CSE-344 FINAL ASSIGNMENT

Author: Hacı Hasan Savan Number: 1901042704

Algorithm Design and Workflow

Upon starting the server side of the Pide Shop simulation, it begins listening for incoming client connections. Once the client side is initiated, it sends an order struct containing meals and other relevant details to the server. The server's listener thread, acting as a manager, reads this order via sockets and enqueues it into the meals_queue, subsequently posting cook semaphores to signal the availability of meals to be prepared. Each meal read from the order triggers a semaphore post, ensuring one cook thread will process each meal. When a cook thread passes through the semaphore, it begins preparing the meal, starting a timer represented by the pseudoInverseMatrix calculation. During preparation, the cook checks for its meal in the oven (managed by an oven linked list). Approximately halfway through the preparation time, the cook attempts to interact with the oven for two purposes: putting a meal in or taking a meal out. If putting a meal in, the cook checks for available space in the oven. If no space is available, the cook waits without attempting to acquire the aperture and door semaphores, preventing potential deadlocks. The waiting cooks will only proceed when another cook, intending to take a meal out, frees up space. Cooks coming to take meals will only race with other cooks aiming to take meals, not those putting meals in, thus ensuring smooth and deadlock-free operations.

Synchronization Elements

Semaphores

Semaphores are synchronization tools used to control access to shared resources by multiple threads. They are particularly useful in managing concurrent processes in critical sections.

In the Pide Shop system, the following semaphores are used:

- meal_semaphore: Controls the availability of meals to be prepared by the cooks.
- oven_semaphore: Limits the number of meals that can be placed in the oven at any time.
- **oven_aperture_semaphore**: Manages access to the oven aperture, ensuring that only a limited number of cooks can use the aperture simultaneously.
- oven_door_semaphore: Controls access to the oven door, ensuring that only a limited number of cooks can use the door simultaneously.

Mutexes

Mutexes (mutual exclusion objects) are used to prevent race conditions by ensuring that only one thread can access a critical section at a time.

The following mutexes are used in the system:

- meals_mutex: Protects access to the meals queue.
- oven_mutex: Protects access to the oven list.
- delivery_mutex: Protects access to the delivery queue.
- delivery_mutex_for_bag: Ensures that only one delivery person can access their delivery bag at a time.
- cooked_meals_mutex: Protects the count of cooked meals.
- all_meals_mutex: Protects access to the linked list of all meals.

Condition Variables

Condition variables are used to block a thread until a particular condition is met. They are usually used with mutexes to avoid race conditions.

The following condition variables are used:

- **cook_cond**: Signals cooks when there are meals available to prepare.
- delivery_cond: Signals delivery persons when there are meals ready to be delivered.

Threads and Their Responsibilities

Client Threads

 Main Thread: Handles the generation and sending of orders, receiving updates, and handling signals for resource cleanup.

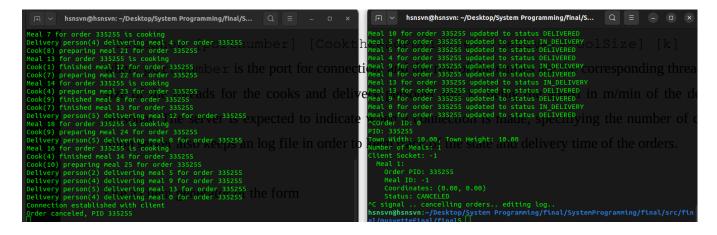
Server Threads

- **Listener Thread**: Represents the manager. Listens for and accepts client connections, reads orders, and handles cancellation orders.
- **Cook Threads**: Represents the cook. Take meals from the meals queue, prepare them, place them in the oven, and update their status.
- Delivery Threads: Represents the Delivery persons. Take meals from the delivery queue, simulate the delivery process, update meal statuses, and handle cancellations during delivery.

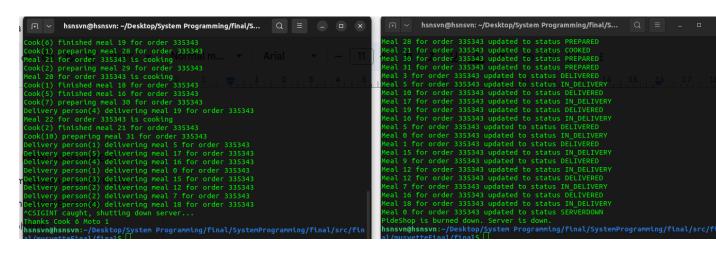
server and client will create separate log files.

Test görüntüleri

CTRL-C in client:



CTRL-C in server:

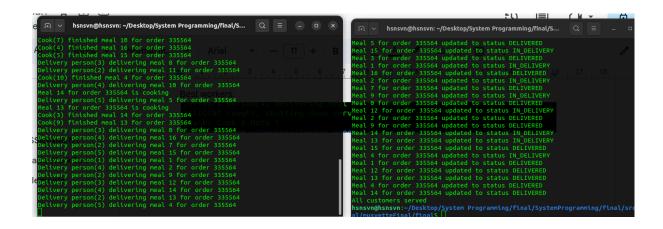


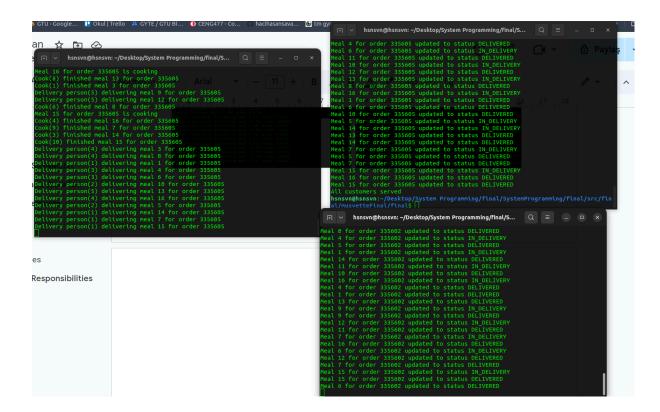
Cook will not wait while the previous is in the oven:

```
hsnsvn@hsnsvn: ~/Desktop/System Programming/final/S...
Cook(10) preparing meal 1 for order 335343 m 
ho 
ho 
ho
Cook(9) preparing meal 2 for order 335343
Cook(8) preparing meal 3 for order 335343
Cook(1) preparing meal-4 for order 335343 he DOIT TOT C
Cook(6) preparing meal 5 for order 335343
Cook(2) preparing meal 6 for order 335343
Cook(5) preparing meal 7 for order 335343 ne COOKS and
Cook(3) preparing meal 9 for order 335343
Cook(4) preparing meal 8 for order 335343
Meal 0 for order 335343 is cooking.
Cook(7) preparing meal 10 for order 335343 Dected to in
Meal 3 for order 335343 is cooking
Cook(8) preparing meal 11 for order 335343
Meal 5 for order 335343 is cooking
Cook(6) preparing meal 12 for order 335343
Meal 1 for order 335343 is cooking
Cook(10) preparing meal 13 for order 335343
Meal 9 for order 335343 is cooking
Cook(3) preparing meal 14 for order 335343
Meal 4 for order 335343 is cooking of of the form
Cook(1) preparing meal 15 for order 335343
Meal 7 for order 335343 is cooking
Cook(5) preparing meal 16 for order 335343
Cook(8) finished meal 3 for order 335343
```

Best workers:

```
^CSIGINT caught, shutting down server...
Thanks Cook 6 Moto 1
```





it will work in big numbers:

