Laboratory #3 – Exec(), System() Shared Memory– March 23th 2021

Exercise 1 (Exec): write a C program that performs the following operations:

- Creates a child process that generates a file called random.txt containing 100 random numbers that are separated by the space character.
- When the file is created the program must instantiate three sub processes:
  - The first process must execute the Linux command wc –m to print on the screen the number of bytes of the file.
  - The second process must execute the Linux command wc –w to count the number of words in the file
  - The third process must execute a custom program (to be programmed by your own) designed to compute the average of all generated number.
- The main process terminates when all three children terminate.

Exercise 2 (System): write a C program that launches the **find** command with specified user path as input argument.

Exercise 3 (Shared Memory): Write a multi-process program that evaluates the following math series:

$$\sum_{0}^{n} x^{i}$$

The main program receives the *x* and *N* values as input (set max value for *N* to 5) and creates the set of necessary processes. Each process evaluates a single x^i instance, referring to its index of creation (e.g., process 1 evaluates x^0, process 2 evaluates x^1, and so on). Impose x being a floating-point value and let only the father compute the final results. Once all the processes complete their job, the main program displays the final result. Use the **shared memory** facilities to exchange each child process result with the father.

Exercise 4 (Shared Memory). Write a modified Linux shell based on a client/server architecture. The new shell must have the following characteristics:

- The client receives commands on the command line.
- Every command is sent to the server and executed in background.
- The shell terminates when the user enters the exit command.

Try to implement the client/server communication using **shared memory**.

Exercise 5 (Shared Memory in POSIX). A program partitions the computation between 2 processes (a parent and a child):

- The parent reads from a file (called “number.dat”) a list of integer numbers bigger than 0 separated by space and sends to the child
- The child prints the received numbers.
- There are no repeated numbers.
- The “number.dat” ends with 0.

Use the **shared memory in POSIX.**