Report on exercise #3

Matteo Corain S256654

Laboratory #2 – System and device programming – A.Y. 2018-19

The proposed solution is functionally identical to the one of the previous exercise, the main difference being the management of the timed wait on the semaphore. In this case, in fact, the function `sem_timedwait()` has been used for the purpose instead of the `wait_with_timeout()` function previously implemented.

With respect to the previous exercise, the needed modifications are all located inside the first thread function; since the `sem_timedwait()` requires as an argument a `struct timespec` whose fields indicate the absolute moment of time (since the beginning of the UNIX epoch) at which the call should timeout, the `thread_runner_1()` function has been modified to perform the following actions:

- Get the current time as a `struct timespec` via the `clock_gettime()` system call (more precise than calling `time(0)`, since it returns the time with nanoseconds granularity), storing it in the `wait_timespec` variable;
- Add the displacement given by the value of `tmax` to the appropriate fields of `wait_timespec`, taking care of the possibility that the nanoseconds field may exceed its maximum value;
- Call `sem_timedwait()` and wait until either the semaphore is unlocked, or the timeout has been reached (call returns `-1` and `errno` is set to `ETIMEDOUT`).