CS F372: Operating Systems

An Assignment Report

On

Scheduling Algorithms and Threading in Linux

BY

OM AGARWAL (2019A7PS0052H)

GAURVIT KUMAR (2019A7PS1278H)

ADITYA P S TOMAR (2019A7PS0127H)

SARANSH DWIVEDI (2019A7PS0173H)

OJASHVI TARUNABH (2019A7PS0025H)

SAMARTH SONI (2019A7PS1274H)

BHARADWAZ RUSHI D (2019A7PS0111H)

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)

HYDERABAD CAMPUS

(NOV 2021)

INTRODUCTION:

Assignment Description-

The given assignment is based on doing multi-process and multi-threaded programming with inter-process and inter-thread communication using pipes and shared memory as well as performing process scheduling (considering uniprocessor environment).

Here we have created 3 child processes(C1, C2, C3) that are to be scheduled using the parent process (Master). Each of the child processes implements two threads for the purpose of process communication(Monitor Thread) and performing tasks(Task Thread). The master performs scheduling of these child processes using Round Robin or First come First Serve algorithms.

At last we do the time calculation and perform the workload analysis with respect to turn around time and waiting time of each of the child process C1, C2, C3.

Program (Assignment_code.c)

Input-

- We need to give the arguments (n1, n2, n3) to be used in the process.
- In the case of Round Robin scheduling, we also need to give the required time quantum (in nanoseconds).
- Two files (Readme1.txt, Readme2.txt) are required in the process I/O where file I/O is needed.

Output-

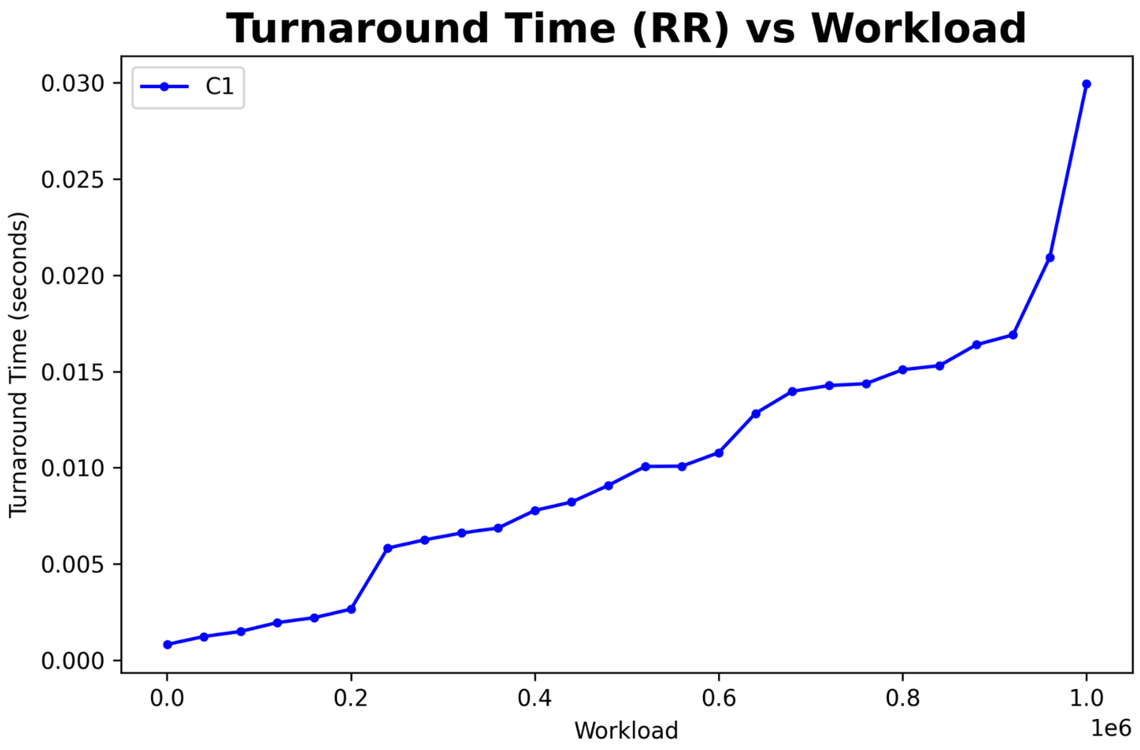
- The result of the work done by each of the processes is displayed on the terminal.
- The Execution time, Turn Around Time and Waiting Time is displayed for each of the processes C1, C2, C3.

