GTU Department of Computer Engineering

CSE 344 - Spring 2024

Final Project Report

ALPER TAVŞANOĞLU

210104004142

Pide Shop

In this project, I implement The Pide Shop system. System is designed to manage a food delivery service where orders are received from clients, prepared by cooks, cooked in an oven, and delivered by delivery personnel to specified locations. The system utilizes multi-threading and networking to handle concurrent client connections and efficiently manage order processing.

The main task of the Pide Shop application is to simulate a food production and delivery system. This involves managing orders for pides from customers, coordinating the preparation and cooking of the pides in a special oven, and organizing their delivery by personnel on motorcycles. The system includes a manager to assign orders, cooks to prepare and cook the pides, and delivery personnel to transport the pides to customers. The application also choose the most efficient personel end of the client.

Design and Architecture

The application consists of two main components:

Server: Accepts connections from clients, receives and logs their positions, and sends messages to the connected clients.

Client: Connects to the server, sends its position, and listens for messages from the server. The server listens on a specified port for incoming client connections.

For each client connection, it has new process to handle communication with that client, allowing the server to handle clients simultaneously.

Client Design

The client performs the following tasks:

Connects to the server using the specified IP address (optional) and port number. Sends the number of clients and its (x, y) position to the server. Listens for messages from the server and displays them. Handles SIGINT (Ctrl-C) signal to close the connection and exit gracefully.

Server Design

The server performs the following tasks:

Accepts incoming client connections. Handle a new process for each client to order delivery. Receives the number of clients, and their positions (x, y coordinates) from each client. Sends messages to the connected clients. Handles SIGINT (Ctrl-C) signal to gracefully shutdown (after all orders done), ensuring all client connections are closed properly.

Threads and Thread Pools

Manager Thread: Coordinates order assignments from a queue to cook threads.

Cook Threads: Prepare orders, manage an oven capacity, and transition orders to the delivery queue.

Delivery Threads: Deliver prepared orders to clients based on their positions.

Client Handling Threads: Manage communication with individual clients, receive orders, and send responses

Data Structures

Order Struct: Stores details about each order, including state (received, preparing, cooking, ready, delivered).

Oven Struct: Manages the oven capacity and current orders being cooked.

Various Mutexes and Condition Variables: Ensure thread synchronization for accessing shared resources like order queues, client sockets, and efficiency tracking arrays.

Signal Handling

Handles SIGINT and SIGTSTP signals for graceful shutdown, closing sockets, and notifying threads to exit.

Efficiency Tracking

cook_efficiency and delivery_efficiency arrays track the number of orders processed by each cook and delivery person, respectively. These are logged and periodically printed to identify the most efficient personnel.

Logging

Utilizes a logging function (log_message) to record significant events (e.g., order assignment, delivery completion) in a file (shop_activities.log).

Socket Communication

Uses TCP sockets for communication between the server and clients. It manages multiple client connections concurrently and sends/receives order details and positions.

Communication Protocol

Server-Client Interaction: Clients connect to the server and send their order details. The server acknowledges connections, receives orders, assigns them to cooks, manages cooking and delivery, and sends status updates back to clients.

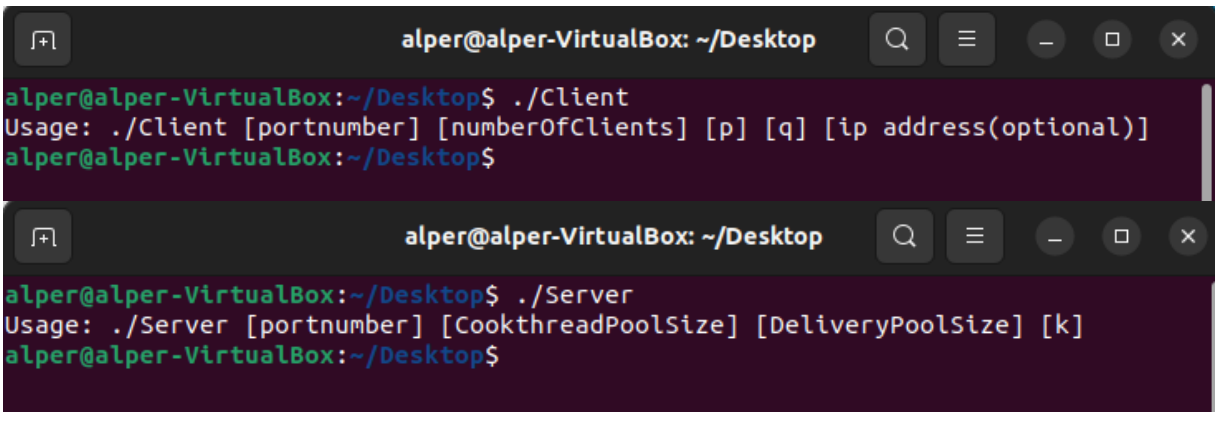
Order Lifecycle: Orders progress through states (ORDER_RECEIVED, ORDER_PREPARING, ORDER_COOKING, ORDER_READY, ORDER_DELIVERED) as they move from receiving to delivery. Each state transition is logged and may trigger updates to client connections.

