**Student Name**: Jami Abhisikth

**Student ID** : 11701172

Roll no : 08

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

#include<stdio.h> struct process

{

int processID; int burstTime; int arrivalTime; int priority;

int waitTime;

};

int total\_time,burst\_time=0;

int total=-1,i=-1; //to calcualte no of process entered in cpu and i to put result in cpu

/\*to make buffer of queue in which process will be entered

and final process after completion to be entered in result of buffer 100;\*/ struct process queue[100],result[100],swap;

int process\_create() //for number of inputs

{

int n;

printf("enter the number of process you want to enter:"); scanf("%d",&n);

return n;

}

void execute() ////function to execute the process

{

if(total>=0)

{

int wait,j;

if(burst\_time!=0 && queue[0].burstTime!=0)

{

queue[0].burstTime--; burst\_time--; queue[0].priority++;

queue[0].arrivalTime=total\_time+1; total\_time++;

//to increase the wait and priority of waiting process for(wait=1;wait<=total;wait++)

{

queue[wait].priority+=2; queue[wait].waitTime=++queue[wait].waitTime;

}

}

//if process gets completed ,it is put in result queue if(queue[0].burstTime==0)

{

i++;

result[i]=queue[0]; for(wait=0;wait<total;wait++)

{

queue[wait]=queue[wait+1];

}

total--;

}

//to sort the process again in by priority for(wait=0;wait<total;wait++)

{

for(j=0;j<total;j++)

{

if(queue[wait].priority<=queue[j].priority)

{

swap=queue[wait]; queue[wait]=queue[j]; queue[j]=swap;

}

}

}

if(queue[0].priority<=queue[1].priority && total>=1)

{

swap=queue[0]; for(wait=0;wait<total;wait++)

{

queue[wait]=queue[wait+1];

}

queue[total]=swap;

}

}

}

void main()

{

int l,j,n=process\_create(),count=0; float avg\_WaitTime=0;

struct process pcreate[n]; for(l=0;l<n;l++)

{

pcreate[l].processID=l+1;

printf("\nEnter the arrival time of process[%d]: ",l+1); scanf("%d",&pcreate[l].arrivalTime);

printf("\nEnter the burst time of process[%d]: ",l+1); scanf("%d",&pcreate[l].burstTime); pcreate[l].priority=0;

pcreate[l].waitTime=0; burst\_time=burst\_time+pcreate[l].burstTime;

}

for(l=0;l<n;l++)

{

for(j=0;j<n;j++)

{

if(pcreate[l].arrivalTime<pcreate[j].arrivalTime)

{

swap=pcreate[l]; pcreate[l]=pcreate[j]; pcreate[j]=swap;

}

if(pcreate[l].arrivalTime==pcreate[j].arrivalTime)

{

if(pcreate[l].burstTime<=pcreate[j].burstTime)

{

swap=pcreate[l]; pcreate[l]=pcreate[j]; pcreate[j]=swap;

}

}

}

}// printing the sorted process id with respect to arrival time and if arrival time is equal than burst time.

printf("VALUES ENTERED:\n\*(TABLE SORTED ACCORDING TO THE AARIVAL TIME)\n\n");

printf(" PROCESS TABLE\n\n");

printf(" PROCESS ID ARRIVAL TIME SERVICE TIME\n\n");

for(l=0;l<n;l++)

{

printf(" %d %d

%d\n\n",pcreate[l].processID,pcreate[l].arrivalTime,pcreate[l].burstTime );

}

total\_time=pcreate[0].arrivalTime; for(j=pcreate[0].arrivalTime;j<=pcreate[n-1].arrivalTime;j++)

{

for(l=0;l<n;l++)

{

if(pcreate[l].arrivalTime==j && count!=n)

{

total++; queue[total]=pcreate[l]; count++;

}

if(count==n) break;

}

execute(); total\_time++;

while(burst\_time!=0 && count==n)

{

execute(); total\_time++;

}

if(count==n) break;

}

printf("PROCESS IN ORDER OF THEIR COMPLETION:\n\n"); printf(" FINAL PROCESS EXECUTION TABLE\n\n");

printf(" PROCESS ID ARRIVAL TIME SERVICETIME WAITING TIME\n");

for(l=0;l<n;l++)

{

for(j=0;j<n;j++)

{

if(result[l].processID==pcreate[j].processID)

{

printf(" %d %d %d

%d\n\n",result[l].processID,pcreate[j].arrivalTime,pcreate[j].burstTime,result[l].waitTime); break;

break;

}

}

avg\_WaitTime+=(result[l].waitTime);

}

printf("AVERAGE WAITING TIME :%f\n",avg\_WaitTime/n);

}

# Problem in terms of Operating System :-

**What is CPU Scheduling**

CPU scheduling is a process which allows one process to use the CPU while the execution of another process. The aim of CPU scheduling is to make the system efficient, fast and fair.

