User Mode

The system is in user mode when the operating system is running a user application such as handling a text editor. The transition from user mode to kernel mode occurs when the application requests the help of operating system or an interrupt or a system call occurs.

The mode bit is set to 1 in the user mode. It is changed from 1 to 0 when switching from user mode to kernel mode.

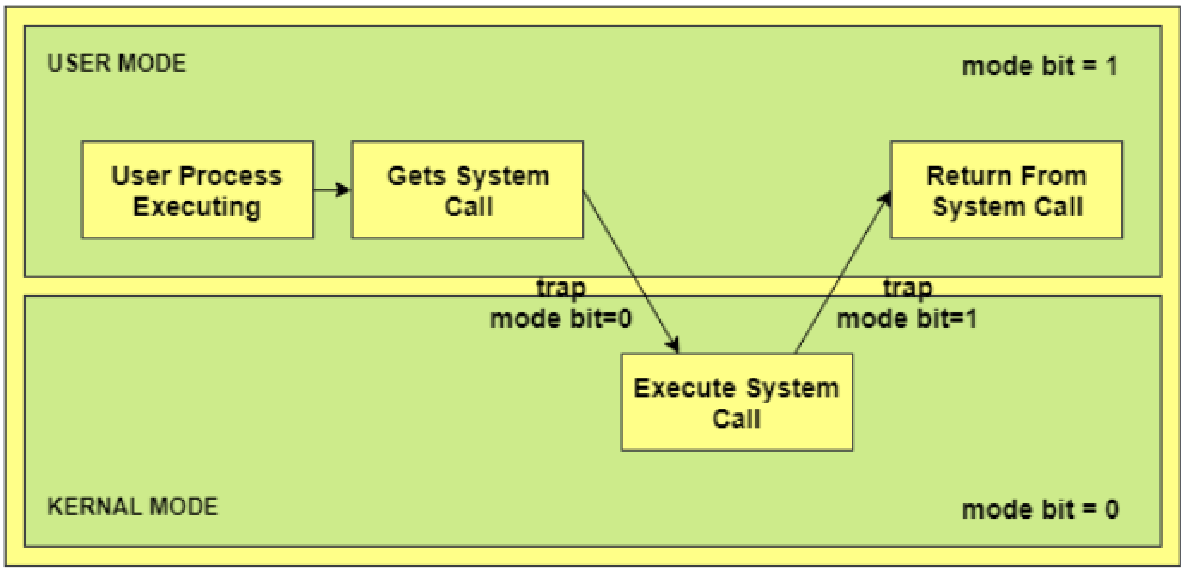
Kernel Mode

The system starts in kernel mode when it boots and after the operating system is loaded, it executes applications in user mode. There are some privileged instructions that can only be executed in kernel mode.

These are interrupt instructions, input output management etc. If the privileged instructions are executed in user mode, it is illegal and a trap is generated.

The mode bit is set to 0 in the kernel mode. It is changed from 0 to 1 when switching from kernel mode to user mode.

An image that illustrates the transition from user mode to kernel mode and back again is −



In the above image, the user process executes in the user mode until it gets a system call. Then a system trap is generated and the mode bit is set to zero. The system call gets executed in kernel mode. After the execution is completed, again a system trap is generated and the mode bit is set to 1. The system control returns to kernel mode and the process execution continues.

Necessity of Dual Mode (User Mode and Kernel Mode) in Operating System

The lack of a dual mode i.e user mode and kernel mode in an operating system can cause serious problems. Some of these are −

- A running user program can accidentaly wipe out the operating system by overwriting it with user data.
- Multiple processes can write in the same system at the same time, with disastrous results.

These problems could have occurred in the MS-DOS operating system which had no mode bit and so no dual mode.

Process switch

It is defined as that the processor switches from one thread/process to another thread or process. It makes the contents of the CPU registers and instruction pointer to be saved.

For the new task, the registers and instruction pointer are loaded into the processor then the execution of the new process may start/resume.

The old programs will not execute further, but the state of that process is saved in memory because when the kernel decides that it is ready to execute it again. This concept is like multitasking, but in reality, only a single process can run at a time on a CPU.

A context switch occurs by hardware or software. A hardware interrupt occurs from a device like a keyboard, mouse, or system timer, which causes code to begin executing the interrupt code. Software switches occur as a result of the kernels which are performing a task switch manually.

This is the way the scheduler uses a context switch.

Features −

- It effects the performance
- It increases load on CPU processor.
- Here every packet is inspected by router or switch processor.
- On every packet load balancing is performed.
- Easy to enable by one command.

Mode switch

The mode switch is used when the CPU changes privilege levels. The kernel works at a higher privilege than a standard user task.

In order to access user tasks that are controlled by the kernel, it is necessary for a mode switch to occur.

The currently executing process does NOT change during a mode switch. The processor uses the modes to protect the OS from misbehaving or malicious programs, as well as control concurrent access to RAM, I/O devices, etc.

A mode switch must occur for a software context switch to occur. Only the Kernel can cause a context switch.