Shutdown: Upon receiving a shutdown signal (SIGINT or SIGTSTP), the server broadcasts signals to all threads to close connections and shut down gracefully.

The PideShop system exemplifies a robust application of multi-threading and network programming to manage concurrent food delivery orders. It demonstrates effective use of synchronization techniques and signal handling for managing server shutdowns. The integration of logging ensures transparency in system activities, aiding in debugging and performance monitoring.

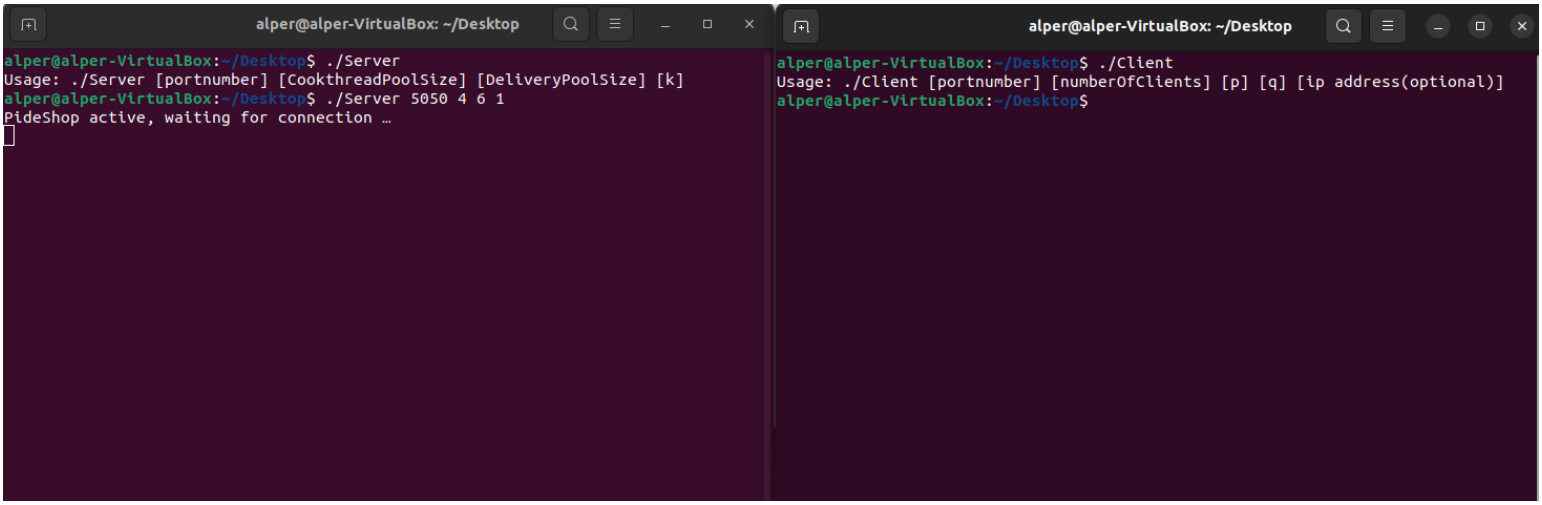
Compile and Examples

For compile the code we can use Makefile. With **make** command program will compiled. Makfile also include makeclean command. **Usage:**

