EOPSY Lab3- Scheduling

Jakub Prochorec

The lab task involved running a few simulations of processes behavior under scheduling algorithms using MOSS Scheduling Simulator. The simulator was a Java program called Scheduling, which was running based on configuration file scheduling.conf which I was modifying myself according to the task needs. I was running the simulation under Ubuntu Linux.

The following parameters were used for all simulations: each process was supposed to run for 2000ms with standard deviation of 0, being I/O blocked every 500ms, and the total allowed time of each simulation was 10000ms. The three simulations were tested on same conditions, but for the total of 2, 5 or 10 processes, respectively.

The first configuration I was running and examining, with the use of Summary-Processes and Summary-Results output files, was on 2 processes only. As noticed, the “first” process had assigned the ID of 0, the second ID of 1. The result was very much as it would make sense and as I expected – the first process would be registered, and then blocked after 500ms. And immediately after that 500ms the second process was registered and also was running for 500ms before being blocked and scheduler handed over the running to first process once again. This exchange was made until first, and then second process was finished. Both processes had 2000ms CPU time and were blocked 3 times each (at 500, 1000 and 1500ms). The total simulation time was 4000ms (so obviously less than 10000ms which was maximum time allowed).

In the second configuration (5 processes) the second trend became clear. Namely speaking, it turned out that at each period 2 processes were running concurrently ( so process with ID 0 and with ID 1 behaved exactly as in first configuration until 4000ms passed, then processes with ID 2 and 3 started to run concurrently(but only after 0 and 1 already finished), and lastly process with ID 4 finished by running alone. Each process was running for 2000ms and was blocked 3 times (though in Summary-Processes the very completion of last process was not noted at 10000ms mark). The total simulation time was 10000ms (exactly as the limit).

During third and last simulation, 10 processes was the setting. The trend of two concurrent processes continued. The main difference was that the total time that would be needed to run all 10 of those processes exceeded 10000ms(in fact it would need to be 20000ms). But because of that dual concurrency, the result visible in Summary-Results was that processes 0-3 were running for full 2000ms being blocked 3 times, then process 4 was running for total of 1000ms being blocked 2 times, and process 5 was running for 1000ms being blocked 1 time. So processes 4-5 didn’t manage to complete, while processes 6-9 had total runtime of 0ms as they were never even registered.

Observing the simulations, the scheme of scheduling and blocking revealed itself. Each process was first registered, running for a given time, then blocked and the CPU time was handed over to the second process. My guess as of why exactly the number of processes that were running concurrently was 2 is actually quite interesting. In fact, I have a processor with 8 threads in my computer, however I was running the simulation under Ubuntu Linux on a Virtual Machine (and I have 2 processor threads assigned for that in settings). So it would make sense that the scheduling algorithm would make use of the two threads that the system was being assigned during the simulation.