# Preemptive Priority Scheduling algorithm

*A Project Report Submitted By*

# Anusha Anand – 2114110004

# Gaurav Bajaj – 2114110006

# Harshita Jain – 2114110061

# Raghav Kwatra - 2214110584

*In partial fulfilment of*

*the requirementFor*

*Project Based Learning of*

*System Programming and Operating systems*

# BACHELOR OF TECHNOLOGY

*In*

# COMPUTER SCIENCE AND ENGINEERING

*For*

*Computer graphics and multimedia*

*Under the guidance of*

***Prof. Reshma Abhang Ma’am***



# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**BHARATI VIDYAPEETH (DEEMED TO BE UNIVERSITY)**

**COLLEGE OF ENGINEERING, PUNE- 43**

2022-23

# BHARATI VIDYAPEETH (DEEMED TO BE UNIVERSITY)

# COLLEGE OF ENGINEERING, PUNE- 43



**CERTIFICATE**

This is to certify that the Project Based Learning report titled **CRT Monitor**, submitted by **Anusha Anand (2114110004), Gaurav Bajaj(2114110006), Harshita Jain (2114110061), Raghav Kwatra(2214110584)** to the Bharati Vidyapeeth (Deemed to be University), College of Engineering, Pune - 43 for the award of the degree of **BACHELOR OF TECHNOLOGY** in Computer Science and Engineering is a bonafede record of the PBL work done by him/them under my supervision.

Place: Pune Name of Subject Teacher :

Prof. Reshma Abhang Ma’am

Date:24/04/23

# Index

|  |  |  |
| --- | --- | --- |
| **Sr.No.** | **Title** | **Page No.** |
| 1 | Introduction to Preemptive Priority Scheduling algorithm | 4 |
| 2 | Algorithm | 5-7 |
| 3 | Codes and Outputs | 8-10 |
| 4 | Snapshot | 11 |
| 5 | Conclusion |  |
| 6 | Reference |  |

**Introduction to Preemptive Priority Scheduling algorithm**

In Preemptive Priority Scheduling, at the time of arrival of a process in the ready queue, its Priority is compared with the priority of the other processes present in the ready queue as well as with the one which is being executed by the CPU at that point of time. The One with the highest priority among all the available processes will be given the CPU next.

The difference between preemptive priority scheduling and non preemptive priority scheduling is that, in the preemptive priority scheduling, the job which is being executed can be stopped at the arrival of a higher priority job.

Once all the jobs get available in the ready queue, the algorithm will behave as non-preemptive priority scheduling, which means the job scheduled will run till the completion and no preemption will be done.

A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. There are six popular process scheduling algorithms which we are going to discuss in this project −

* First-Come, First-Served (FCFS) Scheduling
* Shortest-Job-Next (SJN) Scheduling
* Priority Scheduling
* Shortest Remaining Time
* Round Robin (RR) Scheduling
* Multiple-Level Queues Scheduling

These algorithms are either **non-preemptive or preemptive**. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

**Algorithm**

## **Priority Based Scheduling**

* Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
* Each process is assigned a priority. Process with highest priority is to be executed first and so on.
* Processes with same priority are executed on first come first served basis.
* Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Consider the following table of arrival time, Priority and burst time for seven processes P1, P2, P3, P4, P5, P6 and P7

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Arrival Time | Priority | Burst Time |
| P1 | 0 ms | 3 | 8 ms |
| P2 | 1 ms | 4 | 2 ms |
| P3 | 3 ms | 4 | 4 ms |
| P4 | 4 ms | 5 | 1 ms |
| P5 | 5 ms | 2 | 6 ms |
| P6 | 6 ms | 6 | 5 ms |
| P7 | 10 ms | 1 | 1 ms |

* At time **t = 0**,

Process P1 is available in the ready queue, executing P1 for 1 ms

Remaining B.T for P1 = 8-1 = 7 ms.

* At time **t = 1**,

The priority of P1 is greater than P2, so we execute P1 for 2 ms, from 1 ms to 3 ms.

Remaining B.T for P1 = 7-2 = 5 ms.

* At time **t = 3**,

The priority of P1 is greater than P3, so we execute P1 for 1 ms.

Remaining B.T for P1 = 5-1 = 4 ms.

* At time **t = 4**,

The priority of P1 is greater than P4, so we execute P1 for 1 ms.

Remaining B.T for P1 = 4-1 = 3 ms.

* At time **t = 5**,

The priority of P5 is greater than P1, so we execute P5 for 1 ms.

Remaining B.T for P5 = 6-1 = 5 ms.

* At time **t = 6**,

The priority of P5 is greater than P6, so we execute P5 for 4 ms.

Remaining B.T for P5 = 5-4 = 1 ms.

* At time **t = 10**,

The priority of P7 is greater than P5, so we execute P7 for 1 ms.

Remaining B.T for P7 = 1-1 = 0 ms.

Here Process P7 completes its execution.

