**1.1.1.1.1.1.1**

SCHEDULING ALGORITHM

**REPORT**

Submitted by

**Amreeta Koner 17BCE2033**

Prepared For

# OPERATING SYSTEMS (CSE2005) – PROJECT COMPONENT

**School of Computer Science and Engineering**



**ABSTRACT:**

This project deals with the simulation of CPU scheduling

algorithms with C. The following algorithms are simulated:

First Come First Serve(FCFS)

Shortest Job First(SJF)

SRTF Algorithm

Round Robin

Updated RR algorithm

The metrics such as finishing time, waiting time, total timetaken for the processes to complete, number of rounds, etc are calculated.

Multiprogramming is an important aspect of operating systems (OS); it requires several processes to be kept simultaneously in the memory, the aim of which is maximum CPU utilization.

Among other CPU scheduling algorithms, like the First Come First Serve (FCFS), Shortest Job First (SJF) and Priority Scheduling (PS); Round Robin is considered the most widely used scheduling algorithm in time sharing and real time OS for allocating the CPU to the processes in the memory in order to achieve the aim mentioned above.

This paper proposed a more improvement in the Round Robin CPU scheduling algorithm by improving the algorithm by Manish and AbdulKadir.

By experimental analysis, this proposed algorithm performs better than the simple Round Robin and the Improved Round Robin CPU scheduling algorithms in terms of minimizing average waiting time, average turnaround time and number of context switches.

**KEYWORDS:**

* Operating system
* Multiprogramming,
* CPU utilization,
* CPU scheduling algorithm,
* Round Robin
* CPU scheduling
* RR scheduling algorithm
* Turnaround time
* Waiting time
* Response time
* Context switch rate
* Gantt chart

**INTRODUCTION:**

The main objective of this paper is to develop a new approach for round robin scheduling which help to improve the CPU efficiency in real time and time sharing operating system.

There are many algorithms available for CPU scheduling. But we cannot implemented in real time operating system because of high context switch rates, large waiting time, large response time, large turn around time and less throughput.

The proposed algorithm improves all the drawback of simple round robin architecture as the proposed architecture solves all the problem encountered in simple round robin architecture by decreasing the performance parameters to desirable extent and thereby increasing the system throughput.

**RELATED WORKS:**

* An Additional Improvement in Round Robin (AAIRR) CPU Scheduling Algorithm” International Journal of Advanced Research in Computer Science and Software Engineering.

* An optimized round robin scheduling algorithm for CPU scheduling. International Journal on Computer Science and Engineering.

**APPLIED ALGORITHMS:**

**First Come First Serve (FCFS)**

* First-Come-First-Served algorithm is the simplest scheduling algorithm is the simplest scheduling algorithm. Processes are dispatched according to their arrival time on the ready queue. Being a non preemptive discipline, once a process has a CPU, it runs to completion. The FCFS scheduling is fair in the formal sense or human sense of fairness but it is unfair in the sense that long jobs make short jobs wait and unimportant jobs make important jobs wait.
* FCFS is more predictable than most of other schemes since it offers time. FCFS scheme is not useful in scheduling interactive users because it cannot guarantee good response time. The code for FCFS scheduling is simple to write and understand. One of the major drawback of this scheme is that the average time is often quite long.
* The First-Come-First-Served algorithm is rarely used as a master scheme in modern operating systems but it is often embedded within other schemes.

**Shortest Job First**

* Shortest-Job-First (SJF) is a non-preemptive discipline in which waiting job (or process) with the smallest estimated run-time-to-completion is run next. In other words, when CPU is available, it is assigned to the process that has smallest next CPU burst.
* The SJF scheduling is especially appropriate for batch jobs for which the run times are known in advance. Since the SJF scheduling algorithm gives the minimum average time for a given set of processes, it is probably optimal.
* The SJF algorithm favors short jobs (or processors) at the expense of longer ones.
* The obvious problem with SJF scheme is that it requires precise knowledge of how long a job or process will run, and this information is not usually available.
* The best SJF algorithm can do is to rely on user estimates of run times.

