**OPERATING SYSTEM**

**ASSIGNMENT BASED**

**PROJECT**

**SUBJECT CODE: CSE316**

**Submitted by: Sweta saloni**

**Registration Number: 11808575**

**Roll Number: 29**

**Email Address: Sweta22saloni@gmail.com**

**GitHub Link:**

**Section: EE032**

**Date: 30.03.2020**

**DESCRIPTION**

**Problem Statement:**

Ten students (a, b, c, d, e, f, g, h, i, j) are going to get there pictured clicked by university camera. Only one student can enter the camera room while the other students wait outside the room. The students are waiting in a queue to enter the room. To pass time the students start to play a game. In this game the students give candies to each other in a random manner (assume the students never run out of candies). They decide that the student with highest candies will be allowed to enter. When the student with highest amount of candies enter the room, the student starts the game again. Initially the students do not know if there is any body in the room and they start their game and the student with highest candies enter. Write and implement the algorithm to schedule such and compute the waiting and turnaround time. Consider the arrival time and burst time as given by the user.

**Explanation:**

We have maximum of 10 processes (students) which is arranged in Queue, that is in First in First Out (FIFO) manner. These processes are going to be processed one at a time in non pre-emptive manner, that is if one process is processing in CPU then until and unless the process gets completed no other process will get privilege of getting processed. As per the constraints of these problem we need provide the priority (candies) to the given processes randomly and we need to execute that process first that is having the highest priority accordingly. The Arrival time (AT) and Burst time (BT) of the process will be choose by the user as mentioned in the problem statement. After all process execution we need to provide the Average Waiting Time and Average Turn Around Time of the given processes.

The Problem is based on Priority scheduling algorithm, where we need to execute the process having the maximum priority and the same is demanded by above problem.