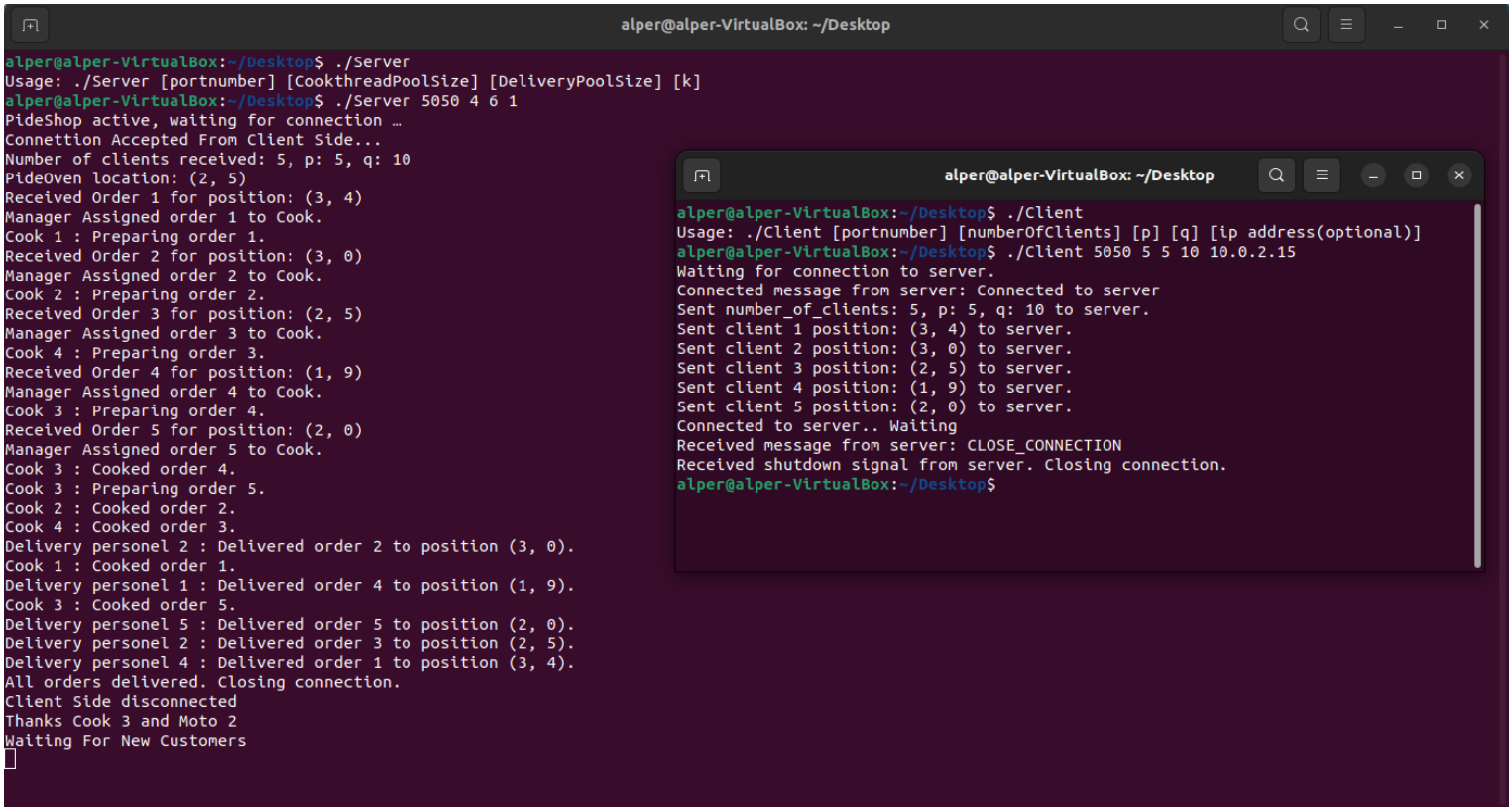


Test Examples

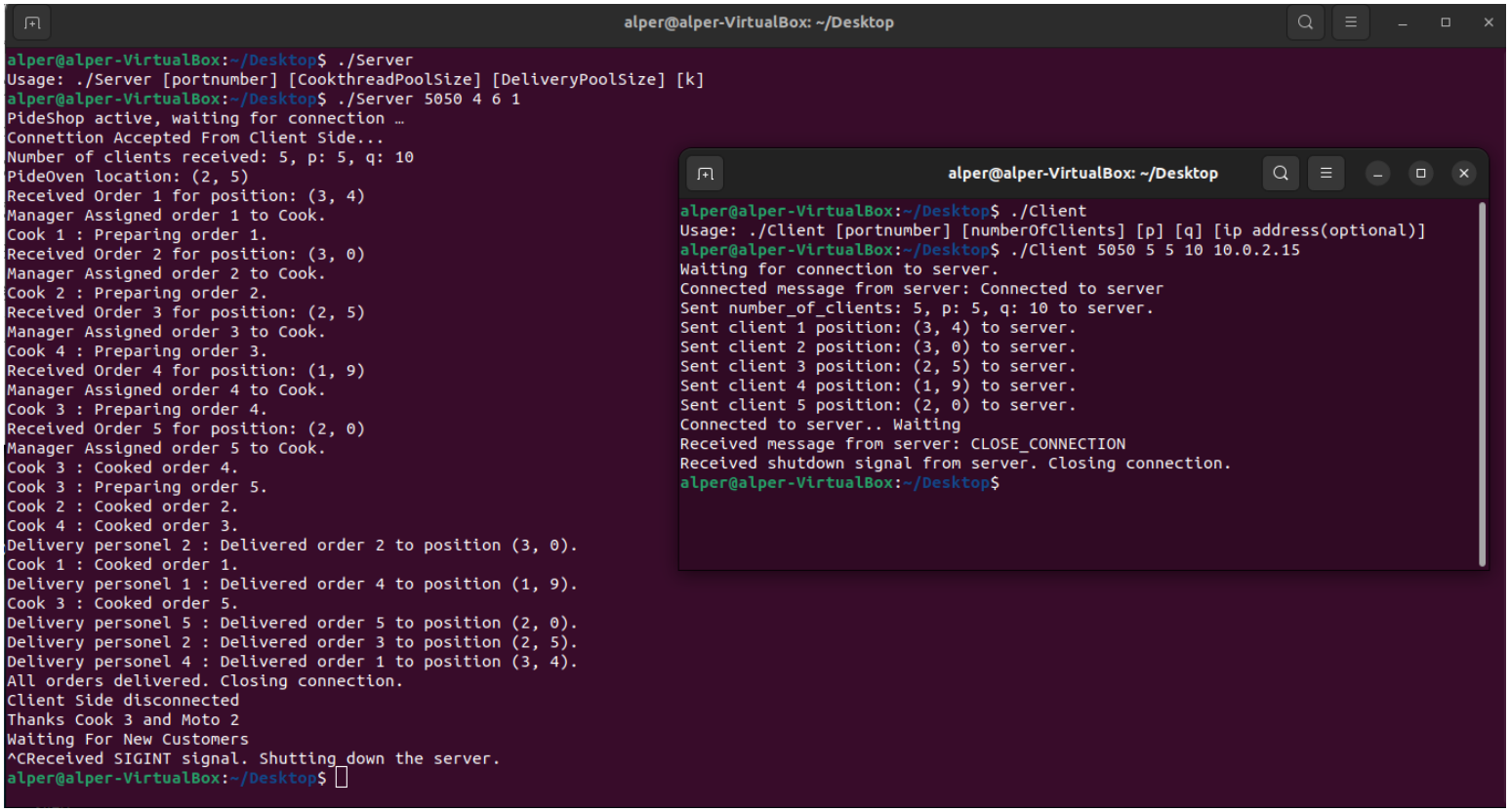
Server waiting for customers



