Operating System(5th semester)

Prepared by SANJIT KUMAR BARIK(ASST PROF ,CSE)

MODULE-I

TEXT BOOK:

1. Operating System Concepts – Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, 8th edition, Wiley-India, 2009.
2. Mordern Operating Systems – Andrew S. Tanenbaum, 3rd Edition, PHI
3. Operating Systems: A Spiral Approach – Elmasri, Carrick, Levine, TMH Edition

S/W: Set of instructions or programs instructing a computer to do specific task

S/w is often divided into three categories:

1. System S/W: It controls all H/W components of the system. It serves as a base for application S/w. System s/w includes device drivers, operating system (OS), compiler, disk formatters and utilities helping the computers to operate more efficiently. It is also responsible for maintaining H/w components. The system s/w is usually written in C programming language.

2. Programming S/w (Utility programs): It is set of tools to aid developer in writing programs. The various tools available are compiler, Linker, debugger, interpreters and text editors.

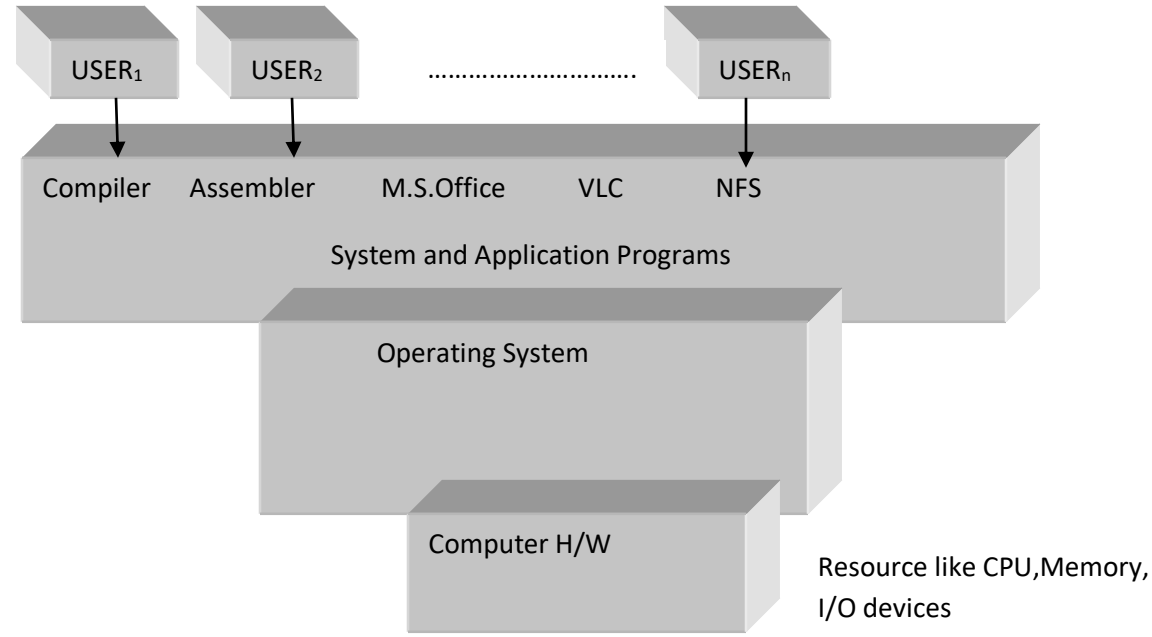
3. Application S/w: It is intended to form certain tasks. A set of programs written for satisfying a specific area of application is called application S/w.
   Ex: Ms Words, Gaming S/w, Database systems, Media player.

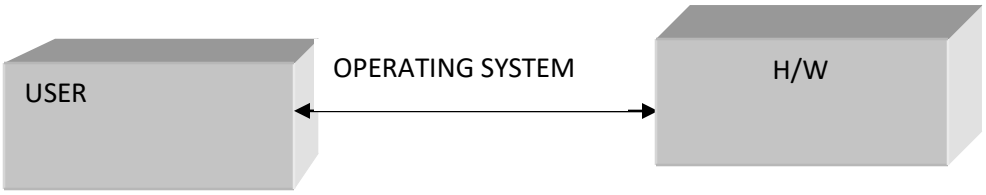
OPEARARING SYSTEM(OS):

It is the system software.

1. It acts as an intermediary between H/W and User
2. Resource Manager: Manage system resources in an unbiased fashion both H/W and S/W
3. Provide a platform on which other application programs are installed.
4. OS is a program that manages the computer H/W
5. It also provides a basis for application programs and acts as an intermediary between computer user and computer H/W

Abstract View of the System





Goals of Operating System

Primary Goal:

1. Convenience/User friendly

2ndary Goal:

1. Efficiency (Minimum input ---------→Max output)

Functions of operating system:

1. process management
2. Memory management
3. .I/O device management
4. File management
5. Network Management
6. Security and Protection

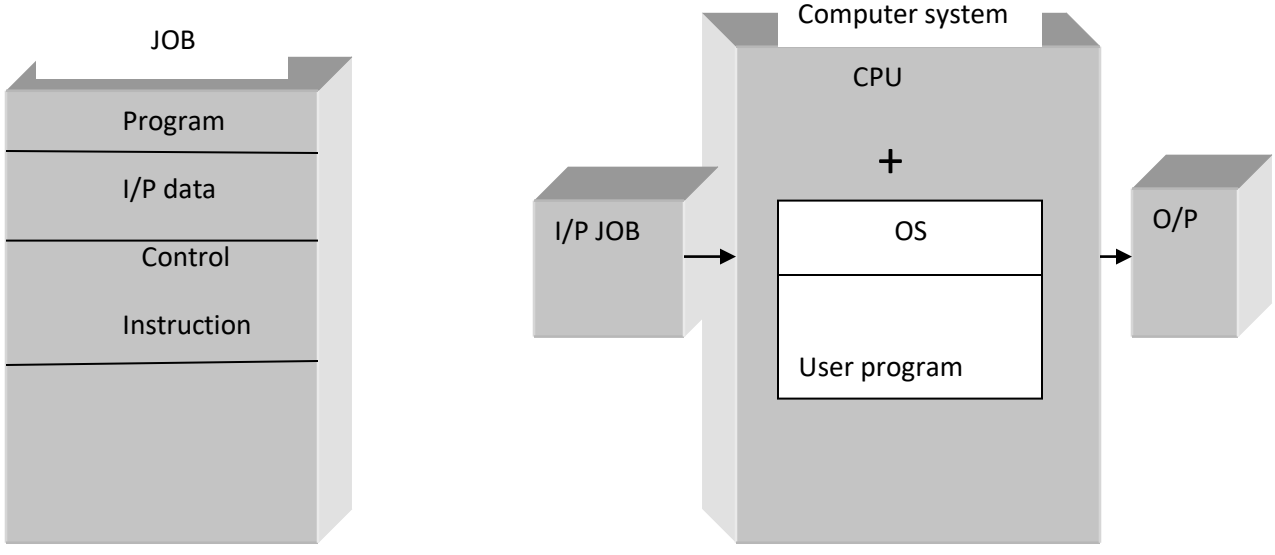
Types of Operating system

1. Batch operating system
2. Time sharing operating system
3. Distributed operating system
4. Network operating system
5. Real time operating system
6. Multiprogramming /Multiprocessing /Multitasking OS

Evolution of Operating System

In staring, mainframe computers, it has following concept:

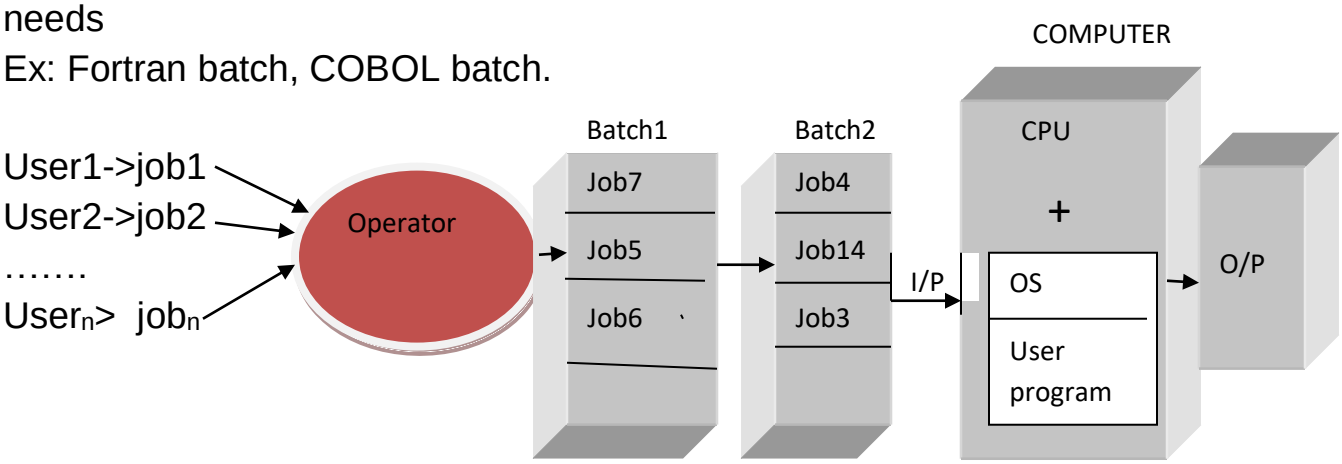
1. Common I/P and O/P devices were card readers and tape drivers
2. Users prepare a job which consist of the program, I/p data and control information
3. I/p job is given in the form of punch cards and results also appear in the form of punch card after processing.
4. So OS was very simple, always presents in the memory. Major task is to translate the control from one job to another.



Disadvantages ?

Batch processing:

1. Jobs with similar needs are batched together and executed through the processor s as a group.
2. Operators sorts jobs as a deck of punch cards into batch with similar needs
   Ex: Fortran batch, COBOL batch.



Advantages:

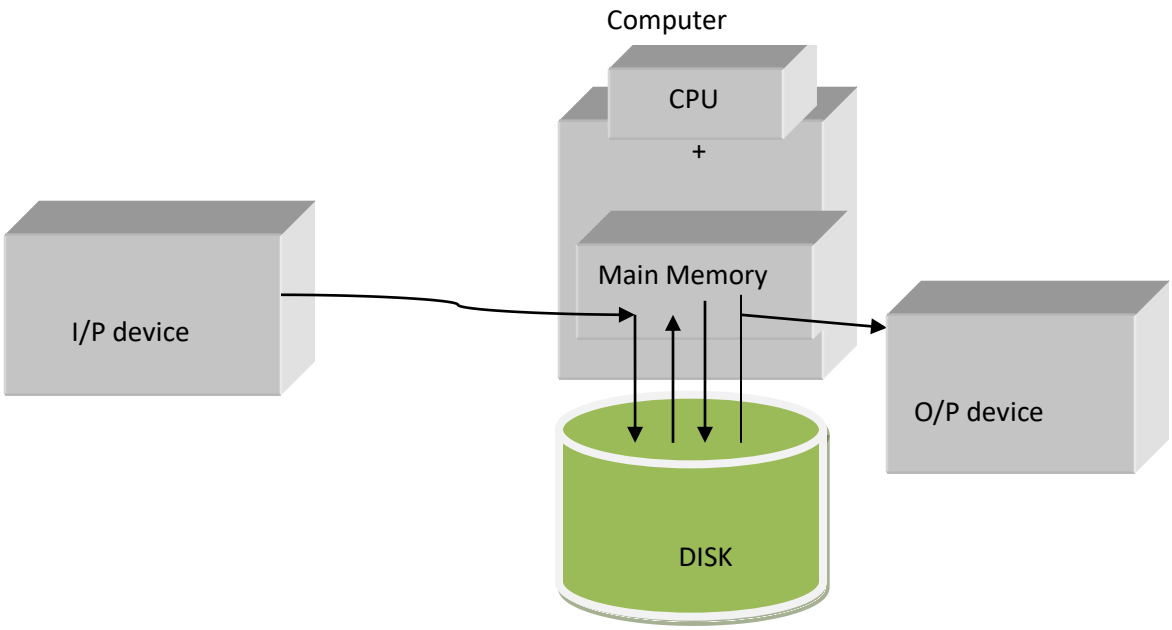
1. In a batch processing job excute one after another , saving time from activities like loading compiler
2. During a batch execution no manual intervention is needed.

Disadvantages:

1. Memory limitation
2. Interaction of I/P and O/P devices directly with CPU.
3. CPU utilization is idle.

Spooling :(simultaneous peripheral operation online)

1. I/P and O/P devices are relatively slow compare to CPU (digital)
2. In spooling , data is stored first onto disk and then CPU interact with Disk(digital) via main memory
3. Spooling is capable of overlapping I/O operations of one job with CPU operations of other jobs.



VIEW of SPOOLING

Advantages:

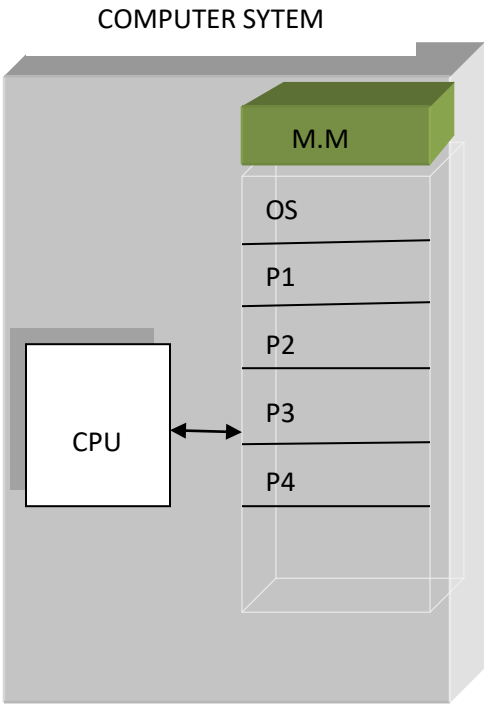
1. No interaction of I/P and O/P device with CPU
2. CPU utilization is more as CPU is busy most of time.

Disadvantages:

1. Spooling was uniprogramming. (Non –primitive scheduling)

Multiprogramming O/S

1. Maximize CPU utilization
2. Multiprogramming means more than one process in main memory which is ready to execute.
3. Process generally requires CPU time and I/O time. So if running process perform I/O or some other event which don't require CPU then instead of sitting idle,CPU make a context switch and picks some other process and this idea will continue.
4. CPU never sits idle unless there is process ready to execute or at time of context switch.



Advantage:

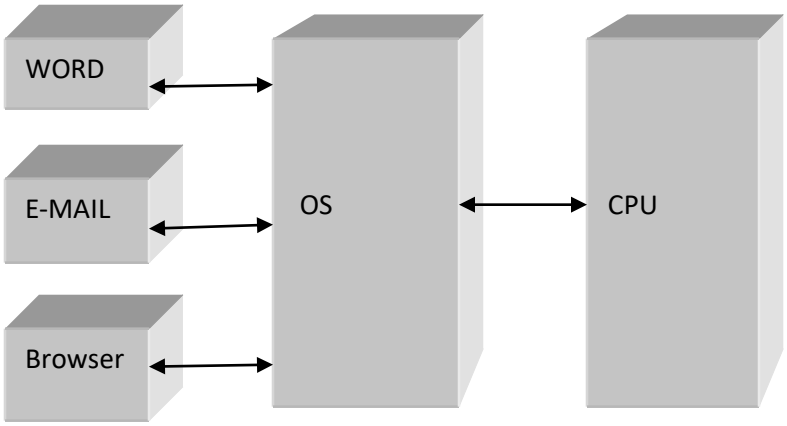
1. High CPU utilization
2. Less waiting time, response time etc.
3. May be extended to multiuser
4. Now-a-days, it is useful when load is more.

Disadvantages:

1. Difficulties in scheduling
2. Main memory management required
3. Memory fragmentation
4. Paging (Non-contiguous memory allocation)

Multitasking operating system/Time sharing /Faire share/Multiprogramming with Round –Robin:

1. Multitasking is the multiprogramming with time sharing
2. Only one CPU but switches between process so quickly that it gives on illusion all executing at same times.
3. The task in multitasking may refer to multiple thread of the same program.
4. Main idea is better response time and executing multiple process together.



Multiprocessing Operating System

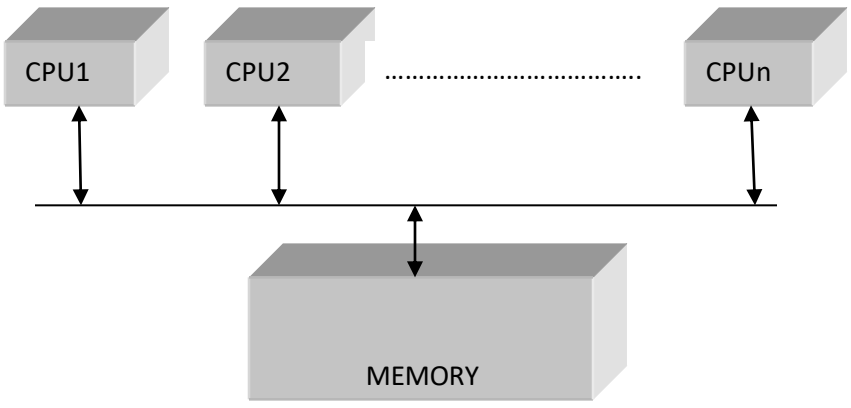
1. Two or more CPU within a single computer in close communication sharing the system bus, memory and other I/O devices
2. Different process may run of different CPU, true parallel execution.
3. Systematic: One OS control all CPU, each CPU has equal rights.
4. Asymmetric :Master slave architecture .System task on one processor and application on other as one CPU will handle all H/W interrupt as I/O devices, they are easy to design but less efficient.

