**ASSIGNMENT REPORT**

**Course Name: Operating Systems**

**Course Code: CSE 316**

**STUDENT NAME:** Utkarsh Upadhyay

**STUDENT ID:** 11712576

**SECTION:** K17WX

**GROUP:** 2

**ROLL NO.:** RK17WXB54



PROBLEM 12:

Design a scheduler with multilevel queue having two queues which will schedule the processes on the basis of pre-emptive shortest remaining processing time first algorithm (SROT) followed by a scheduling in which each process will get 2 units of time to execute. Also note that queue 1 has higher priority than queue 2. Consider the following set of processes (for reference)with their arrival times and the CPU burst times in milliseconds.

Process Arrival Time Burst Time

P1 0 5

P2 1 3

P3 2 3

P4 4 1

----------------------------------------

Calculate the average turnaround time and average waiting time for each process. The

input for number of processes and their arrival time, burst time should be given by the user.

DESCRIPTION OF MULTILEVEL QUEUE SCHEDULING

It may happen that processes in the ready queue can be divided into different classes where each class has its own scheduling needs. For example, a common division is a **foreground (interactive)** process and **background (batch)** processes.These two classes have different scheduling needs. For this kind of situation Multilevel Queue Scheduling is used.Now, let us see how it works.

**Ready Queue** is divided into separate queues for each class of processes. For example, let us take three different types of process System processes, Interactive processes and Batch Processes. All three process have there own queue.

**Scheduling among the queues :** What will happen if all the queues have some processes? Which process should get the cpu? To determine this Scheduling among the queues is necessary. There are two ways to do so –

1. **Fixed priority preemptive scheduling method –** Each queue has absolute priority over lower priority queue. Let us consider following priority order **queue 1 > queue 2 > queue 3**.According to this algorithm no process in the batch queue(queue 3) can run unless queue 1 and 2 are empty. If any batch process (queue 3) is running and any system (queue 1) or Interactive process(queue 2) entered the ready queue the batch process is preempted.
2. **Time slicing** – In this method each queue gets certain portion of CPU time and can use it to schedule its own processes.For instance, queue 1 takes 50 percent of CPU time queue 2 takes 30 percent and queue 3 gets 20 percent of CPU time.

SCHEDULING POLICIES USED:

**SRTF:**

**In this 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.**

**Advantage:**

**1- Short processes are handled very quickly.**

**2- The system also requires very little overhead since it only makes a decision when a process completes or a new process is added.**

**3- When a new process is added the algorithm only needs to compare the currently executing process with the new process, ignoring all other processes currently waiting to execute.**

**Disadvantage:**

**1- Like shortest job first, it has the potential for process starvation.**

**2- Long processes may be held off indefinitely if short processes are continually added.**

**ROUND ROBIN:**

**Round Robin is a CPU scheduling algorithm where each process is assigned a fixed time slot in a cyclic way.**

**It is simple, easy to implement, and starvation-free as all processes get fair share of CPU.**

**One of the most commonly used technique in CPU scheduling as a core.**

**It is preemptive as processes are assigned CPU only for a fixed slice of time at most.**

**The disadvantage of it is more overhead of context switching.**

CODE:

#include<stdio.h>

#include<pthread.h>

#include<unistd.h>

#include<stdlib.h>

#include<string.h>

int main()

{

int at[10],bt[10],rt[10],endTime,i,smallest;

int remain=0,n,time,sum\_wait=0,sum\_turnaround=0;

printf("..................................MULTILEVEL SCHEDULER.................................\n");

printf("Please Read the following Details and Enter The Details Carefully \n\n");

printf("Enter no of Processes : ");

scanf("%d",&n);

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

{

printf("Enter Arrival time for Process P%d : ",i+1);

scanf("%d",&at[i]);

printf("Enter Burst time for Process P%d : ",i+1);

scanf("%d",&bt[i]);

rt[i]=bt[i];

}

printf("\n\nProcess\t|Turnaround Time| Waiting Time\n\n");

rt[9]=9999;

for(time=0;remain!=n;time++)

{

smallest=9;

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

{

if(at[i]<=time && rt[i]<rt[smallest] && rt[i]>0)

{

smallest=i;

}}

rt[smallest]--;

if(rt[smallest]==0)

{

remain++;

endTime=time+1;

printf("\nP[%d]\t|\t%d\t|\t%d",smallest+1,endTime-at[smallest],endTime-bt[smallest]-at[smallest]);

sum\_wait+=endTime-bt[smallest]-at[smallest];

sum\_turnaround+=endTime-at[smallest];

}

}

printf("\n\nAverage waiting time = %f\n",sum\_wait\*1.0/n);

