Unit 3: Operating System Structures

3

Unit Structure

3.1 Learning Objectives

3.2 Introduction

3.3 General Terms of Operating System

3.4 General Booting Sequence

3.5 System Call

3.6 Architecture of Operating System

3.7 Let us sum up

3.8 Check your Progress

3.9 Check your Progress: Possible Answers

3.10 Further Reading

3.11 Assignments

3.1 LEARNING OBJECTIVE

After studying this chapter, students should be able to understand:

- General terms of operating system
- Booting sequence of operating system
- Concept of system call
- Monolithic architecture of operating system
- Microkernel architecture of operating system
- Exokernel architecture of operating system
- Hybrid architecture of operating system

3.2 INTRODUCTION

In above unit-1 we discussed about the basic functions of OS along with types of operating system. Now it's time to learn: How does operating system start? How does it initialize hardware? Which kinds of architectures are followed by generalize operating systems?

Booting is the first process of the operating system through which entire environment is setup. After completion of booting process, OS system starts its initial processes and handsover the control to user by providing user interface. Operating System needs to work with hardware on behalf of user and provide back results which generated by hardware. To work with hardware, operating system uses various kind of system calls. At end of this unit we will discuss about the various architectures which have been developed over time. Before delving into booting process of operating system, there are some basic concept that needs to be understood.

3.3 GENERAL TERMS OF OPERATING SYSTEM

- **BIOS**

  Basic Input-Output System (BIOS) is responsible for performing basic input – output operations. These operations are low level routines that are used by OS to interact with I/O devices such as keyboard, mouse, monitor etc. Later on we will see that BIOS is responsible to initialize the loading process of OS to main memory (RAM) from secondary memory (Hard disk).

- **Bootstrapping**

  The OS is stored on hard disk. We need to load OS from hard disk to the main memory (RAM). To do the same we need to execute a set of instructions that load the OS onto memory. The entire execution process of loading OS from secondary memory to main memory called Bootstrapping or Booting process.

- **Boot Loader / Bootstrap Loader**

  The set of instructions that load OS from hard disk to memory is known as Boot Loader / Bootstrap Loader. Sometime it is also called Boot Software.

- **Boot Device**

  The Operating System is permanently stored in secondary memory. The secondary memory can be hard disk, CD, Pen Drive or any external device. The device that stores the OS is called Boot Device.

- **Boot Sector**

  BIOS contains a program that loads the first sector of the boot device which is called Boot Sector. It is called boot sector because of its location. It is located in the first sector on hard disk (Sector 1, Cylinder 0, and Head 0). It is also called Master Book Record (MBR). Generally boot sector is the part of hard disk with size of 512 bytes. In the MBR the first 446 bytes are the primary boot loader, which is also referred as PBL. The next sixty-four bytes are the partition table, which has the record for each of the hard disk's partitions. The MBR ends with two bytes that should be 0xAA55. These numbers act as validation of this sector indicating that it is the boot sector or Master Boot Record.

- **Privileged Instructions**

  OS's user is not allowed to access attached devices directly. If a user wants to access device and get some result, He / she need to pass the instruction to the operating system. On the behalf of the user, operating system interacts with the device and gives the result to the user. Thus this kind of instructions which are not directly executed by the user but needs to pass by the OS, are called Privileged Instructions.

- **System Call**

  All the Privileged Instructions that need to interact with hardware and other resource are known as system calls. For example when a user wants to display some text on monitor, he/she need to write some output instructions in proper format. This instruction program is called system call.

3.4 GENERAL BOOTING SEQUENCE

Generally booting process is dependent on operating system. It varies with types of operating systems. Here we are going to discuss the general booting sequence. Following steps shows how the operating system can be loaded and start operating.

Step – 1:

Turn on the CPU button. Whenever the computer is switched on, all CPU pins and registers are reset to specific values and the control is transferred to the BIOS in the ROM or flash-RAM.

Step – 2:

The first job of the BIOS is to initialize and identify system devices such as the keyboard, mouse, monitor, video card, hard disk etc. This initialization process is called Power on self-test (POST). Power on Self-Test is the foremost routine which checks and tests the basic hardware. If it fails, it will display error.

