**Unit 2 : Process Management**

* **Process Concept**

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 −

|  |  |
| --- | --- |
| **S.N.** | **Component & Description** |
| 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 section :**  This includes the current activity represented by the value of Program Counter and the contents of the processor's registers. |
| 4 | **Data section :**  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;

}

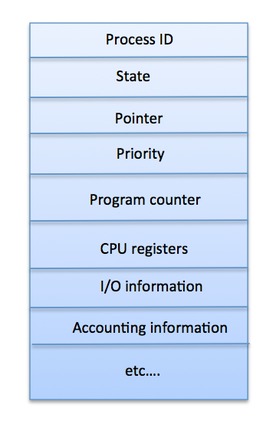
A computer program is a collection of instructions that performs a specific task when executed by a computer. When we compare a program with a process, we can conclude that a process is a dynamic instance of a computer program.

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 Control Block (PCB)**

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

|  |  |
| --- | --- |
| **S.N.** | **Information & Description** |
| 1 | **Process State:**  The current state of the process i.e., whether it is ready, running, waiting, or whatever. |
| 2 | **Process privileges:**  This is required to allow/disallow access to system resources. |
| 3 | **Process ID:**  Unique identification for each of the process in the operating system. |
| 4 | **Pointer:**  A pointer to parent process. |
| 5 | **Program Counter:**  Program Counter is a pointer to the address of the next instruction to be executed for this process. |
| 6 | **CPU registers :**  Various CPU registers where process need to be stored for execution for running state. |
| 7 | **CPU Scheduling Information:**  Process priority and other scheduling information which is required to schedule the process. |
| 8 | **Memory management information:**  This includes the information of page table, memory limits, and Segment table depending on memory used by the operating system. |
| 9 | **Accounting information:**  This includes the amount of CPU used for process execution, time limits, execution ID etc. |
| 10 | **I/O status information :**  This includes a list of I/O devices allocated to the process. |

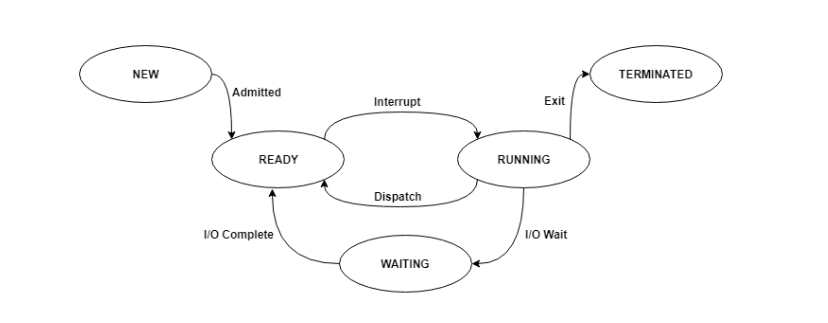
The architecture of a PCB is completely dependent on Operating System and may contain different information in different operating systems. Here is a simplified diagram of a PCB −

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

* **Process States and its Transitions**

## *Process States*

A process changes its state as it executes. This state partially depends on the current activity of a process. The different states that a process is in during its execution are explained using the following diagram −



*The following are the states −*

* **New** - The process is in the new state when it has just been created.
* **Ready** - The process is waiting to be assigned the processor by the short-term scheduler.
* **Running** - The process instructions are being executed by the processor.
* **Waiting** - The process is waiting for some event such as I/O to occur.
* **Terminated -**The process has completed its execution.

## *Process Transitions*

Logically, the first two state are similar. In both case the process is willing to run, but in the ready state there is no CPU temporarily available for it.

1. **Running to ready state:**

* A process in the running state has all of the resources that it needs for further execution, including a processor.
* The long term scheduler picks up a new process from second memory and loads it into the main memory when there are sufficient resources available.
* The process is now in ready state, waiting for its execution.

1. **waiting to ready:**

* Process waiting for some event such as completion of I/O operation, synchronization signal, etc.
* A process moves from waiting state to ready state if the event the  
  process has been waiting for, occurs.
* The process is now ready for execution.

1. **Running to waiting:**

* The process in the main memory that is waiting for some event.
* A process is put in the waiting state if it must wait for some event. For example, the process may request some resources or memory which might not be available.
* The process may be waiting for an I/O operation or it may be waiting for some other process to finish before it can continue execution.

1. **blocked to ready:**

* The process is in secondary memory but not yet ready for execution.
* The process moves from Blocked to Ready state if the event, the process has been waiting for occurs.

1. **Running to terminated:**

* The process has finished execution.
* The OS moves a process from running state to terminated state if the process finishes execution or if it aborts.
* Whenever the execution of a process is completed in running state, it will exit to terminate state, which is the completion of process.
* **Context switch**

The Context switching is a technique or method used by the operating system to switch a process from one state to another to execute its function using CPUs in the system. When switching perform in the system, it stores the old running process's status in the form of registers and assigns the [CPU](https://www.javatpoint.com/cpu-full-form) to a new process to execute its tasks. While a new process is running in the system, the previous process must wait in a ready queue. The execution of the old process starts at that point where another process stopped it. It defines the characteristics of a multitasking operating system in which multiple processes shared the same [CPU](https://www.javatpoint.com/central-processing-unit) to perform multiple tasks without the need for additional processors in the system.

## *The need for Context switching*

A context switching helps to share a single CPU across all processes to complete its execution and store the system's tasks status. When the process reloads in the system, the execution of the process starts at the same point where there is conflicting.

Following are the reasons that describe the need for context switching in the Operating system.

1. The switching of one process to another process is not directly in the system. A context switching helps the operating system that switches between the multiple processes to use the CPU's resource to accomplish its tasks and store its context. We can resume the service of the process at the same point later. If we do not store the currently running process's data or context, the stored data may be lost while switching between processes.
2. If a high priority process falls into the ready queue, the currently running process will be shut down or stopped by a high priority process to complete its tasks in the system.
3. If any running process requires I/O resources in the system, the current process will be switched by another process to use the CPUs. And when the I/O requirement is met, the old process goes into a ready state to wait for its execution in the CPU. Context switching stores the state of the process to resume its tasks in an operating system. Otherwise, the process needs to restart its execution from the initials level.
4. If any interrupts occur while running a process in the operating system, the process status is saved as registers using context switching. After resolving the interrupts, the process switches from a wait state to a ready state to resume its execution at the same point later, where the operating system interrupted occurs.
5. A context switching allows a single CPU to handle multiple process requests simultaneously without the need for any additional processors.

### ***Example of Context Switching***

Suppose that multiple processes are stored in a Process Control Block (PCB). One process is running state to execute its task with the use of CPUs. As the process is running, another process arrives in the ready queue, which has a high priority of completing its task using CPU. Here we used context switching that switches the current process with the new process requiring the CPU to finish its tasks. While switching the process, a context switch saves the status of the old process in registers. When the process reloads into the CPU, it starts the execution of the process when the new process stops the old process. If we do not save the state of the process, we have to start its execution at the initial level. In this way, context switching helps the operating system to switch between the processes, store or reload the process when it requires executing its tasks.

### ***Context switching triggers***

Following are the three types of context switching triggers as follows.

1. **Interrupts**
2. **Multitasking**
3. **Kernel/User switch**

**1. Interrupts**: A CPU requests for the data to read from a disk, and if there are any interrupts, the context switching automatic switches a part of the hardware that requires less time to handle the interrupts.

**2. Multitasking**: A context switching is the characteristic of multitasking that allows the process to be switched from the CPU so that another process can be run. When switching the process, the old state is saved to resume the process's execution at the same point in the system.

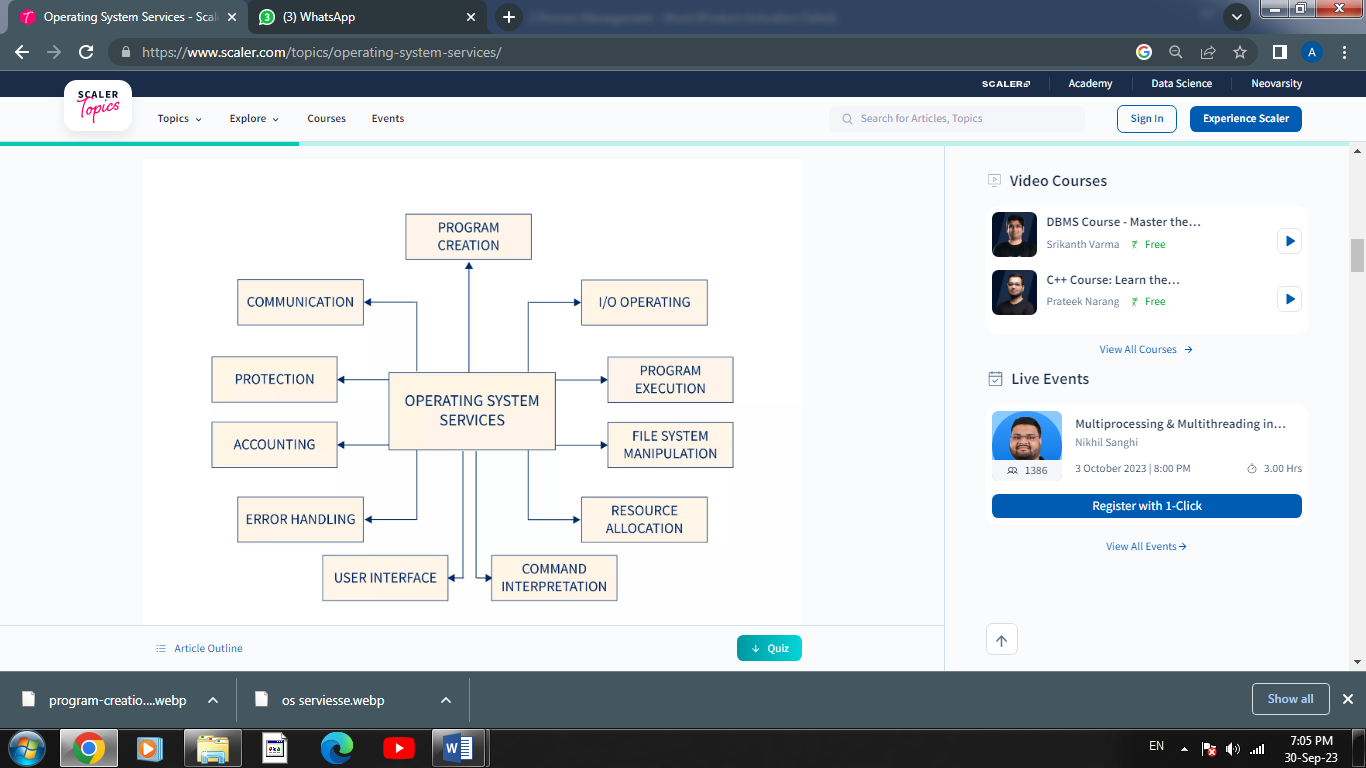
**3. Kernel/User Switch**: It is used in the operating systems when switching between the user mode, and the kernel/user mode is performed.