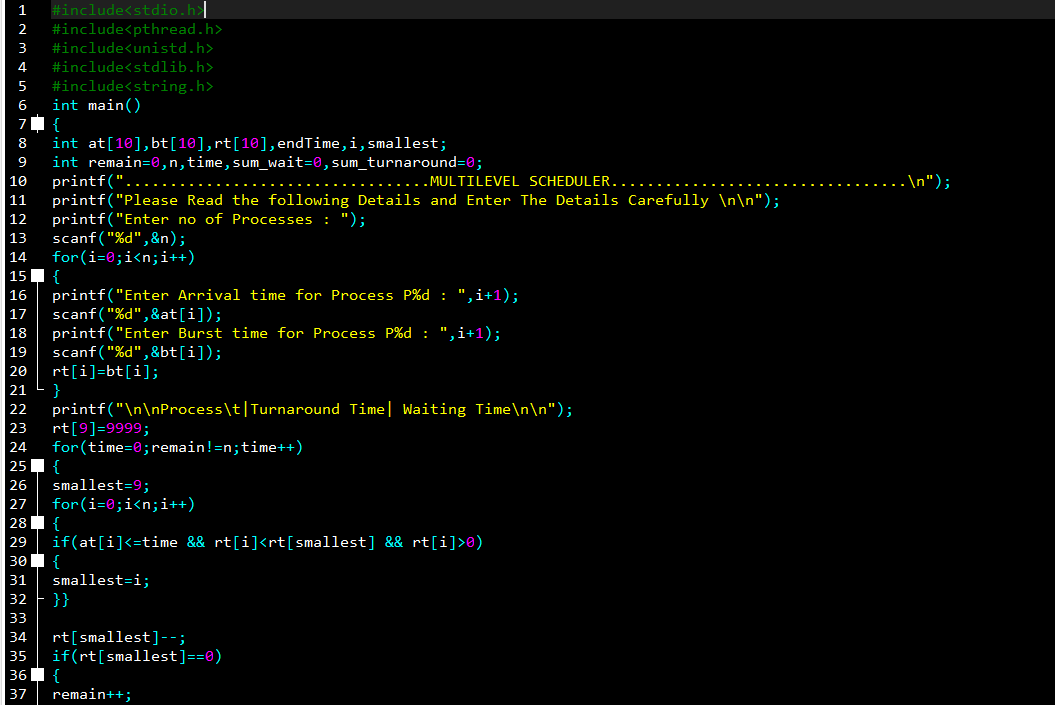
printf("Average Turnaround time = %f",sum\_turnaround\*1.0/5);

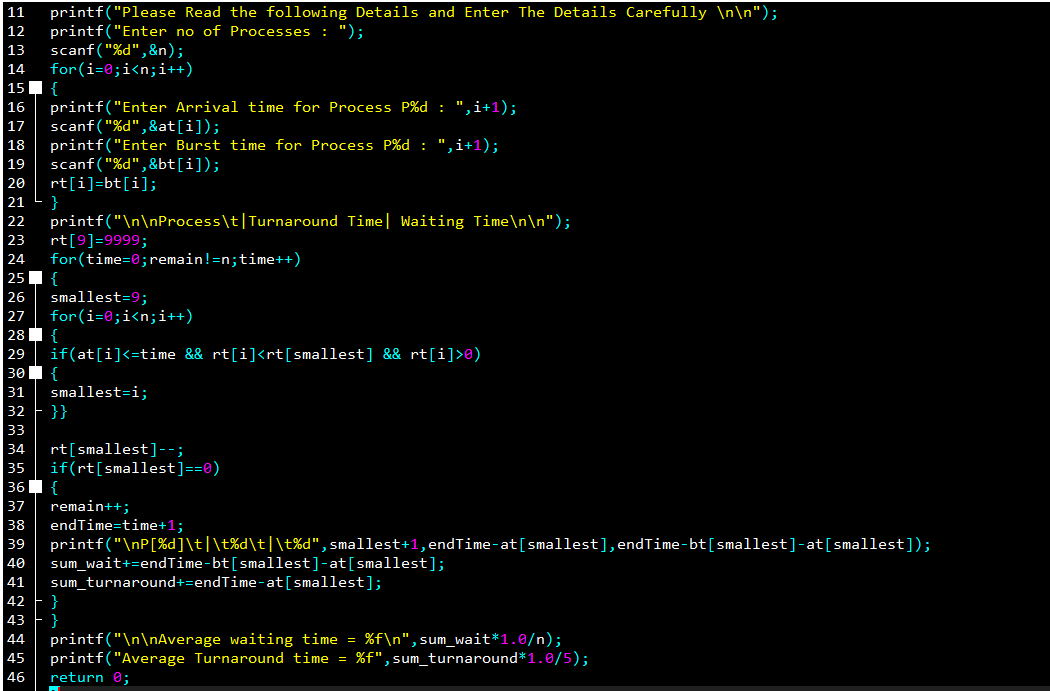
return (0);

}

CONSTRAINTS:

* The input numbers should be positive integers.
* Both srtf and round robin are applied.





BOUNDARY CONDITIONS:

* The burst time values should be integer.
* A maximum of 10 processes can be taken at a time.
* Srtf must be top priority.

COMPLEXITY: O(N)

TEST CASES:

