Athens University of Economics and Business
Department: Computer Science
Course: Operating Systems
Academic Year: 2019-2020
Instructor: Giorgos Xilomenos

2nd Project

Goal: In both projects you will implement a system of pizza order and delivery with the use of POSIX threads package (pthreads). In this second assignment there have been added deliverers who deliver the orders to the clients. The way the cooks perform is also changed. Your code must work properly on the virtual machine that was shown in the labs (link can be found in eclass). Both of these projects are designed for groups of three (3) people, but they can be completed in smaller groups too. The grade of this 2nd task represents 10% of the final grade (i.e. 1 points out of 10).

Objective: The pizzeria has N_{cook} cooks, N_{oven} ovens and $N_{\text{deliverer}}$ r deliverers. The first order is made at the time 0, and every next order is made after a random period of time and it ranges in the interval $[\text{T}_{\text{orderlow}}, \text{T}_{\text{orderhigh}}]$. Every order includes a random integer number of pizzasin the interval $[\text{N}_{\text{orderlow}}, \text{N}_{\text{orderhigh}}]$. The order has to wait until a cook is available. When a cook is available, a T_{prep} amount of time is required for each pizza to be prepared for the oven.Then, the cook waits until an oven is available. When an oven is available, all pizzas of the particular order go into the same oven and bake for T_{bake} amount of time. Note that the cook is available for a next order right after placing the pizzas in the oven (he does not wait for the pizzas to be baked). When the pizzas are baked, the oven turns automatically off and waits for a deliverer. When a deliverer is available, he takes all pizzas of the particular order out of the oven (which is then available for another order). He then packs the order and delivers it to the client. The delivery lasts a random integer period of time in the interval $[\text{T}_{\text{low}}, \text{T}_{\text{high}}]$. After the delivery, the deliverer needs the same amount of time to return to the pizzeria and to take on the next order. Every cook deals with only one order each time from the moment the order is handed to him/her until it is placed inside the oven. Every deliverer delivers only one order each time he goes out of the pizzeria.

Input and data: The following constants will be defined in a header file.

- N_{cook} = 2 cooks (1/3 in comparison to the 1st assignment)
- N_{oven} = 5 ovens
- $N_{\text{deliverer}}$ = 10 deliverers
- $\text{T}_{\text{orderlow}}$ = 1 minute
- $\text{T}_{\text{orderhigh}}$ = 5 minutes

- N_{orderlow} = 1 pizza
- $N_{\text{orderhigh}}$ = 5 pizzas
- T_{prep} = 1 minute
- T_{bake} = 10 minutes
- T_{low} = 5 minutes
- T_{high} = 15 minutes

Your program will take two (exactly) parameters, which will represent the number of clients (N_{cust}) and random seed for random number generator.

Project Output: For each order, when it is received by the client, the following message must be printed:

- Order with number <oid> has been delivered in <X> minutes and was cold for <Y> minutes.

The order of the lines will be random, but the lines should not be mixed. The time <X> represents the period from the entry of the order until its delivery to the client, while time <Y> represents the period from completion of baking until its delivery to the client.
At the end of the program execution, the system will print the following:

- Average and maximum delivery time
- Average and maximum time of orders being cold.

Code Structure: The initial thread of your program will create one thread per order (total of Ncust threads) to which you will pass a thread number (from 1 to Ncust) so that each thread is uniquely identified. Each thread will then perform the steps mentioned above until the order is delivered and will print the appropriate output. The initial thread will print the final output. You will need at least the following:

- An integer variable and a mutex to count the number of available cooks and a condition variable to synchronize orders with cooks so that when no cooks are available, the orders are blocked. You will handle the ovens and distributors in a similar way.
- Double variables and related mutexes for **delivery** and **cold** times.
- A mutex to lock the screen when you print the output.

Pay attention to the correct termination of the threads, to wait for them when needed and (mainly!) to properly free memory that was dynamically allocated.

Hints:

- All hints of the 1st assignment should also be followed in this 2nd assignment. It is recommended to start with the code of the previous assignment and extend it.
- Be aware that cooks are released as soon as an oven is available and that the ovens are released as soon as at least 1 deliverer is available.
- Also pay attention to the fact that the deliverers are busy even after the order has been delivered to the client, as they must return to the pizzeria.

Additional Notes: Your code must consist of a header file (including constants) and a .c code file for the program. These files must be named **pizza2.h** and **pizza2.c**. You should also write a **report** that describes the structure of your code and indicates any restrictions or additional features that you have implemented. The report must be named **project2-report**.pdf. Finally, you must include a file **test-res2.sh** which will compile and execute your program with 2 parameters, 100 (for the clients) and 1000 (for the initial seed).