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:
-|**standdev**| < **x** < |**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.

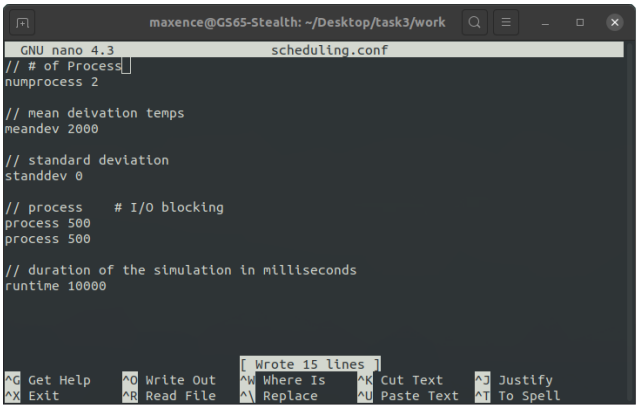
File “Summary-Processes”

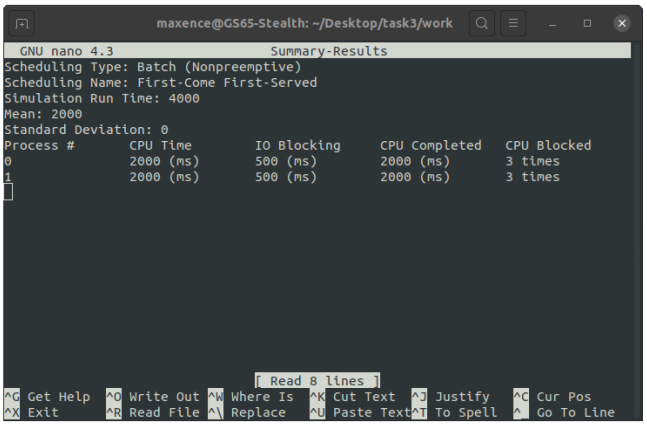
This file allows us to see what’s going on with all the started processes. One line is describing one instance. On each line we gave the following information: The process number (0, 1, 2, …), the status (registered…, I/0 blocked…).
After these informations there is four values in parentheses.
The first 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.)

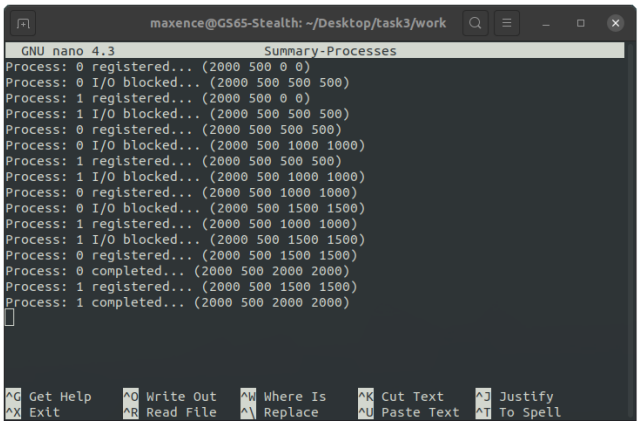
File “Summary-Results”

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:



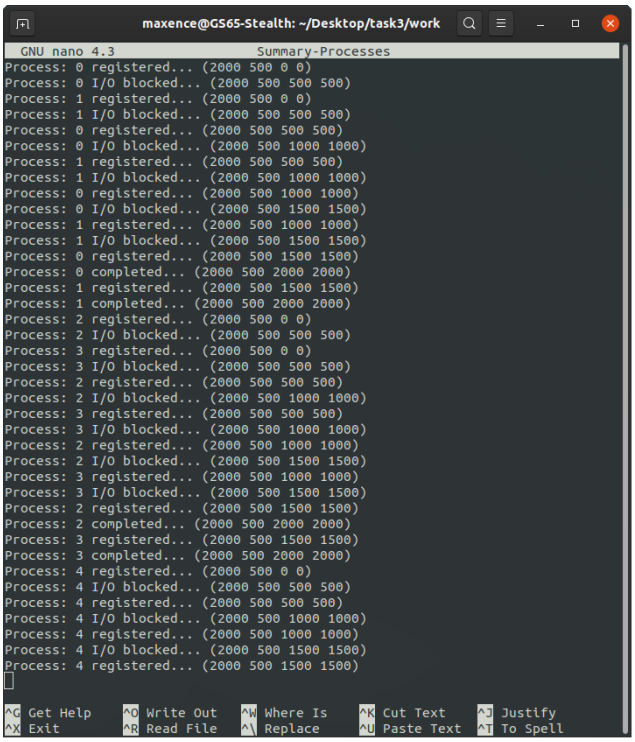


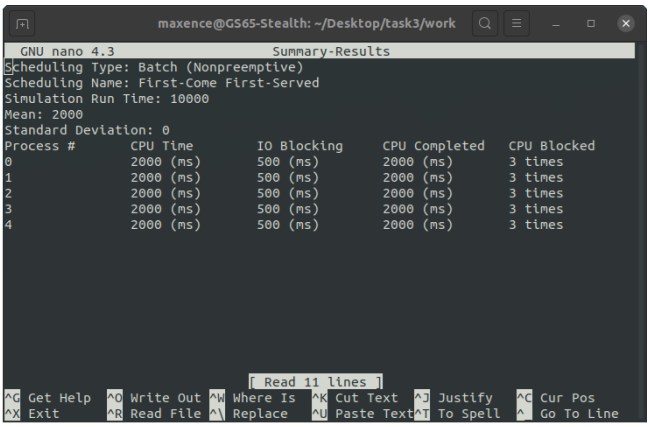


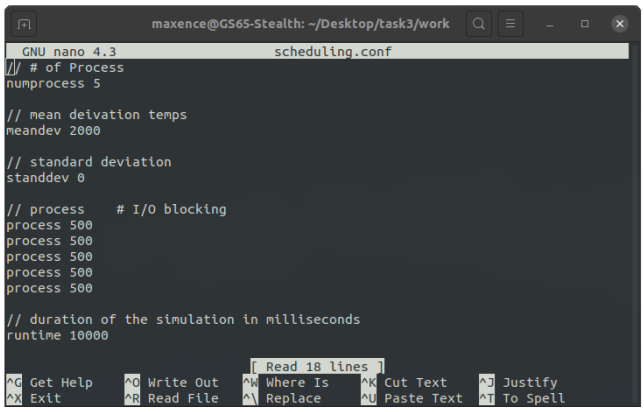
Here we create two processes with the asking parameters as you can see on the first terminal (**scheduling.conf**).
The second terminal (**Summary-Results**) show us that the both processes ran properly. The columns “CPU completed” and “CPU Blocked” testify to it. In fact, we can see that the both processes 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 x 2 = 4000 milliseconds.*

Case with five processes:

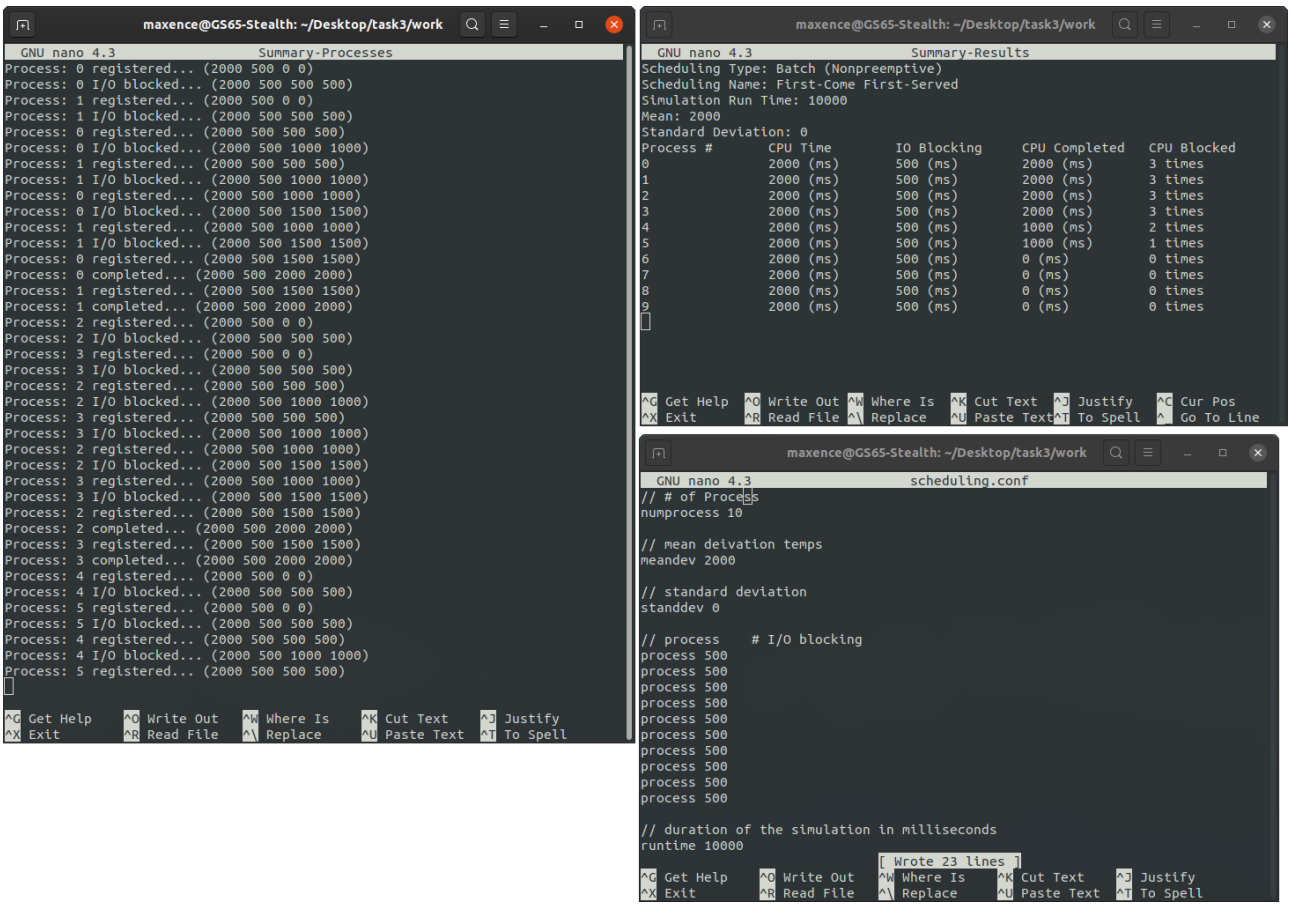






Here we create five processes still with the same parameters. In the Third terminal (**scheduling.conf**) we can see how to configure the file to create five processes.
In the Second terminal (**Summary-Results**) we can see the same things than for the case with two processes.
The changes are in the 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:



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 x 2000) + (2 x 1000) = 10000
So finally, it's logical that all the processes have not all been executed.
If we increase the "**runtime**" to *10 x 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.