A New Scheduling Algorithm made by Combining Round Robin Algorithm and Shortest Job First Algorithm

J COMPONENT PROJECT REPORT

Submitted by

RAJAT JHA – 18BCE2013

OPERATING SYSTEM (CSE 2005)

PROJECT SUPERVISOR

Prof. GLADYS GNANA KIRUBA .B

NOVEMBER 2019

VIT UNIVERSITY VELLORE 632014

Abstract

Process Management is one of the most important task that is performed by the operating systems. The performance of a system depends on the scheduling algorithm used by the CPU. The main aim of the CP scheduling algorithms is to minimize the waiting time, turnaround time and response time and maximizing the CPU utilization. Some of the popular types of scheduling algorithms are First-Come-First-Served(FCFS), Shortest Job First(SJF), Priority Scheduling and Round Robin(RR). Shortest Job First algorithm is a CPU scheduling algorithm, in which the process which is having the minimum burst time is given the first priority. Due to this, this algorithm given the minimum turnaround and waiting time compared to other CPU scheduling algorithms. But the biggest disadvantage of this algorithm is that due to long waiting some processes may be starved thus causing the problem of starvation. Round Robin is a scheduling algorithm in which every process gets executed in a cyclic way. A time quantum is allotted to every process and if the execution of the processes gets completed in that time quantum, then the process will get terminate otherwise the process will again go back to the ready queue, and the previous process will wait for its turn to complete its execution. This algorithm is the fairest algorithm as it gives equal priority to each and every process but due to this the turnaround and waiting time increases. This project is based on making an improved Round Robin scheduling algorithm by using the concept of Shortest Job First algorithm so that the waiting time and the turnaround time can be minimized as compared to the conventional Round Robin scheduling algorithm. Thus, as the waiting time and turnaround time of the processes are decreased, it also decreases the total turnaround time, total waiting time, average turnaround time and average waiting time of the whole system and thus making the system fast.

Introduction

CPU scheduling is a process which allows one process to use the CPU while the execution of another process is on hold (in waiting state) due to unavailability of any resource like I/O etc., thereby making full use of CPU. The aim of CPU scheduling is to make the system efficient, fast and fair. The project is based on making a much more efficient CPU scheduling algorithm which would combine the concepts of Round Robin scheduling algorithm and Shortest Job First(SJF) scheduling algorithm. In Shortest Job First CPU scheduling algorithm, the process with the minimum burst time implements first due to which the average waiting time is the minimum in SJF as compared to all other conventional scheduling algorithms. Round Robin scheduling algorithm is a process-scheduling algorithm which is considered to be very fair compared to the other scheduling algorithms, as it uses time quantum and thus each process is allowed to use the CPU for a given amount of time, and if the process does not finishes it is put into waiting so that the next process which is line is able to use the resources of the CPU for the allotted time. The project features both situations of non-preemptive and preemptive case. In non-preemptive, only burst time of the processes are given and is assumed that all the processes have arrived at the same time whereas in preemptive, both the burst time and the arrival time of the processes are given. The performance parameters used in the project are turnaround time and waiting time. Turnaround Time is the time interval from the time of submission of the process to the time of completion of that processes. Waiting Time is the time a process waits for the resource allocation when several other processes are competing in a multiprogramming system. Thus, this new algorithm helps in minimizing the turnaround and waiting time and also solves the problem of starvation.

Literature Survey

1. AN IMPROVED ROUND ROBIN CPU SCHEDULING ALGORITHM WITH VARYING TIME QUANTUM Manish Kumar Mishra and Dr. Faizur Rashid

Abstract: This article aims at telling the readers about how round robin CPU scheduling algorithm can be combined with the Shortest Job First scheduling algorithm so that the waiting time and the turnaround time can be minimized as compared to the traditional Round Robin Scheduling algorithm.

2. Manish Kumar Mishra & Abdul Kadir Khan, (2012) "An Improved Round Robin CPU Scheduling Algorithm", Journal of Global Research in Computer Science, Vol. 3, No. 6, pp 64-69.

Abstract: This book tells us different types of ways through which the Round Robin Scheduling algorithm can be improved so that the performance of the CPU can be increased.

3. Rami J. Matarneh, (2009) "Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of Now Running Processes", American J. of Applied Sciences, Vol. 6, No. 10, pp1831- 1837.

Abstract: This book was used to understand the concept of self-adjustment time quantum in Round Robin Scheduling Algorithm which depends upon the burst time of the process which is currently running.

4. M.K. Srivastav, Sanjay Pandey, Indresh Gahoi & Neelesh Kumar Namdev, (2012) "Fair Priority Round Robin with Dynamic Time Quantum", International Journal of Modern Engineering Research, Vol. 2, Issue 3, pp 876-881

Abstract: This journal was referred so as to understand about the priority order which is followed during the Fair Priority Round Robin Scheduling Algorithm which is done with the Dynamic Time Quantum.

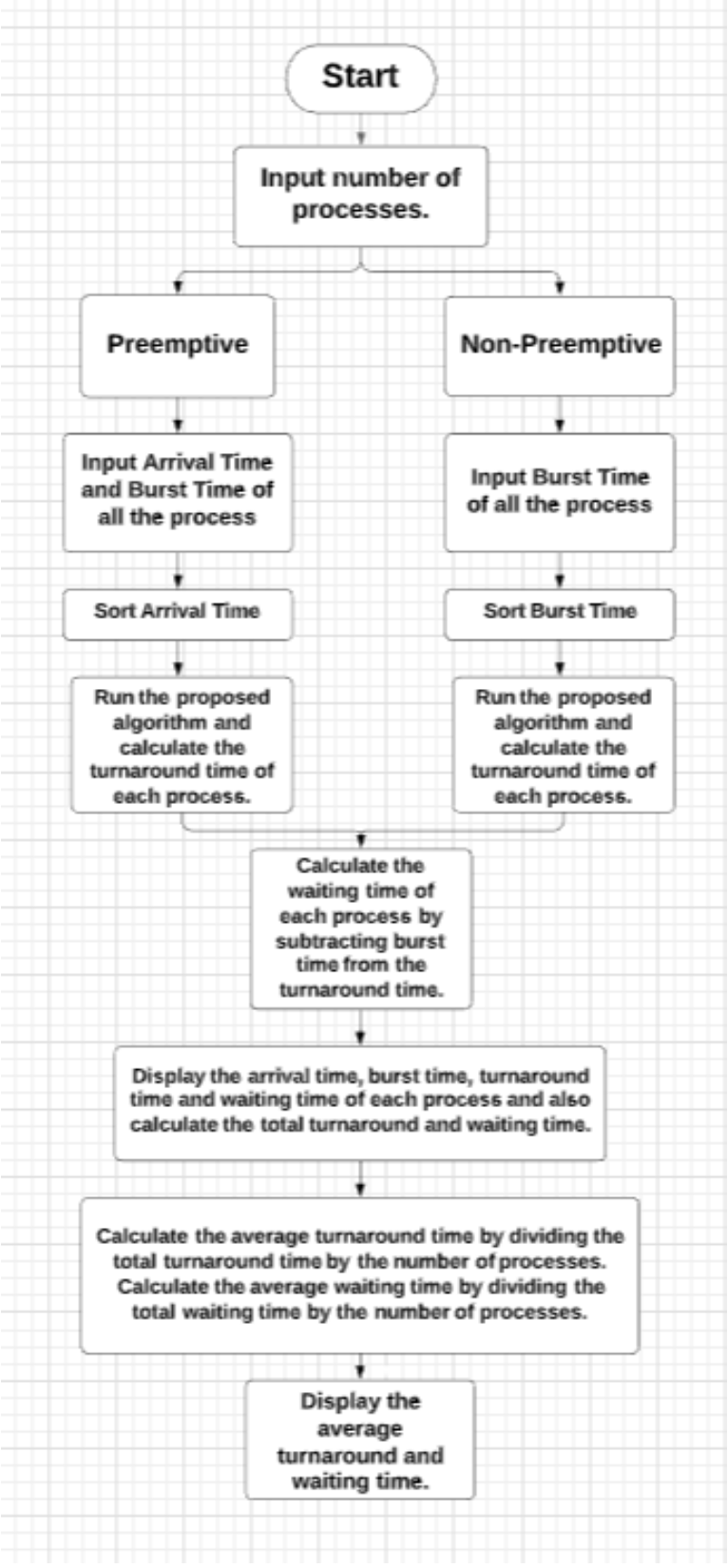
5. Operating Systems_ CPU Scheduling, http://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/5_CPU_Scheduling.html ,accessed 8th October 2013.

Abstract: This website was used to understand the concept of CPU scheduling and to know the various types of CPU scheduling algorithms and how to calculate the waiting time and the turnaround time.

6. Lalit Kishor, Dinesh Goyal, "Time Quantum Based Improved Scheduling Algorithm", Issue International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, 2013.

Abstract: This journal aims on the ways through which the waiting time and the turnaround time of a process can be drastically decreased with the help of Round Robin Scheduling Algorithm using Time Quantum so that the performance of the CPU can increase.

Design(Flowchart)



Module Description

Process management is one of the most important tasks that is performed by the operating system. The performance of any system depends on which CPU scheduling algorithm is used by the operating system. The main aim of the CPU scheduling algorithms is to minimize turnaround time and waiting time. Improper use of CPU can reduce the efficiency of the system in the multiprogramming systems. Thus, different types of CPU scheduling algorithms are used so that the resources of the CPU can be used efficiently and is not wasted.

There are many types of CPU scheduling algorithms but the most used are First Come First Served(FCFS), Shortest Job First(SJF), Priority Scheduling and the Round Robin Scheduling Algorithms. Each of the above has their advantages and disadvantages. This project uses the concepts the Shortest Job First(SJF) and Round Robin scheduling algorithms to make a much better algorithm which can overcome the disadvantages of the above two algorithms.

The algorithm works for both cases: preemptive(different arrival times) and non-preemptive(same arrival time). The performance parameters used in the above algorithm are turnaround time and waiting time. The main aim of the algorithm is to reduce the total turnaround and waiting time and average turnaround and waiting time, while preventing the processes from starvation.

For non-preemptive case, let us assume that there are five processes P1, P2, P3, P4 and P5 with burst times as 3s, 5s, 2s, 1s and 4s respectively. As this is the non-preemptive case, therefore all the processes are arriving at the same time. Using a function sort(), the burst times are sorted in ascending order and stored in an array. Now a queue is made which contains the process id according to the sorted array. The order of the processes will be P4, P3, P1, P5, P2. Let us assume the time quantum as 1s. A variable ‘time’ is initialized to 0 which keeps a track of the time. Another variable count is initialized to 0. A for loop is run with the condition that count should not be equal to 6. After the first iteration, the burst time of P4, P3, P1, P5 and P2 will be 0, 1, 2, 3 and 4 respectively. The process P4 has completed its execution. So, the turnaround time of P4 will be calculated by adding the time quantum(1s) to the current value of ‘time’ and storing it in an array ‘turn’. In the same way, all other processes will get executed in the order P4, P3, P1, P5, P2 as the iterations go on. Thus, at the end of the loop the turnaround time for P4, P3, P1, P5, P2 will be 1s, 6s, 10s, 13s, 15s respectively. After coming out of the loop, waiting time of each process is calculated by subtracting the burst time from the turnaround time and is stored in an array ‘wait’. Thus, the waiting time for P4, P3, P1, P5, P2 will be 0s, 4s, 7s, 9s, 10s respectively. Total turnaround time and total waiting time is calculated which comes out to be 45s and 30s respectively. Average turnaround time and waiting time is calculated by diving the total turnaround time and waiting time by 5, which comes out to be 9s and 6s respectively.

For preemptive case, let us assume that there are five processes P1, P2, P3, P4 and P5 with arrival times as 2s, 3s, 0s, 1s and 4s respectively and burst times as 4s, 6s, 2s, 5s and 3s respectively. Using the sort() function, the arrival time are stored in ascending order and stored a array. Now a queue is made which contains the process id of the processes according to the sorted arrival time array. Also, a new array is created which contains the burst time of the processes according to the sorted arrival time array. The order of the processes will be P3, P4, P1, P2, P5. Let us assume the time quantum as

1s. A variable ‘time’ is initialized to 0 which keeps a track of the time. Another variable count is initialized to 0. A for loop is run with the condition that count should not be equal to 6. Here, whenever a process arrives whose burst time is less than the burst time of previous processes, then the order of the running of the processes is altered and the process that has the minimum burst time will be executed first and so on. Thus, the order of the running of the process will keep on changing as the burst time gets reduced for each and every process and as the burst time of any process becomes 0, then the turnaround time of that process will be calculated. The order of the completion of the processes will be P3, P5, P1, P4, P2. The turnaround time of all the processes after all the processes will be executed will be P3=5s, P5=9s, P1=13s, P4=16s, P2=17s. Waiting time of all the process will be calculated and the values for P3, P5, P1, P4, P2 will be 3s, 6s, 9s, 11s, 11s respectively. Total turnaround time and total waiting time will be calculated and it comes out to be 59s and 39s respectively. Average turnaround time and average waiting time is calculated and it comes out to be 12.0s and 8.0 s respectively.