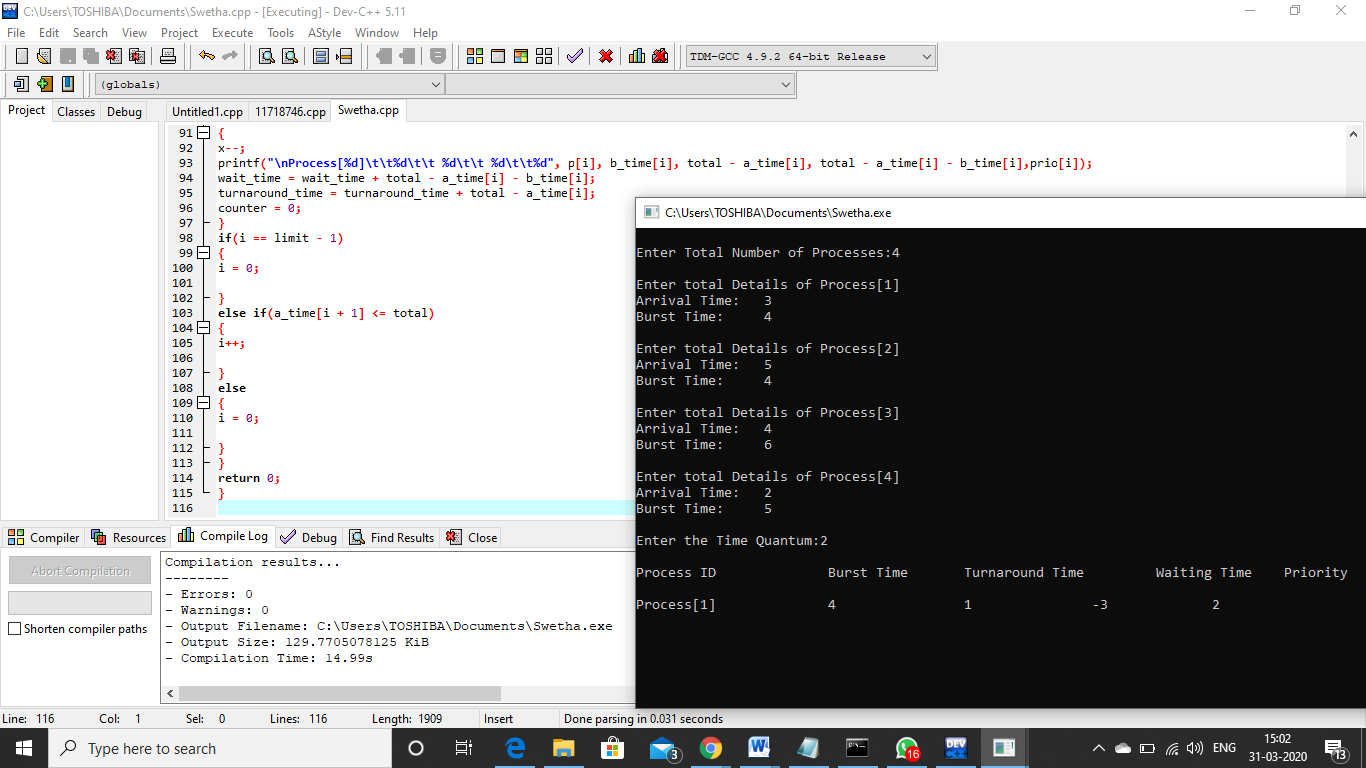
**CODE:**

|  |  |
| --- | --- |
|  | #include<stdio.h> |
|  |  |
|  | int main() |
|  | { |
|  | int i, limit, total = 0, x, counter = 0, time\_quantum,j; |
|  |  |
|  | int wait\_time = 0, turnaround\_time = 0,pos,z,p[10],prio[10], a\_time[10], b\_time[10], temp[10],b; |
|  |  |
|  | float average\_wait\_time, average\_turnaround\_time; |
|  |  |
|  | printf("\nEnter Total Number of Processes:"); |
|  |  |
|  | scanf("%d", &limit); |
|  |  |
|  | x = limit; |
|  | for(i = 0; i < limit; i++) |
|  | { |
|  | p[i]=i+1; |
|  |  |
|  | prio[i]=0; |
|  | printf("\nEnter total Details of Process[%d]\n", i + 1); |
|  | printf("Arrival Time:\t"); |
|  | scanf("%d", &a\_time[i]); |
|  | printf("Burst Time:\t"); |
|  | scanf("%d", &b\_time[i]); |
|  | temp[i] = b\_time[i]; |
|  | } |
|  |  |
|  | printf("\nEnter the Time Quantum:"); |
|  | scanf("%d", &time\_quantum); |
|  | printf("\nProcess ID\t\tBurst Time\t Turnaround Time\t Waiting Time\t Priority\n"); |
|  | for(total = 0, i = 0; x != 0;) |
|  | { |
|  |  |
|  | for(z=0;z<limit;z++) |
|  | { |
|  | int temp1; |
|  | pos=z; |
|  | for(j=z+1;j<limit;j++) |
|  | { |
|  | if(prio[j]<prio[pos]) |
|  | pos=j; |
|  | } |
|  |  |
|  | temp1=prio[z]; |
|  |  |
|  | prio[z]=prio[pos]; |
|  |  |
|  | prio[pos]=temp1; |
|  |  |
|  | temp1=b\_time[z]; |
|  | b\_time[z]=b\_time[pos]; |
|  | b\_time[pos]=temp1; |
|  | temp1=a\_time[z]; |
|  | a\_time[z]=a\_time[pos]; |
|  | a\_time[pos]=temp1; |
|  |  |
|  | temp1=p[z]; |
|  | p[z]=p[pos]; |
|  | p[pos]=temp1; |
|  |  |
|  | temp1=temp[z]; |
|  | temp[z]=temp[pos]; |
|  | temp[pos]=temp1; |
|  | } |
|  | { |
|  | } |
|  |  |
|  | if(temp[i] <= time\_quantum && temp[i] > 0) |
|  | { |
|  | total = total + temp[i]; |
|  | temp[i] = 0; |
|  | counter = 1; |
|  | } |
|  |  |
|  | else if(temp[i] > 0) |
|  | { |
|  | temp[i] = temp[i] - time\_quantum; |
|  | total = total + time\_quantum; |
|  | } |
|  |  |
|  | for(b=0;b<limit;b++) |
|  | { |
|  | if(b==i) |
|  | prio[b]+=1; |
|  | else |
|  | prio[b]+=2; |
|  | } |
|  |  |
|  | if(temp[i] == 0 && counter == 1) |
|  | { |
|  | x--; |
|  | printf("\nProcess[%d]\t\t%d\t\t %d\t\t %d\t\t%d", p[i], b\_time[i], total - a\_time[i], total - a\_time[i] - b\_time[i],prio[i]); |
|  | wait\_time = wait\_time + total - a\_time[i] - b\_time[i]; |
|  | turnaround\_time = turnaround\_time + total - a\_time[i]; |
|  | counter = 0; |
|  | } |
|  | if(i == limit - 1) |
|  | { |
|  | i = 0; |
|  |  |
|  | } |
|  | else if(a\_time[i + 1] <= total) |
|  | { |
|  | i++; |
|  |  |
|  | } |
|  | else |
|  | { |
|  | i = 0; |
|  |  |
|  | } |
|  | } |
|  | return 0; |
|  | } |

