Name: K Srikanth Roll Number: 17ETCS002124

Laboratory 4

Title of the Laboratory Exercise: Programs for process scheduling algorithms

1. Introduction and Purpose of Experiment

   A Process Scheduler schedules different processes to CPU based on particular scheduling algorithms. There are various scheduling algorithms present in each group of operating system. By solving these problems students will be able use different scheduling algorithms as part of their implementation

2. Aim and Objectives

   Aim

   - To develop programs to implement scheduling algorithms

   Objectives

   At the end of this lab, the student will be able to

   - Distinguish different scheduling algorithms
   - Apply the logic of scheduling algorithms wherever required
   - Create C programs to simulate scheduling algorithms

3. Experimental Procedure

   i. Analyse the problem statement

   ii. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code

   iii. Implement the algorithm in C language

   iv. Compile the C program

   v. Test the implemented program

   vi. Document the Results

   vii. Analyse and discuss the outcomes of your experiment

4. Questions

   Write a multithreaded program to simulate the following process scheduling algorithms. Calculate average waiting time and average turnaround time for processes under each scheduling algorithm by separate threads

Instructions: Assume all the processes arrive at the same time. For round robin scheduling algorithm, read the number of processes in the system, their CPU burst times and the size of the time slice. For priority scheduling algorithm, read the number of processes in the system, their CPU burst times and the priorities.

a) Priority

b) Round Robin

5. Calculations/Computations/Algorithms

Priority Scheduling Function

```
1. Start
2. Swap Function
        a. Temp = a
        b. a = b
        c. b = temp
3.  Priority function
        1. Start
        2. Declaring Variables type int and array
        3. Read  total number of processes
        4. for i=0 to N-1
                a. max=i
                b. for j=0 to N-1
                     i . if priority[j]> priority[max] then max=j
                c. swap(priority[i], priority[max])
                d. swap(burst_time[I],burst_time[max]
                e. swap(Process[I],Process[max])
        5. waiting_time[0]=0
        6. turnaround_time[0]=burst_time[0]
```

```
7. for i=0, to N-1
    a. waiting_time[i]=burst_time[i-1]+waiting_time[i-1].
    b. total_waiting_time+=waiting_time[I].
    c. turnaround_time[I]=burst_time[i]+waiting_time[i].
    d. total_turnaround_time+=turnaround_time[i].
8. Display process id, Burst time, waiting time, Turnaround time.
9. Display Average Waiting Time.
10. Display Average Turnaround Time.
11. Stop
```

4. Round Robin Scheduling

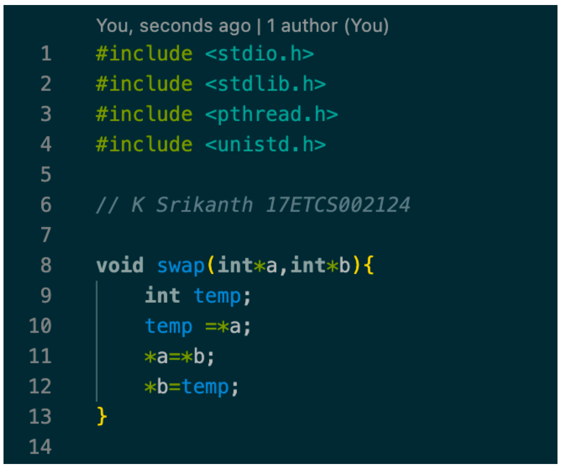
```
1. Start
2. Declare Variables
3. Read total number of processes
4. Read burst time of each process
5. For time=0,i=0,remain_process!=0
    a. If (running_time[I]<=time_slice and running_time[i]>0 )
        i. time +=running_time[I]
        ii. running_time[I]=0
        iii. flag=1.
    b. Else if (running_time[i]>0 )
        i. running_time[i]=time_slice
        ii. time+=time_slice
    c. If (running_time[i]==0 and flag==1)
        i . remain_process—
        ii. Display process id, Burst time, waiting time, Turnaround time.
        iii. wait_time+=time-burst_t[i].
```

```
                    iv. t_around_time+=time.
                    v. flag=0.
                d. If (i==n-1)
                    i . i=0.
                e. Else
                    i . i++
            6. Display Average Waiting Time.
            7. Display Average Turnaround Time.
            8. Stop
5. Main Function
            1. Start
            2. Creating two threads, thread1,thread2
            3. Create thread process for Priority Scheduling with thread 1 and priorityScheduling
            4. Join thread 1
            5. Create thread process for Round Robin with thread 2 and roundRobinScheduling.
            6. Join thread 2
            7. Return 0
            8. Stop
6. Stop
```

