Birzeit University - Faculty of Engineering and Technology
Electrical & Computer Engineering Department - ENCS4330
Real-Time Applications & Embedded Systems - 1st semester - 2021/22

Project #3

POSIX threads under Unix/Linux

Due: November 23, 2021

Instructor: Dr. Hanna Bullata

Laptop-manufacturing factory simulation

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

- There are 10 manufacturing lines inside the factory. Each line is composed of 10 technical employees each executing a step in the laptop manufacturing process. Each step might take a certain amount of time but is bound in a range between a min value and a max value.
- Steps 1 to 5 have to happen in order. Steps 6 to 10 can happen in any order.
- Once the above 10 steps are executed, the laptop manufacturing process is over and the technical employee that executed the last step should put the laptop in a carton box placed at the end of the line.
- Once 10 laptops are placed in a carton box, a storage employee is responsible to collect the filled carton box and place it in the storage room. The storage employee will become absent for a certain user-defined period of time until he finishes his task.
- The storage area can contain a maximum number of manufactured laptops (user-defined). If the storage room capacity goes beyond the maximum threshold, the technical employees should stop manufacturing more laptops until the storage room capacity goes below a minimum threshold value (user-defined).
- The factory employs as well several loading employees whose job is to load trucks with the laptop 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 laptops.
- The factory has a CEO with salary $\texttt{salary}_{\texttt{ceo}}$ and an HR manager with salary $\texttt{salary}_{\texttt{hr}}$. Assume also all technical employees have the same salary $\texttt{salary}_{\texttt{t}}$, all storage employees have the same salary $\texttt{salary}_{\texttt{s}}$, all loading employees have the same salary $\texttt{salary}_{\texttt{l}}$ and all truck drivers have the same salary $\texttt{salary}_{\texttt{u}}$. Assume as well that any extra employee you might need to hire to do the simulation has a salary $\texttt{salary}_{\texttt{a}}$ (e.g. storage room keeper, security personnel, etc). Thus, the only expense the factory has is the employees' salaries.
- Assume also that each laptop's cost is $\texttt{cost}_{\texttt{fab}}$ and that the laptops are sold each at $\texttt{price}_{\texttt{sell}}$.
- If the factory profit (what the factory gains minus all expenses) drops below a user-defined threshold, the HR manager informs the CEO to cut on expenses by suspending some employees. The CEO will suspend the employees belonging to a whole manufacturing line in 1 shot. If the factory gains increase beyond a certain threshold afterwards, the HR manager informs the CEO to cancel the employees' suspension. Obviously, if the factory gains decrease again, the HR manager might ask the CEO to expand the suspension plan to include more manufacturing lines.

- The simulation should end if any of the following is true:
  - The factory has made gains that exceeds a user-defined threshold.
  - The CEO had to suspend more than 50% of the employees.

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!