

CS342 Operating Systems

Project 2

Experiments Report

Ata Seren Osman Semih Tiryaki

21901575 21801994

Section-2 Section-1

Introduction

First, we implemented locks, condition variables, queue and queue functions that we used throughout the program. Then, we implemented process generation thread. After being sure that processes are generated and other mechanisms work successfully, we started to implement algorithms. In parallel, we implemented scheduler thread for the algorithms. After we implement the algorithms, we implemented the device usage and tested them using various print statements. Finally, when we tested all our algorithms, we implemented the statements for several types of outputs. In conclusion, we created a simulation of how processes, CPU and I/O devices work simultaneously.

Computer Specifications

We performed experiments on a laptop with Windows 10 Home, Intel i7-9750H CPU @ 2.60GHz with 4.2GHz Turbo and 16,0 GB of RAM.

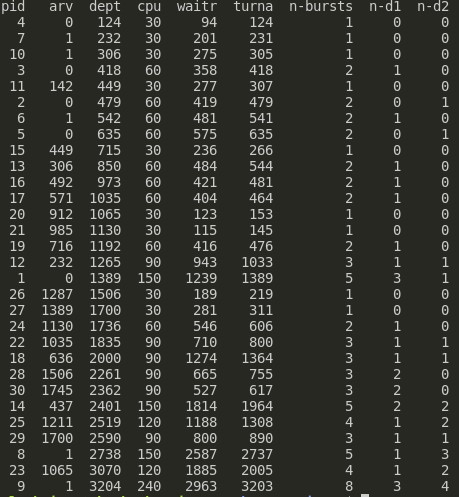
We used Oracle VirtualBox VM to run the code on virtual machine with Ubuntu 20.04. I reserved 5 CPU cores and 10725MB’s of RAM for the virtual machine. In the machine, we used Linux terminal to compile and run the code and VS Code to write the code.

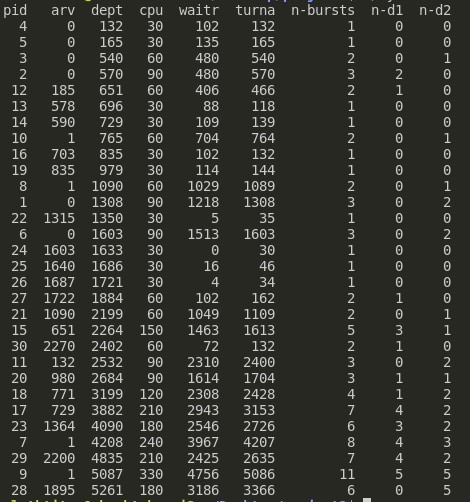
Experiments

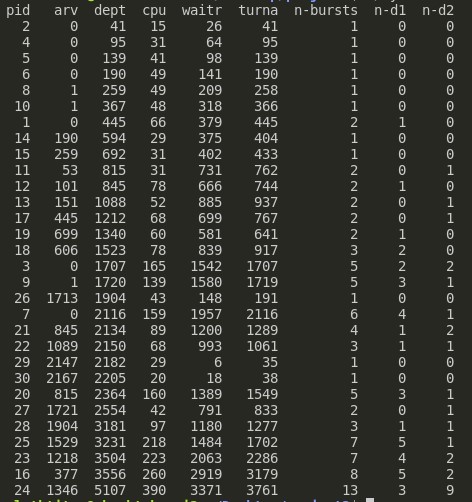
As experiments, we measured the average of waiting and turnaround times of algorithms with different types of burst times.

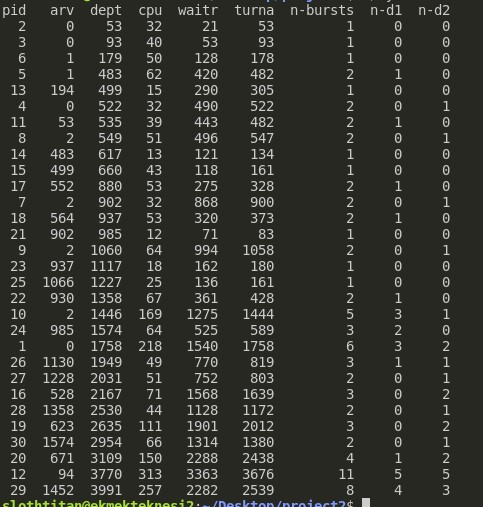
We used 20ms for time quantum 50ms and 150ms for waiting time of device 1 and 2, 30ms as burst length, 10ms and 50ms as min and max bursts. 0.4, 0.3 and 0.3 for probabilities of termination, device 1 and device 2, respectively, 0.5 probability for process generation. 10 as max process in the system, 30 as total process generated and 1 as the outmode.

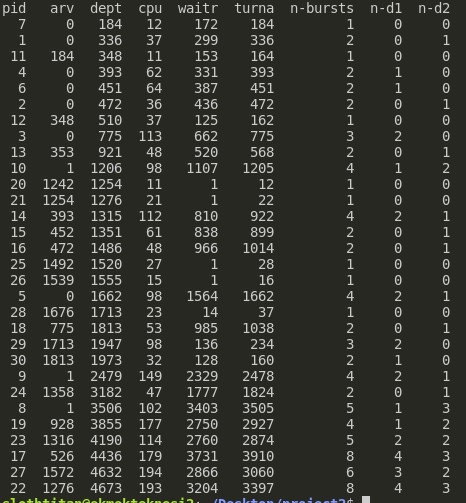
Following screenshots are the outputs of our program where we use the measure the data mentioned above:

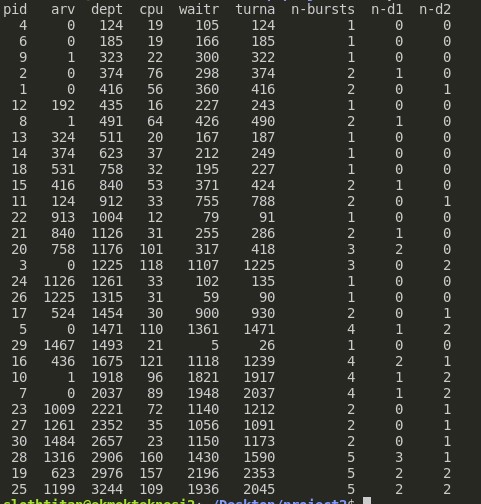
FCFS (fixed)



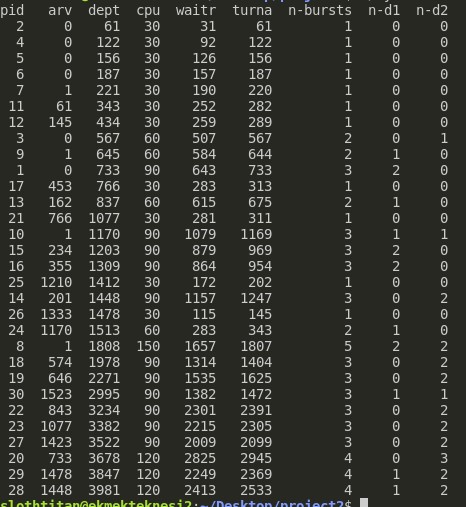
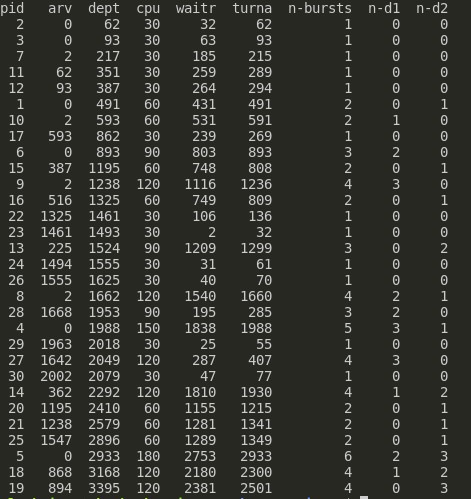
FCFS (uniform)