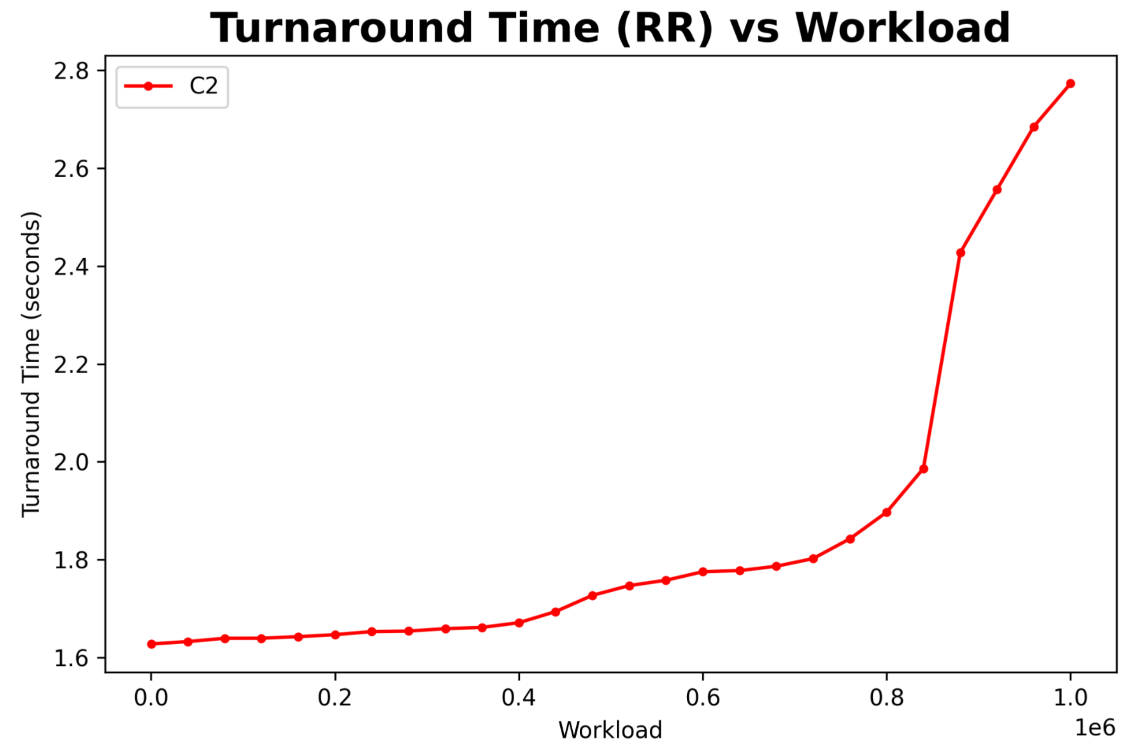
Concepts Utilized-

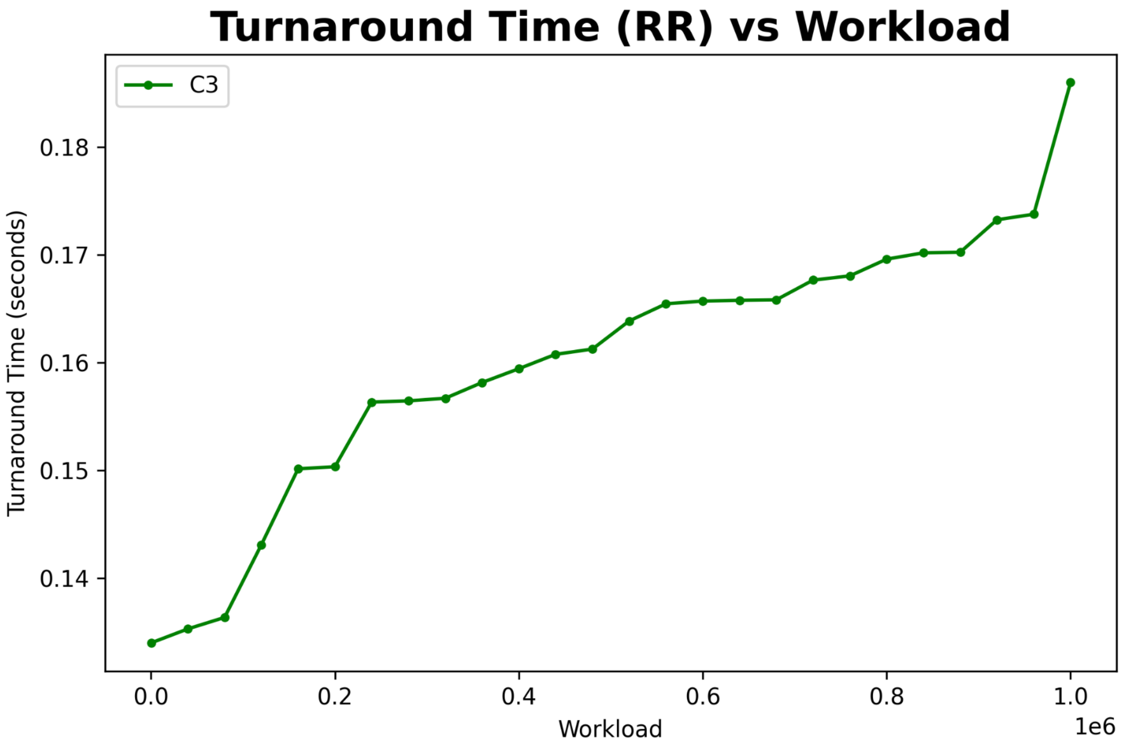
- Process Creation (using fork())

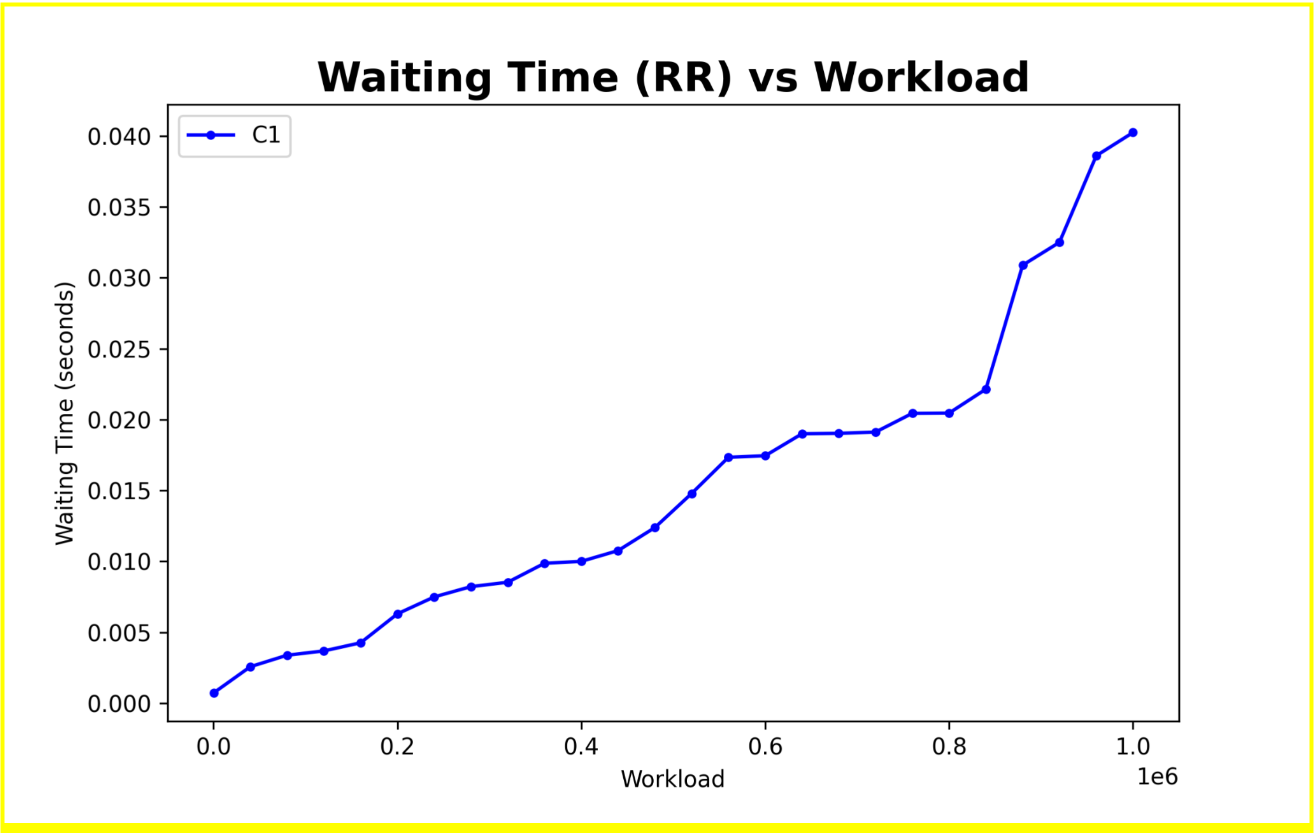
- Inter Process Communication (Pipes and shared memory)
- Thread creation (Pthreads)
- Thread Signalling (Mutex and conditionals)
- Process Scheduling (FCFS & Round Robin)
- Time Calculation (Turn Around Time, Wait Time, Execution Time)

PLOTS :

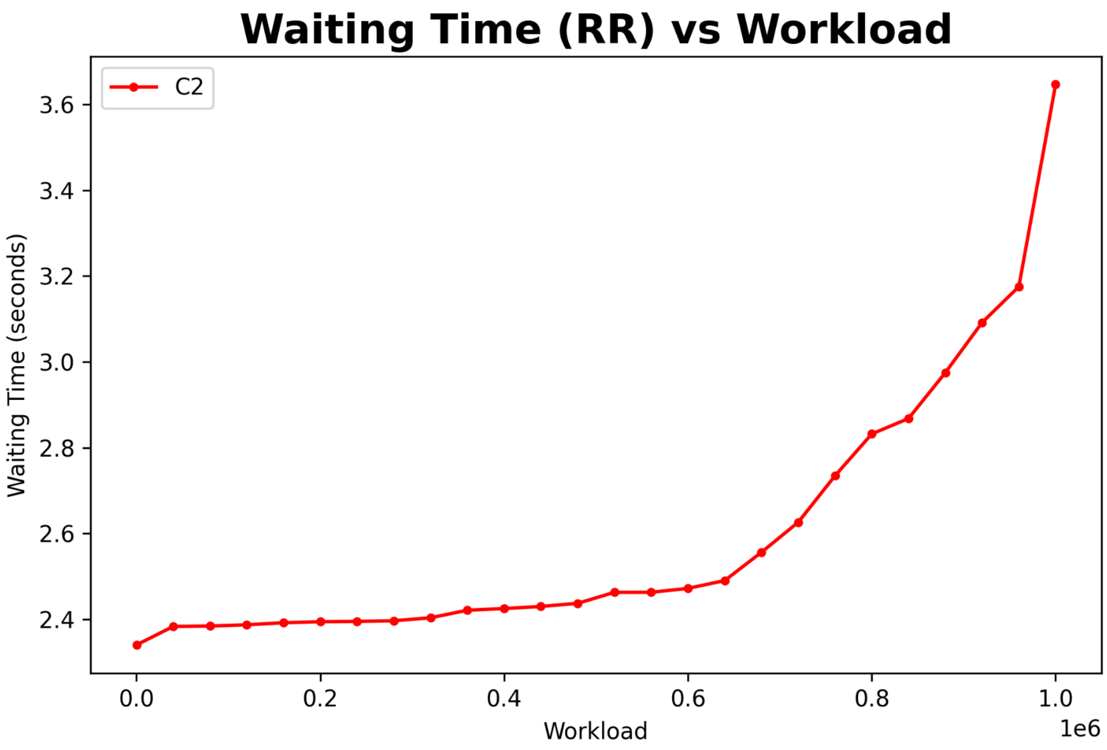

Turnaround Time (RR) vs Workload
C1
Turnaround Time (seconds)
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0.0
0.2
0.4
0.6
0.8
1.0
1e6
Workload


Turnaround Time (RR) vs Workload
C2
Turnaround Time (seconds)
2.8
2.6
2.4
2.2
2.0
1.8
1.6
0.0
0.2
0.4
0.6
0.8
1.0
1e6
Workload