Steps for Mode Switch −

- While Executing a program we have two modes user mode and kernel mode.

- So when a program is executed at the user level then there must be a user domain/mode.
- And when kernel mode is responsible for program execution then there is a kernel mode.
- Now, when a process is in badly need of a system resource then the mode switch occurs.
- It happens with the help of either a system call interface or by using interrupts.
- And after this kernel functions can be called from the user mode and manage the system calls.

Multiprogramming OS is an ability of an operating system that executes more than one program using a single processor machine.

More than one task or program or jobs are present inside the main memory at one point of time.

Buffering and spooling can overlap I/O and CPU tasks to improve the system performance but it has some limitations that a single user cannot always keep CPU or I/O busy all the time.

To increase resource utilization, multiprogramming approaches.

The OS could pick and start the execution of one of the jobs in memory, whenever the jobs does not need CPU that means the job is working with I/O at that time the CPU is idle at that time the OS switches to another job in memory and CPU executes a portion of it till the job issues a request for I/O and so on.

Let's P1 and P2 are two programs present in the main memory. The OS picks one program and starts executing it.

During execution if the P1 program requires I/O operation, then the OS will simply switch over to P2 program. If the p2 program requires I/O then again it switches to P3 and so on.

If there is no other program remaining after P3 then the CPU will pass its control back to the previous program.

Advantages

The advantages of multiprogramming operating system are as follows −

- CPU utilization is high because the CPU is never goes to idle state.
- Memory utilization is efficient.
- CPU throughput is high and also supports multiple interactive user terminals.

Disadvantages

The disadvantages of multiprogramming operating system are as follows −

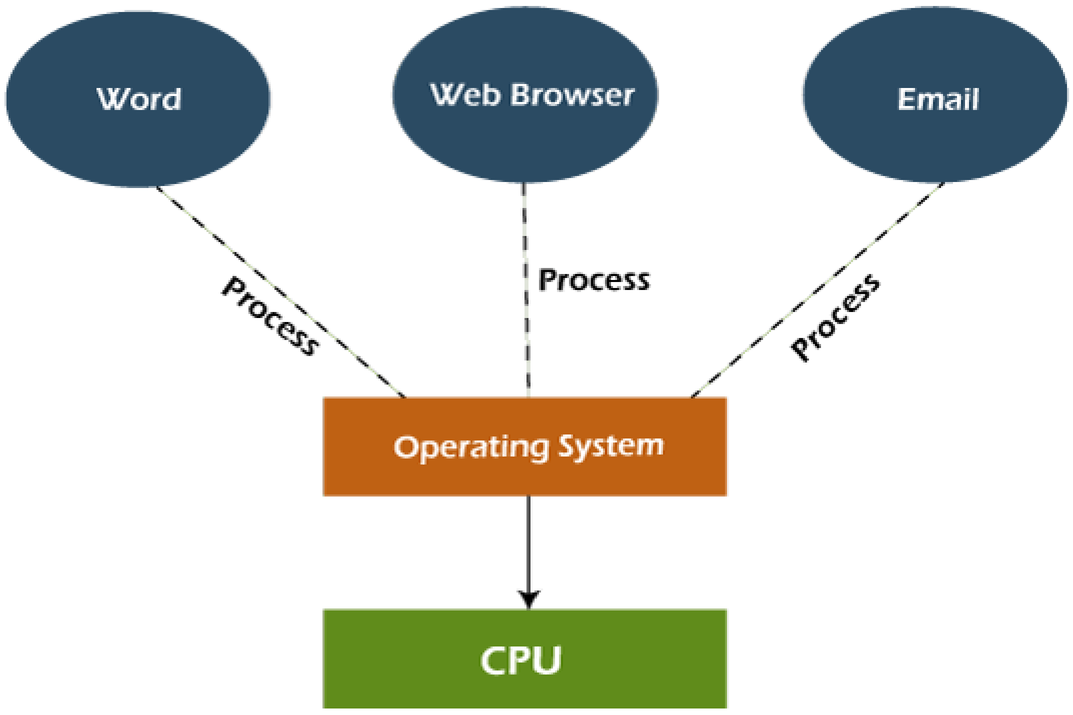
- CPU scheduling is compulsory because lots of jobs are ready to run on CPU simultaneously.
- User is not able to interact with jobs when it is executing.
- Programmers also cannot modify a program that is being executed.

If several jobs are ready in main memory and if there is not enough space for all of them, then the system has to choose them by making a decision, this process is called job scheduling.

When the operating system selects a job from the group of jobs and loads that job into memory for execution, therefore it needs memory management, if several such jobs are ready then it needs CPU scheduling.

Multitasking

Multitasking term used in a modern computer system. It is a logical extension of a multiprogramming system that enables the execution of **multiple** programs simultaneously. In an operating system, multitasking allows a user to perform more than one computer task simultaneously. Multiple tasks are also known as processes that share similar processing resources like a **CPU**. The operating system keeps track of where you are in each of these jobs and allows you to transition between them without losing data.



could execute various programs at the same time, although multitasking was not fully supported. As a result, a single software could consume the entire CPU of the computer while completing a certain activity. Basic operating system functions, such as file copying, prevented the user from completing other tasks, such as opening and closing windows. Fortunately, because modern operating systems have complete multitasking capability, numerous programs can run concurrently without interfering with one other. In addition, many operating system processes can run at the same time.

Types of Multitasking

There are mainly two types of multitasking. These are as follows:

1. **Preemptive Multitasking**
2. **Cooperative Multitasking**

Preemptive Multitasking

Preemptive multitasking is a special task assigned to a computer operating system. It decides how much time one task spends before assigning another task to use the operating system. Because the operating system controls the entire process, it is referred to as **'preemptive'**.

Preemptive multitasking is used in desktop operating systems. **Unix** was the first operating system to use this method of multitasking. **Windows NT** and **Windows 95** were the first versions of Windows that use preemptive multitasking. With **OS X**, the Macintosh acquired proactive multitasking. This operating system notifies programs when it's time for another program to take over the CPU.

Cooperative Multitasking

The term **'Non-Preemptive Multitasking'** refers to cooperative multitasking. The main purpose of cooperative multitasking is to run the present task while releasing the CPU

to allow another process to run. This task is carried out by using **taskYIELD ()**. When the **taskYIELD()** function is called, context-switch is executed.

Windows and MacOS used cooperative multitasking. A Windows program will respond to a message by performing some short unit of work before handing the CPU over to the operating system until the program receives another message. It worked perfectly as long as all programs were written with other programs in mind and bug-free.

Advantages and Disadvantages of Multitasking

Various advantages and disadvantages of multitasking are as follows:

Advantages

Various advantages of multitasking are as follows:

Manage Several Users

This operating system is more suited to supporting multiple users simultaneously, and multiple apps can run smoothly without interfering with system performance.

Virtual Memory

The greatest virtual memory system is found in multitasking operating systems. Because of virtual memory, any program does not require a long wait time to complete its tasks; if this problem arises, those programs are moved to virtual memory.

Good Reliability

Multitasking operating systems give more flexibility to several users, and they are happier as a result. On which each user can execute single or multiple programs simultaneously.

Secured Memory

The multitasking operating systems have well-defined memory management. Due to this operating system does not allow any types of permissions for undesirable programs to waste memory.

Time Shareable

All tasks are allotted a specified amount of time so that they do not have to wait for the CPU.

Background Processing

A multitasking operating system provides a better environment for background processes to run. These background programs are not visible to most users, but they help other programs like firewalls, antivirus software, and others run well.

Optimize the computer resources

A multitasking operating system may manage various computer resources like I/O devices.

- RAM
- Hard Disk
- CPU, and others.

Use Several Programs

Users can run many programs simultaneously, like an internet browser, games, MS Excel, PowerPoint, and other utilities.

Disadvantages

Processor Boundation

The system may run programs slowly because of the poor speed of their processors, and their reaction time might rise when processing many programs. To solve this problem, more processing power is required.

Memory Boundation

The computer's performance may get slow performance due to the multiple programs run at the same time because the main memory gets overloaded while loading multiple programs. Because the CPU is unable to provide different times for each program, reaction time increases. The primary cause of this issue is that it makes use of low-capacity RAM. As a result, the RAM.

CPU Heat Up

The multiple processors are busier at the same time to complete any task in a multitasking environment, so the CPU generates more heat.

Process

A process is basically a program in execution. The execution of a process must progress in a sequential fashion.

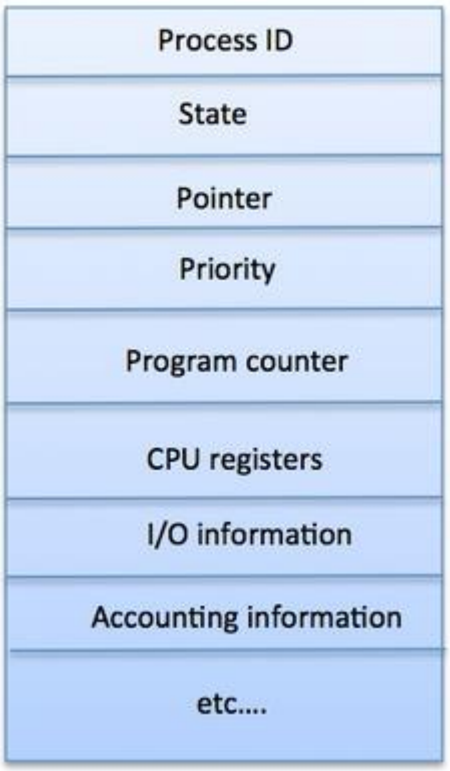
Process Control Block (PCB)

A Process Control Block is a data structure maintained by the Operating System for every process. The PCB is identified by an integer process ID (PID). A PCB keeps all the information needed to keep track of a process as listed below in the table −

S.N.	Information & Description
1	**Process State** The current state of the process i.e., whether it is ready, running, waiting, or whatever.
2	**Process privileges** This is required to allow/disallow access to system resources.

3	**Process ID** Unique identification for each of the process in the operating system.
4	**Pointer** A pointer to parent process.
5	**Program Counter** Program Counter is a pointer to the address of the next instruction to be executed for this process.
6	**CPU registers** Various CPU registers where process need to be stored for execution for running state.
7	**CPU Scheduling Information** Process priority and other scheduling information which is required to schedule the process.
8	**Memory management information** This includes the information of page table, memory limits, Segment table depending on memory used by the operating system.
9	**Accounting information** This includes the amount of CPU used for process execution, time limits, execution ID etc.
10	**IO status information** This includes a list of I/O devices allocated to the process.

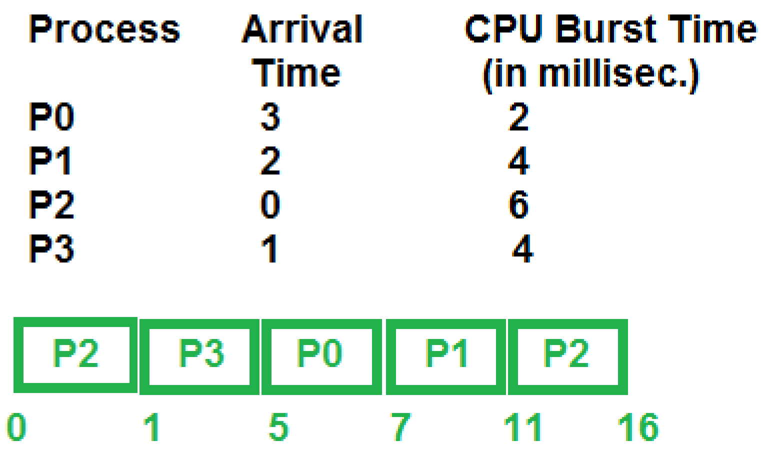
The architecture of a PCB is completely dependent on Operating System and may contain different information in different operating systems. Here is a simplified diagram of a PCB –



The PCB is maintained for a process throughout its lifetime, and is deleted once the process terminates.

1.Preemptive

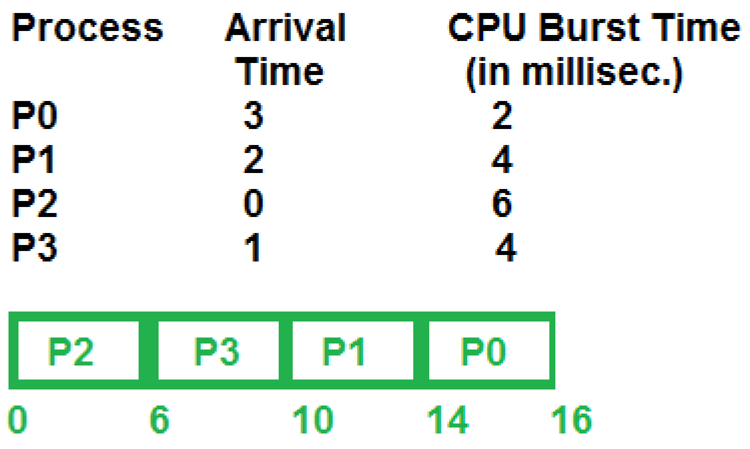
Scheduling:
Preemptive scheduling is used when a process switches from running state to ready state or from the waiting state to ready state. The resources (mainly CPU cycles) are allocated to the process for a limited amount of time and then taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining. That process stays in the ready queue till it gets its next chance to execute.



Process	Arrival Time	CPU Burst Time (in millisec.)
P0	3	2
P1	2	4
P2	0	6
P3	1	4

Preemptive Scheduling

2.Non-Preemptive Scheduling:
Non-preemptive Scheduling is used when a process terminates, or a process switches from running to the waiting state. In this scheduling, once the resources (CPU cycles) are allocated to a process, the process holds the CPU till it gets terminated or reaches a waiting state. In the case of non-preemptive scheduling does not interrupt a process running CPU in the middle of the execution. Instead, it waits till the process completes its CPU burst time, and then it can allocate the CPU to another process.



Process	Arrival Time	CPU Burst Time (in millisec.)
P0	3	2
P1	2	4
P2	0	6
P3	1	4

Non-Preemtive Scheduling

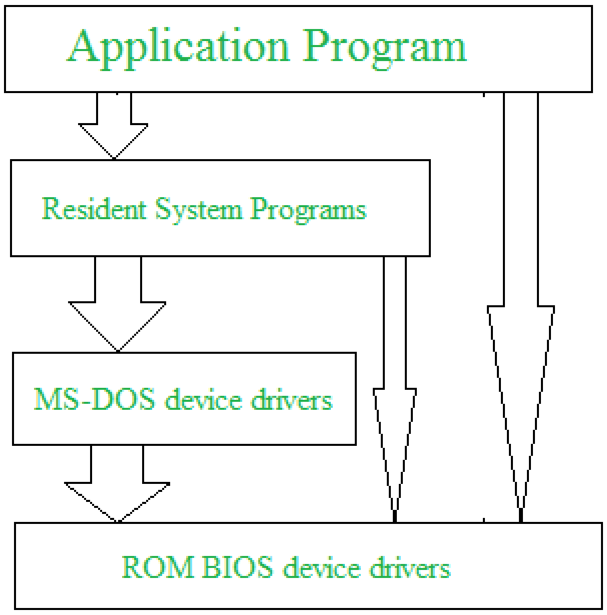
Key Differences Between Preemptive and Non-Preemptive Scheduling:

1. In preemptive scheduling, the CPU is allocated to the processes for a limited time whereas, in Non-preemptive scheduling, the CPU is allocated to the process till it terminates or switches to the waiting state.
2. The executing process in preemptive scheduling is interrupted in the middle of execution when higher priority one comes whereas, the executing process in non-preemptive scheduling is not interrupted in the middle of execution and waits till its execution.
3. In Preemptive Scheduling, there is the overhead of switching the process from the ready state to running state, vise-verse and maintaining the ready queue. Whereas in the case of non-preemptive scheduling has no overhead of switching the process from running state to ready state.
4. In preemptive scheduling, if a high-priority process frequently arrives in the ready queue then the process with low priority has to wait for a long, and it may have to starve. , in the non-preemptive scheduling, if CPU is allocated to the process having a larger burst time then the processes with small burst time may have to starve.
5. Preemptive scheduling attains flexibility by allowing the critical processes to access the CPU as they arrive into the ready queue, no matter what process is executing currently. Non-preemptive scheduling is called rigid as even if a critical process enters the ready queue the process running CPU is not disturbed.
6. Preemptive Scheduling has to maintain the integrity of shared data that’s why it is cost associative which is not the case with Non-preemptive Scheduling.

Parameter	PREEMPTIVE SCHEDULING	NON-PREEMPTIVE SCHEDULING
Basic	In this resources(CPU Cycle) are allocated to a process for a limited time.	Once resources(CPU Cycle) are allocated to a process, the process holds it till it completes its burst time or switches to waiting state.
Interrupt	Process can be interrupted in between.	Process can not be interrupted until it terminates itself or its time is up.
Starvation	If a process having high priority frequently arrives in the ready queue, a low priority process may starve.	If a process with a long burst time is running CPU, then later coming process with less CPU burst time may starve.
Overhead	It has overheads of scheduling the processes.	It does not have overheads.
Flexibility	flexible	rigid

Parameter	PREEMPTIVE SCHEDULING	NON-PREEMPTIVE SCHEDULING
Cost	cost associated	no cost associated
CPU Utilization	In preemptive scheduling, CPU utilization is high.	It is low in non preemptive scheduling.
Waiting Time	Preemptive scheduling waiting time is less.	Non-preemptive scheduling waiting time is high.
Response Time	Preemptive scheduling response time is less.	Non-preemptive scheduling response time is high.
Examples	Examples of preemptive scheduling are Round Robin and Shortest Remaining Time First.	Examples of non-preemptive scheduling are First Come First Serve and Shortest Job First.

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. MS-DOS is an example of such operating system. In MS-DOS application programs are able to access the basic I/O routines. These types of operating system cause the entire system to crash if one of the user programs fails.
Diagram of the structure of MS-DOS is shown below.



Advantages of Simple structure:

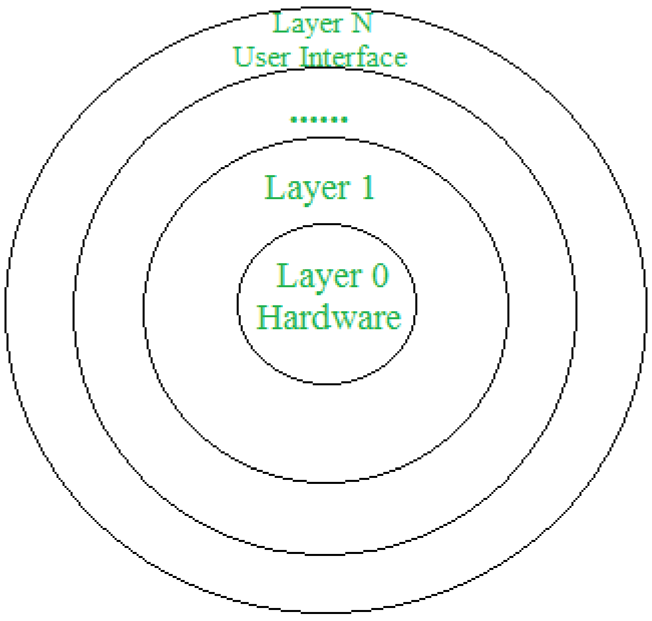
- It delivers better application performance because of the few interfaces between the application program and the hardware.
- Easy for kernel developers to develop such an operating system.

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.

Micro-kernel:
This structure designs the operating system by removing all non-essential components from the kernel and implementing them as system and user programs. This result in a smaller kernel called the micro-kernel.
Advantages of this structure are that all new services need to be added to user space and does not require the kernel to be modified. Thus it is more secure and reliable as if a service fails then rest of the operating system remains untouched. Mac OS is an example of this type of OS.

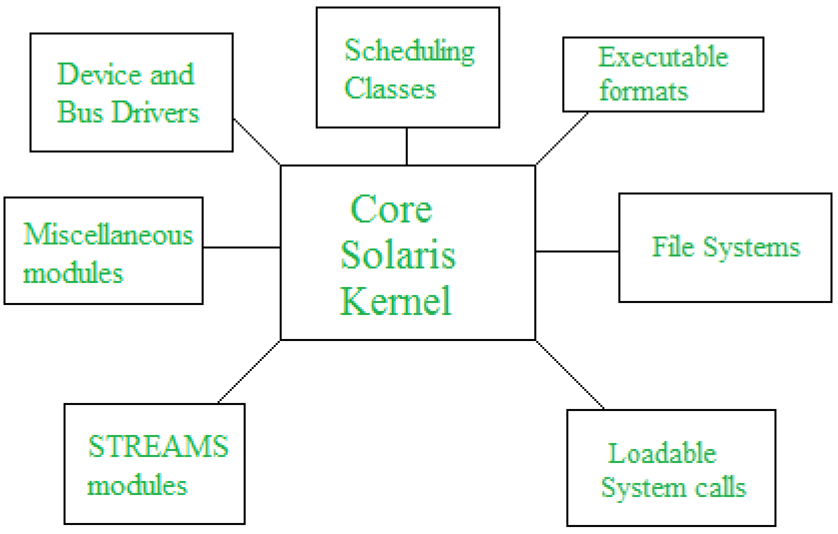
Advantages of Micro-kernel structure:

- It makes the operating system portable to various platforms.
- As microkernels are small so these can be tested effectively.

Disadvantages of Micro-kernel structure:

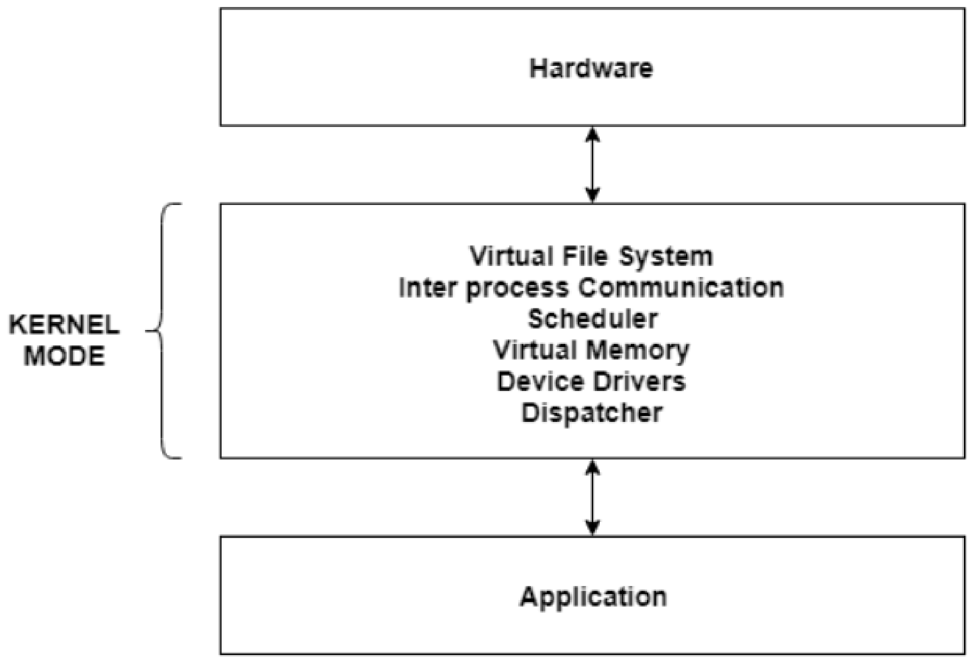
- Increased level of inter module communication degrades system performance.

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.



The entire operating system works in the kernel space in the monolithic system. This increases the size of the kernel as well as the operating system. This is different than the microkernel system where the minimum software that is required to correctly implement an operating system is kept in the kernel.

A diagram that demonstrates the architecture of a monolithic system is as follows −



Monolithic Kernel Operating System

The kernel provides various services such as memory management, file management, process scheduling etc. using function calls. This makes the execution of the operating system quite fast as the services are implemented under the same address space.

Differences Between Microkernel and Monolithic Kernel

Some of the differences between microkernel and monolithic kernel are given as follows −

- The microkernel is much smaller in size as compared to the monolithic kernel.
- The microkernel is easily extensible whereas this is quite complicated for the monolithic kernel.
- The execution of the microkernel is slower as compared to the monolithic kernel.
- Much more code is required to write a microkernel than the monolithic kernel.
- Examples of Microkernel are QNX, Symbian, L4 Linux etc. Monolithic Kernel examples areLinux, BSD etc.

Advantages of Monolithic Kernel

Some of the advantages of monolithic kernel are −

- The execution of the monolithic kernel is quite fast as the services such as memory management, file management, process scheduling etc.are implemented under the same address space.
- A process runs completely in a single address space in the monolithic kernel.
- The monolithic kernel is a static single binary file.

Disadvantages of Monolithic Kernel

Some of the disadvantages of monolithic kernel are −

- If any service fails in the monolithic kernel, it leads to the failure of the entire system.
- To add any new service, the entire operating system needs to be modified by the user.

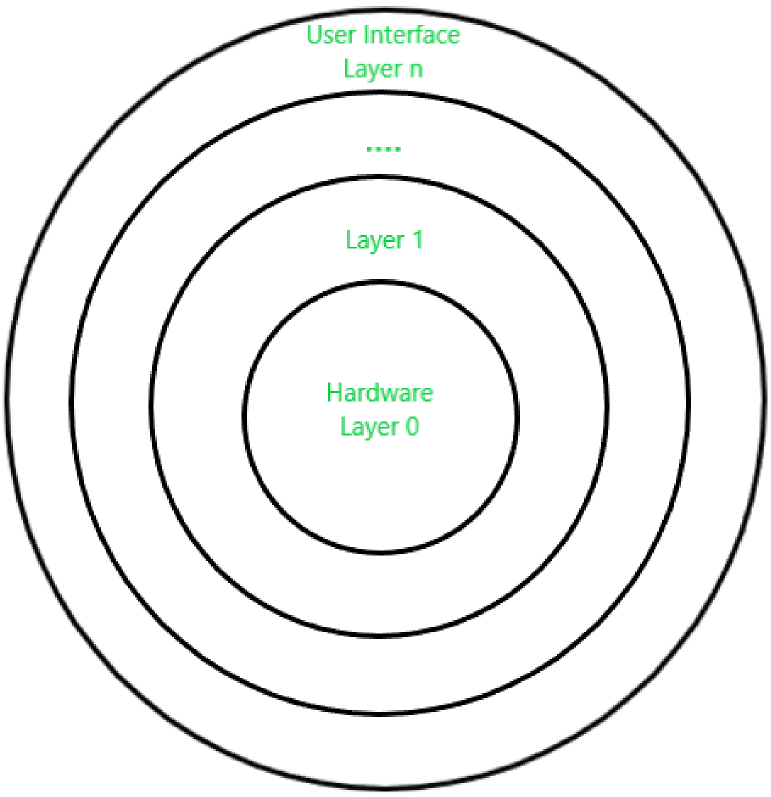
Layered Structure is a type of system structure in which the different services of the OS are split into various layers, where each layer has a specific well-defined task to perform. It was created to improve the pre-existing structures like the Monolithic structure (UNIX) and the Simple structure (MS-DOS).

Example – The Windows NT operating system uses this layered approach as a part of it.

Design Analysis :
The whole Operating System is separated into several layers (from 0 to n) as the diagram shows. Each of the layers must have its own specific function to perform. There are some rules in the implementation of the layers as follows.

1. The outermost layer must be the User Interface layer.
2. The innermost layer must be the Hardware layer.
3. A particular layer can access all the layers present below it but it cannot access the layers present above it. That is layer n-1 can access all the layers from n-2 to 0 but it cannot access the nth layer.

Thus if the user layer wants to interact with the hardware layer, the response will be traveled through all the layers from n-1 to 1. Each layer must be designed and implemented such that it will need only the services provided by the layers below it.



Layered OS Design

Advantages :
There are several advantages to this design :

1. **Modularity :**
   This design promotes modularity as each layer performs only the tasks it is scheduled to perform.
2. **Easy debugging :**
   As the layers are discrete so it is very easy to debug. Suppose an error occurs in the CPU scheduling layer, so the developer can only search that particular layer to debug, unlike the Monolithic system in which all the services are present together.
3. **Easy update :**
   A modification made in a particular layer will not affect the other layers.
4. **No direct access to hardware :**
   The hardware layer is the innermost layer present in the design. So a user can use the services of hardware but cannot directly modify or access it, unlike the Simple system in which the user had direct access to the hardware.

5. **Abstraction :**
   Every layer is concerned with its own functions. So the functions and implementations of the other layers are abstract to it.

Disadvantages :
Though this system has several advantages over the Monolithic and Simple design, there are also some disadvantages as follows.

1. **Complex and careful implementation :**
   As a layer can access the services of the layers below it, so the arrangement of the layers must be done carefully. For example, the backing storage layer uses the services of the memory management layer. So it must be kept below the memory management layer. Thus with great modularity comes complex implementation.

2. **Slower in execution :**
   If a layer wants to interact with another layer, it sends a request that has to travel through all the layers present in between the two interacting layers. Thus it increases response time, unlike the Monolithic system which is faster than this. Thus an increase in the number of layers may lead to a very inefficient design.

Multilevel Feedback Queue Scheduling (MLFQ) CPU Scheduling is like Multilevel Queue(MLQ) Scheduling but in this processes can move between the queues. And thus, much more efficient than multilevel queue scheduling.

Characteristics of Multilevel Feedback Queue Scheduling:

- In a MQS algorithm, processes are permanently assigned to a queue on entry to the system and processes are not allowed to move between queues.
- As the processes are permanently assigned to the queue, this setup has the advantage of low scheduling overhead,
- But on the other hand disadvantage of being inflexible.

Advantages of Multilevel Feedback Queue Scheduling:

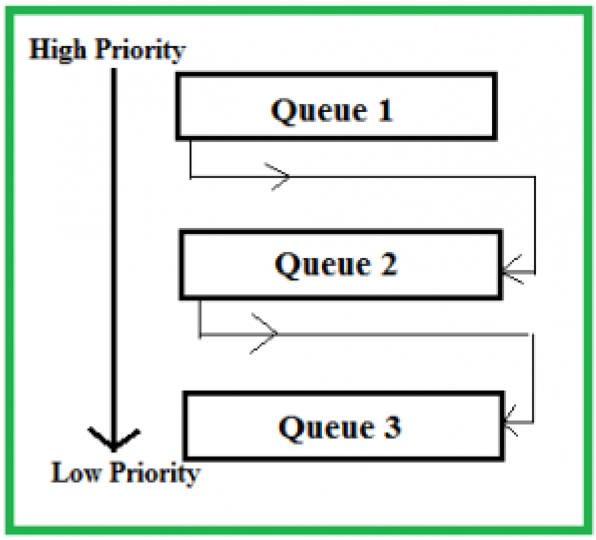
- It is more flexible.
- It allows different processes to move between different queues.
- It prevents starvation by moving a process that waits too long for the lower priority queue to the higher priority queue.

Disadvantages of Multilevel Feedback Queue Scheduling:

- For the selection of the best scheduler, it requires some other means to select the values.
- It produces more CPU overheads.
- It is the most complex algorithm.

Multilevel feedback queue scheduling, however, allows a process to move between queues. Multilevel Feedback Queue Scheduling **(MLFQ)** keeps analyzing the behavior (time of execution) of processes and according to which it changes its priority.

Now, look at the diagram and explanation below to understand it properly.



Now let us suppose that queues 1 and 2 follow round robin with time quantum 4 and 8 respectively and queue 3 follow FCFS.

Implementation of MFQS is given below –

- When a process starts executing the operating system can insert it into any of the above three queues depending upon its **priority**. For example, if it is some background process, then the operating system would not like it to be given to higher priority queues such as queue 1 and 2. It will directly assign it to a lower priority queue i.e. queue 3. Let's say our current process for consideration is of significant priority so it will be given **queue 1**.
- In queue 1 process executes for 4 units and if it completes in these 4 units or it gives CPU for I/O operation in these 4 units then the priority of this process does not change and if it again comes in the ready queue then it again starts its execution in Queue 1.
- If a process in queue 1 does not complete in 4 units then its priority gets reduced and it shifted to queue 2.
- Above points 2 and 3 are also true for queue 2 processes but the time quantum is 8 units. In a general case if a process does not complete in a time quantum then it is shifted to the lower priority queue.
- In the last queue, processes are scheduled in an FCFS manner.
- A process in a lower priority queue can only execute only when higher priority queues are empty.
- A process running in the lower priority queue is interrupted by a process arriving in the higher priority queue.

What is the need for such complex Scheduling?

- Firstly, it is more flexible than multilevel queue scheduling.
- To optimize turnaround time algorithms like SJF are needed which require the running time of processes to schedule them. But the running time of the process is not known in advance. MFQS runs a process for a time quantum and then it can change its priority(if it is a long process). Thus it learns from past behavior of the process and then predicts its future behavior. This way it tries to run a shorter process first thus optimizing turnaround time.
- MFQS also reduces the response time.

Example: Consider a system that has a CPU-bound process, which requires a burst time of 40 seconds. The multilevel Feed Back Queue scheduling algorithm is used and the queue time quantum ‘2’ seconds and in each level it is incremented by ‘5’ seconds. Then how many times the process will be interrupted and in which queue the process will terminate the execution?

Solution:

- Process P needs 40 Seconds for total execution.
- At Queue 1 it is executed for 2 seconds and then interrupted and shifted to queue 2.
- At Queue 2 it is executed for 7 seconds and then interrupted and shifted to queue 3.
- At Queue 3 it is executed for 12 seconds and then interrupted and shifted to queue 4.
- At Queue 4 it is executed for 17 seconds and then interrupted and shifted to queue 5.
- At Queue 5 it executes for 2 seconds and then it completes.
- Hence the process is interrupted 4 times and completed on queue 5.