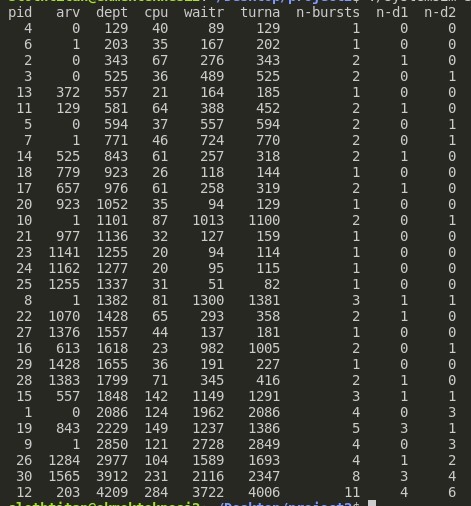


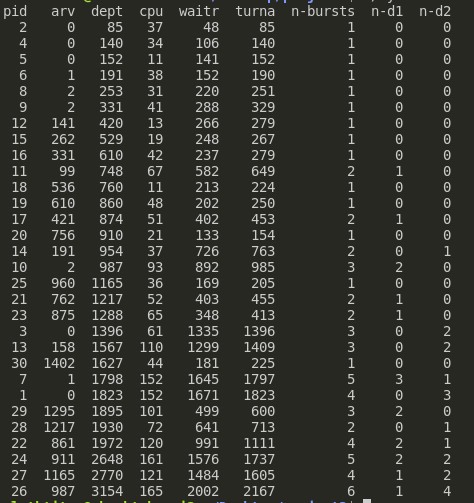
FCFS (exponential)



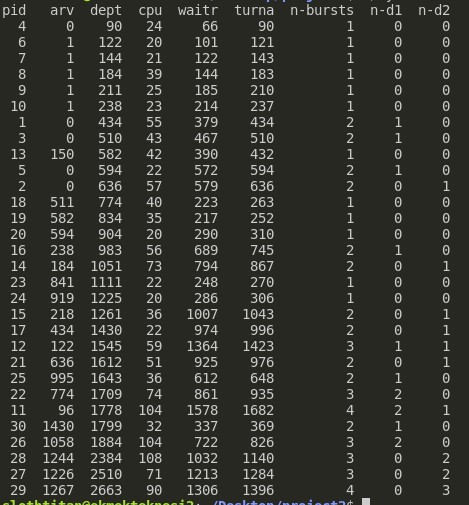
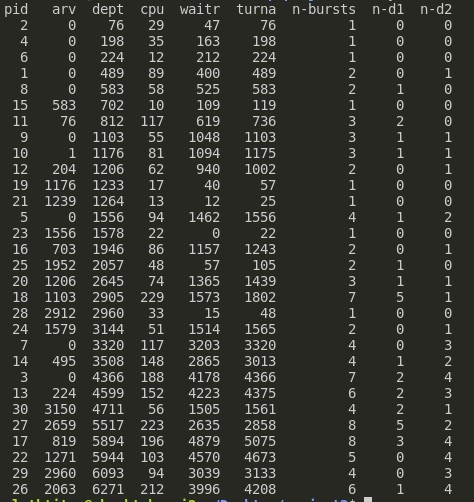
SJF (fixed)

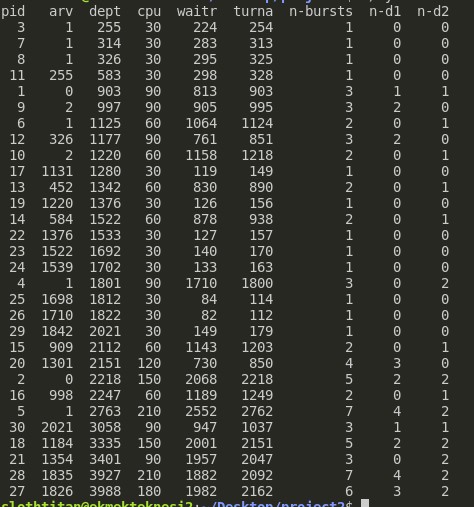
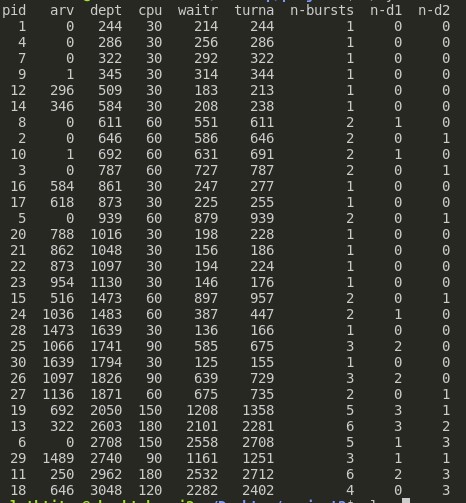