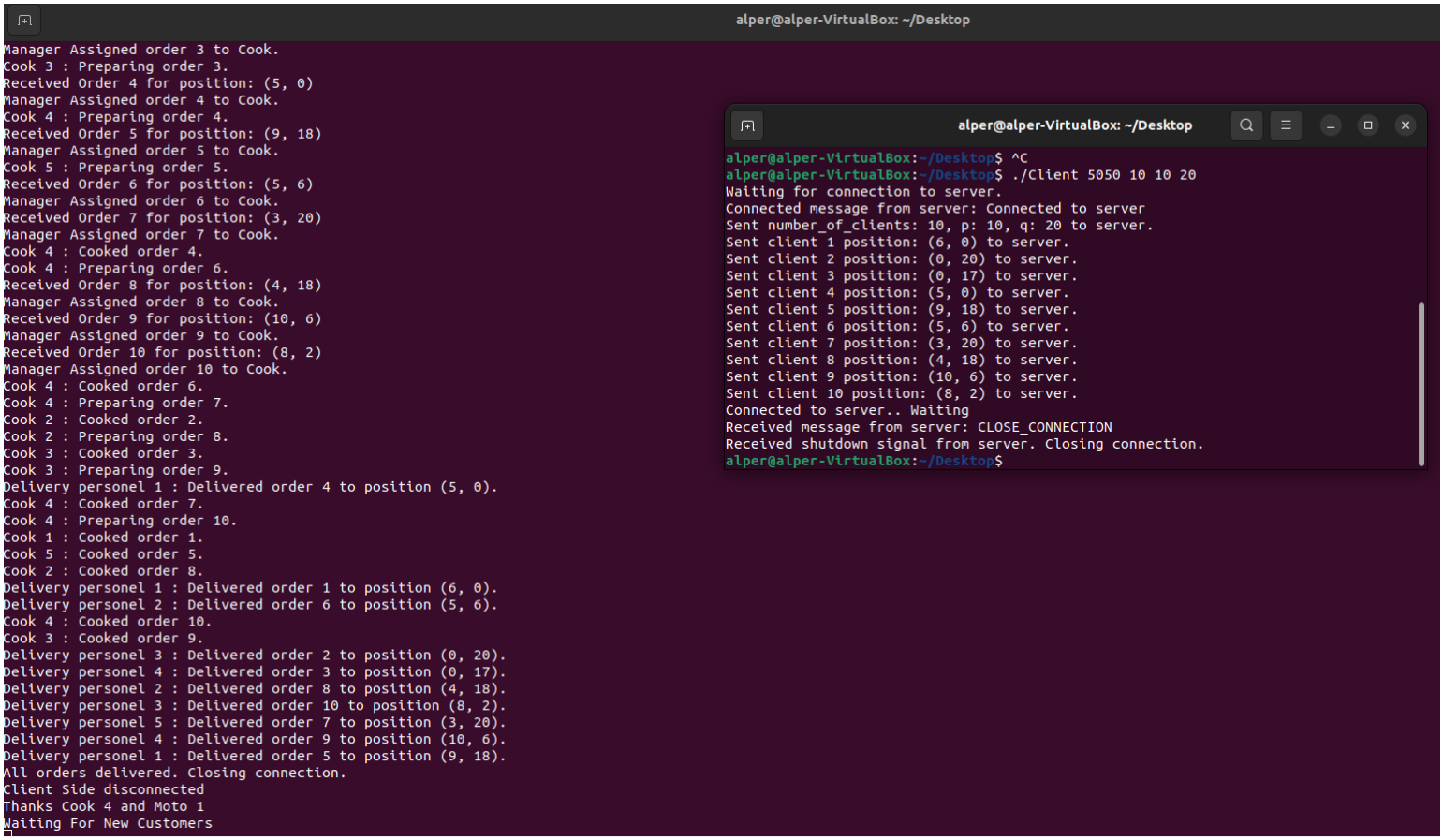
Server still active Client closed after all orders delivered



