CSE-344

SPRING 2024

FINAL PROJECT

REPORT

ONUR ATASEVER

210104004087

In the assignment we are expected to build a system by ensuring proper synchronization between cooks, oven, deliveries and customers. The main idea is using mutexes and semaphores to provide correct synchronization. There are two sides whichare server (pideci) and client (customers). In the client side each customer is created with threads so they can connect to the server at the sime time by using socket. In the server side there are two thread pools for deliveries and cooks. And also there is one more thread for manager.

There are two ways to provide comminucation by using socket. One of them is using only port and other one is ip adress. When the program starts it checks the number of arguments. According to number it decides to use ip or port.

When program starts threads are created. Manager thread is in a infinite loop because it always check if there is a new order. It creates new socket for each accept. It is designed as nonblocking because if the orders are gotten from customers, manager should not wait the orders, it should continue because it gives order to the cooks and then gives to the deliveries. When order comes, manager adds it to the order queue and it checks the available cooks. It finds first available cook and it send condition signal to wake them up. Cook gets order from queue and prepares. Then by using semaphore protection it tries to put the order to the oven. If semaphore returns something other than zero it means entries fort he oven is full and cook should wait. Then it checks the number of orders in the oven with a variable. If they are more than 6 it waits again. When cook gets order from oven it puts order to the delivery order queue. For each order which comes from oven semaphore is used.

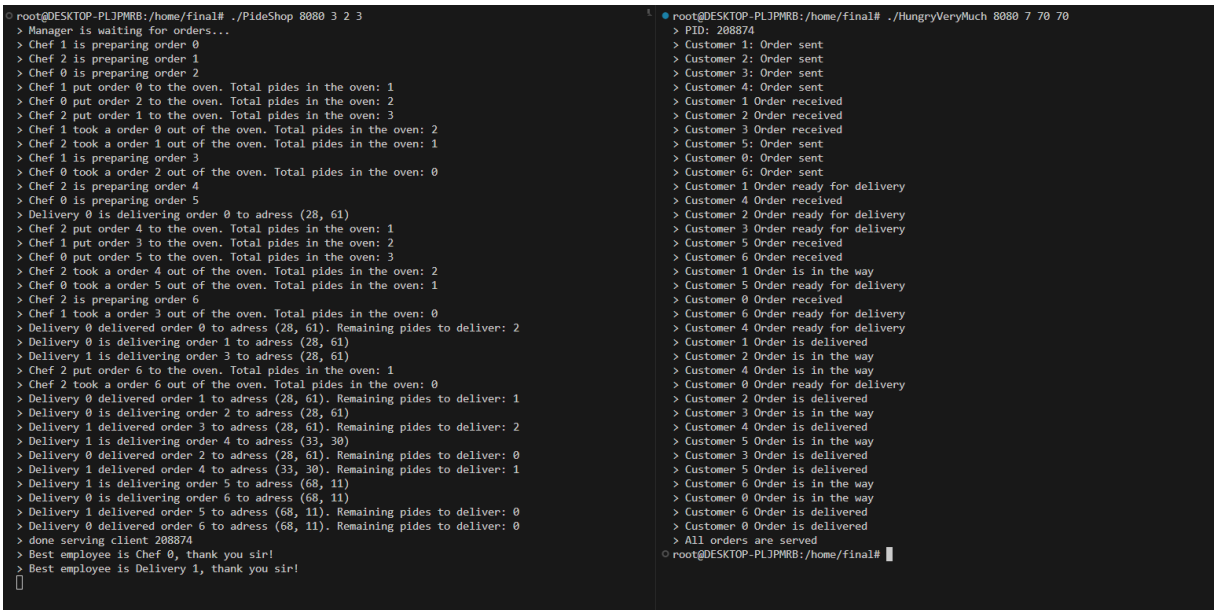
In the manager thread, it tries to find first available delivery to give order. By using semaphore it checks if there are ready order. It uses sem_wait not to block manager thread. If there is order, it pops from general delivery queue and it assings to the delivery's personel delivery queue. It tries to give first delivery person. If delivery person has max order, it goes second delivery and checks the number of order of the delivery. For each delivery assignment, mutex is used. Each mutex is unique fort he deliveries. When the delivery has max number order, it means delivery is ready to go to customers. It sends condition signal to the delivery thread and If delivery person is not available (which means he is assigned for delivering) it checks it's own personal order delivery queue and it distributes orders.

For each step which are order received, prepared, in the way, delivered, server sends message to the client. And for each customer steps are printed. When CTRL C signal comes to the server, signal handler calls kill function to terminate client. To terminate client, it sends sigint signal to the client. In the signal handler which is in the client handles this signal and terminates the client. When the order came, client informations are stored in the order struct (like socket number, client pid, customer

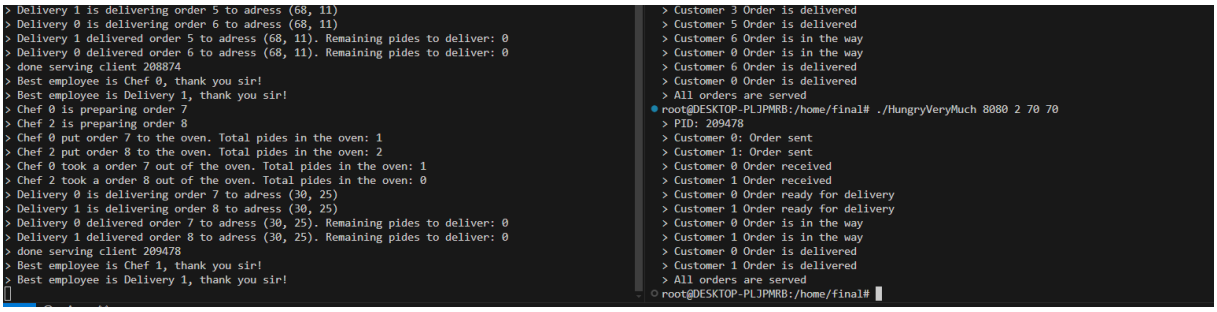
coordinates). When CTRL C interrupt comes to the client side, it sends it to the server side by using SIGPIPE signal. And server deletes queue.

For each operation fort he cook and delivery, when their thread is started to the operation, it records time. There are array of cooks and delivery people. Each of them has their own struct. And their total work time is stored in this struct. In their struct they have their own mutexes and their own order queues. So they do not have to wait other people to do their job. And with special queues each order is assigned to a spesific delivery people and cooks. And ıt makes easier providing sycnhronization. And each location of the customers are generated randomly and stored in order struct.

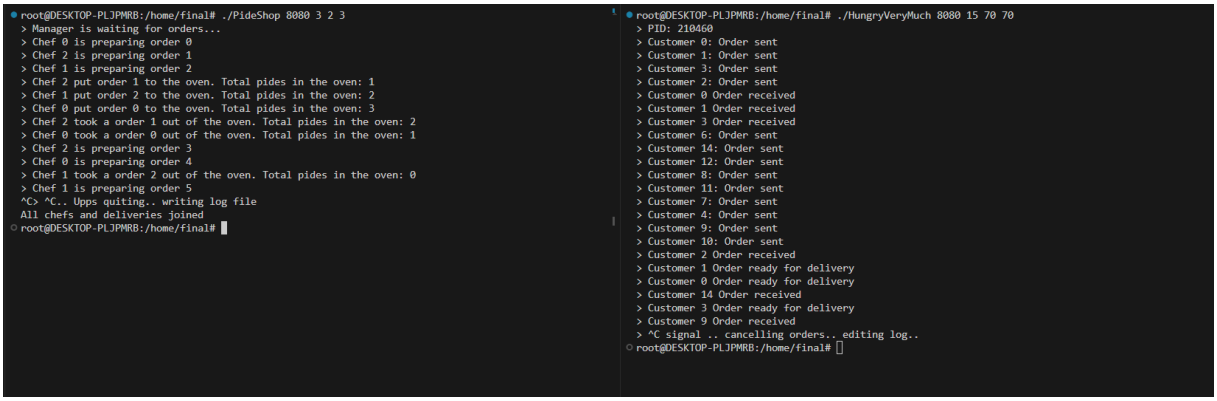
By using port number:



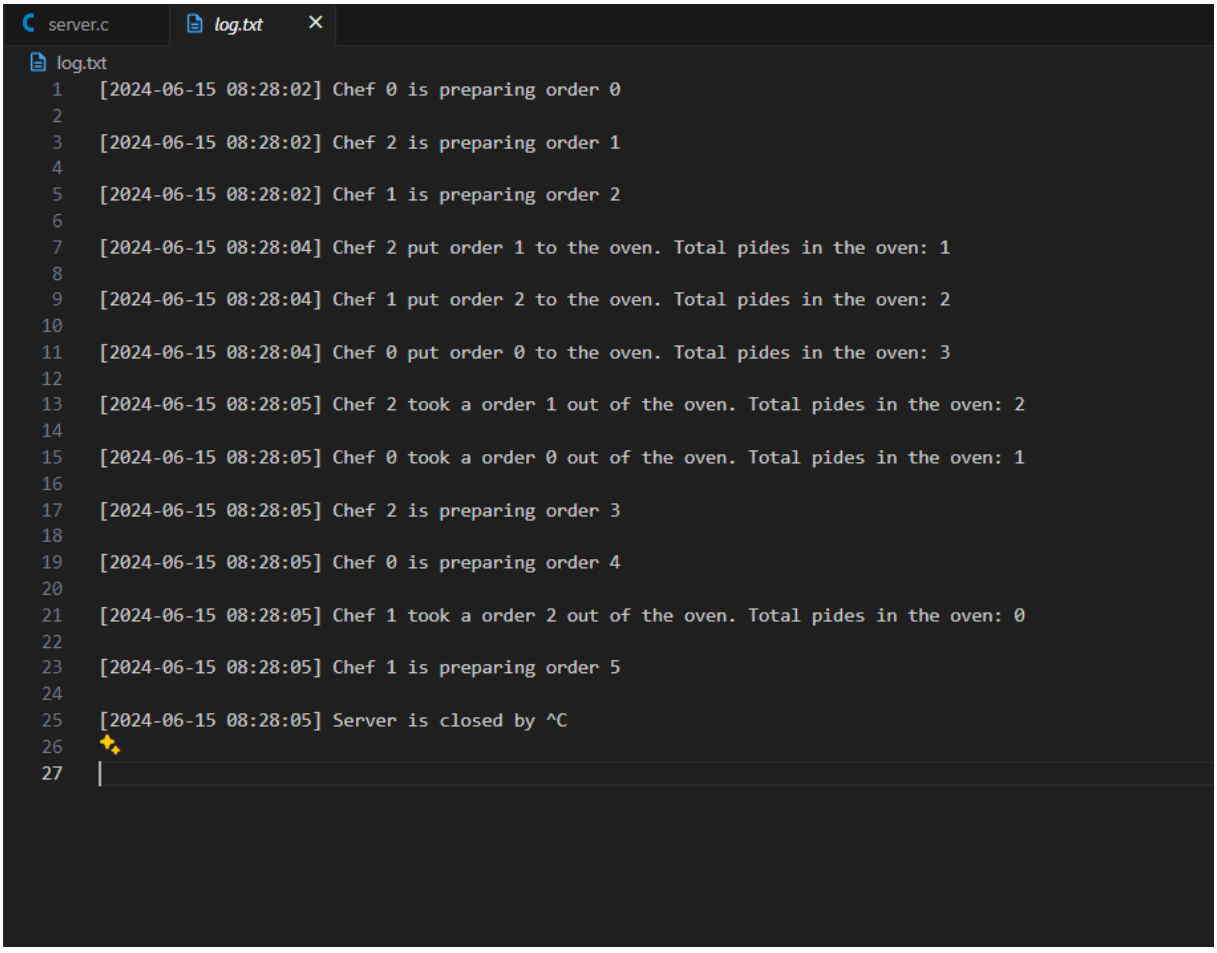
Server is still open an done more client connected:



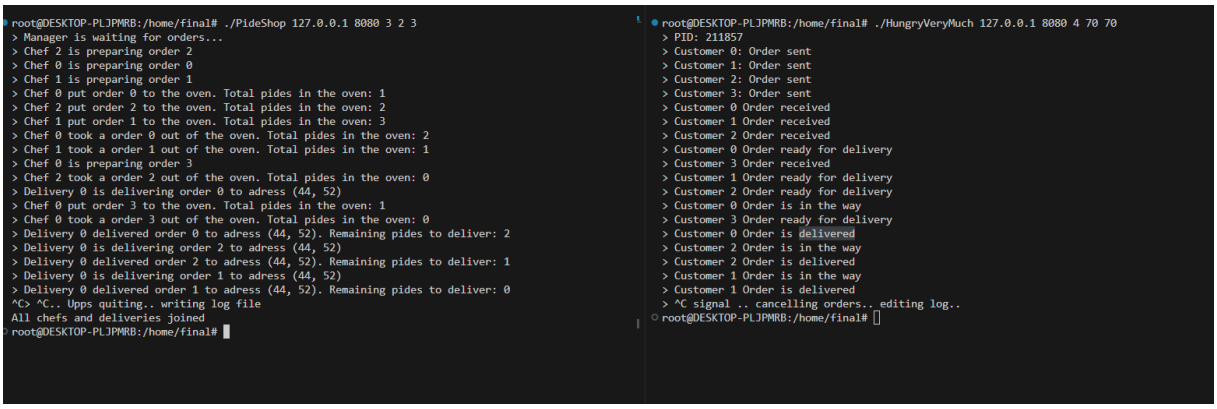
Server interrupted, and client is terminated