* **Services for Process management OS**

The operating system provides services to both the user and the programs running in the system. The operating system itself is a program that provides an environment to run other programs in the system. To the user, the operating system provides various services to run multiple user processes in the system. Operating system services such as process management, memory management, and resource allocation management are provided by the operating system.

***The operating system works as a resource manager for the system. The various services of the operating system for efficient working of the system are:***

* Program execution
* Control Input/output devices
* Program creation
* Error Detection and Response
* Accounting
* Security and Protection
* File Management
* Communication
* User Interface
* Resource allocation
* Command interpretation



### **Program execution**

The operating system loads the program into the memory and takes care of the memory allocation for the program. Program execution is one of the operating system services which also ensures that the program that is started can also end its execution either normally or forcefully.

After the program execution, the operating system also takes care of process synchronization, inter-process communication, and deadlock handling.

### **Control Input/output devices**

The programs running in the system need input and output devices access for performing the input/output operations. The access to the input and output devices is given by the operating system to the program for I/O operations.

I/O operations mean writing or reading operations performed over any file or any input/output device.

### **Program creation**

In order to create, modify and debug programs the operating system provides tools like editors and debuggers to make the task of programmers easy.

### **Error Detection and Response**.

The error can occur in hardware or software in the system. Illegal memory access, memory failure, undefined operation like division by zero etc. are some of the errors that can occur in the system. Operating system detects these errors in the system and takes appropriate actions to handle such errors.

The error can occur in the system in the following devices or programs -

* Network connection error, loose connection of I/O devices, and restricted network calls are some of the error that occur in input/output devices.
* The program runned by the user in the system can also cause errors such as accessing illegal memory, undefined operations such as division by zero, excess use of CPU by a program etc.
* Hardware problems such as trying to insert a program in a already full memory, illegal memory access etc.

### **Accounting**

The operating system keeps track of all the data of performance parameters and response time of the system in order to make it more robust. This data is used to improve the performance of the operating system and minimize the response time of the system.

### **Security and Protection**

If a user downloads a program from the internet there are chances that the program can contain malicious code which can affect other programs in the system. The operating system takes care that such a program is checked for any malicious code before downloading it to the system.

The operating system provides safety and security to the user running their programs in the system. the operating system protects the system from unauthorized access and malicious code which are mostly intruded from outside the system.

### **File Management**

File management is one of the operating system services that handles the memory management for the programs running in the system. The files in the system are stored in secondary storage devices like magnetic tape, magnetic disk, optical disk, etc. Each data storage device has its own speed, data transfer rate, data access methods, etc.

The operating system knows the information of all the types of different files and properties of different storage devices and ensures the proper management and safety of the files stored in the secondary storage devices.

### **Communication**

The processes running in the system need to communicate with each other and also the computers or systems connected over a network need to exchange data with each other over a secure network. operating system uses message passing and shared memory to keep communication effective and safe.

Shared memory is a memory segment in the address space that is used by the processes running in the system to do read/write operations without the interference of the operating system. Shared memory is very efficient for inter-process communication.

### **User Interface**

The user interacts with the system either by command-line interface or Graphical user interface. There is also a batch-based interface that uses commands to execute the files.

A graphical user interface is a more user-friendly way to interact with the system. The GUI provides icons, widgets, texts, labels, and text navigation. The user can easily interact with these icons and widgets with just a click of a mouse or keyboard.

### **Resource allocation**

The processes running in the system require resources to complete their execution. When multiple processes are running in the system concurrently then the resources should be equally allocated to every process running in the system.

The operating system uses CPU scheduling to allocate resources effectively among the processes ensuring better utilization of the CPU.

The resources used by the processes can be CPU cycles, primary memory storage, file storage, and I/O devices.

### **Command Interpretation**

The user interacts with the system through commands and the operating system interprets these commands and inputs and provides appropriate outputs accordingly.

If the interpreter is separate from the kernel then the user can modify the interpreter and prevent any unauthorized access to the system.

* **Scheduling and Types of Schedulers**

**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.

Process scheduling is an essential part of a Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

* **Categories of Scheduling**

There are two categories of scheduling:

1. **Non-preemptive:** Here the resource can’t be taken from a process until the process completes execution. The switching of resources occurs when the running process terminates and moves to a waiting state.
2. **Preemptive**: Here the OS allocates the resources to a process for a fixed amount of time. During resource allocation, the process switches from running state to ready state or from waiting state to ready state. This switching occurs as the CPU may give priority to other processes and replace the process with higher priority with the running process.

* **Process Scheduling Queues**

The OS maintains all Process Control Blocks (PCBs) in Process Scheduling Queues. The OS maintains a separate queue for each of the process states and PCBs of all processes in the same execution state are placed in the same queue. When the state of a process is changed, its PCB is unlinked from its current queue and moved to its new state queue.

