Course Instructor: Dr. Vijayarajan V, Associate Professor, SCOPE, VIT, Vellore

Introduction to Operating Systems

An Operating Systems (OS) course is a great way to learn about the fundamental concepts of operating systems. This course will typically cover topics such as:

- Processes and threads: Processes and threads are the basic units of execution in an operating system. This course will cover the different types of processes and threads, how they are created and managed, and how they interact with each other.
- Memory management: Memory management is the process of allocating and managing memory resources in a computer system. This course will cover the different memory management techniques, such as paging and segmentation, and how they are implemented in operating systems.
- I/O: I/O is the process of transferring data between a computer system and its peripheral devices. This course will cover the different I/O devices, how they are accessed by the operating system, and how I/O requests are scheduled.
- Scheduling: Scheduling is the process of determining which processes should be executed at a given time. This course will cover the different scheduling algorithms, such as round-robin and priority scheduling, and how they are implemented in operating systems.
- Security: Security is a critical issue in operating systems. This course will cover the different security threats, such as viruses and malware, and how they are mitigated by operating systems.

This Operating Systems course is a valuable course for anyone who wants to learn about the fundamentals of operating systems. This course will give you the foundation you need to understand how operating systems work and how they are used.

Here are some of the benefits of taking an Introduction to Operating Systems course:

- Learn about the fundamentals of operating systems: This course will give you a solid understanding of the basic concepts of operating systems, such as processes, threads, memory management, I/O, scheduling, and security.
- Gain hands-on experience with operating systems: This course will typically include hands-on exercises where you will get to work with operating systems. This will give you a practical understanding of how operating systems work.
- Improve your problem-solving skills: This course will help you develop your problem-solving skills. You will be challenged to solve problems related to operating systems. This will help you become a better problem solver.

- Prepare for a career in computer science: This course is a valuable addition to your computer science curriculum. It will give you the foundation you need to pursue a career in computer science.

If you are interested in learning about operating systems, I encourage you to take this Operating Systems course. This course will give you the foundation you need to understand how operating systems work and how they are used.

OLMCA504 - Operating Systems

L T P C

3 1 0 4

Pre-requisite NIL

Syllabus version 1.0

Course Objectives:

1. To understand the basic components of a computer operating system, and the interactions among the various components.

2. To understand the different services provided by operating system at different level.

3. To understand the essential properties of different types of operating systems.

Course Outcomes:

1. Understand the common features of an operating system and what an operating system does for the user.

2. Analyze concurrently executing processes and examine the issues of multicoresystems and parallel programming systems.

3. Develop various applications with deadlock handling capability.

4. Understand the memory management techniques to improve the utilization of the CPU.

5. Develop effective file system management techniques for the modern computer systems.

6. Develop protection and security techniques for the modern computer systems.

7. Understand the principles of distributed systems and develop network based operating systems.

Module:1 Operating-System Structures

Computer-System Organization-Computer-System-Architecture-Operating-System Structure-Virtualization -Kernel Data Structures - Computing Environments - Open-Source Operating Systems-Operating-System Services – System Calls-Types of System Calls - System Boot.

Module:2 Interprocess Communication-Basics

Process Concept - Process Scheduling -Operations on Processes - InterprocessCommunication examples -Multithreading Models- Thread Libraries - CPU Scheduling - Scheduling Algorithms.

Module:3 Process Synchronization & Deadlocks

The Critical-Section Problem - Peterson's Solution - Synchronization Hardware – Mutex Locks – Semaphores - Classic Problems of Synchronization- Monitors – Deadlock Characterization -Methods for Handling Deadlocks- Deadlock Prevention – Deadlock Avoidance - Deadlock Detection - Recovery from Deadlock.

Module:4 Memory Management & Virtual Memory

Background - Swapping - Contiguous Memory Allocation – Segmentation – Paging - Structure of the Page Table- Virtual Memory basics - Demand Paging - Copy-on-Write - Page Replacement - Allocation of Frames – Thrashing.

Module:5 Storage Management

Mass-Storage Structure overview - Disk Structure - Disk Scheduling- Disk Management - File Concept - Access Methods - Directory and Disk Structure- File-System Mounting- File System Implementation- Directory Implementation- Allocation Methods.