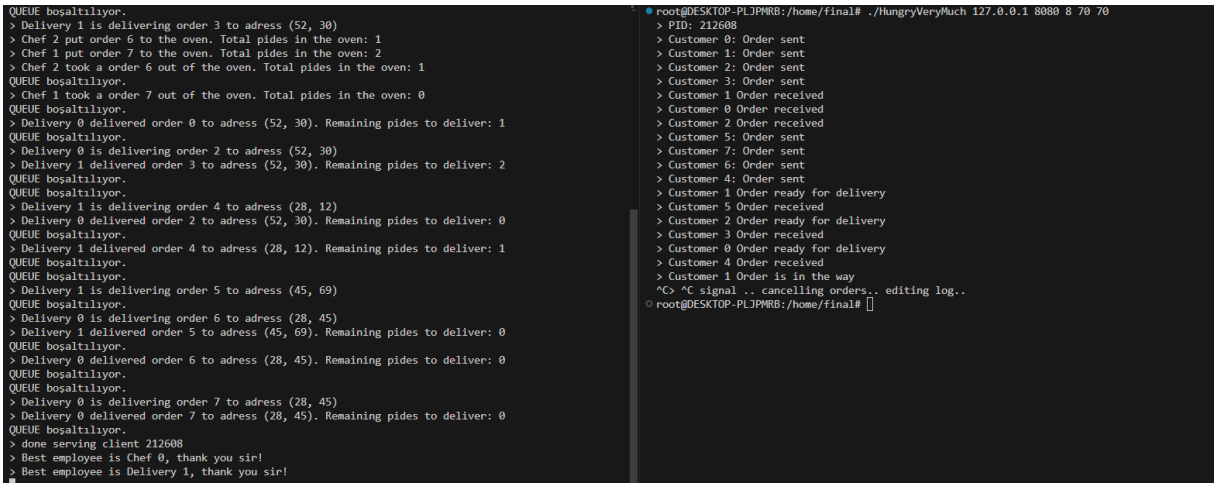
Log file:



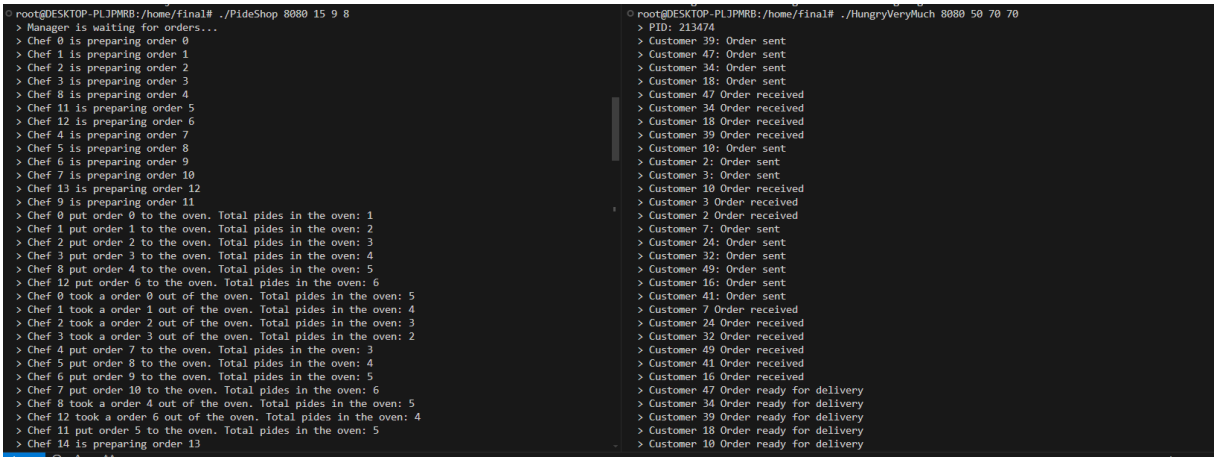
Client connected with ip adress and server interrupted:



Client side is interrupted:



With large numbers of clients, cooks and deliveries:



When it is interrupted: