EOPSY TASK 3

SCHEDULING

LAKSMAN SIVARAM SENTHILKUMAR
317289

CONTENTS

1 INTRODUCTION

2 SCHEDULING

2.1 First-Come, First-Served (FCFS) Scheduling

2.2 Shortest-Job-Next (SJN) Scheduling

2.3 Priority Scheduling

2.4 Shortest Remaining Time

2.5 Round Robin(RR) Scheduling

2.6 Multiple-Level Queues Scheduling

3 FIRST-COME FIRST-SERVED SCHEDULING

3.1 CHARACTERISTICS

3.2 ADVANTAGES

3.3 DISADVANTAGES

3.4 CONVOY EFFECT

4 SIMULATIONS

4.1 1 PROCESS

4.2 2 PROCESS

4.3 5 PROCESS

4.4 10 PROCESS

4.5 CUSTOM CASES

5 CONCLUSION

6 REFERENCES

1 INTRODUCTION

The objective of this task is to get familiarized with scheduling algorithms and techniques, specifically the First-Come First-Served(FCFS) Batch (Non preemptive) scheduling.

2 SCHEDULING

Scheduling algorithms schedule processes on the processor in an efficient and effective manner. There are six popular process scheduling algorithms, namely:-

1. First-Come, First-Served (FCFS) Scheduling
2. Shortest-Job-Next (SJN) Scheduling
3. Priority Scheduling
4. Shortest Remaining Time
5. Round Robin(RR) Scheduling
6. Multiple-Level Queues Scheduling

These algorithms are either non-preemptive or preemptive. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

2.1 First Come First Serve (FCFS)

- Jobs are executed on a first come, first serve basis.
- It is a non-preemptive, preemptive scheduling algorithm.
- Easy to understand and implement.
- Its implementation is based on the FIFO queue.
- Poor in performance as average wait time is high.

2.2 Shortest Job Next (SJN)

- This is also known as the shortest job first, or SJF.
- This is a non-preemptive, preemptive scheduling algorithm.
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.
- Impossible to implement in interactive systems where required CPU time is not known.
- The processor should know in advance how much time the process will take.

2.3 Priority Based 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 the highest priority is to be executed first and so on.
- Processes with the same priority are executed on a first come first served basis.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.

2.4 Shortest Remaining Time

- Shortest remaining time (SRT) is the preemptive version of the SJN algorithm.
- The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion.
- Impossible to implement in interactive systems where required CPU time is not known.
- It is often used in batch environments where short jobs need to give preference.

2.5 Round Robin Scheduling

- Round Robin is the preemptive process scheduling algorithm.
- Each process is provided a fixed time to execute, it is called a quantum.
- Once a process is executed for a given time period, it is preempted and another process executes for a given time period.
- Context switching is used to save states of preempted processes.

2.6 Multiple-Level Queues Scheduling

- Multiple-level queues are not an independent scheduling algorithm. They make use of other existing algorithms to group and schedule jobs with common characteristics.
- Multiple queues are maintained for processes with common characteristics.
- Each queue can have its own scheduling algorithms.
- Priorities are assigned to each queue.

3 FIRST-COME FIRST-SERVE SCHEDULING

FCFS is a non-preemptive and preemptive scheduling algorithm that is easy to understand and use. In this, the process which reaches first is executed first, or in other words, the process which requests first for a CPU gets the CPU first.

FCFS is similar to the FIFO queue data structure. In FCFS, the element which is added in the queue first will leave first. It is used in Batch Operating Systems.

3.1 Characteristics

- FCFS is simple to use and implement.
- In FCFS, jobs are executed in a First-come First-serve (FCFS) manner.
- FCFS supports both preemptive as well as non-preemptive scheduling algorithms.
- FCFS is poor in performance due to high waiting times.

3.2 Advantages

- Easy to program
- First come, first serve
- Simple scheduling algorithm.

3.3 Disadvantages

- Because of their non-preemptive nature, the problem of starvation arises.
- More Average waiting time.
- Due to its simplicity, FCFS is not effective.
- FCFS is not the ideal scheduling for the time-sharing system.
- FCFS is a Non-Preemptive Scheduling algorithm, so allocating the CPU to a process will never release the CPU until it completes its execution.
- In FCFS, it is not possible to use the resources in a parallel manner, which causes the convoy effect, so the resource utilization is poor in FCFS.

