**Q - 1 : EXPLAIN PROCESS AND LIFE CYCLE OF PROCESS.**

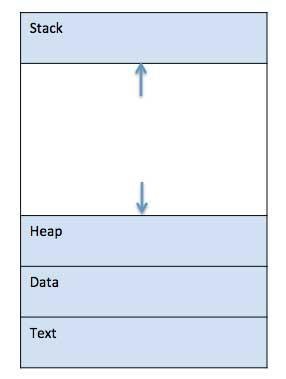
ANS -> Process

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

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

To put it in simple terms, we write our computer programs in a text file and when we execute this program, it becomes a process which performs all the tasks mentioned in the program.

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



**Stack**

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

**Heap**

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

**Text**

This includes the current activity represented by the value of Program Counter and the contents of the processor's registers.

**Data**

This section contains the global and static variables.

**Program**

A program is a piece of code which may be a single line or millions of lines. A computer program is usually written by a computer programmer in a programming language. For example, here is a simple program written in C programming language −

#include <stdio.h>

int main() {

printf("Hello, World! \n");

return 0;

}

A computer program is a collection of instructions that performs a specific task when executed by a computer. When we compare a program with a process, we can conclude that a process is a dynamic instance of a computer program.

A part of a computer program that performs a well-defined task is known as an algorithm. A collection of computer programs, libraries and related data are referred to as a software.

**Process Life Cycle**

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

In general, a process can have one of the following five states at a time.

**Start**

This is the initial state when a process is first started/created.

**Ready**

The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run. Process may come into this state after Start state or while running it by but interrupted by the scheduler to assign CPU to some other process.

**Running**

Once the process has been assigned to a processor by the OS scheduler, the process state is set to running and the processor executes its instructions.

**Waiting**

Process moves into the waiting state if it needs to wait for a resource, such as waiting for user input, or waiting for a file to become available.

**Terminated or Exit**

Once the process finishes its execution, or it is terminated by the operating system, it is moved to the terminated state where it waits to be removed from main memory.

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 −

**Process State**

The current state of the process i.e., whether it is ready, running, waiting, or whatever.

**Process privileges**

This is required to allow/disallow access to system resources.

**Process ID**

Unique identification for each of the process in the operating system.

**Pointer**

A pointer to parent process.

**Program Counter**

Program Counter is a pointer to the address of the next instruction to be executed for this process.

**CPU registers**

Various CPU registers where process need to be stored for execution for running state.

**CPU Scheduling Information**

Process priority and other scheduling information which is required to schedule the process.

**Memory management information**

This includes the information of page table, memory limits, Segment table depending on memory used by the operating system.

**Accounting information**

This includes the amount of CPU used for process execution, time limits, execution ID etc.

**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 −



**Q - 2: DIFFRENTIATE CO-INDEPENDENT AND INDEPENDENT PROCESS.**

ANS -> Processes that executing concurrently in the operating system may be either independent processes or cooperating processes. if a process cannot affect or be affected by the other processes executing in the system then the process is said to be independent.

There are many 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.

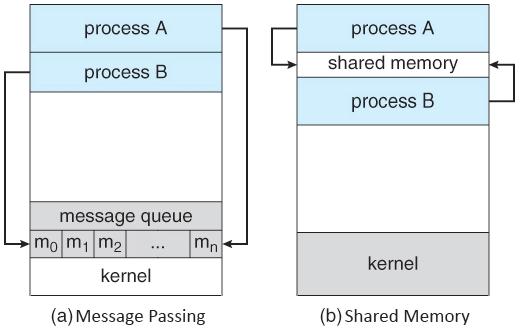
**Computation speedup** - If we want a particular task to run faster, we must break it into subtasks, each of which will be executing in parallel with the others.

**Modularity** - We can construct the system in a modular fashion, dividing the system functions into separate processes or threads.

**Convenience** - Even an individual user can work on many tasks at the same time like a user can be editing, listening to music, and compiling in parallel.

Cooperating processes need an inter-process communication (IPC) mechanism that will allow them to exchange data and information. Two basic models of inter-process communication are there: shared memory and message passing. A region of memory that is shared by cooperating processes is established, in shared-memory model.