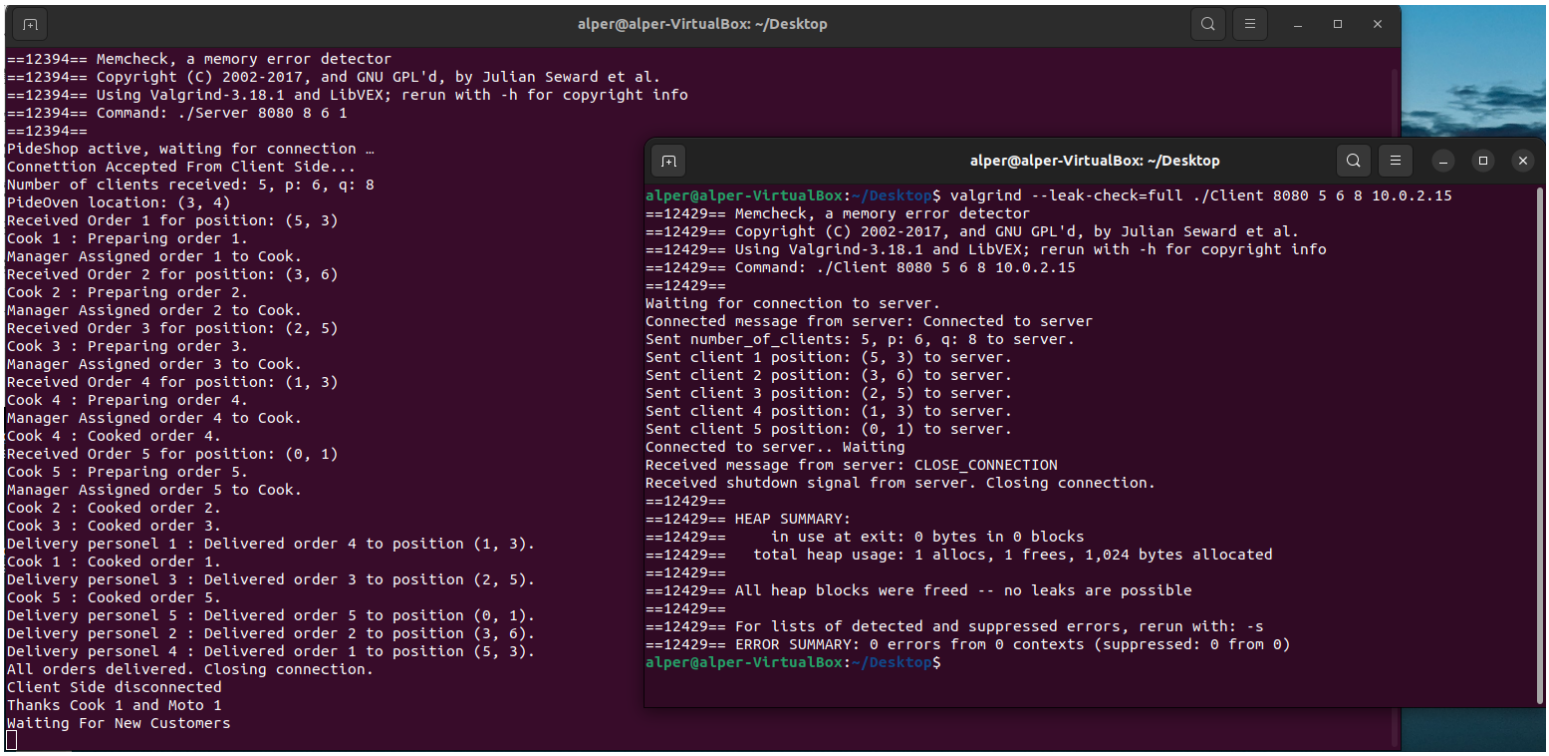
^C Signal Check



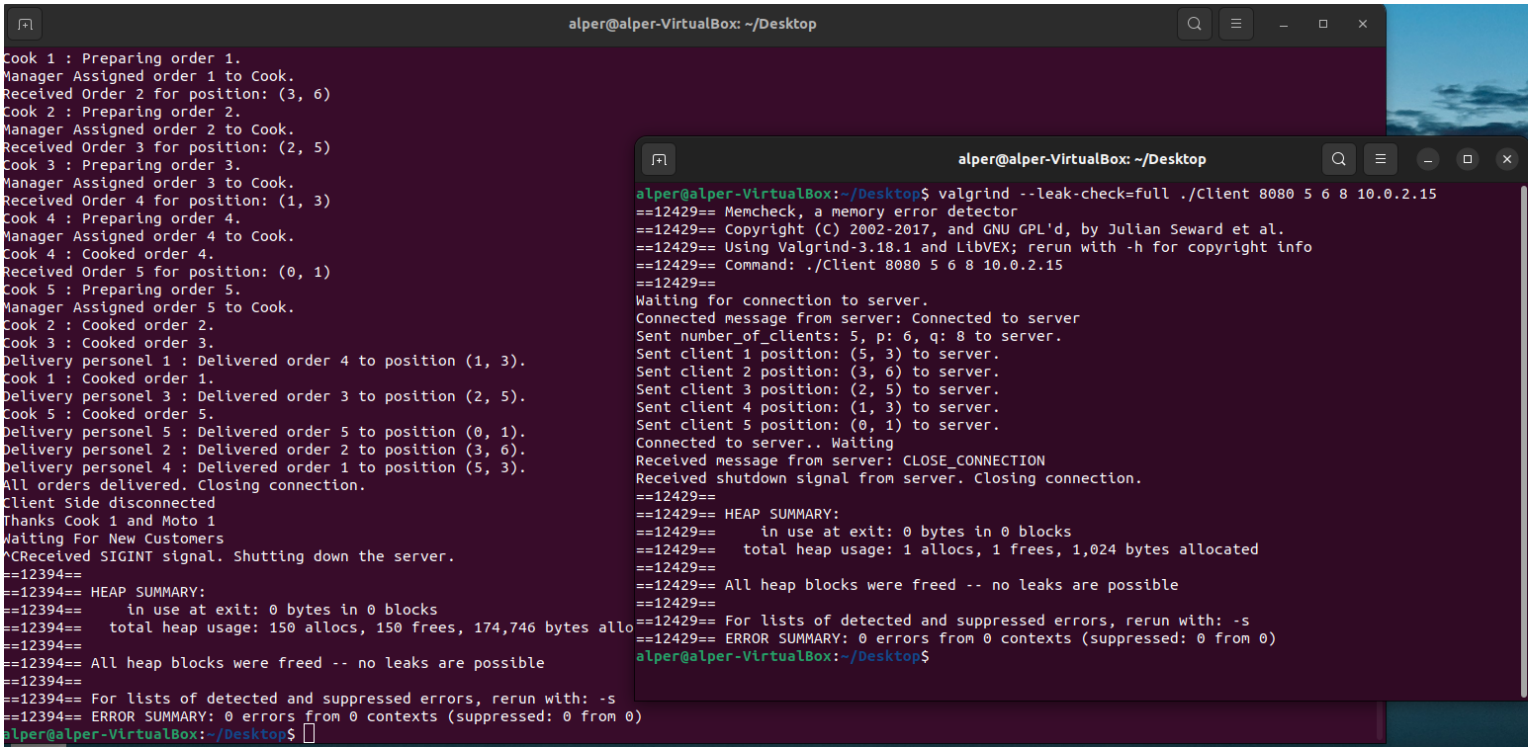
Another Example



