System Programming
4th Laboratory (16 and 18 March 2016)

I

Without suitable synchronization primitives it is difficult to implement correctly the last exercise from last week's lab.
Using a named pipe (FIFO) implement a correct version of the solution.

The **gen_random** that generates random numbers and writes them sequentially in a shared memory region. The program terminates after 10000 numbers were.
the other two programs do the following:

- **count_even** – that reads the shared memory and counts how many even numbers were written by gen_random
- **count_odd** – that reads the shared memory and counts how many odd numbers were written by gen_random

The three processes are not related should be lauched from command line (NOT by fork).

Use the named pipe to offer synchronization between the writer and readers.
How many FIFOs are needed?

References
http://tldp.org/LDP/lpg/node15.html
http://beej.us/guide/bgipc/output/html/multipage/fifos.html

II

Implement the previous system using only FIFOs.
There should be 3 processes: gen_random, count_even and count_odd.
How many FIFOs are needed.

III

Implement a simple benchmark of the various pipe (or FIFOs), POSIX message queues, sockets (UNIX domain) and shared memory (+pipe for sync).
The parent process should send the numbers between 0 and UINT_MAX to the child process (about 16Gb). The child process should count them.
At the end of the transmission and reception, the parent process should print the spent time.