System Programming
5th Laboratory (29 and 31 March 2017)

I

Modify exercise I from Lab 4 so that three different processes verify if a number is multiple of 7, 19 or 7 and 19. Each of the 3 processes will verify one of these characteristics:

- process 1 will verify if the number is multiple of 7
- process 2 will verify if the number is multiple of 19
- process 3 will verify if the number is multiple of 7 and 19

The parent process will generate sequential numbers between 0 and 999999 in a sequential manner and write them to one pipe to be processed by the other processes.
Eacj of the child processes will print its count.

II

Implement a system with multiple processes that verify and print if a number is prime.
The Parent process generates 999999 random numbers and sends them to the various processes.
The child processes do not know how many numbers are generated. The parent process receives as argument (argc/argv) the number of child processes.

II

Modify the previous program so that the child processes do not print the prime values, but send to the parent the its partial count. Only the parent prints the sum on the screen.

IV

Observe the times each of the previous solutions (serial and parallel) take to execute. Explain why the behavior is different in these solutions.

V

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 launched from command line (NOT by fork).

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

VI

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

How many FIFOs are needed.

VII

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.

REFERENCES

http://tldp.org/LDP/lpg/node7.html
http://beej.us/guide/bgipc/output/html/multipage/index.html