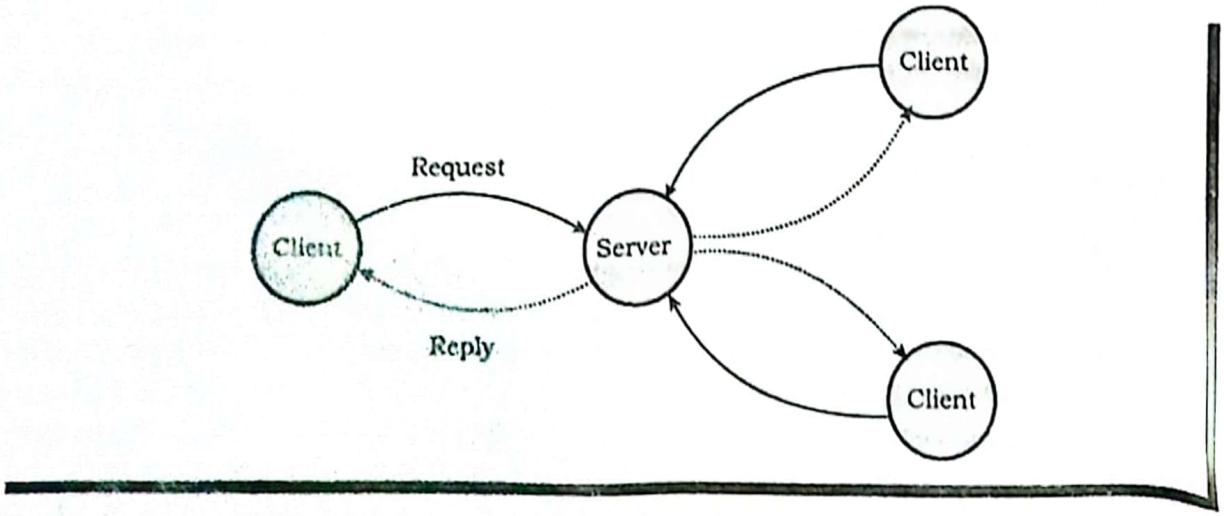


Fig 2.12: Client/server model

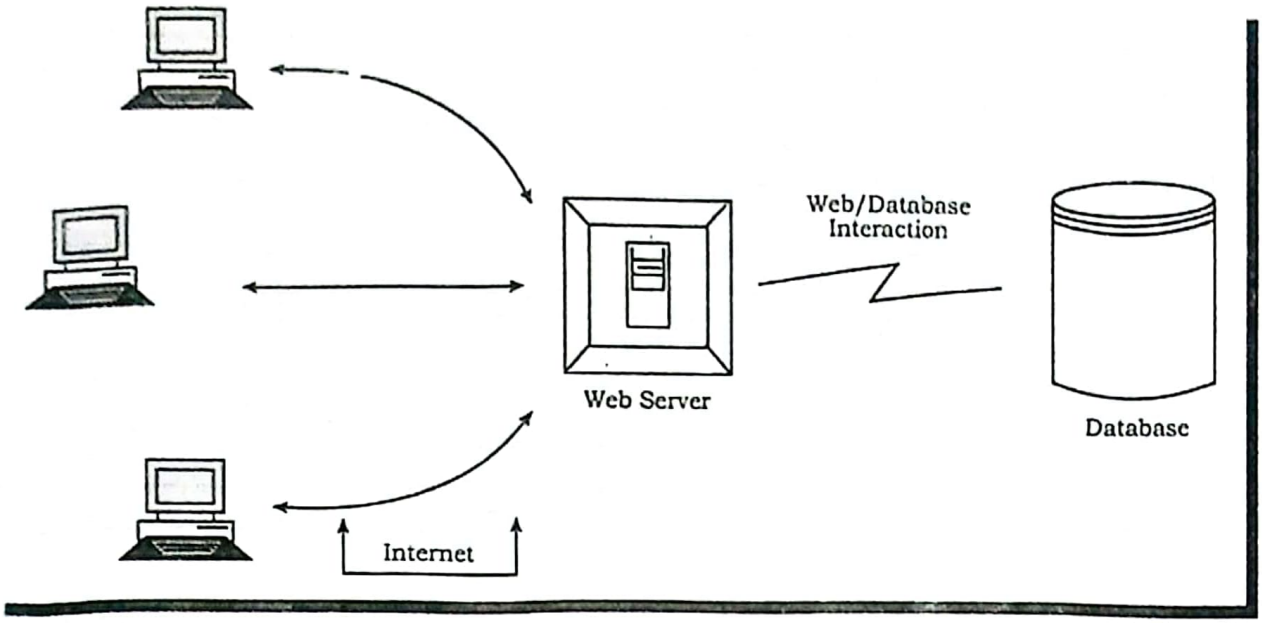


Fig 2.13: Client/server model

Advantages of Client-Server model

- Centralized system with all data in a single place.
- Cost efficient requires less maintenance cost and Data recovery is possible.
- The capacity of the Client and Servers can be changed separately.