3.4 Convoy Effect

In FCFS, Convoy Effect is a condition that arises in the FCFS Scheduling algorithm when one process holds the CPU for a long time, and another process can get the CPU only when the process holding the CPU finishes its execution. Due to this, resource utilization is poor and also affects the performance of the operating system.

4 SIMULATIONS

4.1 1 PROCESS

CONFIGURATION

```
numprocess 1
meandev 2000
standdev 0
process 500
runtime 10000
```

PROCESS SUMMARY

```
Process: 0 registered... (2000 500 0 0)
Process: 0 I/O blocked... (2000 500 500 500)
Process: 0 registered... (2000 500 500 500)
Process: 0 I/O blocked... (2000 500 1000 1000)
Process: 0 registered... (2000 500 1000 1000)
Process: 0 I/O blocked... (2000 500 1500 1500)
Process: 0 registered... (2000 500 1500 1500)
Process: 0 completed... (2000 500 2000 2000)
```

RESULT SUMMARY

```
Scheduling Type: Batch (Nonpreemptive)
Scheduling Name: First-Come First-Served
Simulation Run Time: 2000
Mean: 2000
Standard Deviation: 0
Process #  CPU Time   IO Blocking     CPU Completed   CPU Blocked
0          2000 (ms)  500 (ms)   2000 (ms)  3 times
```

OBSERVATION

The process is blocked 3 times, every 500ms and has a total runtime of:

*4 parts * 500 ms = 2000 ms*

4.2 2 PROCESS

CONFIGURATION

```
numprocess 2
meandev 2000
standdev 0
process 500
process 500
runtime 10000
```

PROCESS SUMMARY

```
Process: 0 registered... (2000 500 0 0)
Process: 0 I/O blocked... (2000 500 500 500)
Process: 1 registered... (2000 500 0 0)
Process: 1 I/O blocked... (2000 500 500 500)
Process: 0 registered... (2000 500 500 500)
Process: 0 I/O blocked... (2000 500 1000 1000)
Process: 1 registered... (2000 500 500 500)
Process: 1 I/O blocked... (2000 500 1000 1000)
Process: 0 registered... (2000 500 1000 1000)
Process: 0 I/O blocked... (2000 500 1500 1500)
Process: 1 registered... (2000 500 1000 1000)
Process: 1 I/O blocked... (2000 500 1500 1500)
Process: 0 registered... (2000 500 1500 1500)
Process: 0 completed... (2000 500 2000 2000)
Process: 1 registered... (2000 500 1500 1500)
Process: 1 completed... (2000 500 2000 2000)
```

RESULT SUMMARY

```
Scheduling Type: Batch (Nonpreemptive)
Scheduling Name: First-Come First-Served
Simulation Run Time: 4000
Mean: 2000
Standard Deviation: 0
Process #  CPU Time   IO Blocking     CPU Completed   CPU Blocked
0          2000 (ms)  500 (ms)   2000 (ms)  3 times
1          2000 (ms)  500 (ms)   2000 (ms)  3 times
```

OBSERVATION

Using non preemptive (also called cooperative) scheduling, specifically the First-Come First-Served algorithm, once the CPU time is allocated to a process, the process ‘holds’ the CPU until it gets terminated or blocked (I/O blocked in the discussed laboratory project). As the name suggests, the process first to come has priority to be finished first.

The total run time (4000 ms) was equal to the sum of the two processes, which proves the previous sentences, as they were executed in sequential manner (thus CPU Blocked equal to 3 for both of them). Although the duration of simulation was specified to 10000 ms in the config, the programme finished earlier, as every process is blocked 3 times, dividing the execution into 4 parts:

*4 parts * 500 ms = 2000 ms, 2000 ms * 2 processes = 4000 ms*

4.3 5 PROCESS

CONFIGURATION

```
numprocess 5
meandev 2000
standdev 0
process 500
process 500
process 500
process 500
process 500
runtime 10000
```

PROCESS SUMMARY

```
Process: 0 registered... (2000 500 0 0)
Process: 0 I/O blocked... (2000 500 500 500)
Process: 1 registered... (2000 500 0 0)
Process: 1 I/O blocked... (2000 500 500 500)
Process: 0 registered... (2000 500 500 500)
Process: 0 I/O blocked... (2000 500 1000 1000)
Process: 1 registered... (2000 500 500 500)
Process: 1 I/O blocked... (2000 500 1000 1000)
Process: 0 registered... (2000 500 1000 1000)
Process: 0 I/O blocked... (2000 500 1500 1500)
Process: 1 registered... (2000 500 1000 1000)
Process: 1 I/O blocked... (2000 500 1500 1500)
Process: 0 registered... (2000 500 1500 1500)
Process: 0 completed... (2000 500 2000 2000)
```

```
Process: 1 registered... (2000 500 1500 1500)
Process: 1 completed... (2000 500 2000 2000)
Process: 2 registered... (2000 500 0 0)
Process: 2 I/O blocked... (2000 500 500 500)
Process: 3 registered... (2000 500 0 0)
Process: 3 I/O blocked... (2000 500 500 500)
Process: 2 registered... (2000 500 500 500)
Process: 2 I/O blocked... (2000 500 1000 1000)
Process: 3 registered... (2000 500 500 500)
Process: 3 I/O blocked... (2000 500 1000 1000)
Process: 2 registered... (2000 500 1000 1000)
Process: 2 I/O blocked... (2000 500 1500 1500)
Process: 3 registered... (2000 500 1000 1000)
Process: 3 I/O blocked... (2000 500 1500 1500)
Process: 2 registered... (2000 500 1500 1500)
Process: 2 completed... (2000 500 2000 2000)
Process: 3 registered... (2000 500 1500 1500)
Process: 3 completed... (2000 500 2000 2000)
Process: 4 registered... (2000 500 0 0)
Process: 4 I/O blocked... (2000 500 500 500)
Process: 4 registered... (2000 500 500 500)
Process: 4 I/O blocked... (2000 500 1000 1000)
Process: 4 registered... (2000 500 1000 1000)
Process: 4 I/O blocked... (2000 500 1500 1500)
Process: 4 registered... (2000 500 1500 1500)
```

RESULT SUMMARY

```
Scheduling Type: Batch (Nonpreemptive)
Scheduling Name: First-Come First-Served
Simulation Run Time: 10000
Mean: 2000
Standard Deviation: 0
Process #  CPU Time   IO Blocking     CPU Completed    CPU Blocked
0          2000 (ms)  500 (ms)   2000 (ms)  3 times
1          2000 (ms)  500 (ms)   2000 (ms)  3 times
2          2000 (ms)  500 (ms)   2000 (ms)  3 times
3          2000 (ms)  500 (ms)   2000 (ms)  3 times
4          2000 (ms)  500 (ms)   2000 (ms)  3 times
```

OBSERVATION
Continuing the discussion of the nonpreemptive mode, when a process is blocked by the next one, the simulation goes back to the previous, partially executed process. Therefore, the processes are executed in pairs: 0-1, 2-3, and the 4 th process has no pair due to an odd number of total processes (5).

Total runtime (10 000 ms) should be just enough to run all the processes, as the total simulation run time is equal to this number, but the simulation software exits before the last process could finish. This is shown in a custom case later.

4.3 10 PROCESS

CONFIGURATION
```
numprocess 10
meandev 2000
standdev 0
process 500
process 500
process 500
process 500
process 500
process 500
process 500
process 500
process 500
process 500
runtime 10000
```

PROCESS SUMMARY
```
Process: 0 registered... (2000 500 0 0)
Process: 0 I/O blocked... (2000 500 500 500)
Process: 1 registered... (2000 500 0 0)
Process: 1 I/O blocked... (2000 500 500 500)
Process: 0 registered... (2000 500 500 500)
Process: 0 I/O blocked... (2000 500 1000 1000)
```

```
Process: 1 registered... (2000 500 500 500)
Process: 1 I/O blocked... (2000 500 1000 1000)
Process: 0 registered... (2000 500 1000 1000)
Process: 0 I/O blocked... (2000 500 1500 1500)
Process: 1 registered... (2000 500 1000 1000)
Process: 1 I/O blocked... (2000 500 1500 1500)
Process: 0 registered... (2000 500 1500 1500)
Process: 0 completed... (2000 500 2000 2000)
Process: 1 registered... (2000 500 1500 1500)
Process: 1 completed... (2000 500 2000 2000)
Process: 2 registered... (2000 500 0 0)
Process: 2 I/O blocked... (2000 500 500 500)
Process: 3 registered... (2000 500 0 0)
Process: 3 I/O blocked... (2000 500 500 500)
Process: 2 registered... (2000 500 500 500)
Process: 2 I/O blocked... (2000 500 1000 1000)
Process: 3 registered... (2000 500 500 500)
Process: 3 I/O blocked... (2000 500 1000 1000)
Process: 2 registered... (2000 500 1000 1000)
Process: 2 I/O blocked... (2000 500 1500 1500)
Process: 3 registered... (2000 500 1000 1000)
Process: 3 I/O blocked... (2000 500 1500 1500)
Process: 2 registered... (2000 500 1500 1500)
Process: 2 completed... (2000 500 2000 2000)
Process: 3 registered... (2000 500 1500 1500)
Process: 3 completed... (2000 500 2000 2000)
Process: 4 registered... (2000 500 0 0)
Process: 4 I/O blocked... (2000 500 500 500)
Process: 5 registered... (2000 500 0 0)
Process: 5 I/O blocked... (2000 500 500 500)
Process: 4 registered... (2000 500 500 500)
Process: 4 I/O blocked... (2000 500 1000 1000)
Process: 5 registered... (2000 500 500 500)
```