# Preemptive Priority scheduling :-

If the new process arrived at the ready queue has a higher priority than the currently running process,theCPUispreempted,whichmeanstheprocessingofthecurrentprocessisstopped and the incoming new process with higher priority gets the CPU for itsexecution.

|  |  |  |  |
| --- | --- | --- | --- |
| **Process Id** | **Priority** | **Arrival Time** | **Burst Time** |
| 1 | 2 | 0 | 1 |
| 2 | 6 | 1 | 7 |
| 3 | 3 | 2 | 3 |
| 4 | 5 | 3 | 6 |
| 5 | 4 | 4 | 5 |
| 6 | 10 | 5 | 15 |
| 7 | 9 | 15 | 8 |

process is P6 with the least priority, the Operating System has no choice unless of executing it. This will be executed at the last.



# Algorithm:

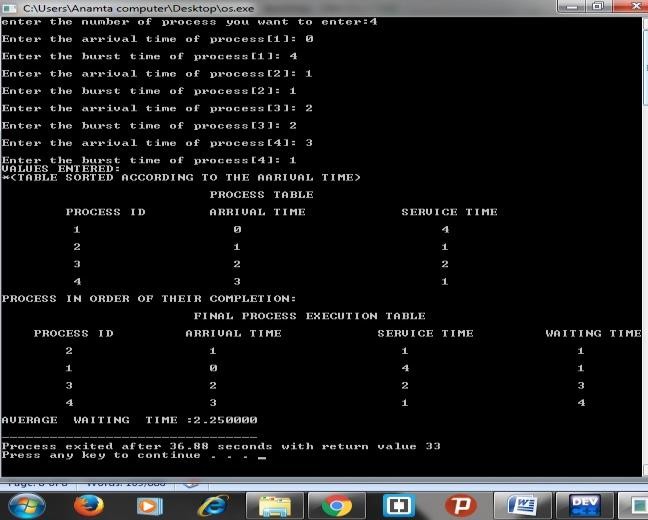
1. Start theprocess
2. Accept the userinput
   1. Take the number ofprocess
   2. Take the arrival time of theprocess
   3. Take the burst time of theprocess
3. Check if the process is in theexecution
   1. Change the priority rate byOne
4. If process is in readyqueue
   1. Change the priority rate byTwo
5. If process completed, put it in result queue got-to step 2 and compare the arrival timewith previous one
6. Sort the result queue in arrival time and if the that is tie go with bursttime
7. Print the step6.

**Complexity Analysis :**

1. Complexity of the starting the process is null
2. Accepting the user input has the complexity O(n) as iterate ntimes
3. Check if the process is in the execution by using **if** statement having the constant complexity and the one **for** loop makesO(n)
4. Check if the process is inready queue similar to the step 3 makes itO(n)
5. Check the completion of the process using **if** and **for** loop givesO(n)
6. Sorted with O(n^2) and check for tie with constantcomplexity

**Constraints:**

1. If high priority processes take lots of CPU time, then the lower priority processes may starve and will be postponed for an indefinitetime
2. AprocesswillbeblockedwhenitisreadytorunbuthastowaitfortheCPUbecausesome other process is runningcurrently
3. Ifanewhigherpriorityprocesskeepsoncominginthereadyqueue,thentheprocesswhich is in the waiting state may need to wait for a long duration of time.
4. If the system eventually crashes, all low priority processes getlost.



**Boundary value :**

1. Boundary value will be the maximum number of process we can take in queue in is 100 , And gives result up to 100 resultsonly.
2. First all the priority issame.
3. Priority rate can change by 1 or 2only.
4. We can only take the integer value in Cprogramming.

**Test cases :**

We can take any test case with in boundary to get a well -understandable output but I am taking the input from the given question

|  |  |  |
| --- | --- | --- |
| ProcessID | Arrival Time | Service Time |
| P1 | 0 | 4 |
| P2 | 1 | 1 |
| P3 | 2 | 2 |
| P4 | 3 | 1 |

By providing the above input The average time is: **2.25**