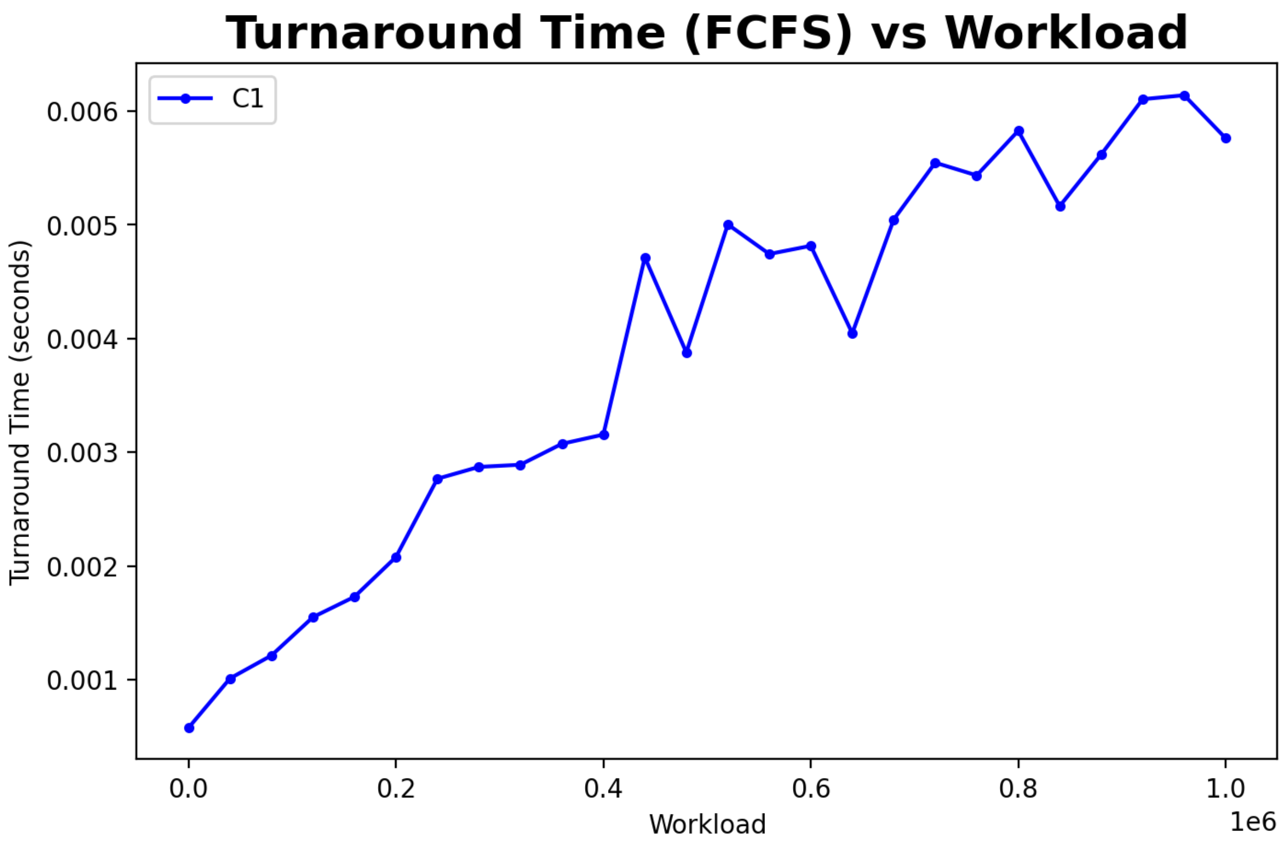

Turnaround Time (RR) vs Workload
C3
Turnaround Time (seconds)
0.18
0.17
0.16
0.15
0.14
0.0
0.2
0.4
0.6
0.8
1.0
1e6
Workload