RESULT SUMMARY

```
Scheduling Type: Batch (Nonpreemptive)
Scheduling Name: First-Come First-Served
Simulation Run Time: 10000
Mean: 2000
Standard Deviation: 0
Process #  CPU Time   IO Blocking     CPU Completed    CPU Blocked
0          2000 (ms)  500 (ms)   2000 (ms)  3 times
1          2000 (ms)  500 (ms)   2000 (ms)  3 times
2          2000 (ms)  500 (ms)   2000 (ms)  3 times
3          2000 (ms)  500 (ms)   2000 (ms)  3 times
4          2000 (ms)  500 (ms)   2000 (ms)  3 times
```

OBSERVATION

The first notable thing is that the set up run time (10 000 ms) was not enough to execute all the processes (as only 0-3 (inclusive) have finished totally). The processes 4 and 5 were executed only partially (approximately 2/3 and 1/3, consequently), as processes 6-9 (inclusive) haven't even started due to lack of time. Knowing that each process takes 2000 ms to execute, we would need at least 20000 ms to fully execute all 10 of them.

CUSTOM CASES

CASE 1

CONFIGURATION

```
numprocess 5
meandev 2000
standdev 0
process 500
process 500
process 500
process 500
process 500
runtime 10001
```

PROCESS SUMMARY

```
Process: 0 registered... (2000 500 0 0)
Process: 0 I/O blocked... (2000 500 500 500)
Process: 1 registered... (2000 500 0 0)
Process: 1 I/O blocked... (2000 500 500 500)
Process: 0 registered... (2000 500 500 500)
Process: 0 I/O blocked... (2000 500 1000 1000)
Process: 1 registered... (2000 500 500 500)
Process: 1 I/O blocked... (2000 500 1000 1000)
Process: 0 registered... (2000 500 1000 1000)
Process: 0 I/O blocked... (2000 500 1500 1500)
Process: 1 registered... (2000 500 1000 1000)
Process: 1 I/O blocked... (2000 500 1500 1500)
Process: 0 registered... (2000 500 1500 1500)
Process: 0 completed... (2000 500 2000 2000)
Process: 1 registered... (2000 500 1500 1500)
Process: 1 completed... (2000 500 2000 2000)
Process: 2 registered... (2000 500 0 0)
Process: 2 I/O blocked... (2000 500 500 500)
Process: 3 registered... (2000 500 0 0)
Process: 3 I/O blocked... (2000 500 500 500)
Process: 2 registered... (2000 500 500 500)
Process: 2 I/O blocked... (2000 500 1000 1000)
Process: 3 registered... (2000 500 500 500)
Process: 3 I/O blocked... (2000 500 1000 1000)
Process: 2 registered... (2000 500 1000 1000)
Process: 2 I/O blocked... (2000 500 1500 1500)
Process: 3 registered... (2000 500 1000 1000)
Process: 3 I/O blocked... (2000 500 1500 1500)
Process: 2 registered... (2000 500 1500 1500)
Process: 2 completed... (2000 500 2000 2000)
Process: 3 registered... (2000 500 1500 1500)
Process: 3 completed... (2000 500 2000 2000)
Process: 4 registered... (2000 500 0 0)
Process: 4 I/O blocked... (2000 500 500 500)
Process: 4 registered... (2000 500 500 500)
Process: 4 I/O blocked... (2000 500 1000 1000)
Process: 4 registered... (2000 500 1000 1000)
```

```
Process: 4 I/O blocked... (2000 500 1500 1500)
Process: 4 registered... (2000 500 1500 1500)
Process: 4 completed... (2000 500 2000 2000)
```