Advantage:

1. Increase throughput
2. Increase reliability
3. Cost saving
4. True parallel processing
5. Better efficiency

Disadvantages:

1. More Complex
2. Overhead as coupling reduce throughput
3. Large main memory



FUNCTION OF OPERATING SYSTEM

1. Initial Loading of program
2. Process management
3. Main memory management
4. File Management
5. I/O system Management
6. 2ndary storage management
7. Networking
8. Protection or Security
9. Command – Interpreter System

1. Initial Loading of program

IO.SYS : Control input output devices
MSDOS.SYS:Load MS- Dos
CONFIG.SYS : Configures the device drivers
COMMAND.COM: Load the command line interpreter

2. Process management

- A program does nothing unless their instruction are executed by a CPU.A process is a program in execution. EX:

  - ✓ A time shared user program is a process.
  - ✓ A word processing program being run by an individual user on PC is a process.
  - ✓ A system task such as sending O/P to a printer is also a process.
- A process needs certain resources including CPU time, Memory files and I/O devices to accomplish its task.
- These resources are either given to process when it is created or allocated to it while it is running.

- ✓ Batch job is also kind of process
- ✓ A system task such as spooling O/P to a printer also is a process.

- OS is responsible for the following activities of process management:

  - ✓ Creating and deleting both user and system process
  - ✓ Suspending and resuming process
  - ✓ Providing mechanism for process synchronization
  - ✓ Providing mechanism for process communication
  - ✓ Providing mechanism for deadlock handling

3. Main memory Management

Main memory is a large array of words or bytes ranging from hundreds of thousands to billions. Main memory stores the quickly accessible data shared by the CPU and I/O devices. The CPU reads instruction from main memory during instruction fetch cycle and it both reads/writes from main memory during data fetch cycle. The main memory is generally the one large storage device that the CPU is able to address and access directly.

EX: for the CPU to process data from disk .Those data must be transferred to main memory by CPU generated I/O calls. The instruction must be in the memory for the CPU to execute them.

OS is responsible for the following activities in connection with main memory management:

- ✓ Keeping track of which parts of memory are currently being used and by whom.
- ✓ Deciding which processes are to be loaded into memory when memory space becomes available(in case multiprogramming)

- ✓ Allocating and De-allocating memory space as needed.(Recover memory space when process no longer needs it or has been terminated)

4. File management:

Computer can store information on several different types of physical media: magnetic tape, magnetic disk and optical disk. Each medium is controlled by a device such as disk drive or tap drive those has unique characteristics. These characteristics include access speed, capacity,transfer rate and access method(:sequential or random).

For convenient use of computer system, OS provides a uniform logical view of information storage .The OS abstracts from the physical properties of its storage devices to define logical storage unit file. A file is collection of related information defined by its creator. The OS is responsible for the following activities of file management:

- ✓ Creating and deleting files.
- ✓ Creating and deleting directories
- ✓ Supporting primitives for manipulating file and directories
- ✓ Mapping files into 2ndary storage.
- ✓ Backing up files on Non-volatile media(stable-storage media)
- ✓ Keep track of resources, its location, use, status etc. These collective facilities are often called file system

5. I/O system management: (Device management System)

One of the purposes of an OS is to hide the peculiarities of specific H/W devices from the User.
Ex: In Unix the peculiarities of I/O devices are hidden from the bulk of the OS itself by I/O subsystem.
The I/O subsystem consists of:

- ✓ A memory management component that includes buffering, catching and spooling.
- ✓ A general device driver interfaces drivers for specific hardware devices. Only the device driver knows the peculiarities of specific device to which it is assigned.

6. Secondary storage management :

The main purpose of computer system is to execute programs. These programs with the data they access must be in the main memory during execution. As the main memory is too small to accommodate all the data and programs,and also the data that it holds are lost when power is lost, the computer system must provide 2ndary storage device to back-up-main memory. Most modern computer systems are disks as the storage medium to store program and data. The O/S is responsible for the following activities of disk management:-

- ✓ Free space management
- ✓ Storage allocation
- ✓ Disk scheduling (FCFS, SSTF,SCAN,LOOK C-SCAN,C-LOOK)

7. Networking:

A distributed system is collection of processors that don't share memory, peripherals devices or a clock. Each processor has its own local memory, clock and processors communicate with one another through various communication lines such as a high speed buses or networks.The processor in the system are connected through communications networks which are configured in a number of different ways. The communication

network design must consider message routing and connection strategies: How to design a packet, what should be policies of routing,how to route the packet.

8. Protection and security:

If a computer system has multiuser and allow the concurrent execution of multiple processes, then the various processes must be protected from one another's activities. For that purpose , mechanism ensure that files, memory segment, CPU and other resources can be operated on by only those processes that have gained proper authorization from the O/S.

9. Command Interpretation:

One of the most important functions of the OS is connected interpretation where it acts as the interface between the user and OS.

- ✓ It provides better collection of command and command interpreter to the user.
- ✓ So that user can interact with the O/S.

OPERATING SYSTEM SERVICES:

An operating system provides an environment for the execution of the program. It provides some services to the program. The various services provided by an operating system are as follows:

1. Program execution: The system must be able to load a program into memory and to run that program. The program must be able to terminate this execution either normally or abnormally.
2. I/O operation: A running program requires I/O. This I/O may involve a file or I/O device for specific process. Some special function can be desired. Therefore the O/S must provide to do I/O operations.
3. File system manipulation: The program need to create and delete files by name, read and write files. Therefore the O/S must maintain each and every file correctly.
4. Communication: The communication is implemented via shared memory or by the technique of message passing in which packets of information are moved between the processes by the O/S.
5. Error Detection: The O/s should take the appropriate actions for the occurrence of any type like arithmetic overflow, access to the illegal memory location and too large user's CPU time.
6. Resource Allocation: When multiple users are logged on to the system, the resources must be allocated to each of them. For correct distribution of resource among the various process, the OS uses the CPU scheduling , which determines which process will be allocated with resource
7. Accounting: The OS keep track of which users use ,how many, and which kind of computer resources

8. Protection: The O/S is responsible for both H/W as well as S/W protection. The O/S protects the information stored in multiuser computer system

SYSTEM CALLS:

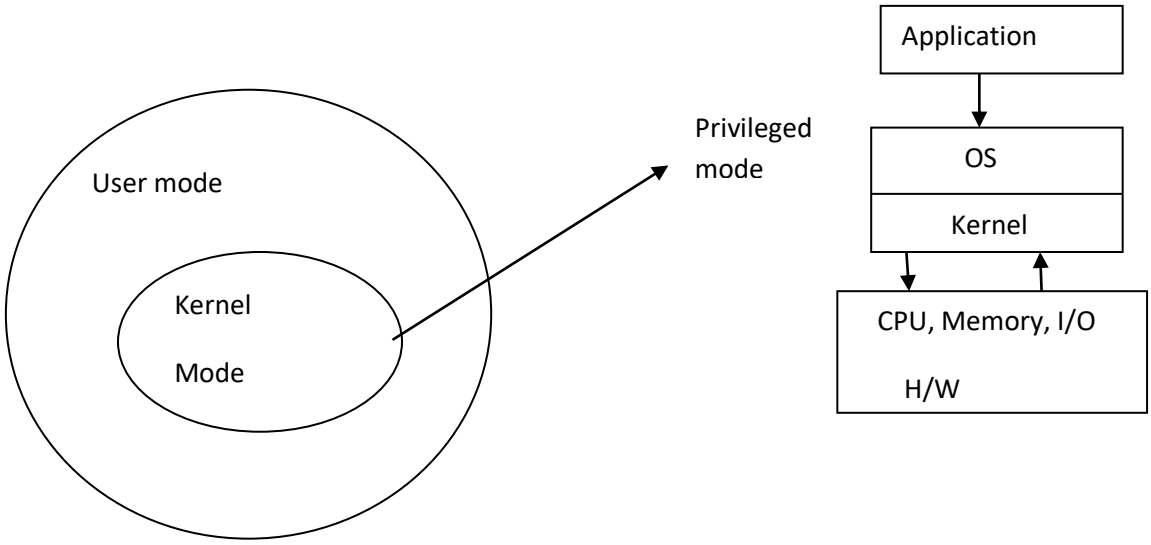
System calls provide the interface between a process & the OS. These are usually available in the form of assembly language instruction. Some systems allow system calls to be made directly from a high level language program like C, BCPL and PERL etc. systems calls occur in different ways depending on the computer in use. System calls can be roughly grouped into 5 major categories.

1. Process control
2. File manipulation
3. Device Management
4. Information Maintenance
5. Communication

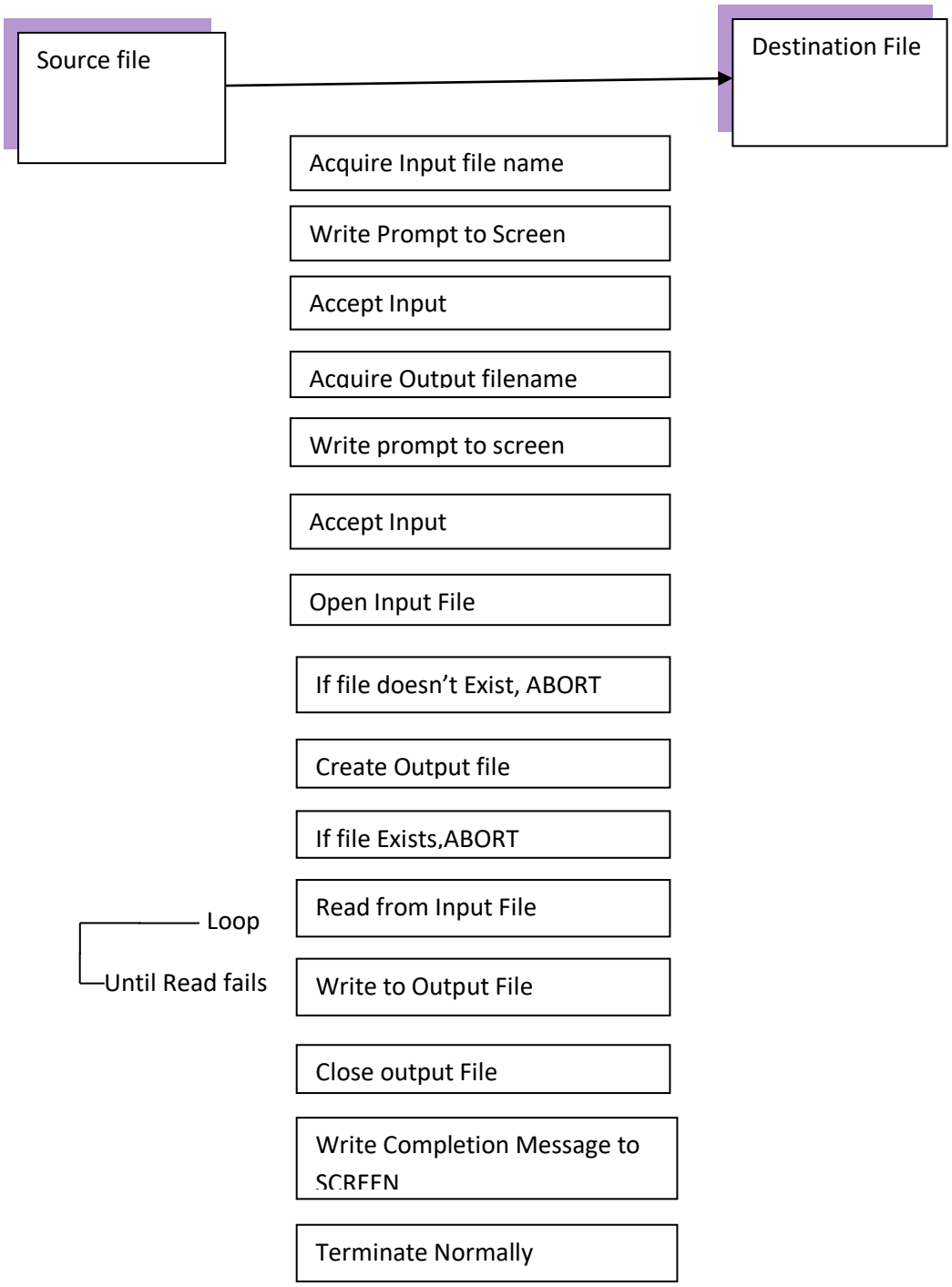
- System calls provide an interface to the service made available by an Operating system.
- System call is the programmatic way in which a computer program requests a service from kernel of the operating system.
- These calls are generally available as routines written in C and C++.

*BCPL: Basic combined programming Language

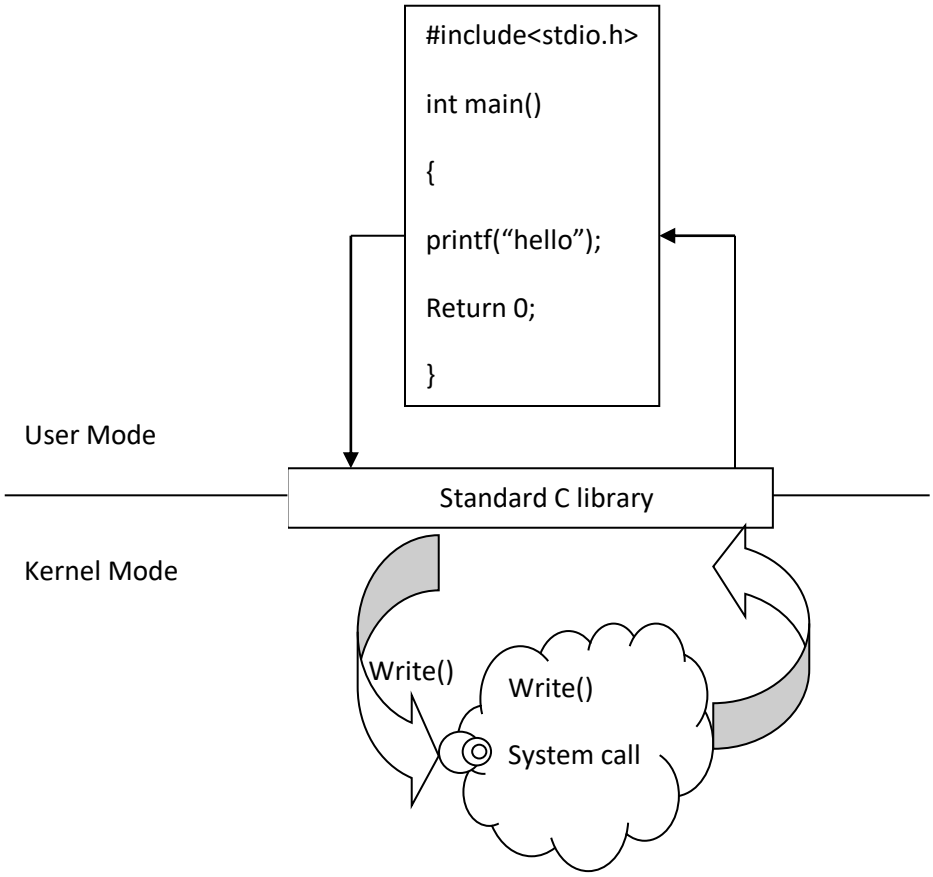
*PERL: Practical Extraction and report language



Ex: of Systems call sequence for writing a simple program to read data from one file and copy them to another file.



Ex:2 The standard C library provides a portion of system call interface for many versions of UNIX and LINUX. Let's assume a C program invokes the printf() statement. The C library intercepts this call and invokes the necessary system call(or calls) in the operating system-in this instance, the write() system call. The C library takes the value returned by write () and passes it back to the user program



1. Process Control:

- **End, abort:** A running program needs to be able to has its execution either normally (end) or abnormally (abort).
- **Load, execute:** A process or job executing one program may want to load and executes another program.

- **Create Process, terminate process:** There is a system call specifying for the purpose of creating a new process or job (create process or submit job). We may want to terminate a job or process that we created (terminates process, if we find that it is incorrect or no longer needed).
- **Get process attributes, set process attributes:** If we create a new job or process we should able to control its execution. This control requires the ability to determine & reset the attributes of a job or processes (get process attributes, set process attributes).
- **Wait time:** After creating new jobs or processes, we may need to wait for them to finish their execution (wait time).
- **Wait event:** We may wait for a specific event to occur (wait event).

  Example of system call: fork(),exit(),kill(),nice()
- Allocate and free memory:

2. File Manipulation:

- **Create file, delete file:** We first need to be able to create & delete files. Both the system calls require the name of the file & some of its attributes.
- **Open file, close file:** Once the file is created, we need to

open it & use it. We close the file when we are no longer using it.

- **Read, write, reposition file:** After opening, we may also read, write or reposition the file (rewind or skip to the end of the file).

- **Get file attributes, set file attributes:** For either files or directories, we need to be able to determine the values of various attributes & reset them if necessary. The system calls get file attribute & set file attributes are required for their purpose.

  Example of system call: Create(),open(),close(),read(),write()

3. Device Management:

- **Request device, release device:** If there are multiple users of the system, we first request the device. After we finished with the device, we must release it.

- **Read, write, reposition:** Once the device has been requested & allocated to us, we can read, write & reposition the device.
- Get device attributes, set device attributes:
- Logically attach or detach devices:

4. Information maintenance:

- **Get time or date, set time or date:** Most systems have a system call to return the current date & time or set the current date & time.

- **Get system data, set system data:** Other system calls may return information about the system like number of current users, version number of OS, amount of free memory etc.

- **Get process attributes, set process attributes:** The OS keeps information about all its processes & there are system calls to access this information.

  Example of system call:gettime(),getdate(),settime(),getprocees(),set process(),get system data(),set system data(),set file() and so on.

5. **Communication:**

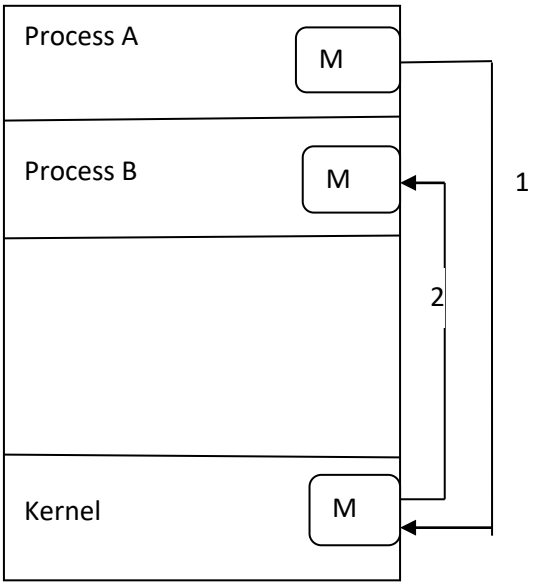
- create, delete communication connection
- Send, receive messages
- Transfer status information
- Attach or detach remote devices

There are two modes of communication such as:

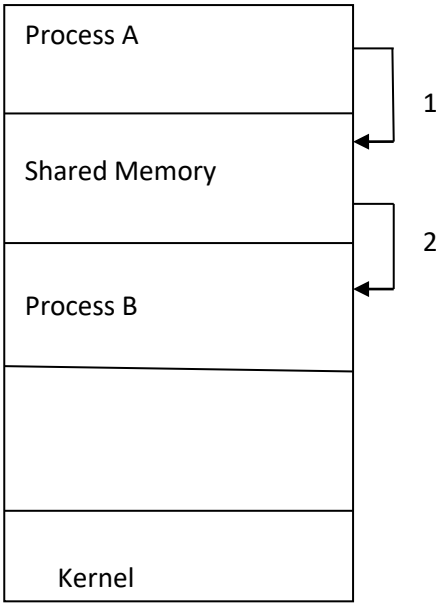
- ✓ **Message passing model:** Information is exchanged through an inter process communication facility provided by operating system. Each computer in a network has a name by which it is known. Similarly, each

process has a process name which is translated to an equivalent identifier by which the OS can refer to it. The get hostid and get processed systems calls to do this translation. These identifiers are then passed to the general purpose open & close calls provided by the file system or to specific open connection system call. The recipient process must give its permission for communication to take place with an accept connection call. The source of the communication known as client & receiver known as server exchange messages by read message & write message system calls. The close connection call terminates the connection.

- ✓ **Shared memory model:** processes use map memory system calls to access regions of memory owned by other processes. They exchange information by reading & writing data in the shared areas. The processes ensure that they are not writing to the same location simultaneously.



Message Passing



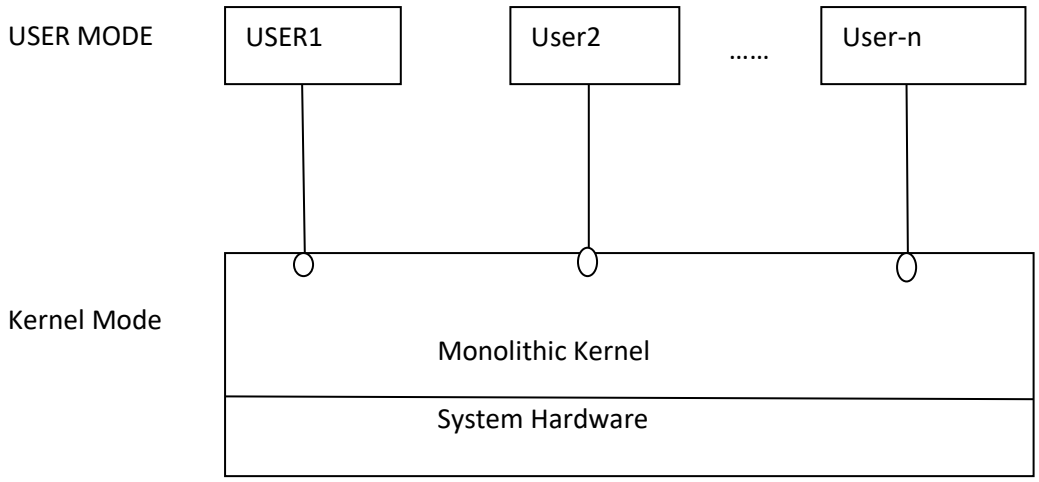
Shared Memory

Operating system structure

1. Monolithic Systems (Simple structure)
2. Layered Structure
3. Microkernel systems

1. Monolithic Systems (Simple structure)

Monolithic operating system is one of the simple operating system architecture and is in use from very early times. Earlier computer systems were relatively simple having single CPU,a small size memory and few input output devices. In order to manage these simple systems, monolithic kernels were best suited. In monolithic kernel design all the components of the operating system like process management, memory management, device management, file management, etc are all integrated under a single unit in the kernel address space. There is one code (program) consisting of routines, subroutines and data is written and stored in memory. The components of the operating system use this code and provide necessary functionality. The various components of operating system can also communicate with each others required in order to accomplish the given task. OS/360,VMS and Linux are all examples of Monolithic operating system.



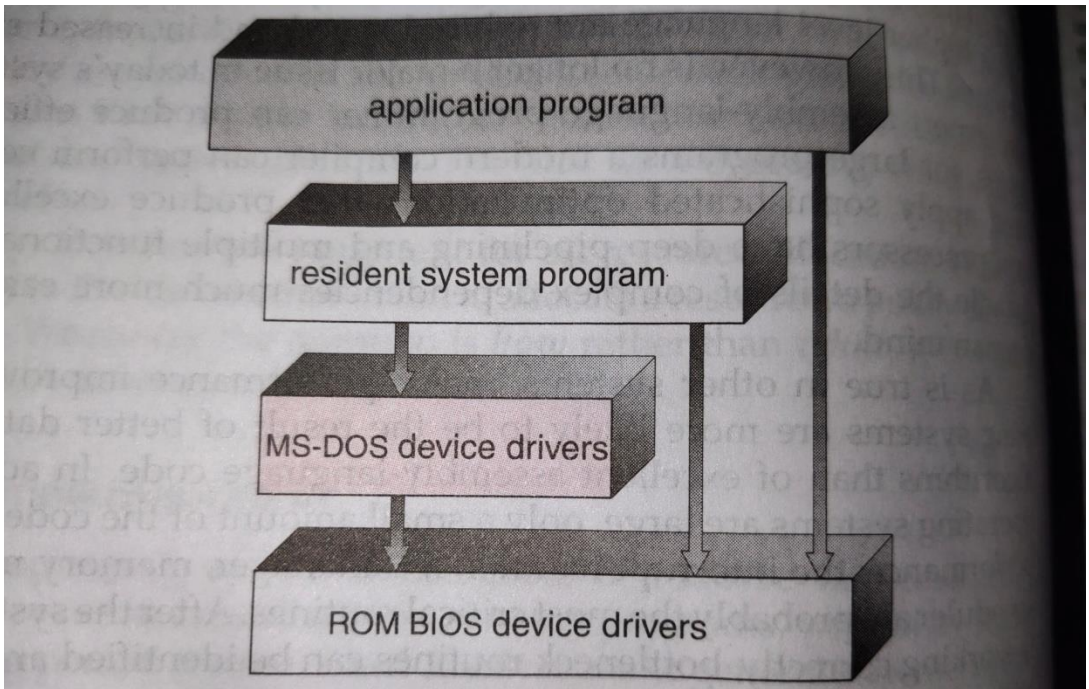


Figure 2.11 MS-DOS layer structure.

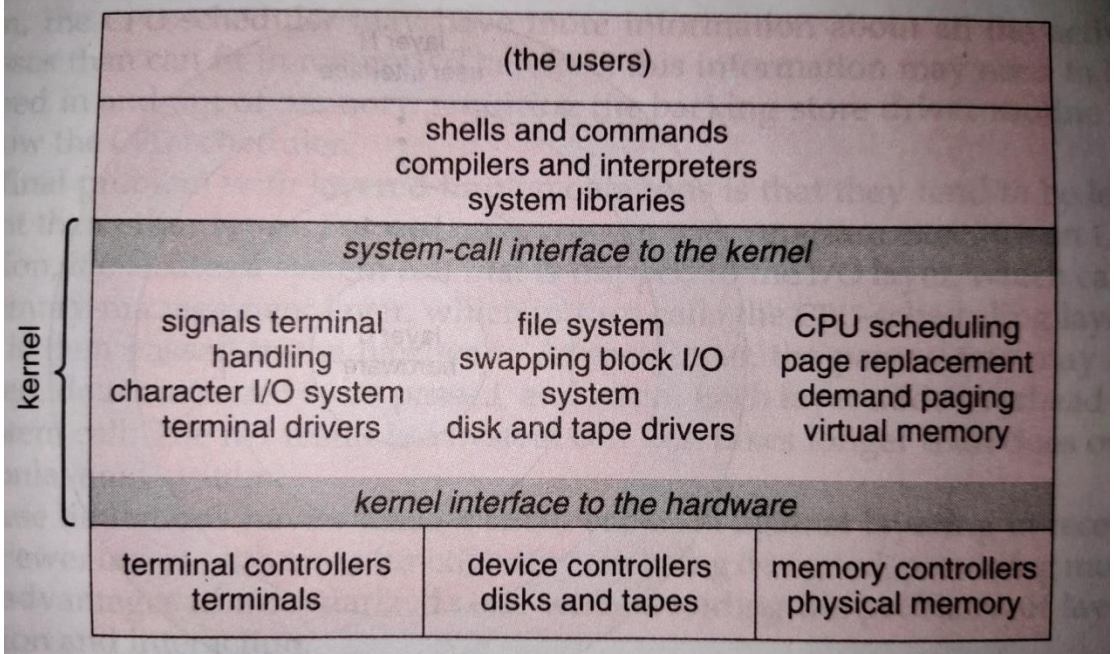


Figure 2.12 Traditional UNIX system structure.

Advantages of monolithic Kernels:

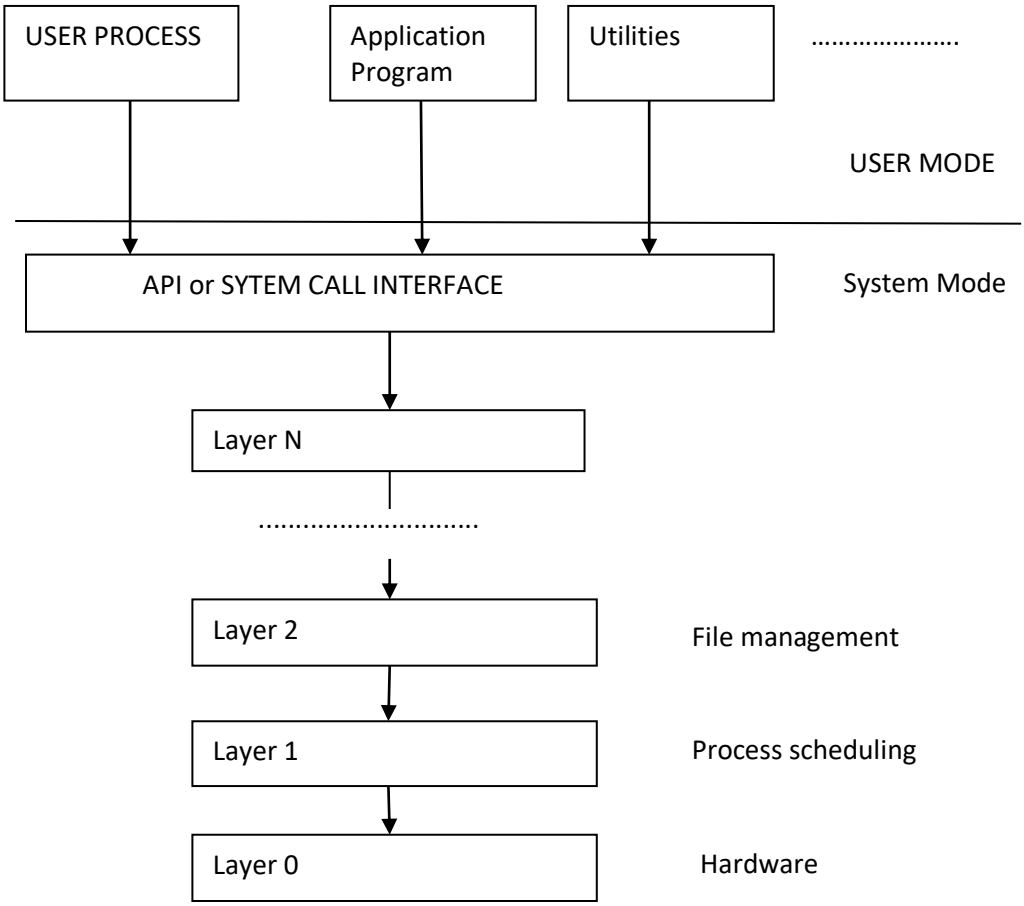
- Monolithic kernels are quite fast. The code is accessible to all the components of the operating system and this code can be executed without much restriction making the overall system fast.
- Monolithic kernels provide direct communication between components which makes the system more efficient to work

Disadvantages:

- Monolithic kernels are more prone to errors and bugs as user process runs in same address spaces as that of kernel. Also maintaining the code also becomes difficult.
- Harder to port because of dependency on code.
- Adding /removing any feature or functionality in monolithic system are quite difficult and often require rewriting and recompiling the whole code again.

2. Layered Structure

The first layer based operating system was proposed by E.W dijkstra and his team. In layered approach the operating system consist of several layers where each layer has a well defined functionality and each layer can be designed, coded and tested independently. The layers are arranged in increasing order of abstraction, means the lowest layer(0) interacts and deals with the underlying hardware and topmost layer(N) provides an interface to the application programs and user program(process).Each layer relies on the service of the layers below it. The communication takes place only between adjacent layers. For example: layer 3 can request a service from layer 2 immediately below it and it can provide service only to the layer 4 immediately above .Each layer knows what services are provided by the layer above it but the details as how these services are provided are hidden. The different layers can be file management layer, memory management layer, communication management layer, User program layer etc.



Example of layered operating system is THE operating system which consists of 6 layers .Another example is MULTICS system.

Advantages of layered structure:

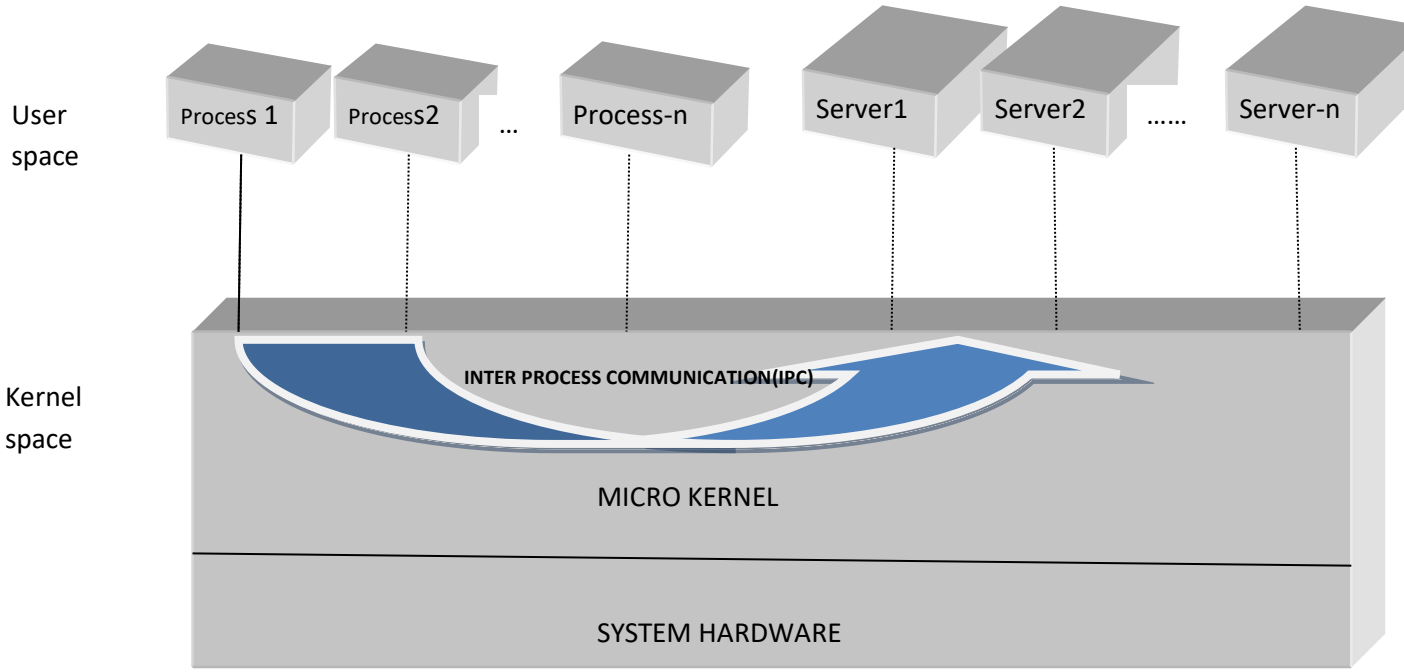
- In the layered approach it is easier to add any new features or make changes in one layer without affecting the other layers of the operating system.
- Any bug or problem in a particular layer is restricted to that layer only. That layer can be debugged, corrected and redesigned without affecting the functionality of other layers.
- Easy to add new layer as and when required.

Disadvantages:

- The more the layers, the more is overhead incurred to maintain them.
- If the functionality is not properly divided among the layers, it might be possible that one layer has too much functionality. If this happens that layer may be overburden leading to overall low system performance.

3. Microkernel systems:

Microkernel systems were designed to solve the problems faced by monolithic kernels. Micro kernel architecture aims at keeping minimum functionality within the kernels but at the same time provided the abstraction for a complete operating system designing and implementation. The basic functionalities like :
Process scheduling, inter-process communication, thread management, low level space management and interrupt management were included within the kernel address space only. However other functionalities like memory management, file management, I/O device management, Networking etc. were not included within the kernel instead each of these functionalities are implemented through separate server process. The server processes reside outside the kernel in the user address space .Each server process provides separate functionalities and occupies separate memory space. The user process acts as client. Whenever user process need do something ,it send request to required server process and wait for reply. The micro kernel using inter process communication (IPC) sends the user request to the specific server. The server processes the request and sends back the output and other details back to the user process that generated the request.



Example of Micro kernel operating system:

Windows NT,WindowsXp, Mach ,AIX,OS X, MINIX etc are OS ,based on micro kernel.

Advantages:

- It is easier to add new functionality or to modify existing functionality in micro kernel system
- System is very reliable because fault in one server process do not halt working of entire system. Other server process continues to its work.
- Easily portable to different computer platforms.
- Minimum functionality within the kernel makes the kernel small and easy to maintain.

Disadvantages:

- It has low execution rate as higher inter-process communication and context switching involved.

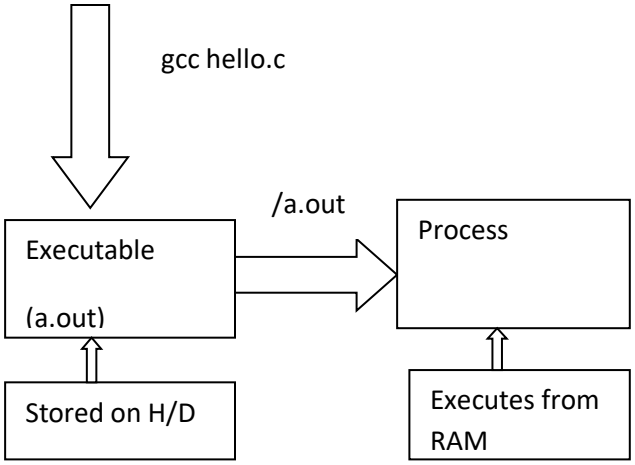
Process Management

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

Ex: We write our computer program in C-editor and we execute this program, it becomes a process which performs all the tasks mentioned in the program.

Ex: # include<stdio.h>

```
int main()
{
char str[ ]=”Hello”;
printf(“%s”, str);
}
```



- ✓ A program in execution
- ✓ Present in the RAM

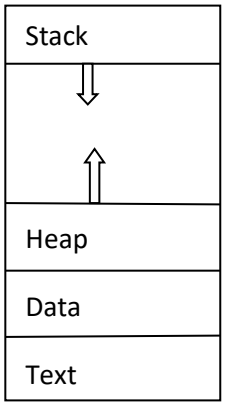
Difference between program and process:

Basic	program	Process
1. Basic	Program is set of instruction	When a program is executed ,it is known as process

2. Nature	Passive	Active
3. Lifespan	Longer	Limited
4. Required resources	Program is stored on disk in some file and doesn't require any other resources	Process hold resources such as CPU,Memory,Address ,Disk,I/O etc

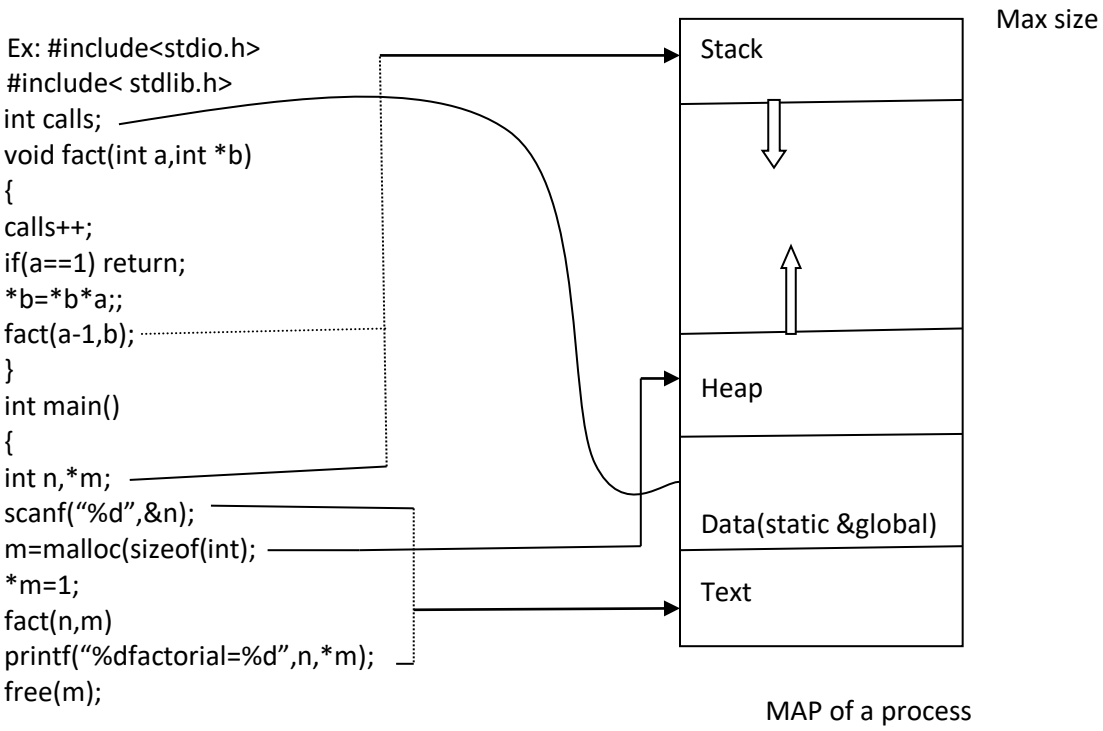
Process can be divided into four sections:

1. Stack 2. Heap 3. Text 4. Data



1. Stack : The process stack contains the temporary data such as method/Function parameters, return address and local variables.
2. Heap: This is dynamically allocated memory to a process during its run time.
3. Text: It contains the executable code. It also contains current activity represented by the value of program counter and contents of processors registers

4. Data: This section contains the global and static variable
   - ❖ Text sections are equivalent, the Data, Heap and Stack sections vary for same program of different users



- ❖ max size is fixed and decided by OS

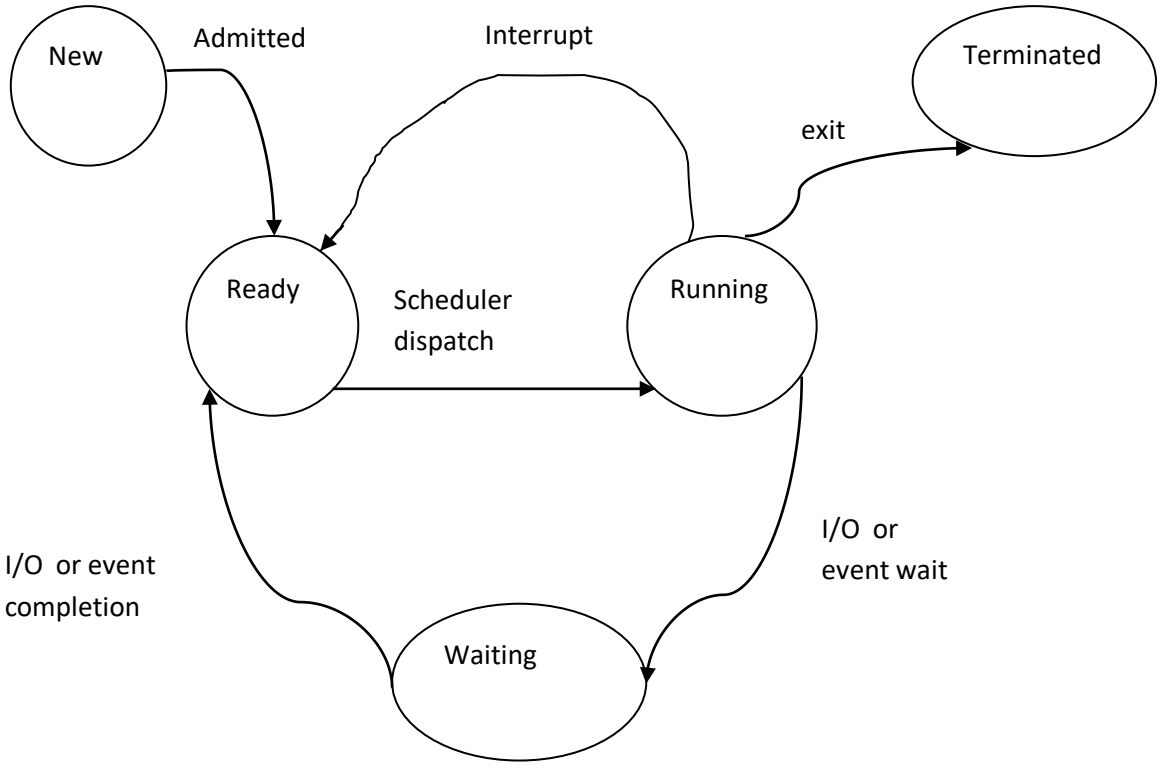
Process Life Cycle (Process State)

When a process executes, it passes through different states. These stages may differ in different operating systems and the names of these states are not standardized.

1. New: The process is being created.
2. Running: Instructions are being executed.
3. Waiting: The process is waiting for some event to occur (Ex: I/O completion or reception of a signal)
4. Ready: The process is waiting to be assigned to a processor.
5. Terminated: The process has finished execution.

- ❖ As a process executes, it changes its state.
- ❖ The state of process is defined in part by the current activity of that process.

Diagram of process state



Process Control Block(PCB) OR Task control Block (TCB):

A process control block is a data structure maintained by OS for every process. The PCB is identified by an integer process ID (PID).It contains many pieces of information associated with a specific process. These are:

1. Process state: The state may be New, Ready, Running, Waiting, Halted and so on.
2. Process privileges: This is required to allow /disallow access to system resources.
3. Program Counter: The counter indicates the address of the next instruction to be executed for this process
4. Process ID: Unique identification for each of the process in the O/S
5. CPU register: Various CPU registers (AC,index registers,stack pointer ,GPR registers) where process need to be stored for execution for running state.Along with the program counter ,this state information must be saved when an interrupt occurs, to allow the process to be continued correctly afterward(refer figure 3.4)
6. CPU-scheduling information: information includes process priority and other scheduling information which is required to schedule the process.
7. Memory –management information: this includes the information of page table, memory limits, segment table depending on memory system used by the O/S.
8. Accounting Information:This includes the amount of CPU and real time used, time limits, account numbers, process ID and so on.
9. I/O status information: Information includes the list of I/O devices allocated to process, a list of open files and so on.

- The PCB is maintained for a process throughout its life time and is deleted once the process terminated

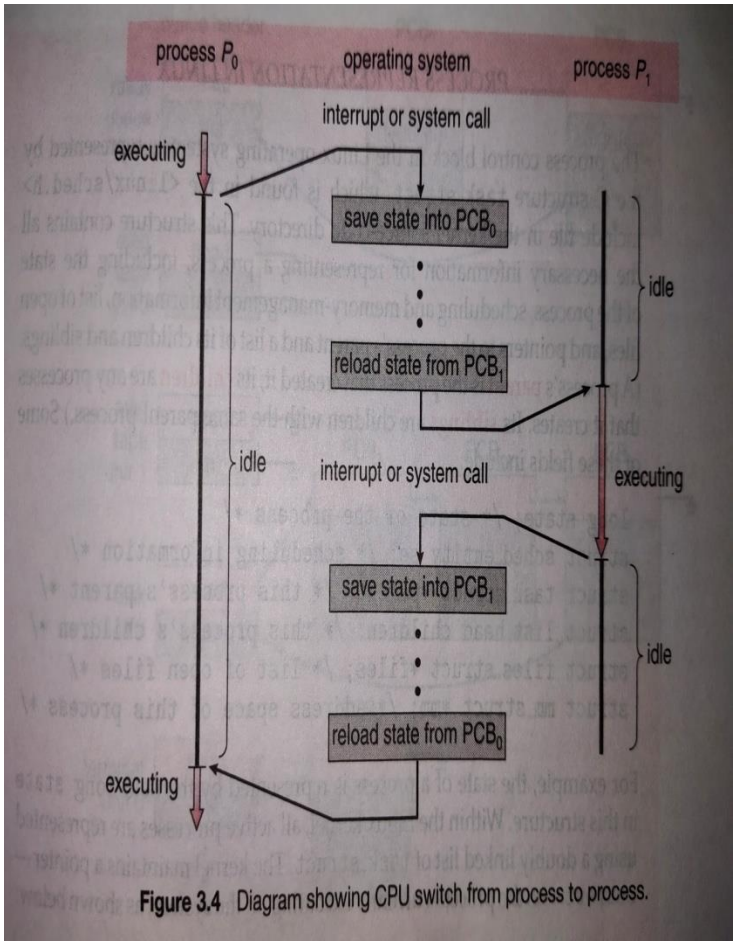


Figure 3.4 Diagram showing CPU switch from process to process.

Process ID
State
Program counter
CPU registers
CPU –scheduling Information
Memory-Management
Accounting information
I/O status information
……

PCB LAYOUT

Process scheduling

- The objective of multiprogramming is to have some process running at all times, to maximize CPU utilization
- The objective of time sharing is to switch the CPU among processes so frequently that users can interact with each program while it is running.
- To meet these objectives the process scheduler selects an available process (Possibly from set of several available process) for program execution on the CPU.
    - ✓ For a single –process system, there will never be more than one running process.
    - ✓ If there are more processes, the rest will have to wait until the CPU is free and can be rescheduled.

Definition: The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.

Scheduling Queues:

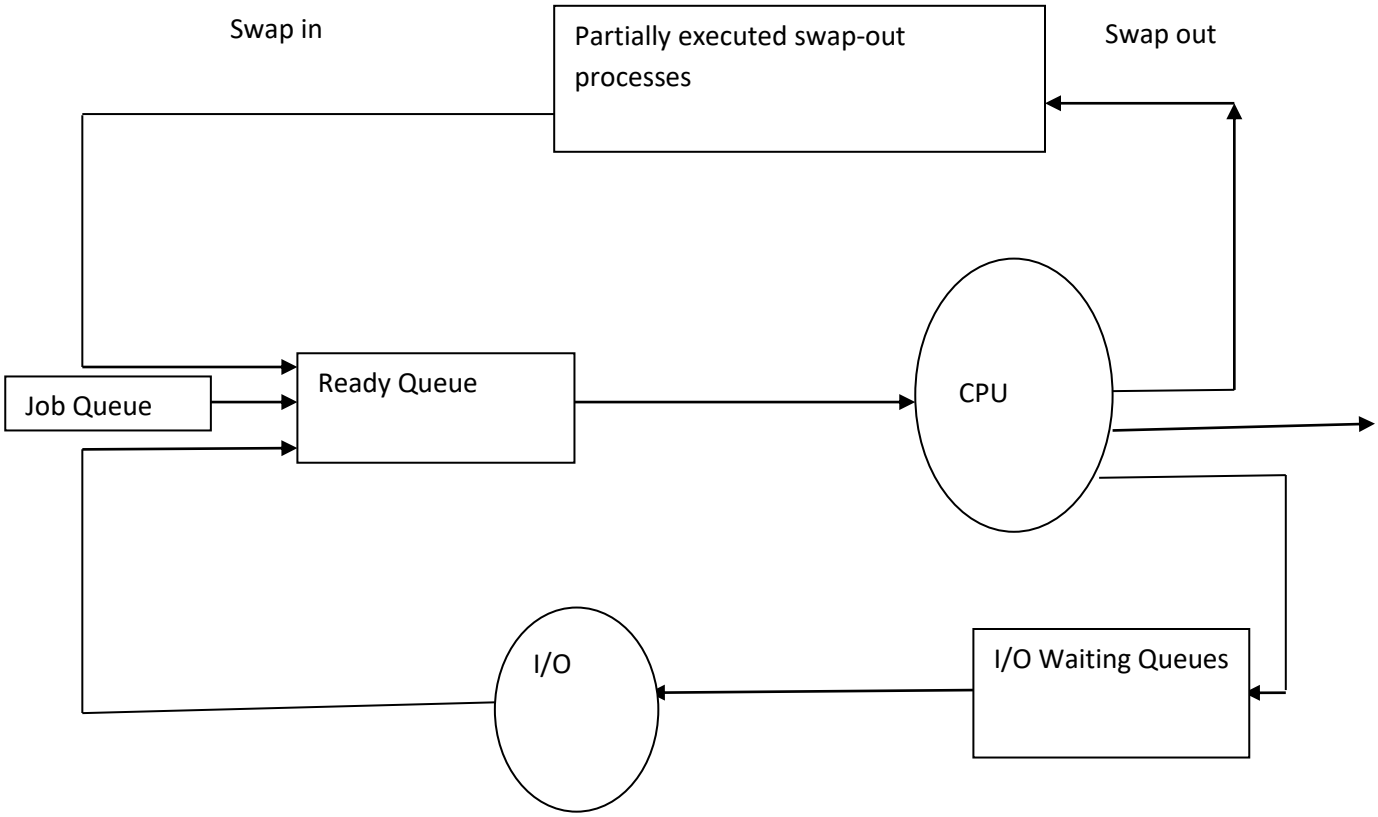
Job Queue: As process enters the system, they are put into job queue, which consist of all processes in the system.

Ready Queue: The process that are residing in main memory and are ready and waiting to execute are kept on a list called the ready queue.

Device queue (I/O waiting Queue):

The processes which are blocked due to unavailability of I/O device constitute this queue.

The OS can use different policies to manage each queue (FIFO,R-R, Priority and so on).The Os scheduler determines how to move processes between the ready and run queues which can only have one entry per processor core on the system(Explain in Below diagram).



Schedulers:

A process migrates between the various scheduling queues throughout its life time purposes. The OS must select for the scheduling processes from these queues in some fashion. This selection process is carried out by appropriate scheduler.

Definition: Schedulers are the special system S/W which handles process scheduling in various ways. Their main task is to select the jobs to be submitted into system and to decide which process to run.

Schedulers are of three types:

1. Long-Term schedulers
2. Short-Term schedulers.
3. Medium-term schedulers.

1. Long-Term schedulers(JOB schedulers)
   Long term schedulers select process from disk (job-Queue) and loads them into memory for execution. It controls the degree of multiprogramming i.e no. of process in the memory. It executes less frequently than other schedulers. If the degree of multiprogramming is stable than average rate of process creation is equal to average departure rate of processes leaving the system. So ,the long term scheduler is needed to be invoked only when a process leaves the system.

   The primary goal (objective) of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and Process-Bound.

I/O bound: An I/O bound process spends in doing I/O operation (Ex:C-programming)

CPU bound: It spends more times in doing computation than I/O opearations (Ex: Complex sorting program)

2. Short-Term schedulers.(CPU Schedulers):

   It selects among the process that are ready (ready queue) and allocates the CPU to one of them. Short term schedulers is known as dispatcher, make the decision of which process to execute next. It selects a new process for the CPU quite frequently. It execute at least one in 100ms.Due to the short duration of time between executions, it is very fast.

3. Medium-term scheduler:

   The main idea behind this scheduler is that sometimes it is advantageous to remove process from memory and hence reduce the degree of multiprogramming .At some later time, the process can be reintroduced into memory and its execution can be continued from where it had left off. This is called as swapping .The process is swapped out and swapped in by medium term scheduler.

   Swapping is necessary to improve the process mix or due to some changes in memory requirements, the available memory limit is executed which requires some memory to be freed up. (explain in above fig.)

Comparison among scheduler

	Long Term scheduler	Short term scheduler	Medium term scheduler
1	It is a job-scheduler	It is CPU scheduler	It is process swapping scheduler
2	Speed is lesser than short term schduler	Speed is fastest among other two	Speed is in between both short and long term scheduler
3.	It controls the degree of multiprogramming	It provides lesser controls over degree of multiprogramming	It reduces the degree of multiprogramming
4	It is almost absent or minimal in time sharing	It is also minimal in time sharing system	It is the part of time sharing system
5	It selects process from pool (job queue) and loads them into memory for execution	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued

Context switch:

A context switch is the mechanism to store and restore the state or context of CPU in process control block (PCB) so that a process execution can be resumed from the same point at a later time. Using this technique, a context switch enables multiple processes to share a single CPU. Context switching is essential part of multitasking operating features.

When the scheduler switches the CPU from executing one process to execute another, the state from current running process is stored into the process control block. After this ,the state for the process to run next is loaded from its own PCB and used to set the PC,registers and so on. At that point, the 2nd process can start executing.(refer fig.3.4)

Context switches are automatically intensive since register and memory state must be saved and restored. To avoid the amount of context switching time, some H/W systems employs two or more sets of processor registers when the process is switched, the following information is stored for later use.

1.PC 2.sheduling information 3.base and limit registers use 4.currently used register 5.changed state 6.I/O state information 7.Accounting information

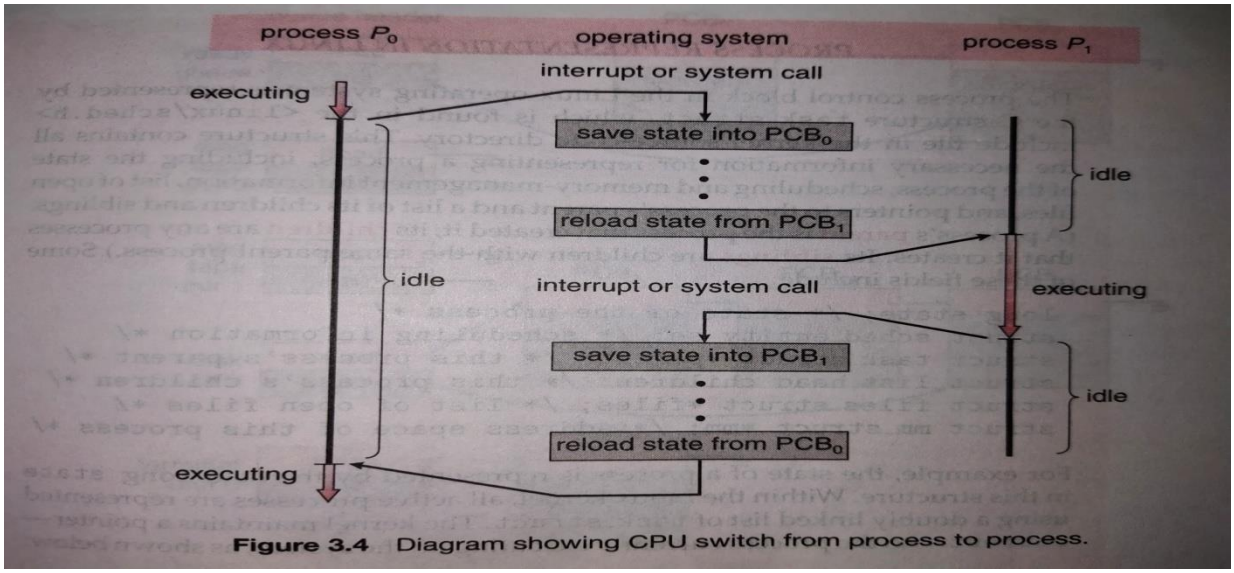


Figure 3.4 Diagram showing CPU switch from process to process.

CPU Scheduling

CPU scheduling is the process which allows one process to use the CPU while the execution of another process is on hold (waiting state) due to unavailability of any resource like I/O etc, thereby making full use of CPU. The aim of CPU scheduling is to make the system efficient, fast and fair.

Whenever the CPU becomes idle, the operating system must select one of processes in the ready queue to be executed .The selection process is carried out by the short-term scheduler.(or CPU scheduler).

The scheduler selects from among the processes in the memory that are ready to execute and allocates the CPU to one of them.

CPU scheduling deals with the problem of deciding which of the processes in the ready queue is to be allowed first to CPU.

Dispatcher:

Another component involved in the CPU scheduling function is the Dispatcher. The Dispatcher is the module that gives control of the CPU to the process selected by the short –term scheduler .This function involves:

1. Switching context
2. Switching to user mode
3. Jumping to the proper location in the user program from where it left last time.

The dispatcher should be as fast as possible, given that is invoked during process switching. The time taken by the dispatcher to stop one process and start another is known as the dispatch latency.

Scheduling can be two ways:

1. Non-preemptive Scheduling:
   Under non-preemptive scheduling, once the CPU has been allocated to a process, and then process keeps the CPU until it releases the CPU either by terminating or by switching to the waiting state.
   Ex: Microsoft 3.1 ,Apple Macintosh O/S
2. Preemptive scheduling:
   Under this scheduling the process has to leave the CPU forcefully on the basis of some criteria like running to ready and waiting to ready state transaction.
   Ex: At times to run a certain task that has a higher priority before another task although it is running. Therefore the running task is interrupted for some time and resumed later when the priority task has finished its execution.

CPU scheduling decisions may take place under the following four circumstances :->

1. When a process switches from the running state to the waiting state
   (For I/O request or invocation of wait for the termination of one of the child processes)
   i.e. non primitive

2. When a process switches from the running state to the ready state

   (Ex: when an interrupt occur i.e preemptive)

3. When a process switches from the waiting to ready state (Ex: Completion of I/O i.e preemptive)

4. When a process terminates i.e. non –preemptive

In circumstances 1 and 4, there is no choice in terms of scheduling. A new process (if one exists in the ready queue) must be selected for execution. There is a choice, however in circumstances 2 and 3.

When scheduling takes place only under circumstances 1 and 4, we say the scheduling scheme is non–preemptive; otherwise the scheduling scheme is preemptive.

Scheduling Criteria:

There are many different criteria's to check when considering the best scheduling algorithm:

1. CPU utilization
   We want to keep the CPU as busy as possible. CPU utilization may range from 0 to 100%.In a real system, CPU usage should range from 40 %(Light –loaded) to 90 % (heavily loaded).
2. Throughput:
   It is the total no. of processes completed per unit time. This may range 10/sec to 1/hour depending on the specific process.(length of process).
3. Turnaround Time:
   It is the amount of time taken to execute a particular process i.e the interval from time of submission of the process to the time of completion of the process.
   Turnaround Time (TAT) = finishing time-arrival time.
   (Turnaround time is the sum of periods spend waiting to get memory,waiting in the ready queue ,executing on the CPU and doing I/O operation)
   TAT=Completion Time (C.T)- Arrival time(A.T)
4. Waiting Time:
   Waiting time is the sum of periods spends waiting in the ready queue.
   Waiting Time(WT)=starting time-arrival time
   (The CPU scheduling algorithms doesn't affect the amount the time during which a process executes or does I/O operation, it affects only the amount of time that a process spends waiting in ready queue)
   Waiting Time=TAT-BT (Burst time)

Burst time/execution time/running time=It is the time, process require for running on CPU

5. Response Time:

   Amount of Time it takes from when a request was submitted until the first response is produced.

   It is the time, the first response and not the completion of process execution.

   Response Time(RT)=First response-Arrival Time.

   (i.e time between a process enters ready queue and get scheduled on the CPU for the first time)

Optimization criteria:

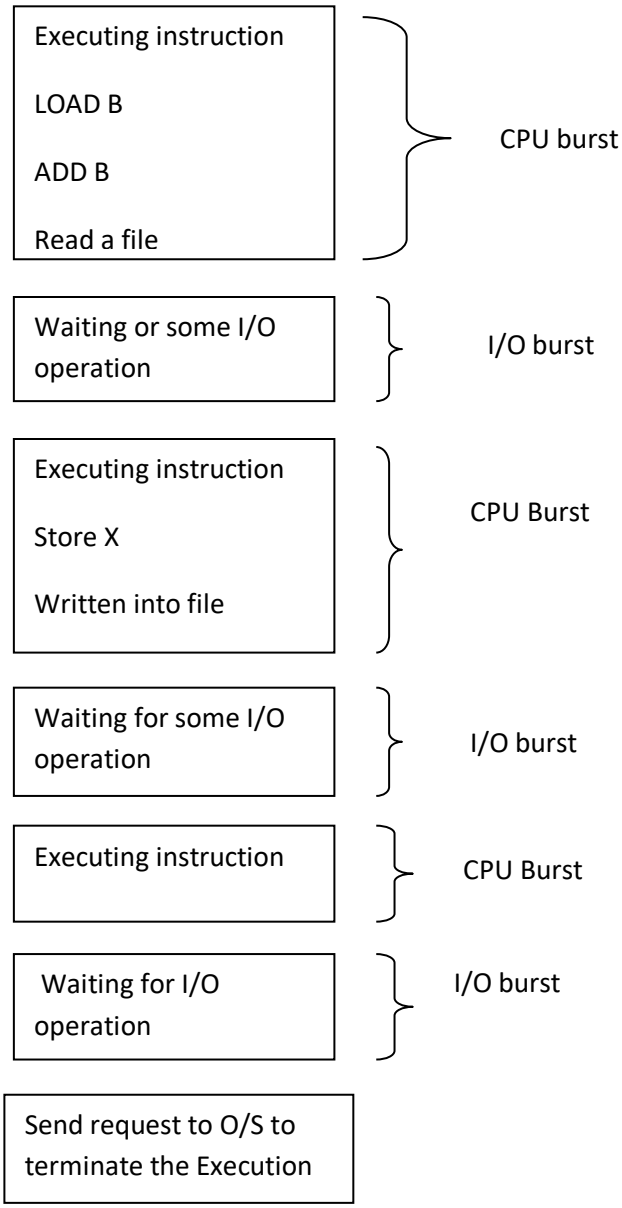
1. Maximum CPU utilization
2. Maximum throughput
3. Minimum Turnaround Time
4. Minimum waiting Time
5. Minimum response time

CPU and I/O cycles:

Process execution consists of cycle of CPU execution and I/O Wait. Process alternate between these two states

The success of CPU scheduling depends on the following property of process:-

- Process execution of cycle of CPU execution and input output wait.
- Process execution begins with CPU burst .That is followed by an I/O burst then another CPU burst and so on.
- Process alternate between these two states.
- But the last CPU burst will end with a system request to terminate execution, rather than with I/O burst.



(Alternating sequence of CPU and I/O burst)

CPU scheduling Algorithm:

1. *First come first scheduling algorithm(FCFS)*
2. *Shortest Job first scheduling algorithm(SJF)*
3. *Priority scheduling algorithm*
4. *Round robin scheduling algorithm*
5. *Multilevel queue scheduling*
6. *Multilevel feedback queue scheduling*

First come first scheduling algorithm(FCFS)

This simplest CPU scheduling algorithm.In this scheme, the process which requests the CPU first, that is allocated to CPU first.

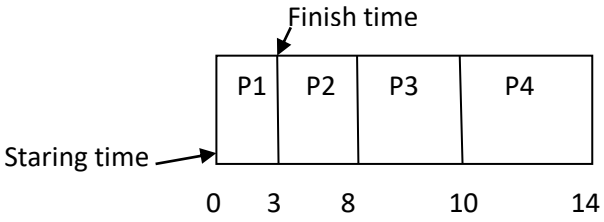
- The implementation of FCFS algorithm is managed with FIFO queue.
- The average waiting time under FCFS policy is quite long.
- FCFS algorithm is non preemptive (means once CPU has been allocated to a process then the process keeps the CPU until the release the CPU either by terminating or requesting I/O)

Problem:

Process	CPU Time
P1	3
P2	5
P3	2
P4	4

Using FCFS Algorithm find the average waiting time and average turnaround Time (TAT) if order is P1, P2, P3, and P4.

Solution: Gantt chart will be:



The waiting time for process P1=0, P2=3, P3=8, P4=10

So ,Avg waiting time =0+3+8+10=21/4=5.25ms

The Turnaround Time (TAT) = W.T+B.T

TAT for process p1=0+3=3

TAT for process p2=3+5=8

So, Average TAT=3+8+10+14=35/4=8.75 ms

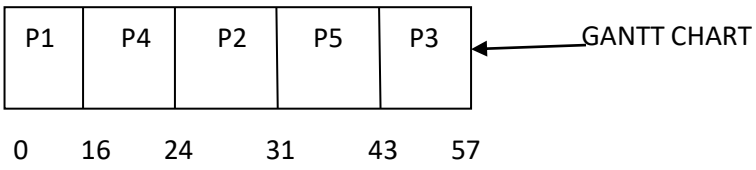
TAT for process p3=8+2=10

TAT for process p4=10+4=14

Problem2: Find the Average waiting time and Average TAT of the following:

Process	Arrival Time	CPU Burst
P1	0	16
P2	5	7
P3	10	14
P4	4	8
P5	7	12

Solution:



Process	Arrival Time (A.T)	CPU Burst (B.T)	Completion Time (C.T)	Turn Around Time (TAT)	Waiting Time (W.T)
P1	0	16	16	16	0
P2	5	7	31	26	19
P3	10	14	57	47	33
P4	4	8	24	20	12
P5	7	12	43	36	24

WT = T.A.T - B.T

TAT = C.T - A.T

Average TAT=?

Average Waiting Time (W.T)= ?

Advantage of FCFS:

- It is used in the batched systems
- It is easy to understand and implement programmatically, using a queue data structure where new process enters through the tail of the queue and scheduler selects process from the head of the queue.
- Real life example of FCFS Scheduling is buying tickets at ticket counter.

  Disadavantages:

  - It is non-preemptive algorithm, which means the process priority doesn't matter

    If a process with very least priority is being executed (ex: **Daily** routine back up) which takes more time and all of a sudden some after high priority process arrives (Ex: interrupt to avoid system crash), the high priority process will have to

wait and hence in this case, the system will crash just because of improper process scheduling

- Not optimal average waiting time
- Resource utilization in parallel is not possible which leads to ***convoy effect*** and hence poor resource (CPU,I/O etc) utilization.

Convoy effect:

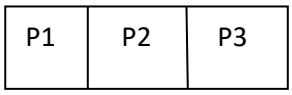
Convoy effect is a situation where many process who need to use a resource for short time and blocked by one process holding that resource for a long time.

- This essentially leads to poor utilization of resource and poor performance.

Ex: Problem on convoy effect:

A.

P.No	A.T	B.T
1	0	20
2	1	2
3	2	2



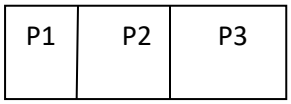
0 20 22 24

P.No	C.T	TAT	W.T
P1	20	20	0
P2	22	21	19
P3	24	22	20

Avg W.T =0+19+20=39/3=13 ms

B.

P.No	A.T	B.T
1	0	2
2	1	2
3	2	20



0 2 4 24

P.No	C.T	TAT	W.T
P1	2	2	0
P2	4	3	1
P3	24	22	2

Avg W.T=0+1+2=3/3=1 ms

- In FCFS, if the 1st process is having large burst time, then it will have drastic effect on average waiting time of all process. This effect is called *convoy effect*.

Q.: Find the Average waiting time and Average TAT of the following:

P.No	A.T	B.T
1	0	4
2	1	3
3	2	1
4	3	2
5	4	5

Find the avg time=? 3.8ms

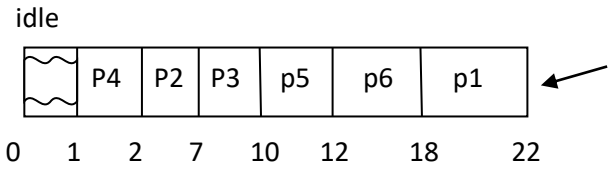
and Avg. TAT=? 6.8

Criteria= Arrival rate

Mode= FCFS Non-Preemptive

Q.

P.No	A.T	B.T
1	6	4
2	2	5
3	3	3
4	1	1
5	4	2
6	5	6



P.No	W.T	B.T	C.T	T.A.T	W.T
1	6	4	22	16	12
2	2	5	7	5	0
3	3	3	10	7	4
4	1	1	2	1	0
5	4	2	12	8	6
6	5	6	18	13	7

Avg W.T=29/6=4.83 ms

- If the arrival time of the processes are same/matching then schedule the process which has lowest process ID

P.No	A.T	B.T
1	8	2
2	3	4
3	7	6
4	10	3
5	2	1
6	3	1

Gantt chart

0 2 3 7 8 14 16 19

WT = T.A.T - B.T

TAT = C.T - A.T

P.No	A.T	B.T	C.T	T.A.T	W.T
1	8	2	16	8	6
2	3	4	7	4	0
3	7	6	14	7	1
4	10	3	19	9	6
5	2	1	3	1	0
6	3	1	8	5	4

Average waiting time= 6+0+1+6+0+4=17/6=2.83 ms

2. Shortest job first scheduling algorithm

Shortest job first scheduling works on the process with shortest burst time or duration first.

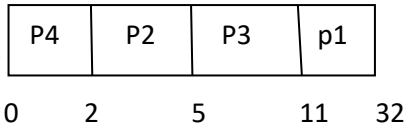
- This is the best approach to minimize waiting time.
- This is used in batch systems
- It is two types: Non- preemptive and preemptive
- To successfully implement it, the burst time /duration time of the processes should be known to processor in advance, which is practically not feasible all the time.
- This scheduling algorithm is optimal if all the job /processes are available at the same time (either arrival time is '0' for all or arrival time is same for all)

Non-preemptive shortest job first

Consider the below processes available in the ready queue for execution, with arrival time as '0' for all and given burst times.

Process	Burst Time
P1	21
P2	3
P3	6
P4	2

Gantt chart



P.no	B.T	C.T	T.A.T	W.T
P1	21	32	43	11
P2	3	5	5	2
P3	6	11	11	5
P4	2	02	2	0

Avg.waiting time = (11+2+5+0)/4=18/4=4.5 ms

For FCFS avg W.T= 18.75 ms

Problem with Non-preemptive SJF

If the arrival time for processes are different, which means all the processes are not available in the ready queue at time '0' and some jobs arrive after some time, in such situation, sometimes process with the short burst time have to wait for the current process's execution to finish, because in non-preemptive SJF scheduling, on arrival of a process with short duration, the existing job/process's execution is not halted /stopped to execute the short job first.

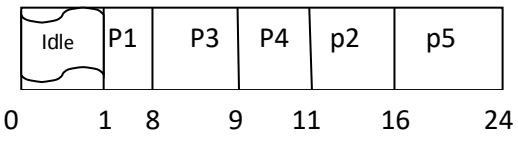
- This leads to the problem of starvation where a shorter process has to wait for a long time until the current longer process gets executed .This happen if shorter jobs keep coming, but this can be solved using the concept of ***aging***. ***Aging*** *is used to gradually increase the priority of a task, based on its waiting time in the ready queue.*

Q.1 Find the avg waiting time and TAT as per following using SJF Non-preemptive

P.No	A.T	B.T
1	1	7
2	2	5
3	3	1
4	4	2
5	5	8

Solution: Criteria: short Burst time and Non preemptive

Gantt chart:



P.No	A.T	B.T	C.T	T.A.T	W.T
1	1	7	8	7	0
2	2	5	16	14	9
3	3	1	9	6	5
4	4	2	11	7	5
5	5	8	24	19	11

T.A.T= C.T-A.T=8-1=7 and so on Avg W.T=(0+9+5+5+11)/5=30/5=6ms

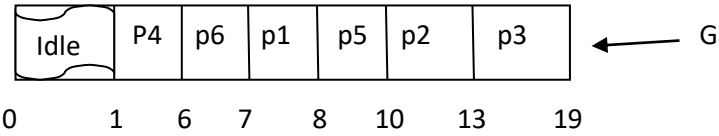
W.T=T.A.T- B.T=7-7=0 and so on Avg TAT=(7+14+6+7+19)/5=10.6 ms

Q2. If the bust times of the process are same /matching then schedule the process which has lowest arrival time

Using Non-preemptive SJF find avg W.T. ?

P.No	A.T	B.T
1	6	1
2	3	3
3	4	6
4	1	5
5	2	2
6	5	1

Solution:



P.No	A.T	B.T	C.T	T.AT	W.T
1	6	1	8	2	1
2	3	3	13	10	7
3	4	6	19	15	9
4	1	5	6	5	0
5	2	2	10	8	6
6	5	1	7	2	1

Avg WT= (1+7+9+6+1+0)/6=24/6=4ms

Avg TAT=(2+10+15+5+5+8+2)/6=42/6=7ms

Q. find out the avg waiting time and avg turn around time of the following process (using SJF non preemptive algorithm)

Process	A.T	B.T
P1	0	8
P2	1	4
P3	2	9
P4	3	5

Avg W.T= 7.75 ms

Preemptive shortest job first

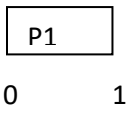
In preemptive shortest job–first scheduling , jobs are put into ready queue as they arrive, but as a process with short burst time arrives ,the existing process is preempted or removed from execution and shorter job is executed first.

Preemptive SJF is also called SRTF (shortest remaining time first)

Q1. Find out Avg WT of following (using SJF preemptive)

P.No	A.T	B.T
1	0	9
2	1	4
3	2	1

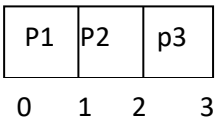
Solution: step1



P.No	A.T	B.T
1	0	~~9~~ 8
2	1	4
3	2	1

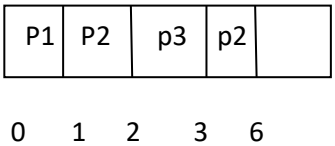
Step2

P.No	A.T	B.T
1	0	~~9~~ 8
2	1	~~4~~ 3
3	2	~~1~~ x



Step3-

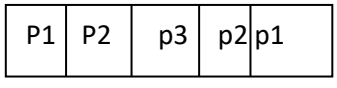
P.No	A.T	B.T
1	0	~~9~~ 8
2	1	~~4~~ ~~3~~ x
3	2	~~1~~ x



Step4.

P.No	A.T	B.T
1	0	~~9~~ ~~8~~ x
2	1	~~4~~ ~~3~~ x
3	2	~~1~~ x

Gantt chart



Step-5

P.No	A.T	B.T	C.T	TAT	WT
1	0	9	14	14	5
2	1	4	6	5	1
3	2	1	3	1	0

Avg w.t= (5+1+0)/3=3 ms

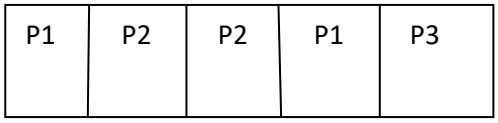
Q2. Find the average waiting time as per following (Using SJF preemptive)

P.No	AT	BT
1	0	9
2	1	4
3	2	9

Solution:

P.No	AT	BT		
1	0	~~9~~	~~8~~	X
2	1	~~4~~	~~3~~	X
3	2	~~9~~		X

Gantt chart →



0 1 2 5 13 22

P.No	AT	BT	CT	TAT	WT
1	0	9	13	13	4
2	1	4	5	4	0
3	2	9	22	20	11

Average WT=(4+0+11)/3=15/3=5ms

Average TAT=(13+4+20)/3=37/3=12.3ms

Priority scheduling Algorithm

In this scheduling a priority is associated with each process and the CPU is allocated to process with highest priority.

- Equal priority process are scheduled in FCFS manner
- Process with highest priority is executed first and so on.
- Priority scheduling may be Non-preemptive or preemptive scheduling

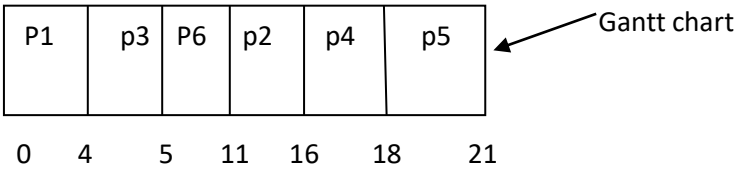
Q.1

P.No	A.T	B.T	priority	
1	0	4	4	
2	1	5	5	
3	2	1	7	High
4	3	2	2	
5	4	3	1	Low
6	5	6	6	

Criteria=priority

Mode=Non-preemptive

Solution:

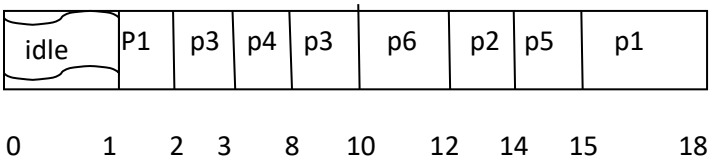


P.No	A.T	B.T	priority	C.T	TAT	WT
1	0	4	4	4	4	0
2	1	5	5	16	15	10
3	2	1	7	5	3	2
4	3	2	2	18	15	13
5	4	3	1	21	17	14
6	5	6	6	11	6	0

Avg W.Tt= 6.5 ms

Q2. Priority based: preemptive:

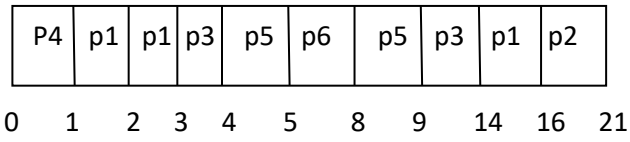
	Priority	P.No	A.T	B.T	C.T
Low	4	1	1	~~4~~ 3	18
	5	2	2	2	14
	7	3	2	~~3~~ 2	10
High	8	4	3	5	08
	5	5	3	1	15
	6	6	4	2	12



Avg wt time= 13+10+50+11+6=45/6=7.5 ms

Q3. Find the Avg waiting time of following using preemptive priority based scheduling

Priority	P.no	A.T	B.T
5	1	1	4
2	2	2	5
6	3	3	6
4	4	0	1
7	5	4	2
8	6	5	3



Avg W.T=?

Disadvantages:

A major problem with priority scheduling algorithm is indefinite blocking or starvation

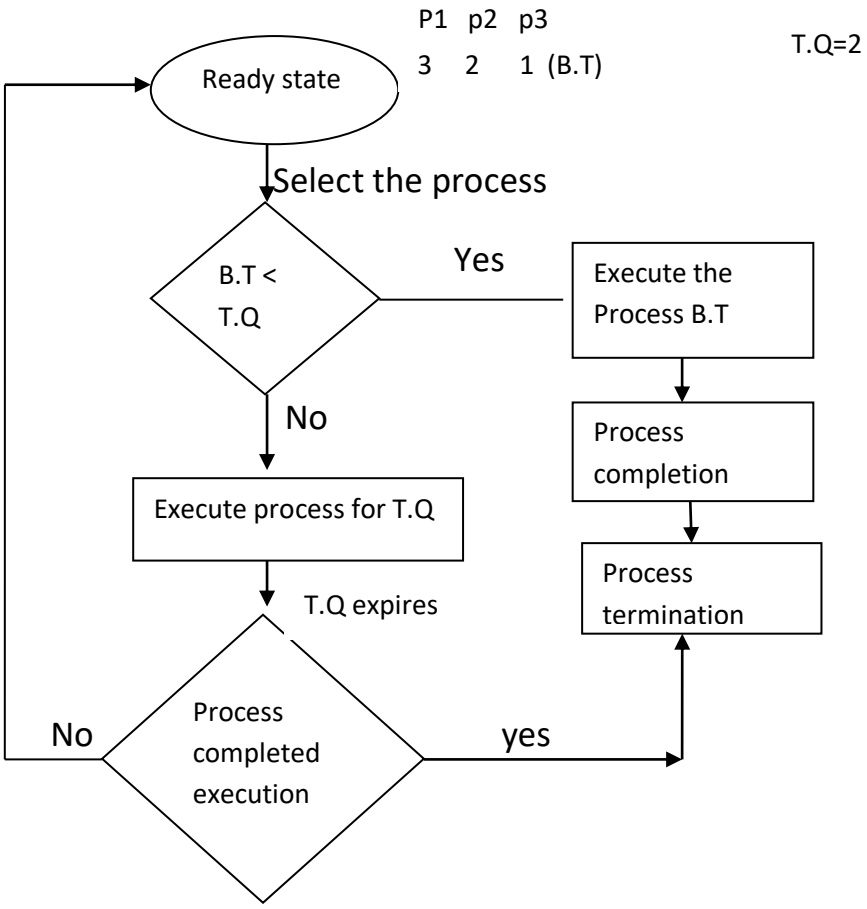
Round Robin Scheduling:

- A fixed time is allotted to each process,called quantum or time slice for execution.
- Once a process is executed for given time period that process is preempted and other process executes for a given time period.
- This type of algorithm is designed only for the time sharing system
- It is similar to FCFS scheduling with preemption condition to switch between process.
- Context switching is used to save states of preempted process.
- It is always preempted

Disadvantages:

- The average waiting time under the round robin policy is quite long

Flowchart of R-R scheduling Algorithm



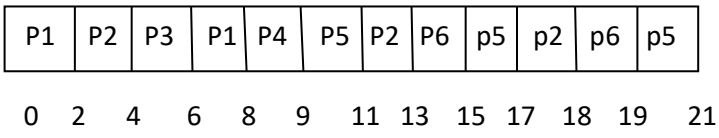
Q1. Find out the Average waiting time of following using R-R scheduling (given TQ=2)

P.no	A.T	B.T
1	0	4
2	1	5
3	2	2
4	3	1
5	4	6
6	5	3

Solution:

Ready queue

P1	P2	P3	P1	P4	P5	P2	P6	p5	p2	p6	p5



P.no	A.T	B.T	C.T	TAT	WT
1	0	4	08	0 8	04
2	1	5	18	17	12
3	2	2	06	04	02
4	3	1	09	06	05
5	4	6	21	17	11
6	5	3	19	14	11

Avg waiting time= 45/6=7.5ms

Avg TAT=66/6=11ms

P.no	A.T	B.T	
1	0	~~4~~ ~~2~~	
2	1	~~5~~ ~~3~~	1
3	2	~~2~~	
4	3	1	
5	4	~~6~~ ~~4~~	2
6	5	~~3~~ 1	

Ready queue

Step 1:

P1 p2 p3 p1

Gantt chart

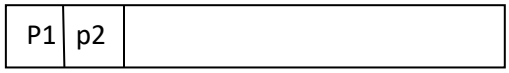
0 2

Ready queue

step2

P1 p2 p3 p1 p4 p5 p2

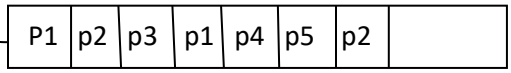
Gantt chart



0 2 4

Step7

P1 p2 p3 p1 p4 p5 p2 p6 p5p2



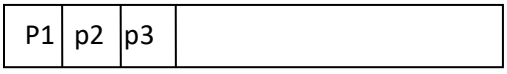
0 2 4 6 8 9 11 13

Ready queue

step3.

P1 p2 p3 p1 p4 p5 p2 p6

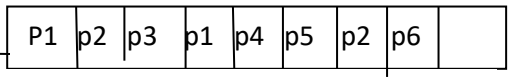
Gantt chart



0 2 4 6

Step-8

P1 p2 p3 p1 p4 p5 p2 p6 p5p2 p6



0 2 4 6 8 9 11 13 15

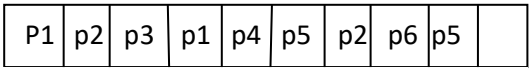
Step 4

P1 p2 p3 p1 p4 p5 p2 p6

0 2 4 6 8

Step9

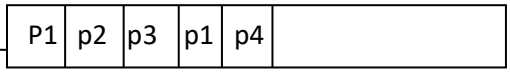
P1 p2 p3 p1 p4 p5 p2 p6 p5p2 p6 p5



0 2 4 6 8 9 11 13 15 17

Step 5

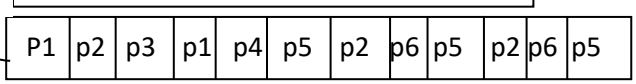
P1 p2 p3 p1 p4 p5 p2 p6



0 2 4 6 8 9

Step-10

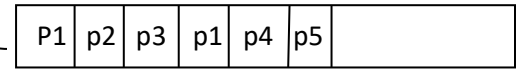
P1 p2 p3 p1 p4 p5 p2 p6 p5p2 p6 p5



0 2 4 6 8 9 11 13 15 17 18 19 21

Step6

P1 p2 p3 p1 p4 p5 p2 p6 p5

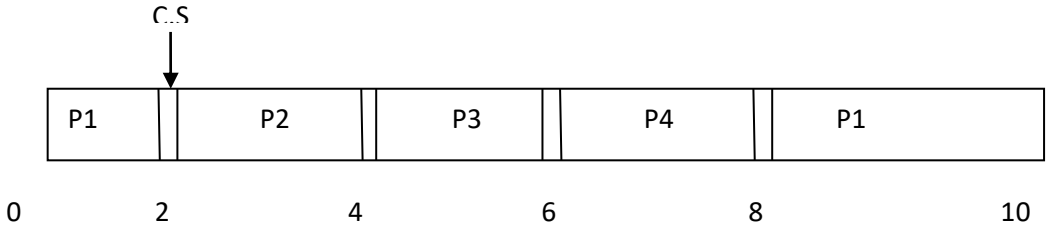


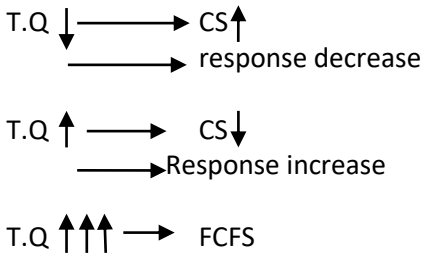
0 2 4 6 8 9 11

Objectives of R-R scheduling:

- If the time quantum is less , then the number context switches will increase and response time will be less
- If the quantum time is large , then number of context switches will decrease and response time will be more
- If the time quantum is very-very large , then this algorithm generates as FCFS algorithm
- Round robin is used to decrease the response time.

EX:





Q. Consider a system which has ‘n ‘process sharing the CPU in round robin Fashion. The context switching time is ‘s’ units. Then what must be the time quantum ‘q’ such that each process is guaranteed to get its turn at the CPU for every ‘t’ seconds of time.

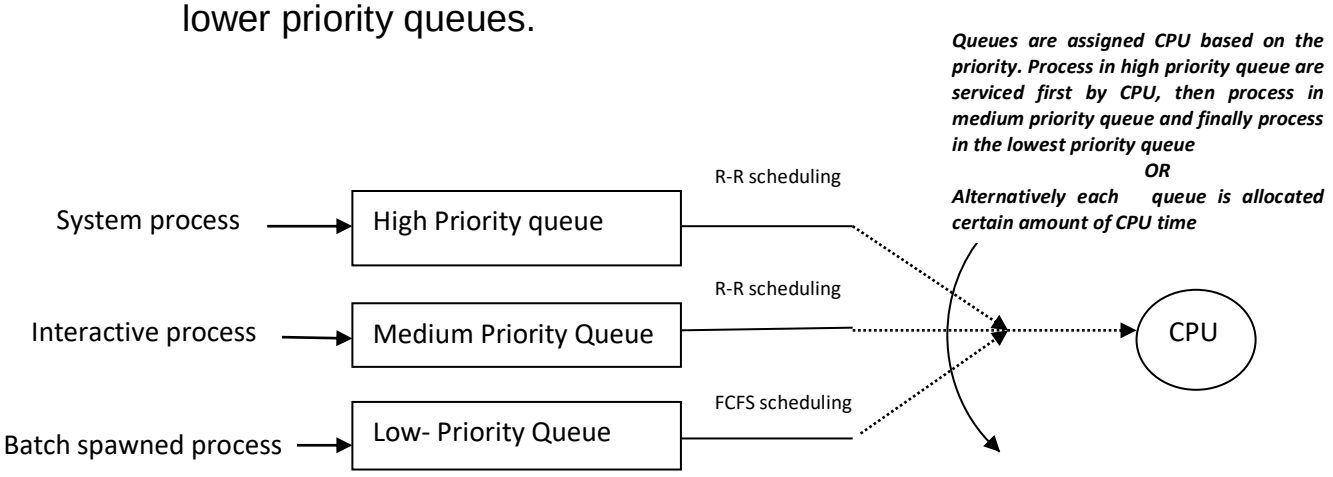
a)q=(t-ns)/n+1 b) q=(t +ns)/n-1 c) q=(t-ns)/n-1 d)q=(t-ns)/n+1

Multilevel Queue Scheduling

In multilevel queue scheduling instead of maintaining single ready queue, separate queues are maintained one for each different type of processes. The different type of processes can be system process (High priority), interactive processes (medium priority) and batch spawned processes (lowest priority).Each queue will have different scheduling policy .Mostly system process and interactive process uses round robin scheduling, while batch job use the FCFS scheduling.

There are two ways to service processes in the queue. They two ways are.

1. The processes in the highest priority queue are serviced first by CPU (according to their scheduling algorithm).When highest priority queue becomes empty; CPU switches to medium priority queue and service all the processes. When medium priority queue is empty, CPU finally switches to the lowest priority queue and services all the processes till that queue becomes empty.
   One problem in this approach is that processes in the low priority queue can be serviced until the queue above it with higher priority becomes empty. This can result in starvation for the processes in the lower priority queues.



2. Alternatively , each queue can be allocated certain amount of the CPU time, the queues can then divide and schedule this allocated CPU time among processes in that queue . The highest priority queue could be allocated 80% of the CPU time, while the lowest priority queue allocated 20 %.

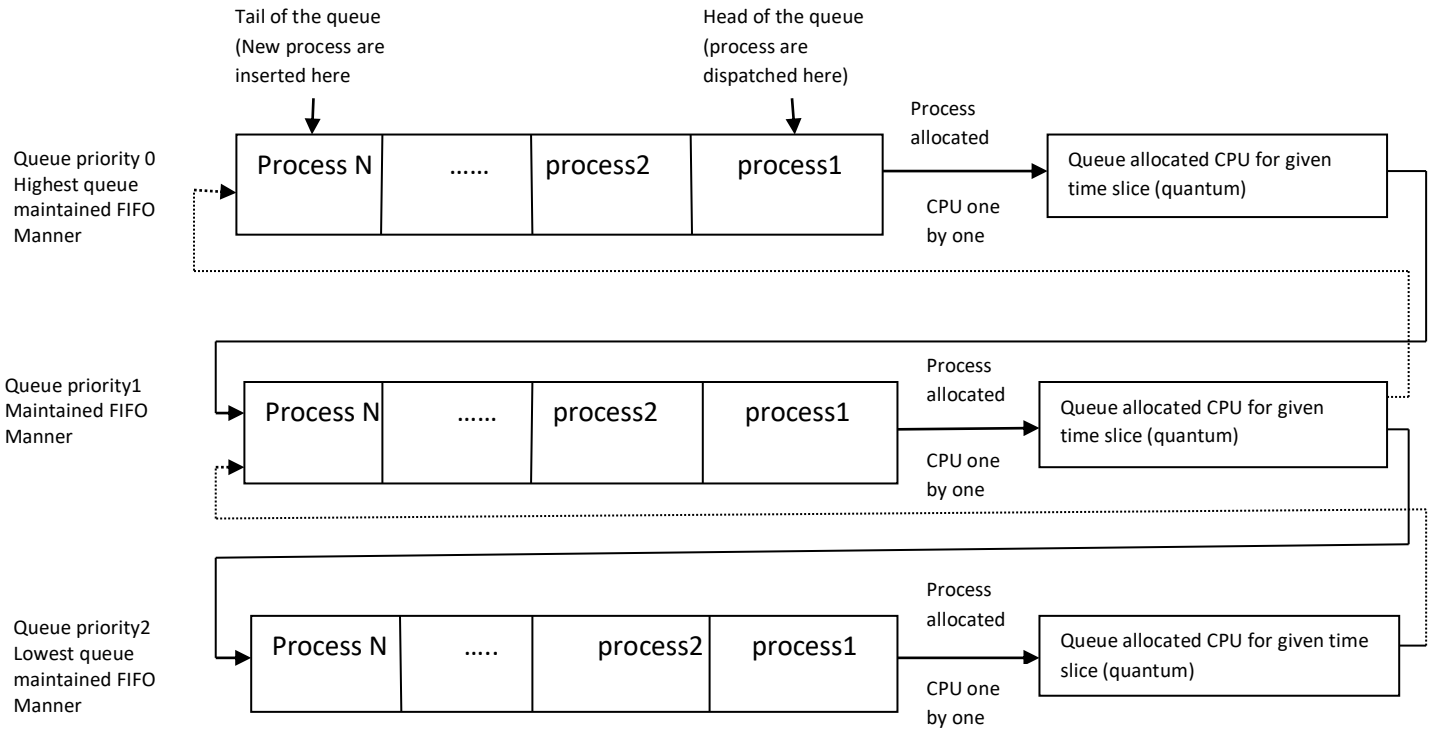
Multilevel Feedback Queue Scheduling:

In a multilevel queue scheduling algorithm the process based on their type are assigned to a specific queue. The processes of one queue are not allowed to move into another queue. Multilevel feedback queue scheduling gives flexibility and allows a process to move from one queue to another. This movement of process depends on its run-time behavior or we can say CPU burst of the process.

In a multilevel queue scheduling algorithm, queues are arranged according their priority, with highest priority queue at top. Each queue is assigned some percentage of CPU time (or time slice).The processes within the queue are processed in FIFO manner. When a new process enters the system, it is inserted at the tail (end) of the top level queue. After some time the process reaches the head of the queue and assigned the CPU. If the process gets completed within the given time-slice of the queue to which it belongs, the process exits the system. If the process is not completed the within the given time- slice of the queue to which belongs, then the process is moved to the tail of lower priority queue. The same process follows in this queue also and if the process is still not completed within the given time –slice of the queue to which it belongs, it is moved to the tail of yet another lower priority queue. In MFQS, a process is given one chance to complete at a given queue level otherwise it is moved down to a lower priority queue.

If a process has a long CPU burst not to be completed within time slice, it is moved to lower priority queue. This approach leaves I/O bound and interactive processes in the higher priority queues. Also, a process that waits too long in a lower priority queue may be moved to a higher priority

Queue .This aging technique prevents lower priority processes from being starved for CPU time



Process that does not get completed within the given time slice of the queue to which it belongs is moved to the tail of lower priority queue

Process that wait too long in a lower priority queue may be moved to the higher priority queue

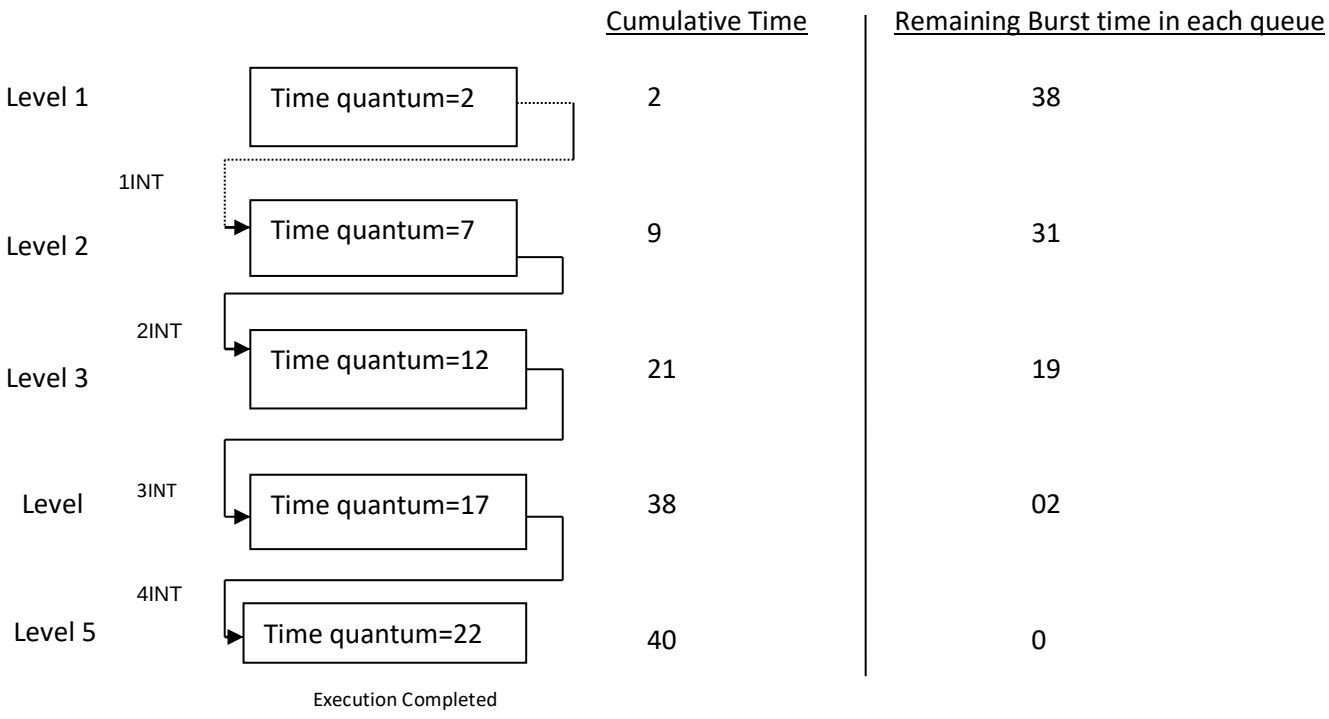
Scheduling parameters used in MFQS

In general some parameters are kept in mind while designing a multilevel feedback queue scheduler. These are:

- How many levels of queues are there?
- What scheduling algorithm is to be followed in queues at different levels
- When to move a process to higher priority queue
- When to move a process to a lower priority queue
- To determine which queue a process will be inserted when that process needs service(each time it returns to the ready state)

Q. *Consider a system which has CPU bound process which requires Burst time of 40 time units. Multilevel feedback queue scheduling is used .The time quantum is 2 units and it will be incremented by 5 units in each level. How many times the process will be interrupted and in which queue, process will complete execution?*

a)4,5 b)5,6 c)3,4 d)5,5



Interprocess Communication

Processes executing concurrently in the operating system may be either independent processes or cooperating processes. A process is ***independent*** if it cannot affect or be affected by the other processes executing in the system. Any process that does not share data with any other process is independent. A process is ***cooperating*** if it can affect or be affected by the other processes executing in the system. Clearly, any process that shares data with other processes is a cooperating process.

There are several reasons for providing an environment that allows process cooperation:

- **Information sharing**. Since several users may be interested in the same piece of information (for instance, a shared file), we must provide an environment to allow concurrent access to such information.
- **Computation speedup**. If we want a particular task to run faster, we must break it into subtasks, each of which will be executing in parallel with the others. Notice that such a speedup can be achieved only if the computer has multiple processing cores.
- **Modularity**. We may want to construct the system in a modular fashion, dividing the system functions into separate processes or threads, as we discussed in Chapter 2.
- **Convenience**. Even an individual user may work on many tasks at the same time. For instance, a user may be editing, listening to music, and compiling in parallel.

Cooperating processes

Cooperating processes require an interprocess communication (IPC) mechanism that will allow them to exchange data and information.

There are two fundamental models of interprocess communication:

(1) Shared memory

(2) Message passing

- In the shared-memory model, a region of memory that is shared by cooperating processes is established.
  Processes can then exchange information by reading and writing data to the shared region.
- In the message passing model, communication takes place by means of messages exchanged between the cooperating processes.

