Kariharan Thilagakumar 100922048
Loic Touzard 101057279

# SYSC 4001
# Assignment 2

**Part II – Simulator.**

Here we will analyze in many different tests case the behaviour of several schelduler algorithms. We will then try to figure out their strengh, weakness, comparing one to another in many situations.

The schelduler we're talking above is the short-term schelduler. Its job is to elect, quickly, a new process currently in WAITING state, to become the next RUNNING process.

The algorithm implemented are the following :

- FCFS : First Come First Serve. The WAITING state container is a queue FIFO and the next process to run is simply the first that came in the queue.
- PRIORITY : The WAITING state container is a priority heap base on the priority criteria of the processes. The lowest priority is 0 and it goes increasing. The next process to dispatch is the process in WAITING state having the highest priority
- SJF : The Waiting state container is a priority heap based on the totalCPUTime criteria. The next process to be running is the one having its global expected execution duration begin the shortest.

The comparison between these algorithm will be based on several metrics :

- Simulation duration : the total time the simulation ran to execute all the process
- Throughput : The number of process that run per time unit (ms)
- Average Turnaround Time : The average duration taken by a process between its NEW state and its TERMINATED state
- Average Waiting Time : The average time process takes waiting in the READY state.
- Average Response Time : The average time taken by a process between its NEW state and its first IO (or termination if there is none).

## Case 1) The Convoy effect

file : data/inputConvoy.txt

```
Process 1 : arrivalTime 0 | totalCPUTime 3 | IOFrequency 1 | IODuration 5 |
priority 0 | state UNKNOWN
Process 2 : arrivalTime 0 | totalCPUTime 3 | IOFrequency 1 | IODuration 5 |
priority 0 | state UNKNOWN
Process 3 : arrivalTime 2 | totalCPUTime 30 | IOFrequency 14 | IODuration 1 |
priority 0 | state UNKNOWN
Process 4 : arrivalTime 2 | totalCPUTime 3 | IOFrequency 1 | IODuration 5 |
priority 0 | state UNKNOWN
Process 5 : arrivalTime 3 | totalCPUTime 3 | IOFrequency 1 | IODuration 5 |
priority 0 | state UNKNOWN
Process 6 : arrivalTime 3 | totalCPUTime 3 | IOFrequency 1 | IODuration 5 |
priority 0 | state UNKNOWN
Process 7 : arrivalTime 3 | totalCPUTime 3 | IOFrequency 1 | IODuration 5 |
priority 0 | state UNKNOWN
```

We have 7 processes. 6 short processes with frequent IO and one with rare IO but long execution time duration.

```
./Simulator data/inputConvoy.txt data/outputConvoySJF.txt FCFS GANTT
Starting Simulation of 7 processes in NONPREEMPTIVE mode with FCFS algorithm...
P1 | P2 | P3 | P3 | P3 | P3 | P3 | P3 | P3 | P3 | P3 | P3 | P3 | P3 | P3 |
P4 | P5 | P6 | P7 | P1 | P2 | P3 | P3 | P3 | P3 | P3 | P3 | P3 | P3 | P3 |
P3 | P3 | P3 | P3 | P4 | P5 | P6 | P7 | P1 | P2 | P3 | P3 | P4 | P5 | P6 | P7
End of Simulation at 48 ms.
Statistics of the system :
Throughput : 0.15 process/ms
Average Turnaround Time : 42.86 ms
Average Waiting Time : 27.14 ms
Average Response Time : 11.43 ms
CPU Utilization : 100.00 %
```

In this case we can observe what is called the Convoy effect, that is tha main problem non-preemptive algorithm has. It occurs when a long process with rare IO comes to RUNNING state while process with a lot of IO have to wait the long one to terminate its execution. The drawback of this is a low usage of the IO-CPU in parallel with the main CPU.

We can see that at t=2 and t=23 P3 thake the RUNNING state, and during 14ms any other process can run their IO in parallel.

Thisexample runs with FCFS because it is a non-preemptive algorithm, but it also occurs with FCFS or Priority that have no preemption.

## Case 2) Short processes versus long processes

file : inputManyDurations.txt

```
Process 1 : arrivalTime 0 | totalCPUTime 10 | IOFrequency 0 | IODuration 0 |
priority 0 | state UNKNOWN
Process 2 : arrivalTime 0 | totalCPUTime 9 | IOFrequency 0 | IODuration 0 |
priority 0 | state UNKNOWN
Process 3 : arrivalTime 0 | totalCPUTime 8 | IOFrequency 0 | IODuration 0 |
priority 0 | state UNKNOWN
Process 4 : arrivalTime 0 | totalCPUTime 7 | IOFrequency 0 | IODuration 0 |
priority 0 | state UNKNOWN
Process 5 : arrivalTime 0 | totalCPUTime 6 | IOFrequency 0 | IODuration 0 |
priority 0 | state UNKNOWN
Process 6 : arrivalTime 0 | totalCPUTime 5 | IOFrequency 0 | IODuration 0 |
priority 0 | state UNKNOWN
Process 7 : arrivalTime 0 | totalCPUTime 4 | IOFrequency 0 | IODuration 0 |
priority 0 | state UNKNOWN
Process 8 : arrivalTime 0 | totalCPUTime 3 | IOFrequency 0 | IODuration 0 |
priority 0 | state UNKNOWN
Process 9 : arrivalTime 0 | totalCPUTime 2 | IOFrequency 0 | IODuration 0 |
priority 0 | state UNKNOWN
```

Here we have 9 processes arriving at the same time (0ms) but ordered in the inverse order of their duration (longest first in the list). They have no IO operation.

Let's compare the execution between FCFS and SJF.

```
./Simulator data/inputManyDurations.txt data/outputManyDurationsFCFS.txt FCFS
GANTT
Starting Simulation of 9 processes in NONPREEMPTIVE mode with FCFS algorithm...
P1 | P1 | P1 | P1 | P1 | P1 | P1 | P1 | P1 | P1 | P2 | P2 | P2 | P2 | P2 | P2 |
P2 | P2 | P2 | P3 | P3 | P3 | P3 | P3 | P3 | P3 | P3 | P4 | P4 | P4 | P4 | P4 |
P4 | P4 | P5 | P5 | P5 | P5 | P5 | P5 | P6 | P6 | P6 | P6 | P6 | P7 | P7 | P7 |
P7 | P8 | P8 | P8 | P9 | P9
End of Simulation at 54 ms.
Statistics of the system :
Throughput : 0.17 process/ms
Average Turnaround Time : 36.67 ms
Average Waiting Time : 30.67 ms
Average Response Time : 36.67 ms
CPU Utilization : 100.00 %
```

```
./Simulator data/inputManyDurations.txt data/outputManyDurationsSJF.txt SJF
GANTT
Starting Simulation of 9 processes in NONPREEMPTIVE mode with SJF algorithm...
P9 | P9 | P8 | P8 | P8 | P6 | P6 | P6 | P6 | P6 | P5 | P5 | P5 | P5 | P5 | P5 |
P4 | P4 | P4 | P4 | P4 | P4 | P4 | P7 | P7 | P7 | P7 | P3 | P3 | P3 | P3 | P3 |
P3 | P3 | P3 | P2 | P2 | P2 | P2 | P2 | P2 | P2 | P2 | P1 | P1 | P1 | P1 |
P1 | P1 | P1 | P1 | P1 | P1
End of Simulation at 54 ms.
Statistics of the system :
Throughput : 0.17 process/ms
Average Turnaround Time : 24.00 ms
Average Waiting Time : 18.00 ms
Average Response Time : 24.00 ms
CPU Utilization : 100.00 %
```

We can see that FCFS will execute the process in the order of their position in the list, beginning with the longest (P1), finishing with the shortest (P9). On the other hand SJF will execute the

shortest first (P9) finishing with (P1).

If we compare their statistics they have the same total duration, so the same throughput, the same CPU Utilization.

But if we look at the Average waiting time they are very different. FCFS has the greatest value with 30.67 ms whereas SJF as the smallest with 18.00ms. That mean that the process are waiting much more in the READY state with FCFS, indeed in this case all the process have to wait the longest process to terminate first. With SJF the proces wait much less because the shortest come first so the second didn't wait that much, and so on. SJF is the best algorithm for the Waiting time metric, but don't forget that he requires also more information : expected duration of the process, that have to be computed or given.

## Case 3) Many durations but delayed

file1 : inputHappylyDelayed.txt

```
Process 1 : arrivalTime 0 | totalCPUTime 1 | IOFrequency 0 | IODuration 0 |
priority 0 | state UNKNOWN
Process 2 : arrivalTime 1 | totalCPUTime 30 | IOFrequency 0 | IODuration 0 |
priority 0 | state UNKNOWN
```

file2 : inputSadlyDelayed.txt

```
Process 1 : arrivalTime 0 | totalCPUTime 30 | IOFrequency 0 | IODuration 0 |
priority 0 | state UNKNOWN
Process 2 : arrivalTime 1 | totalCPUTime 1 | IOFrequency 0 | IODuration 0 |
priority 0 | state UNKNOWN
```

Here we have 2 process one which length is 30ms, the other length is 1ms.

The difference in these files is their arrival time order. In HappylyDelayed the 1ms arrives first at 0 then the 30ms second at 1ms delay. In SadlyDelayed the 30ms arrives first at 0ms, then the 1ms second at 1ms.

Let's run SJF on it to see how it behaves.

```
./Simulator data/inputHappylyDelayed.txt data/outputHappylyDelayedSJF.txt SJF
GANTT
Starting Simulation of 2 processes in NONPREEMPTIVE mode with SJF algorithm...
P1 | P2 | P2 | P2 | P2 | P2 | P2 | P2 | P2 | P2 | P2 | P2 | P2 | P2 | P2 |
P2 | P2 | P2 | P2 | P2 | P2 | P2 | P2 | P2 | P2 | P2 | P2 | P2 | P2
End of Simulation at 31 ms.
Statistics of the system :
Throughput : 0.06 process/ms
Average Turnaround Time : 15.50 ms
Average Waiting Time : 0.00 ms
Average Response Time : 15.50 ms
CPU Utilization : 100.00 %
```

In the first case all is going perfectly, the 1ms process execute first, when it ends the 30ms process arrives and start running. We have a waiting time of 0ms.

```
./Simulator data/inputSadlyDelayed.txt data/outputSadlyDelayedSJF.txt SJF GANTT
Starting Simulation of 2 processes in NONPREEMPTIVE mode with SJF algorithm...
P1 | P1 | P1 | P1 | P1 | P1 | P1 | P1 | P1 | P1 | P1 | P1 | P1 | P1 | P1 |
P1 | P1 | P1 | P1 | P1 | P1 | P1 | P1 | P1 | P1 | P1 | P1 | P1 | P2
End of Simulation at 31 ms.
Statistics of the system :
Throughput : 0.06 process/ms
Average Turnaround Time : 30.00 ms
Average Waiting Time : 14.50 ms
Average Response Time : 30.00 ms
CPU Utilization : 100.00 %
```

In the second case the 1ms delay makes the 1ms process wait for 29ms before the running process release the CPU... It makes the average waiting time be 14.50ms just because of a 1ms delay... sadly delayed.

This behaviour is due to the abscence of preemption. We wouldn't have this problem with a preemptive SJF because when the 1ms process would arrived while the 30ms process is running, it will interrput the running process and be elected next.

## Case 4) User process versus high priority system process

file : inputPriority.txt

```
Process 1 : arrivalTime 0 | totalCPUTime 5 | IOFrequency 1 | IODuration 1 |
priority 0 | state UNKNOWN
Process 2 : arrivalTime 0 | totalCPUTime 5 | IOFrequency 1 | IODuration 1 |
priority 1 | state UNKNOWN
Process 3 : arrivalTime 0 | totalCPUTime 5 | IOFrequency 1 | IODuration 1 |
priority 0 | state UNKNOWN
Process 4 : arrivalTime 3 | totalCPUTime 3 | IOFrequency 1 | IODuration 1 |
priority 0 | state UNKNOWN
Process 5 : arrivalTime 3 | totalCPUTime 3 | IOFrequency 1 | IODuration 1 |
priority 1 | state UNKNOWN
Process 6 : arrivalTime 3 | totalCPUTime 3 | IOFrequency 1 | IODuration 1 |
priority 3 | state UNKNOWN
Process 7 : arrivalTime 3 | totalCPUTime 10 | IOFrequency 1 | IODuration 1 |
priority 10 | state UNKNOWN
```

This file has 7 processes. 3 processes arrive at t=0ms, they have low priorities ( 0, 1, 0 ). Then 4 processus arrive at t=3ms, 3 of them with low priorities (0, 1, 3) and one (Process 7) with a high priority of 10.

We could consider P1 to P6 to be user process with low priorities and P7 to be an important system process.

We'll test this case with FCFS, SJF, PRIORITY algorithm and pay more attention about P7.

```
./Simulator data/inputPriority.txt data/outputPriorityFCFS.txt FCFS GANTT
Starting Simulation of 7 processes in NONPREEMPTIVE mode with FCFS algorithm...
P1 | P2 | P3 | P1 | P2 | P4 | P5 | P6 | P7 | P3 | P1 | P2 | P4 | P5 | P6 | P7 |
P3 | P1 | P2 | P4 | P5 | P6 | P7 | P3 | P1 | P2 | P7 | P3 | P7 |    | P7 |    |
P7 |    | P7 |    | P7 |    | P7
End of Simulation at 39 ms.
Statistics of the system :
Throughput : 0.18 process/ms
Average Turnaround Time : 24.14 ms
Average Waiting Time : 15.43 ms
Average Response Time : 3.43 ms
CPU Utilization : 87.18 %
```

```
./Simulator data/inputPriority.txt data/outputPrioritySJF.txt SJF GANTT
Starting Simulation of 7 processes in NONPREEMPTIVE mode with SJF algorithm...
P1 | P3 | P2 | P4 | P5 | P4 | P5 | P4 | P5 | P2 | P6 | P3 | P6 | P2 | P6 | P3 |
P2 | P1 | P3 | P2 | P1 | P3 | P1 | P7 | P1 | P7 |    | P7 |    | P7 |    | P7 |
| P7 |    | P7 |    | P7 |    | P7 |    | P7
End of Simulation at 42 ms.
Statistics of the system :
Throughput : 0.17 process/ms
Average Turnaround Time : 18.43 ms
Average Waiting Time : 9.71 ms
Average Response Time : 5.43 ms
CPU Utilization : 80.95 %
```

In these two cases P7 is in a bad situation and end last.

For FCFS it arrives at t=3ms but start running for the first time at t=8ms. Whenever P7 has IO, and come back to READY state it has to wait all the queue before running again.

For SJF it arrives at t=3ms but start running for the first time at t=23ms. He is running last because it is the longest process (10ms) next to the other ones.

```
./Simulator data/inputPriority.txt data/outputPriorityPRIORITY.txt PRIORITY
GANTT
Starting Simulation of 7 processes in NONPREEMPTIVE mode with PRIORITY
algorithm...
P2 | P3 | P2 | P7 | P6 | P7 | P6 | P7 | P6 | P7 | P2 | P7 | P5 | P7 | P2 | P7 |
P5 | P7 | P2 | P7 | P5 | P7 | P4 | P1 | P3 | P4 | P1 | P3 | P4 | P1 | P3 | P1 |
P3 | P1
End of Simulation at 34 ms.
Statistics of the system :
Throughput : 0.21 process/ms
Average Turnaround Time : 22.14 ms
Average Waiting Time : 13.43 ms
Average Response Time : 8.57 ms
CPU Utilization : 100.00 %
```

But with the PRIORITY algorithm P7 arrives at t=3ms, and directly takes the RUNNING states. It stops every seconds to do IO but whenever it is ready again, it goes back to RUNNING states. This lead this process to finish its execution at t=22ms which is very good, and thanks to its high priority. But as a drawback the low priority processes have to wait much more before executing, if many high priority are created regularly, it may lead to starvation of these low priority processes.

## Conclusion :

We here have only non-preemptive algorithm, so the Convoy effect can occurs in particular situations causing a bad parallel use of the ressources.

The FCFS algorithm has the advantage of being fair, next to SJF and PRIORITY (not fair) that can lead to starvation, depending of the spawn rate of certain processes. One other advantage of FCFS is that it is the easiest to implement, and requires no extra information on the processes.

The SJF algorithm optimize the Average Waiting Time metric, by making the long processe to wait a bit instead of the short processes to wait a lot. In its non-preemptive mode it is sensitive to the delays that are badly set, which wouldn't occur in its preemptive mode.

The PRIORITY algorithm allows important processes to be executed the earliest possible, which is important if we consider vital system process and normal user process running at the same time. Consequently, it may not give the best average waiting time.

PRIORITY and SJF algorithm need extra information and more computation, that is obviously a drowback as we are in the short-term schelduler where the overhead time should be the shortest.