

MINI PROJECT

This project is a command-line application designed to simulate the functions of an online store, using a client-server architecture. Requests made by the client are transmitted to the server program using sockets, which then processes the requests and returns the appropriate results. User roles are assigned different privileges - a regular user, who is a customer, can only view their own cart, while the Admin user has additional privileges to update items in the store.

The program utilizes Socket programming for communication between the server and the client. Both the client and server sides utilize system calls to interact with each other, while the server side utilizes file locking mechanisms to interact with the data files.

Architecture -

The file named Client.c contains the code that is intended to be executed on the client side, while Server.c contains the code intended to be executed on the server side.

Customers.txt is the file that stores information about the users registered with the online store. Orders.txt stores the contents of each user's cart. Records.txt contains the store's inventory, including the products available, their prices, and quantities.

Receipt.txt contains the receipt generated for the user after payment. AdminReceipt.txt is used to maintain a record of the updates, deletions, and additions made by the admin to the products. Headers.h is a file that includes the structures and macros that will be utilized in the program.

Run the code- In terminal 1 run the command `$ gcc -o Server Server.c` followed by `$./Server`

In terminal 2 run the command `$ gcc -o Client Client.c` `$./Client`

Structs and macros- The structure named "product" is used to store information about a product, including its ID, name, quantity, and price. The "cart" structure is utilized to store information about a customer's cart, including their ID and a list of products in their cart. The maximum number of products allowed in a cart is set to 20, which is referred to as MAX_PROD. Lastly, the "index" structure includes a customer's ID and their cart offset within the orders.txt file. Customers.txt acts as an index file to orders.txt.

Functionality- The program offers various functionalities to normal users or customers, such as registering as a new customer, viewing available products in the store with their

respective quantities and prices, adding products to their cart with the desired quantity, editing the quantities of existing products in their cart, deleting items from their cart, and proceeding to pay for all the products in their cart. Once payment is successful, a receipt is generated for the user to view their order.

On the other hand, the admin user serves as the administrator of the online store and has access to additional functionalities, such as adding a new product to the store, updating the quantity or price of an existing product, explicitly removing products from the store, and viewing the current inventory of the store. Any updates made by the admin are recorded in the adminReceipt.txt file for transaction history purposes.

Server serves the request of client/admin.

Implementation-

The functions `dispmenuuser` and `dispmenuadmin`

are used to display menus to the customer and admin respectively, and are called inside while loops to allow the user to choose from the available options. The menus contain different options for the customer (options a-g) and admin (options a-f). The functions `prodprint` and `getinv` are used to format and display reports to the user based on the responses received from the server.

The functions `calactot()` and `getreceipt`

are used to calculate the total cost of items in the user's cart and generate a receipt after payment has been made. `getreceipt` function sends the calculated total and cart information to the server for receipt generation. These functions are called when the user chooses to pay for the items in their cart.

The functions `takecustid()`, `takeprodid()`, `takeqty()`, and `takeprice()` are used to take input from the user and ensure that the input values are non-negative. Negative input values may lead to unexpected results and are not desirable. These functions enforce constraints on the user input values.

Customer-

1. To exit the menu, the loop is terminated using the "break" statement.
2. The "getinv" function is used to request product information from the server, and the response is then displayed to the user.

3. To retrieve a customer's cart, the user is prompted to enter their customer ID, and the server returns the corresponding cart. If an invalid customer ID is entered, an error message is displayed.
4. To add a product to the cart, the user provides their customer ID, the product ID, and the desired quantity. Any errors are communicated to the user, and if successful, the cart is updated on the server.
5. Editing a product in the cart follows the same procedure as adding a product.
6. When paying for the cart, the cart is displayed to the user along with the total cost. The user then enters the payment amount, and if the correct amount is entered, a receipt is generated. If an incorrect amount is entered, the user is prompted to enter it again.
7. To register as a new customer, the user confirms their registration and is provided with an automatically generated customer ID.

Admin-

1. When adding a new product, the user inputs the product id, name, quantity in stock, and price. This information is then sent to the server to add the product to the inventory. If there are any errors, an appropriate error message is displayed.
2. To delete a product from the inventory, the admin only needs to input the product id. If the id is correct, the product is deleted from the inventory. However, deleted products may still be visible in customer carts until they go to the payment page, where the updated values are displayed. A quantity or price of 0 indicates that the product is no longer available.
3. To edit a product's quantity or price, the admin inputs the product id and the new quantity or price. If there are no errors, the product is edited and an appropriate message is displayed. Like with product deletion, the updated values may not be immediately seen in customer carts, but will be displayed on the payment page.
4. To display the inventory, the `getinv` function is called, which retrieves the inventory from the server and displays it to the admin.
5. To exit the admin menu, the loop is simply broken. When adding a new product, the admin inputs the product id, name, quantity in stock, and price. This information is then sent to the server to add the product to the inventory. If there are any errors, an appropriate error message is displayed.