Module:6 Protection & Security

Goals of Protection - Principles of Protection - Domain of Protection - Access Matrix - Implementation of the Access Matrix- Access Control - The Security Problem- Program Threats - System and Network Threats- Cryptography as a Security Tool.

Module:7 Distributed Systems

Advantages of Distributed Systems-Types of Network based Operating Systems -Network Structure-Communication Structure -Communication Protocols.

Total Lecture hours: 120 hours

Item 64/11 - Annexure - 8

Text Book(s)

1. Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, Operating system concepts, 2013, Ninth Edition, Wiley publications.

Reference Books

1. W. Stallings Operating Systems: Internals and Design Principles, Seventh Edition, 2013, Pearson Education publications.

2 Andrew S. Tanenbaum, Herbert BOS, Modern Operating Systems, 2015, Fourth Edition, Pearson Education publications.

Mode of Evaluation: Quiz / Assignment / Mid-Term, FAT (Tentatively)

Recommended by Board of Studies 15.11.2021

Approved by Academic Council No. 64 Date 16-12-2021

View of operating system

- User view: The user view of the computer varies by the interface being used. The examples are -windows XP, vista, windows 7 etc. Most computer user sit in the in front of personal computer (pc) in this case the operating system is designed mostly for easy use with some attention paid to resource utilization. Some user sit at a terminal connected to a mainframe/minicomputer. In this case other users are accessing the same computer through the other terminals. There user are share resources and may exchange the information. The operating system in this case is designed to maximize resources utilization to assume that all available CPU time, memory and I/O are used efficiently and no individual user takes more than his/her fair and share. The other users sit at workstations connected to network of other workstations and servers. These users have dedicated resources but they share resources such as networking and servers like file, compute and print server. Here the operating system is designed to compromise between individual usability and resource utilization.

- System view: From the computer point of view the operating system is the program which is most intermediate with the hardware. An operating system has resources as

hardware and software which may be required to solve a problem like CPU time, memory space, file storage space and I/O devices and so on. That's why the operating system acts as manager of these resources. Another view of the operating system is it is a control program. A control program manages the execution of user programs to present the errors in proper use of the computer. It is especially concerned of the user the operation and controls the I/O devices.

Operating Systems Booting

OS boot, or operating system boot, is the process of starting up a computer and loading the operating system into memory. The boot process typically involves the following steps:

1. The BIOS (Basic Input/Output System) is loaded from ROM (Read-Only Memory).
2. The BIOS performs a power-on self-test (POST) to check the hardware for errors.
3. The BIOS loads the boot loader from the boot device.
4. The boot loader loads the operating system kernel into memory.
5. The kernel starts the operating system.

The boot device is the storage device from which the operating system is loaded. A modern PC's BIOS firmware supports booting from various devices, typically a local solid state drive or hard disk drive via the GPT or Master Boot Record (MBR) on such a drive or disk, an optical disc drive (using El Torito), a USB mass storage device (USB flash drive, memory card reader, USB hard disk drive, USB optical disc drive, USB solid state drive, etc.), or a network interface card (using PXE).

There are two types of booting:

- Cold boot: This is when the computer is turned on for the first time or after it has been shut down.
- Warm boot: This is when the computer is restarted.

The boot process can be affected by a number of factors, including the hardware configuration, the operating system, and the boot device. If there are any problems with the boot process, the computer may not start up or it may start up in a limited mode.

Here are some of the common problems that can occur during the OS boot process:

- Missing or corrupt boot files: This can prevent the operating system from loading.
- Hardware errors: These can prevent the BIOS from loading or the operating system from starting up.
- Incorrect boot device configuration: This can cause the computer to boot from the wrong device.

- Viruses or malware: These can corrupt the boot files or prevent the operating system from starting up.

If you are having problems with the OS boot process, you can try the following troubleshooting steps:

- Check the hardware configuration: Make sure that all of the hardware is properly connected and that the BIOS settings are correct.
- Reinstall the operating system: This will replace the boot files and any other corrupted files.
- Run a virus scan: This will check for viruses or malware that may be preventing the operating system from starting up.

If you are still having problems after trying these steps, you may need to contact a computer technician for help.