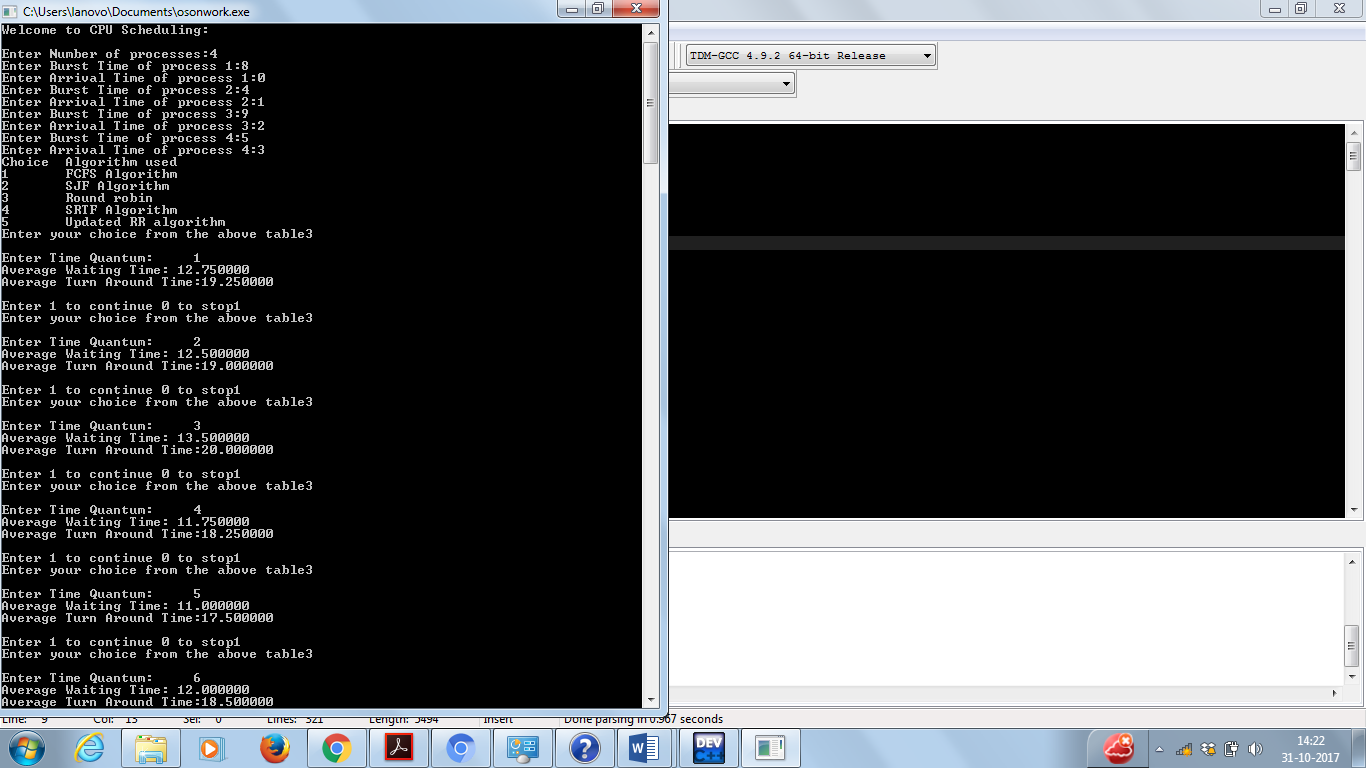
**SRTF Algorithm**

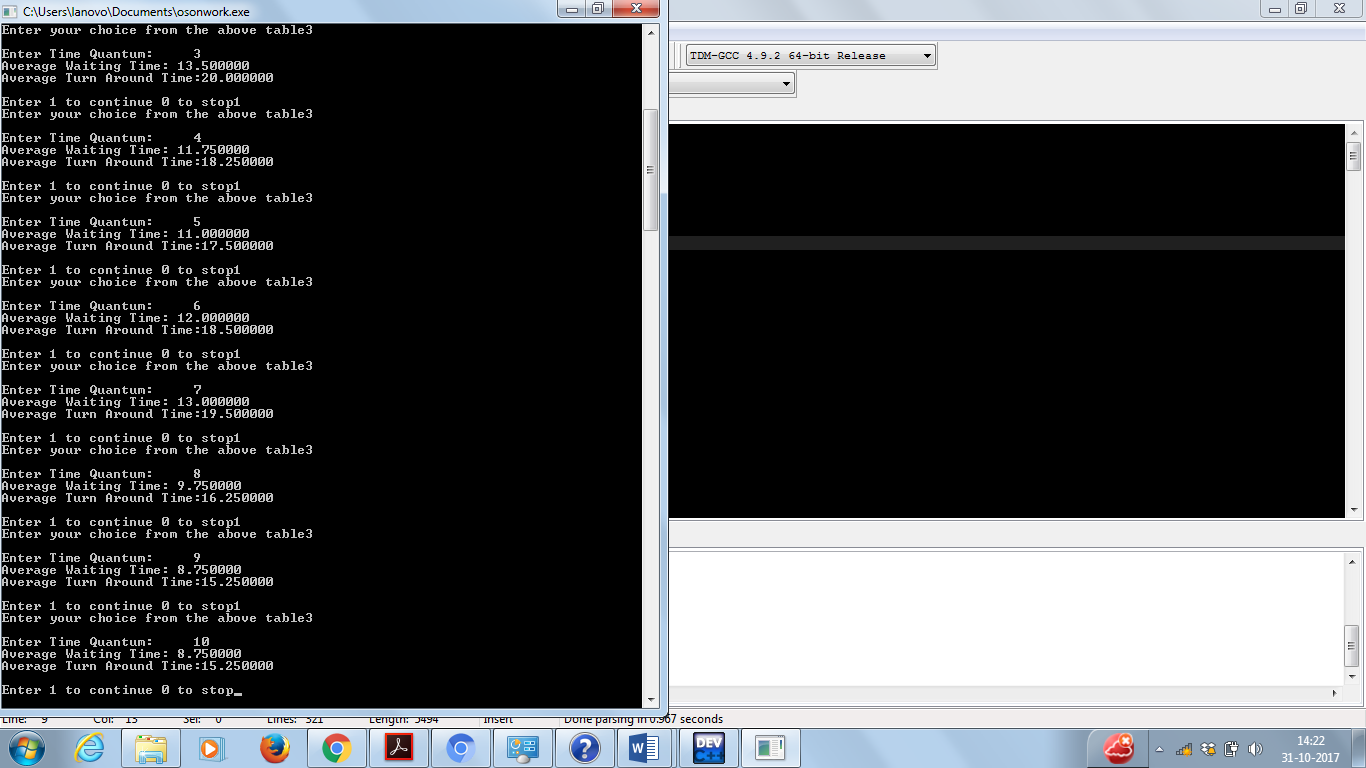
* The SRT is the preemtive counterpart of SJF and useful in time-sharing environment.
* In SRT scheduling, the process with the smallest estimated run-time to completion is run next, including new arrivals.
* In SJF scheme, once a job begin executing, it run to completion.
* In SJF scheme, a running process may be preempted by a new arrival process with shortest estimated run-time.
* The algorithm SRT has higher overhead than its counterpart SJF.
* The SRT must keep track of the elapsed time of the running process and must handle occasional preemptions.
* In this scheme, arrival of small processes will run almost immediately. However, longer jobs have even longer mean waiting time.

**Round Robin**

* One of the oldest, simplest, fairest and most widely used algorithm is round robin (RR).
* In the round robin scheduling, processes are dispatched in a FIFO manner but are given a limited amount of CPU time called a time-slice or a quantum.
* If a process does not complete before its CPU-time expires, the CPU is preempted and given to the next process waiting in a queue. The preempted process is then placed at the back of the ready list.
* Round Robin Scheduling is preemptive (at the end of time-slice) therefore it is effective in time- sharing environments in which the system needs to guarantee reasonable response times for interactive users.
* The only interesting issue with round robin scheme is the length of the quantum. Setting the quantum too short causes too many context switches and lower the CPU efficiency. On the other hand, setting the quantum too long may cause poor response time and appoximates FCFS.
* In any event, the average waiting time under round robin scheduling is often quite long.

**Performance Analysis**







TAT

AWT

TQ10

TQ9

TQ8

TQ7

TQ6

TQ5

TQ4

TQ3

TQ2

TQ1

0

5

10

15

20

25

**PROPOSED ALGORITHM:**

The proposed algorithm focuses on the improvement on CPU scheduling algorithm. The algorithm reduces the waiting time and turnaround time drastically compared to the other Scheduling algorithm and simple Round Robin scheduling algorithm. This proposed algorithm works in a similar way as but with some modification.It executes the shortest job having minimum burst time first instead of FCFS simple Round robin algorithm and it also uses Smart time quantum instead of static time quantum. Instead of giving static time quantum in the CPU scheduling algorithms, our algorithm calculates the Smart time quantum itself according to the burst time of all processes. The proposed algorithm eliminates the discrepancies of implementing simple round robin architecture.

* In the first stage of the innovative algorithm CPU scheduling algorithms all the processes are arranged in the increasing order of CPU burst time. It means it automatically assign the priority to the processes. Process having low burst time has high priority to the process have high burst time.
* Then in the second stage the algorithm calculates the mean of the CPU burst time of all the processes. After calculating the mean, it will set the time quantum dynamically i.e. (**average of mean and highest burst time)/2**.
* Then in the last stage algorithm pick the first process from the ready queue and allocate the CPU to the process for a time interval of up to 1 Smart time quantum. If the remaining burst time of the current running process is less than 1 Smart time quantum then algorithm again allocate the CPU to the Current process till it execution. After execution it will remove the terminated process from the ready queue and again go to the stage 3.

The flowchart for proposed algorithm is shown below in figure 1:





If remaining burst time of current process < 1 STQ

**YES**

**NO**

Arrange them in FCFS order

Allocate the CPU up to 1 STQ

Process the queue until it’s empty



Start

User Input the

Arrange the process in the ascending order of their burst

Compute the mean of all Burst time



Process having same burst time as STQ

**YES**

**NO**

Set the Smart time quantum, STQ=(Mean+Highest Burst time)/2

Pick the next process

Reallocate the CPU till it’s execution is complete

Remove the current process and put it at the tail for further execution

END

Calculate the average waiting time and average turn around time

**LITERATURE SURVEY:**

1. **Abstract-** Multiprogramming is an important aspect of operating systems (OS); it requires several processes to be kept simultaneously in the memory, the aim of which is maximum CPU utilization. Among other CPU scheduling algorithms, like the First Come First Serve (FCFS), Shortest Job First (SJF) and Priority Scheduling (PS); Round Robin is considered the most widely used scheduling algorithm in time sharing and real time OS for allocating the CPU to the processes in the memory in order to achieve the aim mentioned above. This paper proposed a more improvement in the Round Robin CPU scheduling algorithm by improving the algorithm by Manish and AbdulKadir. By experimental analysis, this proposed algorithm performs better than the simple Round Robin and the Improved Round Robin CPU scheduling algorithms in terms of minimizing average waiting time, average turnaround time and number of context switches.

**Keywords: operating system, multiprogramming, CPU utilization, CPU scheduling algorithm, Round Robin.**

1. **Abstract -** The main objective of this paper is to develop a new approach for round robin scheduling which help to improve the CPU efficiency in real time and time sharing operating system. There are many algorithms available for CPU scheduling. But we cannot implemented in real time operating system because of high context switch rates, large waiting time, large response time, large trn around time and less throughput. The proposed algorithm improves all the drawback of simple round robin architecture. The author have also given comparative analysis of proposed with simple round robin scheduling algorithm. Therefore, the author strongly feel that the proposed architecture solves all the problem encountered in simple round robin architecture by decreasing the performance parameters to desirable extent and thereby increasing the system throughput.

**Keywords: CPU scheduling,RR scheduling algorithm,Turnaround time,Waiting time,Response time,,Context switch rate, Gantt chart**

|  |  |  |
| --- | --- | --- |
| P2 | P1 | P3 |

|  |  |  |
| --- | --- | --- |
| P2 | P3 | P1 |

|  |  |  |  |
| --- | --- | --- | --- |
| P2 | P1 | P3 | P2 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P2 | P1 | P2 | P3 | P1 | P2 |

|  |  |  |
| --- | --- | --- |
| P2 | P3 | P1 |

**EXPERIMENTAL ANALYSIS:**

No. of processes : 3

|  |  |  |
| --- | --- | --- |
| Process | Burst time | Arrival time |
| 1 | 5 | 2 |
| 2 | 7 | 1 |
| 3 | 3 | 5 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| GANTT CHART:  FCFS : |  | | | | | |
|  |
| 1  Awt=(6+0+7)/3=4.34 Atat=(11+7+10)/3=9.34 | 8 |  | 13 |  |  | 16 |
| SJF : |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| 1  Awt=(12)/3=4  Atat=(27)/3=9 | 8 |  | 11 |  |  | 16 |
| SRTF : |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| 1 2  Awt=(10)/3=3.34 Atat=(25)/3=8.34 |  | 7 |  | 10 |  | 16 |
| RR : |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| 1 4  Awt=(17)/3=5.67 Atat=(32)/3=10.67 | 7 | 10 | 13 |  | 15 | 16 |
| URR: |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| 1  Awt=(17)/3=5.67 Atat=(32)/3=10.67 | 8 |  | 11 |  |  | 16 |