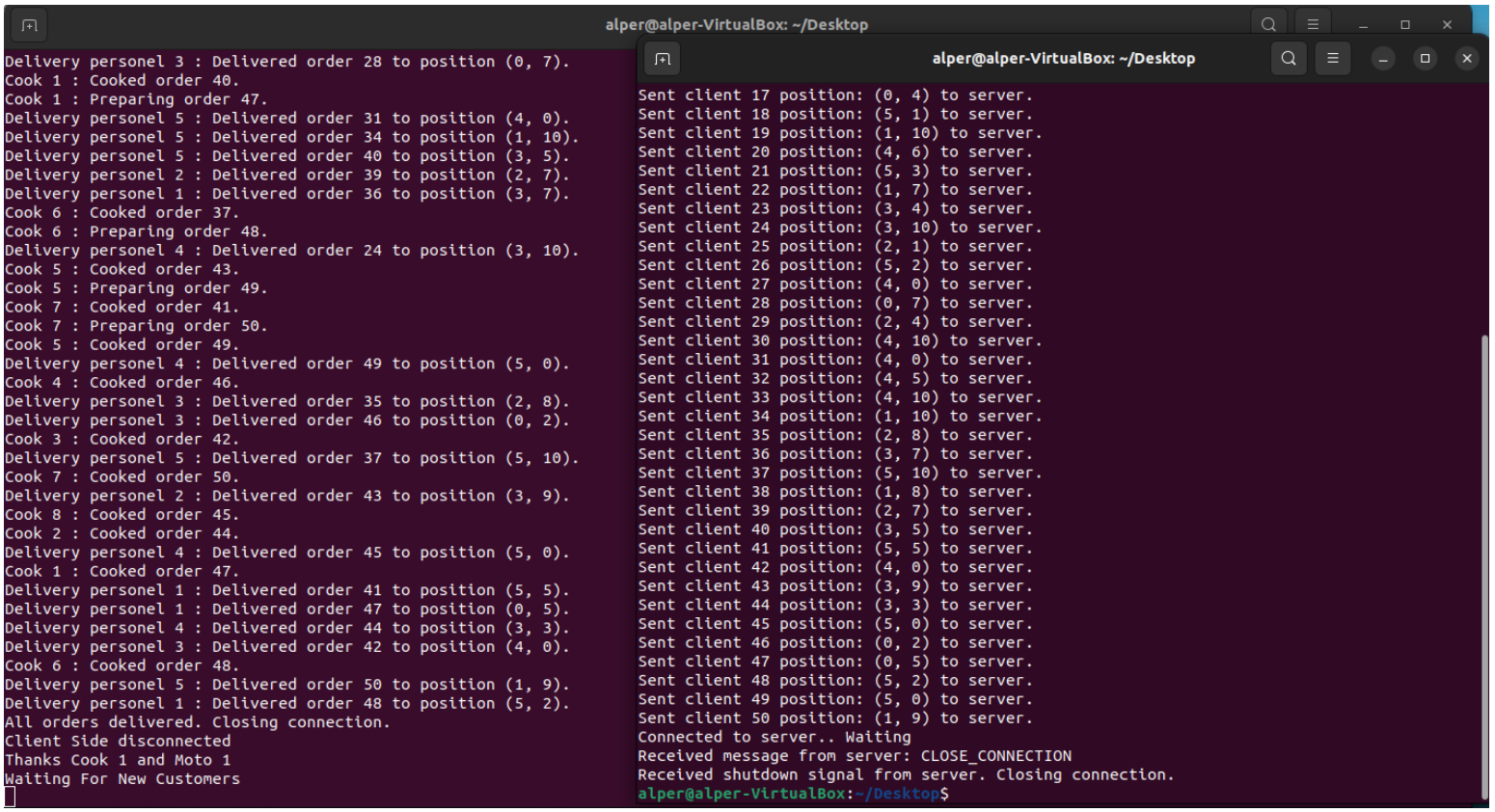
Example With Valgrind –leak-check



Then Close the Server with ^C



Tests With Multiple Clients (50)

