Birzeit University - Faculty of Engineering and Technology
Electrical & Computer Engineering Department - ENCS4330
Real-Time Applications & Embedded Systems - 2^{nd} semester - 2021/22

Project #2
Interprocess communication techniques under Linux
Due: May 15, 2022

Instructor: Dr. Hanna Bullata

Fun Game Simulation

We would like to create a multi-processing application that simulates a fun game between 2 teams, each composed of 4 players. We'll call the players of team 1 P_{11}, P_{12}, P_{13} and P_{14} consecutively. We'll also call the players of team 2 P_{21}, P_{22}, P_{23} and P_{24} consecutively. A parent process is responsible to create the players of team 1 & team 2 and prepare the environment for them to behave as expected.

The behavior of the whole system should be as follows:

1. The first player of team 1 (P_{11}) is supposed to move water from an intially filled huge tank located at position `A` to a huge tank initially empty located at position `B` using a water bag placed on its back. While running from location `A` to `B`, some of the water might drop off the bag. In addition, P_{11} might fall accidentaly and thus lose the portion of water in its bag.

   After spilling the water in a tank located at position `B`, player P_{11} goes back to location `A` to bring more water until the huge tank at `A` becomes empty or the round time is over.

2. Much as the first player of team 1, the first player of team 2 (P_{21}) is supposed to move water from a huge tank located at position `C` to another huge tank located at position `D` using a water bag placed on its back. While running from location `C` to `D`, some of the water might drop off the bag. In addition, P_{21} might fall accidentaly and thus lose the portion of water in its bag.

   After spilling the water in a tank located at position `D`, player P_{21} goes back to location `C` to bring more water until the huge tank at `D` becomes empty or the round time is over.

3. While running from location `A` to `B`, player P_{11} will be annoyed by 2 players of team 2 P_{22} and P_{23}. These 2 players have initially empty bags on their backs and will try with a cup of water at their hand to steal the water from P_{11}'s bag and bring the stolen water back into the huge tank at location `A`. They will do that as long as player P_{11} did not empty its bag into the tank and location `B`. Once player P_{11} pours the water in its bag into the tank at location `B`, players P_{22} and P_{23} need to wait until P_{11} fills its bag again at location `A`.

   Note that only 1 player will be able to steal water from player P_{11}'s bag at a time.

4. The same behavior in step **3** is mirrored to player P_{21} of team 2. Just replace P_{11} by P_{21} and players P_{22} and P_{23} by P_{12} and P_{13}. Replace also location `A` by `C` and location `B` by `D`.

5. To help player P_{11} of team 1, player P_{14} of team 1 will try to annoy the 2 players P_{22} and P_{23} of team 2 by putting sand in their back bag and thus slowing them down so they don't steal much water from its team member P_{11}. The more sand it puts in their back bags, the less water they will be able to steal. Player P_{14} can put sand in one's player back bag at a time, not both at the same time. While putting sand in the back bag of a player, the other player will try to push P_{14} so it falls on the ground and thus fail to do so.

6. Step **5** above is mirrored to team 2. Just replace players P_{11} by P_{21} and players P_{22} and P_{23} by P_{12} and P_{13} consecutively. Finally, replace P_{14} by P_{24}.

7. Once the round time is over or one of the huge tanks is empty, a referee measures the amount of water in tanks at locations `B` and `D`. If more water is present in tank at location `B` than `D`, team 1 is declared the winner of the round. Otherwise, team 2 is declared the winner of the round.

8. The above behavior goes on until one of the teams has won *3 rounds in a row*. The winner is declared the winner of the game.

What you should do

- Write the code for the above-described game using multi-processing approach.
- Use any IPC technique(s) that suits your needs.
- Do your best to have a good looking output on the screen.
- Compile and test your program.
- Check that your program is bug-free and runs as expected. 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`.
- In order to avoid hard-coding values in your programs, think of creating a text file that contains all the values that should be user-defined and give the file name as an argument to the main program. That will spare you from having to change your code permanently and re-compile.
- Send the zipped folder that contains your source code and your executable(s) before the deadline as a reply to my message entitled **"Project 2: IPC"**. If the deadline is reached and you are still having problems with your code, just send it as is!