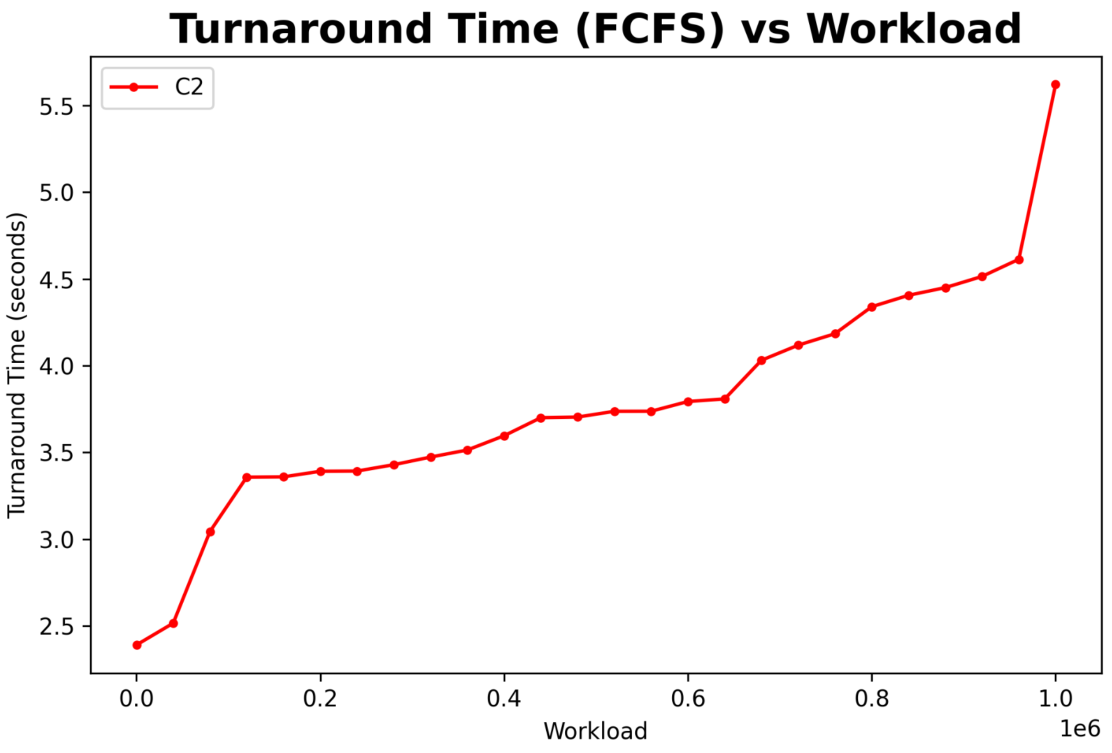


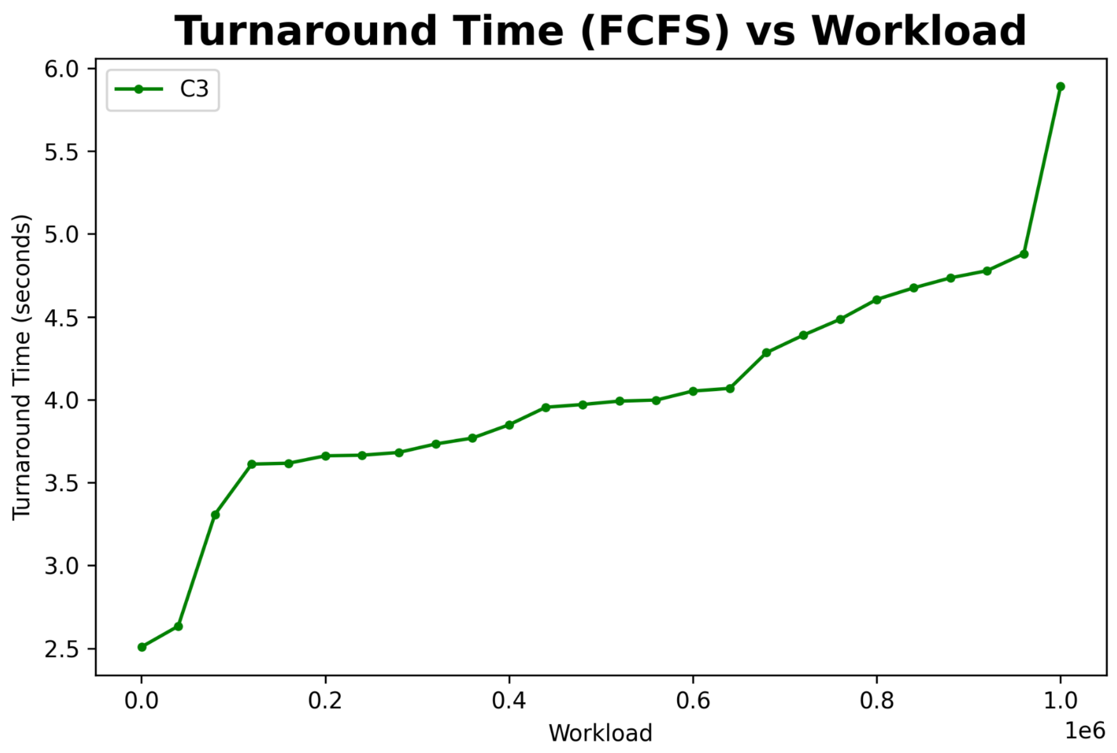
Waiting Time (RR) vs Workload
C1
Waiting Time (seconds)
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0.0
0.2
0.4
0.6
0.8
1.0
Workload
1e6

