Exercise 1

- Write a program *ex1.c* which creates two new processes, any of them should not be a child of the other (their parents should be the "main" process).
- Print the ID of each three processes created in the program and their parent IDs. Print the execution time for each process in milliseconds. The execution time starts from the first instruction after *fork*.
- Submit the file *ex1.c* with a supplement script file *ex1.sh* to run the program
- **Hint:** Use the library **time.h** and function **clock**. Check the example https://cplusplus.com/reference/ctime/clock/.
- **Hint:** Use the library **unistd.h** for process creation.

Exercise 2 (1/2)

- Write a program *ex2.c* creates two vectors u and v using arrays of *120* elements. Set random values to the vectors from the range *[0-99]* using function rand.
- Create an array of n processes where n is an input number from stdin.
- Calculate the dot product of the vectors by the processes. Distribute the calculation among the processes **equally**. For instance, if $n = 6$ then each process will contribute in calculating only 20 components of the result vector. The final aggregation result should be printed by only the "main" process.
- Your program should open a file *temp.txt* for reading and writing computation results. This file will be the shared medium among the processes.

Exercise 2 (2/2)

- Each child process should write the result of its calculation to the shared file.
- The "main" process will read the calculation results from the file and aggregate them.
- All processes have access to both vectors u, v and the shared file.
- Submit the file *ex2.c*, *temp.txt* with a supplement script file *ex2.sh* to run the program
- **Hint:** The dot product of u and v is : $u * v = \sum_{i,j} u_i * v_j$
- **Hint:** Make sure that the parent process waits for all its children.
- We did not determine a specific format for writing the computation results in the file. You may write each process result in a new line or in a one line separated with commas...etc
- **Note:** We assume that $n = 2k$ where $k \in \{1, 2, 3, 4, 5\}$.

Exercise 3

In this exercise, you will do an experiment, observe and record the results.

- Write a program *ex3.c* that calls fork() in a loop for n times and sleeps for 5 seconds after each call. Run the program as a background process and then run **pstree** command several times. The variable n should be read from the **command line** as an argument and not from stdin.
- Run the program for n=3. Look at the output and tell how many processes are created. Explain the result.
- Run the program so that it would call fork() 5 times. See how the result changes.
- Explain the results in each run and the difference between results in your own words. Add the explanation to a text file *ex3.txt*.
- Write a script *ex3.sh* which includes the commands to replicate the experiment for both runs.
- Submit *ex3.c*, *ex3.sh* and *ex3.txt*.

Exercise 4

- Write a program *ex4.c* to create your own simplistic shell. It should read user input and be able to run a command without arguments, such as pwd, ls, top, pstree and so on.
- Extend your previous code to handle commands with arguments and run the processes in background (another process). Do not use here the shell operator & for running the process in the background.
- **Note:** Here, it is not permitted to use the library function *system* but you should use one of the system calls for executing the commands such as *execve*.
- **Hint:** use man fork and man execve.
- Submit *ex4.c* and *ex4.sh* script to run your program.