Online Store Mini Project

Aasmaan Gupta IMT2021006

Overview-

This project is designed to simulate the functioning of an OnlineStore through a terminal-based application. The project utilizes a client-server architecture, where client requests are sent via sockets to the server program. The server processes these requests and returns the necessary results. Different privileges are assigned based on the user type. Regular users, known as customers, have the ability to view their own shopping carts, while the Admin user possesses special privileges to update store items.

The program employs socket programming to facilitate communication between the client and server. Interactions between the client and server are facilitated through system calls, and the server interacts with data files using file locking mechanisms.

Files-

The project consists of several files with specific functionalities:

- Client.c: This file contains the code that is executed on the client side.
- Server.c: This file contains the code that is executed on the server side.
- Customers.txt: This file stores the information of registered customers for the store.
- Orders.txt: This file stores the shopping carts for each user.
- Records.txt: This file holds the current inventory of the store, including product details, quantities, and prices.
- receipt.txt: This file is used to store the receipt generated for the user after making a payment.
- AdminReceipt.txt: This file maintains a history of updates, deletions, and additions made by the admin to the products in the store.
- headers.h: This file contains the necessary structs and macros used throughout the program.

How to run Code-

-open two terminal windows.
-In first Window compile Server.c as shown below

```
% gcc -o Server Server.c
mayank@Mayanks-MacBook-Air FinalOsProject % ./Server
Initializing the server...
Server initialization completed successfully...
```

-Then In Second Window, run the commands

```
[mayank@Mayanks-MacBook-Air FinalOsProject % gcc -o Client Client.c
[mayank@Mayanks-MacBook-Air FinalOsProject % ./Client
```

-Then run the program as is directed by the client program.

Functionality of Client.c-

The program provides specific functionalities to different types of users:

Normal User (Customer):

- Register as a new customer.
- View the available items in the store, including quantities and prices.
- Add products to the cart, specifying the desired quantity.
- Edit the quantities of existing products in the cart, including deleting items.
- Proceed to make payment for all the products in the cart. Upon successful payment, a receipt is generated for the customer to view their order.

Admin User:

- Add a new product to the online store.
- Update the quantity or price of an existing product in the store.
- Remove products from the store.
- View the inventory, which displays the items currently available in the store.
- Any updates made by the admin are recorded in the adminReceipt.txt file to maintain a transaction history.

Functionalities of the server-

The server component of the program is responsible for serving the requests of both customers and the admin. It is designed to support the functionalities mentioned earlier. Here are the key aspects of the server's functionality:

1. Handling Client Requests: The server receives requests from clients (customers or the admin) and processes them accordingly. It listens for incoming requests and responds appropriately based on the type of request.

2. User Authentication: The server authenticates clients to determine whether they are customers or the admin. This ensures that only authorised users can access specific functionalities.

3. Customer Functionality: For customer requests, the server handles functionalities such as registration, viewing available items and their details (quantities and

prices), adding products to the cart, modifying cart items (including quantity changes and deletions), and facilitating payment processing.

4. Admin Functionality: For admin requests, the server supports functionalities such as adding new products to the store, updating quantities or prices of existing products, removing products from the store, and providing inventory information.

5. Data Management: The server interacts with various data files, such as customers.txt, orders.txt, records.txt, receipt.txt, and adminReceipt.txt, to read and update relevant information. It ensures data consistency and proper synchronisation when multiple clients access or modify the data simultaneously.

6. Logging and History: The server maintains a transaction history by logging updates, deletions, and additions made by the admin in the adminReceipt.txt file. This helps track changes and provides an audit trail.

Socket SetUp On Client Side-

The main function sets up the socket connection to the server. Here's the rephrased code:

```
printf("-----------------------------------------\n");
printf("Initializing connection to the server...\n");

int sockfd = socket(AF_INET, SOCK_STREAM, 0);

if (sockfd == -1)
{
    perror("Error: ");
    return -1;
}

struct sockaddr_in serv;

serv.sin_family = AF_INET;
serv.sin_addr.s_addr = INADDR_ANY;
serv.sin_port = htons(5555);

if (connect(sockfd, (struct sockaddr *)&serv, sizeof(serv)) == -1)
{
    perror("Error: ");
    return -1;
}
```

