

**Mawlana Bhashani Science and Technology University**

**Lab-Report**

Report No:09

Course code: ICT-3110

Course title: Operating System Lab

Date of Performance:09/09/2020

Date of Submission:10/09/2020

**Submitted by Submitted To**

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

Name:Md Asikur Rahman

ID:IT-18025

3rd  year 1st semester

Session: 2017-2018

Dept. of ICT

MBSTU.

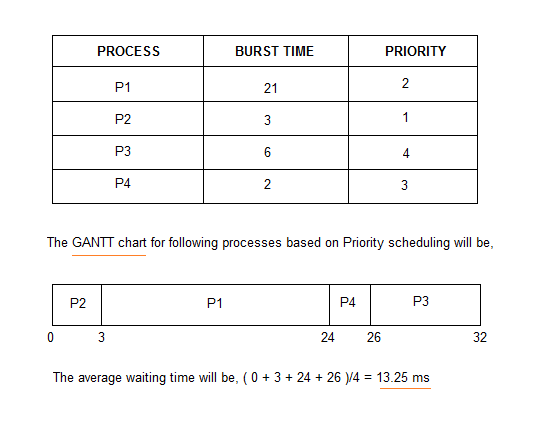
# Experiment no : 09

# Experiment Name : Implementation of Priority Scheduling Algorithm

# Theory:

**Priority Scheduling** is a method of scheduling processes that is based on priority. In this algorithm, the scheduler selects the tasks to work as per the priority.

The processes with higher priority should be carried out first, whereas jobs with equal priorities are carried out on a round-robin or FCFS basis. Priority depends upon memory requirements, time requirements, etc.

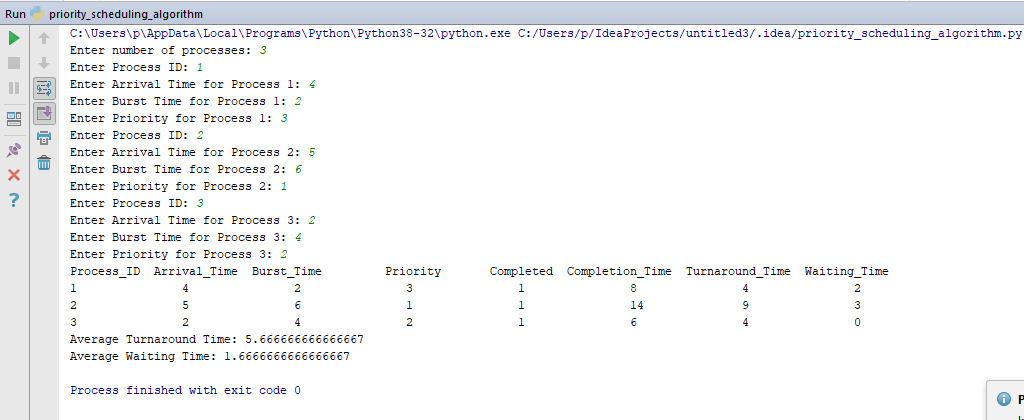


# Working Procedure:

Coding implementation with python ….

*''' Created by asik  
 date:04/09/2020  
'''***class** Priority:  
  
 **def** processData(self, no\_of\_processes):  
 process\_data = []  
 **for** i **in** range(no\_of\_processes):  
 temporary = []  
 process\_id = int(input(**"Enter Process ID: "**))  
  
 arrival\_time = int(input(**f"Enter Arrival Time for Process {process\_id}: "**))  
  
 burst\_time = int(input(**f"Enter Burst Time for Process {process\_id}: "**))  
  
 priority = int(input(**f"Enter Priority for Process {process\_id}: "**))  
  
 temporary.extend([process\_id, arrival\_time, burst\_time, priority, 0])  
 **'''  
 '0' is the state of the process. 0 means not executed and 1 means execution complete  
 '''** process\_data.append(temporary)  
 Priority.schedulingProcess(self, process\_data)  
  
