OPERATING SYSTEMS

Programming Assignment

Purpose: In this project you will implement an ordering and distribution system for pizza with using the POSIX thread package POSIX threads (pthreads). In this system, orders are placed and paid for electronically, and then each order is prepared, baked and finally distributed to the customer. In these systems we have a large number of orders which are serviced by a limited number of service points, so your schedule must implement mutual exclusion (with mutexes) and synchronization (with condition variables). Your code should work correctly on the virtual machine available in CSLAB and on the web (see the eclass wiki).

Object: The pizzeria only makes two types of pizza, plain and special. It has an online ordering and payment system, Ncook makers, Noven ovens, Npacker employees packing clerks and Ndeliverer distributors. The first customer logs into the ordering system on time 0, and each subsequent customer logs in after a random integer time interval in the range [Torderlow,Torderhigh]. When a customer connects to the ordering system, orders a random integer number of pizzas in the interval [Norderlow,Norderhigh]. Each of these has probability Pplain to be simple, otherwise it is special. The system requires a random integer time interval in the range [Tpaymentlow,Tpaymenthigh] to charge the credit card of the customer. With probability Pfail the charge fails and the order is cancelled, otherwise the order is registered, the store's revenue is increased by Cplain or Cspecial per pizza, depending on the pizzas in the order, and finally we update the number of sales per type of pizza. The registered order initially waits until a cook is available. When this happens, the preparer needs Tprep time to prepare each pizza. Then, the manufacturer waits until enough ovens become available so that all pizzas can be baked in parallel, because each oven can only hold one pizza. When enough ovens are available are available, each pizza is placed in one oven and the pizza maker takes over the next order. The pizzas are baked for Tbake time. When baking is complete, the ovens are closed automatically and wait for a packer to remove the pizzas from the ovens and pack them, which takes Tpack time per pizza. At this time and the ovens are released. When a Deliverer becomes available, it picks up a packaged pizza order and delivers it to the customer, with delivery taking a random integer amount of time interval in the region [Tdellow,Tdelhigh]. After delivery, the distributor needs exactly the same

time to return to the store and take the next order. **Warning**: each cook deals with only one order until he puts it in the oven, the ovens are released only when packing is finished, and each distributor carries only one order on each route.

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

- Ncook=2 cooks
- Noven=15 ovens
- Npacker=2 packers
- Ndeliverer=10 deliverers
- Torderlow=1 minute
- Torderhigh=3 minutes
- Norderlow=1 pizza
- Norderhigh=5 pizzas
- Pplain=60%
- Tpaymentlow=1 minute
- Tpaymenthigh=3 minutes
- Pfail=10%
- Cplain=10 euros
- Cspecial=12 euros
- Tprep=1 minute
- Tbake=10 minutes
- Tpack=1 minute
- Tdellow=5 minutes
- Tdelhigh=15 minutes

Your program will accept two (exactly) parameters with the number of clients to service, Ncust, and a random seed for the random number generator.

Output : For each order, at least the following information will be printed, at appropriate time, on the screen:
- The order with number <oid> failed/registered. [At the moment of failure or registration of the order.]
- Order number <oid> was prepared in <X> minutes. [At the moment when the order with the number number of items with order number #1515 is completed. packing of a successful order.]
- Order number <oid> was delivered in <Y> minutes. [At the moment when the delivery of a successful order.]

The order of the lines will be random, but the lines should not intertwine. To <X> represents the interval from the customer's appearance until the order is packed, while <Y> is the interval from the customer's appearance until the order is delivered. At the end of the execution, the system will print the following:

- The total sales revenue, how many pizzas were sold of each type and the number of successful and failed orders.

- The average and maximum customer service time (from the moment the customer is shown the customer, until the order is delivered) - only for successful orders.

- The average and maximum cooling time of orders (from the moment the order is completed baking to the moment the order is delivered) - only for successful orders.

Code structure: the initial thread of your program will create one thread per client/order (Ncust threads in total) to which you will pass a thread number (from 1 to Ncust) to use as the order number. Each thread will then execute the above steps until the order is completed and prints the appropriate output.
Finally, 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 the orders with the cooks so that when no cooks are available, the orders are blocked. In a similar way you should handle the ovens, the packaging employees and deliverers.
- Variables and associated mutex for revenue and statistics.
- A mutex for locking the screen when you print the output.

Hints:
- To complete your project in a reasonable amount of time, treat all times given as seconds instead of minutes.
- When compiling, you must give the -pthread option to use the POSIX threads library.
- To simulate the time that takes some interval (e.g. baking) you use the unsigned int sleep(unsigned int seconds).
- To generate an array of pseudorandom numbers, you will use the int rand_r(unsigned int *seedp). Use the % operator to restrict the the range of values of the random numbers.
- To calculate the waiting times, use int clock_gettime(clockid_t clk_id, const struct timespec *tp) at the beginning and end of each operation, with the CLOCK_REALTIME constant as the first parameter, convert the resulting times from seconds to minutes.
- Use a while loop to check the wait condition and do pthread_cond_wait for as many times as needed.
- Beware that we may need to bind multiple ovens for a order.
- Attention to when each resource (manufacturer, oven, packing clerk and distributor).
- Attention to terminating threads correctly, waiting for them where needed and (most importantly!) releasing correctly as much memory as you commit.

Deliverables: Your code must consist of a file of statements (including constants) and a C code file for the program. The name of the files must be “Pizzeria.h” for the statements, “Pizzeria.c” for the C code. Except the code, you must write a report describing the structure of your code and mention any restrictions or additional features you have implemented. The report should be a PDF file with a name in the format p3x-p3y-p3z-pizzeria.pdf. Finally, you should include a file named test-res.sh which compiles and execute your program with 100 clients and an initial seed of 1000.