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

1st 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 system orders are entered, each order is prepared and when the preparation is done it is delivered to the customer. In the first task you will build a simplified system, which in the second task will become more complex. In these systems we have a large number of orders which are served by a limited service points, so your program must implement mutual exclusion (with mutexes) and synchronization (with condition variables). 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 the 1st task represents 20% of the final grade (i.e. 2 points out of 10).

Objective: The pizzeria has N_{cook} cooks and N_{oven} ovens. In the 1st assignment, the pizzeria does not have a delivery service, i.e. customers can only take their orders out. 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 $[T_{orderlow}, T_{orderhigh}]$. Every order includes a random integer number of pizzas in the interval $[N_{orderlow}, N_{orderhigh}]$. 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, while the cook is busy watching this oven. When the baking is over, the cook puts the completed order on the delivery bench and is available for the preparation of another order. The cook deals with only one order each time from the moment the order is handed to him/her until it is placed on the delivery bench.

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

- N_{cook} = 6 cooks
- N_{oven} = 5 ovens
- $T_{orderlow}$ = 1 minute
- $T_{orderhigh}$ = 5 minutes
- $N_{orderlow}$ = 1 pizza
- $N_{orderhigh}$ = 5 pizzas
- T_{prep} = 1 minute

- T_{bake} = 10 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 prepared in <X> 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 completion. After all orders are received by the clients, the system will print the following message:

- Average and maximum time for order completion.

Code Structure: The initial thread of your program will create one thread per order (total of N_{cust} threads) to which you will pass a thread number (from 1 to N_{cust}) so that each thread is uniquely identified. Each thread will then perform the steps mentioned above until the order is completed 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.
- An integer variable and a mutex to count the number of available ovens and a condition variable to synchronize orders with ovens, so that when no ovens are available, orders are blocked.
- Double variables and related mutexes for completion 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 where needed and (mainly!) to properly free memory that was dynamically allocated.

Hints:

- In order for your program to be completed in a reasonable time, act like seconds are minutes. For example if the print message of an order's completion is 29 minutes, in reality it will be 29 seconds.
- When compiling you must enter the *-pthread* option to use POSIX threads library.
- To simulate the time it takes for baking etc., you will use *unsigned int sleep(unsigned int seconds).* With this function you can also create orders from the initial thread.
- To generate a series of pseudo-random numbers, you will use *int rand_r(unsigned int *seedp).* By giving different numbers to the seed, you will have different execution sequences. Use the % operator to limit the range of values of random numbers.
- To calculate the waiting times, use *int clock_gettime(clockid_t clk_id, const struct timespec *tp)* at the beginning and at the end of each operation, with the constant *CLOCK_REALTIME* as the first parameter. At the output, convert the resulting times from seconds to minutes.

- Use a while loop to check the standby condition and call *pthread_cond_wait* as many times as needed.
- Note that the cooks are busy until the end of the baking of the particular order.

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 **pizza1.h** and **pizza1.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 **project1-report.pdf**. Finally, you must include a file named **test-res1.sh** which will compile and execute your program with 2 parameters, 100 (for the clients) and 1000 (for the initial seed).