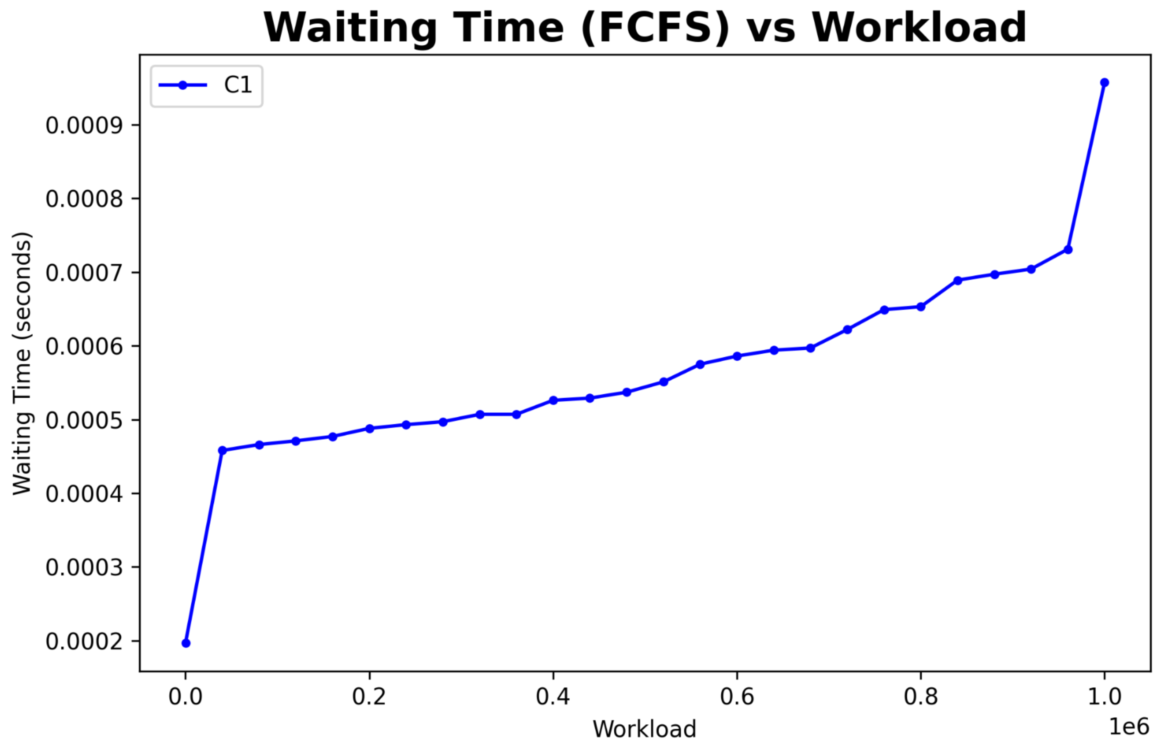


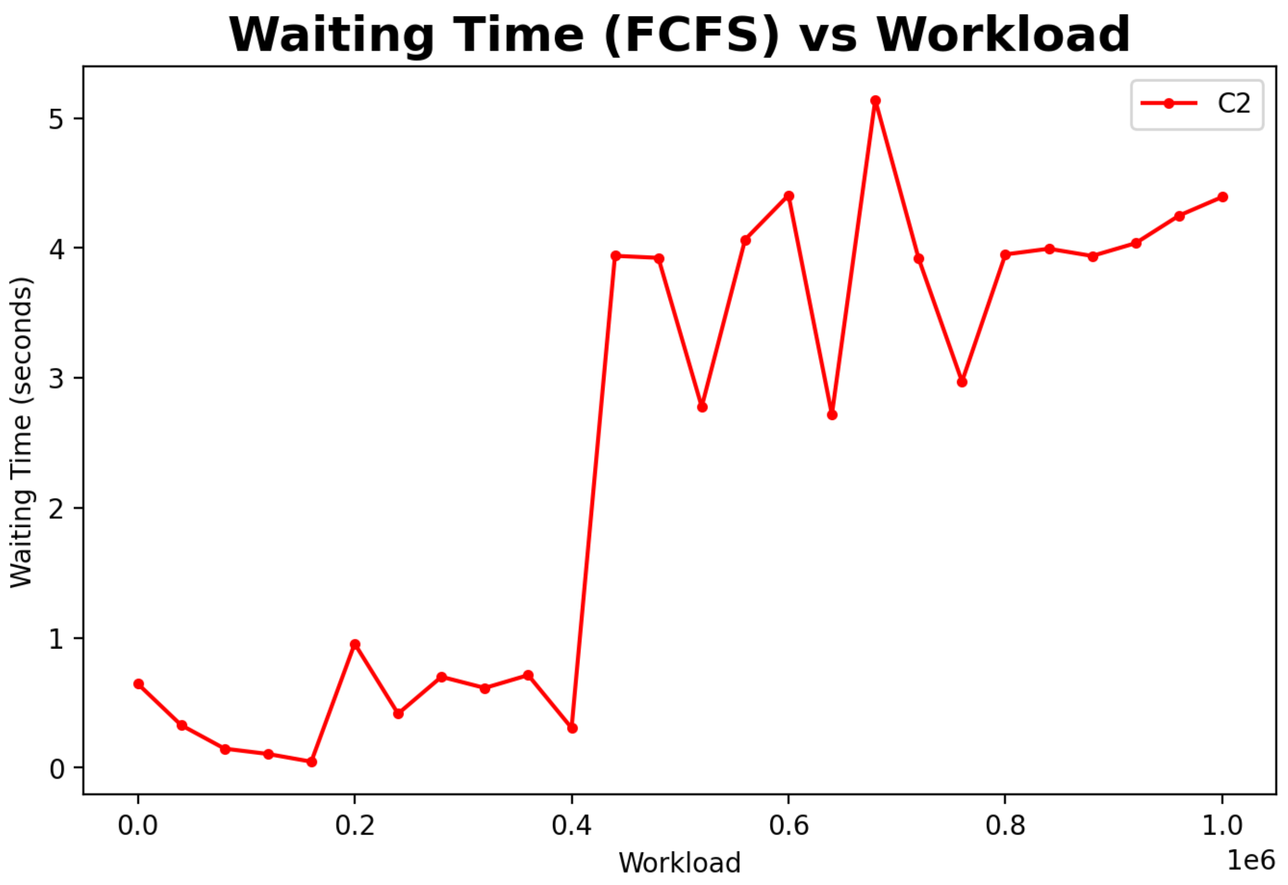
Waiting Time (RR) vs Workload
C2
Waiting Time (seconds)
3.6
3.4
3.2
3.0
2.8
2.6
2.4
0.0
0.2
0.4
0.6
0.8
1.0
Workload
1e6

