Birzeit University - Faculty of Engineering and Technology
Electrical & Computer Engineering Department - ENCS4330
Real-Time Applications & Embedded Systems - 1st semester - 2022/23

Project #3
POSIX threads under Unix/Linux
Due: January 16, 2023

Instructor: Dr. Hanna Bullata

Chocolate-manufacturing factory simulation

We would like to build a multi-threading application that simulates the behavior of a chocolate-manufacturing factory. The system behaves as follows:

- The factory produces 3 different types of chocolate. We'll call them chocolate `typeA`, `typeB` and `typeC`. The factory has also 7 manufacturing lines to produce the chocolate products, 3 of them for `typeA` chocolate, 2 for `typeB` chocolate and 2 for `typeC` chocolate.
- To produce `typeA` chocolate, we have 8 employees per manufacturing line each executing a step in the chocolate-manufacturing process. Each step might take a certain amount of time but is bound in a range between a $\min_A$ value and a $\max_A$ value. Steps 1 to 4 have to happen in order. Steps 4 to 8 can happen in any order.
- To produce `typeB` chocolate, we have 6 employees per manufacturing line each executing a step in the chocolate-manufacturing process. Each step might take a certain amount of time but is bound in a range between a $\min_B$ value and a $\max_B$ value. Steps 1 to 6 have to happen in order.
- To produce `typeC` chocolate, we have 5 employees per manufacturing line each executing a step in the chocolate-manufacturing process. Each step might take a certain amount of time but is bound in a range between a $\min_C$ value and a $\max_C$ value. Steps 1 to 3 have to happen in order. Steps 4 and 5 can happen in any order.
- The chocolate products that are produced by all the manufacturing lines are collected in patches of 10 pieces per type by 2 employees and sent to manufacturing line 8 that is responsible to print the expiration date on the chocolate cover. Assume printing the expiration date takes the same amount of time for all chocolate types.
- Once the printing is over, the chocolate products are separated in different huge containers according to the type by 3 employees. 3 other employees are responsible to prepare and fill the carton boxes that contain the chocolate products by collecting from the containers. Assume each carton must contain 20 chocolate products.
- Once carton boxes are ready, 2 storage employees are responsible to collect the filled carton boxes and place them in the storage area. The storage employees will become absent for a certain user-defined period of time while taking a particular box to the storage area.
- The storage area can contain a maximum number of manufactured chocolate carton boxes (user-defined). If the storage area capacity goes beyond the maximum threshold, the manufacturing lines must be suspended until the storage area capacity goes below a minimum threshold value (user-defined).
- The factory employs as well 2 loading employees whose job is to load trucks with the merchandise. Assume each truck can hold a user-defined number of `typeA`, `typeB`

and `typeC` products in each trip. Of course, each truck becomes unavailable for a user-defined period when shipping the chocolate cartons. Keep in mind that trucks are filled in order (meaning we don't switch to filling another truck until the current one has reached the maximum capacity which is user-defined). Assume as well we have 3 shipping trucks.

- The simulation should end if any of the following is true:
  - The factory has produced `typeA`, `typeB` and `typeC` carton boxes that exceed a user-defined threshold for each type.
  - 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 multi-threading approach.
- 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!