Simulation of Preemptive Process Scheduling Algorithms

LAB # 11

Spring 2023

CSE-204L Operating Systems Lab

Submitted by: **Ali Asghar**

Registration No.: **21PWCSE2059**

Class Section: **C**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Submitted to:

Engr. Madiha Sher

Date:

26th June 2023

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

IMPLEMENT THE ROUND-ROBIN SCHEDULING ALGORITHM.

PURPOSE:

A Round Robin Scheduler algorithm is designed especially for time sharing systems. It is similar to FCFS scheduling but preemption is added to switch between processes

Preemption: The act of interrupting a currently running task in order to give time to another task.

DESCRIPTION:

To implement Round robin scheduling we keep the ready queue as a FIFO queue of processes. New processes are added to the tail of the ready queue. The CPU scheduler picks the first process from the ready queue, sets a **timer to interrupt** after one time quantum and dispatches the process. A small unit of time called a time quantum or time slice is defined. A time quantum is generally from 10 to 100 milliseconds.

The process may have CPU burst of less than one time quantum. In this case the process itself will release the CPU voluntarily .The scheduler will then proceed to the next process in the ready queue. Otherwise if the CPU burst of the currently running process is longer than 1 time quantum, the timer will go off and will cause

an interrupt to the operating system. The average waiting time under Round Robin policy is how ever quite long.

Task 1: Take Arrival Time = 0 and Burst Time = Multiple of 5.

Code:

```c
#include<stdio.h>

#define AT 1
#define BT 2
#define ST 3
#define ET 4
#define WT 5
#define TAT 6
#define RT 7

void get_input(int n, float p[][10]){

    for(int i =0; i<n; i++){
        printf("Enter process %d burst time\n", i+1);
        scanf("%f", &p[i][BT]);
        p[i][RT] = p[i][BT];
    }
}


void initialize_arr(int n, float p[][10]){

    for(int i =0; i<n; i++){
        p[i][0] = i;
        p[i][AT]=0;
        p[i][ET]=0;
        p[i][ST] = 0;
    }
}
```

```c
}

void print_array(int n, float p[][10]){
        printf("Job Number:\tArrival time\tburst time:\tStart time:\tEnd time:\tWait time:\tTurnAround time:\tRemaining time\n");

    for(int i =0; i<n; i++){
        printf("-----%.1f-----\t-----%.1f------\t-----%.1f-----\t-----%.1f-----\t-----%.1f-----\t-----%.1f-----\t-----%.1f-----\t-----%.1f-----\t-----%.1f-----\n", p[i][0], p[i][AT], p[i][BT],  p[i][ST], p[i][ET], p[i][WT], p[i][TAT], p[i][RT]);

    }
}


void rr(int n, float p[][10], int q, int k){


    float t=0;

    for(int i=0; i<6; i++){
        for(int j=0; j< n; j++){

            if(p[j][RT] > 0){
                if(i == 0){
                    p[j][ST] = t;
                }

                    p[j][RT] -= q;
```

Bracket match found on line: 21

```c
                    p[j][RT] -= q;
                    t += q;

                if(p[j][RT] <= 0){
                    p[j][ET] = t; //End Time
                    p[j][WT] = p[j][ET] - p[j][BT]- p[j][AT];//Wait Time = End Time - Burst Time + Arrival Time
                    p[j][TAT] = p[j][WT] + p[j][BT]; //TurnAround Time = Burst Time + Wait Time
                }
            }
        }
    }
    printf("\nTOTAL TIME:%.2f\n",t);

}


void calculate(int n, float p[][10], float* a1, float* a2){


    //Wait Time and Turn Around Time Calculation
    for(int i = 0; i<n; i++){
        *(a1) += p[i][WT];
        *(a2)  += p[i][TAT];
    }
        *(a1) /= n;
        *(a2)  /= n;
}
```

```c
    for(int i = 0; i<n; i++){
        *(a1) += p[i][WT];
        *(a2)  += p[i][TAT];
    }
        *(a1) /= n;
        *(a2)  /= n;
}


int main(){

    int n;
    float avgwt , avgtat;

    printf("Enter the total number of jobs\n");
    scanf("%d", &n);

    float processes[n][10];

    get_input(n, processes);
    initialize_arr(n, processes);
    //int k = longest_job_index(n, processes);
    rr(n, processes,5,0);
    print_array(n, processes);
    calculate(n, processes, &avgwt, &avgtat);
    printf("Average Waiting Time:%.2f\n",avgwt);
    printf("Average Turn Around Time:%.2f\n",avgtat);
    return 0;
}
```

Output:

```
ali@Ubuntu22:~/Desktop/OS Lab/Lab11$ gcc t1.c -o t1.o
ali@Ubuntu22:~/Desktop/OS Lab/Lab11$ ./t1.o
Enter the total number of jobs
5
Enter process 1 burst time
15
Enter process 2 burst time
30
Enter process 3 burst time
5
Enter process 4 burst time
10
Enter process 5 burst time
20

TOTAL TIME:80.00
Job Number:     Arrival time    burst time:     Start time:     End time:       Wait time:      TurnAround time:        Remaining time
-----0.0-----   -----0.0------  -----15.0-----  -----0.0-----   -----50.0-----  -----35.0-----  -----50.0-----  -----0.0-----
-----1.0-----   -----0.0------  -----30.0-----  -----5.0-----   -----80.0-----  -----50.0-----  -----80.0-----  -----0.0-----
-----2.0-----   -----0.0------  -----5.0-----   -----10.0-----  -----15.0-----  -----10.0-----  -----15.0-----  -----0.0-----
-----3.0-----   -----0.0------  -----10.0-----  -----15.0-----  -----40.0-----  -----30.0-----  -----40.0-----  -----0.0-----
-----4.0-----   -----0.0------  -----20.0-----  -----20.0-----  -----70.0-----  -----50.0-----  -----70.0-----  -----0.0-----
Average Waiting Time:35.00
Average Turn Around Time:51.00
ali@Ubuntu22:~/Desktop/OS Lab/Lab11$
```

Task 2: Take Arrival Time other than 0 and Burst Time other than Multiple of 5.

Code:

t1.c t2.c t1.c

```c
#include<stdio.h>

void main()
{
    // initlialize the variable name
    int i, NOP, sum=0,count=0, y, quant, wt=0, tat=0, at[10], bt[10], temp[10];
    float avg_wt, avg_tat;
    printf(" Total number of process in the system: ");
    scanf("%d", &NOP);
    y = NOP; // Assign the number of process to variable y

    // Use for loop to enter the details of the process like Arrival time and the Burs
    for(i=0; i<NOP; i++)  {
        printf("\n Enter the Arrival and Burst time of the Process[%d]\n", i+1);
        printf(" Arrival time is: \t");  // Accept arrival time
        scanf("%d", &at[i]);
        printf(" \nBurst time is: \t"); // Accept the Burst time
        scanf("%d", &bt[i]);
        temp[i] = bt[i]; // store the burst time in temp array
    }
    // Accept the Time qunat
    printf("Enter the Time Quantum for the process: \t");
    scanf("%d", &quant);
```

t1.c t2.c t1.c rr1.c

```c
    printf("Enter the Time Quantum for the process: \t");
    scanf("%d", &quant);
    // Display the process No, burst time, Turn Around Time and the waiting time
    printf("\n Process No \t\t Burst Time \t\t TAT \t\t Waiting Time ");
    for(sum=0, i = 0; y!=0; ) {
    if(temp[i] <= quant && temp[i] > 0) {
        sum = sum + temp[i];
        temp[i] = 0;
        count=1;
    }
    else if(temp[i] > 0)  {
        temp[i] = temp[i] - quant;
        sum = sum + quant;
    }
    if(temp[i]==0 && count==1) {
        y--; //decrement the process no.
        printf("\nProcess No[%d] \t\t %d\t\t\t\t %d\t\t\t %d", i+1, bt[i], sum-at[i], sum-at[i]-bt[i]);
        wt = wt+sum-at[i]-bt[i];
        tat = tat+sum-at[i];
        count =0;
    }
    if(i==NOP-1)
    {
```

Activate Windows

```c
        printf("\nProcess No[%d] \t\t %d\t\t\t\t %d\t\t\t %d", i+1, bt[i], sum-at[i], sum-at[i]-bt[i]);
        wt = wt+sum-at[i]-bt[i];
        tat = tat+sum-at[i];
        count =0;
    }
    if(i==NOP-1)
    {
        i=0;
    }
    else if(at[i+1]<=sum)
    {
        i++;
    }
    else
    {
        i=0;
    }
}
    // represents the average waiting time and Turn Around time
    avg_wt = wt * 1.0/NOP;
    avg_tat = tat * 1.0/NOP;
    printf("\n Average Turn Around Time: \t%f", avg_wt);
    printf("\n Average Waiting Time: \t%f", avg_tat);
}
```

Output:

```
ali@Ubuntu22:~/Desktop/OS Lab/Lab11$ ./rr1.o
 Total number of process in the system: 5

 Enter the Arrival and Burst time of the Process[1]
 Arrival time is:       3

Burst time is:  4

 Enter the Arrival and Burst time of the Process[2]
 Arrival time is:       3

Burst time is:  5

 Enter the Arrival and Burst time of the Process[3]
 Arrival time is:       2

Burst time is:  5

 Enter the Arrival and Burst time of the Process[4]
 Arrival time is:       6

Burst time is:  2

 Enter the Arrival and Burst time of the Process[5]
 Arrival time is:       7

Burst time is:  1
Enter the Time Quantum for the process:         5

 Process No              Burst Time              TAT             Waiting Time
Process No[1]            4                               1                       -3
Process No[2]            5                               6                       1
Process No[3]            5                               12                      7
Process No[4]            2                               10                      8
Process No[5]            1                               10                      9
 Average Turn Around Time:      4.400000
 Average Waiting Time:  7.800000ali@Ubuntu22:~/Desktop/OS Lab/Lab11$
```