Name: K Srikanth   Roll Number: 17ETCS002124

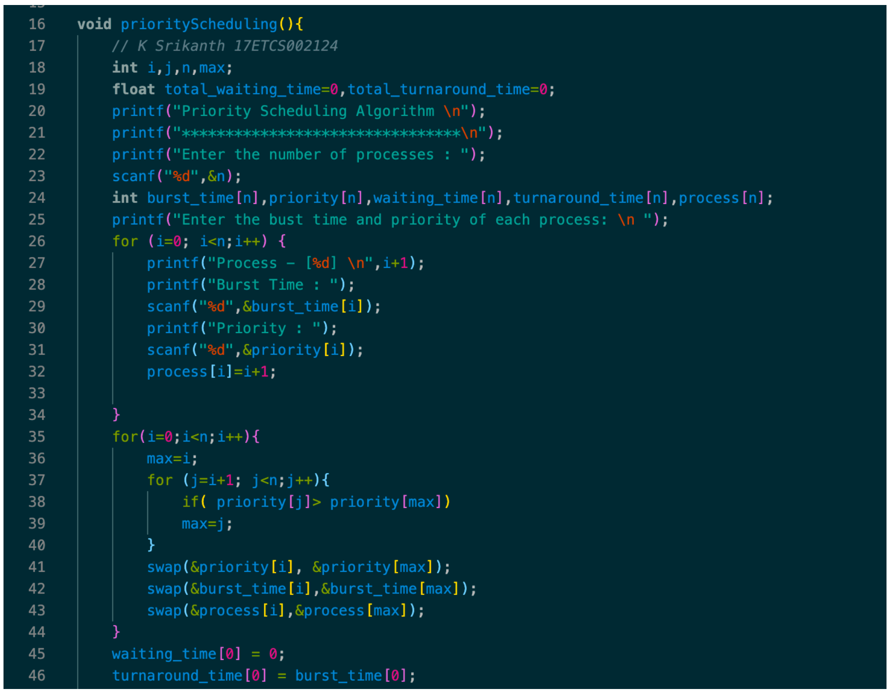
Priority

Code



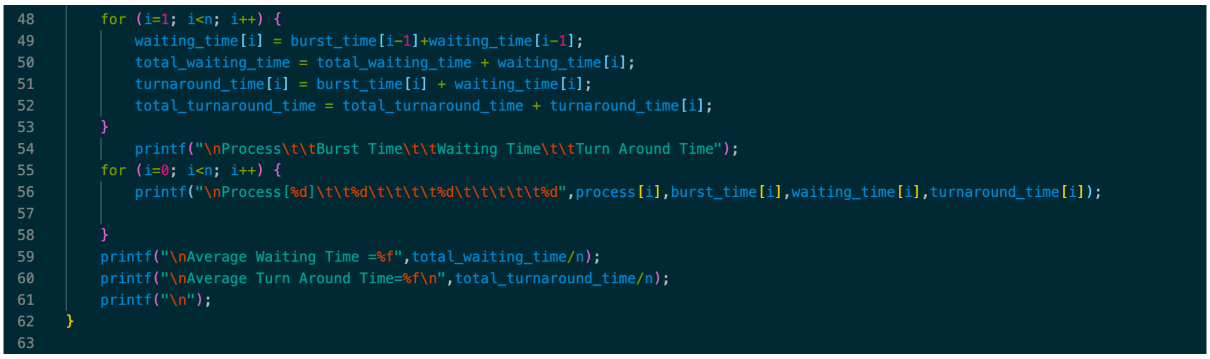
```
You, seconds ago | 1 author (You)
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>

// K Srikanth 17ETCS002124

void swap(int*a,int*b){
    int temp;
    temp =*a;
    *a=*b;
    *b=temp;
}

```

Figure 1 C Program for Swap Function



```
void priorityScheduling(){
    // K Srikanth 17ETCS002124
    int i,j,n,max;
    float total_waiting_time=0,total_turnaround_time=0;
    printf("Priority Scheduling Algorithm \n");
    printf("********************************\n");
    printf("Enter the number of processes : ");
    scanf("%d",&n);
    int burst_time[n],priority[n],waiting_time[n],turnaround_time[n],process[n];
    printf("Enter the bust time and priority of each process: \n ");
    for (i=0; i<n;i++) {
        printf("Process - [%d] \n",i+1);
        printf("Burst Time : ");
        scanf("%d",&burst_time[i]);
        printf("Priority : ");
        scanf("%d",&priority[i]);
        process[i]=i+1;

    }
    for(i=0;i<n;i++){
        max=i;
        for (j=i+1; j<n;j++){
            if( priority[j]> priority[max])
            max=j;
        }
        swap(&priority[i], &priority[max]);
        swap(&burst_time[i],&burst_time[max]);
        swap(&process[i],&process[max]);
    }
    waiting_time[0] = 0;
    turnaround_time[0] = burst_time[0];
```

Figure 2 C Program for Priority Scheduling Function

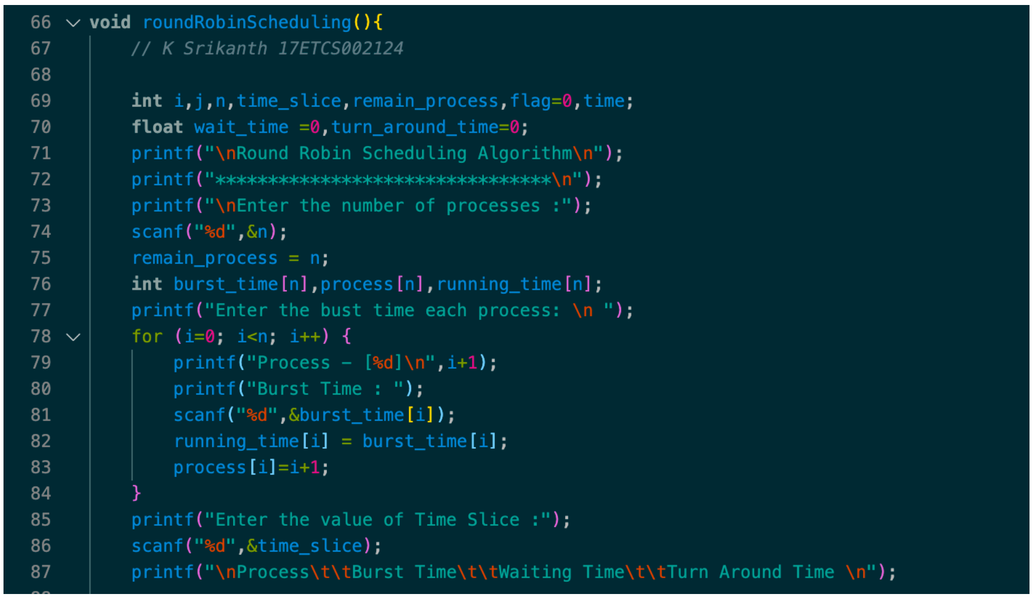


```
    for (i=1; i<n; i++) {
        waiting_time[i] = burst_time[i-1]+waiting_time[i-1];
        total_waiting_time = total_waiting_time + waiting_time[i];
        turnaround_time[i] = burst_time[i] + waiting_time[i];
        total_turnaround_time = total_turnaround_time + turnaround_time[i];
    }
        printf("\nProcess\t\tBurst Time\t\tWaiting Time\t\tTurn Around Time");
    for (i=0; i<n; i++) {
        printf("\nProcess[%d]\t\t%d\t\t\t\t%d\t\t\t\t\t%d",process[i],burst_time[i],waiting_time[i],turnaround_time[i]);

    }
    printf("\nAverage Waiting Time =%f",total_waiting_time/n);
    printf("\nAverage Turn Around Time=%f\n",total_turnaround_time/n);
    printf("\n");
}