The Operating System maintains the following important process scheduling queues −

* **Job queue** − This queue keeps all the processes in the system.
* **Ready queue** − This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.
* **Device queues** − The processes which are blocked due to unavailability of an I/O device constitute this queue.



The OS can use different policies to manage each queue (FIFO, Round Robin, Priority, etc.). 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; in the above diagram, it has been merged with the CPU.

**Types of Schedulers:**

Schedulers are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types −

1. **Long-Term Scheduler**
2. **Short-Term Scheduler**
3. **Medium-Term Scheduler**
4. **Long Term Scheduler**

It is also called a job scheduler. A long-term scheduler determines which programs are admitted to the system for processing. It selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling.

The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound. It also controls the degree of multiprogramming. If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system.

On some systems, the long-term scheduler may not be available or minimal. Time-sharing operating systems have no long term scheduler. When a process changes the state from new to ready, then there is use of long-term scheduler.

1. **Short Term Scheduler**

It is also called as CPU scheduler. Its main objective is to increase system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them.

Short-term schedulers, also known as dispatchers, make the decision of which process to execute next. Short-term schedulers are faster than long-term schedulers.

1. **Medium Term Scheduler**

Medium-term scheduling is a part of swapping. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium-term scheduler is in-charge of handling the swapped out-processes.

A running process may become suspended if it makes an I/O request. A suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage. This process is called swapping, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix.

* ***Comparison among Schedulers***

|  |  |  |  |
| --- | --- | --- | --- |
| **S.N.** | **Long-Term Scheduler** | **Short-Term Scheduler** | **Medium-Term Scheduler** |
| 1 | It is a job scheduler | It is a CPU scheduler | It is a process swapping scheduler. |
| 2 | Speed is lesser than short term scheduler | 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 control over degree of multiprogramming | It reduces the degree of multiprogramming. |
| 4 | It is almost absent or minimal in time sharing system | It is also minimal in time sharing system | It is a part of Time sharing systems. |
| 5 | It selects processes from pool 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. |

* **Scheduling Algorithms**

There are various algorithms which are used by the Operating System to schedule the processes on the processor in an efficient way**.**

## FCFS Scheduling Mathematical Examples

In CPU-scheduling problems some terms are used while solving the problems, so for conceptual purpose the terms are discussed as follows −

* **Arrival time (AT**) − Arrival time is the time at which the process arrives in ready queue.
* **Burst time (BT) or CPU time** of the process − Burst time is the unit of time in which a particular process completes its execution.
* **Completion time (CT)** − Completion time is the time at which the process has been terminated.
* **Turn-around time (TAT**) − The total time from arrival time to completion time is known as turn-around time. TAT can be written as,

**Turn-around time (TAT) = Completion time (CT) – Arrival time (AT)  or , TAT = Burst time (BT) + Waiting time (WT)**

* **Waiting time (WT)** − Waiting time is the time at which the process waits for its allocation while the previous process is in the CPU for execution. WT is written as,

**Waiting time (WT) = Turn-around time (TAT) – Burst time (BT)**

|  |
| --- |
| * **Response time (RT**) − Response time is the time at which CPU has been allocated to a particular process first time.   In case of non-preemptive scheduling, generally Waiting time and Response time is same.   * **Gantt chart −** Gantt chart is a visualization which helps to scheduling and managing particular tasks in a project. It is used while solving scheduling problems, for a concept of how the processes are being allocated in different algorithms. |

There are 7 popular process scheduling algorithms which we are going to discuss in this chapter –

1. **First-Come, First-Served (FCFS) Scheduling**
2. **Shortest -Job-**  **first (SJF) Scheduling**
3. **Priority Scheduling**
4. **Shortest Remaining Time**
5. **Round Robin(RR) Scheduling**
6. **Multiple-Level Queues Scheduling**
7. **Multiple-Level Queues with feedback Scheduling**
8. **First-Come First-Served (FCFS) Scheduling (*non-preemptive*)**

FCFS is considered as simplest CPU-scheduling algorithm. In FCFS algorithm, the process that requests the CPU first is allocated in the CPU first. The implementation of FCFS algorithm is managed with FIFO (First in first out) queue. FCFS scheduling is non-preemptive. Nonpreemptive means, once the CPU has been allocated to a process, that process keeps the CPU until it executes a work or job or task and releases the CPU, either by requesting I/O.

## Real Life Example Of FCFS Scheduling

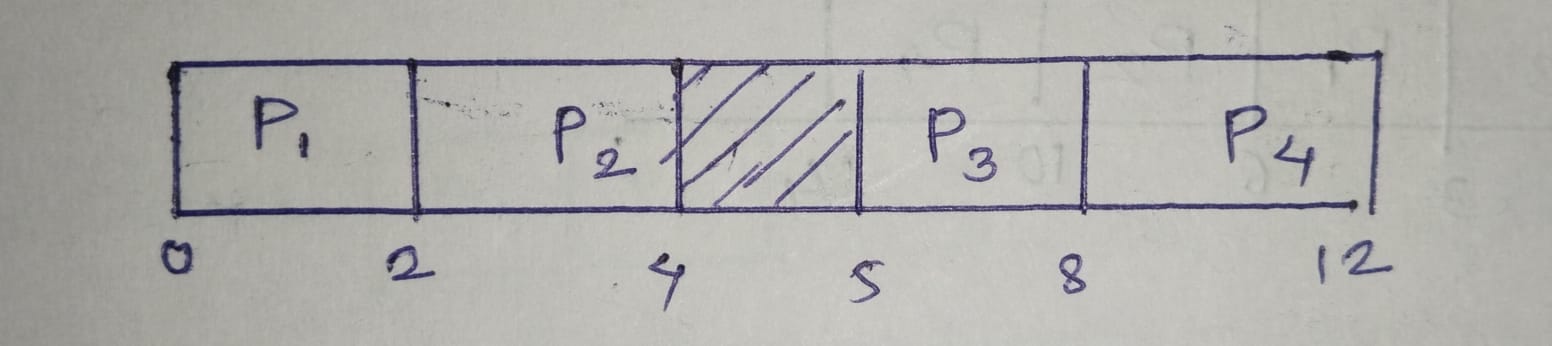
As a real life example of FCFS scheduling a billing counter system of shopping mall can be observed. The first person in the line gets the bill done first and then the next person gets the chance to get the bill and make payment and so on.

### **Problem 1**

Consider the given table below and find Completion time (CT), Turn-around time (TAT), Waiting time (WT), Response time (RT), Average Turn-around time and Average Waiting time?

|  |  |  |
| --- | --- | --- |
| **Process ID** | **Arrival time** | **Burst time** |
| **P1** | **0** | **2** |
| **P2** | **1** | **2** |
| **P3** | **5** | **3** |
| **P4** | **6** | **4** |

### ***Solution (answer)***

**Step 1 : Prepare Gantt chart**

**Step 2:** For this problem **CT, TAT, WT, RT** is shown in the given table –

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Process ID** | **Arrival time(AT)** | **Burst time(BT)** | **CT** | **TAT=CT-AT** | **WT=TAT-BT** | **RT** |
| P1 | 0 | 2 | 2 | 2-0=2 | 2-2=0 | 0 |
| P2 | 1 | 2 | 4 | 4-1=3 | 3-2=1 | 1 |
| P3 | 5 | 3 | 8 | 8-5=3 | 3-3=0 | 0 |
| P4 | 6 | 4 | 12 | 12-6=6 | 6-4=2 | 2 |

**\*RT(Response Time) =CPU first time - AT**

* **Average TAT time = (**2+3+3+6)/4 = **3.5 time unit** (time unit can be considered as milliseconds)
* **Average WT**= (0+1+0+2)/4 = 3/4 **= 0.75 time unit** (time unit can be considered as milliseconds)

***FOR EXPLANING THIS ALGORITHM CLICK TO BELOW LINK :***

[***https://youtu.be/MZdVAVMgNpA?si=8jlNmjjU09tmfBZq***](https://youtu.be/MZdVAVMgNpA?si=8jlNmjjU09tmfBZq)

1. **Shortest Job First (SJF)***OR***Shortest Job Next (SJN )Scheduling (*non-preemptive)***

Shortest Job First (SJF) also known as Shortest Job Next (SJN )Scheduling

Shortest job first scheduling is the job or process scheduling algorithm that follows the non-preemptive scheduling discipline. In this, scheduler selects the process from the waiting queue with the least completion time and allocate the CPU to that job or process. Shortest Job First is more desirable than FIFO algorithm because SJF is more optimal as it reduces average wait time which will increase the throughput.

SJF algorithm can be preemptive as well as non-preemptive. Preemptive scheduling is also known as shortest-remaining-time-first scheduling. In Preemptive approach, the new process arises when there is already executing process. If the burst of newly arriving process is lesser than the burst time of executing process than scheduler will preempt the execution of the process with lesser burst time.

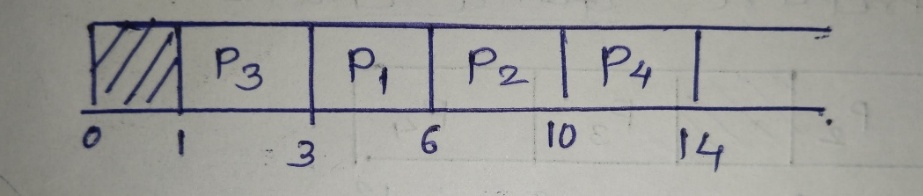
## Example

We are given with the processes P1, P2, P3, P4 and P5 having their corresponding burst time given below . Find CT, TAT, AVG TAT, WT, AVG WT, RT ?

|  |  |  |
| --- | --- | --- |
| **Process** | **Arrival time** | **Burst time** |
| **P1** | **1** | **3** |
| **P2** | **2** | **4** |
| **P3** | **1** | **2** |
| **P4** | **4** | **4** |

## Solution (answer)

* **Step 1 : Prepare Gantt chart**



* **Step 2 :** For this problem CT, TAT,WT , RT is shown in the given table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Process ID** | **Arrival time** | **Burst time** | **CT** | **TAT=CT-AT** | **WT=TAT-BT** | **RT** |
| P1 | 1 | 3 | 6 | 6-1=5 | 5-3=2 | 2 |
| P2 | 2 | 4 | 10 | 10-2=8 | 8-4=4 | 4 |
| P3 | 1 | 2 | 3 | 3-1=2 | 2-2=0 | 0 |
| P4 | 4 | 4 | 14 | 14-4=10 | 10-4=6 | 6 |