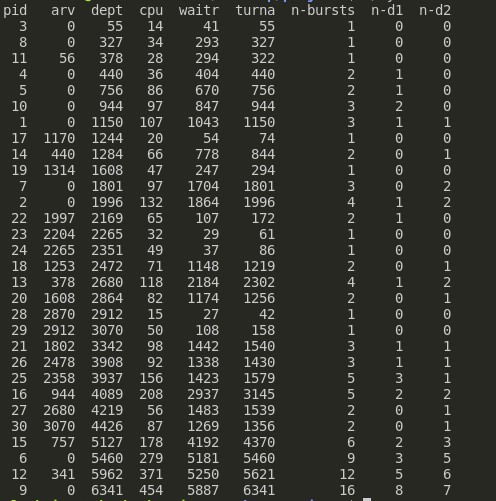
SJF (uniform)

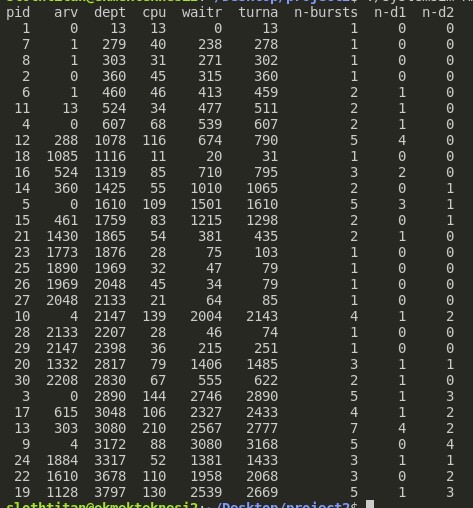


SJF (exponential)

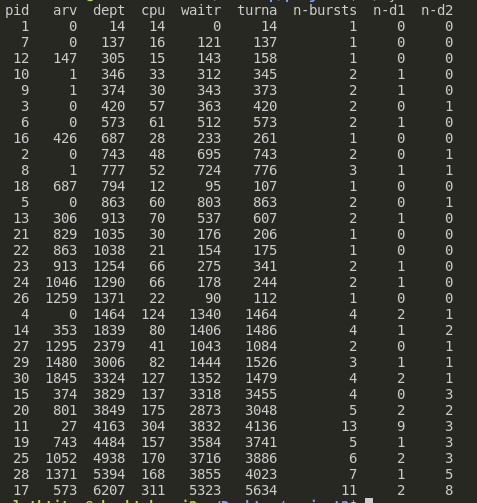
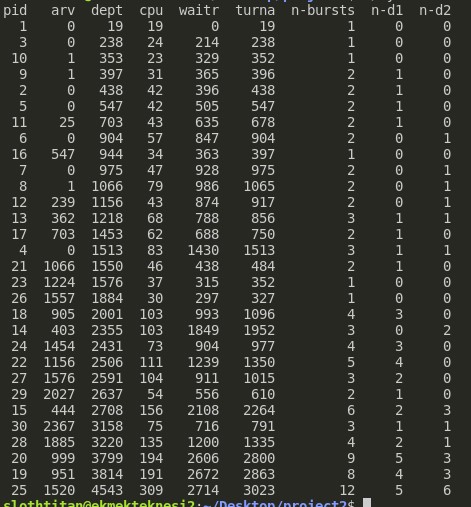


RR (fixed)

RR (uniform)



RR (exponential)



From these outputs, we received following averages:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | FCFS  fixed | FCFS  uniform | FCFS  exp. | SJF  fixed | SJF  uniform | SJF  exp. | RR  fixed | RR  uniform | RR  exp. |
| Waiting | 704 | 901.43 | 1062.5 | 786.3 | 757.06 | 596.56 | 887.6 | 1448.5 | 950.43 |
| Turnaround | 825.26 | 1001.66 | 1157.63 | 856.3 | 830.2 | 644.03 | 963.6 | 1556 | 1042.81 |

Results

According to our experiment results, SJF algorithm provides faster processing than every other algorithm for almost every burst type. By decreasing the waiting time of shorter processes, SJF algorithm decreases the waiting and turnaround time for the processes and therefore, becomes the most efficient algorithm among 3. In addition, by changing the time quantum, different results could be obtained by the experiments.