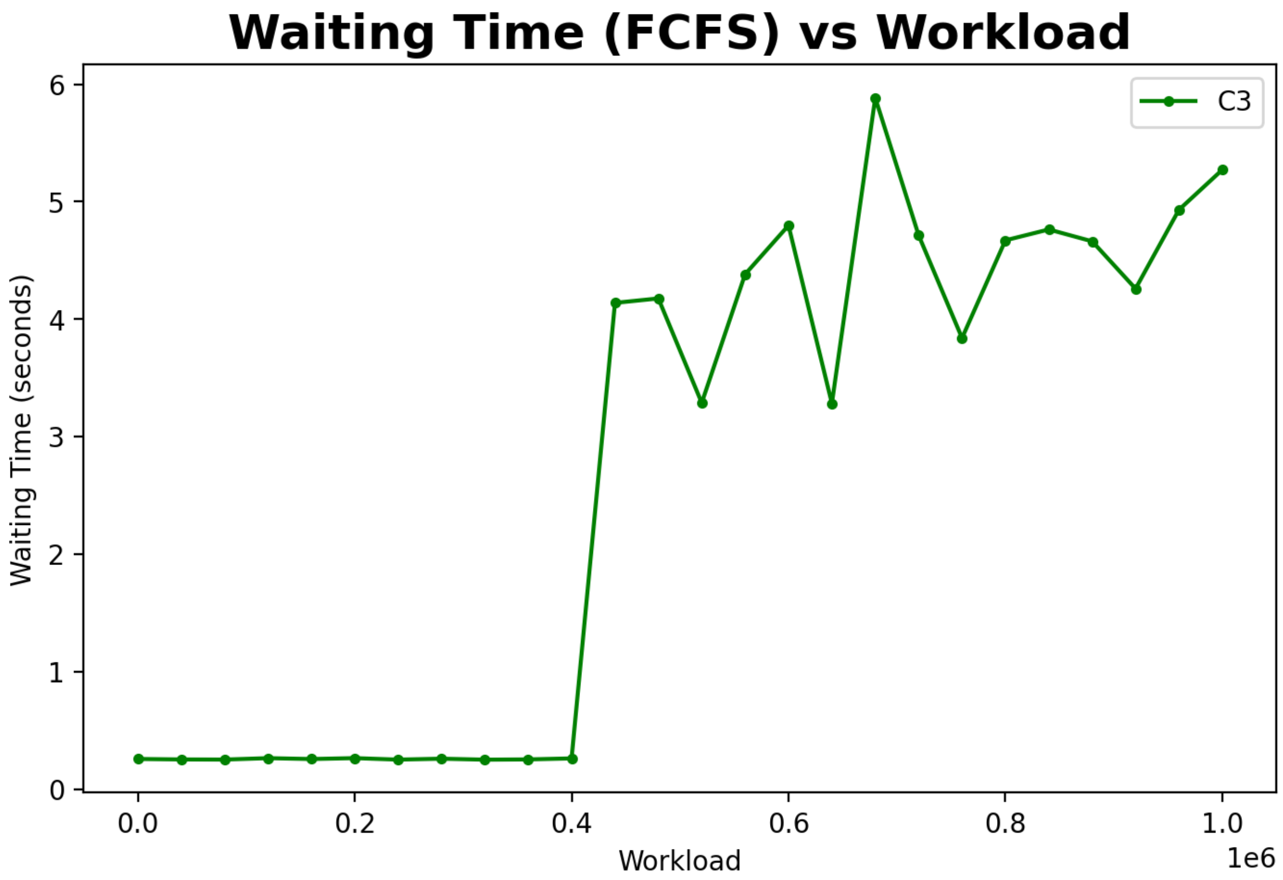

Turnaround Time (FCFS) vs Workload
C1
Turnaround Time (seconds)
0.001
0.002
0.003
0.004
0.005
0.006
0.0
0.2
0.4
0.6
0.8
1.0
1e6
Workload