Disadvantages of Client-Server model

- Clients are prone to viruses, Trojans and worms if present in the Server or uploaded into the Server.
- Server is prone to Denial of Service (DOS) attacks.
- Data packets may be spoofed or modified during transmission.
- Phishing or capturing login credentials or other useful information of the user is common and MITM (Man in the Middle) attacks are common.

VIRTUAL MACHINES

Virtual Machine abstracts the hardware of our personal computer such as CPU, disk drives, memory, NIC (Network Interface Card) etc, into many different execution environments as per our requirements, hence giving us a feel that each execution environment is a single computer. For example, VirtualBox.

Virtualization technology enables a single PC or server to simultaneously run multiple operating systems or multiple sessions of a single OS. A machine with virtualization software can host numerous applications, including those that run on different operating systems, on a single platform. The host operating system can support a number of virtual machines, each of which has the characteristics of a particular OS. The solution that enables virtualization is a virtual machine monitor (VMM), or hypervisor.

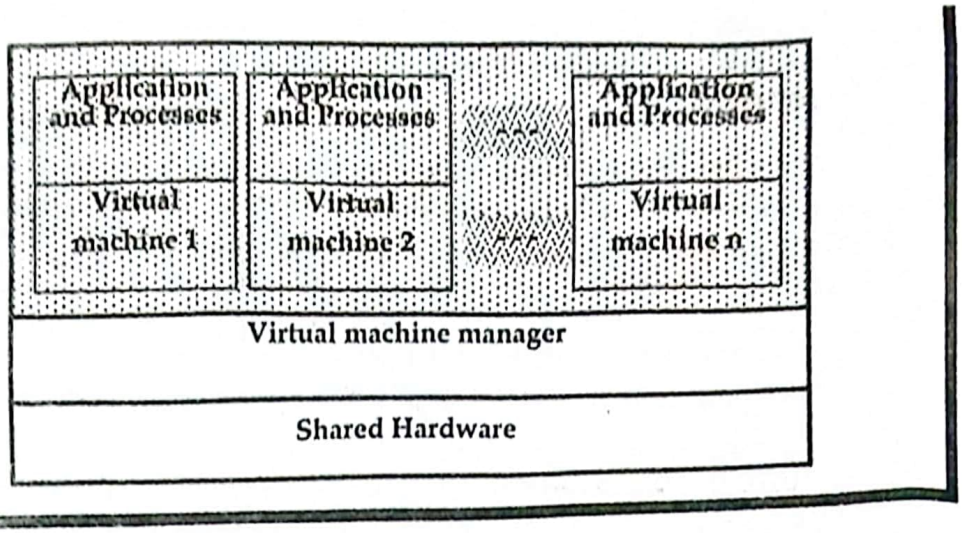


Fig 2.14: Virtual Machine Concept

An organization can have multiple VMs running different operating systems (OS) all stored on one host machine that's monitored by a hypervisor. There are two types of hypervisors, Type 1 and Type 2.There are different types of virtual machines and each offer different functions:

- A system virtual machine is an environment that allows multiple instances of the operating system (VMs) to run on a host system, sharing the physical resources.
- On the other hand, a process virtual machine, also known as an application VM, is used to execute computer programs in a platform-independent environment; it's designed to run applications in the same way, irrespective of the platform.

Type 1 hypervisors run directly on the physical hardware (usually a server), taking the place of the OS. Typically, we use a separate software product to create and manipulate VMs on the hypervisor. Some management tools, like VMware's vSphere, let you select a guest OS to install in the VM. We can use one VM as a template for others, duplicating it to create new ones. Depending on our needs, we might create multiple VM templates for different purposes, such as software testing, production databases, and development environments.

Type 2 hypervisors run as an application within a host OS and usually target single-user desktop or notebook platforms. With a Type 2 hypervisor, we manually create a VM and then install a guest OS in it. We can use the hypervisor to allocate physical resources to our VM, manually setting the amount of processor cores and memory it can use. Depending on the hypervisor's capabilities, we can also set options like 3D acceleration for graphics.

