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

Project #3
POSIX threads under Unix/Linux
Due: June 3, 2022

Instructor: Dr. Hanna Bullata

Detergent-manufacturing factory simulation

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

- The detergent factory has 5 independant manufacturing lines. Each line is composed of 10 technical employees each executing in sequence a step in the detergent manufacturing process. The manufacturing steps have to happen in order so as a technical employee cannot carry out its work before the employee that precedes it has completed its work. Of course, the technical employees do not all work at the same speed.

- The produced detergent is composed of multiple raw materials that are supplied by two other factories. We'll call the supplier factories `factory_A` and `factory_B`.

- `factory_A` has `A` trucks that can ship the first raw product needed by the detergent factory. Once a truck is fully loaded, it moves in the direction of the detergent factory. Assume the capacity of each truck is `CA`.

- `factory_B` has `B` trucks that ship the second raw product needed by the detergent factory. Once a truck is fully loaded, it moves in the direction of the detergent factory. Assume the capacity of each truck is `CB`.

- Once a truck carrying raw materials (either `A` or `B`) reaches the detergent factory, it is unloaded by `L_1` employees and is placed in a storage area `S_1`.

- `L_2` employees are responsible to carry the boxes of raw material `A` and `B` from the storage area `S_1` to the 5 manufacturing lines.

- The first technical employee of each manufacturing line takes 2 pieces of product `A` and 1 piece of product `B`, does some processing as part of its detergent manufacturing step.

- Once the above 10 steps are executed, the detergent manufacturing process is over and the technical employee that executes the last step should put the detergent in a carton box placed at the end of the line.

- Once `N` detergent products are placed in a carton box, a storage employee is responsible to collect the filled carton box and place it in storage room `S_2`. The storage employee will become absent for a certain user-defined period of time until he finishes his task. Assume the detergent factory has `L_3` storage employees.

- Since raw material `A` is being consumed at a quicker pace than raw material `B`, the detergent factory will stop ordering product `B` once the ratio product `B` to product `A` exceeds 2. The detergent factory will resume ordering product `B` once the ratio product `B` to product `A` gets less than 1.

- The storage area `S_2` can contain a maximum number of manufactured detergents (user-defined). If the storage room capacity goes beyond the maximum threshold, the manufacturing lines should stop manufacturing more detergents until the storage room capacity goes below a minimum threshold value (user-defined).
- The detergent factory employs as well several loading employees whose job is to load trucks that belong to the detergent factory with the detergent products cartons in the storage room. Assume each truck can hold a user-defined number of cartons in each trip. Of course, each truck becomes unavailable for a user-defined period when shipping the detergent products.
- The simulation ends if any of the following is true:
  - The factory has manufactured a number of detergent products that exceeds a user-defined threshold.
  - The simulation has been running for more than a user-defined number of 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.
- 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!