No. of processes : 4

|  |  |  |
| --- | --- | --- |
| Process | Burst time | Arrival time |
| 1 | 8 | 0 |
| 2 | 4 | 1 |
| 3 | 9 | 2 |
| 4 | 5 | 3 |

GANTT CHART:

FCFS :

|  |  |  |  |
| --- | --- | --- | --- |
| P1 | P2 | P3 | P4 |

0 8 12 21 26

Awt=(7+10+18)/4=8.75 Atat=(8+11+19+23)/4=15.25

SJF :

|  |  |  |  |
| --- | --- | --- | --- |
| P1 | P2 | P4 | P3 |

0 8 12 17 26

Awt=(7+15+9)/4=7.75 Atat=(8+11+24+14)/4=14.25

SRTF :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P1 | P2 | P4 | P1 | P3 |

0 1 5 10 17 26

Awt=(9+15+2)/4=6.5 Atat=(17+4+24+7)/4=13

RR :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| P1 | P2 | P3 | P4 | P1 | P3 | P4 | P3 |

0 4 8 12 16 20 24 25 26

Awt=(12+3+15+17)/4=10.75 Atat=(20+22+24+7)/4=17.25

URR :

|  |  |  |  |
| --- | --- | --- | --- |
| P1 | P2 | P3 | P4 |

0 8 12 21 26

Awt=(7+10+18)/4=9.25 Atat=(8+11+19+23)/4=15.75

**Graphical comparison of burst and arrival time**

8

7

6

5

4

3

2

1

0

P1

Arrival time

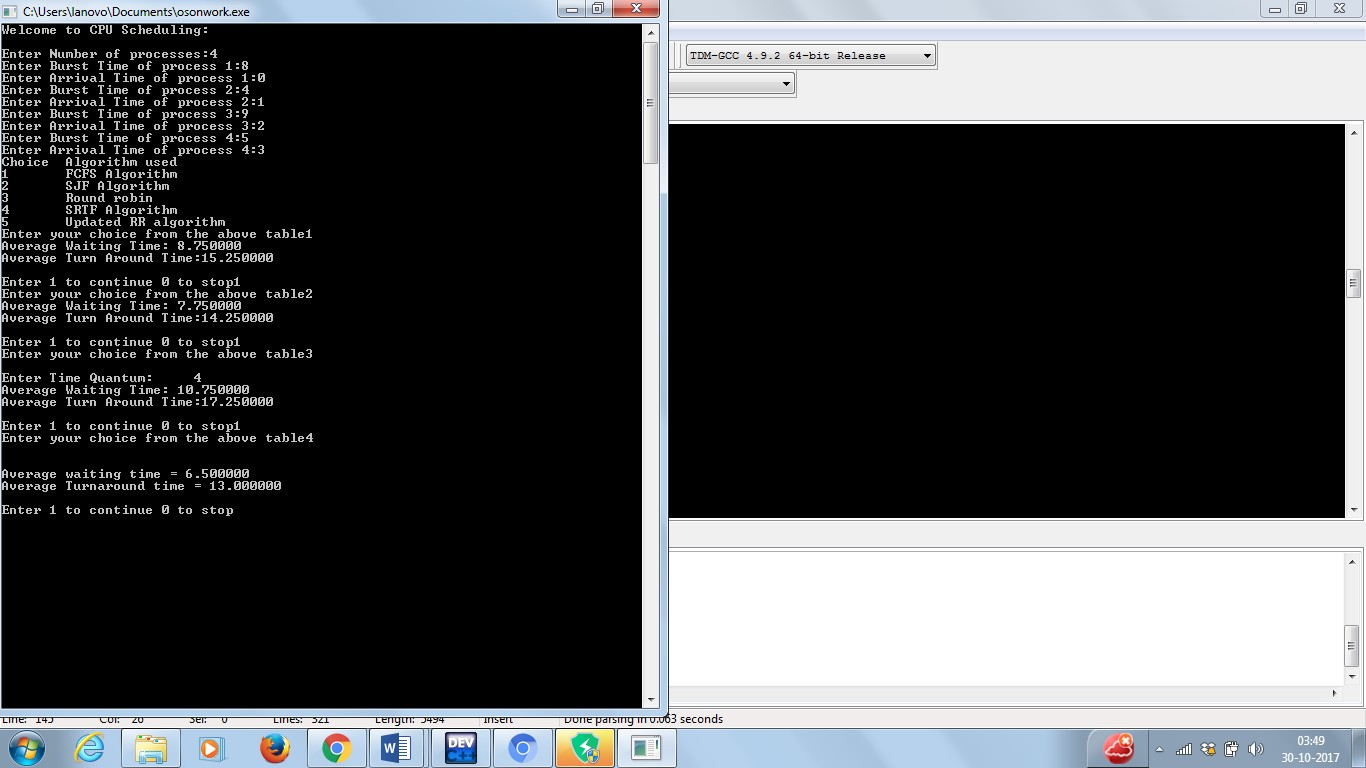
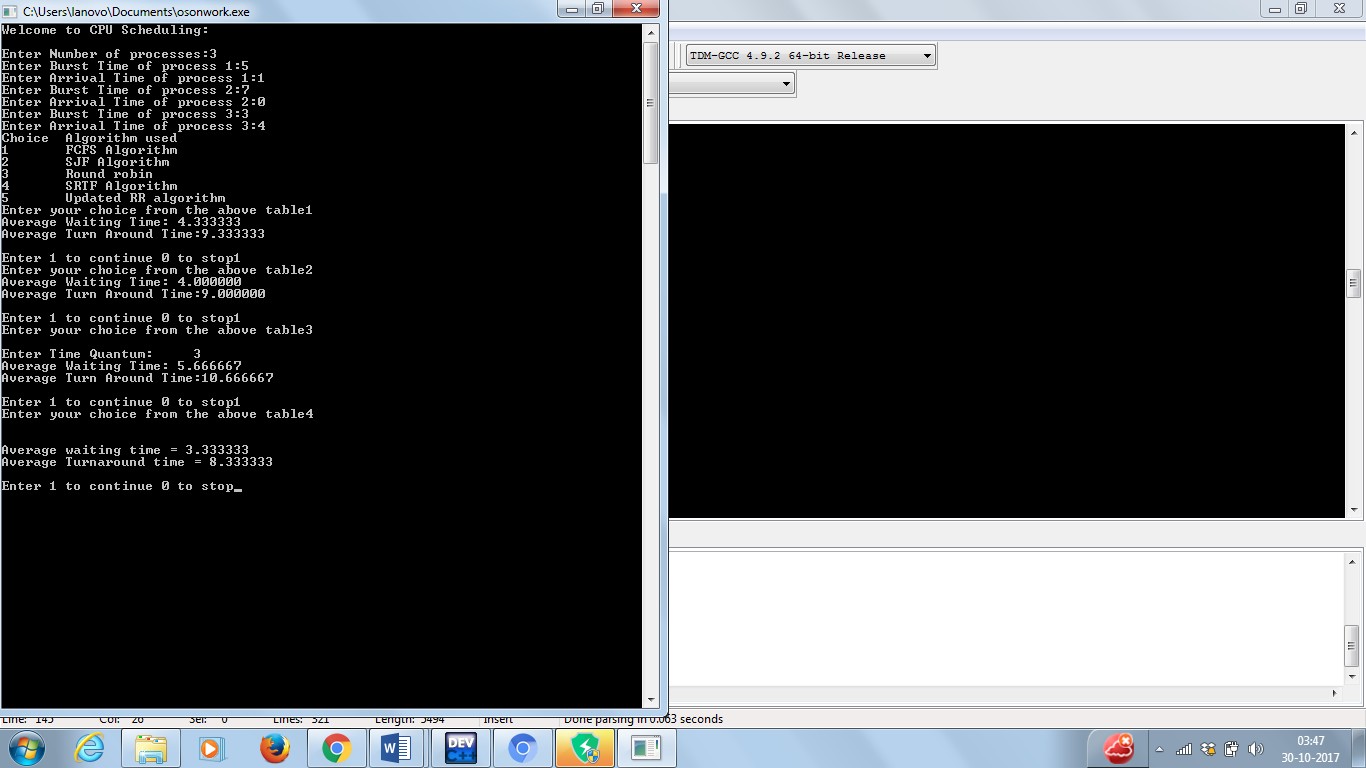
Burst time