* At time **t = 11**,

Now we take the process which is having the highest priority.

Here we find P5 is having the highest priority & execute P5 completely

Remaining B.T of P5 = 1-1 = 0 ms.

Here Process P5 completes its execution.

* At time **t = 12**,

Now we take the process which is having the highest priority.

Here we find P1 is having the highest priority & execute P1 completely

Remaining B.T of P1 = 3-3 = 0 ms.

Here Process P1 completes its execution.

* At time **t = 15**,

Now we take the process which is having the highest priority.

Here we find P2 is having the highest priority & execute P2 completely

Remaining B.T of P2 = 2-2 = 0 ms.

Here Process P2 completes its execution.

* At time **t = 17**,

Now we take the process which is having the highest priority.

Here we find P3 is having the highest priority & execute P3 completely

Remaining B.T of P3 = 4-4 = 0 ms.

Here Process P3 completes its execution.

* At time **t = 21**,

Now we take the process which is having the highest priority.

Here we find P4 is having the highest priority & execute P4 completely

Remaining B.T of P4 = 1-1 = 0 ms.

Here Process P4 completes its execution.

* At time **t = 22**,

Now we take the process which is having the highest priority.

Here we find P6 is having the highest priority & execute P6 completely

Remaining B.T of P6 = 5-5 = 0 ms.

Here Process P6 completes its execution.

**Gantt chart:**

A picture containing text, font, line, screenshot

Description automatically generated

A picture containing text, screenshot, number, font

Description automatically generated

Here, ***H – Higher Priority, L – Least Priority***

**Codes**

#include<stdio.h>

struct process

{

int WT,AT,BT,TAT,PT;

};

struct process a[10];

int main()

{

int n,temp[10],t,count=0,short\_p;

float total\_WT=0,total\_TAT=0,Avg\_WT,Avg\_TAT;

printf("Enter the number of the process\n");

scanf("%d",&n);

printf("Enter the arrival time , burst time and priority of the process\n");

printf("AT BT PT\n");

for(int i=0;i<n;i++)

{

scanf("%d%d%d",&a[i].AT,&a[i].BT,&a[i].PT);

// copying the burst time in

// a temp array fot futher use

temp[i]=a[i].BT;

}

// we initialize the burst time

// of a process with maximum

a[9].PT=10000;

for(t=0;count!=n;t++)

{

short\_p=9;

for(int i=0;i<n;i++)

{

if(a[short\_p].PT>a[i].PT && a[i].AT<=t && a[i].BT>0)

{

short\_p=i;

}

}

a[short\_p].BT=a[short\_p].BT-1;

// if any process is completed

if(a[short\_p].BT==0)

{

// one process is completed

// so count increases by 1

count++;

a[short\_p].WT=t+1-a[short\_p].AT-temp[short\_p];

a[short\_p].TAT=t+1-a[short\_p].AT;

// total calculation

total\_WT=total\_WT+a[short\_p].WT;

total\_TAT=total\_TAT+a[short\_p].TAT;

}

}

Avg\_WT=total\_WT/n;

Avg\_TAT=total\_TAT/n;

// printing of the answer

printf("ID WT TAT\n");

for(int i=0;i<n;i++)

{

printf("%d %d\t%d\n",i+1,a[i].WT,a[i].TAT);

}

printf("Avg waiting time of the process is %f\n",Avg\_WT);

printf("Avg turn around time of the process is %f\n",Avg\_TAT);

return 0;