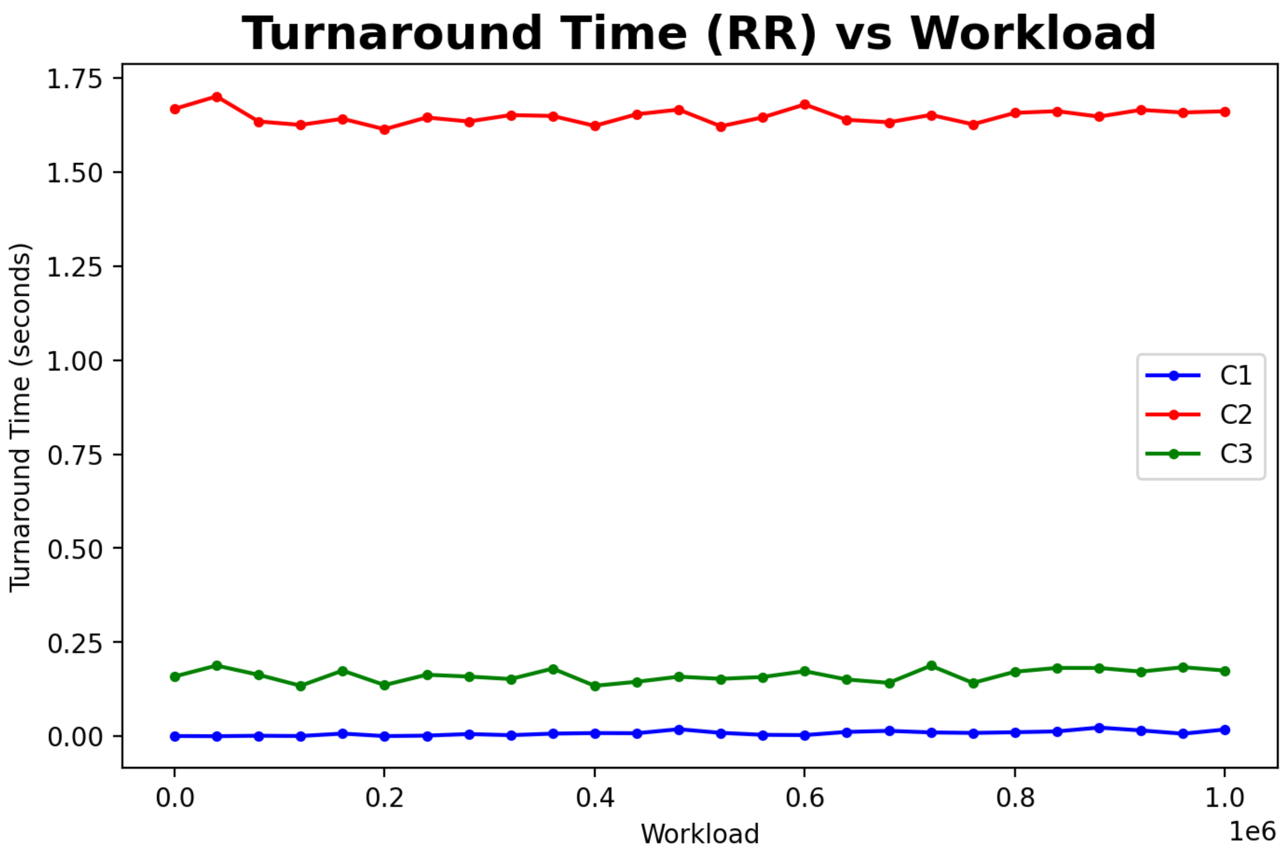

Turnaround Time (FCFS) vs Workload
C2
Turnaround Time (seconds)
2.5
3.0
3.5
4.0
4.5
5.0
5.5
0.0
0.2
0.4
0.6
0.8
1.0
1e6
Workload

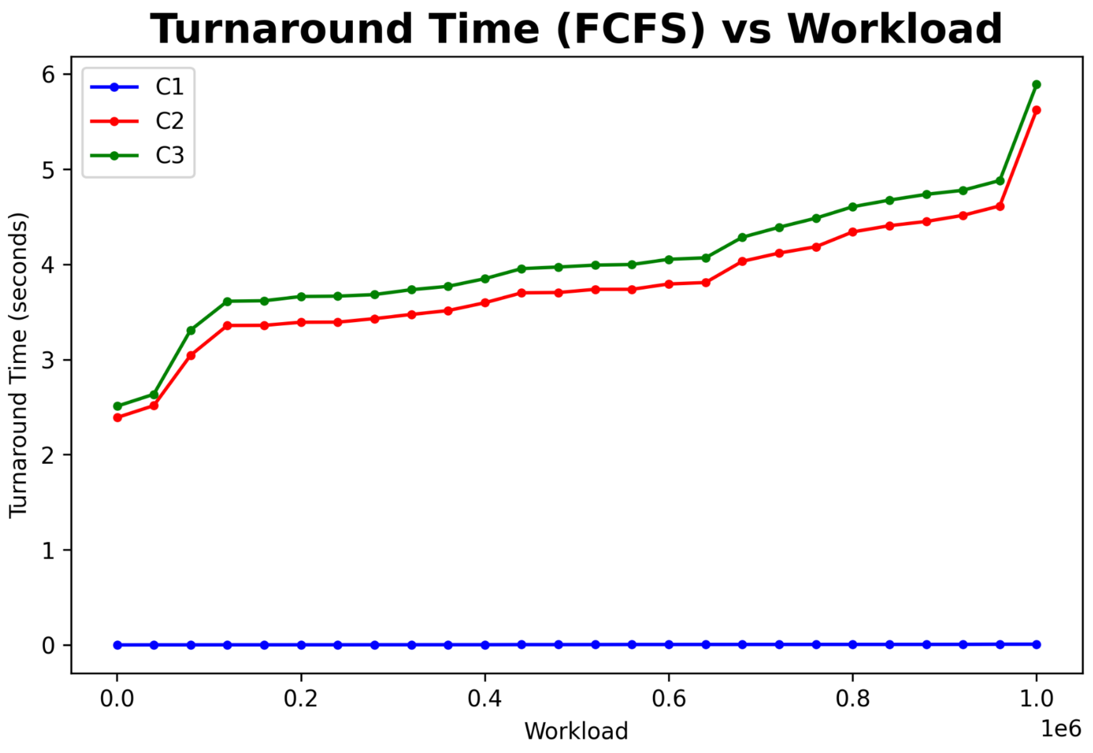

Turnaround Time (FCFS) vs Workload
C3
Turnaround Time (seconds)
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0
1e6
Workload


Waiting Time (FCFS) vs Workload
C1
Waiting Time (seconds)
0.0009
0.0008
0.0007
0.0006
0.0005
0.0004
0.0003
0.0002
0.0
0.2
0.4
0.6
0.8
1.0
1e6
Workload


Waiting Time (FCFS) vs Workload
C2
Waiting Time (seconds)
0
1
2
3
4
5
0.0
0.2
0.4
0.6
0.8
1.0
1e6
Workload


Waiting Time (FCFS) vs Workload
Waiting Time (seconds)
Workload
1e6
C3
0
1
2
3
4
5
6
0.0
0.2
0.4
0.6
0.8
1.0


Turnaround Time (RR) vs Workload
Turnaround Time (seconds)
Workload
1e6
C1
C2
C3
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
0.0
0.2
0.4
0.6
0.8
1.0


Turnaround Time (FCFS) vs Workload
Turnaround Time (seconds)
Workload
1e6
C1
C2
C3
0
1
2
3
4
5
6
0.0
0.2
0.4
0.6
0.8
1.0

CONCLUSION:

From the above graphs we can conclude that:

- There is a general trend of Turn Around Time and Wait Time increasing with the workload. This is evident for both cases of scheduling algorithms. This is because our program is of O(n) time complexity and we have assumed a uniprocessor environment.
- For comparing the graphs of all processes run in FCFS and all processes run with RR, we see that on an average, the waiting time of Round Robin scheduling is lesser than that of FCFS scheduling.
- As there is a good amount of context switching in Round Robin, it has more overhead than FCFS.
- The response time of FCFS (for example c1) for short processes is lesser than in the case of Round Robin.

THANK YOU