RESTAURANT FUNCTIONING USING MULTI THREADING AND SYNCHRONIZATION

Advanced Operating Systems Project

Contents

Abstract

- The Main goal of the project is to master the use of multi programming and synchronization primitives. Functioning of restaurant includes many tasks which has to be done in an order so as to satisfy the customer.
- Some of the main tasks in restaurant is checking if the tables are available, assigning the tables to the customers in the order of arrival, availability of cooks, availability of items in the menu.
- To ensure all these tasks are performed without any conflicts the multithreading and synchronization comes into action.

Introduction

- A restaurant requires careful coordination of resources, which are the tables in the restaurant, the cooks available to cook the order, and the machines available to cook the food (we assume that there is no contention on other possible resources, like servers to take orders and serve the orders).
- Restaurants are dynamic environments where efficiency is key to delivering exceptional customer service.
- Efficiently managing customer arrivals, cook operations, and table availability is crucial for a seamless dining experience.
- Multithreading and Synchronization techniques can be used to optimize restaurant functions.

- Multithreading enables concurrent execution of tasks, while synchronization ensures safe access to shared resources.
- In this project, we consider the arrival time of customers, number of cooks available, number of tables available and items availability.
- The output can be obtained as at what time the table can be assigned to the customer, how much time will it take for the food to be prepared and when the customer leaves the table based on this, we can generate whether the next customer gets the table or not upon the arrival time.

- In this project, we have four sets of parameters - the number of eaters (diners) that wish to enter the restaurant, the times they approach the restaurant and what they order, the number of tables in the restaurant (there can only be as many eaters as there are tables at a time; other eaters must wait for a free table before placing their order), and the number of cooks in the kitchen that process orders.
- Eaters in the restaurant place their orders when they get seated. These orders are then handled by available cooks. Each cook handles one order at a time. A cook handles an order by using machines to cook the food items.

- We made the following assumptions. There are only three types of food served by the restaurant - Burger, Fries, and Coke. Each person entering the restaurant occupies one table and orders one or more burgers, zero or more orders of fries, and zero or one glass of coke.
- The cook needs to use the burger machine for 5 minutes to prepare each burger, fries machine for 3 minutes for one order of fries, and the soda machine for 1 minute to fill a glass with coke.
- The cook can use at most one of these three machines at any given time.
- Once the food (all items at the same time) are brought to a diner's table, they take exactly 30 minutes to finish eating them, and then leave the restaurant, making the table available for another diner (immediately).

Multi-threading

- Multithreading is a programming technique that allows a program to perform multiple tasks concurrently within a single process. It is a way to enhance the performance and responsiveness of an application by dividing work into smaller tasks that can be executed simultaneously

Synchronization

- Synchronization is a key aspect of multithreading that involves coordinating the execution of multiple threads to ensure correct access to shared resources and avoid data inconsistencies.
- Avoid Race Conditions: Occurs when multiple threads access shared data simultaneously, leading to unpredictable results.
- Data Integrity: Protects shared resources (e.g., variables, data structures) from being corrupted by concurrent access.
- Prevent Deadlocks: Ensures that threads do not get stuck waiting for each other indefinitely.

Implementation

- We create one thread for each arriving diner, which will then compete for an available table. There will also be one thread for each cook, all of them active for the entire duration when the restaurant is open.
- Tables and machines for cooking the food are resources coordinated among the threads.
- Output contains the following information - when each diner was seated, which table they were seated in, which cook processed their order, when each of the machines was used for their orders, and when the food was brought to their table. Finally, the time when the last diner leaves the restaurant.

- Multiple threads are used to handle incoming customer arrivals concurrently.
- Each thread represents a customer entering the restaurant, allowing for efficient processing of arrivals.
- Synchronization techniques are applied to ensure that multiple threads can safely access and modify the customer arrival queue without conflicts.
- When multiple threads are handling incoming customer arrivals, synchronization ensures that they don't interfere with each other while accessing and modifying the customer arrival queue.

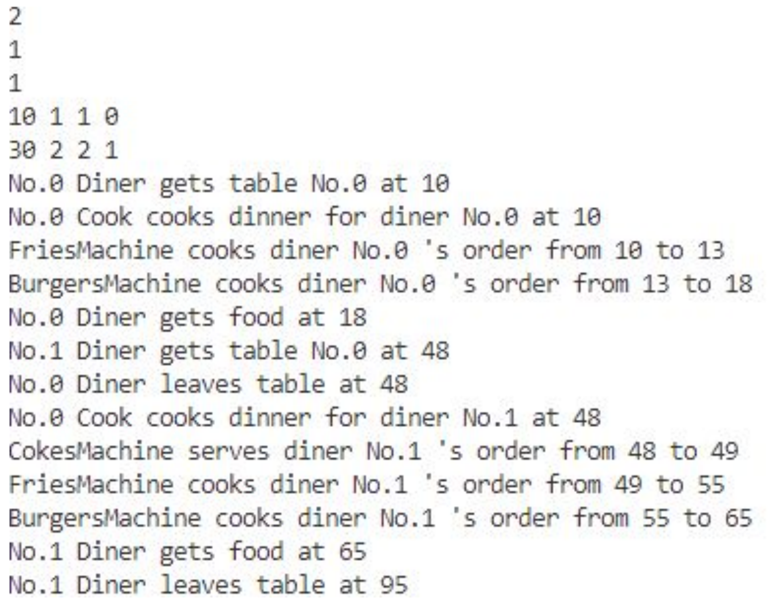
- Multithreading is employed to process cooking orders from the queue simultaneously.
- Each cooking task is executed in a separate thread, enabling multiple orders to be prepared concurrently.
- Synchronization mechanisms are implemented to coordinate access to cooking resources, such as kitchen equipment among the cooking threads.
- Synchronization techniques are used to ensure that table availability data is accessed and updated safely by the table management threads, preventing conflicts and inconsistencies.

- There are two main threads in this project diner thread and cook thread.
- Customer Arrival: Implementing threads handle incoming customer arrivals. Using a synchronized queue data structure to manage the customer arrival queue. Each thread in the pool represents a customer arrival task, allowing for concurrent processing.
- Cook Processing: Creating cooking threads to process orders from the queue. Utilizing synchronization primitives to coordinate access to cooking resources. Ensuring that each cooking thread retrieves orders from the queue in a synchronized manner to avoid conflicts.

Results

Input:
No. of diners
No. of tables
No. of cooks
Arrival time, no. of burgers, fries, cokes for diners

Output:



```
2
1
1
10 1 1 0
30 2 2 1
No.0 Diner gets table No.0 at 10
No.0 Cook cooks dinner for diner No.0 at 10
FriesMachine cooks diner No.0 's order from 10 to 13
BurgersMachine cooks diner No.0 's order from 13 to 18
No.0 Diner gets food at 18
No.1 Diner gets table No.0 at 48
No.0 Diner leaves table at 48
No.0 Cook cooks dinner for diner No.1 at 48
CokesMachine serves diner No.1 's order from 48 to 49
FriesMachine cooks diner No.1 's order from 49 to 55
BurgersMachine cooks diner No.1 's order from 55 to 65
No.1 Diner gets food at 65
No.1 Diner leaves table at 95
```

Thank you