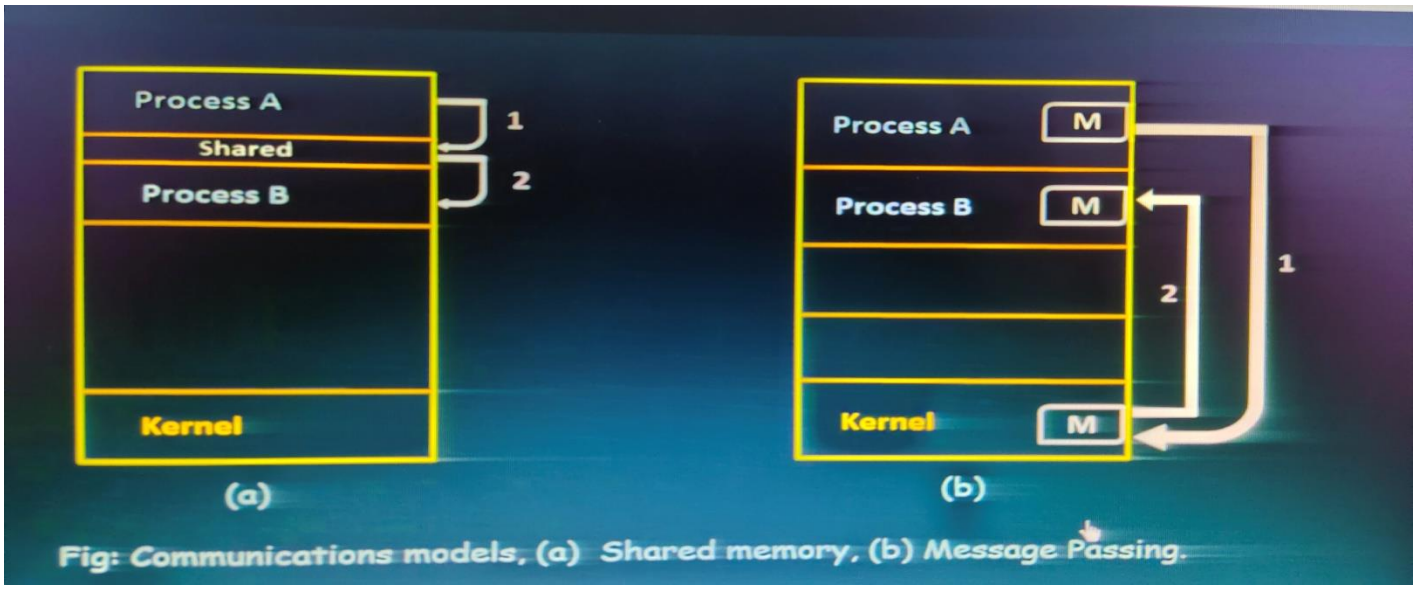
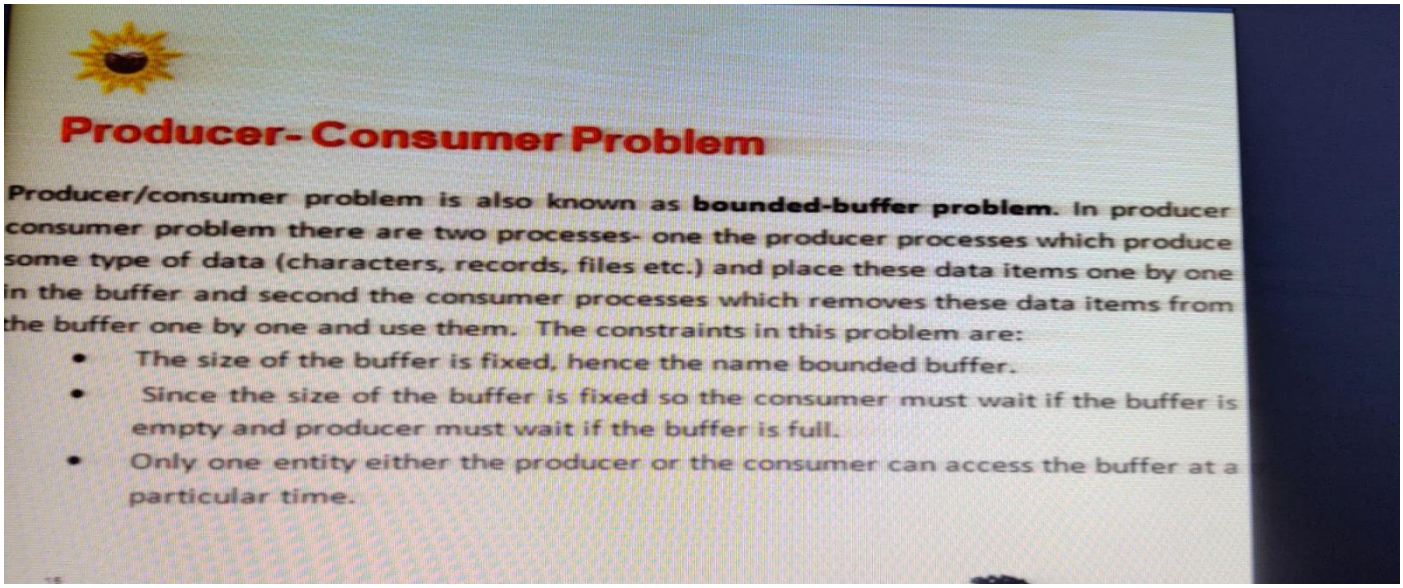


Fig: Communications models, (a) Shared memory, (b) Message Passing.

Shared –memory system:

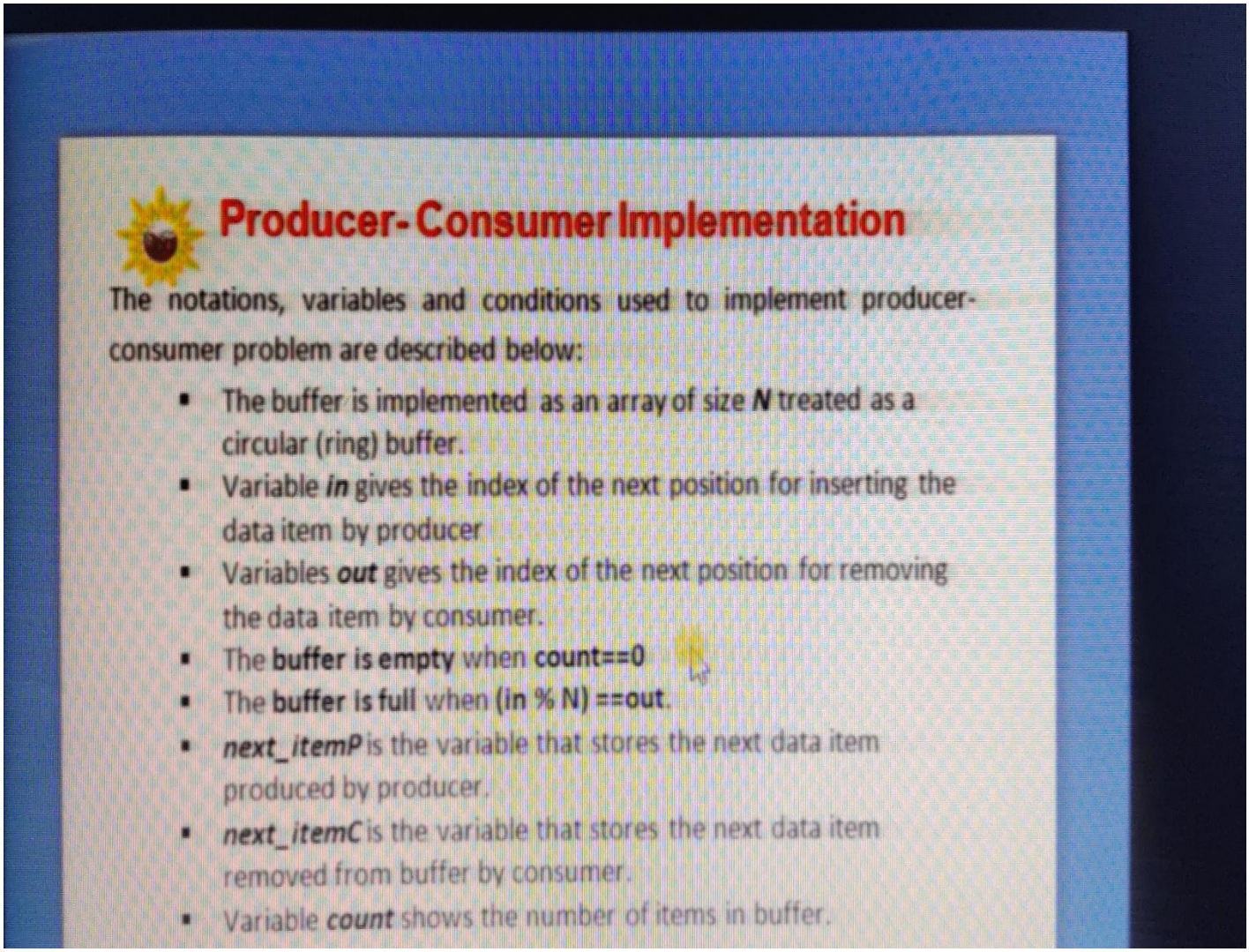
The co-operating process can be well understood with help of producer-consumer problem also known as bounded-buffer problem by shared –memory systems



Producer-Consumer Problem

Producer/consumer problem is also known as **bounded-buffer problem.** In producer consumer problem there are two processes- one the producer processes which produce some type of data (characters, records, files etc.) and place these data items one by one in the buffer and second the consumer processes which removes these data items from the buffer one by one and use them. The constraints in this problem are:

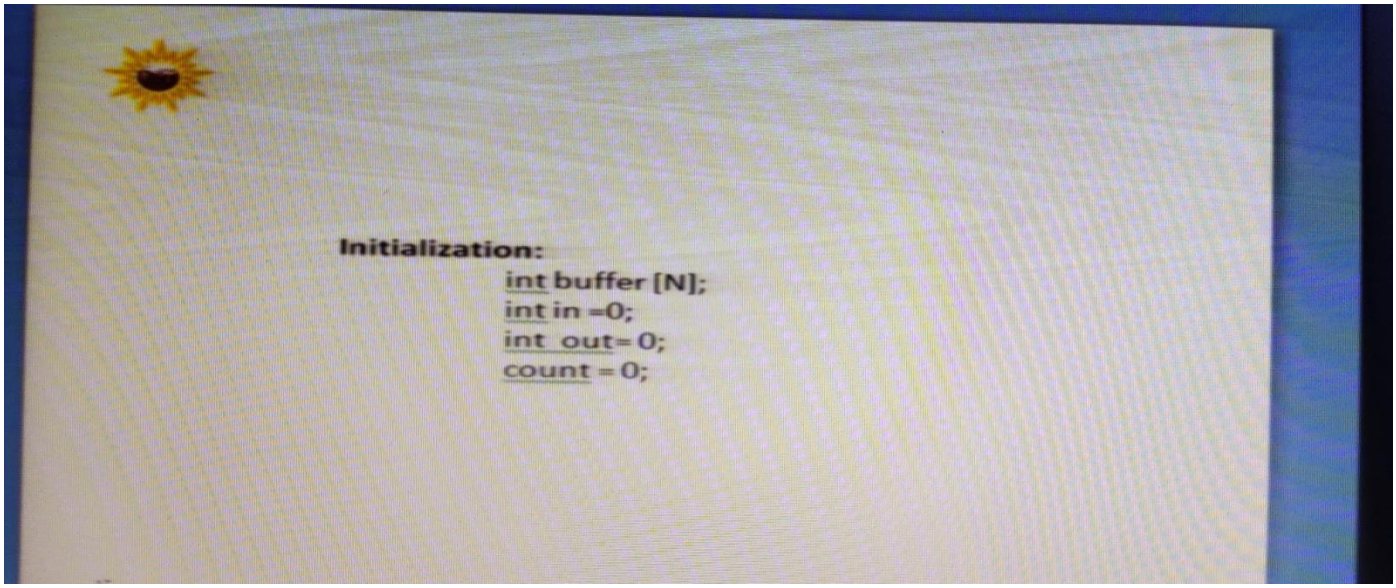
- The size of the buffer is fixed, hence the name bounded buffer.
- Since the size of the buffer is fixed so the consumer must wait if the buffer is empty and producer must wait if the buffer is full.
- Only one entity either the producer or the consumer can access the buffer at a particular time.



Producer-Consumer Implementation

The notations, variables and conditions used to implement producer-consumer problem are described below:

- The buffer is implemented as an array of size ***N*** treated as a circular (ring) buffer.
- Variable ***in*** gives the index of the next position for inserting the data item by producer
- Variables ***out*** gives the index of the next position for removing the data item by consumer.
- The **buffer is empty** when **count==0**
- The **buffer is full** when **(in % N) ==out.**
- ***next_itemP*** is the variable that stores the next data item produced by producer.
- ***next_itemC*** is the variable that stores the next data item removed from buffer by consumer.
- Variable ***count*** shows the number of items in buffer.



Initialization:

```
int buffer [N];
int in =0;
int out= 0;
count = 0;
```

Producer

```
while (1)
        {
            while (count ==N)
            ;                    /* When count ==N, producer do nothing as buffer is full*/

            /* Store data item next_itemP at buffer position pointed by variable in*/
            buffer[in] = next_itemP;

            /* buffer is bounded circular buffer so variable in can range between 0 to N-1 only*/
            in =( in + 1) % N;
            count = count +1;

        }
```

Suppose N= 10

in	count	next_itemP	while (count ==N)	buffer[in] = next_itemP;	in =(in + 1) % N;	count = count +1
0	0	AA	0==10 No	buffer[0] = next_itemP; buffer[0] = AA;	(0+1) %10 =1	1
1	1	BB	1==10 No	buffer[1] = next_itemP; buffer[1] = BB;	(1+1) %10 =2	2
..						
8	8	SS	8==10 No	buffer[8] = next_itemP; buffer[8] = SS;	(8+1) %10 =9	9
9	9	TT	9==10 No	buffer[9] = next_itemP; buffer[9] = TT;	(9+1) %10 =0	10
0	10	YY	10==10 YES	Do Nothing	Do Nothing	

Consumer

```
while(1)
{
        while(count==0)
        ;                    /* When count==0, consumer do nothing as buffer is empty*/

        /* data at buffer position pointed by variable out is stored in variable next_itemC*/
        next_itemC= buffer[out];

        /* buffer is bounded circular buffer so variable in can range between 0 to N-1 only*/
        out =(out + 1) % N;
        count = count -1;

}
```

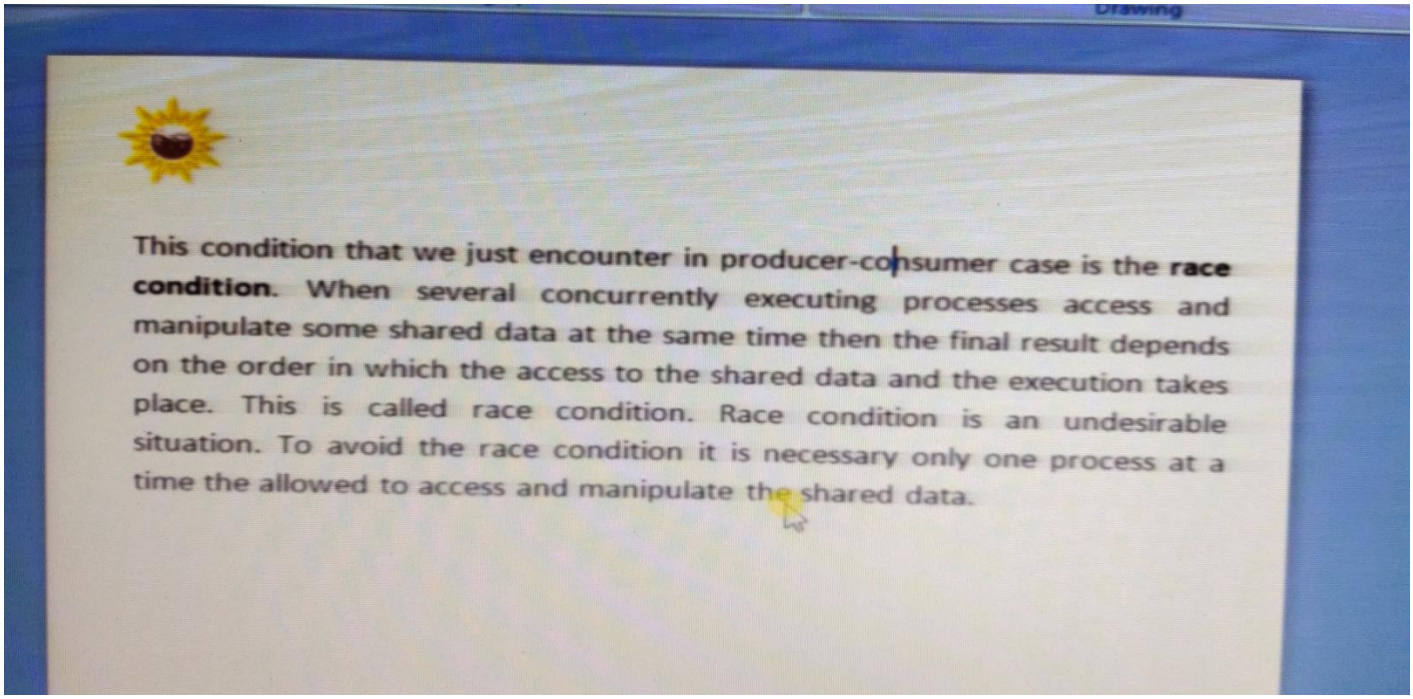
out	count	while (count==0)	next_itemC=; buffer[out]	out =(out + 1) % N;	count = count -1
0	10	10==0 No	next_itemC=; buffer[0]	(0+1) %10 =1	9
1	9	9==0 No	next_itemC=; buffer[1]	(1+1) %10 =2	8
9	1	1==0 No	next_itemC=; buffer[9]	(9+1) %10 =0	0
0	0	0==0 YES	Do Nothing	Do Nothing	

Font | Paragraph | Drawing

When both producer and consumer processes run concurrently they may not give the correct result. Let us suppose that current value of variable counter is 10 and both producer and consumer processes are running currently so in the producer process the value of counter = 10. Now suppose the processor switch from producer to consumer processes and consumer start executing .Here in consumer process the value of counter =10. And after the consumer is executed the new value of counter is equals to 9. Now again the processor switch from consumer to producer. The producer will start from point where it is left previously. So for producer the value of countess is still 10. After the producer execute the value of counter becomes 11. If we run the processes in reverse order then the value of Counter would be 9. This is because we begin with consumer process with value of counter equals to 10.Now the processor switch from consumer to producer. In producer process the value of counter is equals to 10 and after the execution of producer the value of counter becomes 11. Again the processor returns back to consumer the consumer executes and make the value of counter =9 .The correct result of Counter is 10 and will be achieved only when both the processes runs one after the other.

Producer	Consumer
10	
	10
	9
11	
Value of counter =11	

Consumer	Producer
10	
	10
	11
9	
Value of Counter =9	



This condition that we just encounter in producer-consumer case is the **race condition**. When several concurrently executing processes access and manipulate some shared data at the same time then the final result depends on the order in which the access to the shared data and the execution takes place. This is called race condition. Race condition is an undesirable situation. To avoid the race condition it is necessary only one process at a time the allowed to access and manipulate the shared data.