Top of Form

Bottom of Form

**Compile and run:**

****

**ALGORITHM**

1. Initialize number of process/student [max=10]
2. Repeat step a. to d. for (process!=0)
   1. Name of process/student
   2. Arrival time
   3. Burst time
   4. Random priority
3. Repeat step a. to d. for (process!=0)
   1. Display Name of process/student
   2. Display Arrival time
   3. Display Burst time
   4. Display Random priority
4. Finding the smallest arrival time
   1. Let fst\_exe=max possible num
   2. Repeat step a. to d. for (process!=0)
      1. If fst\_exe>at[process] && priority[process]!=0 then:
         1. Fst\_exe=at[process]
         2. Current\_process=num of that[process]
      2. Else fst\_exe= = at[process] priority[process]!=0 then:
         1. If priority[current\_process]<priority[process] then:
            1. Current\_process=num of that[process]
         2. else (priority of current[process]= =priority[process])
            1. if(burst time[process]<burst time[current\_process])

Current\_process=num of that[process]

//As the process having smallest arrival time will be not be compare with any other process’s priority and that’s why it will executed first.

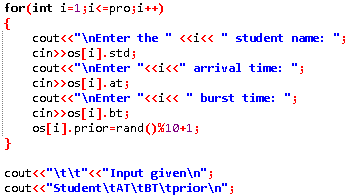
1. Repeat step a. to d. for (process! =0)
   1. [process execute]? Make its priority 0
   2. Calculate the current time with formula ct[process]+bt[process]
   3. Calculate the turn around time with formula ct[process]-at[process]
   4. Calculate the wait time of with formula tat[process]-bt[process]
   5. Repeat step a. to d. for (process! =0)
      1. [Arrival time of process>current time?]
         1. Priority[process]= -1
         2. Wait time[process]=0
      2. Look for next process to be executed in basis of priority of process
         1. [process execute]? Make its priority 0
         2. Calculate the current time with formula ct[process]+bt[process]
         3. Calculate the turn around time with formula ct[process]-at[process]
         4. Calculate the wait time of with formula tat[process]-bt[process]
   6. Calculate the sum of all the wait time obtained
   7. Calculate all the turn around time obtained
   8. Calculate the average of both by dividing it by number of process.
   9. Display all the outputs.

**CODE SNIPPET**

* + - 1. Ten students (a, b, c, d, e, f, g, h, i, j) are going to get there pictured clicked by university camera.

The student here is actually referring to the number of process. The problem says that we need to provide maximum of only 10 processes and not more than that.

* + - 1. The students are waiting in a queue to enter the room.

We need to arrange and display the input in First in First Out method.

* + - 1. In the game student need to give the candies to each other in random manner.

Candies here basically refer to the priority of the student. The problem is actually asking us to provide the priority randomly to all the processes, it can be given by the user or can be randomly generated by the system. I have used the method of generating the numbers randomly by using rand function.

* + - 1. They decide that the student with highest candies will be allowed to enter.

Here the question says we need to execute the process which have maximum priority and need to execute the last that has least priority.

* + - 1. Initially the students do not know if there is any body in the room and they start their game and the student with highest candies enter.

This line indirectly tells us that right now the CPU is not processing any task hence it is ideal. We can directly provide the input that has the maximum priority in the Queue.

* + - 1. Consider the arrival time and burst time as given by the user.

Here the problem says we need these two inputs Arrival time of the process and burst time of the process to be given by user.

.

THANK YOU