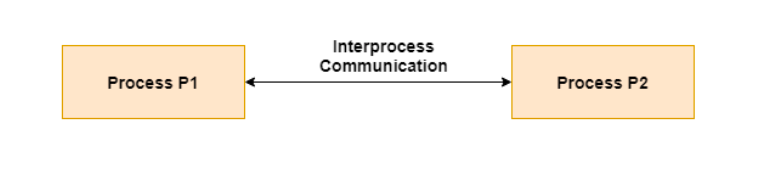
Processes can then exchange their information by reading and writing data to the shared region. In message-passing model, communication takes place by means of messages exchanged between the cooperating processes. The two communications models are contrasted in Figure below. Both of the models which are mentioned, common in operating systems, and many systems implement both. Message passing is needful for exchanging smaller amounts of data, because no conflicts need be avoided. It is also easier to implement in a distributed system than shared memory. Once shared memory is initiated, all accesses are treated as routine memory accesses, and no assistance from the kernel is required. It suffers from cache coherency issues, which arise because shared data migrate among the several caches. Since the number of processing cores on systems increases, it is possible that we will see message passing as the preferred mechanism for IPC.



**Q - 3 : EXPLAIN IPC WITH RELEVANT EXAMPLE.**

ANS -> Interprocess communication is the mechanism provided by the operating system that allows processes to communicate with each other. This communication could involve a process letting another process know that some event has occurred or the transferring of data from one process to another.

A diagram that illustrates interprocess communication is as follows −



Synchronization in Interprocess

**Communication**

Synchronization is a necessary part of interprocess communication. It is either provided by the interprocess control mechanism or handled by the communicating processes. Some of the methods to provide synchronization are as follows −

**Semaphore**

A semaphore is a variable that controls the access to a common resource by multiple processes. The two types of semaphores are binary semaphores and counting semaphores.

**Mutual Exclusion**

Mutual exclusion requires that only one process thread can enter the critical section at a time. This is useful for synchronization and also prevents race conditions.

**Barrier**

A barrier does not allow individual processes to proceed until all the processes reach it. Many parallel languages and collective routines impose barriers.

**Spinlock**

This is a type of lock. The processes trying to acquire this lock wait in a loop while checking if the lock is available or not. This is known as busy waiting because the process is not doing any useful operation even though it is active.

Approaches to Interprocess Communication

The different approaches to implement interprocess communication are given as follows −

**Pipe**

A pipe is a data channel that is unidirectional. Two pipes can be used to create a two-way data channel between two processes. This uses standard input and output methods. Pipes are used in all POSIX systems as well as Windows operating systems.

**Socket**

The socket is the endpoint for sending or receiving data in a network. This is true for data sent between processes on the same computer or data sent between different computers on the same network. Most of the operating systems use sockets for interprocess communication.

**File**

A file is a data record that may be stored on a disk or acquired on demand by a file server. Multiple processes can access a file as required. All operating systems use files for data storage.

**Signal**

Signals are useful in interprocess communication in a limited way. They are system messages that are sent from one process to another. Normally, signals are not used to transfer data but are used for remote commands between processes.

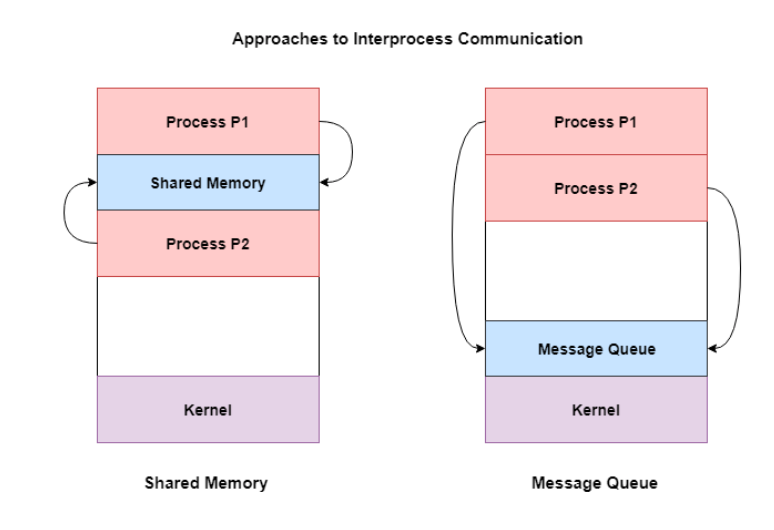
**Shared Memory**

Shared memory is the memory that can be simultaneously accessed by multiple processes. This is done so that the processes can communicate with each other. All POSIX systems, as well as Windows operating systems use shared memory.

**Message Queue**

Multiple processes can read and write data to the message queue without being connected to each other. Messages are stored in the queue until their recipient retrieves them. Message queues are quite useful for interprocess communication and are used by most operating systems.

A diagram that demonstrates message queue and shared memory methods of interprocess communication is as follows −



**Q - 4 : DEFINE FOLLOWING TERMS.**

* **WT**
* **TAT**
* **ET**
* **CT**
* **THROUGH PUT**
* **AT**

**WT - Waiting Time (WT):**

The time spent by a process waiting in the ready queue for getting the CPU.

The time difference b/w Turnaround Time and Burst Time is called Waiting Time.

**Turnaround Time (TAT):**

It is the time interval from the time of submission of a process to the time of the completion of the process.

The difference b/w Completion Time and Arrival Time is called Turnaround Time.

**EXIT TIME (ET):**

Exit time is the time when a process completes its execution and exit from the system.

**Completion Time (CT)**:

This is the time when the process completes its execution.

**Throughput** −

Throughput is the amount of work completed in a unit of time. In other words throughput is the processes executed to number of jobs completed in a unit of time. The scheduling algorithm must look to maximize the number of jobs processed per time unit.

**Arrival Time (AT):**

This is the time when the process has arrived in the ready state.

**Q - 5: FULL FORMS**

* **RR -** Round Robin
* **SJF -** Shortest job next
* **FCFS -** First Come First Serve Scheduling
* **SRTN -** Shortest Remaining Time Next

**Q - 6 : EXPLAIN FCFS, SRTN, SJF, RR ALGORITHM WITH EXAMPLE.**

ANS ->

**FCFS SCHEDULING**

First come first serve (FCFS) scheduling algorithm simply schedules the jobs according to their arrival time. The job which comes first in the ready queue will get the CPU first. The lesser the arrival time of the job, the sooner will the job get the CPU. FCFS scheduling may cause the problem of starvation if the burst time of the first process is the longest among all the jobs.

**Advantages of FCFS**

* Simple
* Easy
* First come, First serve

**Disadvantages of FCFS**

* The scheduling method is non preemptive, the process will run to the completion.
* Due to the non-preemptive nature of the algorithm, the problem of starvation may occur.
* Although it is easy to implement, but it is poor in performance since the average waiting time is higher as compare to other scheduling algorithms.

**Example**

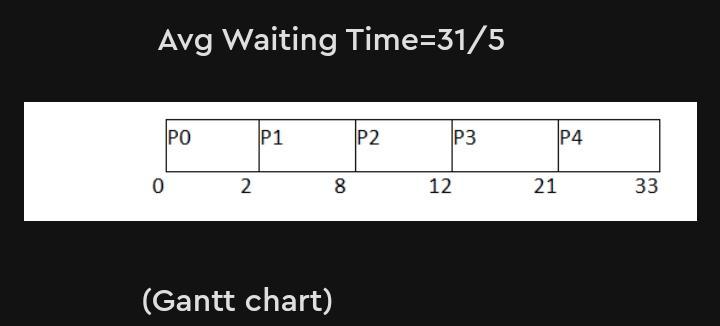
Let's take an example of The FCFS scheduling algorithm. In the Following schedule, there are 5 processes with process ID P0, P1, P2, P3 and P4. P0 arrives at time 0, P1 at time 1, P2 at time 2, P3 arrives at time 3 and Process P4 arrives at time 4 in the ready queue. The processes and their respective Arrival and Burst time are given in the following table.

The Turnaround time and the waiting time are calculated by using the following formula.

**Turn Around Time = Completion Time - Arrival Time**

**Waiting Time = Turnaround time - Burst Time**

| **PROCESS ID** | **ARRIVAL TIME** | **BURST TIME** | **COMPLETION TIME** | **TURN AROUND TIME** | **WAITING TIME** |
| --- | --- | --- | --- | --- | --- |
| **0** | **0** | **2** | **2** | **2** | **0** |
| **1** | **1** | **6** | **8** | **7** | **1** |
| **2** | **2** | **4** | **12** | **10** | **6** |
| **3** | **3** | **9** | **21** | **18** | **9** |
| **4** | **6** | **12** | **33** | **29** | **17** |

****

**Shortest Remaining Time First (SRTF) Scheduling Algorithm**

This Algorithm is the preemptive version of SJF scheduling. In SRTF, the execution of the process can be stopped after certain amount of time. At the arrival of every process, the short term scheduler schedules the process with the least remaining burst time among the list of available processes and the running process.

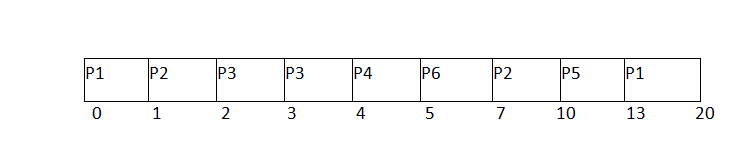
Once all the processes are available in the ready queue, No preemption will be done and the algorithm will work as SJF scheduling. The context of the process is saved in the Process Control Block when the process is removed from the execution and the next process is scheduled. This PCB is accessed on the next execution of this process.

**Example**

In this Example, there are five jobs P1, P2, P3, P4, P5 and P6. Their arrival time and burst time are given below in the table.

| **PROCESS ID** | **ARRIVAL TIME** | **BURST TIME** | **COMPLETION TIME** | **TURN AROUND TIME** | **WAITING TIME** | **RESPONSE TIME** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 8 | 20 | 20 | 12 | 0 |
| 2 | 1 | 4 | 10 | 9 | 5 | 1 |
| 3 | 2 | 2 | 4 | 2 | 0 | 2 |
| 4 | 3 | 1 | 5 | 2 | 1 | 4 |
| 5 | 4 | 3 | 13 | 9 | 6 | 10 |
| 6 | 5 | 2 | 7 | 2 | 0 | 5 |

Avg Waiting Time = 24/6



* Since, at time 0, the only available process is P1 with CPU burst time 8. This is the only available process in the list therefore it is scheduled.
* The next process arrives at time unit 1. Since the algorithm we are using is SRTF which is a preemptive one, the current execution is stopped and the scheduler checks for the process with the least burst time.
* Till now, there are two processes available in the ready queue. The OS has executed P1 for one unit of time till now; the remaining burst time of P1 is 7 units. The burst time of Process P2 is 4 units. Hence Process P2 is scheduled on the CPU according to the algorithm.
* The next process P3 arrives at time unit 2. At this time, the execution of process P3 is stopped and the process with the least remaining burst time is searched. Since the process P3 has 2 unit of burst time hence it will be given priority over others.
* The Next Process P4 arrives at time unit 3. At this arrival, the scheduler will stop the execution of P4 and check which process is having least burst time among the available processes (P1, P2, P3 and P4). P1 and P2 are having the remaining burst time 7 units and 3 units respectively.
* P3 and P4 are having the remaining burst time 1 unit each. Since, both are equal hence the scheduling will be done according to their arrival time. P3 arrives earlier than P4 and therefore it will be scheduled again.
* The Next Process P5 arrives at time unit 4. Till this time, the Process P3 has completed its execution and it is no more in the list. The scheduler will compare the remaining burst time of all the available processes. Since the burst time of process P4 is 1 which is least among all hence this will be scheduled.
* The Next Process P6 arrives at time unit 5, till this time, the Process P4 has completed its execution. We have 4 available processes till now, that are P1 (7), P2 (3), P5 (3) and P6 (2). The Burst time of P6 is the least among all hence P6 is scheduled. Since, now, all the processes are available hence the algorithm will now work same as SJF. P6 will be executed till its completion and then the process with the least remaining time will be scheduled.

**SJF**

[**https://www.javatpoint.com/os-sjf-scheduling**](https://www.javatpoint.com/os-sjf-scheduling)

**RR**

[**https://www.geeksforgeeks.org/program-round-robin-scheduling-set-1/**](https://www.geeksforgeeks.org/program-round-robin-scheduling-set-1/)