Overview of Operating Systems:

An operating system (OS) is a software that manages computer hardware and software resources and provides common services for computer programs.

The OS is a vital part of the computer system, and it is responsible for the following tasks:

- Resource allocation: The OS allocates hardware resources, such as CPU time, memory, and I/O devices, to computer programs.
- Process management: The OS manages the execution of computer programs, including creating, suspending, and terminating processes.
- Memory management: The OS manages the memory of the computer system, including allocating and deallocating memory to computer programs.
- File management: The OS manages the files on the storage devices of the computer system, including creating, reading, writing, and deleting files.
- I/O management: The OS manages the I/O devices of the computer system, including controlling the flow of data between the devices and the computer programs.
- Security: The OS provides security mechanisms to protect the computer system from unauthorized access.

The OS is a complex piece of software, and it is typically divided into two main parts: the kernel and the user interface.

- The kernel: The kernel is the core of the OS, and it is responsible for the most important tasks, such as resource allocation, process management, and memory management.
- The user interface: The user interface provides a way for users to interact with the OS, and it typically includes a graphical user interface (GUI) or a command-line interface (CLI).

There are many different operating systems available, each with its own strengths and weaknesses. Some of the most popular operating systems include:

- Windows: Windows is the most popular operating system in the world, and it is used by millions of people on personal computers, laptops, and tablets.
- Linux: Linux is a free and open-source operating system that is used by millions of people on servers, desktops, and embedded devices.
- macOS: macOS is the operating system used by Apple computers, and it is known for its user-friendly interface and its integration with other Apple products.
- Android: Android is a mobile operating system that is used by billions of people on smartphones and tablets.
- iOS: iOS is the operating system used by Apple iPhones and iPads, and it is known for its smooth performance and its wide range of apps.

The choice of operating system depends on the needs of the user. For example, Windows is a good choice for personal computers, while Linux is a good choice for servers. macOS is a good choice for Apple computers, while Android and iOS are good choices for mobile devices.

OS History Primer:

The history of operating systems can be divided into four generations:

First generation (1945-1955): The first generation of operating systems were simple batch systems that ran one program at a time. These systems were used for scientific computing and were not very user-friendly.

Second generation (1955-1965): The second generation of operating systems introduced the concept of multiprogramming, which allowed multiple programs to be loaded into memory and run concurrently. These systems were still not very user-friendly, but they were a significant improvement over the first generation.

Third generation (1965-1980): The third generation of operating systems introduced the concept of time-sharing, which allowed multiple users to share a single computer system. These systems were much more user-friendly than the previous generations and were used for a variety of applications, including business, education, and research.

Fourth generation (1980-present): The fourth generation of operating systems introduced the concept of graphical user interfaces (GUIs), which made it much easier for users to interact with the operating system. These systems are still used today for a variety of applications, including personal computing, servers, and embedded systems.

Some of the most important operating systems from each generation include:

First generation:

- UNIVAC I: The first commercial computer, which was released in 1951.
- IBM 7090: A mainframe computer that was released in 1956.

Second generation:

- CP/M: A popular operating system for microcomputers that was released in 1971.
- Multics: A large-scale operating system that was developed by MIT, Bell Labs, and General Electric.

Third generation:

- UNIX: An operating system that was developed at Bell Labs in the 1970s.
- MS-DOS: An operating system for personal computers that was released by Microsoft in 1981.

Fourth generation:

- Windows: An operating system for personal computers that was released by Microsoft in 1985.
- macOS: An operating system for Apple computers that was released in 1984.
- Linux: A free and open-source operating system that was released in 1991.

The history of operating systems is a long and complex one, but it is clear that these systems have played a vital role in the development of computing. Operating systems have made it possible for people to use computers for a variety of tasks, and they continue to evolve as new technologies emerge.

OS History:

Main-frame Systems:

Batch processing is a method of processing programs in which a group of jobs are submitted to the system together and then executed one after the other. This type of processing is typically used for jobs that do not require interaction with the user, such as scientific computations or batch data processing.

Multiprogramming is a method of processing programs in which multiple programs are loaded into memory and executed concurrently. This type of processing allows the CPU to be utilized more efficiently, as it is not waiting for a single program to finish before executing the next one.