Result

SAMPLE CODE

Non-Preemptive Scheduling

```
void non_preemptive(int x)
{
    int i,y,z,burst[100],burstl[100],queue[100],j,wait[100],turn[100],time=0,count=1;
    for(i=1;i<=x;i++)
    {
        y=process[i].p_id;
        printf("\nPROCESS %d",y);
        printf("\nEnter the burst time of process %d:",y);
        scanf("%d",&process[i].bt);
    }
    printf("\nEnter the time quantum:");
    scanf("%d",&quantum);
    printf("\n\nPROCESS\t\tBURST TIME");
    for(i=1;i<=x;i++)
    {
        printf("\n%d\t\t%d",process[i].p_id,process[i].bt);
    }
    for(i=1;i<=x;i++)
        burst[i]=process[i].bt;
    sort(burst,x);
    for(i=1;i<=x;i++)
        burstl[i]=burst[i];
    for(i=1;i<=x;i++)
    {
        for(j=1;j<=x;j++)
        {
            if(burst[i]==process[j].bt)
            {
                queue[i]=process[j].p_id;
            }
        }
    }
    i=1;
    for(time=0;count!=x+1;time++)
    {
        while(burst[i]==0)
        {
            i=(i+1)%(x+1);
        }
        if(burst[i]>=quantum)
        {
            burst[i]=burst[i]-quantum;
            if(burst[i]==0)
            {
                turn[i]=time+quantum;
                count++;
```

Preemptive Scheduling

```
void preemptive(int x)
{
    int count,j,remain,time,time_quantum,i,z,k,a,y,n;
    int at[10],bt[10],rt[10],btl[10],queue[10],p_id[10],turn[10],wait[10],bt2[10],tat[10],wt[10];
    int ttat=0,twt=0;
    float atat,awt;
    remain=x;
    for(i=1;i<=x;i++)
    {
        y=process[i].p_id;
        printf("\nPROCESS %d",y);
        printf("\nEnter the arrival time of process %d:",y);
        scanf("%d",&process[i].at);
        printf("Enter the burst time of process %d:",y);
        scanf("%d",&process[i].bt);
        rt[i]=process[i].at;
    }
    sort(rt,x);
    for(i=1;i<=x;i++)
    {
        for(j=1;j<=x;j++)
        {
            if(rt[i]==process[j].at)
            {
                queue[i]=process[j].p_id;
                btl[i]=process[j].bt;
                bt2[i]=process[j].bt;
            }
        }
    }
    printf("\nEnter the time quantum:");
    scanf("%d",&quantum);
    printf("\n\nPROCESS\t\tBURST TIME\tARRIVAL TIME");
    for(i=1;i<=x;i++)
    {
        printf("\n%d\t\t%d\t\t%d",process[i].p_id,process[i].bt,process[i].at);
    }
    for(time=0,count=1,i=1;count!=x+1;time++)
    {
        n=0;
        while(rt[i]>time||btl[i]==0)
        {
            i=(i+1)%(x+1);
            n=n+1;
            if(n==x)
                break;
```

SAMPLE OUTPUT

Non Preemptive Scheduling

```
Enter the total number of processes:5

1.Non Preemptive Scheduling
2.Preemptive Scheduling
Which type of scheduling do you want to continue with:1

PROCESS 1
Enter the burst time of process 1:3

PROCESS 2
Enter the burst time of process 2:5

PROCESS 3
Enter the burst time of process 3:2

PROCESS 4
Enter the burst time of process 4:1

PROCESS 5
Enter the burst time of process 5:4

Enter the time quantum:1


PROCESS         BURST TIME
1               3
2               5
3               2
4               1
5               4

PROCESS BURST TIME      TURNAROUND TIME WAITING TIME
1       3               10              7
2       5               15              10
3       2               6               4
4       1               1               0
5       4               13              9

Average Turnaround Time:9.0
Average Waiting Time:6.0
Process returned 25 (0x19)   execution time : 21.565 s
Press any key to continue.
```

