Birzeit University - Faculty of Engineering and Technology
Electrical & Computer Engineering Department - ENCS4330
Real-Time Applications & Embedded Systems - 2^{nd} semester - 2024/2025

Project #2
Interprocess communication techniques under Linux
Due: May 2, 2025

Instructor: Dr. Hanna Bullata

Bakery Management Simulation

We would like to build a multi-processing application that simulates the behavior of the daily operation of a bakery and that takes advantage of the IPC techniques. The simulation can be explained as follows:

- The bakery produces a user-defined number of bread categories in addition to a user-defined number of sandwiches, user-defined different cake flavors, sweets flavors, sweet and savory patisseries, etc.
- The bakery employs a user-defined number of different chefs and a user-defined number of bakers in addition to a user-defined number of sellers to deal with customers and a user-defined number of supply-chain employees.
- The chefs belong to different teams. Some of these teams are as follows:
  - Teams that prepare the paste,
  - Teams that prepare the cakes,
  - Teams that prepare the sandwiches,
  - Teams that prepare the sweets,
  - Teams that prepare sweet patisseries,
  - Teams that prepare savory patisseries.

  The sweet and savory patisseries chefs depend on the teams that prepare the paste. The teams that prepare the sandwiches depend on the teams that prepare the bread. The other teams are independent as long as the items they need to do their work are available.
- The bakers belong to different teams as well. They take the items produced by the chefs and take care of putting them in ovens and making sure to take them out of the ovens at the right time. The following teams exist for the bakers:
  - Teams that bake cakes and sweets,
  - Teams that bake sweet and savory patisseries,
  - Teams that bake bread.
- The supply-chain employees are responsible for the purchase of the following items:
  - Wheat,
  - Yeast,
  - Butter,
  - Milk,
  - Sugar and salt,

  - Sweet items,
  - Cheese and salami for the sandwiches.

  The purchased quantities belong to user-defined ranges. If any of the above items is missing, the baker teams that depend on these items will stop working.

- The bakery will adjust its daily operations based on customer requested items and available purchased items. For example, if there is a shortage in sweet patisseries relatively to savory patisseries, the bakery management might decide to move chefs around to boost the sweet patisseries production (e.g. move chefs from the savory patisseries teams to the sweet patisseries teams).

  In addition to the above, if the bakery is out of cheese and/or salami, it will stop producing sandwiches!

- Items the customers purchase have user-defined prices. Thus each time an item is sold to customers, the bakery profit increases.

- It happens that customers might not be satisfied with the quality of the items they purchased from the bakery. Once a customer complains, the bakery policy is always to reimburse the unhappy customer without asking questions. The bad side of things is that customers present at the bakery at a time when a customer complains might change their minds and leave the bakery without purchasing anything.

  In addition, customers who wait too long before being handled by the bakery sellers might get frustrated and leave the bakery without purchasing anything.

- The simulation ends if any of the following is true:
  - The number of frustrated customers exceeds a user-defined threshold.
  - The number of customers that complained exceeds a user-defined threshold.
  - The number of customers that requested missing items exceeds a user-defined threshold.
  - The bakery has made a daily profit that exceeds a user-defined threshold.
  - 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-processing approach. Make sure the created processes consume CPU time when needed only.

- 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 user-defined 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!