Birzeit University - Faculty of Engineering and Technology
Electrical & Computer Engineering Department - ENCS4330
Real-Time Applications & Embedded Systems - 1^{st} semester - 2022/2023

Project #2
Interprocess communication techniques under Linux
Due: December 23, 2023

Instructor: Dr. Hanna Bullata

OIM simulation

We would like to create a multi-processing application that simulates the daily operation of the occupation interior ministry offices (OIM for short) at Sheikh Jarrah neighborhood that handle issues relating to Palestinian citizens living in Jerusalem. That will give you the opportunity to see the pain Palestinians living in Jerusalem have to go through when they need to see issued some official documents such as birth certificates, travel documents, family reunion documents, ID-related problems, etc. I will remove some extra real painful steps to make it easy on you to simulate the application.

The application scenario can be explained as follows:

- People start arriving randomly at the OIM rolling gates early in the morning (e.g. 5am) and keep on piling up at the gates until it is 8am (gate opening). There are 2 rolling gates, 1 for males and 1 more females. When at 8am the rolling gates are enabled to have people enter, the order of entering for both men and women is unfortuanely not the order of arrival. Assume though that the more time you spend at the gates, the bigger is your chances to get in. In addition, some people might decide just to leave unserved (random in nature).

  People continue to arrive randomly at the OIM offices after 8am up to 1pm.

- Inside the gates, there are obviously 2 queues: 1 for males and 1 for females. When the 2 queues are filled beyond a certain threshold, the rolling gates are disabled until the number of people in the queues drop below a certain threshold. Keep in mind that the rolling gates for men and women are independent so they don't need to be enabled or disabled simultaneously.

- An occupation security officer signals randomly to the head of one of the 2 queues to advance and go through a metal detector machine (body check). Picking the head of the male or female queue is random. People at the metal detector machines spend some random amount of time before they can proceed to the inner grouping area.

- Once inside the inner grouping area, people get queuing tickets according to the purpose of their requests. The tickets are labeled `Bx` for birth certificate requests, `Tx` for travel document requests, `Rx` for family reunion requests and `Ix` for ID-related requests. The letter `x` is a serial number.

- Assume that some of the tellers are dedicated to handle birth certificate requests only, some are dedicated to handle travel document requests only, some are dedicated to handle family reunion requests only and the remaining ones handle ID-related requests. However, if no requests are present for a certain category, tellers can handle requests for other categories as well.

- When requests are handled by tellers, these tellers become busy for a certain period of time. Requests can get served or labeled incomplete for missing documents (as usual :-). As such, people can leave OIM either satisfied or unhappy depending on how their requests were handled.

- While waiting for their requests to be handled, some people get impatient and leave the inner grouping area unserved. The patience level is usually random in nature and different between humans.
- The simulation ends if any of the following is true:
  - The number of people that left OIM offices unserved is above a user-defined threshold.
  - The number of people that left OIM offices unhappy is above a user-defined threshold.
  - The number of people that left OIM offices satisfied is above a user-defined threshold.

What you should do

- Write the code for the above-described application using a multi-processing approach.
- 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.
- Test your program.
- 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!