RESULT SUMMARY

```
Scheduling Type: Batch (Nonpreemptive)
Scheduling Name: First-Come First-Served
Simulation Run Time: 10000
Mean: 2000
Standard Deviation: 0
Process #  CPU Time   IO Blocking     CPU Completed   CPU Blocked
0          2000 (ms)  500 (ms)   2000 (ms)  3 times
1          2000 (ms)  500 (ms)   2000 (ms)  3 times
2          2000 (ms)  500 (ms)   2000 (ms)  3 times
3          2000 (ms)  500 (ms)   2000 (ms)  3 times
4          2000 (ms)  500 (ms)   2000 (ms)  3 times
```

OBSERVATION

This test is to show that when the process runtime is exactly equal to the simulation time, the completion of the last process is not registered as the simulation ends just before that. So, here in this case, modifying the simulation time to just one over the process runtime from 5 process tests, we can see that the completion of the 5th process is registered.

CASE 2

CONFIGURATION

```
numprocess 5
meandev 2000
standdev 0
process 100
process 200
process 250
process 400
process 500
runtime 10001
```

PROCESS SUMMARY

```
Process: 0 registered... (2000 100 0 0)
Process: 0 I/O blocked... (2000 100 100 100)
Process: 1 registered... (2000 200 0 0)
Process: 1 I/O blocked... (2000 200 200 200)
Process: 0 registered... (2000 100 100 100)
Process: 0 I/O blocked... (2000 100 200 200)
Process: 1 registered... (2000 200 200 200)
Process: 1 I/O blocked... (2000 200 400 400)
Process: 0 registered... (2000 100 200 200)
Process: 0 I/O blocked... (2000 100 300 300)
Process: 1 registered... (2000 200 400 400)
Process: 1 I/O blocked... (2000 200 600 600)
Process: 0 registered... (2000 100 300 300)
Process: 0 I/O blocked... (2000 100 400 400)
Process: 1 registered... (2000 200 600 600)
Process: 1 I/O blocked... (2000 200 800 800)
Process: 0 registered... (2000 100 400 400)
Process: 0 I/O blocked... (2000 100 500 500)
Process: 1 registered... (2000 200 800 800)
Process: 1 I/O blocked... (2000 200 1000 1000)
Process: 0 registered... (2000 100 500 500)
Process: 0 I/O blocked... (2000 100 600 600)
Process: 1 registered... (2000 200 1000 1000)
Process: 1 I/O blocked... (2000 200 1200 1200)
Process: 0 registered... (2000 100 600 600)
Process: 0 I/O blocked... (2000 100 700 700)
Process: 1 registered... (2000 200 1200 1200)
Process: 1 I/O blocked... (2000 200 1400 1400)
Process: 0 registered... (2000 100 700 700)
Process: 0 I/O blocked... (2000 100 800 800)
Process: 1 registered... (2000 200 1400 1400)
Process: 1 I/O blocked... (2000 200 1600 1600)
Process: 0 registered... (2000 100 800 800)
Process: 0 I/O blocked... (2000 100 900 900)
Process: 1 registered... (2000 200 1600 1600)
Process: 1 I/O blocked... (2000 200 1800 1800)
```

```
Process: 0 registered... (2000 100 900 900)
Process: 0 I/O blocked... (2000 100 1000 1000)
Process: 1 registered... (2000 200 1800 1800)
Process: 1 completed... (2000 200 2000 2000)
Process: 0 registered... (2000 100 1000 1000)
Process: 0 I/O blocked... (2000 100 1100 1100)
Process: 2 registered... (2000 250 0 0)
Process: 2 I/O blocked... (2000 250 250 250)
Process: 0 registered... (2000 100 1100 1100)
Process: 0 I/O blocked... (2000 100 1200 1200)
Process: 2 registered... (2000 250 250 250)
Process: 2 I/O blocked... (2000 250 500 500)
Process: 0 registered... (2000 100 1200 1200)
Process: 0 I/O blocked... (2000 100 1300 1300)
Process: 2 registered... (2000 250 500 500)
Process: 2 I/O blocked... (2000 250 750 750)
Process: 0 registered... (2000 100 1300 1300)
Process: 0 I/O blocked... (2000 100 1400 1400)
Process: 2 registered... (2000 250 750 750)
Process: 2 I/O blocked... (2000 250 1000 1000)
Process: 0 registered... (2000 100 1400 1400)
Process: 0 I/O blocked... (2000 100 1500 1500)
Process: 2 registered... (2000 250 1000 1000)
Process: 2 I/O blocked... (2000 250 1250 1250)
Process: 0 registered... (2000 100 1500 1500)
Process: 0 I/O blocked... (2000 100 1600 1600)
Process: 2 registered... (2000 250 1250 1250)
Process: 2 I/O blocked... (2000 250 1500 1500)
Process: 0 registered... (2000 100 1600 1600)
Process: 0 I/O blocked... (2000 100 1700 1700)
Process: 2 registered... (2000 250 1500 1500)
Process: 2 I/O blocked... (2000 250 1750 1750)
Process: 0 registered... (2000 100 1700 1700)
Process: 0 I/O blocked... (2000 100 1800 1800)
Process: 2 registered... (2000 250 1750 1750)
Process: 2 completed... (2000 250 2000 2000)
Process: 0 registered... (2000 100 1800 1800)
Process: 0 I/O blocked... (2000 100 1900 1900)
```

```
Process: 3 registered... (2000 400 0 0)
Process: 3 I/O blocked... (2000 400 400 400)
Process: 0 registered... (2000 100 1900 1900)
Process: 0 completed... (2000 100 2000 2000)
Process: 3 registered... (2000 400 400 400)
Process: 3 I/O blocked... (2000 400 800 800)
Process: 4 registered... (2000 500 0 0)
Process: 4 I/O blocked... (2000 500 500 500)
Process: 3 registered... (2000 400 800 800)
Process: 3 I/O blocked... (2000 400 1200 1200)
Process: 4 registered... (2000 500 500 500)
Process: 4 I/O blocked... (2000 500 1000 1000)
Process: 3 registered... (2000 400 1200 1200)
Process: 3 I/O blocked... (2000 400 1600 1600)
Process: 4 registered... (2000 500 1000 1000)
Process: 4 I/O blocked... (2000 500 1500 1500)
Process: 3 registered... (2000 400 1600 1600)
Process: 3 completed... (2000 400 2000 2000)
Process: 4 registered... (2000 500 1500 1500)
Process: 4 completed... (2000 500 2000 2000)
```

RESULT SUMMARY

```
Scheduling Type: Batch (Nonpreemptive)
Scheduling Name: First-Come First-Served
Simulation Run Time: 10000
Mean: 2000
Standard Deviation: 0
Process #  CPU Time   IO Blocking     CPU Completed   CPU Blocked
0          2000 (ms)  100 (ms)   2000 (ms)  19 times
1          2000 (ms)  200 (ms)   2000 (ms)  9 times
2          2000 (ms)  250 (ms)   2000 (ms)  7 times
3          2000 (ms)  400 (ms)   2000 (ms)  4 times
4          2000 (ms)  500 (ms)   2000 (ms)  3 times
```

OBSERVATION
In this test we set different blocking periods for the 5 processes. As expected, the total runtime is 10000ms, as shown below:
*20 * 100ms = 2000 ms, 10 * 200ms = 2000ms, 8 * 250 = 2000ms, 5 * 400ms = 2000ms, 500 * 4 = 2000ms, total 5 * 2000 = 10000ms.*
Also, we can see that processes have different numbers of I/O blocks based on their block timings.

5 CONCLUSION

First Come First Serve (FCFS) is the easiest and simplest CPU scheduling algorithm, which obviously has its pros and cons. On the presented simulations we can see that due to limitations of time some of the processes might not finish, or even not start. Although this algorithm is primitive, it certainly is really intuitive and easy to understand, though its usage might not be sufficient for some of the applications. The CPU allocation is managed with a FIFO queue.

6 REFERENCES

[1]https://man7.org/linux/man-pages/man7/sched.7.html

[2]https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/6_CPU_Scheduling.html

[3]https://www.ia.pw.edu.pl/~tkruk/edu/eopsy/lab/task3.tgz