  
 **def** schedulingProcess(self, process\_data):  
 start\_time = []  
 exit\_time = []  
 s\_time = 0  
 process\_data.sort(key=**lambda** x: x[1])  
 **'''  
 Sort processes according to the Arrival Time  
 '''  
 for** i **in** range(len(process\_data)):  
 ready\_queue = []  
 temp = []  
 normal\_queue = []  
 **for** j **in** range(len(process\_data)):  
 **if** (process\_data[j][1] <= s\_time) **and** (process\_data[j][4] == 0):  
 temp.extend([process\_data[j][0], process\_data[j][1], process\_data[j][2], process\_data[j][3]])  
 ready\_queue.append(temp)  
 temp = []  
 **elif** process\_data[j][4] == 0:  
 temp.extend([process\_data[j][0], process\_data[j][1], process\_data[j][2], process\_data[j][3]])  
 normal\_queue.append(temp)  
 temp = []  
 **if** len(ready\_queue) != 0:  
 ready\_queue.sort(key=**lambda** x: x[3], reverse=**True**)  
 **'''  
 Sort the processes according to the Priority, considering Higher the Value, Higher the Priority  
 '''** start\_time.append(s\_time)  
 s\_time = s\_time + ready\_queue[0][2]  
 e\_time = s\_time  
 exit\_time.append(e\_time)  
 **for** k **in** range(len(process\_data)):  
 **if** process\_data[k][0] == ready\_queue[0][0]:  
 **break** process\_data[k][4] = 1  
 process\_data[k].append(e\_time)  
 **elif** len(ready\_queue) == 0:  
 **if** s\_time < normal\_queue[0][1]:  
 s\_time = normal\_queue[0][1]  
 start\_time.append(s\_time)  
 s\_time = s\_time + normal\_queue[0][2]  
 e\_time = s\_time  
 exit\_time.append(e\_time)  
 **for** k **in** range(len(process\_data)):  
 **if** process\_data[k][0] == normal\_queue[0][0]:  
 **break** process\_data[k][4] = 1  
 process\_data[k].append(e\_time)  
 t\_time = Priority.calculateTurnaroundTime(self, process\_data)  
 w\_time = Priority.calculateWaitingTime(self, process\_data)  
 Priority.printData(self, process\_data, t\_time, w\_time)  
  
  
 **def** calculateTurnaroundTime(self, process\_data):  
 total\_turnaround\_time = 0  
 **for** i **in** range(len(process\_data)):  
 turnaround\_time = process\_data[i][5] - process\_data[i][1]  
 **'''  
 turnaround\_time = completion\_time - arrival\_time  
 '''** total\_turnaround\_time = total\_turnaround\_time + turnaround\_time  
 process\_data[i].append(turnaround\_time)  
 average\_turnaround\_time = total\_turnaround\_time / len(process\_data)  
 **'''  
 average\_turnaround\_time = total\_turnaround\_time / no\_of\_processes  
 '''  
 return** average\_turnaround\_time  
  
  
 **def** calculateWaitingTime(self, process\_data):  
 total\_waiting\_time = 0  
 **for** i **in** range(len(process\_data)):  
 waiting\_time = process\_data[i][6] - process\_data[i][2]  
 **'''  
 waiting\_time = turnaround\_time - burst\_time  
 '''** total\_waiting\_time = total\_waiting\_time + waiting\_time  
 process\_data[i].append(waiting\_time)  
 average\_waiting\_time = total\_waiting\_time / len(process\_data)  
 **'''  
 average\_waiting\_time = total\_waiting\_time / no\_of\_processes  
 '''  
 return** average\_waiting\_time  
  
  
 **def** printData(self, process\_data, average\_turnaround\_time, average\_waiting\_time):  
 process\_data.sort(key=**lambda** x: x[0])  
 **'''  
 Sort processes according to the Process ID  
 '''** print(**"Process\_ID Arrival\_Time Burst\_Time Priority Completed Completion\_Time Turnaround\_Time Waiting\_Time"**)  
 **for** i **in** range(len(process\_data)):  
 **for** j **in** range(len(process\_data[i])):  
 print(process\_data[i][j], end=**"\t\t\t\t"**)  
 print()  
 print(**f'Average Turnaround Time: {average\_turnaround\_time}'**)  
  
 print(**f'Average Waiting Time: {average\_waiting\_time}'**)  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 no\_of\_processes = int(input(**"Enter number of processes: "**))  
 priority = Priority()  
 priority.processData(no\_of\_processes)

# Output:



# Discussion:

We learn from this lab

1.[What is Priority scheduling?](https://www.guru99.com/priority-scheduling-program.html#1)

2.[Types of Priority Scheduling](https://www.guru99.com/priority-scheduling-program.html#2)

3.[Characteristics of Priority Scheduling](https://www.guru99.com/priority-scheduling-program.html#3)

4.[Example of Priority Scheduling](https://www.guru99.com/priority-scheduling-program.html#4)

5.[Advantages of priority scheduling](https://www.guru99.com/priority-scheduling-program.html" \l "5)