Birzeit University - Faculty of Engineering and Technology
Electrical & Computer Engineering Department - ENCS4330
Real-Time Applications & Embedded Systems - 2^{nd} semester - 2022/2023

Project #1
Signals & Pipes under Unix/Linux
Due: May 8, 2023

Instructor: Dr. Hanna Bullata

The competition

We would like to create a multi-processing application based on the signals and pipes/fifos facilities under Linux. The idea is to have a parent process fork 5 children processes. The first 2 processes are partners that belong to team 1 while processes 3 & 4 are partners that belong to team 2. Process 5 is the co-processor that helps its parent doing calculations.

The competition can be explained as follows:

1. On startup, the parent process will fork the 5 children processes.
2. The parent process will write down 2 comma-separated integer values in a file called `range.txt`. These 2 values are the min and max that makes up a range.
3. Once instructed to do so, the first 4 processes will generate 4 floating values in the range decided on by the parent. Each process will note down its generated value in a file that is named after its pid. Each process should inform the parent once its value is ready to be picked.
4. Once the parent process gets the 4 values, it will send them as a message (e.g. comma-separated message) through a pipe to its fifth child process (the co-processor).
5. The co-processor will sum the first 2 values and create the value Sum1. In addition, it will sum the third and fourth values to create the value Sum2. The co-processor will then return the 2 sums to the parent as a message (e.g. comma-separated message).
6. The parent process decides which team has the higher sum.
7. The application runs the steps 2 ... 6 as many times as instructed by the user. If no argument is provided, assume 5 rounds.
8. On termination, the parent declares the winner of the competition, kills all its child processes, removes any allocated resource and exits.

What you should do

- Write the code for the parent and the children. The parent's code should be written in the file `parent.c` while the children's code should be written in the file `child.c`.
- Check that your program is bug-free. Use the `gdb` debugger in case you are having problems during writing the code (and most probably you will :-). In such a case, compile your code using the `-g` option of the `gcc`.
- Use graphics elements from opengl library in order to best illustrate the application. Nothing fancy, just simple and elegant elements are enough.
- Insert delays between rounds (e.g. few seconds).
- Send the zipped folder that contains your source code and your executable before the deadline. If the deadline is reached and you are still having problems with your code, just send it as is!