Birzeit University - Faculty of Engineering and Technology
Electrical & Computer Engineering Department - ENCS4330
Real-Time Applications & Embedded Systems - $1^{st}$ semester - 2023/2024

**Project #1**
**Interprocess communication techniques under Linux**
**Due: December 10, 2023**

**Instructor:** Dr. Hanna Bullata

## Supermarket cashier simulation

We would like to create a multi-processing application that simulates the behavior of cashiers while scanning customers' purchased items in a big supermarket. The application can be explained as follows:

- Customers arrive randomly at the supermarket. The rate of arrival is random but must belong to a user-defined range.
- Customers choose random items and random quantities of each item. Assume the supermarket has a list of user-defined items and each item has a user-defined price. Of course, when an item becomes out of stock, it can't be picked by customers.
- Customers spend a random amount of time shopping at the supermarket. The shopping time must belong to a user-defined range.
- When customers are done shopping, they usually choose the cashier to pay for purchased items depending on the following factors:
  - The cashier's queue size.
  - The total number of items in all shopping carts for all customers of the queue.
  - How quickly each cashier scans the customers' purchased items.
  - The cashier's behavior with customers (kindness, smile, etc).
- Each cashier spends a random amount of time to scan each purchased item. The time range is user-defined.
- Cashiers' positive behavior (kindness, smile, etc) drops with time. Assume that when the positive behavior drops to 0 for a particular cashier, that cashier must leave the simulation and customers at his/her queue must look for a different queue to pay for purchased items. However, if the cashier whose positive behavior got to 0 is already handling a customer, it continues with that customer before leaving.
- When customers pay the cashier for the purchased items, they leave the supermarket.
- When customers wait too long at the cashier's queue to pay, they might get impatient. If they wait more that a user-defined time, they might decide to leave the supermarket without buying any item.
- The simulation ends if any of the following is true:
  - The number of cashiers whose positive behavior dropped to 0 and left the simulation is above a user-defined threshold.
  - The number of customers that became impatient and left the supermarket without buying is above a user-defined threshold.
  - One of the cashiers made income to the supermarket more that a user-defined threshold.

## What you should do

- In order to implement the above-described application, you can choose any combination of IPC techniques (message queues, semaphores, shared memory) in addition to pipes, fifos & signals. Of course you don't need to use all of them. Just pick the one(s) that serve the purpose of the application.
- Write the code for the above-described application using a multi-processing approach.
- In order to avoid hard-coding values in your programs, think of creating a text file that contains all the values and ranges 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.
- Think of creating a text file that contains the list of items that the supermarket sells. Include in that file as well the quantity per item, price per item, etc.
- 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.
- 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 before the deadline. If the deadline is reached and you are still having problems with your code, just send it as is!