Multitasking is a logical extension of multiprogramming. In multitasking systems, the CPU executes multiple tasks by switching between them typically using a small time quantum. This allows the user to interact with the system while other tasks are running in the background.

Time-sharing is a special case of multitasking in which multiple users are able to interact with the system at the same time. This is achieved by dividing the CPU time between the users, so that each user gets a small amount of time to execute their tasks.

Mainframe systems are typically used for batch processing and time-sharing applications. Batch processing is well-suited for mainframe systems because it allows the system to be utilized efficiently for long-running jobs. Time-sharing is also well-suited for mainframe systems because it allows multiple users to share the system resources.

Here is a table that summarizes the key differences between batch processing, multiprogramming, and multitasking or time-sharing systems

Feature	Batch processing	Multiprogramming	Multitasking or time-sharing
Number of programs executed at a time	One	Multiple	Multiple
Interaction with the user	No	No	Yes
CPU utilization	Efficient	Efficient	Less efficient
Suitable applications	Scientific computations, batch data processing	Long-running jobs	Interactive applications

Desktop Systems:

Here are some of the most important desktop operating systems that have been released over the years:

- CP/M: CP/M was one of the first commercially successful desktop operating systems. It was released in 1971 and was used on a variety of early personal computers, including the Altair 8800 and the IBM PC.
- DOS: DOS was a popular operating system for personal computers in the 1980s and early 1990s. It was developed by Microsoft and was based on CP/M.
- Windows: Windows is the most popular desktop operating system in the world. It was first released in 1985 and has been through many different versions since then.
- macOS: macOS is the operating system used on Apple computers. It was first released in 1984 and is based on Unix.
- Linux: Linux is a free and open-source operating system that is based on Unix. It was first released in 1991 and is now used on a wide variety of devices, including personal computers, servers, and smartphones.

These are just a few of the most important desktop operating systems that have been released over the years. The history of desktop operating systems is a long and complex one, but it is clear that these systems have played a vital role in the development of personal computing.

Network Systems:

LAN (Local Area Network) is a computer network that connects computers within a limited geographical area, such as a home, office, or school. LANs are typically used to share resources, such as printers, files, and internet access.

WAN (Wide Area Network) is a computer network that connects computers over a large geographical area, such as a city, state, or country. WANs are typically used to connect different LANs together.

MAN (Metropolitan Area Network) is a computer network that connects computers within a metropolitan area, such as a city or town. MANs are typically used to connect different LANs together and to provide high-speed internet access.

SAN (Storage Area Network) is a high-speed network that connects storage devices, such as disk arrays and tape drives. SANs are typically used to store and manage large amounts of data.

From an OS viewpoint, LAN, WAN, MAN, and SAN are all important network types that can be used to connect computers and share resources. The OS provides the software that allows these networks to function. For example, the OS provides the drivers that allow the

computers to connect to the network and the protocols that allow the computers to communicate with each other.

Network type	Geographical area	Typical use
LAN	Limited	Sharing resources, internet access
WAN	Large	Connecting LANs, internet access
MAN	Metropolitan	Connecting LANs, high-speed internet access

Multiprocessor Systems:

Multiprocessor systems are computer systems that have multiple processors. These systems can be used to improve performance, fault tolerance, and scalability.

Fault tolerance is the ability of a system to continue to operate even if one or more of its components fail. In a multiprocessor system, fault tolerance can be achieved by using redundant processors. If one processor fails, the other processors can continue to operate.

Useful degradation is the ability of a system to continue to operate even if its performance is degraded. In a multiprocessor system, useful degradation can be achieved by using load balancing. This means that the workload is distributed evenly among the processors, so that the failure of one processor does not significantly affect the performance of the system.

Symmetric multiprocessing (SMP) is a type of multiprocessor system in which all of the processors are equal. This means that they can all access the same memory and I/O devices. SMP systems are typically used for general-purpose computing.

Asymmetric multiprocessing (AMP) is a type of multiprocessor system in which the processors are not equal. This means that some of the processors may have more privileges than others. AMP systems are typically used for specialized applications, such as servers.

Loosely coupled multiprocessor systems are systems in which the processors are not tightly integrated. This means that they may not share memory or I/O devices. Loosely coupled systems are typically used for distributed computing applications.

Tightly coupled multiprocessor systems are systems in which the processors are tightly integrated. This means that they share memory and I/O devices. Tightly coupled systems are typically used for real-time applications.

From an OS viewpoint, multiprocessor systems present a number of challenges. These challenges include:

- Synchronization: The OS must ensure that the processors do not access the same data at the same time.
- Scheduling: The OS must schedule the processors to ensure that they are all utilized efficiently.
- Fault tolerance: The OS must be able to handle the failure of one or more processors.

The OS can use a variety of techniques to address these challenges. These techniques include:

- Mutexes: Mutexes are used to protect shared data from concurrent access.
- Semaphores: Semaphores are used to control the access of multiple processes to a shared resource.
- Scheduling algorithms: Scheduling algorithms are used to determine which processor should execute a particular task.
- Fault-tolerant mechanisms: Fault-tolerant mechanisms are used to ensure that the system can continue to operate even if one or more processors fail.

Clustered Systems:

A clustered system is a group of independent computers that are connected together to work as a single system. These systems are typically used to improve performance, fault tolerance, and scalability.

From an OS viewpoint, clustered systems present a number of challenges. These challenges include:

- Synchronization: The OS must ensure that the nodes in the cluster do not access the same data at the same time.
- Scheduling: The OS must schedule the nodes in the cluster to ensure that they are all utilized efficiently.
- Fault tolerance: The OS must be able to handle the failure of one or more nodes in the cluster.

Distributed Systems:

A distributed operating system (DOS) is a software system that manages a group of independent computers that are connected together to work as a single system. These systems are typically used to improve performance, fault tolerance, and scalability.

From an OS viewpoint, distributed operating systems present a number of challenges. These challenges include:

- Communication: The OS must be able to communicate between the different nodes in the distributed system.
- Synchronization: The OS must ensure that the different nodes in the distributed system are synchronized so that they can work together.
- Security: The OS must be able to secure the distributed system from unauthorized access.

The OS can use a variety of techniques to address these challenges. These techniques include:

- Communication protocols: The OS can use communication protocols to communicate between the different nodes in the distributed system.
- Synchronization mechanisms: The OS can use synchronization mechanisms to ensure that the different nodes in the distributed system are synchronized.
- Security mechanisms: The OS can use security mechanisms to secure the distributed system from unauthorized access.

Here are some of the benefits of using distributed operating systems:

- Performance: Distributed operating systems can improve performance by distributing the workload across multiple nodes.
- Fault tolerance: Distributed operating systems can improve fault tolerance by continuing to operate even if one or more nodes fail.
- Scalability: Distributed operating systems can be scaled up by adding more nodes.

Here are some of the challenges of using distributed operating systems:

- Complexity: Distributed operating systems can be complex to design, implement, and manage.
- Cost: Distributed operating systems can be more expensive than single-node operating systems.
- Security: Distributed operating systems can be more vulnerable to security attacks, as they are typically connected to the internet.

Embedded Systems:

The Embedded operating system is the specific purpose operating system used in the computer system's embedded hardware configuration. These operating systems are designed to work on dedicated devices like automated teller machines (ATMs), airplane systems, digital home assistants, and the internet of things (IoT) devices.

Real-time Systems:

A real-time system is a system that must respond to events within a specified time constraint. These systems are typically used in applications where the timely response to events is critical, such as air traffic control, medical imaging, and industrial control.

From an OS viewpoint, real-time systems present a number of challenges. These challenges include:

- Timing: The OS must ensure that the system responds to events within the specified time constraint.
- Scheduling: The OS must schedule the tasks in the system to ensure that they are all executed on time.
- Preemption: The OS must be able to preempt tasks that are running late to ensure that the system meets its timing constraints.

The OS can use a variety of techniques to address these challenges. These techniques include:

- Time slicing: Time slicing is a scheduling algorithm that allows each task to run for a short period of time before being preempted by another task.
- Priority-based scheduling: Priority-based scheduling is a scheduling algorithm that assigns each task a priority and then schedules the tasks in order of priority.
- Deadline scheduling: Deadline scheduling is a scheduling algorithm that assigns each task a deadline and then schedules the tasks to ensure that they are all executed before their deadlines.

There are two main types of real-time systems: hard real-time systems and soft real-time systems.

- Hard real-time systems are systems in which missing a deadline can have catastrophic consequences. These systems are typically used in applications where human safety is at stake, such as air traffic control and medical imaging.
- Soft real-time systems are systems in which missing a deadline can have undesirable consequences, but the system will not be catastrophically affected. These systems are typically used in applications where the timely response to events is important, but not critical, such as industrial control and multimedia streaming.

Here are some of the benefits of using real-time systems:

- Timeliness: Real-time systems can ensure that events are responded to within a specified time constraint.

- Reliability: Real-time systems can be more reliable than non-real-time systems, as they are designed to handle unexpected events.
- Safety: Real-time systems can be used in applications where human safety is at stake, as they are designed to respond to events in a timely manner.

Here are some of the challenges of using real-time systems:

- Complexity: Real-time systems can be complex to design, implement, and test.
- Cost: Real-time systems can be more expensive than non-real-time systems.
- Performance: Real-time systems may not be as efficient as non-real-time systems, as they are designed to respond to events in a timely manner.

Hand-held Systems:

Handheld systems are small, portable computers that are typically used for personal productivity, entertainment, and communication. These systems are typically powered by batteries and have limited resources, such as memory and processing power.

From an OS viewpoint, handheld systems present a number of challenges. These challenges include:

- Power management: Handheld systems must be designed to conserve power, as they are typically powered by batteries.
- Resource management: Handheld systems must be designed to efficiently manage their limited resources, such as memory and processing power.
- User interface: Handheld systems must have a user interface that is easy to use with a small touchscreen.

The OS can use a variety of techniques to address these challenges. These techniques include:

- Power management: The OS can use techniques such as dynamic power management and sleep mode to conserve power.
- Resource management: The OS can use techniques such as memory management and process scheduling to efficiently manage the system's resources.
- User interface: The OS can use techniques such as gesture recognition and voice input to make the user interface more accessible.

Here are some of the benefits of using handheld systems:

- Portability: Handheld systems are small and portable, making them easy to carry around.
- Accessibility: Handheld systems can be used in a variety of environments, such as on the go or in crowded places.
- Personalization: Handheld systems can be personalized to the user's needs, such as by installing different apps and widgets.

Here are some of the challenges of using handheld systems:

- Limited resources: Handheld systems have limited resources, such as memory and processing power.
- Security: Handheld systems can be more vulnerable to security attacks, as they are typically connected to the internet.
- Battery life: Handheld systems typically have short battery life, so they need to be charged frequently.

Device Organization:

Device organization in operating systems is the way that the operating system (OS) manages the hardware devices in a computer system. The OS provides an abstraction of the hardware devices to the user, so that the user can interact with the devices without having to know the details of how the devices work.

There are two main types of device organization: monolithic and layered.

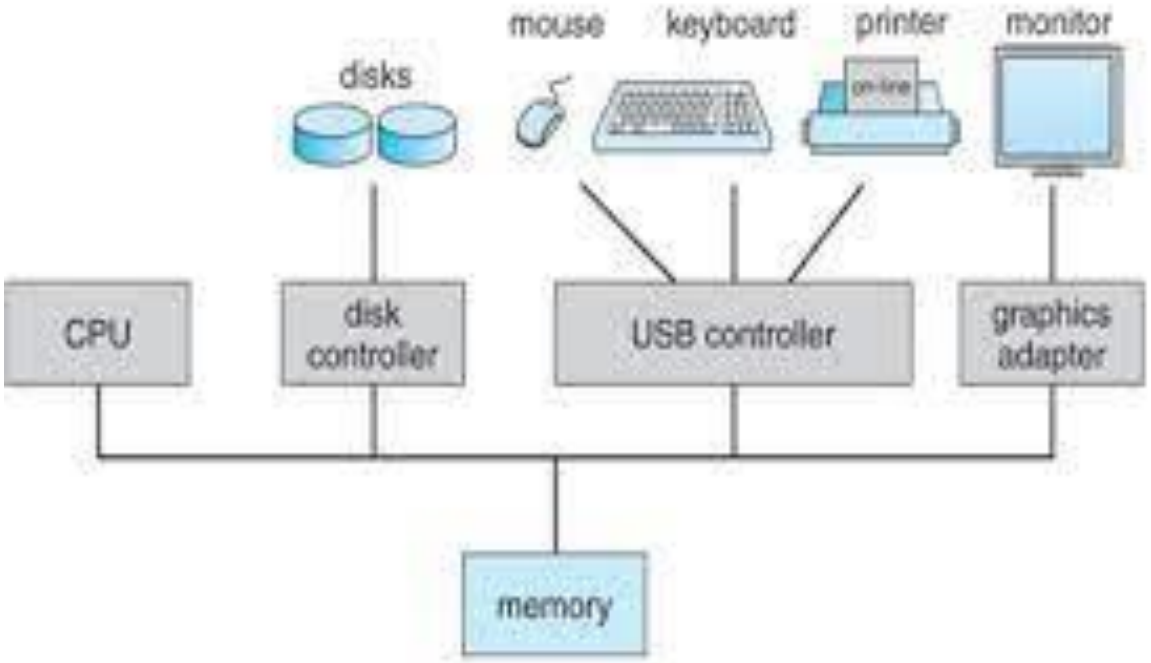
- Monolithic device organization is the simplest type of device organization. In monolithic device organization, the OS has a single module that handles all of the device drivers. This makes the OS simpler to design and implement, but it can also make the OS less modular and less extensible.
- Layered device organization is a more complex type of device organization. In layered device organization, the OS has a layered architecture, with each layer responsible for a different set of devices. This makes the OS more modular and extensible, but it can also make the OS more complex to design and implement.

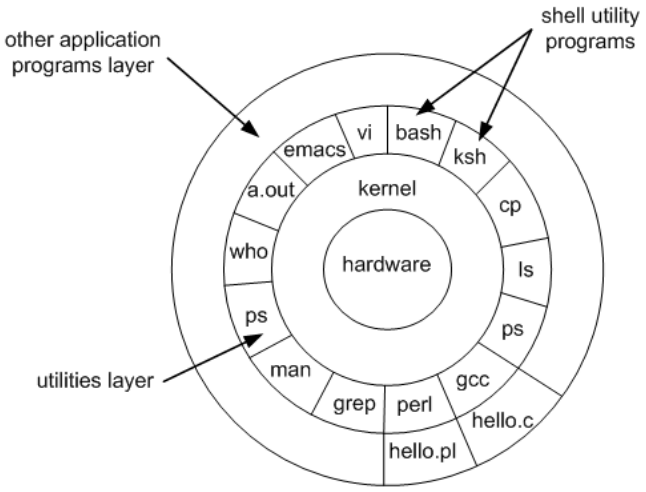
The OS uses a variety of techniques to manage the hardware devices in a computer system. These techniques include:

- Device drivers: Device drivers are software modules that allow the OS to interact with the hardware devices.
- Device files: Device files are special files that represent the hardware devices in the file system.

- Device interrupts: Device interrupts are signals that the hardware devices send to the OS when they need attention.
- I/O ports: I/O ports are special memory locations that the hardware devices use to communicate with the OS.

The OS uses these techniques to provide a consistent and efficient way for users to interact with the hardware devices in a computer system.





Basic Structure of Operating System

Interrupts:

An interrupt is a signal that a hardware device sends to the operating system (OS) when it needs attention. Interrupts are used to notify the OS of events that happen outside of the normal flow of execution, such as when a device needs to be read or written to.

The OS uses interrupts to handle a variety of tasks, such as:

- Input/output (I/O): Interrupts are used to handle I/O requests from devices, such as disks, keyboards, and mice.
- Exception handling: Interrupts are used to handle exceptions, such as divide-by-zero errors and page faults.
- Scheduling: Interrupts can be used to schedule the execution of processes.

There are two main types of interrupts: hardware interrupts and software interrupts.

- Hardware interrupts are generated by hardware devices. When a hardware device needs attention, it sends an interrupt signal to the CPU.
- Software interrupts are generated by software. When a program wants to notify the OS of an event, it can generate a software interrupt.

The OS handles interrupts by using an interrupt handler. An interrupt handler is a piece of code that is executed when an interrupt occurs. The interrupt handler typically performs the following tasks:

- Acknowledges the interrupt: The interrupt handler acknowledges the interrupt by sending a signal back to the device that generated the interrupt. This tells the device that the OS has received the interrupt and that it is now handling the request.
- Handles the interrupt: The interrupt handler handles the interrupt by performing the appropriate action. For example, if the interrupt was generated by a disk device, the interrupt handler might read or write data from the disk.
- Returns from the interrupt: Once the interrupt handler has finished handling the interrupt, it returns from the interrupt. This returns control back to the program that was interrupted.

The OS uses a variety of techniques to manage interrupts. These techniques include:

- Interrupt priority: Interrupts can be assigned priorities. This allows the OS to prioritize the execution of interrupt handlers.
- Interrupt masking: Interrupt masking allows the OS to disable interrupts temporarily. This can be used to prevent interrupts from interrupting the execution of critical code.

- Interrupt nesting: Interrupt nesting allows the OS to handle multiple interrupts at the same time. This is used to handle interrupts that occur while the OS is already handling another interrupt.

Kernel State Transitions:

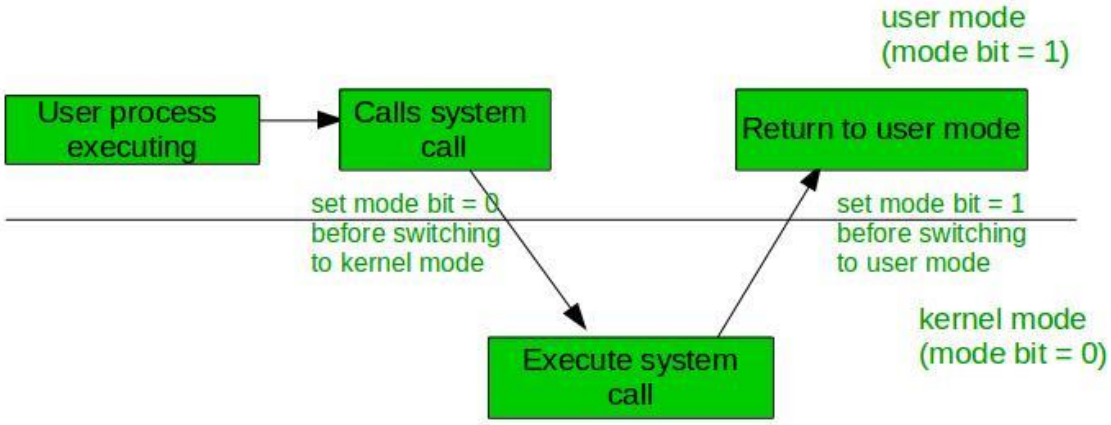
In computer architecture, kernel mode and user mode are two distinct modes of operation for a computer processor. Kernel mode is the most privileged mode, and user mode is less privileged. In kernel mode, the processor has access to all of the system resources, while in user mode, the processor only has access to a limited set of resources.

The transition between kernel mode and user mode is controlled by the operating system (OS). When a process needs to access a system resource that is only available in kernel mode, the process must make a system call. A system call is a special instruction that tells the OS to switch to kernel mode and execute a kernel routine.

The OS uses a variety of techniques to ensure that the transition between kernel mode and user mode is secure. These techniques include:

- Hardware support: The hardware provides support for kernel mode and user mode. For example, the hardware may have separate registers for kernel mode and user mode.
- Privileged instructions: Only privileged instructions can be executed in kernel mode. This prevents user processes from executing code that could damage the system.
- Memory protection: The OS uses memory protection to prevent user processes from accessing kernel memory. This prevents user processes from reading or writing kernel data.

The transition between kernel mode and user mode is an important security feature of the OS. By ensuring that only privileged code can be executed in kernel mode, the OS can protect the system from malicious code.


user mode
(mode bit = 1)
User process executing
Calls system call
Return to user mode
set mode bit = 0 before switching to kernel mode
set mode bit = 1 before switching to user mode
kernel mode
(mode bit = 0)
Execute system call