Step – 3:

After the execution of POST, the BIOS determines the Boot Sector (or Master Boot Record) and reads it. This boot sector contains a program called MBR. It copies MBR to RAM. The MBR first examines the partition table and determines which partition is active.

Step – 4:

In the particular partition, there is a boot loader / bootstrap loader, which loads in to memory. The boot loader now loads operating system from secondary memory to RAM (bootstrapping).

Step – 5:

Once the operating system has loaded into RAM, the boot process relinquishes control to the operating system. After that operating system first queries the BIOS for the configuration informations. The operating system checks drivers for the attached devices and load into the kernel space.

Step – 6:

At the end system will display user login screen. The user programs are loaded into the memory as the user log in to system.

It might be noted that this sequence may slightly change in different OS.

3.5 SYSTEM CALL

We know that Privileged Instructions are called system calls. The role of system calls is important for understanding the operations of operating system. Before going into details of system calls, we need to understand the working mode of operating system.

Modern operating system have two modes – Kernel Mode and User Mode. All the user processes are executed in user mode and all privileged operations are executed in kernel mode. Many times user processes need to access system devices that need to execute privileged operations. The privileged operations are not allowed to execute in user mode, they are only allowed to be execute in kernel mode. Since operating system prevent direct access of kernel mode, problem is how user mode processes execute privileged operations.

User mode process uses the kernel mode function by a special interface. This interface permits interaction between user mode and kernel mode. This interface is called system call. In other word system call is an instruction which requests operating system to perform the specific operation that need hardware access. Generally system calls are made by the user program in the following situations:

- To create, delete or open a file in the file system.
- To create and manage new processes.
- To create network connection.
- To send and receive data packets from network
- To access hardware
- To manage main memory
- To protect the data.

3.5.1 TYPES OF SYSTEM CALLS

System calls are used to access the system resources. The type of system calls depends on the use of these resources. Basically there are five broad categories of system calls, which are as follow:

- **Device Management System Calls:**

  The user needs to access devices such as keyboard, mouse, and monitor. However, he/she cannot access them directly. These system calls are responsible for device manipulation such as reading data from keyboard, displaying data on monitoring etc. The general command related to this category are request of the devices, release of the device, read / write operation etc.

- **File Management System Calls:**

  The user store his/her data in various kind of files. User needs to create, update, open, close, read, write or delete a file in the computer system. The user thus needs to access this resource through file management system calls. These system calls are responsible for file manipulation such as creating a file, reading a file, writing into a file etc.

- **Communications System Calls:**

In computer system, processes need to communicate internally. All the communication operations are performed by the communications system calls. These system calls are useful for inter process communication. They also deal with creating and deleting a communication connection.

- **Process Control System Calls:**

The process is the basic entity in the operating system. The process needs to be created, deleted or aborted. All the process management are performed by the process control system call. These system calls deal with processes such as process creation, process termination and other activities.

- **Information Maintenance System Calls:**

Information Maintenance system calls are for accounting and providing information to the user. Such information can be about a process, memory, hard disk space, operating system, computer system etc. These system calls handle information and its transfer between the operating system and the user program.

3.6 ARCHITECTURE OF OPERATING SYSTEM

To get efficient performance from the operating system, it should be partitioned into subsystems. This sub systems are separated based on its tasks, inputs, outputs, and performances. These sub systems can then be arranged in to architectural configurations. An Operating system works on any one architecture from the four type mentioned below:

- Monolithic Architecture
- Microkernel Architecture
- Exokernel Architecture
- Hybrid Architecture

3.6.1 MONOLITHIC ARCHITECTURE

Now we know that in the modern operating system, operating system has user mode and kernel mode. Kernel mode is use to access the hardware. The kernel has unrestricted access to all resources and hardware of the system.

The earlier operating systems were developed in the same way as a programming. The OSs has one file with all functionalities and calling each other without any restrictions. Due to limited functionalities provided by OSs, all are placed in kernel mode. This kind of structure is called Monolithic architecture.

In the early monolithic architecture, every component of the operating system was contained within the kernel. It means that all the functions were executed in single space called kernel space. The advantage of this kind of structure is that functions can communicate efficiently to each other. Monolithic architecture has drawback also, as all functionalities placed in single layer (kernel mode), it is difficult to isolate errors or difficult to modifications in a functions or modules.