Preemptive Scheduling

```
Enter the total number of processes:5

1.Non Preemptive Scheduling
2.Preemptive Scheduling
Which type of scheduling do you want to continue with:2

PROCESS 1
Enter the arrival time of process 1:2
Enter the burst time of process 1:4

PROCESS 2
Enter the arrival time of process 2:3
Enter the burst time of process 2:6

PROCESS 3
Enter the arrival time of process 3:0
Enter the burst time of process 3:2

PROCESS 4
Enter the arrival time of process 4:1
Enter the burst time of process 4:5

PROCESS 5
Enter the arrival time of process 5:4
Enter the burst time of process 5:3

Enter the time quantum:1


PROCESS         BURST TIME      ARRIVAL TIME
1               4               2
2               6               3
3               2               0
4               5               1
5               3               4

PROCESS ARRIVAL TIME    BURST TIME      TURNAROUND TIME WAITING TIME
1       2               4               13              9
2       3               6               17              11
3       0               2               5               3
4       1               5               16              11
5       4               3               9               6

Average Turnaround Time:12.0
Average Waiting Time:8.0
Process returned 25 (0x19)   execution time : 21.514 s
Press any key to continue.
```

Conclusion

One of the important tasks of the operating system is the allocation of CPU to the processes waiting for execution. There are many scheduling algorithms with some advantages and disadvantages. A new algorithm was developed based on the concepts of Round Robin and Shortest Job First(SJF) CPU scheduling algorithms.

The results of the sample input mentioned in the modular description showed that the improved round robin algorithm for the non-preemptive situation gave the average turnaround time as 9.0s and average waiting time as 6.0s, whereas the conventional round robin algorithm gave average turnaround time as 10.2s and the average waiting time as 7.2s. These values increase as there is an increase in the values of the burst time.

When considering for the preemptive situation with the sample inputs mentioned in the modular description, the improved round robin algorithm gave the average turnaround time as 12.0s and the average waiting time as 8.0s, whereas the conventional round robin algorithm gave the average turnaround time as 12.4s and the average waiting time as 8.4s. These values increase as there is an increase in the values of the arrival time and the burst time.

Thus, from the above results we can conclude that the improved round robin algorithm is useful in decreasing the values of the average turnaround time and the average waiting time compared to that of the conventional round robin algorithm, while also making it sure that no process goes through the starvation situation. Due to the decreased values, the CPU will be able to complete the process in a less time than it would normally take. This will also increase the CPU utilization as more number of processes will be able to execute in the given time as compared to the conventional round robin algorithm. Each and every process is getting equal priority in this algorithm unlike other scheduling algorithms like SJF or priority scheduling.

References

1. Soraj, H and Roy, K.C: Adaptive Round Robin scheduling using shortest burst approach, based on smart time slice", International Journal of Data Engineering (IJDE), Volume 2, Issue 3, www.cscjournals.org/csc/manuscript/Journals/IJDE/.../IJDE-57.pdf ,accessed 01th December 2012.

2. AN IMPROVED ROUND ROBIN CPU SCHEDULING ALGORITHM WITH VARYING TIME QUANTUM Manish Kumar Mishra and Dr. Faizur Rashid

3. Manish Kumar Mishra & Abdul Kadir Khan, (2012) "An Improved Round Robin CPU Scheduling Algorithm", Journal of Global Research in Computer Science, Vol. 3, No. 6, pp 64-69.

4. Rami J. Matarneh, (2009) "Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of Now Running Processes", American J. of Applied Sciences, Vol. 6, No. 10, pp1831- 1837.

5. Operating Systems_ CPU Scheduling, http://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/5_CPU_Scheduling.html ,accessed 8th October 2013.

6. Lalit Kishor, Dinesh Goyal, "Time Quantum Based Improved Scheduling Algorithm", Issue International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, 2013.

7. M.K. Srivastav, Sanjay Pandey, Indresh Gahoi & Neelesh Kumar Namdev, (2012) "Fair Priority Round Robin with Dynamic Time Quantum", International Journal of Modern Engineering Research, Vol. 2, Issue 3, pp 876-881

8. Debashree Nayak, Sanjeev Kumar Malla & Debashree Debadarshini, (2012) "Improved Round Robin Scheduling using Dynamic Time Quantum", International Journal of Computer Applications, Vol. 38, No. 5, pp 34-38.

9. Abdulrazak Abdulrahim, Saleh E. Abdullahi & Junaidu B. Sahalu, (2014) "A New Improved Round Robin (NIRR) CPU Scheduling Algorithm", International Journal of Computer Applications, Vol. 90, No. 4, pp 27-33.

10. Rakesh Kumar Yadav, Abhishek K Mishra, Navin Prakash & Himanshu Sharma, (2010) "An Improved Round Robin Schedduling Algorithm for CPU Scheduling", International Journal on Computer Science and Engineering, Vol. 2, No. 4, pp 1064-1066s.