Advantages and benefits of VMs

VMs offer several benefits over traditional physical hardware:

- **Resource utilization and improved ROI:** Because multiple VMs run on a single physical computer, customers don't have to buy a new server every time they want to run another OS, and they can get more return from each piece of hardware they already own.
- **Scale:** With cloud computing, it's easy to deploy multiple copies of the same virtual machine to better serve increases in load.

- **Portability:** VMs can be relocated as needed among the physical computers in a network. This makes it possible to allocate workloads to servers that have spare computing power. VMs can even move between on-premises and cloud environments, making them useful for hybrid cloud scenarios in which you share computing resources between your data center and a cloud service provider.
- **Flexibility:** Creating a VM is faster and easier than installing an OS on a physical server because you can clone a VM with the OS already installed. Developers and software testers can create new environments on demand to handle new tasks as they arise.
- **Security:** VMs improve security in several ways when compared to operating systems running directly on hardware. A VM is a file that can be scanned for malicious software by an external program. We can create an entire snapshot of the VM at any point in time and then restore it to that state if it becomes infected with malware, effectively taking the VM back in time. The fast, easy creation of VMs also makes it possible to completely delete a compromised VM and then recreates it quickly, hastening recovery from malware infections.

THE SHELL

A shell is software that provides an interface between users and operating system of a computer to access the services of a kernel. The operating system carries out the system calls. Editors, compilers, assemblers, linkers, and command interpreters definitely are not part of the operating system, even though they are important and useful, called the shell. Shell is not part of the operating system, but makes heavy use of many operating system features and thus serves as a good example of how the system calls can be used. It is also the primary interface between a user and the operating system, unless the user is using a graphical user interface. There are two types of shell:

- **Command-line shell (CLI)** (e.g. Bash (sh), Command Prompt (cmd), C shell, Bourne shell, Korn shell (ksh) etc.)

  A command-line interface (CLI) is an operating system shell that uses alphanumeric characters typed on a keyboard to provide instructions and data to the operating system, interactively. For example, a teletypewriter can send codes representing keystrokes to a command interpreter program running on the computer; the command interpreter parses the sequence of keystrokes and responds with an error message if it cannot recognize the sequence of characters, or it may carry out some other program action such as loading an application program, listing files, logging in a user and many others. Operating systems such as UNIX have a large variety of shell programs with different commands, syntax and capabilities. Some operating systems had only a single style of command interface; commodity operating systems such as MS-DOS came with a standard command interface but third-party interfaces were also often available, providing additional features or functions such as remote program execution.

- **GUI Shell** (e.g. Windows Explorer or Windows Shell)

  Graphical shells (or desktop shells) provide means for manipulating programs based on graphical user interface (GUI), by allowing for operations such as opening, closing, moving and resizing windows, as well as switching focus between windows. Graphical shells may be included with desktop environments or come separately, even as a set of loosely coupled utilities.

A third type of shell is recently developed – a GCLI (Graphical Command Line Interface) shell. A GCLI shell combines the features of both CLI and GUI shell and provides an interface which is both user-friendly and powerful.

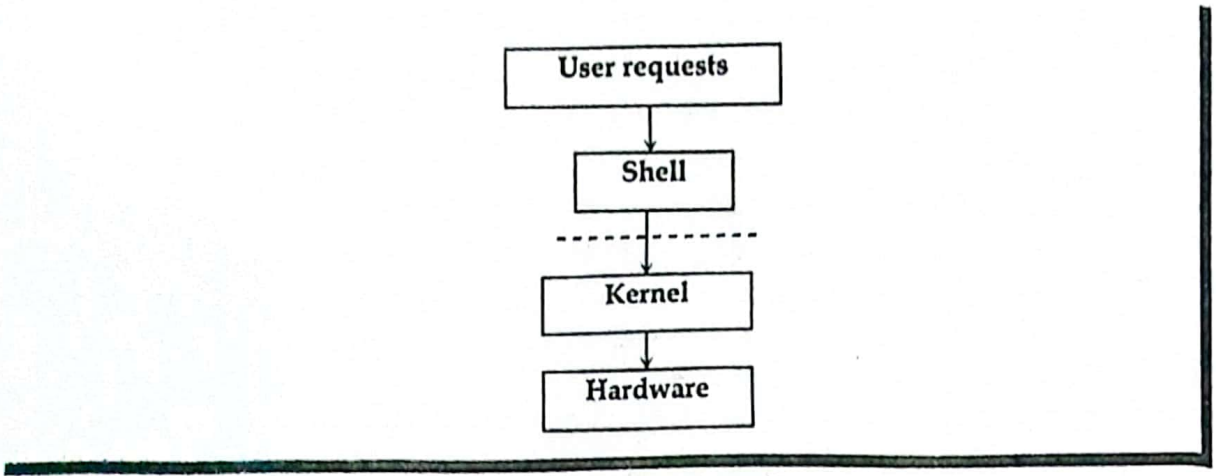


Fig 2.15: Shell space in computer system

Operating systems provide various services to their users, including file management, process management (running and terminating applications), batch processing, and operating system monitoring and configuration.