Socket SetUp On Server Side-

The main function sets up the socket connection to the Client. Here's the rephrased code:

```
printf("Initializing the server...\n");

// file containing all the records is called records.txt

int fdRecord = open("Records.txt", O_RDWR | O_CREAT, 0777);
int fdCart = open("Orders.txt", O_RDWR | O_CREAT, 0777);
int fdCust = open("Customers.txt", O_RDWR | O_CREAT, 0777);
int fdAdmin = open("AdminReceipt.txt", O_RDWR | O_CREAT, 0777);

lseek(fdAdmin, 0, SEEK_END); // This is done to ensure that any data written to

int sockfd = socket(AF_INET, SOCK_STREAM, 0); //

if (sockfd == -1)
{
    perror("Error: ");
    return -1;
}
// the struct sockaddr_in structure is used to represent the server address (se
// client address (cli) for the socket communication.
struct sockaddr_in serv, cli;

// Overall, these lines initialize the server's address structure with the appr
// IP address (any available interface), and port number (5555) that the server
serv.sin_family = AF_INET;         // This line sets the address family of the
serv.sin_addr.s_addr = INADDR_ANY; // the server will be able to accept connect
serv.sin_port = htons(5555);

int opt = 1;
if (setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt)))
{
    perror("Error: ");
    return -1;
} // allowing the server to reuse the address it was bound to even if it was re

if (bind(sockfd, (struct sockaddr *)&serv, sizeof(serv)) == -1)
{
    perror("Error: ");
    return -1;
} // bind function is used to associate a socket with a specific address and po

if (listen(sockfd, 5) == -1)
{
    perror("Error: ");
    return -1;
}

int size = sizeof(cli);
printf("Server initialization completed successfully...\n");
```

Implementing Server Side functions-

In the program, the input sent by clients regarding the user type and their choices is read, and the appropriate helper functions are called based on the case.

Here's a breakdown of the user functions:

1. If the user chooses to exit the program, the loop is broken, and the program terminates.
2. If the user chooses to see the inventory, the DisplayProducts() function is called to display the list of available products.
3. If the user chooses to view their cart, the DisplayCart() function is called to show the contents of their cart.
4. If the user chooses to add a product to their cart, the AddProductToCart() function is called to facilitate the addition of the product.
5. If the user chooses to edit a product in their cart, the UpdateProductInCart() function is called to allow modifications to the product quantity or removal from the cart.
6. If the user chooses to pay for their cart, the BillPayment() function is called to initiate the payment process.
7. Lastly, if the user wants to register as a new customer, the CustomerAdd() function is called to add them as a new customer.

Here are the admin functions described:

1. If the admin chooses to add a product to the store, the AddProduct() function is called.
2. If the admin chooses to delete a product from the inventory, the ProductDelete() function is called.
3. If the admin chooses to edit the price of a product in the inventory, the UpdateProduct() function is called with the `ch` parameter set to 1.
4. If the admin chooses to edit the quantity of a product in the inventory, the UpdateProduct() function is called with the `ch` parameter set to 2.
5. If the admin chooses to see the inventory, the DisplayProducts() function is called to display the list of products.
6. If the admin chooses to exit the program, the loop is broken, and the program terminates.

Concepts Used In Implementation-

The program incorporates file locking, socket programming, and file handling in the following ways:

1. File Locking:
    - When searching for a customer ID in the "customers.txt" file, a mandatory read lock is applied to the entire file.
    - Once the offset of the customer's cart is found, record locking is used at that offset in the "orders.txt" file to lock the cart for reading or writing.
    - When searching for a product ID in the list of products or displaying the inventory, a mandatory read lock is applied to all products.
    - When updating a specific product, the read lock on the products is released, and a write lock is acquired on that product to modify it.
2. Socket Programming:
    - The program utilizes socket system calls for communication between the server and the client.
    - Server-side socket system calls (socket, bind, listen, accept) are used to set up the server-side socket, while client-side socket system calls (socket, connect) are used on the client side.
    - The fork() system call is used to establish a concurrent server, allowing multiple clients to be handled simultaneously.
    - Read and write operations from the socket employ read and write system calls.
3. File Handling:
    - Basic file handling functions such as open, read, and write are employed to perform read and write operations on the files.