S.Y.B.Sc. (C. S.)
SEMESTER - III (CBCS)

OPERATING SYSTEM

SUBJECT CODE: USCS303

Dr. Suhas Pednekar
Vice Chancellor
University of Mumbai, Mumbai

Prof. Ravindra D. Kulkarni
Pro Vice-Chancellor,
University of Mumbai

Prof. Prakash Mahanwar
Director,
IDOL, University of Mumbai

Programme Co-ordinator : **Shri Mandar Bhanushe**
Head, Faculty of Science and Technology IDOL,
Univeristy of Mumbai – 400098

Course Co-ordinator : **Mr Sumedh Shejole**
Assistant Professor,
IDOL, University of Mumbai- 400098

Editor : **Ms. Priya Jadhav,**
Assistant Professor,
N. G. Acharya & D. K. Marathe College,
Chembur, Mumbai, Maharashtra 400071

Course Writers : **Dr. Ninad More**
Chhatrapati Shivaji Maharaj University,
Panvel, Navi Mumbai

: **Mr. Santosh Patilshirke**
SJRS College
Kalwa, Thane, Maharashtra 400605

: **Ms. Rashmi Gera**
Pt. Jawahar Lal Nehru Govt. College,
Faridabad

August 2022, Print - 1

Published by : Director,
Institute of Distance and Open Learning,
University of Mumbai,
Vidyanagari,Mumbai - 400 098.

DTP composed and Printed by: Mumbai University Press

CONTENTS

S.Y.B.Sc. (C. S.)

SEMESTER - IIi (CBCS)

OPERATING SYSTEM

SYLLABUS

Course: USCS303	TOPICS (Credits : 02 Lectures/Week:03) Operating System	
Objectives: Learners must understand proper working of operating system. To provide a sound understanding of Computer operating system, its structures, functioning and algorithms. **Expected Learning Outcomes:** 1. To provide a understanding of operating system, its structures and functioning 2. Develop and master understanding of algorithms used by operating systems for various purposes.		
Unit I	**Introduction and Operating-Systems Structures:** Definition of Operating system, Operating System's role, Operating-System Operations, Functions of Operating System, Computing Environments **Operating-System Structures:** Operating-System Services, User and Operating-System Interface, System Calls, Types of System Calls, Operating-System Structure **Processes**: Process Concept, Process Scheduling, Operations on Processes, Interprocess Communication	**15L**

	Threads: Overview, Multicore Programming, Multithreading Models	
Unit II	**Process Synchronization:** General structure of a typical process, race condition, The Critical-Section Problem, Peterson's Solution, Synchronization Hardware, Mutex Locks, Semaphores, Classic Problems of Synchronization, Monitors **CPU Scheduling:** Basic Concepts, Scheduling Criteria, Scheduling Algorithms (FCFS, SJF, SRTF, Priority, RR, Multilevel Queue Scheduling, Multilevel Feedback Queue Scheduling), Thread Scheduling **Deadlocks:** System Model, Deadlock Characterization, Methods for Handling Deadlocks, Deadlock Prevention, Deadlock Avoidance, Deadlock Detection, Recovery from Deadlock	**15L**
Unit III	**Main Memory:** Background, Logical address space, Physical address space, MMU, Swapping, Contiguous Memory Allocation, Segmentation, Paging, Structure of the Page Table **Virtual Memory:** Background, Demand Paging, Copy-on-Write, Page Replacement, Allocation of Frames, Thrashing **Mass-Storage Structure:** Overview, Disk Structure, Disk Scheduling, Disk Management **File-System Interface:** File Concept, Access Methods, Directory and Disk Structure, File-System Mounting, File Sharing **File-System Implementation:** File-System Structure, File-System Implementation, Directory Implementation, Allocation Methods, Free-Space Management	**15L**

Textbook(s):

1. Abraham Silberschatz, Peter Galvin, Greg Gagne, Operating System Concepts, Wiley,8th Edition

Additional Reference(s):

1. Achyut S. Godbole, Atul Kahate, Operating Systems, Tata McGraw Hill
2. Naresh Chauhan, Principles of Operating Systems, Oxford Press
3. Andrew S Tanenbaum, Herbert Bos, Modern Operating Systems, 4e Fourth Edition, Pearson Education, 2016

1

OPERATING SYSTEM

Unit Structure

1.0 Objectives of Operating System

- The objectives of the operating system are −
- To make the computer system convenient to use in an efficient manner.
- To hide the details of the hardware resources from the users.
- To provide users a convenient interface to use the computer system.
- To act as an intermediary between the hardware and its users, making it easier for the users to access and use other resources.
- To manage the resources of a computer system.
- To keep track of who is using which resource, granting resource requests, and mediating conflicting requests from different programs and users.
- To provide efficient and fair sharing of resources among users and programs.

1.1 Introduction to Operating Systems

A computer system has many resources (hardware and software), which may be expected to do a task. The for the most part required resources are I/O gadget, memory, archive additional room, CPU, etc. The functioning structure goes probably as a director of the above resources and assigns them to express tasks and customers, whenever imperative to play out a particular endeavour. Along these lines the operating system is the resource manager for example it can deal with the asset of a PC framework inside. The assets are processor, memory, documents, and I/O device. In straightforward terms, a working framework is an interface between the PC client and the machine.

It is very important for you that each PC should have a working framework to run different projects. The operating system mainly coordinates the use of the hardware among the various system programs and application programs for various users.

An operating system acts additionally like government means an operating system a working framework plays out no valuable capacity without help from anyone else; however it gives a climate inside which different projects can accomplish valuable work.

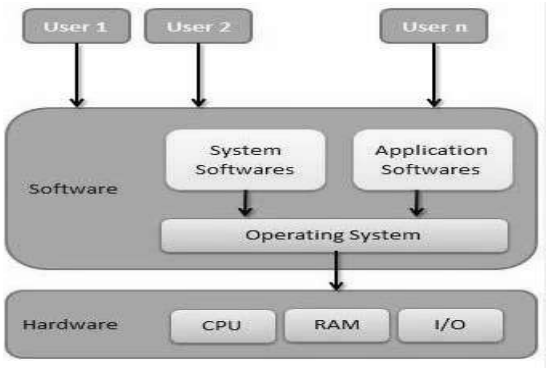
1.2 Introduction and Operating-Systems Structures:

1.2.1 Definition of Operating System

Operating system is s collection of set of programs which manages all the services of the computer system. It is a program that acts as an intermediary between users of a computer network.

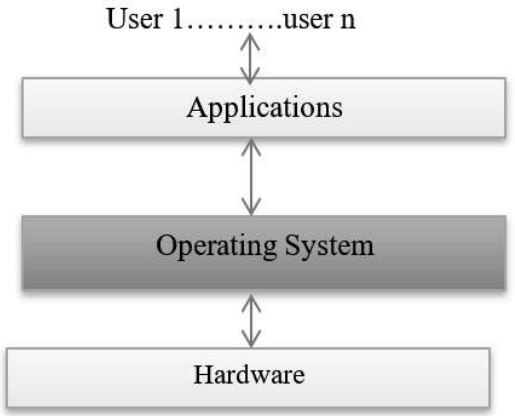
An operating system is a program which manages all the computer hardware's.

- It provides the base for application program and acts as an intermediary between a user and the computer hardware.
- The operating system has two objectives such as: Firstly, an operating system controls the computer's hardware. The second objective is to provide an interactive interface to the user and interpret commands so that it can communicate with the hardware.
- The operating system is very important part of almost every computer system



Operating System

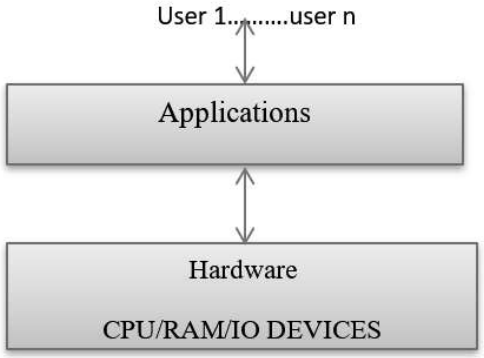
- It is a program that controls the execution of application program.
- Act as an interface between application and hardware.
- Execute user programs and make solving user problems easier
- Make the computer system convenient to use
- Use the computer hardware in an efficient manner



With operating system

Here, User can easily manage the hardware by using operating system.

User 1…….user n



Without Operating System

Here, User cannot easily manage the hardware by using without operating system.

Main objective of an operating system:

i. Convenience
ii. Efficiency
iii. Ability to solve the problems

1) Convenience: It provides users the services (Processor/Memory/Files and I/O etc.) to execute the programs in a convenient manner.

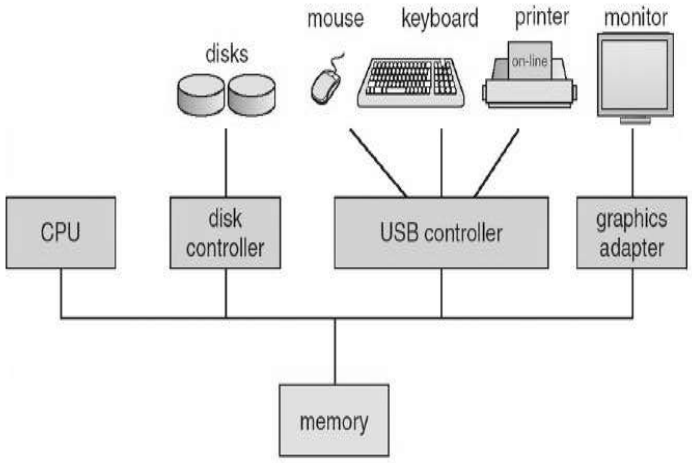
2) Efficiency: Operating system allows the computer system resources (Hardware/Software/Data/Network etc.) to be used efficiently.
3) Ability to solve the problems: Operating system should be performed in such a way to permit the effective deployment, testing and introduction of new system functions.

Computer Start-up

- bootstrap program is loaded at power-up or reboot
- Typically stored in ROM or EPROM, generally known as firmware
- Initializes all aspects of system
- Loads operating system kernel and starts execution

Computer System Organization

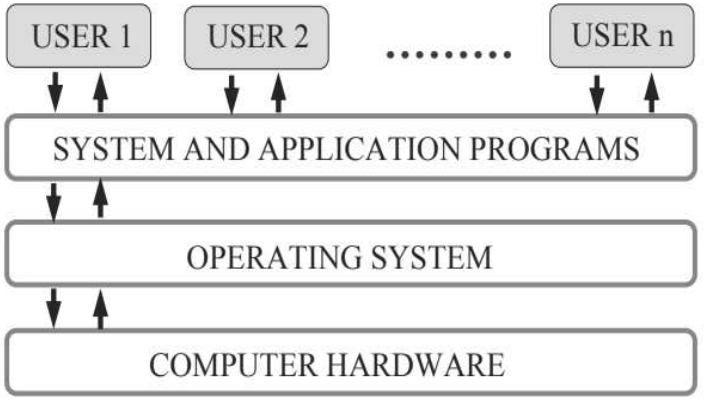
- Computer-system operation
- One or more CPUs, device controllers connect through common bus providing access to shared memory
- Concurrent execution of CPUs and devices competing for memory cycles



Computer System Organization

1. 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 client sit at a terminal associated with a centralized server/minicomputer. For this situation different clients are getting to similar PC through different terminals. Their clients are share assets and may trade the data. The operating system in this case is designed to maximize resources utilization to assume that all available CPU time, memory and I/O are utilized proficiently and no singular client takes more than his/her reasonable and share. The different clients sit at workstations associated with 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 assets as equipment and programming which might be needed to tackle an issue 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.



View of operating system

2. History of operating system:

The First Generation (1940's early 1950's):

1) When electronic computers were first introduced in the 1940's.
2) Computer without any operating system.
3) During this generation computers were generally used to solve simple math calculation.

The Second Generation (1955 – 1965)

1) The first OS was introduced in the early 1956's.
2) It was GM-NAA I/O was made by (General Motors for IBM'S machine 704)
3) OS in the 1956's were called single stream batch processing system.

The Third Generation (1966 -1980)

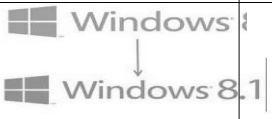
1) By the late 1960's OS designers were able to develop the system of multiprogramming in which computer able to perform multiple tasks at the same time.
2) Mini PCs beginning with DEC PDP-1(Digital Equipment Corporation's - Programmed Data Processor-1) in 1961.The PDP-1 had just 4K of 18 bit words yet at $120,000 per machine.
3) These PDP'S helped lead to the making of PCs which are made in the fourth era/ generation.

The Fourth Generation (1980- Present Day)

1) The fourth generation of operating system saw the creation of personal computing.
2) These PCs very similar to the mini computer.
3) Windows operating system was created in 1975.
4) They introduced the MS-DOS in 1981.
5) Windows went on to become the largest operating system used in technology today.
6) Along with Microsoft, Apple is the other operating system created in 1980's.

Different versions of Microsoft Windows

Windows 95		August 1995
Windows 98		June 1998
Windows ME - Millennium Edition		September 2000
Windows NT 31 - 4.0		1993-1996
Windows 2000		February 2000

Windows XP		October 2001
Windows Vista		November 2006
Windows 7		October, 2012
Windows 8 and 8.1		October, 2012
Windows 10		July 2015
Windows 11		2021

1.2.2 Operating System Role's:

An operating system provides an environment in which application software's such as(word processors, Database S/W, Graphic S/W, Spread sheet S/W, Presentation S/W, Web browsers, Microsoft access, Photoshop etc.) can be installed.

So an operating system (Like Windows, Ubuntu, and Linux, CentOS, Debian and so on) provides an environment in which a user can perform a task. Example formulate my information in suitable format i.e. say a word file using an application software like Microsoft word and save it onto my hardware (Hard Disk) in a convenient and efficient manner.

Operating System Diagram User

Computer-System Operation

1) Input /Output devices, and the CPU can execute concurrently
2) Every device controller is in charge of a particular device type
3) All the device controller has a local buffer
4) CPU moves data from/to main memory to/from local buffers
5) The Input/Out is from the device to the local buffer of the controller
6) The Device controller informs the CPU that it has completed its activity by causing an *interrupt*

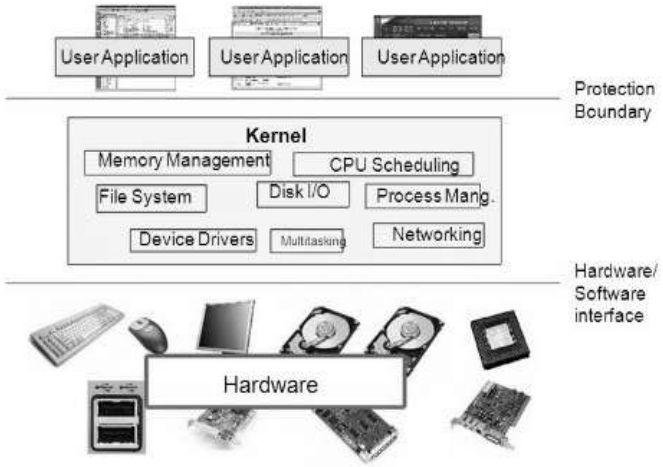
Common Functions of Interrupts

1) The Interrupt transfers handle to the interrupt service routine generally, through the interrupt vector, which carries the addresses of all the service routines
2) The Interrupt architecture must save the address of the interrupted instruction
3) The Incoming interrupts are *disabled* while another interrupt is being- processed to prevent a *lost interrupt.* A *trap* is a software-generated interrupt caused either by an error or a user request an OS is interrupt-driven

1.2.3 Operating system operations

1) Interrupt driven by hardware
2) Software error or request creates exception or trap
3) Division by zero, request for operating system service
4) Other process problems include infinite loop, processes modifying each Other or the operating system

5) In the Dual-mode operation allows operating System(OS) to protect itself and to the others system components
6) In the User mode and kernel mode
7) Mode bit provided by hardware



Operating system operations

Dual-Mode Operation:

In the order to ensure the proper execution of the OS (operating system), we must be able to distinguish between the execution of operating-system code and user-defined code. The methodology taken by most PC frameworks is to give equipment support that permits us to separate among different methods of execution. At any rate, we need two separate methods of activity: client mode and piece mode (likewise called director mode, framework mode, or favoured mode). A bit, called the mode bit, is added to the equipment of the PC to show the current mode: kernel (0) or client (1). With the mode bit, we can recognize an errand that is executed for the benefit of the working framework and one that is executed for the client.

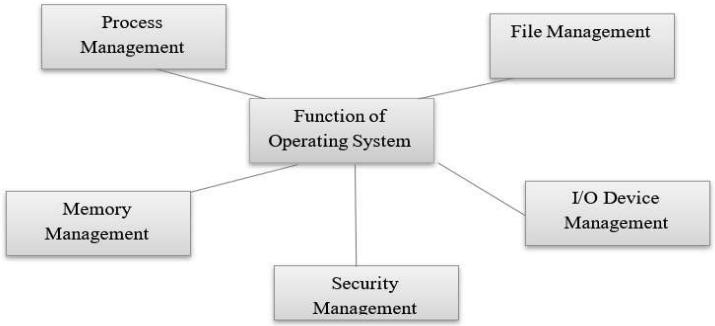
- When the computer system is executing on behalf of a client application, the system is in user mode.
- However, when a user application requests a service from the operating system (via a system call), it must transition from user to kernel mode to full-fill the request. As we shall see, this architectural

enhancement is useful for many other aspects of system operation as well.

- At system boot time, the hardware starts in kernel mode.
- The operating system (OS) is then loaded and starts the user applications in the user mode.
- Whenever a trap or interrupt occurs, the hardware switches from the user mode to kernel mode (that is, changes the state of the mode bit to 0).
- Thus, whenever the operating system gains control of the computer, it is in kernel mode.
- The system always switches to user mode (by setting the mode bit to 1) before passing control to a user program.

The dual mode of operation provides us with the means for protecting the operating system (OS) from deviant clients and wayward clients from one another. We achieve this insurance by assigning a portion of the machine directions that might cause hurt as favoured guidelines. The equipment permits advantaged directions to be executed uniquely in bit mode. In the event that an endeavour is made to execute a favoured guidance in client mode, the equipment doesn't execute the guidance yet rather regards it as unlawful and traps it to the operating system. The instruction to switch to user mode is an example of a privileged instruction. Some other examples include I/O (Input/ Output) control, timer management, and interrupt management. As we shall see throughout the text, there are many additional privileged instructions.

1.2.4 Function of Operating System:



Function of Operating System

1) Process Management:

- In process management, operating systems do the management of CPU.
- The operating system takes care of the allotment of CPU to different processes.
- When CPU is free, operating system selects a process from job queue and allocates the CPU to the process.
- When process execution will complete, operation system free the processor and again select another process for execution.
- Selection of process form job queue is done by using various CPU scheduling techniques like FCFS, SJ, ROUND ROBIN etc.

2) File Management:

File: A file is a logical related collection of information. File is stored in secondary storage (Magnetic Disk, Tape, Optical Disk etc.). The operating system (OS) manages the files, folders and directory systems on a computer. An operating system does the following activities for file management.

- Creating and deleting files
- Creating and deleting directories
- Operating systems (OS) keep information of filers using file allocation table (FAT)
- Operating system takes care that files are opened with proper access rights (Read /RW).
- File manager of operating system (OS) helps to create, edit, copy, allocate memory to the files and also update the file allocation table FAT.

3) Memory Management:

- Memory is the large array of words or bytes each with its own address.
- Main memory is directly accessed by CPU.
- For a program to be executed it must be in the main memory.
- An operating system does the following activities for memory management.

1) Keeps track of primary memory using free space management (Which partition is used and which is free).
2) Allocates the memory to process when it requests it.
3) De-allocates the memory when a process terminated and the same area is allocated to another process.

4. I/O Device Management:

- Operating system (OS) manages I/O devices and makes the I/O process effective.

- Operating system (OS) accepts the input from the I/P device(Keyboard) stores it in main memory, ask the CPU to process it and finally provides the result to the devices(Screen /Printer) for output.
- Operating systems (OS) controls all I/O devices attached to computer with the help of small software called device drivers.
- Operating system (OS) track of all devices with the help of I/O controller.
- Operating decides which process gets the device when and for how much time.
- Allocate and de-allocate device in the efficient way.

5. Security Management:

There are various virus threats which an interface with the normal operations of computer and can be very harmful and result in loss of data or system crashes.

Operating system of a computer has a number of built in tools to protect against security threats including the use of virus scanning utilities and setting up a firewall to block suspicious network activity. Operating system (OS) also helps to prevent unauthorized access to programs and data by use of password in user login etc.

1.2.5 Computing Environments

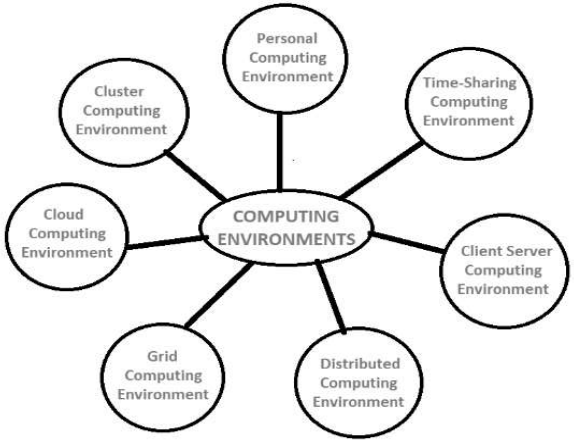
In the world of technology where every task is performed with help of computers, these computers have become one part of human life. Computing is nothing but process of completing a task by using this computer technology and it may involve computer hardware and/or software. But computing uses some form of computer system to manage, process, and communicate information. After getting some idea about computing now let's understand about computing environments.

When a problem is solved by the computer, during that computer uses many devices, arranged in different ways and which work together to solve problems. This constitutes a computing environment where various number of computer devices arranged in different ways to solve different types of problems in different ways. In different computing environments computer devices are arranged in different ways and they exchange information in between them to process and solve problem. One computing environment consists of many computers other computational devices, software and networks that to support processing and sharing information and solving task.

Based on the organization of different computer devices and communication processes there exists multiple types of computing environments. Now let's know about different types of computing environments.

Types of Computing Environments:

They are the various types of computing environments. They are:



Types of Computing Environments

1) Personal Computing Environment :

In individualized computing climate there is an independent machine. Complete program lives on PC and executed there. Diverse independent machines that establish an individualized computing climate are PCs, mobiles, printers, PC frameworks, scanners and so forth that we use at our homes and workplaces.

2) Time-Sharing Computing Environment :

In Time Sharing Computing Environment multiple users share system simultaneously.Different users (different processes) are allotted different time slice and processor switches rapidly among users according to it, for example, student listening to music while coding something in an IDE. Windows 95 and later versions, UNIX, IOS, Linux operating systems (OS) are the examples of this time sharing computing environment.

3) Client Server Computing Environment :

In client server computing environment two machines are involved i.e., client machine and server machine, sometime same machine also serve as client and server. In this computing environment client requests resource/service and server provides that respective resource/service. A server can provide service to multiple clients at a time and here mainly communication happens through computer network.

4) Distributed Computing Environment :

In a distributed computing environment multiple nodes are connected together using network but physically they are separated. A single task is performed by different functional units of different nodes of distributed unit. Here different programs of an application run simultaneously on different nodes, and communication happens in between different nodes of this system over network to solve task.

5) Grid Computing Environment :

In grid computing environment, multiple computers from different locations work on single problem. In this system set of computer nodes running in cluster jointly perform a given task by applying resources of multiple computers/nodes. It is network of computing environment where several scattered resources provide running environment for single task.

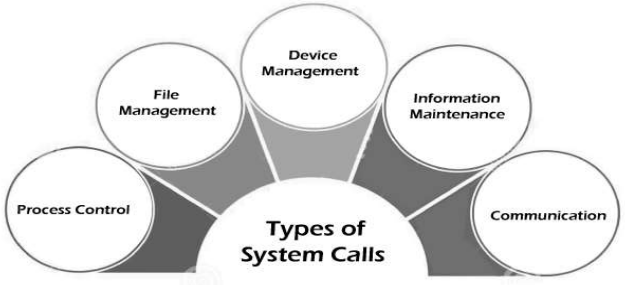
6) Cloud Computing Environment :

In cloud computing environment on demand availability of computer system resources like processing and storage are availed. Here computing is not done in individual technology or computer rather it is computed in cloud of computers where all required resources are provided by cloud vendor. This environment primarily comprised of three services i.e software-as-a-service (SaaS), infrastructure-as-a-service (IaaS), and platform-as-a-service (PaaS).

7) Cluster Computing Environment :

In cluster computing environment cluster performs task where cluster is a set of loosely or tightly connected computers that work together. It is viewed as single system and performs task parallelly that's why also it is similar to parallel computing environment. Cluster aware applications are especially used in cluster computing environment.

1.2.6 Types of operating system:

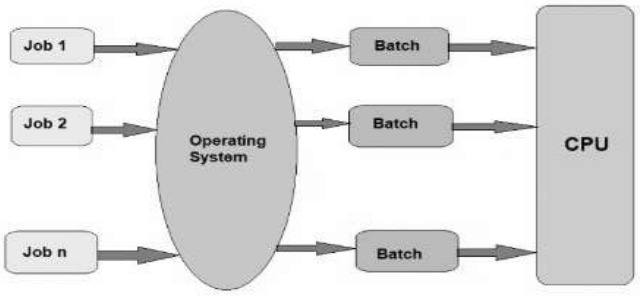


1) Batch processing system

2) Time sharing operating system
3) Distributed operating system
4) Network operating system
5) Real time operating system
6) Multiprogramming Operating System
7) Multiprocessor Operating Systems

1) Batch processing system:

The batch operating system will work to submit similar kinds of job tighter. In this operating system user do not interact directly with our computer system.



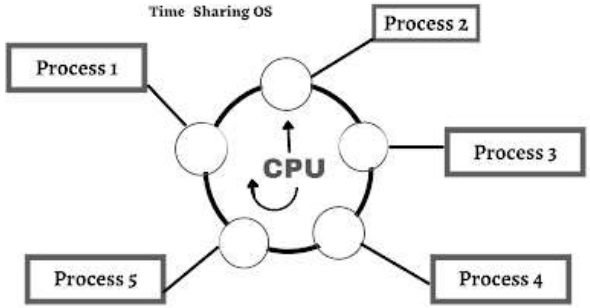
Batch Operating System

Advantage:
It is useful when we working with large files which can take move time to execute.
Disadvantage:

Once the job is submitted user did not have any interaction with it.

2) Time sharing operating system:

The time sharing sharing operating system works with time sharing concept. Here CPU will provide a same time period to each and every process to complete its task as soon as possible weather it is a long process or short process.

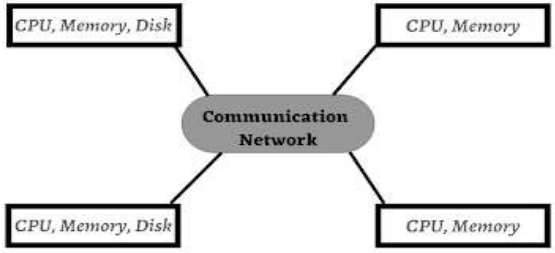


Time sharing operating system

Advantage:
If the process of a tasks are completed then the time between the other task increases
Disadvantage:
In this operating system every process having higher priority will not get the chance to be executed first priority because in time sharing operating system given equal opportunity to each process.

3) Distributed operating system

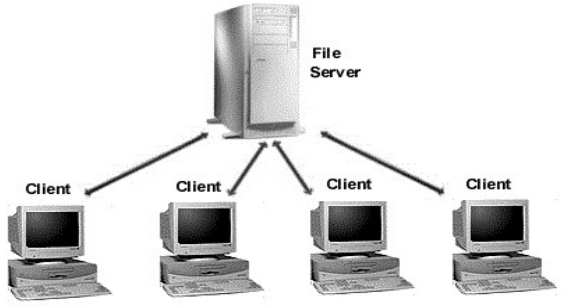
When many computers are interconnected each other through a network for the purpose of sharing their task then it is called distributed operating system.



Advantage:
If a node is overused the distributed operating system shares that load to other node of the network.
Disadvantage:
If there is a problem in the communication network then the whole communication will be broken.

4) Network operating system:

Network operating system has a server that connects many client computers. So we can easily share own files, resources and many more from the server machine to all machine which are connected through a server.



Network operating system

Advantage:

In network operating system, new technologies software Up gradation and hardware can be easily integrated into the computer system.

Disadvantage:

High cost of server and regular maintenance is required.

5) Real time operating system:

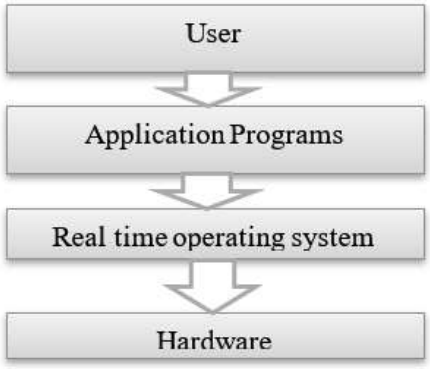
Real time operating system is very useful where we required a quick response. Example is missile system. In this operating system CPU provides maximum offers to its tasks with quick response.

Advantage:

The real time operating system provide a quick response hence, generally used in scientific engineering, NASA and many more organizations.

Disadvantage:

This operating system is very costly.



Real time operating system

6) Multiprogramming Operating System

This kind of OS is utilized to execute more than one position at the same time by a solitary processor. It expands CPU usage by getting

sorted out positions so the CPU consistently has one task to execute. The idea of multiprogramming is portrayed as follows:

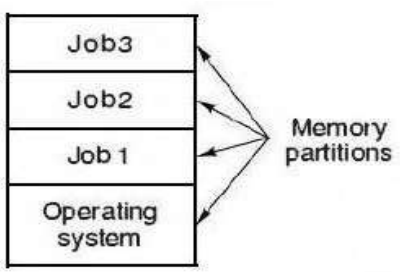
- All the positions that enter the framework are put away in the work pool (in plate). The working framework stacks a bunch of occupations from work pool into primary memory and starts to execute.
- during execution, the work might need to sit tight for some undertaking, like an I/O activity, to finish. In a multiprogramming framework, the working framework just changes to another work and executes. At the point when that work needs to stand by, the CPU is changed to another work, etc.
- When the primary occupation completes the process of pausing and it gets the CPU back.
- as long as no less than one occupation needs to execute, the CPU is rarely inactive. Multiprogramming working frameworks utilize the instrument of occupation planning and CPU booking.

Benefits of multiprogramming frameworks

1) The CPU is utilized a large portion of time and never become inactive
2) The framework looks quick as every one of the errands runs in equal
3) Short time occupations are done quicker than long time tasks
4) Multiprogramming frameworks support duplicate clients
5) Resources are utilized pleasantly
6) Total read time taken to execute program/work diminishes
7) Response time is more limited
8) In a few applications numerous assignments are running and multiprogramming frameworks better handle these sort of utilizations

Disadvantages of multiprogramming systems

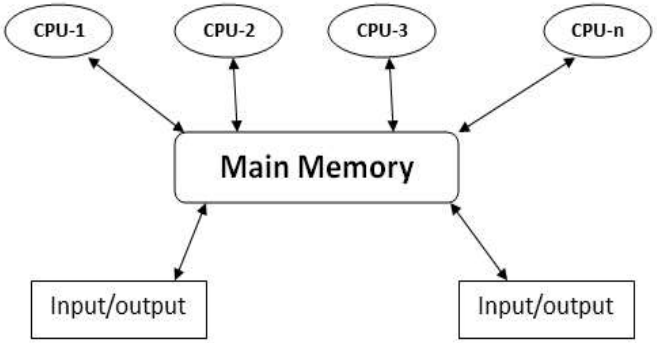
1) It is difficult to program a system because of complicated schedule handling
2) Tracking all tasks/processes is sometimes difficult to handle
3) Due to high load of tasks, long time jobs have to wait long



Multiprogramming Operating System

7) Multiprocessor Operating Systems

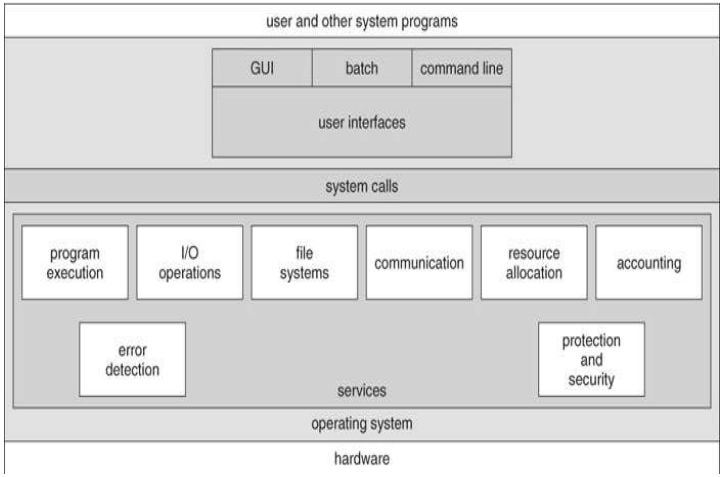
Multiprocessor operating systems are also known as parallel OS or tightly coupled OS. Such operating systems have more than one processor in close communication that sharing the computer bus, the clock and sometimes memory and peripheral devices. It executes multiple jobs at same time and makes the processing faster. Multiprocessor systems have three main advantages: ⇐ Increased throughput: By increasing the number of processors, the system performs more work in less time. The speed-up ratio with N processors is less than N. ⇐ Economy of scale: Multiprocessor systems can save more money than multiple single-processor systems, because they can share peripherals, mass storage, and power supplies. ⇐ Increased reliability: If one processor fails to done its task, then each of the remaining processors must pick up a share of the work of the failed processor. The failure of one processor will not halt the system, only slow it down. The ability to continue providing service proportional to the level of surviving hardware is called graceful degradation. Systems designed for graceful degradation are called fault tolerant.



Multiprocessor operating systems

1.3 Operating-System Structures

1.3.1 Operating System Services:



Operating-System Structures

Operating system services are:

1) Program execution
2) I/O Operation
3) Error Detection
4) Resource Allocation
5) File Management
6) Memory Management
7) Communication

1) Program execution:
The system must be able to load a program into memory and to run that program.

2) I/O Operation:
A running programs my required I/O. It allocates and interacts of various I/O devices when difficult programs are being executed.

3) Error Detection:
The operating system needs to be aware of possible error. Error may be occurring in the CPU, memory, hardware, I/O devices and in user programs. It is duty of operating system that appropriate error message whenever an error occurs and takes the suitable action for this error.

4) Resource Allocation:
When multiple users logged on the system or multiple jobs are running on the same time then resources must be allocated to each of them.

5) **File management:**
   This is service or function of the operating system to keeping the record of files on various storage devices and more all this files from one device to another.
6) **Memory Management**
   It includes the assignment of main memory and other storage areas to the system programs, user program and data.
7) **Communication:**

   The principle objective of the working frameworks is to give collaboration among client and equipment. One more arrangement of OS capacities exists for guaranteeing the productive activity of the actual framework by means of asset sharing

Bookkeeping - To monitor which clients use how a lot and what sorts of PC assets

Assurance and security - The proprietors of data put away in a multiuser or organized PC framework might need to control utilization of that data, simultaneous cycles ought not to meddle with one another

Assurance includes guaranteeing that all admittance to framework assets is controlled

Security of the framework from outcasts requires client confirmation, reaches out to protecting outer I/O gadgets from invalid access endeavours

On the off chance that a framework is to be ensured and secure, safety measures should be organized all through it. A chain is just pretty much as solid as its most fragile connection.

1.3.2 User and Operating system interface

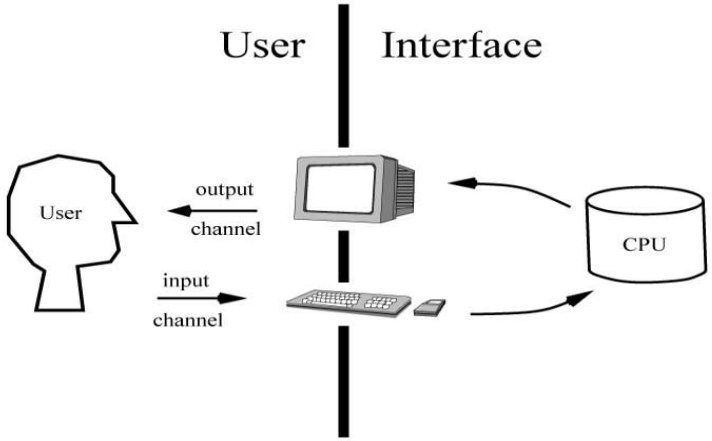
There are two fundamental approaches for users to interface with operating system.

1) Provide a command line interface (CLI) or command interpreter that allows the users directly enter commands that are to be performed by the operating system.
2) Allows the user to interface with operating system via a graphical user interface or GUI.
3) Some operating system includes the command interpreter in the kernel.
4) Others such as windows and UNIX treat the command interpreter as a special program (Like CMD in window and terminal in LINUX).
5) On the system with multiple command interpreters to choose from, the interpreter as known as shells.

Example:
1) Bourne Shell
2) C Shell
3) Bourne Again shell (BASH)
4) Korn shell etc.

So there are two approaches in which actually command interpreter execute the task.

1) Code for perform the certain task is included command interpreter itself.
2) Command interpreter does not contain any code itself but the code are written in certain programs.
   The choice of whether to use a command line or GUI interface is mostly one of the personal preference. System administrators who manages computers and power users who have deep knowledge of a system frequently use in command line interface.



User Operating System Interface

1.3.3 System Calls:

System Calls: - Programming interface to the services provided by the operating system (OS). Typically written in a high-level language (C or C++)

For the most part got to by programs by means of an undeniable level Application Program Interface (API) as opposed to coordinate framework call usenThree most normal APIs are Win32 API for Windows, POSIX API for POSIX-based frameworks (counting for all intents and purposes all adaptations of UNIX, Linux, and Mac OS X), and Java API for the Java virtual machine (JVM)

Why use APIs instead of framework calls?(Note that the framework call names utilized all through this text are conventional)

For performing any operation a user must have to request for a service call which is also known as the SYSTEM CALL or we can say

- Programming interface to the services provided by the operating system.
- They are typically written in a high level language (C or C++).
- There are two modes in the operation of system which user more or system mode.

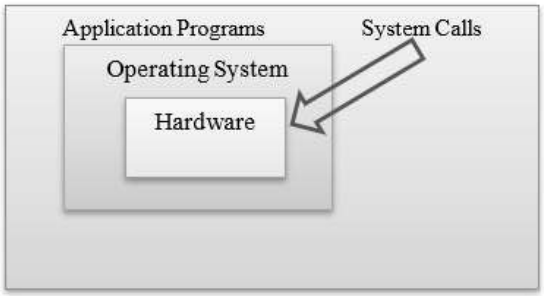
1) In user mode: All user processes are executed.
2) In system mode: All privileged operations are executed.

The user program and kernel functions are being executed in their respective space allotted in the main memory partitions.

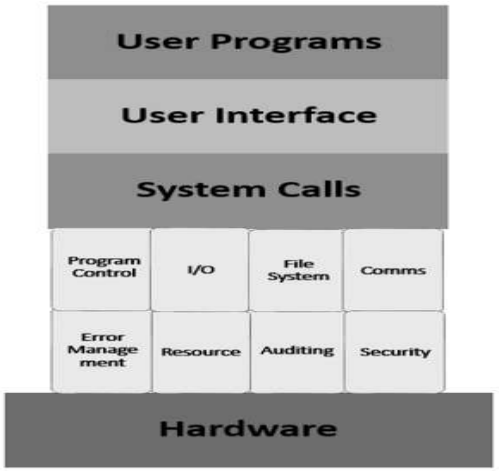
User mode programs need to execute some privileged operations, which are not permitted in user mode functions, user mode interface between user mode and kernel mode. This interface is called system calls. So system calls is an interface between the user programs and the operating system.

System calls expose all kernel functionalities that user mode program's require basically the system call is an instruction that request operating system to perform the desired operation that needs hardware access and other privileged operations.

System call generates an interrupt that causes the operating to gain control of the CPU. The operating system to find out the types of system call and corresponding interrupt handler routine is executed to perform the operation.



System calls are inherently used for security reasons. Due to the use of system calls user is not able to enter into operating system or any others user regions similarly I/O devices are also safe from any misuse by the user. Thus through the system calls kernel, other users programs and I/O devices are safe and secure malicious user programs.



Making a system calls:

Now systems calls are directly available and used in high level languages like C and C++. So it has become easy to use system calls in programs. For a C programmer, system calls are same as calling procedure or function. The difference between a system calls and normal function call is that system calls enters kernel but a normal function calls does not.

Executing the system calls:

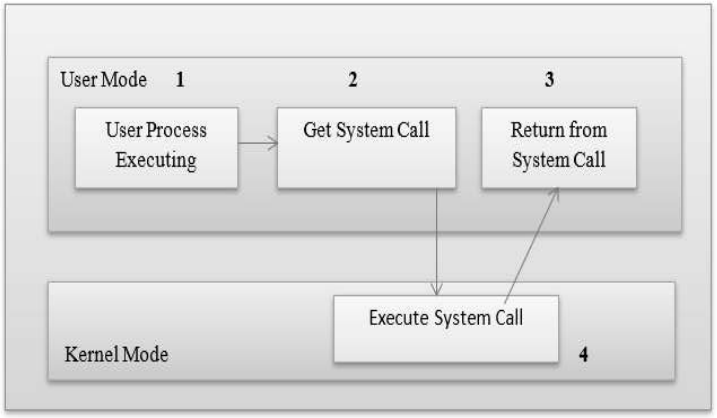
There is a sequence of steps to execute a system calls. For this, there is need to pass various parameters of system calls to operating system. For passing these parameters to operating system three methods are used as follows:

1) Register method where in the parameters are stored in the registers of the CPU.
2) If parameters are not in number, compared to size of registers, a block of memory is used and address of that block is stored in registers.
3) Stack method where in parameters are pushed onto the stack and popped off by the operating system.

Sequence in which system calls is executed:

1) In the user program when a system a call is executed, first of all its parameters are pushed onto the stack and later on saved in processor registers.
2) The corresponding library procedure for the system is executed.
3) There is a partition code for every system call by which kernel identifies which system call function or hander need to be executed. Therefore library procedure places the system call number in the procedure registers.

4) Then library procedure traps to the kernel by executing interrupt registers. With this interrupt execution, the user mode switches to kernel mode by loading program status word (PSW register to operating system.



1.3.4 Types of system calls:

1) Process control system calls

A process is a basic entity in the system. The processes in the system need to be created, deleted and aborted. Beside these many operations are required on the processes for their management.

These are some example:

- FORK ( ): Create a process
- EXIT ( ): Terminate a process
- KILL ( ): Terminate a process abnormally
- NICE ( ):Increase a priority of a process

2) File management system calls:

Creation, deletion, opening, closing, reading and writing are some general operation of files. Similarly for organizing files, there is a directory system and there by system calls managing them.

- CREATE ( ): To create a new file
- OPEN ( ): To open a file
- CLOSE ( ): To close a file
- READ ( ): To read a file
- WRITE ( ) : To write a file
- LINK( ):Give another name to a file
- UNLINK( ): Delete a file in a directory
- MKdir( ):Create a new directory

3) Device management system calls:

The general commands are:

- Request of device
- Release of device
- Read and Write operation and so on.

4) Information maintenance system calls:

Many system calls exits simply for the purpose of transferring information between the user program and the operating system.

- Return current date and time
- Number of current users
- Version number of operating system
- Amount of free memory

Get Time ( )	Set Date ( )	Get process ( )
Get Date ( )	Get system data ( )	Set process ( )
Set Time ( )	Set system data ( )	Get file ( ) and so on.

5) Commination's system calls

There is a need for commination among the processes in the system.

General operations are:

- Opening and closing the connection
- Sending and receiving messages
- Reading and writing messages and so on.

Msgsnd( ): Sanding a message

Msgrec( ): Receiving a message

These system calls may be related to communication between processes either on the same machine or between processes on different nodes of a network. Thus inter process communication is provided by the operating system through these communication related system calls.

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()

1.4 Processes:

1.4.1 Operating-System Structure

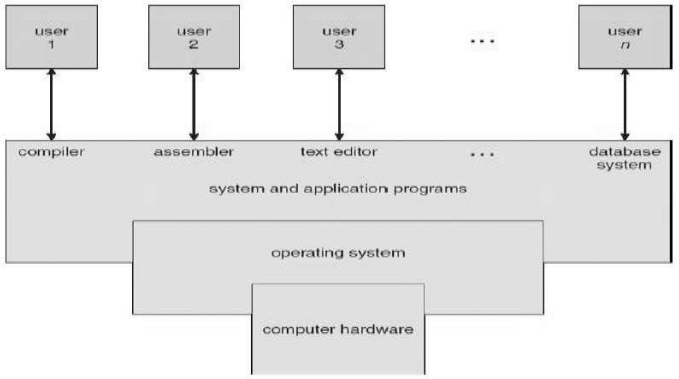
Computer system can be divided into four components

1) Hardware :- Provides the basic computing resources CPU, memory, Input / Output (I/O) devices

2) Operating system (OS) :- Controls and coordinates use of hardware among various applications & users

3) Application programs – define the ways in which the system resources are used to solve the computing problems of the users Word processors, compilers, web browsers, database systems, video games, etc.

4) Users People, machines, other computers the operating system can be implemented with the help of the various structures. The structure of the operating system (OS) depends mainly on how the various common components of the operating system are interconnected and melded into the kernel. Depending on this we have following structures of the operating system:

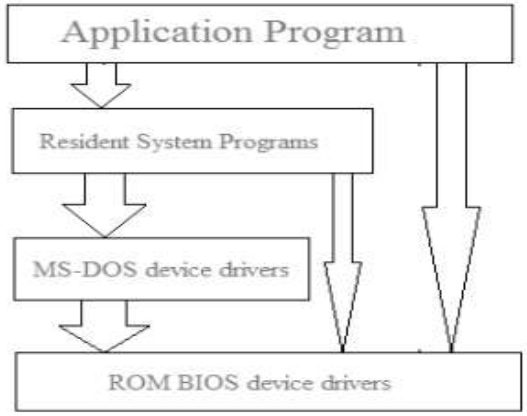


Operating-System Structure

Simple structure:

Such operating systems do not have well defined structure and are small, simple and limited systems. The interfaces and levels of functionality are not well separated. The MS-DOS is an example of such operating system. In the MS-DOS application programs are able to access the basic input/output (I/O) devices. These types of operating system cause the entire system to crash if one of the user programs fails.

Diagram of the structure of the MS-DOS is shown below.



Advantages of Simple structure:

- It delivers better application performance because of the few interfaces between the application program and the hardware.
- Easy for kernel developers to develop such an operating system.

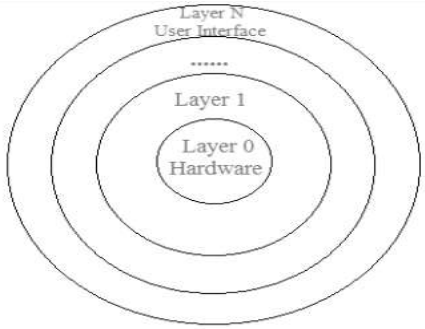
Disadvantages of Simple structure:

- The structure is very complicated as no clear boundaries exists between modules.
- It does not enforce data hiding in the operating system.

Layered structure:

An OS can be broken into pieces and retain much more control on system. In this structure the OS is broken into number of layers (levels). The bottom layer (layer 0) is the hardware and the topmost layer (layer N) is the user interface. These layers are so designed that each layer uses the functions of the lower level layers only. This simplifies the debugging process as if lower level layers are debugged and an error occurs during debugging then the error must be on that layer only as the lower level layers have already been debugged.

The main disadvantage of this structure is that at each layer, the data needs to be modified and passed on which adds overhead to the system. Moreover careful planning of the layers is necessary as a layer can use only lower level layers. UNIX is an example of this structure.



Advantages of Layered structure:

- Layering makes it easier to enhance the operating system as implementation of a layer can be changed easily without affecting the other layers.
- It is very easy to perform debugging and system verification.

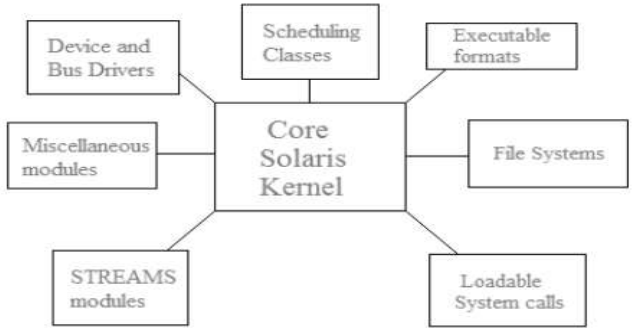
Disadvantages of Layered structure:

- In this structure the application performance is degraded as compared to simple structure.
- It requires careful planning for designing the layers as higher layers use the functionalities of only the lower layers.

Modular structure or approach:

It is considered as the best approach for an OS. It involves designing of a modular kernel. The kernel has only set of core components and other services are added as dynamically loadable modules to the kernel either during run time or boot time. It resembles layered structure due to the fact that each kernel has defined and protected interfaces but it is more flexible than the layered structure as a module can call any other module.

For example Solaris OS is organized as shown in the figure.



Open-Source Operating Systems

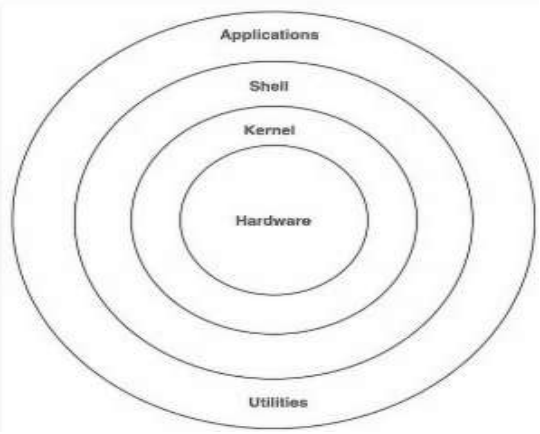
The first Open Source software is made available in 1997. Now there are Open Source alternatives for every Software application irrespective of the industry.

From the very beginning of the 21st-century, technical advancements and innovations lead to the creation of many Open Source Operating Systems. Here is everything you need to know about the Open Source Operating Systems. Open source refers to the computer software or applications where the owners or copyright holders allow the users or third party to see, use and provide the right to modify the source code of the product. An Open-source Operating System is the Operating System in which source code is visible publically and editable.

Types of Open Source Operating System

Most of the Open Source Operating Systems are Linux based.

1) **Linux Kernel** is created by **Linus Torvalds.** It provides the core functions needed for an Operating System like Parcelling of data, processing of memory, and interactions with the computer hardware. Linux is open-source many developers studied the source code and created many supportive plug-ins and operating systems for their needs. Though Linux is the heart of the operating systems, there are also some Open Source Operating Systems that are **not based on Linux.**



1) **Linux Distributions** are available and some of the popular Linux distributions are:
2) MX Linux
3) Manjaro
4) Linux Mint
5) elementary
6) Ubuntu
7) Debian
8) Solus
9) Fedora
10) openSUSE
11) Deepin

Advantages of Open Source Operating Systems:

1) **Cost-efficient –** Most of the Open Source OS is free. And some of them are available at a very cheap rate than the commercial closed products.

2) **Reliable and efficient –** Most of them are monitored by thousands of eyes since the source code is public. So if there is any vulnerability or bugs, they are fixed by the best developers around the world

3) **Flexibility-** The great advantage is you can customize it as per your need. And there is creative freedom.

Disadvantages of Open Source Operating Systems:

1) **Security risk –** Though the bugs are identified, there is a risk of attacks as the source code is available to the attackers.

2) **Complicated –** It is not user-friendly as the closed ones. You need to have the minimum technical knowledge to use this software

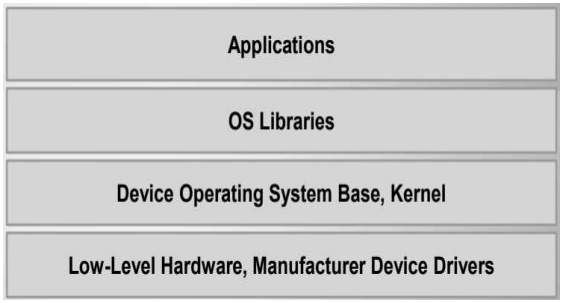
3) **No support –** If you meet with the problem, then there is no customer support to help you out.

1.4.2 Mobile OS (Operating System

Design and capabilities of a Mobile OS (Operating System) is very different than a general purpose OS running on desktop machines: – mobile devices have constraints and restrictions on their physical characteristic such as screen size, memory, processing power and etc. – Scarce availability of battery power – Limited amount of computing and communication capabilities.

Thus, they need different types of operating systems depending on the capabilities they support. e.g. a PDA OS is different from a Smartphone OS. • Operating System is a piece of software responsible for management of operations, control, coordinate the use of the hardware among the various application programs, and sharing the resources of a device.

A mobile OS is a software platform on top of which other programs called application programs, can run on mobile devices such as PDA, cellular phones, smartphone and etc.



There are many mobile operating systems. The followings demonstrate the most important ones: –

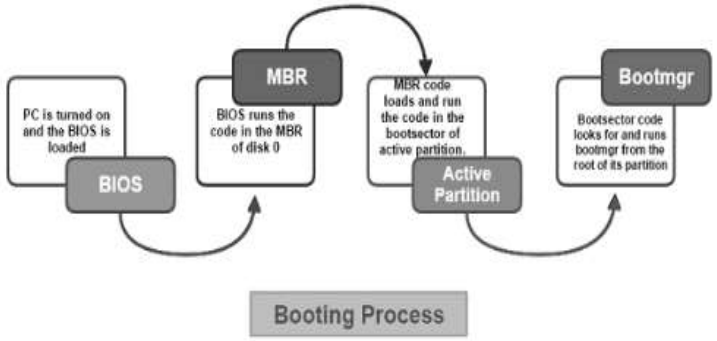
1) Java ME Platform –
2) Palm OS –
3) Symbian OS –
4) Linux OS –
5) Windows Mobile OS –
6) BlackBerry OS –
7) iPhone OS –
8) Google Android Platform

1.4.3 Booting Process of Operating System:

When you start the computer, it is observed that some initial text information is displayed on the screen. This is displayed by the firmware. The booting instructions are stored in ROM (read-only memory). Then the booting process starts. After booting, an operating system gets loaded in the main memory (RAM) of the computer.

Let us understand the complete booting process.

1) When you power on the computer, the CPU (central processing unit) activates the BIOS (basic input output system).
2) The first program activated is POST (power on selftest). Using the CMOS (complementary metal oxide semiconductor) memory it checks all the hardware and confirms that they are functioning properly.
3) After that it reads the MBR (master boot record) in boot drive in accordance with the firmware 'bootstrap loader' which is provided by the computer manufacturer.
4) Then the computer loads in the operating system in boot drive to the RAM.
5) Once this is performed, the operating system takes over the control of the computer and displays a user interface to the user.



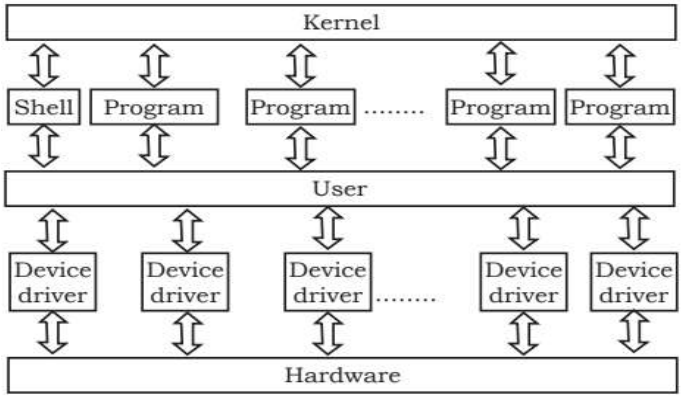
1.4.4 Components of Operating System

We identify the operating system by its user interface. The look or initial screen of various operating systems looks different, but architectural view of the various operating systems remains the same. There are essentially three components of operating system as described below:

1. The device driver

2. The kernel

3. The shell



Components of Operating System

1) The Device Driver:
This component is close to computer hardware. The device drivers are required for proper functioning of the devices attached to the computer system. These drivers can be installed or uninstalled as and when required. The kernel uses it for operating and controlling.

2) The Kernel:
It is the core of the operating system. It performs all the major functions of the operating system. It manages resources, controls program execution, and schedules program execution. It is the main operating system. It detects the new hardware when attached and installs the device driver for it to function properly.
3) The Shell:
We identify the operating system by how the shell looks. It provides the user interface to interact with the kernel and hardware. There are two types of user interface— command line interface (CLI) and graphical user interface (GUI) as explained in the Chapter earlier.

1.4.5 Processes:

Process Concept:

- An operating system executes a variety of programs:
- Batch system – jobs
- Time-shared systems – user programs or tasks
- Textbook uses the terms *job* and *process* almost interchangeably

Process – a program in execution; process execution must progress in sequential fashion

A process includes:

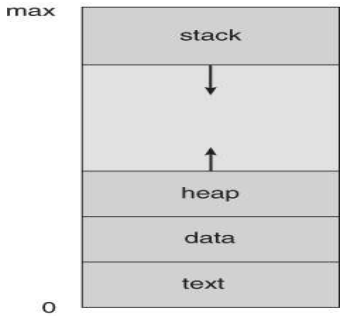
- program counter
- stack
- data section

Process: A process is an instance of a program in execution.

In batch operating systems work in terms of "jobs" used. In many advanced process methods are till expressed in terms of jobs, example is Job Scheduling.

The Process:

Process memory is mainly divided into four different categories as shown in figure.



A process in memory

- The text section comprises the complied program code and read from non-volatile storage when the actual program is launched.
- The data section stores static and global variables.
- The heap is used for dynamic memory allocation and is managed via calls to delete, new, malloc, free, etc.
- The stack is used for mainly in local variables. Space on the stack is reserved for local variables only when they are declared.
- When processes are swapped out of memory and later restored, additional information must also be stored and restored.

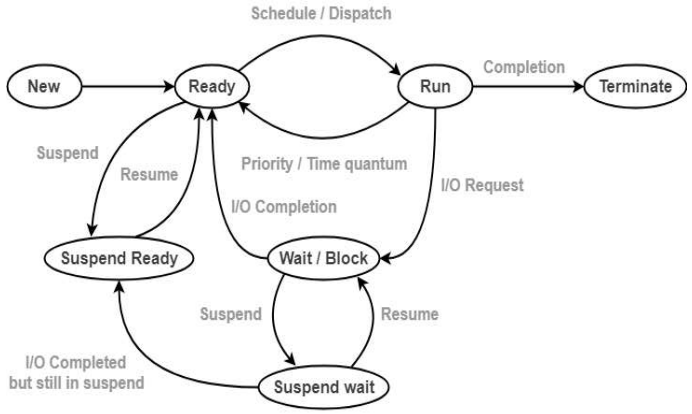
1.4.6 Process State:

Processes may be in one of SIX states as shown in figure below:

1) New State
2) Ready State
3) Running State
4) Waiting or Block State
5) Suspended Ready State

6) Terminated

- New: The process is in the stage of being created.
- Ready State: In first stage, a process moves from new state to ready state after it is loaded into the main memory and this process is ready for execution.
- Running state: A process moves from ready state to run state after it is assigned the CPU for each process for execution.
- Waiting or Block State: The process cannot run at the time because process is waiting for some I/O resources or some event to occur.
- Suspended Ready State: A process moves from ready state to suspend ready state if a process with higher priority has to be executed but the main memory is full.
- Terminated: the process has completed.



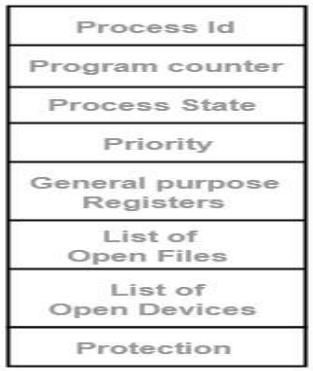
Process State Diagram

State	Present in Memory
New state	Secondary Memory
Ready state	Main Memory
Run state	Main Memory
Wait state	Main Memory
Suspend wait state	Secondary Memory
Suspend ready state	Secondary Memory
Terminate state	–

1.4.7 Process Control Block-

Process Control Block (PCB) is a data structure that stores all information about a particular each process. This all information about a particular each process is required by the CPU while executing the process time.

The Process Control Block diagram of a process looks like-



Process Control Block (PCB)

1. **Process Id**
   - Process Id is a unique Id for each process that identifies process of the system uniquely.
   - A process Id is assigned to each process during its creation of process.
2. **Program Counter**
   - Before process execution, program counter is initialized with the address of the first instruction of the program.
   - After executing a first instruction, value of program counter is automatically incremented to point to the next instruction of process.
   - This program counter process repeats till the end of the program.
3. **Process State**
   - Each process goes through different states (process state, running state, waiting state etc.) during its lifetime.
   - Process state specifies the current state of the process.

4. **Priority**
   - Process with the very highest priority is allocated the CPU first among all the other processes.
5. **General Purpose Registers**
   - General purpose registers are mainly used to hold the data of process generated during its execution time.
   - Each process has its own set of registers which are maintained by its process control block (PCB).
6. **List of Open Files**
   - Each process requires some important files which must be present in the main memory during its execution time.
   - Process control block (PCB) maintains a list of files used by the process during its execution.
7. **List of Open Devices-**
   - Process control block (PCB) maintains a list of open devices used by the each process during its execution time.

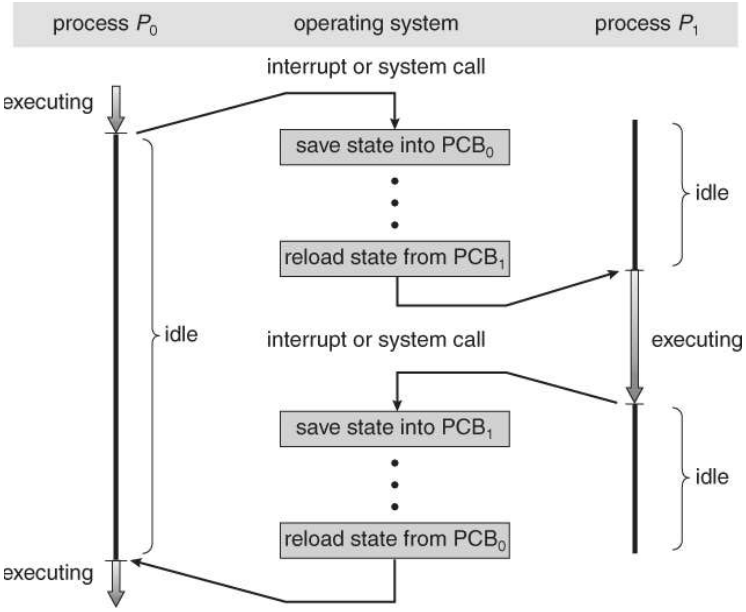


Diagram showing CPU switch from process to process

1.4.8 Process Scheduling:

- Schedulers in Operating System are special type of system software.
- They help in scheduling the processes in various ways.
- They are mainly responsible for selecting the jobs to be submitted into the system and deciding which process to run.

There are 3 kinds of schedulers-

1) Long-term scheduler
2) Short-term scheduler
3) Medium-term scheduler

1. Long-term Scheduler-

- Long-term scheduler is also called as Job Scheduler**.**
- In long term scheduler, it selects a balanced mix of I/O bound and CPU bound processes from the secondary memory such as new state.
- Then, it loads the selected processes into the main memory (ready state) for time of execution.

2. Short-term Scheduler-

- Short-term scheduler is also called as CPU Scheduler**.**
- It decides which individual process to execute next from the ready queue.
- After short-term scheduler decides the each process, Dispatcher assigns the each decided process to the CPU for execution purpose.

3. Medium-term Scheduler-

- Medium-term scheduler processes swaps-out from main memory to secondary memory to free up the main memory when it required.
- Thus, medium-term scheduler mainly reduces the degree of multiprogramming.
- After some time when main memory becomes available to use, medium-term scheduler swaps-in and swapped-out process to the main memory and its execution is continued from where it left off.
- Swapping technique may also be required to improve the process mix.

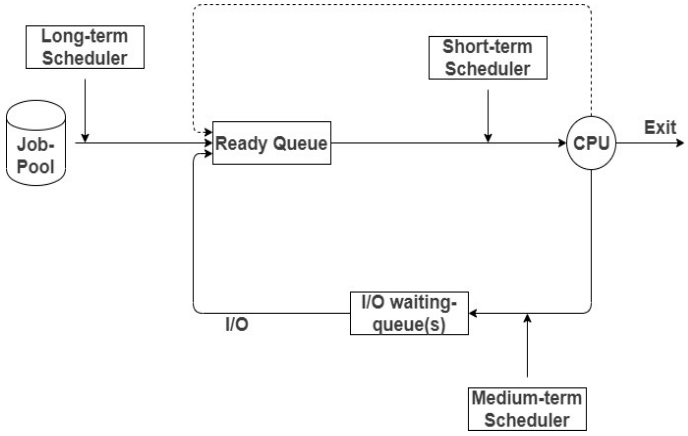


Diagram representation of process scheduling

Comparison of Schedulers

Long-term scheduler	Medium-term scheduler	Short-term scheduler
It is a job scheduler	It is a process swapping scheduler.	It is a CPU scheduler
It controls the degree of multiprogramming.	It reduces the degree of multiprogramming.	It provides lesser control over degree of multiprogramming.
Speed is lesser than short-term scheduler	Speed is in between the long-term and short-term schedulers.	Speed is fastest among the other two.
It is minimal or almost absent in time sharing system.	It is a part of time sharing system.	It is also minimal in time sharing system.
It selects processes from new state and loads them into ready state.	It swaps-out processes from main memory to secondary memory and later swaps in.	It selects processes from the ready state and assigns to the CPU.
Operates less frequently since processes are not rapidly created.	Operates more frequently than long-term scheduler but less frequently than short-term scheduler.	Operates very frequently to match the speed of CPU since CPU rapidly switches from one process to another

1.4.9 Operations on processes:

There are many types of operations that can be performed on processes. Some of these are process operation such as:

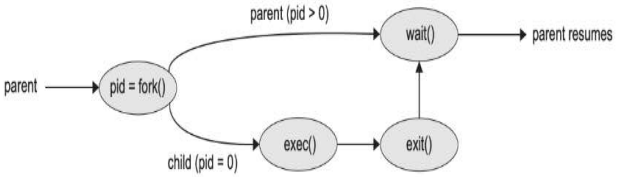
1) Creation
2) Process pre-emption
3) Process blocking
4) Process termination

1) Creation: Process Creation

Processes essential to be created in the system for different operations. This can be done by the following events −

- User request for new process creation
- Automatic system initialization
- Execution of a each process creation system call by a running process
- Batch job initialization

A process may be created by another new process using fork () system calls. The method creating process is called the parent process and the already created process is the child process. A child process can have only one parent but a parent process may have many children. Both the parent processes and child processes have the same memory image, open files, and environment strings.

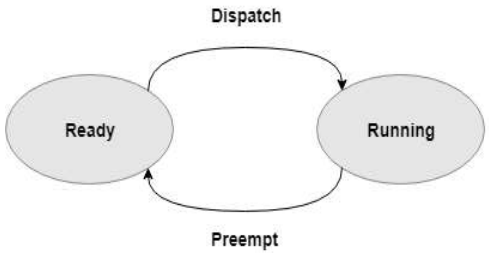


Process creation using FORK ( ) System Calls

2) Process Pre-emption

An interrupt instrument is used in process pre-emption that suspends the process executing presently and the next process to execute is determined by using a short-term scheduler. The main aims of process pre-emption is to makes sure that every processes get some CPU time for execution.

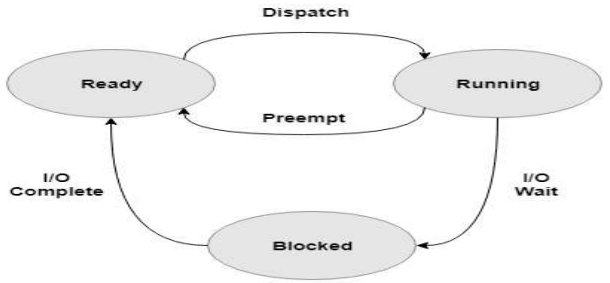
A diagram of process pre-emption is as follows:



3) Process Blocking

The process is blocked if it is process is waiting for some event to occur or process demanding is some I/O resources. This happening may be I/O as the I/O events are executed in the main memory and don't require the processor. After the event process is complete, the process again goes to the ready state.

A diagram that determines process blocking is as follows −

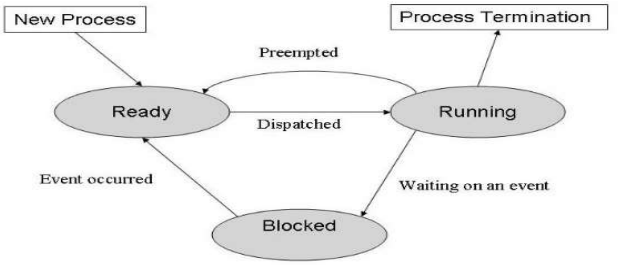


Process Blocking

4) Process Termination

After the process has successfully completed the execution task of its last instruction, it is terminated. The given resources held by a process are released after it is terminated.

A child process can be terminated by its parent process if its task is no longer used. The child process sends current status information to the parent process before it termination. Also, when a parent process is terminated, its child processes are terminated automatically as well as the child processes cannot run if the parent processes are already terminated.



Process Termination

1.4.10 Inter-process communication:

Working with multiple processes, require an inter process communication (IPC) method which will allow them to exchange data between multiple processes along with various useful information.

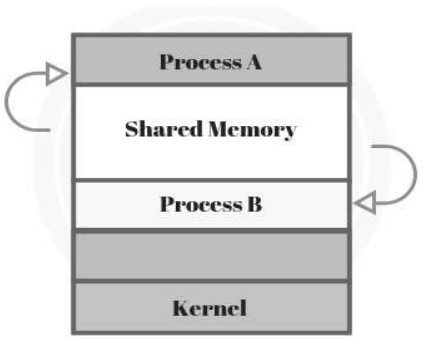
There are two primary models of inter-process communication:

1) Shared memory and
2) Message passing.

In the shared-memory model of inter-process communication, a region of memory which is shared by cooperating processes gets established. Processes can be then able to exchange all information by writing and reading all the data to the shared region. In the message-passing form, communication takes place by way of messages exchanged among the cooperating processes.

Shared Memory Process:

Inter-process communication (IPC) usually uses shared memory that requires communicating between different processes for establishing a region of shared memory. Typically, a shared-memory region resides within the address space of any process creating the shared memory segment. Other processes that want to communicate using this shared-memory segment must connect it to their address space.



Shared memory in operating system

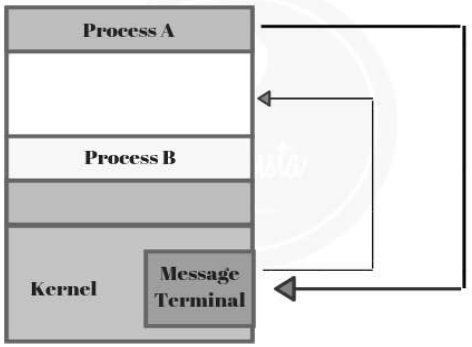
Message Passing Process:

Message passing systems are mainly support at a minimum system calls for purpose of send message and receive message.

A communication link must be developed between the cooperating processes before messages can be sent.

There are three key issues to be resolved in message passing process:

1) Direct or indirect communication
2) Synchronous or asynchronous communication
3) Automatic or explicit buffering



Message passing in operating system

1.5 Threads

4.1) Thread:

- To introduce the notion of a thread — a fundamental unit of CPU utilization that forms the basis of multithreaded computer systems
- To discuss the APIs for the Pthreads, Win32, and Java thread libraries
- To examine issues related to multithreaded programming

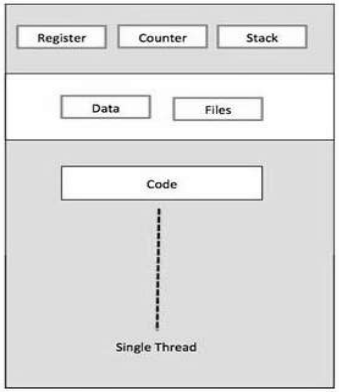
Benefits

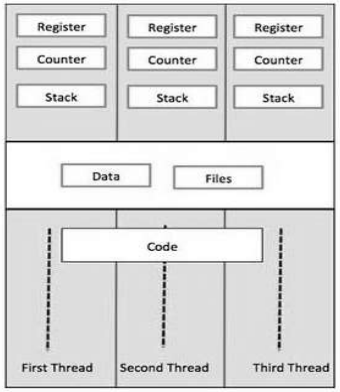
- Responsiveness
- Resource Sharing
- Economy
- Scalability

Thread is a part of execution unit that are mainly consists of its own program counter, a stack, and a set of registers where the program counter mainly keeps track of which instruction to execute next, a stack mainly contains the history of execution and a set of registers mainly hold its current working variables. Threads are also called as Lightweight processes method. Threads are a very popular way to improve the performance of an application through parallelism. Threads are mainly used to represent a software approach in order to increase the performance of an operating system just by reducing the overhead thread that is mainly equivalent to a classical process.

It is significant to note here that each thread goes to exactly one process and outside a process no threads exist. Each thread basically represents the flow of control separately. In the implementation of network servers and web servers threads have been successfully used. Threads provide a suitable foundation for the parallel execution of applications on shared-memory multiprocessors.

The given below figure shows the working of a single-threaded process and a multithreaded process:





0

Single process P with Single Thread Single Process P with three threads

Difference between Process VS Thread

S.N.	Process	Thread
1	Process is heavy weight or resource intensive.	Thread is light weight, taking lesser resources than a process.
2	Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system.
3	In multiple processing environments, each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
4	If one process is blocked, then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, a second thread in the same task can run.
5	Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.
6	In multiple processes each process operates independently of the others.	One thread can read, write or change another thread's data.

Advantages of Thread:

- Threads require minimize the context switching time.
- Use of threads provides concurrency within a process.
- Efficient communication between processes.
- It is more efficient to create and context switch threads.
- Threads allow using of multiprocessor architectures to a better scale and efficiency.

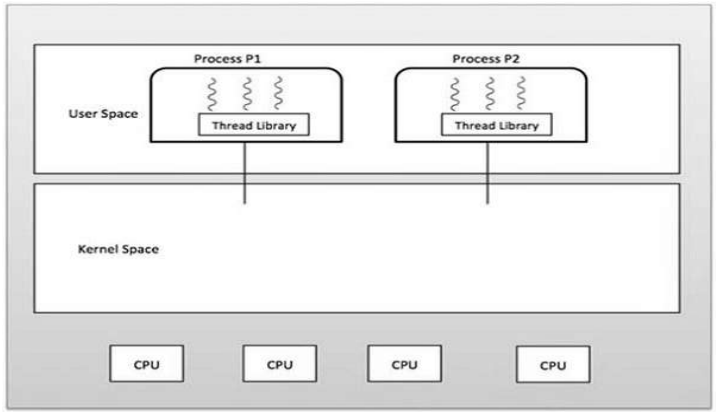
1.5.1 Types of Thread

Threads are implemented in following two ways −

1) User Level Threads
2) Kernel Level Threads

1) User Level Threads

In the thread, the thread executive's kernel is not aware of the occurrence of threads. The string library covers code for making and obliterating strings, for the passing message and information b/w strings, for planning string execution, and for saving and re-establishing string settings. The application begins with a solitary string.



Advantages

1) Thread switching does not need Kernel mode privileges.
2) User level thread can run on any type of operating system.
3) Scheduling can be application specific in the user level thread.
4) User level threads are fast to the create and manage very easily.

Disadvantages

1) In a typical operating system, most system calls are blocking.

2) Multithreaded application cannot take advantage of multiprocessing.

2) Kernel Level Threads

In this case, thread management is done by the Kernel only. There is no thread management code present in the application area. Kernel threads are supported directly by the operating system. Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process.

The Kernel maintains context type of information for the process as a whole and for individual's threads within the process. Scheduling by the Kernel is done on a thread basis. The Kernel is mainly performs three operations 1) thread creation, 2) scheduling and 3) management in Kernel space. Kernel threads are generally slower to create and manage than the user level threads.

Advantages

1) If one thread in a process is blocked, the Kernel can schedule another thread of the same process.

Disadvantages

1) Kernel threads are generally slower to create and manage than the user threads.

Compare User Level Thread VS Kernel Level Thread

S.N.	User-Level Threads	Kernel-Level Thread
1	User-level threads are faster to create and manage.	Kernel-level threads are slower to create and manage.
2	Implementation is by a thread library at the user level.	Operating system supports creation of Kernel threads.
3	User-level thread is generic and can run on any operating system.	Kernel-level thread is specific to the operating system.
4	Multi-threaded applications cannot take advantage of multiprocessing.	Kernel routines themselves can be multithreaded.

1.5.2 Multithreading Models

Multithreading models are three types

1) Many too many relationships.

2) Many to one relationship.

3) One to one relationship.

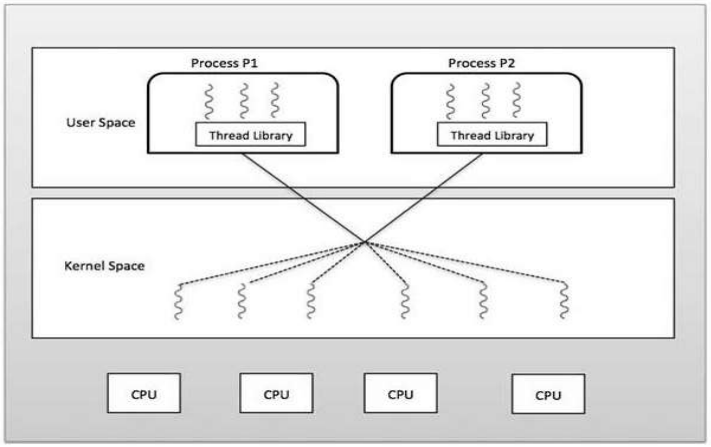
1) Many to Many Model

The many-to-many model multiplexes any number of the user threads onto an equal or smaller number of kernel threads.

The following diagram shows the many-to-many threading model where six user level threads are multiplexing with six kernel level threads. In this model, developers can create as many user threads as per necessary and the corresponding Kernel threads can run in parallel on a multiprocessor machine.

Example:

1) The Windows NT/2000



Many to Many Model

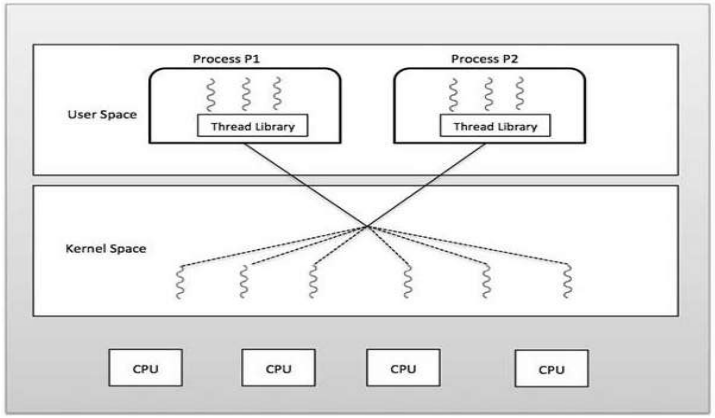
2) Many to One Model

When thread makes a blocking system call, the entire process will be blocked. When thread makes a blocking system call, the entire process will be blocked automatically. Only one thread can access the Kernel at a time, so multiple threads are unable to run in parallel on multiprocessors.

Examples:

1) Solaris Green Threads

2) GNU Portable Threads



Many to One Model

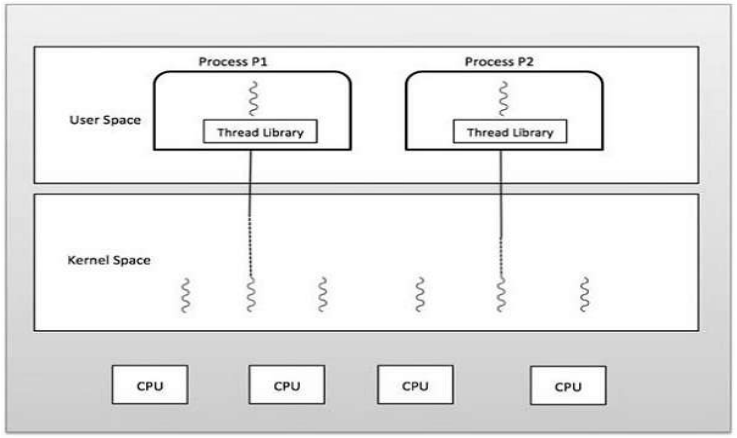
3) One to One Model

There is the one-to-one relationship of user-level thread to the kernel-level thread. This model provides more concurrency than the many-to-one model. It also allows another thread to run when a thread makes a blocking system call. It supports multiple threads to execute in parallel on microprocessors.

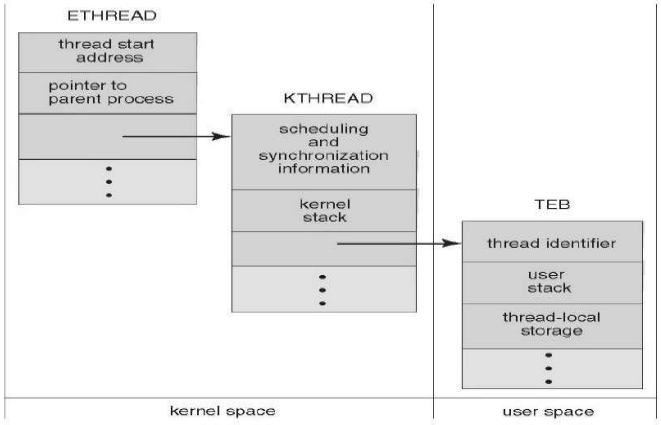
Examples

1) Windows NT/XP/2000

2) Linux



Windows XP Threads:



Implements the one-to-one mapping, kernel-level

Each thread contains

1) A thread id
2) Register set
3) Separate user and kernel stacks
4) Private data storage area
5) The register set, stacks, and private storage area are known as the context of the threads
6) The primary data structures of a thread include:
7) ETHREAD (executive thread block)
8) KTHREAD (kernel thread block)
9) TEB (thread environment block)

Linux Threads

flag	meaning
CLONE_FS	File-system information is shared.
CLONE_VM	The same memory space is shared.
CLONE_SIGHAND	Signal handlers are shared.
CLONE_FILES	The set of open files is shared.

1) Linux refers to them as *tasks* rather than *threads*
2) Thread creation is done through clone() system call
3) clone() allows a child task to share the address space of the parent task (process)

1.5.3 Multicore Programming:

Multicore programming benefits to the create concurrent systems for deployment on multicore processor as well as multiprocessor systems. A multicore processor system is basically a single processor with multiple execution cores in one chip. It has multiple numbers of processors on the motherboard or chip. A Field-Programmable Gate Array (FPGA) is power be included in a multiprocessor system. A FPGA is an integrated circuit containing an array of programmable logic blocks and a hierarchy of reconfigurable interconnects. Input data is processed by to produce outputs. It can be a processor in a multicore or multiprocessor system, or a FPGA.

The multicore programming approach has following advantages & minus;

1) The Multicore programming and FPGA processing helps to increase the performance of an embedded system.

2) To also help to complete scalability, so the system can consider the advantage of increasing the numbers of cores and FPGA processing power over time.

Concurrent systems that we create using multicore programming have various jobs to complete in parallel; this is known as concurrent execution. When multiple parallel tasks are executed by a processor, it is known as multitasking. A CPU scheduler handles the tasks that execute in parallel. The CPU implements tasks using operating system threads. So that tasks can execute autonomously but have some data transfer between them, such as data transfer between a data acquisition module and controller for the system. Data transfer occurs when there is a data dependency.

1.6 Conclusion:

In conclusion, an operating system is a software that manages computer hardware and software resources, and to provide public services for computer programs. The operating system is an important part of the system software in a computer system. ... In addition, there really is no such thing as a perfect operating system.

1.7 Summary:

In the introduction of operating system (OS), we saw that product can be generally separated into two gatherings: application programming and framework programming. Working frameworks are a sort of framework programming that permits applications to interface with PC equipment. Four significant classes of working frameworks are clump, timesharing, individualized computing, and committed. Assets are any items that can be assigned inside a framework, and the working framework is liable for overseeing them. A few assets, for example, essential memory can be space-multiplexed while different assets, for example, the CPU should be time-multiplexed

A cycle is an executing program. Since most PCs permit numerous cycles to execute at the same time, the working framework should deal with the request in which they execute. Three instances of interaction booking calculations are First Come First Serve, Round Robin, and Shortest Process Next.

1.8 EXERCISE:

1. Write a short note on operating system.

2. Explain various operating system structures.

3. Explain various operating system services.

4. Write a short note on Inter-process communication.

5. State and explain various multi-threading models.

1.9 References:

1. Operating Systems in Depth, Thomas W. Doeppner, Wiley.

2. Operating System Programming and Operating Systems, D M Dhamdhere, II nd Revised Edition, Tata McGraw.

3. Operating Systems, Achyut S. Godbole, 2nd edition, Tata McGraw Hill.

4. Application development using Android, Hello, Android, mobile development platform, Ed Burnette, 3rd Edition.

5. Linux Command Line & Shell Scripting, Richard Blum and Christine Bresnahan, 2nd edition, Wiley.

Text Books:

1. Modern Operating Systems, Tanenbaum, III rd Edition, PHI

2. Operating System-Internal & Design Principles, VI th Edition, William Stallings, Pearson

3. Operating Systems Concepts, Silberschatz A., Galvin P., Gagne G, VIII th Edition Wiley.

4. Principles of Operating Systems, Naresh Chauhan, First Edition , Oxford university press.

❄❄❄❄❄❄❄

2

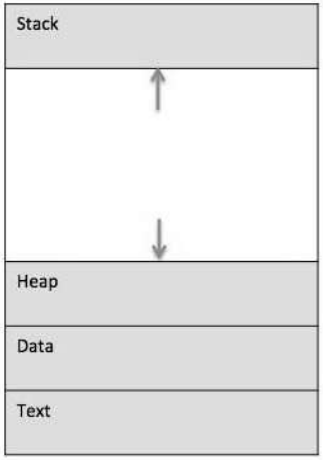
PROCESS SYNCHRONIZATION

Unit Structure

2.1 GENERAL STRUCTURE IF A TYPICAL PROCESS

Process is basically a program in execution. A process is defined as an entity which represents the basic unit of work to be implemented in the system

When a program is loaded into the memory and it becomes a process, it can be divided into four sections ─ stack, heap, text and data. The following image shows a simplified layout of a process inside main memory



Stack

The process Stack contains the temporary data such as method/function parameters, return address and local variables.