```

Figure 3 Continued C Program for Priority Scheduling Function

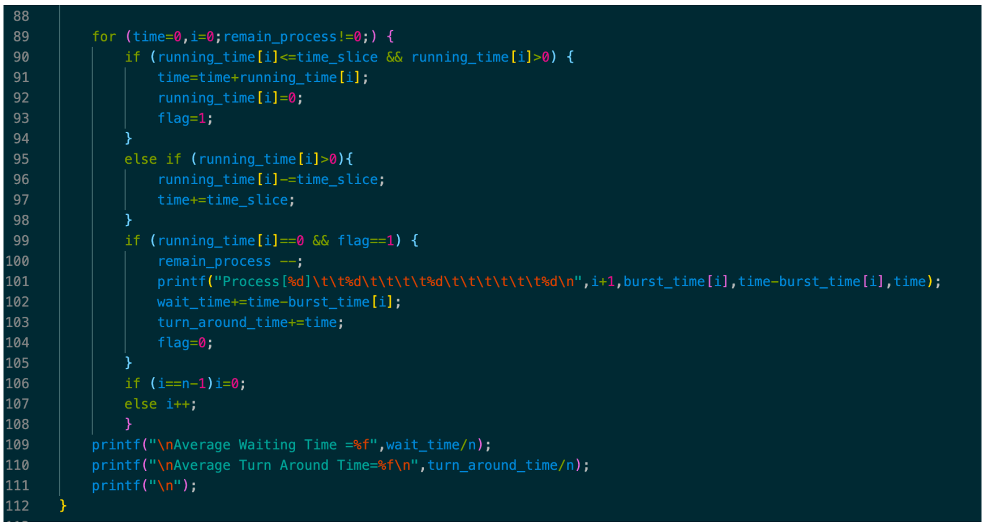
Round Robin

Code



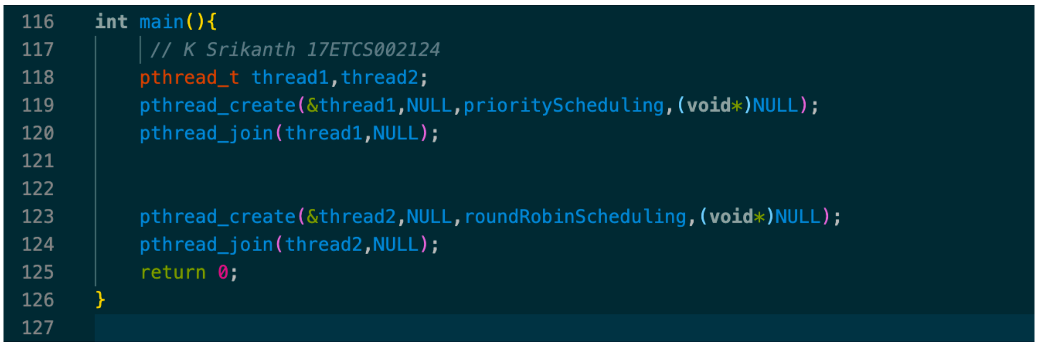
```c
void roundRobinScheduling(){
    // K Srikanth 17ETCS002124

    int i,j,n,time_slice,remain_process,flag=0,time;
    float wait_time =0,turn_around_time=0;
    printf("\nRound Robin Scheduling Algorithm\n");
    printf("********************************\n");
    printf("\nEnter the number of processes :");
    scanf("%d",&n);
    remain_process = n;
    int burst_time[n],process[n],running_time[n];
    printf("Enter the bust time each process: \n ");
    for (i=0; i<n; i++) {
        printf("Process - [%d]\n",i+1);
        printf("Burst Time : ");
        scanf("%d",&burst_time[i]);
        running_time[i] = burst_time[i];
        process[i]=i+1;
    }
    printf("Enter the value of Time Slice :");
    scanf("%d",&time_slice);
    printf("\nProcess\t\tBurst Time\t\tWaiting Time\t\tTurn Around Time \n");
```

Figure 4 C Program for Round Robin Function



```c

    for (time=0,i=0;remain_process!=0;) {
        if (running_time[i]<=time_slice && running_time[i]>0) {
            time=time+running_time[i];
            running_time[i]=0;
            flag=1;
        }
        else if (running_time[i]>0){
            running_time[i]-=time_slice;
            time+=time_slice;
        }
        if (running_time[i]==0 && flag==1) {
            remain_process --;
            printf("Process[%d]\t\t%d\t\t\t\t%d\t\t\t\t\t\t%d\n",i+1,burst_time[i],time-burst_time[i],time);
            wait_time+=time-burst_time[i];
            turn_around_time+=time;
            flag=0;
        }
        if (i==n-1)i=0;
        else i++;
        }
    printf("\nAverage Waiting Time =%f",wait_time/n);
    printf("\nAverage Turn Around Time=%f\n",turn_around_time/n);
    printf("\n");
}
```

Figure 5 Continued C Program for Round Robin Function

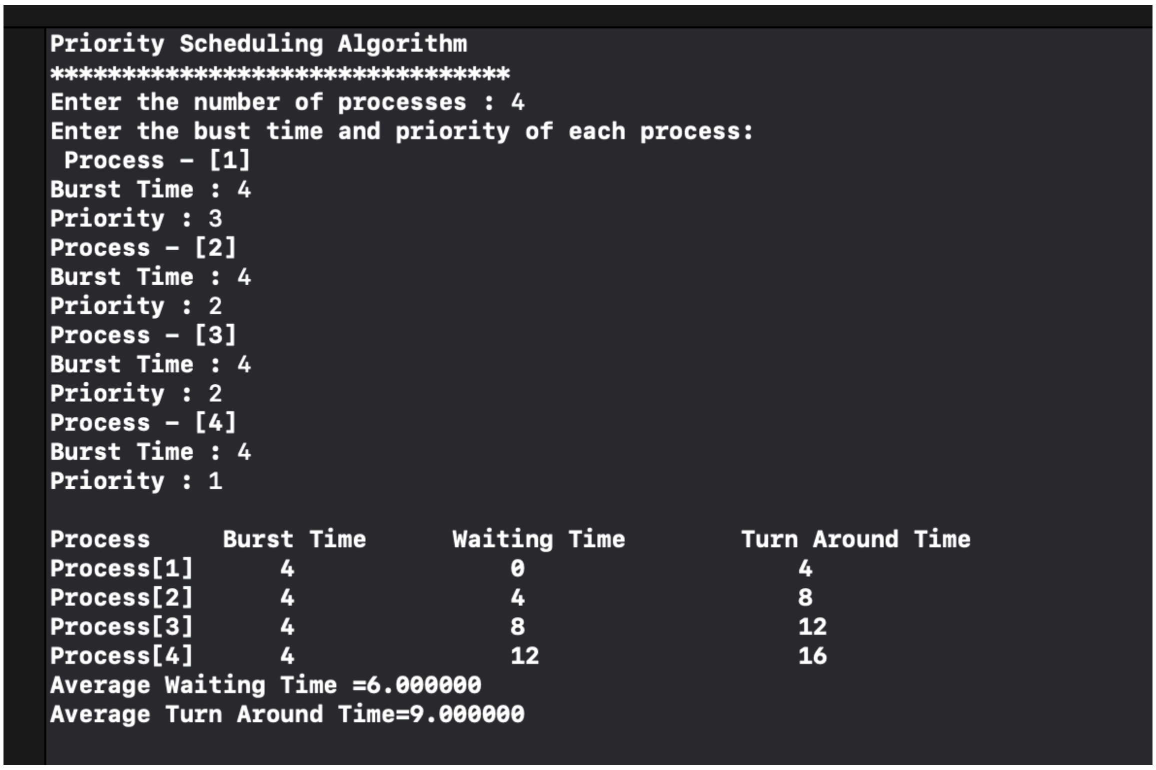


```c
int main(){
    // K Srikanth 17ETCS002124
    pthread_t thread1,thread2;
    pthread_create(&thread1,NULL,priorityScheduling,(void*)NULL);
    pthread_join(thread1,NULL);


    pthread_create(&thread2,NULL,roundRobinScheduling,(void*)NULL);
    pthread_join(thread2,NULL);
    return 0;
}
```

Figure 6 C Program for Main Function

6. Presentation of Results

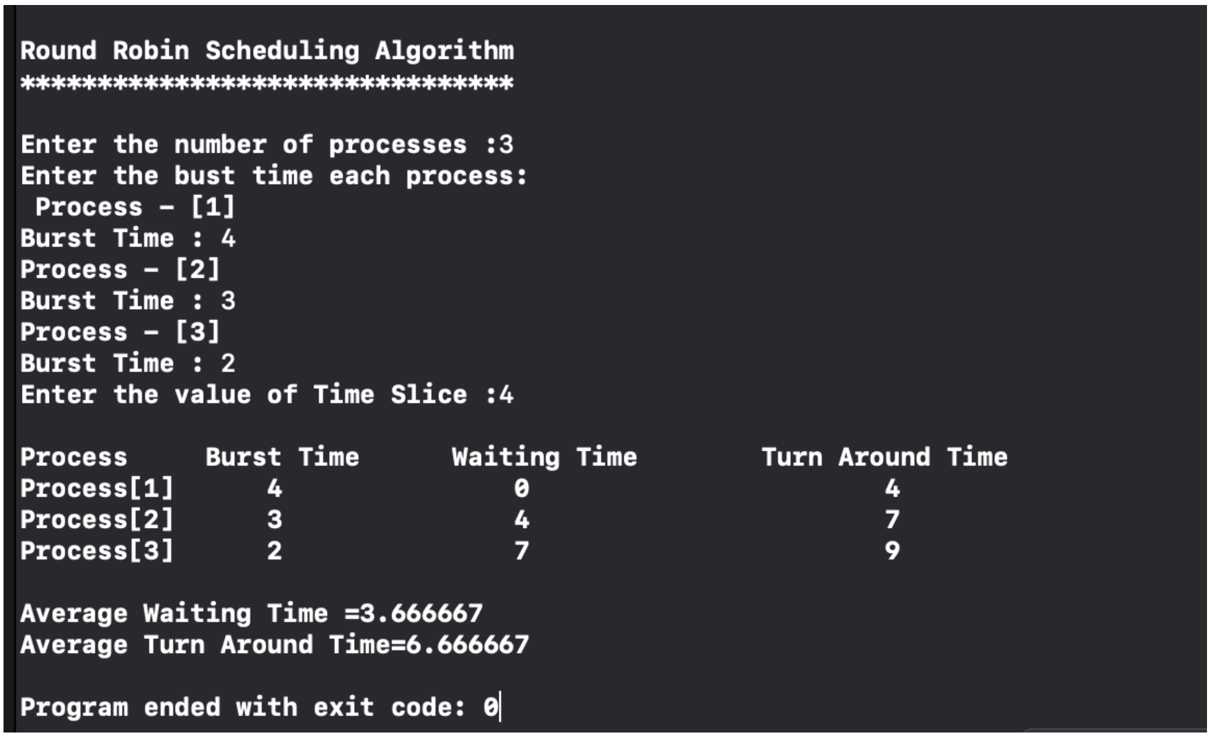
Priority Scheduling



```
Priority Scheduling Algorithm
********************************
Enter the number of processes : 4
Enter the bust time and priority of each process:
 Process - [1]
Burst Time : 4
Priority : 3
Process - [2]
Burst Time : 4
Priority : 2
Process - [3]
Burst Time : 4
Priority : 2
Process - [4]
Burst Time : 4
Priority : 1

Process     Burst Time      Waiting Time        Turn Around Time
Process[1]      4               0                   4
Process[2]      4               4                   8
Process[3]      4               8                   12
Process[4]      4               12                  16
Average Waiting Time =6.000000
Average Turn Around Time=9.000000
```

Figure 7 C Program output for Priority Scheduling using threads

Round Robin



```
Round Robin Scheduling Algorithm
********************************

Enter the number of processes :3
Enter the bust time each process:
 Process - [1]
Burst Time : 4
Process - [2]
Burst Time : 3
Process - [3]
Burst Time : 2
Enter the value of Time Slice :4

Process     Burst Time      Waiting Time        Turn Around Time
Process[1]      4               0                       4
Process[2]      3               4                       7
Process[3]      2               7                       9

Average Waiting Time =3.666667
Average Turn Around Time=6.666667

Program ended with exit code: 0
```

Figure 8 C Program output for Round Robin using threads

7. Analysis and Discussions

Priority Scheduling

Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems. Each process is assigned a priority. Process with highest priority is to be executed first and so on. Processes with same priority are executed on first come first served basis. Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Round Robin Scheduling:

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.

8. Conclusions

The priority of a process can be selected based on memory requirement, time requirement or user preference. For example, a high end game will have better graphics that means the process which updates the screen in a game will have higher priority so as to achieve better graphics performance.

In Round Robin, each process is served by the CPU for a fixed time quantum, so all processes are given the same priority. Starvation doesn't occur because for each round robin cycle, every process is given a fixed time to execute. No process is left behind. It is best suited for time sharing system, client server architecture and interactive system.

It’s better to use priority scheduling, if priorities are known at the beginning, instead at the time of execution. For Round Robin Scheduling smaller value of time quantum serves better in terms of response time.

9. Comments

1. Limitations

In preemptive priority scheduling, a higher priority process can execute ahead of an already executing lower priority process. If lower priority process keeps waiting for higher priority processes, starvation occurs. It also needs another scheduling algorithm is required to schedule the processes which have same priority.

Round Robin Scheduling leads to starvation for processes with larger burst time as they have to repeat the cycle many times. Its performance heavily depends on time quantum. It is more overhead of context switching.

2. Learning happened

Scheduling algorithms like priority and round robin, its implementation, Differences, and advantages of one over another.

3. Recommendations

The above program can be also implemented using structures, which could best utilise memory.