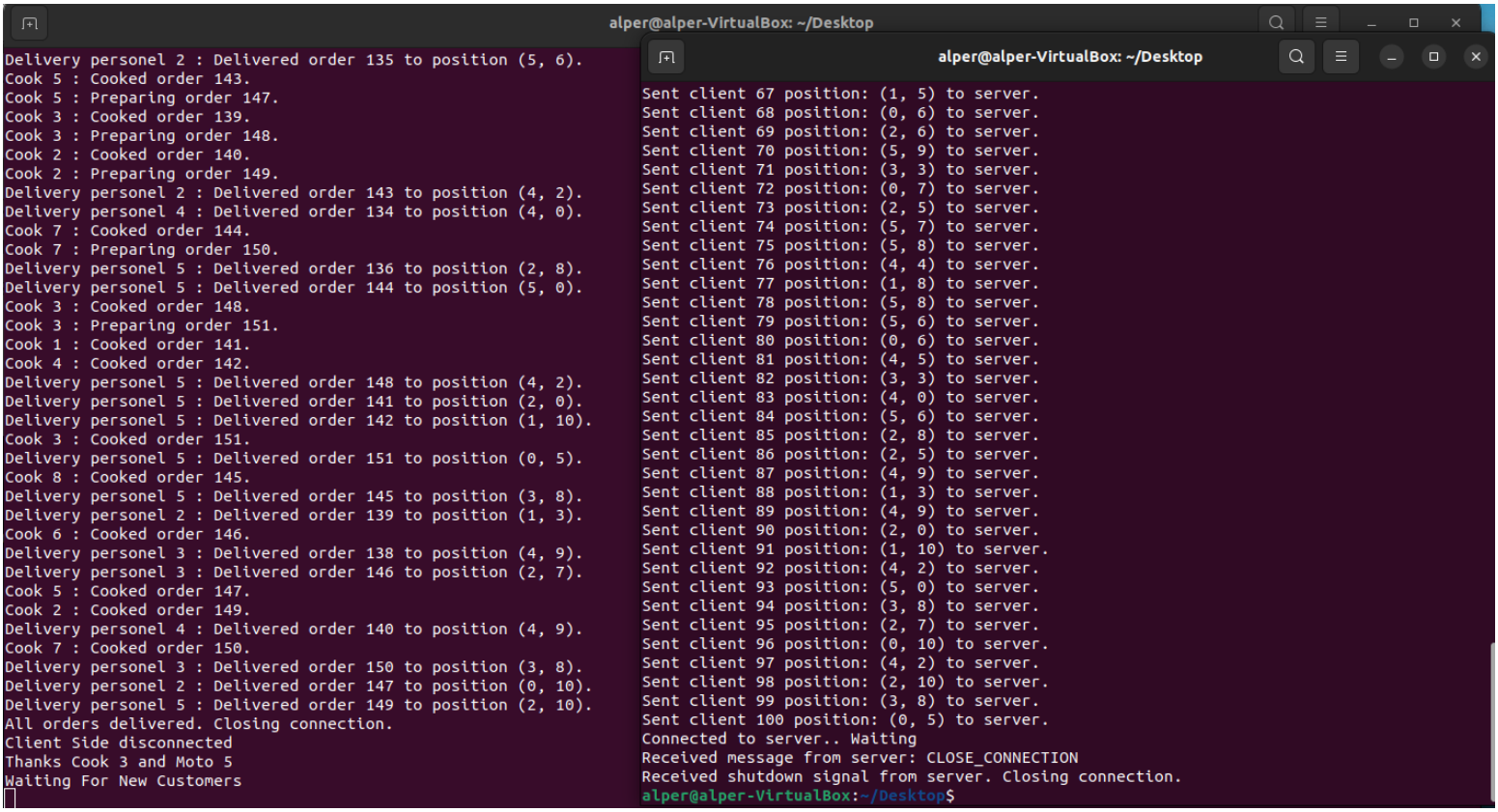


Log File

```
[Sat Jun 15 06:09:37 2024] PideShop active, waiting for connection …
[Sat Jun 15 06:09:48 2024] Connection Accepted From Client Side...
[Sat Jun 15 06:09:48 2024] Number of clients received: 50, p: 5, q: 10
[Sat Jun 15 06:09:48 2024] PideOven location: (2, 5)
[Sat Jun 15 06:09:48 2024] Received Order 1 for position: (2, 7)
[Sat Jun 15 06:09:48 2024] Manager Assigned order 1 to Cook.
[Sat Jun 15 06:09:48 2024] Cook 1 : Preparing order 1.
[Sat Jun 15 06:09:48 2024] Received Order 2 for position: (3, 10)
[Sat Jun 15 06:09:48 2024] Manager Assigned order 2 to Cook.
[Sat Jun 15 06:09:48 2024] Cook 2 : Preparing order 2.
[Sat Jun 15 06:09:48 2024] Received Order 3 for position: (5, 4)
[Sat Jun 15 06:09:48 2024] Manager Assigned order 3 to Cook.
[Sat Jun 15 06:09:48 2024] Cook 3 : Preparing order 3.
[Sat Jun 15 06:09:48 2024] Received Order 4 for position: (2, 5)
[Sat Jun 15 06:09:48 2024] Manager Assigned order 4 to Cook.
[Sat Jun 15 06:09:48 2024] Cook 4 : Preparing order 4.
[Sat Jun 15 06:09:48 2024] Received Order 5 for position: (5, 3)
[Sat Jun 15 06:09:48 2024] Manager Assigned order 5 to Cook.
[Sat Jun 15 06:09:48 2024] Cook 5 : Preparing order 5.
[Sat Jun 15 06:09:48 2024] Received Order 6 for position: (0, 1)
[Sat Jun 15 06:09:48 2024] Manager Assigned order 6 to Cook.
[Sat Jun 15 06:09:48 2024] Cook 6 : Preparing order 6.
[Sat Jun 15 06:09:48 2024] Cook 4 : Cooked order 4.