Heap

This is dynamically allocated memory to a process during its run time.

Heap

This is dynamically allocated memory to a process during its run time.

Data

This section contains the global and static variables.

2.2 RACE CONDITION

When more than one processes are executing the same code or accessing the same memory or any shared variable in that condition there is a possibility that the output or the value of the shared variable is wrong so for that all the processes doing the race to say that my output is correct this condition known as a race condition.

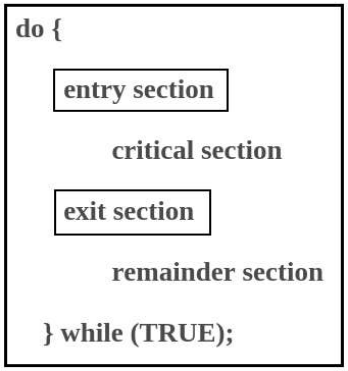
Several processes access and process the manipulations over the same data concurrently, then the outcome depends on the particular order in which the access takes place.

A race condition is a situation that may occur inside a critical section. This happens when the result of multiple thread execution in the critical section differs according to the order in which the threads execute.

Race conditions in critical sections can be avoided if the critical section is treated as an atomic instruction. Also, proper thread synchronization using locks or atomic variables can prevent race conditions.

2.3 CRITICAL SECTION PROBLEM

Critical section is a code segment that can be accessed by only one process at a time. Critical section contains shared variables which need to be synchronized to maintain consistency of data variables.



In the entry section, the process requests for entry in the **Critical Section.**

Any solution to the critical section problem must satisfy three requirements:

- **Mutual Exclusion** : If a process is executing in its critical section, then no other process is allowed to execute in the critical section.
- **Progress** : If no process is executing in the critical section and other processes are waiting outside the critical section, then only those processes that are not executing in their remainder section can participate in deciding which will enter in the critical section next, and the selection can not be postponed indefinitely.
- **Bounded Waiting** : A bound must exist on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.

2.1.4 PETERSON'S SOLUTION

Peterson's Solution is a classical software based solution to the critical section problem.

In Peterson's solution, we have two shared variables:

- boolean flag[i] :Initialized to FALSE, initially no one is interested in entering the critical section
- int turn : The process whose turn is to enter the critical section.

```
do {

    flag[i] = TRUE ;
    turn = j ;
    while (flag[j]  &&  turn == j) ;

        critial section

    flag[i] = FALSE ;

        remainder section

} while (TRUE) ;
```


Peterson's Solution preserves all three conditions :

- Mutual Exclusion is assured as only one process can access the critical section at any time.
- Progress is also assured, as a process outside the critical section does not block other processes from entering the critical section.
- Bounded Waiting is preserved as every process gets a fair chance.

Disadvantages of Peterson's Solution

- It involves Busy waiting
- It is limited to 2 processes.

2.1.5 SYNCHRONIZATION HARDWARE

TestAndSet
TestAndSet is a hardware solution to the synchronization problem. In TestAndSet, we have a shared lock variable which can take either of the two values, 0 or 1.

0 – Unlock

1 – Lock

Before entering into the critical section, a process inquires about the lock. If it is locked, it keeps on waiting until it becomes free and if it is not locked, it takes the lock and executes the critical section.

In TestAndSet, Mutual exclusion and progress are preserved but bounded waiting cannot be preserved.

2.1.6 MUTEX LOCKS

A **mutex** is a binary variable whose purpose is to provide locking mechanism. It is used to provide mutual exclusion to a section of code, means only one process can work on a particular code section at a time.

Mutex is a mutual exclusion object that synchronizes access to a resource. It is created with a unique name at the start of a program. The Mutex is a locking mechanism that makes sure only one thread can acquire the Mutex at a time and enter the critical section. This thread only releases the Mutex when it exits the critical section.

Eg :-

Wait (mutex);

… … …

Critical Section

… … …

Signal (mutex);

2.1.7 SEMAPHORE

A semaphore is a signalling mechanism and a thread that is waiting on a semaphore can be signaled by another thread. This is different than a mutex as the mutex can be signaled only by the thread that called the wait function.

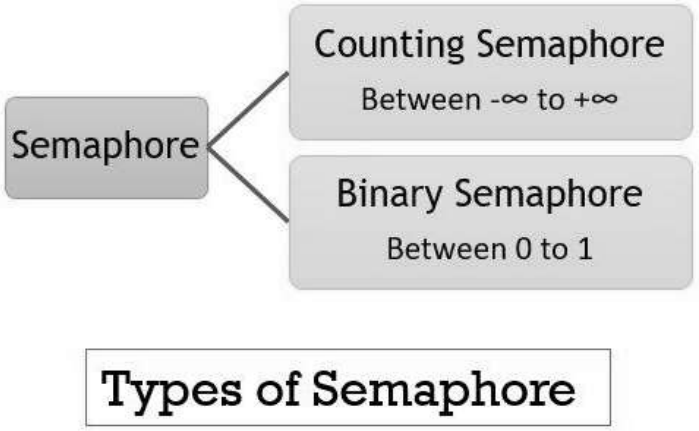
A semaphore uses two atomic operations, wait and signal for process synchronization.

The wait operation decrements the value of its argument S, if it is positive. If S is negative or zero, then no operation is performed.

```
wait(S)
{
  while (S<=0);
  S--;
}
```

The signal operation increments the value of its argument S.

```
signal(S)
{
  S++;
}
```



There are mainly two types of semaphores i.e.

1. counting semaphores
2. binary semaphores.

The Counting Semaphores are integer value semaphores and have an unrestricted value domain. These semaphores are used to coordinate the resource access, where the semaphore count is the number of available resources.

The Binary Semaphores are like counting semaphores but their value is restricted to 0 and 1. The wait operation only works when the semaphore is 1 and the signal operation succeeds when semaphore is 0.

CLASSIC SYNCHRONIZATION PROBLEMS

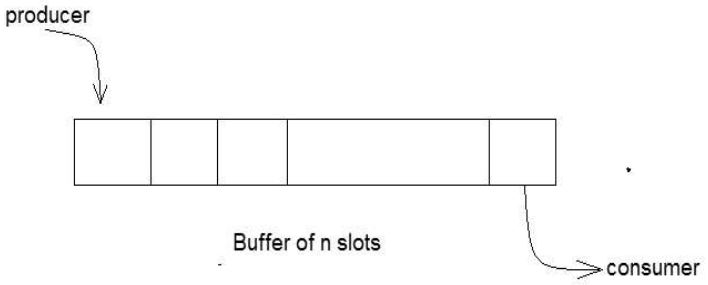
The classical problems of synchronization are as follows:

1. Bound-Buffer problem(producer-consumer problem)
2. Sleeping barber problem
3. Dining Philosophers problem
4. Readers and writers problem

Bounded Buffer Problem

Bound-Buffer problem

Also known as the **Producer-Consumer problem**. In this problem, there is a buffer of n slots, and each buffer is capable of storing one unit of data. There are two processes that are operating on the buffer – Producer and Consumer. The producer tries to insert data and the consumer tries to remove data.

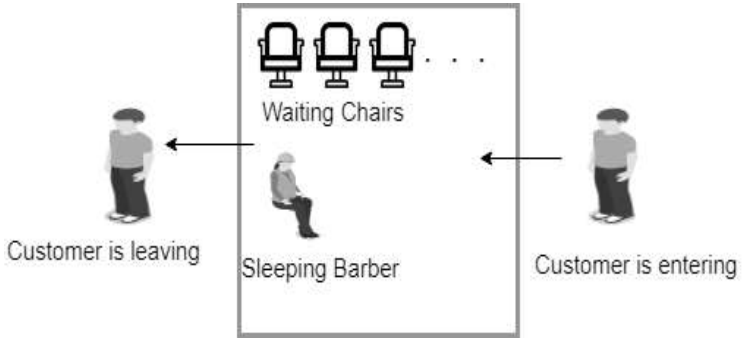


If the processes are run simultaneously, they will not yield the expected output.

The solution to this problem is creating two semaphores, one full and the other empty to keep a track of the concurrent processes.

Sleeping Barber Problem

This problem is based on a hypothetical barbershop with one barber.

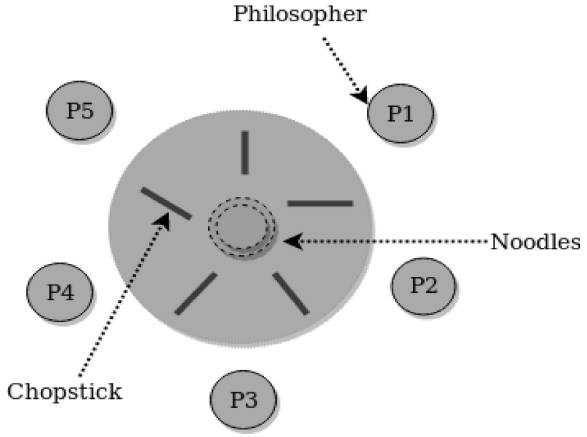


When there are no customers the barber sleeps in his chair. If any customer enters he will wake up the barber and sit in the customer chair. If there are no chairs empty they wait in the waiting queue.

Dining Philosopher's problem

This problem states that there are K number of philosophers sitting around a circular table with one chopstick placed between each pair of philosophers. The philosopher will be able to eat if he can pick up two chopsticks that are adjacent to the philosopher.

This problem deals with the allocation of limited resources.



Readers and Writers Problem

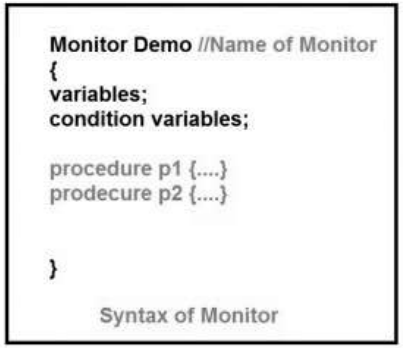
This problem occurs when many threads of execution try to access the same shared resources at a time. Some threads may read, and some may write. In this scenario, we may get faulty outputs.

2.1.8 Monitors in Process Synchronization

The monitor is one of the ways to achieve Process synchronization. The monitor is supported by programming languages to achieve mutual exclusion between processes. For example Java Synchronized methods. Java provides wait() and notify() constructs.

1. It is the collection of condition variables and procedures combined together in a special kind of module or a package.
2. The processes running outside the monitor can't access the internal variable of the monitor but can call procedures of the monitor.
3. Only one process at a time can execute code inside monitors.

Monitor syntax:



Condition Variables:

Two different operations are performed on the condition variables of the monitor.

1. **Wait**
2. **Signal**

Example for 2 condition variables x,y

condition x, y; // Declaring variable

Wait operation

x.wait() : Process performing wait operation on any condition variable are suspended. The suspended processes are placed in block queue of that condition variable.

Note: Each condition variable has its unique block queue.

Signal operation

x.signal(): When a process performs signal operation on condition variable, one of the blocked processes is given chance.

If (x block queue empty)

// Ignore signal

else

// Resume a process from block queue.

Advantages of Monitor:

Monitors have the advantage of making parallel programming easier and less error prone than using techniques such as semaphore.

Disadvantages of Monitor:

Monitors have to be implemented as part of the programming language . The compiler must generate code for them. This gives the compiler the additional burden of having to know what operating system facilities are available to control access to critical sections in concurrent processes. Some languages that do support monitors are Java,C#,Visual Basic,Ada and concurrent Euclid.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

❄❄❄❄❄❄❄

3

MAIN MEMORY

Unit Structure