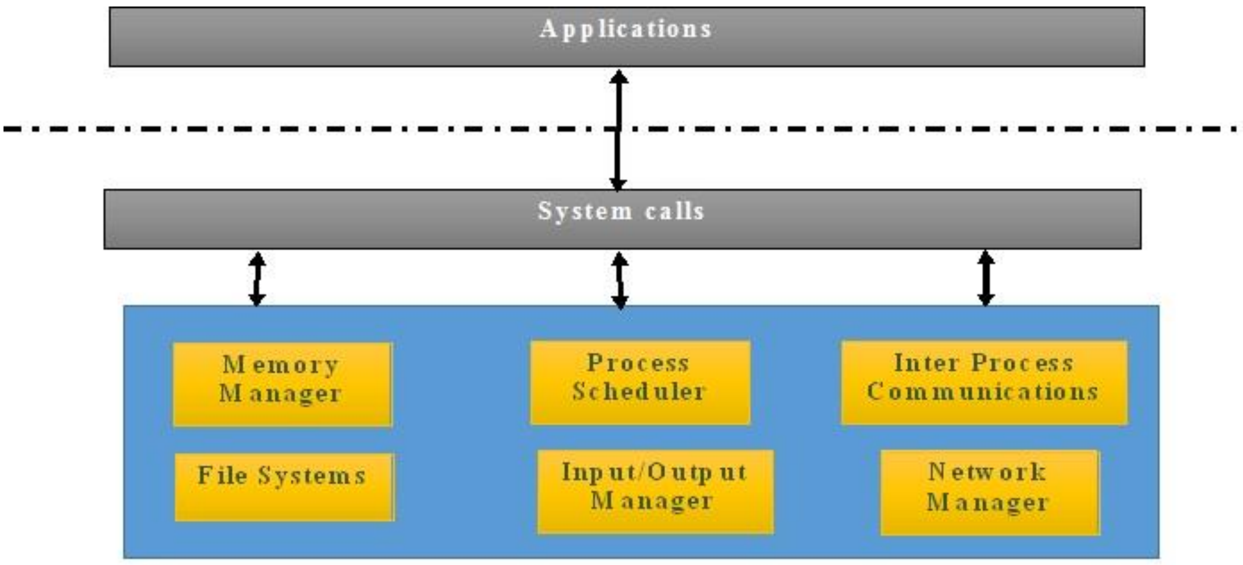

Figure-14 Monolithic Architecture

Figure-14 shows the monolithic architecture. Microsoft Window 95, Microsoft Window 98, BSDs, Solaris, DOS, initial version of UNIX and Linux are the operating system, which are built on monolithic architecture.

3.6.2 MICROKERNEL ARCHITECTURE

As operating system became larger and more complex, the size and work of kernel has also increased. Large sized kernels are difficult to maintain. OS developer also faced problem of extensibility, efficiency and reliability of kernel.

So developers tried to invent a new OS architecture that had less load in kernel mode. As a result, Microkernel architecture was developed. In micro kernel architecture very small number of essential functionalities are handled by the kernel. The kernel which manages only essential functionalities is called microkernel and architecture is called microkernel architecture. The essential functionalities may be process management, inter – process communication, memory management and so on. In this kind of architecture, other non-essential functionalities are moved up in the user mode. The non-essential functionalities may be file system management, network management, process scheduler, device manager and so on.

This kind of architecture divides the large sized kernel in to two parts: Kernel mode and User mode. Each mode has specific limited functionalities to handle. As a result, kernel (microkernel) is extensible, portable and scalable.

Microkernel architecture has been used in many operating system. Examples of operating systems that use a microkernel are - QNX, Integrity, PikeOS, Symbian, L4Linux, Singularity, K42, Mac OS X, HURD, Minix, and Coyotos. Figure-15 shows Microkernel Architecture.

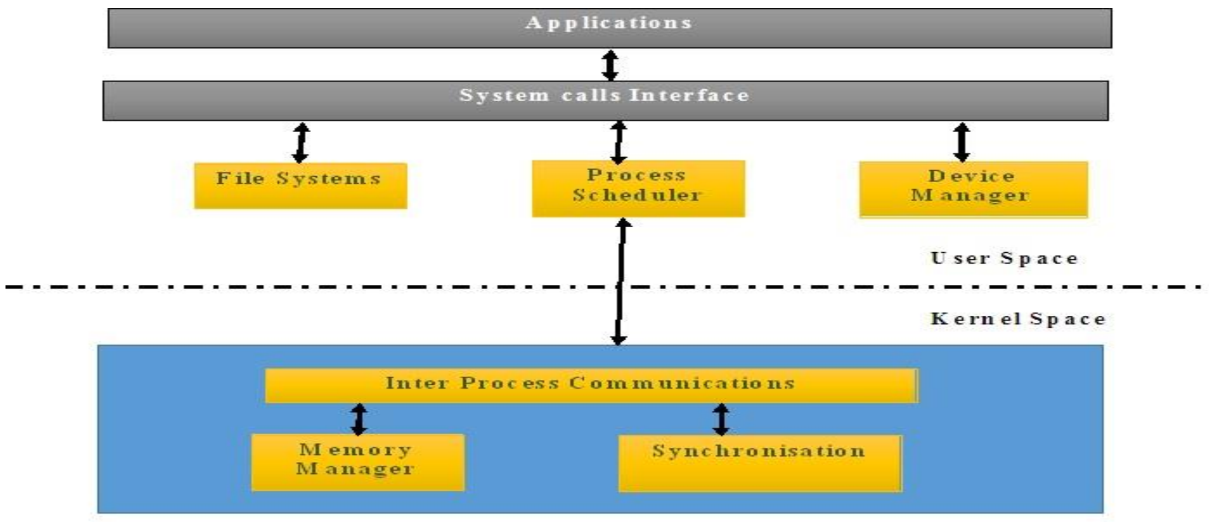

Figure-15 Microkernel Architecture

Table-4 shows the comparison between Monolithic Architecture and Microkernel Architecture.

Functionality	Monolithic Architecture	Microkernel Architecture
Work	All kind of functionalities are executed in single space called kernel space.	Essential functionalities are executed in kernel space and non-essential functionalities are executed in user space.
Size	Monolithic kernel is large in size.	Microkernel is small in size
Execution	Fast execution	Slow execution
Extensible	It is hard to extend	It is easy to extend
Security	If any service crashes, the entire system crashes.	If any service crashes, it does not effect on entire system.
Example	Linux, Free BDS, Microsoft Window 98, Microsoft Window 95, Solaris, Dos	Symbian, L4Linux, Mac OS, Minix

Table-4 Monolithic v/s Microkernel Architecture

3.6.3 EXOKERNEL ARCHITECTURE

Everyone knows that operating systems work as an interface between user's applications and hardware. User's applications need to send hardware instructions to OS, OS passes these instructions to the hardware on behalf of the user and sends back the result. This working methodology of kernel makes the lives of application programmers easier. Along with this they also realize that giving so much abstraction to the developer affects the performance of application. Applications that run on such architecture suffer with slower execution of functions resulting in performance issues.

To solve the performance issues, MIT provided new system with the concept of kernel with minimum functionality and provide access of resources to the application's developer as well. This architecture is designed to separate resource protection from management to facilitate application-specific customization. This kernel is called exokernel. In order words exokernel work as an executive for application programs. It ensures the safe use of resources and allocates them to the applications.

Here, new concept “library OS” is introduced in exokernel architecture. The “library OS” request the exokernel to allocate resources like disk space, memory address, CPU etc. and use the resources in the way it suits the application. For an example, an application can manage its own disk-block cache. It can also share the pages with the other applications, but the exokernel allows cached pages to be shared securely across all applications. Thus, the exokernel protects pages and disk blocks, but applications manage them. Figure-16 shows exokernel architecture.

