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

Project #2
Interprocess communication techniques under Linux
Due: May 28, **2023**

Instructor: Dr. Hanna Bullata

Message encoding/decoding

We would like to help militant groups into sending encoded messages that only the right receiver is able to decode using a multi-processing approach. We'll assume that if the receiver is able to get the encoded messages before a master spy process and is able to decode them correctly, then the operation is successful. Otherwise, if a master spy process is able to get to the messages and decode them first, then the operation has failed.

The application scenario can be explained as follows:

- A parent process must create a single sender process, single receiver process, a single master spy process, a user-defined number of helper processes and a user-defined number of spy processes.
- A sender process gets the message from an input file (e.g. `sender.txt`). The message might be composed of multiple lines and multiple paragraphs. The sender should split the input file by column based on the blank character between words.
- The sender process should create as many children processes as needed depending on the number of columns in the file. To each child process, a column message must be sent containing all the words in that column. The columns should be of equal size. As such, if a column contains an empty string in any line, it should be replaced by the string "`alright`".
- The children processes should each encode the column message that is passed to it before placing the encoded message in the shared memory. Each encoding child process acts as follows:
  - For the first column, for each word in the string, we add 1 to the first character modulo 26, 2 to the second character modulo 26, etc. The child process responsible for encoding that string will place it in the first location in the shared memory.
  - For the second column, for each word in the string, we add 2 to the first character modulo 26, 4 to the second character modulo 26, etc. The child process responsible for encoding that string will place it in the second location in the shared memory.
  - Same logic applies to the consecutive columns and words of each column.
  - Special characters must be encoded as follows:

∗ ! → 1	? → 2	, → 3
∗ ; → 4	: → 5	% → 6

  - Numbers are encoded as 1,000,000 - number.
  - Each encoded column must have a prefix or suffix added to it so that the receiver process is able to identify the column number correctly.
- Helper processes will continuously swap the messages that are present in the shared memory to make it hard for spy processes to get all the columns of the file. For example, a particular helper process might at some point swap between the encoded messages in locations 3 & 10 of the shared memory.

- Spy processes will continuously access the shared memory locations randomly to get the encoded messages before sending them to the master spy process.
- The master spy process tries to order the columns in the right order after getting them from the spy processes. It will drop columns it already received. When the master spy is confident it got all the columns, it tries to decode the messages in a file called `spy.txt` before informing the parent process.
- The receiver process will continuously access the shared memory locations randomly to get the encoded messages. Similar to the master spy process, it will order the columns it gets in the right order and drops the columns it already received. When it is confident it got all the columns, it tries to decode the messages in a file called `receiver.txt` before informing the parent process.
- The parent process decides if the receiver process was able to get the correct file from the sender before the master spy process. If true, then the operation is labeled a successful operation. Otherwise, it is labeled as a failed operation.
- The simulation ends if any of the following is true:
  - The number of failed decoding operations by the receiver exceeds a user-defined threshold.
  - The number of successful decoding operations by the receiver exceeds a user-defined threshold.

What you should do

- Write the code for the above-described application using a multi-processing approach.
- 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`.
- 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.
- Use graphics elements from opengl library in order to best illustrate the application. Nothing fancy, just simple and elegant elements are enough.
- Test your program.
- 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!