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

**Project #2**
**POSIX threads under Unix/Linux**
**Due: January 12, 2024**

**Instructor:** Dr. Hanna Bullata

## Supermarket Product Shelving Simulation

We would like to build a combined multi-processing and multi-threading application that simulates the behavior of a supermarket employees who are responsible to place products on shelves. The system behaves as follows:

- Assume the supermarket sells $p$ products. The value of $p$ is user-defined. For each product $p_i$, a certain amount $am_i$ is placed on the shelves and the remaining amount is kept in a storage area. The value of $am_i$ is also user-defined.
- Assume the supermarket employs a user-defined number of shelving teams. Each team is composed of a team manager and a user-defined number of employees.
- When the amount of a particular product on the shelves drops below a user-defined threshold, one of the team managers becomes responsible to get from the supermarket storage area the necessary amount from that product to put on the shelves and places the product on a rolling cart. The selection of the team manager is random.

  Once the cart is filled and driven to the correct location, the team employees will place the product items on the shelves.
- Shelving teams can place only one product item at a time. As such, they can't be involved in shelving of multiple product items at the same time.
- 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. Of course, when an item becomes out of stock, it can't be picked by customers.
- The simulation ends if any of the following is true:
  - The storage area is out of stock.
  - The simulation has been running for more than a user-defined amount of time (in minutes).

## What you should do

- Implement the above problem on your Linux machines using a combined multi-processing and multi-threading approach. The customers are processes and the shelving teams are processes. Each shelving team manager and team employees are threads that belong to a particular shelving process.
- 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`.
- 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.
- Be reaslistic in the choices that you make!
- 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!