**\*RT(Response Time) =CPU first time - AT**

* **Average Turn-around time(TAT)**= (5+8+2+10)/4 = 25/4 **=6.25 time unit** (time unit can be considered as milliseconds)
* **Average waiting time(WT)**= (2+4+0+6)/5 = 12/4 = **3 time unit** (time unit can be considered as milliseconds)

***FOR EXPLANING THIS ALGORITHM CLICK TO BELOW LINK :***

[***https://youtu.be/VCIVXPoiLpU?si=TCmCsdg4gW2SGdtg***](https://youtu.be/VCIVXPoiLpU?si=TCmCsdg4gW2SGdtg)

1. **Shortest Remaining Time Next/First Scheduling (*preemptive*)**

In previous post, we have discussed [Set 1](https://www.geeksforgeeks.org/program-shortest-job-first-sjf-scheduling-set-1-non-preemptive/) of SJF i.e. non-preemptive. In this post we will discuss the preemptive version of SJF known as Shortest Remaining Time First (SRTF).

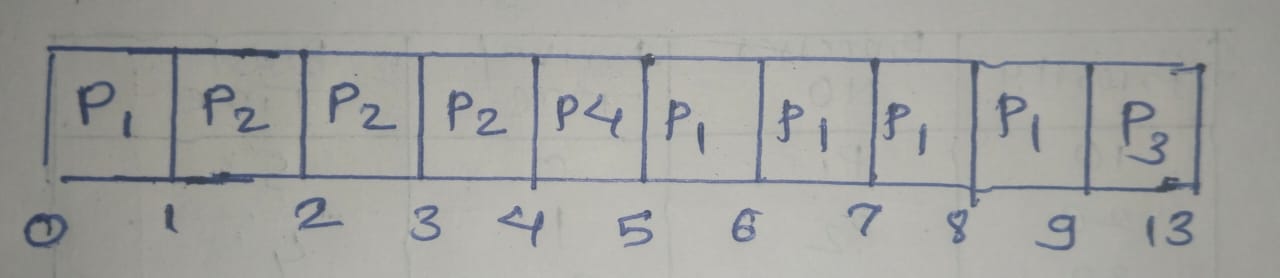
In the **Shortest Remaining Time First (SRTF) scheduling algorithm**, the process with the smallest amount of time remaining until completion is selected to execute. Since the currently executing process is the one with the shortest amount of time remaining by definition, and since that time should only reduce as execution progresses, processes will always run until they complete or a new process is added that requires a **smaller amount of time.**

## Example : We are given with the processes P1, P2, P3, P4 and P5 having their corresponding burst time given below . Find CT, TAT, AVG TAT, WT, AVG WT, RT ?

|  |  |  |
| --- | --- | --- |
| **Process** | **Arrival time** | **Burst time** |
| **P1** | **0** | **5** |
| **P2** | **1** | **3** |
| **P3** | **2** | **4** |
| **P4** | **4** | **1** |

**Solution (answer)**

* **Step 1 : Prepare Gantt chart**



* **Step 2 :** For this problem CT, TAT, WT, RT is shown in the given table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Process ID** | **Arrival time(AT)** | **Burst time(BT)** | **CT** | **TAT=CT-AT** | **WT=TAT-BT** | **RT** |
| P1 | 0 | 5 | 9 | 9-0=9 | 9-5=4 | 0 |
| P2 | 1 | 3 | 4 | 4-1=3 | 3-3=0 | 0 |
| P3 | 2 | 4 | 13 | 13-2=11 | 11-4=7 | 7 |
| P4 | 4 | 1 | 5 | 5-3=1 | 1-1=0 | 0 |

\***RT(Response Time) =CPU first time - AT**

* **Average Turn-around time(TAT)**= (9+3+11+1)/4 = 24/4 **=6 time unit** (time unit can be considered as milliseconds)
* **Average waiting time(WT)**= (4+0+7+0)/4 = 11/4 = **2.75 time unit** (time unit can be considered as milliseconds)

***FOR EXPLANING THIS ALGORITHM CLICK TO BELOW LINK :***

[***https://youtu.be/hoN7\_VMzw\_g?si=E-oqHcwExvRpnzrk***](https://youtu.be/hoN7_VMzw_g?si=E-oqHcwExvRpnzrk)

1. **Round Robin /Time Slice Algorithm *( preemptive)***

**Round Robin** is a CPU scheduling algorithm where each process is assigned a fixed time slot in a cyclic way. It is basically the preemptive version of First come First Serve CPU Scheduling algorithm.

* Round Robin CPU Algorithm generally focuses on Time Sharing technique.
* The period of time for which a process or job is allowed to run in a pre-emptive method is called time **quantum**.
* Each process or job present in the ready queue is assigned the CPU for that time quantum, if the execution of the process is completed during that time then the process will **end** else the process will go back to the **waiting table** and wait for its next turn to complete the execution.

