

TASK 3 EOPSY

Explanation of the following files:

File "scheduling.conf"

This file is used to configure the program. It's where we can modify all the parameters.

In the first part you can choose the number of process ("**numprocess**") that you want to create.

In the second part we can set "**meandev**" in milliseconds which represent the running time of a process.

In the third part we can modify the "**standdev**" which represent a sort of uncertainty of +/- **x** milliseconds. This uncertainty is applied to the "**runtime**". At the end the running time will be modifying of **x** milliseconds where:

$-|\text{standdev}| < x < |\text{standdev}|$.

In the fourth part we declare all the process like the following: "**process x**" where **x** is in millisecond.

To declare all the processes properly, we should add as many lines as there are processes to create.

Every **x** millisecond, the process is blocked.

And finally, in the fifth part, we can set the running time of the simulation like that: "**runtime x**" where **x** is in milliseconds.

File "Summary-Processes"

This file allows us to see what's going on with all the started processes. One line is describing one instance. On each line we gave the following information: The process number (0, 1, 2, ...), the status (registered..., I/O blocked...).

After these informations there is four values in parentheses.

The first one represents the "**meandev**", the running time of the process.

The second represents the duration between each blocking.

The third represent after how many milliseconds the process has been blocked.

The fourth represent after how many milliseconds the process has been unblocked. (The blocked/unblocked is almost instant.)

File "Summary-Results"

This file allows us see how long the simulation ran and how long all the process should ran. The standard deviation value is also inscribed. Above we have one line for each process with its number id, the supposed running time, the time between each blocking, the duration time which the process ran and the occurred block number.

Case with two processes:

The image displays three terminal windows from a GNU nano 4.3 editor, showing the configuration, results, and process details of a simulation.

Terminal 1: scheduling.conf

```
GNU nano 4.3 scheduling.conf
// # of Process
numprocess 2

// mean delvation temps
meandev 2000

// standard deviation
standdev 0

// process # I/O blocking
process 500
process 500

// duration of the simulation in milliseconds
runtime 10000
```

Terminal 2: Summary-Results

```
GNU nano 4.3 Summary-Results
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
```

Terminal 3: Summary-Processes

```
GNU nano 4.3 Summary-Processes
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 1500 1500)
Process: 0 I/O blocked... (2000 500 1000 1000)
Process: 1 registered... (2000 500 1500 1500)
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)
```

Here we create two processes with the asking parameters as you can see on the first terminal (**scheduling.conf**).

The second terminal (**Summary-Results**) show us that the both processes ran properly. The columns “CPU completed” and “CPU Blocked” testify to it. In fact, we can see that the both processes ran 2000 milliseconds and they have been blocked three times.

The third terminal (**Summary-Processes**) show us a lot about how the simulation is working. The processes are working each has their turn. First one registered then blocked and the second do the same. After the first one registered and block again and so on.

Finally, the processes are not doing their tasks at the same time. The second terminal testify it with the line where “**Simulation Run Time: 4000**” is inscribed. We have to processes running 2000 milliseconds each and the simulation run time is equal to $2000 \times 2 = 4000$ milliseconds.

Case with five processes:

```
maxence@GS65-Stealth: ~/Desktop/task3/work
GNU nano 4.3 Summary-Processes
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 2000 2000)
Process: 2 completed... (2000 500 1500 1500)
Process: 3 registered... (2000 500 2000 2000)
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 I/O blocked... (2000 500 1500 1500)
Process: 4 registered... (2000 500 1500 1500)
Process: 4 registered... (2000 500 1500 1500)
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^D Justify
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell

maxence@GS65-Stealth: ~/Desktop/task3/work
GNU nano 4.3 Summary-Results
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

maxence@GS65-Stealth: ~/Desktop/task3/work
GNU nano 4.3 scheduling.conf
// # of Process
numprocess 5

// mean deviation temps
meandev 2000

// standard deviation
standdev 0

// process # I/O blocking
process 500
process 500
process 500
process 500
process 500

// duration of the simulation in milliseconds
runtime 10000
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^D Justify
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell
```

Here we create five processes still with the same parameters. In the Third terminal (**scheduling.conf**) we can see how to configure the file to create five processes.

In the Second terminal (**Summary-Results**) we can see the same things than for the case with two processes.

The changes are in the first terminal (**Summary-Processes**) where we can observe that processes are working by pairs. Process 0 with process 1, process 2 with process 3 and the fifth process worked alone.

So, now we know that if two processes are done, another pair processes will start working.

Case with ten processes:

The image displays three terminal windows from a GNU nano 4.3 editor, showing the execution of a scheduling simulation with ten processes.

Terminal 1: Summary-Processes

```
maxence@GS65-Stealth: ~/Desktop/task3/work
GNU nano 4.3 Summary-Processes
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 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: 4 registered... (2000 500 1000 1000)
Process: 5 registered... (2000 500 500 500)
```

Terminal 2: Summary-Results

```
maxence@GS65-Stealth: ~/Desktop/task3/work
GNU nano 4.3 Summary-Results
Scheduling Type: Batch (Nonpreemptive)
Scheduling Name: First-Come First-Served
Simulation Run Time: 10000
Mean: 2000
Standard Deviation: 0
Process #    CPU Time    I/O 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)         1000 (ms)         2 times
5            2000 (ms)      500 (ms)         1000 (ms)         1 times
6            2000 (ms)      500 (ms)         0 (ms)            0 times
7            2000 (ms)      500 (ms)         0 (ms)            0 times
8            2000 (ms)      500 (ms)         0 (ms)            0 times
9            2000 (ms)      500 (ms)         0 (ms)            0 times
```

Terminal 3: scheduling.conf

```
maxence@GS65-Stealth: ~/Desktop/task3/work
GNU nano 4.3 scheduling.conf
// # of Processes
numprocess 10

// mean delvation temps
meandev 2000

// standard deviation
standdev 0

// process # I/O blocking
process 500
process 500
process 500
process 500
process 500
process 500
process 500
process 500
process 500
process 500

// duration of the simulation in milliseconds
runtime 10000
```

He we create ten processes still with the same parameters. The third terminal (**scheduling.conf**) show us how to configure the file to create 10 processes.

The second terminal (**Summary-Results**) show us new things. In fact, we can see that all the processes did not run completely if it's not at all. But when we look at it, we can see that four processes have been properly executed (0, 1, 2 & 3). The four first processes ran 2000 milliseconds each and have been blocked 3 times each. But processes 4 & 5 ran only 1000 milliseconds each. The process 4 have been blocked two times and the process 5 have been blocked just one time. When we add all the running time of all processes, we can note that the simulation run time have been properly completed:

$$(4 \times 2000) + (2 \times 1000) = 10000$$

So finally, it's logical that all the processes have not all been executed.

If we increase the **"runtime"** to $10 \times 2000 = 20000$ milliseconds in **"scheduling.conf"** all the processes will be correctly executed.

In the first terminal (**Summary-Processes**) if we try to count the milliseconds, we will arrive to the same conclusion done just before.