}

**Output**

**Enter the number of the process**

**7**

**Enter the arrival time , burst time and priority of the process**

**AT BT PT**

**0 8 3**

**1 2 4**

**3 4 4**

**4 1 5**

**5 6 2**

**6 5 6**

**10 1 1**

**ID WT TAT**

**1 7 15**

**2 14 16**

**3 14 18**

**4 17 18**

**5 1 7**

**6 16 21**

**7 0 1**

**Avg waiting time of the process is 9.857142**

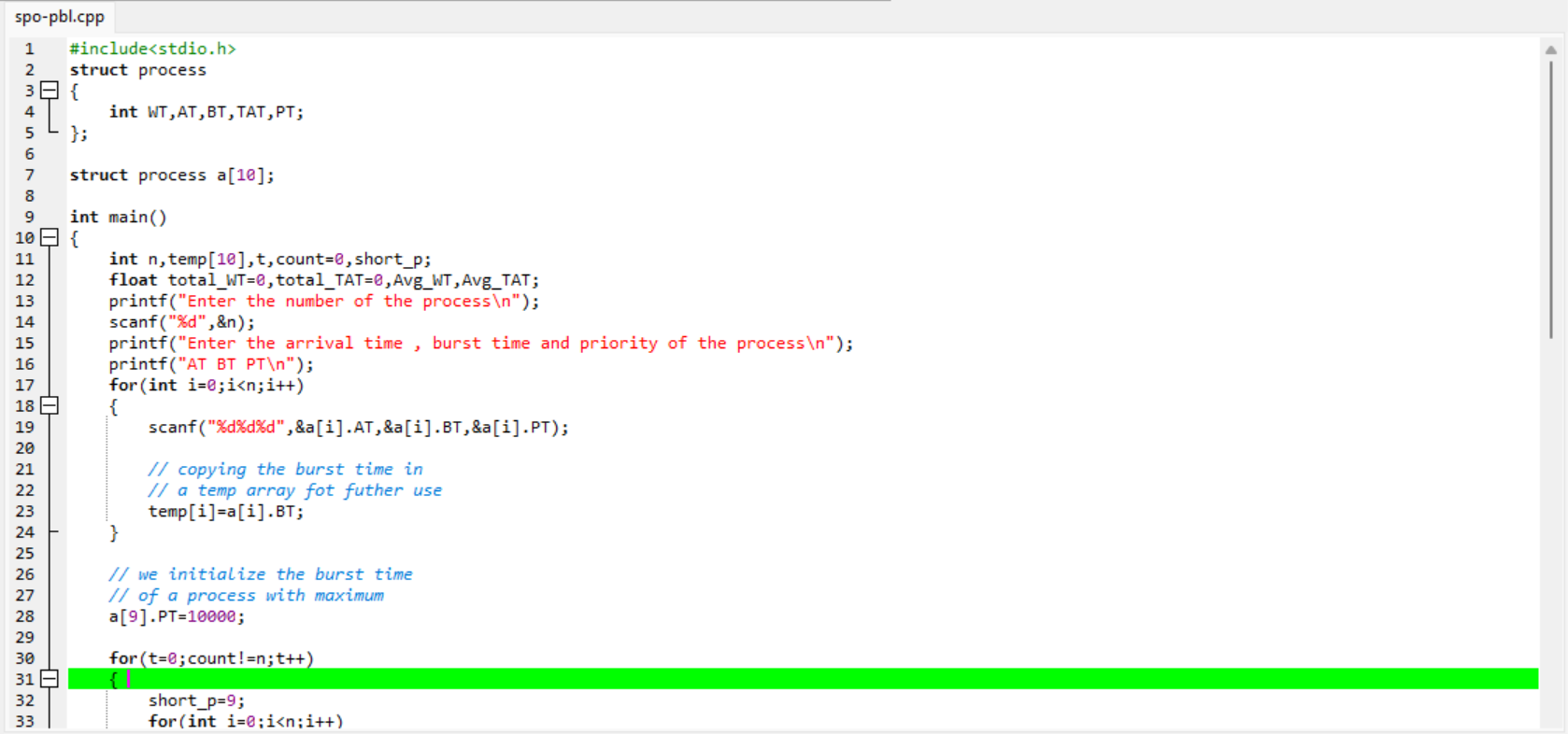
**Avg turn around time of the process is 13.714286**

**--------------------------------**

**Process exited after 155.6 seconds with return value 0**

**Press any key to continue . . .**

**Snapshots**

****

**A screenshot of a computer

Description automatically generated**

**Conclusion**

In this project based learning on “Preemptive Priority Scheduling algorithm” we went through the codes and outputs of this algorithm as already mentioned above in the report.

In summary, [Pre-emptive Priority Scheduling algorithm is a CPU scheduling algorithm that works on the basis of priority assigned to each process**1**](https://www.geeksforgeeks.org/preemptive-priority-cpu-scheduling-algortithm/). [In this algorithm, the scheduler selects the tasks to work as per their priority](https://www.guru99.com/priority-scheduling-program.html). [In Pre-emptive Priority Scheduling, the tasks are mostly assigned with their priorities and the CPU has been allocated to a specific process](https://www.guru99.com/priority-scheduling-program.html).

[The main advantage of this algorithm is that it ensures that high priority processes are never starved](https://www.geeksforgeeks.org/preemptive-priority-cpu-scheduling-algortithm/). [However, it can lead to lower priority processes being blocked indefinitely](https://www.baeldung.com/cs/scheduling-types).

**Reference**

1. [Preemptive Priority Scheduling algorithm - Search (bing.com)](https://www.bing.com/search?q=Preemptive+Priority+Scheduling+algorithm&qs=n&form=QBRE&sp=-1&lq=0&pq=preemptive+priority+scheduling+algorithm&sc=5-40&sk=&cvid=56360493984743D78DFD603FC32DCBF1&ghsh=0&ghacc=0&ghpl=)
2. [Preemptive Priority CPU Scheduling Algorithm - GeeksforGeeks](https://www.geeksforgeeks.org/preemptive-priority-cpu-scheduling-algortithm/)
3. [OS Non Preemptive Priority Scheduling - javatpoint](https://www.javatpoint.com/os-non-preemptive-priority-scheduling)
4. [Operating System Scheduling algorithms (tutorialspoint.com)](https://www.tutorialspoint.com/operating_system/os_process_scheduling_algorithms.htm)