[Sat Jun 15 06:09:48 2024] Cook 6 : Cooked order 6.
[Sat Jun 15 06:09:48 2024] Received Order 7 for position: (5, 6)
[Sat Jun 15 06:09:48 2024] Manager Assigned order 7 to Cook.
[Sat Jun 15 06:09:48 2024] Cook 7 : Preparing order 7.
[Sat Jun 15 06:09:48 2024] Received Order 8 for position: (4, 0)
[Sat Jun 15 06:09:48 2024] Manager Assigned order 8 to Cook.
[Sat Jun 15 06:09:48 2024] Cook 8 : Preparing order 8.
[Sat Jun 15 06:09:48 2024] Received Order 9 for position: (2, 0)
[Sat Jun 15 06:09:48 2024] Manager Assigned order 9 to Cook.
[Sat Jun 15 06:09:48 2024] Cook 4 : Preparing order 9.
[Sat Jun 15 06:09:48 2024] Received Order 10 for position: (4, 1)
[Sat Jun 15 06:09:48 2024] Received Order 11 for position: (2, 6)
[Sat Jun 15 06:09:48 2024] Manager Assigned order 10 to Cook.
[Sat Jun 15 06:09:48 2024] Cook 6 : Preparing order 10.
[Sat Jun 15 06:09:48 2024] Received Order 12 for position: (0, 4)
[Sat Jun 15 06:09:48 2024] Received Order 13 for position: (5, 2)
[Sat Jun 15 06:09:48 2024] Received Order 14 for position: (5, 9)
[Sat Jun 15 06:09:48 2024] Received Order 15 for position: (3, 0)
[Sat Jun 15 06:09:48 2024] Received Order 16 for position: (2, 8)
[Sat Jun 15 06:09:48 2024] Manager Assigned order 11 to Cook
```

Tests With Multiple Clients (100)

I make his test without closing server so total orders 50+100



Tests With Multiple Clients (200)

I make his test without closing server so total orders 50+100+200