Birzeit University - Faculty of Engineering and Technology
Electrical & Computer Engineering Department - ENCS4330
Real-Time Applications & Embedded Systems - 1^{st} semester - 2021/22

Project #1
Signals & pipes under Unix/Linux
Due: October 17, 2021

Instructor: Dr. Hanna Bullata

Guessing Game Simulation

We would like to create a multi-processing application that simulates a guessing game between 2 processes using signals and pipes. A parent process will create 2 children processes and 1 referee process. We'll call the first child process P_1 and the second child process P_2. We will refer to the referee process by the symbol R.

The behavior of the whole system should be as follows:

1. Upon creation, the children P_1 and P_2 will be sensitive to signal `SIGUSR1`. The parent process will be sensitive to both signals `SIGINT` and `SIGQUIT`.
2. The parent process will ask P_1 and P_2 to pick each 10 random integer numbers in the range $[1 \ldots 100]$.
3. Once the 10 random integer numbers are picked, P_1 will write the numbers in a file called `child1.txt` where each number will be in a separate line in the file.
4. P_2 will write the integer numbers it picked in a file called `child2.txt` where each number will be in a separate line in the file.
5. Once file `child1.txt` is ready, P_1 will inform the parent process that its file is ready to be processed.
6. Once file `child2.txt` is ready, P_2 will inform the parent process that its file is ready to be processed.
7. Upon being informed by P_1 and P_2, the parent process will inform the referee process R to decide on the winner. It does that by sending a message to R that contains the files `child1.txt` and `child2.txt` separated by a dash character (e.g. `child1.txt-child2.txt`).
8. The referee process R opens the 2 files and compares the integer numbers line by line. If a number picked by P_1 in a particular line is higher than the respective number picked by P_2, score1 is incremented. If a number picked by P_2 in a particular line is higher than the respective number picked by P_1, score2 is incremented. Otherwise, score1 and score2 remain unchanged. The above behavior is done for every line in the 2 files. Afterwards, R process deletes the files `child1.txt` and `child2.txt`.
9. R process will send to the parent process the scores of P_1 and P_2 processes separated by a dash (e.g. `5-3`).
10. The parent process parses the received message from process R and increments global counters associated with P_1 and P_2 respectively using the score values. We'll call the global counters of P_1 and P_2 bigScore1 and bigScore2 respectively. This ends the current round.
11. Go to step **2.** above for a new round unless either bigScore1 or bigScore2 counters has reached 50.

12. The parent process declares the winner of the game and the number of needed rounds then kills P_1, P_2, R and then exits. If both counters bigScore1 and bigScore2 reach 50 at the same time, then both P_1 and P_2 are winners of the game.

What you should do

- Write the code for the parent process in addition to processes P_1, P_2 and R.
- Compile and test your program.
- 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`.
- Send the zipped folder that contains your source code and your executable(s) before the deadline. If the deadline is reached and you are still having problems with your code, just send it as is!