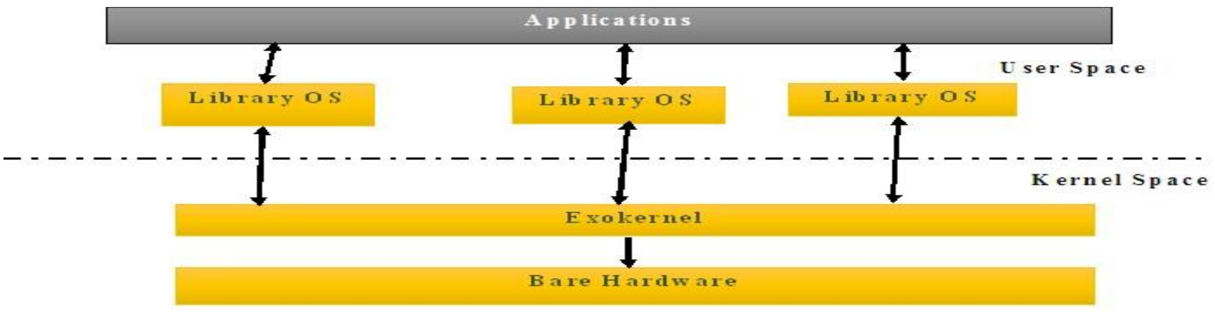

Figure-16 Exokernel Architecture

The performance of system will increase significantly by this kind of architecture. It has drawback also, its interface design is too complex and system consistency is very less. Exokernel was developed in 1994 by MIT. They developed two exokernels, namely Aegis and XOR, this concept has not been used in any commercial operating system and still research is going on.

3.6.4 HYBRID ARCHITECTURE

Many operating systems are not based on one architecutural model of the operating system. They may contain combination of architecture of multiple operating systems that have different approaches to performance, security, usability needs etc. This is known as a hybrid operating system.

A kernel having the mixed approach of various architecture is called hybrid kernel. The hybrid kernel attempts to combine the features and aspects of the microkernel and the monolithic kernel. This means that the kernel structure should be similar to a microkernel but the structure should be implemented like a monolithic kernel.

A well-known example of the hybrid kernel is the Microsoft Windows NT kernel. It is called a hybrid kernel instead of a monolithic kernel as the emulation subsystems run on the user mode rather than the kernel mode, unlike in monolithic kernel. The NT kernel cannot be called a microkernel as well. This is because almost all the system components run on the same address space as the kernel, which is a feature of the monolithic kernel.

3.7 LET US SUM UP

In this unit we learnt about the booting sequence of operating system, concept of system call and various architecture of operating system. Let's quickly review the main points of the unit.

- BIOS is responsible for the basic input and output operations.
- At the time of switch on, the set of instructions that load operating system from hard disk to main memory is call bootstrapping.
- System call is responsible to interact with hardware and other resources.
- Device management system call, file management system call, communications system call, process control system call, information maintenances system call are the type of system calls used by operating system.
- In the monolithic architecture all kinds of functionalities are executed in kernel space.
- In the microkernel architecture only essential functionalities are executed in kernel space and non-essential functionalities are executes in user space.
- To increase the performance in the microkernel architecture, new architecture is invented called exokernel architecture.
- A kernel having the mixed approach of various architecture is called hybrid kernel.

3.8 CHECK YOUR PROGRESS

Fill in the blanks.

1. BIOS stands for ________.
2. The set of instruction that load the operating system is called _______.
3. The set of instruction that are not directly executed by the user but need to pass by the user is called ______instruction.
4. POST stands for _____.
5. Privileged instructions is also called _____.
6. To manage the attached hardware ________ system calls is used.
7. To manage the files _________ system calls is used.
8. To manage the process _______ system calls is used.
9. To manage the user _______ system calls is used.
10. _______ architecture is large in size.
11. _______ architecture is slow in execution.
12. Linux is the example of ________ architecture.
13. Mac is the example of _________ architecture.
14. Exokernel was invented by _______.
15. A kernel having mixed approach of various architecture is called ________ kernel.

3.9 CHECK YOUR PROGRESS: POSSIBLE ANSWERS

1.	Basic Input –Output System	2.	Boot loader	3.	Privileged
4.	Power on self-test	5.	System call	6.	Device management
7.	File management	8.	Process control	9.	Information maintenance
10.	Monolithic	11.	Microkernel	12.	Monolithic
13	MicroKernel	14.	MIT	15.	Hybrid

3.10 FURTHER READING

- Naresh Chauhan (2014), Principals of Operating System, Oxford.

3.11 ASSIGNMENTS

Write answers of following Questions.

1. Define the terms : BIOS, Boot loader, Privileged Instructions
2. Writes the steps for booting sequence.
3. What is system call? Explain types of system calls in detail.
4. Explain Monolithic architecture with diagram
5. Explain Microkernel architecture with diagram.
6. State the difference between microkernel and monolithic architecture.