Most operating system shells are not direct interfaces to the underlying kernel, even if a shell communicates with the user via peripheral devices attached to the computer directly. Shells are actually special applications that use the kernel API in just the same way as it is used by other application programs. A shell manages the user-system interaction by prompting users for input, interpreting their input, and then handling an output from the underlying operating system. Since the operating system shell is actually an application, it may easily be replaced with another similar application, for most operating systems.

Trap: A trap is a software generated interrupt caused either by an error (for example, division by zero or invalid memory access) or by a specific request from a user that an operating system service be provided.

Difference between Kernel and Shell

The main difference between kernel and shell is that the kernel is the core of the operating system that controls all the tasks of the system while the shell is the interface that allows the users to communicate with the kernel.

- ***Definition***

  The kernel is a computer program which acts as the core of the computer's operating system and has the control over everything in the system. A shell is a computer program which works as the interface to access the services provided by the operating system.

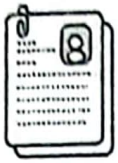
- **Usage**
  Kernel is the core of the system that controls all the tasks of the system. Shell is the interface between the kernel and user.
- **Types**
  Kernel does not have different types. Shell has different types such as Bourne shell, C shell, Korn Shell, Bourne Again Shell, etc.

EXERCISE

Multiple Choice Questions

1. The initial program that is run when the computer is powered up is called _____
   a) Boot program b) boot loader
   c) Initialize d) bootstrap program
2. How does the software trigger an interrupt?
   a) Sending signals to CPU through bus
   b) Executing a special operation called system call
   c) Executing a special program called system program
   d) Executing a special program called interrupt trigger program
3. In the layered approach of Operating Systems _____
   a) Bottom Layer(0) is the User interface b) Highest Layer(N) is the User interface
   c) Bottom Layer(N) is the hardware d) Highest Layer(N) is the hardware
4. The kernel is _____ of user threads.
   a) A part of b) The creator of
   c) Unaware of d) Aware of
5. Because the kernel thread management is done by the Operating System itself ___
   a) kernel threads are faster to create than user threads
   b) kernel threads are slower to create than user threads
   c) kernel threads are easier to manage as well as create then user threads
   d) none of the mentioned
6. If a kernel thread performs a blocking system call, _____
   a. the kernel can schedule another thread in the application for execution
   b. the kernel cannot schedule another thread in the same application for execution
   c. the kernel must schedule another thread of a different application for execution
   d. the kernel must schedule another thread of the same application on a different processor

7. Interfaces of the operating system provides.....
   a) Optimization b) Designing c) Reusability d) Portability
8. Environment for execution of programs is provided by.....
   a) Inputs b) outputs c) Operating system d) memory
10. Environment in which programs of the computer system are executed is called.....
   a) Operating system b) nodes c) Clustered system d) both a and b
10. System resources of computer system can be utilized better in.....
   a) Single program environment b) dual program environment
   c) Core environment d) multi program environment



Subjective Questions

1. How does a microkernel differ from a conventional kernel? Briefly list the motivations ar difficulties behind this.
2. What is a software interrupt? Explain why these might not necessarily be found in monolithic operating system, but are always found in kernel-based systems?
3. Briefly explain the terms "process" and "context switch"
4. What is system program? How it is differ from application programs? Explain
5. What is system call? Describe their importance.
6. Describe structure of OS with suitable example.
7. What is micro kernel? How it is differ from kernel? Explain
8. What is the use of kernel in OS? Explain
9. How multiprogramming differ from multi-tasking? Explain
10. What is Distributed OS? How it is differ from real time OS? Explain.
11. What is multi-processing system? Explain with suitable example.
12. What is program? How it is differ from software? Explain.
13. Define shell. How it is differ from kernel? Explain.
14. Define kernel. Describe different types of kernel with suitable example.
15. What is monolithic kernel? How it is differ from Micro kernel? Explain.
16. Define virtual machine with suitable example.
17. Describe client server model with suitable example.
18. What is system call? Explain types of system call with proper example.
19. What are the advantages and disadvantages of virtual machine? Explain.
20. What is hybrid kernel? Explain their features and advantages.

ANSWERS KEY

1. (d)	2. (b)	3. (b)	4. (c)	5. (b)	6. (a)	7. (d)	8. (c)	9. (a)	10. (d)

□□□