P2

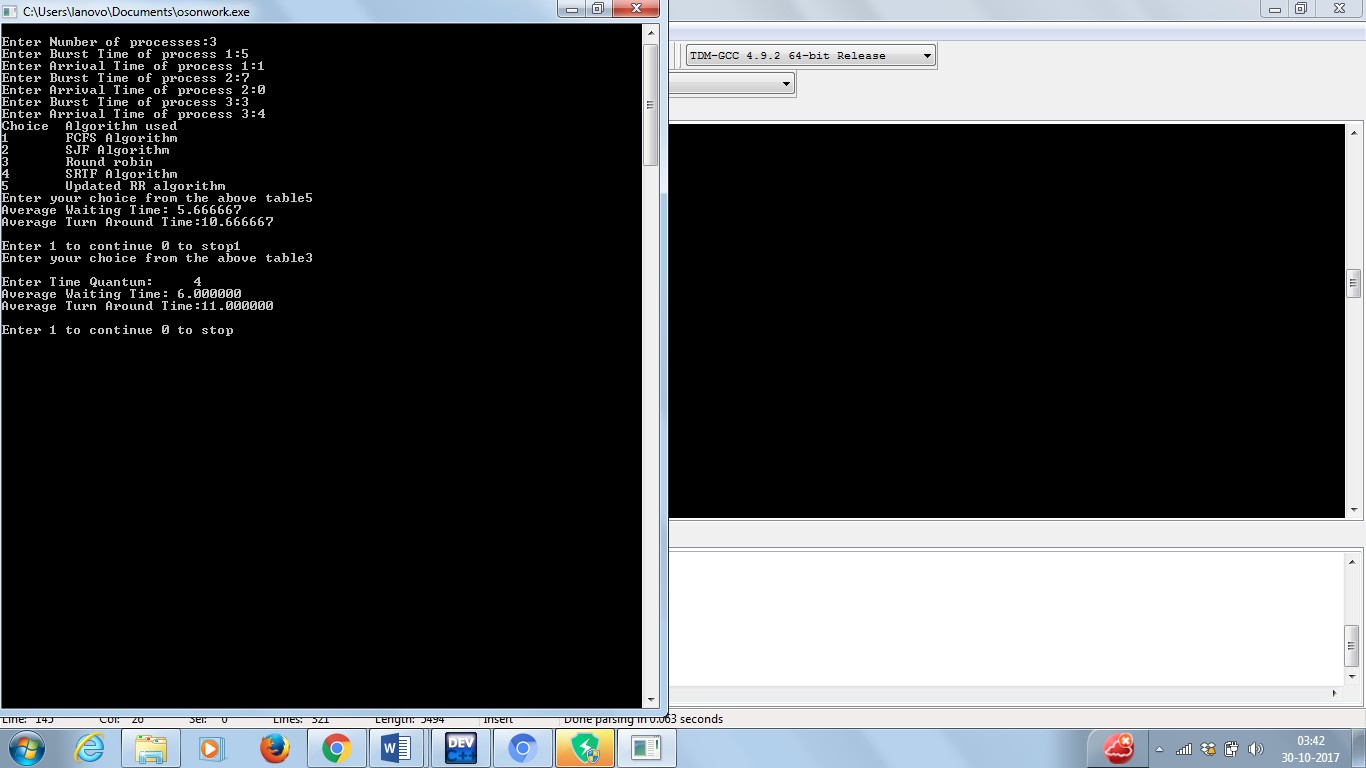
P3

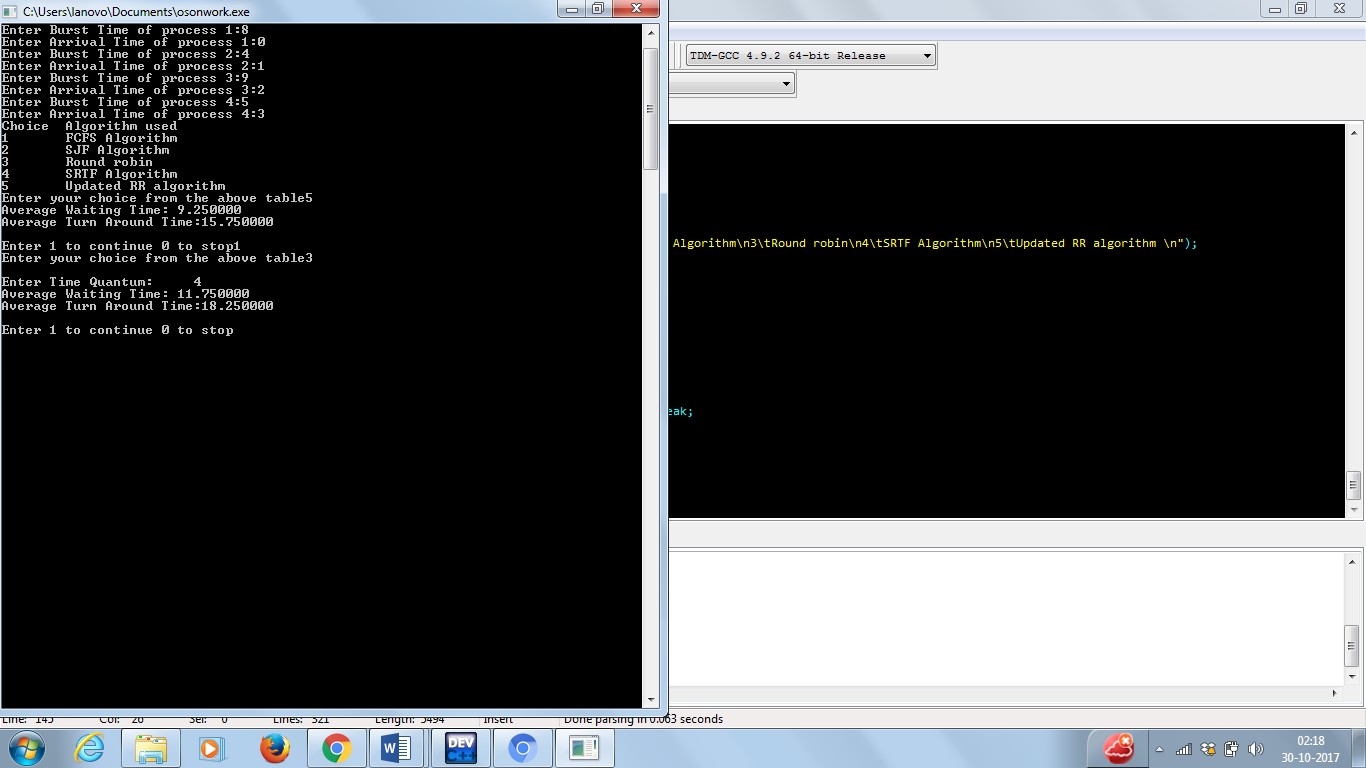
**SNAP CHATS:**

**THE FIRST FOUR ALGORITHMS:**



**UPDATED RR ALGORITHM**





**CONCLUSION AND FUTURE WORK:**

Results have shown that the proposed algorithm gives better results in terms of average waiting time, average turnaround time and number of context switches in all cases of process categories than the simple Round Robin CPU scheduling algorithm. In all these proposed algorithms time quantum is static due to which in these cases the number of context switches, average waiting time and average turnaround time will be very high and in our proposed algorithm, time quantum is calculated dynamically according to the burst time of all processes and it will find out a smart time quantum for all processes which gives good performance as compared to FCFS, RR, SJF and XX.

**REFERENCES:**

[1]. Abdulraza, Abdulrahim,Salisu Aliyu, Ahmad M Mustapha, Saleh E Abdullahi, “ An Additional Improvement in Round Robin (AAIRR) CPU Scheduling Algorithm,”International Journal of Advanced Research in Computer Science and Software EngineeringVolume 4, Issue 2, February 2014 ISSN: 2277 128X.

[2]. Singh, A., Goyal, P., & Batra, S. (2010). An optimized round robin scheduling algorithm for CPU scheduling. International Journal on Computer Science and Engineering, 2(07), 2383-2385.