## Characteristics of Round Robin CPU Scheduling Algorithm

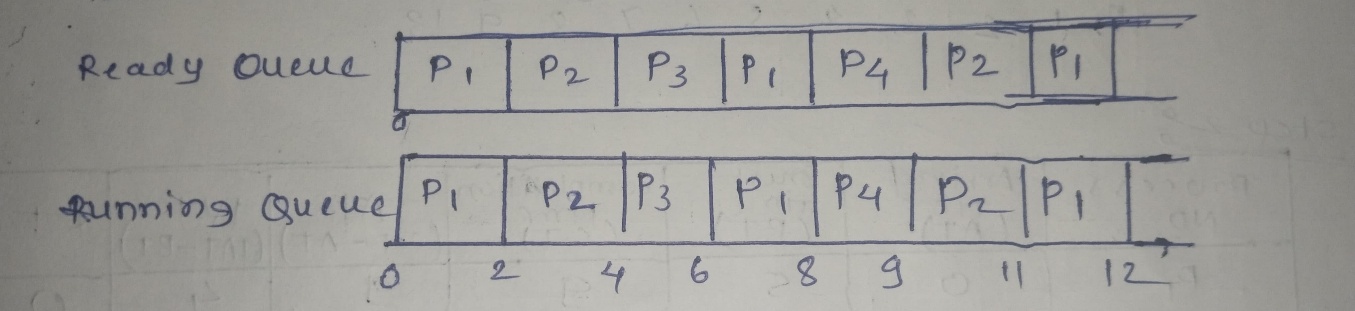
1. It is simple, easy to implement, and starvation-free as all processes get fair share of CPU.
2. One of the most commonly used technique in [CPU scheduling](https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/) as a core.
3. It is [preemptive](https://www.geeksforgeeks.org/preemptive-and-non-preemptive-scheduling/) as processes are assigned CPU only for a fixed slice of time at most.
4. The disadvantage of it is more overhead of context switching.

## Example : We are given with the processes P1, P2, P3, P4 and P5 having their corresponding burst time given below . Find CT, TAT, AVG TAT, WT, AVG WT, RT ?

|  |  |  |
| --- | --- | --- |
| **Process** | **Arrival time** | **Burst time** |
| **P1** | **0** | **5** |
| **P2** | **1** | **4** |
| **P3** | **2** | **2** |
| **P4** | **4** | **1** |

**Solution (answer)**

* **Step 1 : Prepare Gantt chart**



* **Step 2 :** For this problem CT, TAT, WT, RT is shown in the given table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Process ID** | **Arrival time(AT)** | **Burst time(BT)** | **CT** | **TAT=CT-AT** | **WT=TAT-BT** | **RT** |
| P1 | 0 | 5 | 12 | 12-0=12 | 12-5=7 | 0 |
| P2 | 1 | 4 | 11 | 11-1=10 | 10-4=6 | 1 |
| P3 | 2 | 2 | 6 | 6-2=4 | 4-2=2 | 2 |
| P4 | 4 | 1 | 9 | 9-4=5 | 5-1=4 | 4 |

**\*RT(Response Time) =CPU first time - AT**

* **Average Turn-around time(TAT)**= (12+10+4+5)/4 = 31/4 **=7.75 time unit** (time unit can be considered as milliseconds)
* **Average waiting time(WT)**= (7+6+2+4)/4 = 19/4 = **4.75 time unit** (time unit can be considered as milliseconds)

***FOR EXPLANING THIS ALGORITHM CLICK TO BELOW LINK :***

[**https://youtu.be/TxjIlNYRZ5M?si=QToOCkago8ZuAqyn**](https://youtu.be/TxjIlNYRZ5M?si=QToOCkago8ZuAqyn)

1. **Priority Based Scheduling**  ***(preemption)***

In the **Shortest Remaining Time First (SRTF) scheduling algorithm**, the process with the smallest amount of time remaining until completion is selected to execute. Since the currently executing process is the one with the shortest amount of time remaining by definition, and since that time should only reduce as execution progresses, processes will always run until they complete or a new process is added that requires a **smaller amount of time.**

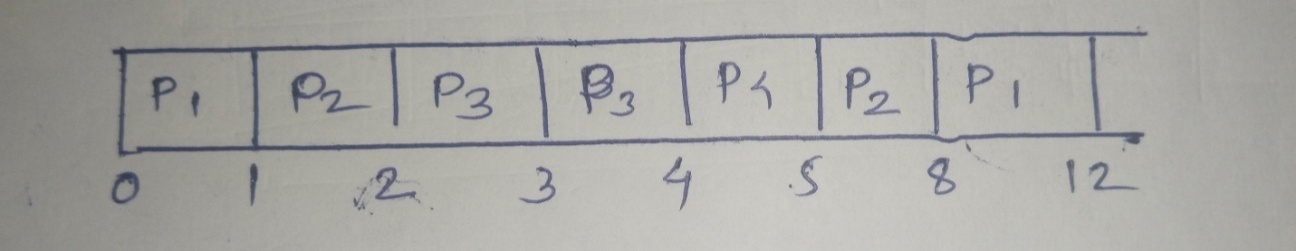
The Preemptive version of Shortest Job First(SJF) scheduling is known as Shortest Remaining Time First (SRTF). With the help of the SRTF algorithm, the process having the smallest amount of time remaining until completion is selected first to execute.

## Example : We are given with the processes P1, P2, P3, P4 and P5 having their corresponding burst time given below . Find CT, TAT, AVG TAT, WT, AVG WT, RT ?

|  |  |  |  |
| --- | --- | --- | --- |
| **Priority** | **Process** | **Arrival time** | **Burst time** |
| **10** | **P1** | **0** | **5** |
| **20** | **P2** | **1** | **4** |
| **30** | **P3** | **2** | **2** |
| **40** | **P4** | **4** | **1** |

**Solution (answer)**

* **Step 1 : Prepare Gantt chart**



* **Step 2 :** For this problem CT, TAT, AVG TAT, WT, AVG WT, RT is shown in the given table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Process ID** | **Arrival time(AT)** | **Burst time(BT)** | **CT** | **TAT=CT-AT** | **WT=TAT-BT** | **RT** |
| P1 | 0 | 5 | 12 | 12-0=12 | 12-5=7 | 0 |
| P2 | 1 | 4 | 8 | 8-1=7 | 7-4=3 | 0 |
| P3 | 2 | 2 | 4 | 4-2=2 | 2-2=0 | 0 |
| P4 | 4 | 1 | 5 | 5-4=1 | 1-1=0 | 0 |

**\*RT(Response Time) =CPU first time - AT**

* **Average Turn-around time(TAT)**= (12+7+2+1)/4 = 22/4 **=5.5 time unit** (time unit can be considered as milliseconds)
* **Average waiting time(WT)**= (7+3+0+0)/4 = 10/4 = **2.75** **time unit** (time unit can be considered as milliseconds)

***FOR EXPLANING THIS ALGORITHM CLICK TO BELOW LINK :***

<https://youtu.be/rsDGfFxSgiY?si=LG1Ku-Xpkt0SmEw->

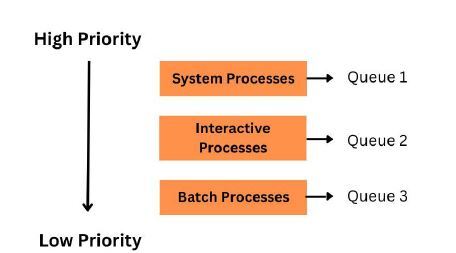
* **Multi-Level Queue (MLQ)**

Operating systems use a particular kind of scheduling algorithm called multilevel queue scheduling to control how resources are distributed across distinct tasks. It is an adaptation of the conventional queue-based scheduling method, in which processes are grouped according to their priority, process type, or other factors

The system can allocate system resources based on the priority and needs of the processes by assigning a separate scheduling algorithm to each queue. For instance, the background queue may employ first-come-first-serve scheduling to maximize the usage of system resources for longer-running activities, while the foreground queue might use Round Robin scheduling to prioritize interactive processes and speed up reaction time.

The goal of multilevel queue scheduling is to strike a compromise between fairness and performance. The system can increase overall performance while making sure that all processes are treated equitably by giving some processes more priority than others and distributing resources accordingly

## Components of the MLQ architecture



In an MLQ (Multilevel Queue) architecture, the system is divided into multiple queues or levels based on the type of processes. The MLQ architecture aims to efficiently handle different types of processes and prioritize their execution based on their characteristics.

Operating systems employ the potent scheduling technique known as multilevel queue scheduling to manage several queues of processes with various requirements and features. Multilevel queue scheduling can improve system performance by categorizing processes into various queues according to their characteristics, such as priority, execution time, and resource requirements.

* **Features of Multilevel Queue (MLQ) CPU Scheduling**

The features of a Multilevel Queue(MLQ) CPU Scheduling are as follows:

**Process Priorities**

* Priority is given to each process. The highest priority process should be carried out first, and so on.
* The process's type, importance, and attributes determine the priority of a process in MLQ CPU scheduling.
* For instance, interactive operations, such as user input and output, might be prioritized over batch processes, such as file backups.

**Various Queues**

* MLQ scheduling uses numerous queues, each with a different priority level.
* Processes with greater priorities are put in queues with higher priority levels.
* Processes with lower priorities are put in queues with lower priority levels.

**Pre-emption Method**

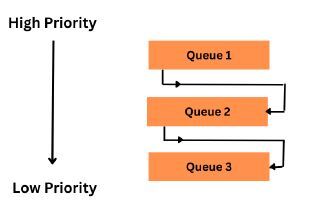
* Preemption in computing is the act of briefly stopping a running process intending to pick it back up later.
* This ensures that high-priority processes are carried out promptly.

**Resource Allocation**

* The process of allocating resources is how a computing system attempts to satisfy an application's hardware needs.
* MLQ scheduling ensures that procedures with higher priority levels are executed promptly while allowing lower-priority processes to run when ready.

**Feedback System**

* To guarantee that an interactive procedure is completed promptly, its priority may be raised if it has been sitting in a lower-priority queue for a long period.
* A feedback mechanism may be used to change the process's priority based on a process's behavior over time.
* **Multilevel Feedback Queue (MLFQ)**



MLFQ Programming is an instance of CPU scheduling technique that works through preserving several queues based on priority, each with a distinct time quantum. The delay period of those with greater importance queues is shorter, whereas the duration period of the ones with lower importance holds is longer.

When a fresh procedure emerges, it is assigned to the top of the importance queue. The timer for the CPU chooses the most significant operation from the most important queue and allocates the processing power to it. The procedure is permitted to continue for a set amount of time or as long as it is completed, whatever comes first. If the task finishes prior to the duration of the period running out, the processor's scheduler moves on to the next task in the queue.

**Advantages of Multilevel Feedback Queue Scheduling**:

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

**Disadvantages of Multilevel Feedback Queue Scheduling:**

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