1. The functions `setcustlock()`, `unlock()`, `rdlckprod()`, `wrlckprod()`, and `cartlckoffset()` are used for file locking purposes whenever the data files are accessed. These functions are responsible for locking the corresponding parts of the file, which allows us to access that portion in the function. Here are the descriptions of each function:

- setcustlock(): This sets a mandatory read lock on the entire customers.txt file, which is useful when we want to view the current registered customers with the store.
- rdlockprod(): This sets a mandatory read lock on the entire records.txt file, which is useful when we want to check or display particular products.
- wrlockprod(): This sets a write lock on the current product in order to update it. This function is useful when we want to modify or delete a product.
- cartlockoffset(): This function takes the cartOffset as an argument (which we obtain from the getOffset() function) and locks that particular cart for reading/writing.
- unlock(): This function is used to unlock any given lock.

Moving forward, it is assumed that whenever we access a file, the appropriate lock is implemented and unlocking is done wherever necessary.

2. The getOffset() function is called by many functions in the server code. This function iterates over the customers.txt file to find the cart offset for the given customer ID. If the offset is found, it is returned, otherwise -1 is returned. This offset is used whenever we need to update a user's cart.

3. The prodadd() function is used by the admin to add a product to the inventory. This function takes the product parameters and checks if the product ID is a duplicate or not. If it is, an appropriate message is printed; otherwise, the product is added. A log statement is written to the adminReceipt.txt file to record the addition of the product.

4. The prodlist() function is used to display the products in the inventory. It can be used by both admin and user.

5. The delprod() function is used by the admin to delete a product. It first checks whether the product ID is valid or not. If it is, the product is deleted; otherwise, an appropriate error message is printed. A log statement is written to the adminReceipt.txt file to record the deletion of the product.

6. The produpdate() function is used to update the price or quantity of a product in the inventory. If 0 is passed as the quantity, the deleteProduct() function is called. Otherwise, the corresponding update is performed based on the ch argument passed into the function. When ch=1, the price is updated; otherwise, the quantity is updated. A log statement is written to the adminReceipt.txt file to record the update of the product.

7. The addcust() function adds a new customer with an auto-generated customer ID. The customer ID is generated by calculating the max of previous customer IDs plus 1.


8. The viewCart() function takes the customer ID as an argument and calls the getOffset() function to obtain the cart offset. If the cart offset is -1 (invalid), an appropriate message is printed. Otherwise, the correct cart is obtained and written to the client.

9. The addprodcart() function is used to add a product to a cart. This function reads the customer ID, product ID, and quantity from the client. If the customer ID is invalid or the product ID already exists in the cart, the corresponding message is printed. It also checks whether the requested quantity is in stock or not. If all of these conditions are satisfied, the product is added to the customer's cart. There is also a limit of MAX PROD on the number of products that can be in a cart

Locking of files-

1. In order to locate the customer id within the customers.txt file, we implement a mandatory read lock that covers the entire file.
2. After obtaining the cart offset for a given customer, we apply record locking to that specific offset within the orders.txt file to lock the cart for reading or writing.
3. To search for a product id within the product list or display the inventory, we impose a mandatory read lock that encompasses all products.
4. Upon updating a specific product, we release the read lock on the products and obtain a write lock on that product to perform the necessary modifications.

Screenshots-



The screenshot displays a code editor with a file named `Server.c` open. The code is in C and shows a function `editprodcart(int, int, int, int)`. The code is as follows:

```
296
297     int offset = getOffset(cust_id, fd_custs);
298     struct cart c;
299
300     if (offset == -1){
301
302         struct cart c;
303         c.custid = -1;
304         write(new_fd, &c, sizeof(struct cart));
305     }else{
306         struct cart c;
307         struct flock lock_cart;
```

Below the code editor, there is a terminal window showing the compilation and execution of the program. The terminal output is as follows:

```
cd "/Users/shreyangupta/Downloads/OnlineStore-main 2/" && gcc Server.c -o Server && "/Users/shreyangupta/Downloads/OnlineStore-main 2/"Server
(base) shreyangupta@Shreyans-MacBook-Air ~ % cd "/Users/shreyangupta/Downloads/OnlineStore-main 2/" && gcc Server.c -o Server && "/Users/shreyangupta/Downloads/OnlineStore-main 2/"Server
Server.c:659:62: warning: passing 'int *' to parameter of type 'socklen_t *' (aka 'unsigned int *') converts between pointers to integer types with different sign [-Wpointer-sign]
    int new_fd = accept(sockfd, (struct sockaddr *)&cli, &size);
                                                             ^
/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/include/sys/socket.h:724:73: note: passing argument to parameter here
int     accept(int, struct sockaddr * __restrict, socklen_t * __restrict)
                                                                    ^
1 warning generated.
Setting up server
Server set up successfully
```

The screenshot shows the VS Code interface with the Explorer sidebar on the left displaying a project named 'ONLINESTORE-MAIN 2'. The Explorer contains files: adminReceipt.txt, Client, Client.c (selected), customers.txt, orders.txt, Project_Report.pdf, README.md, receipt.txt, records.txt, Server, and Server.c. The main editor displays the code for Client.c, which includes a takecustid() function. The code is as follows:

```
Client.c > takecustid()
90     write(sockfd, &c, sizeof(struct cart));
91
92 }
93
94 //input functions
95 int takecustid(){
96     int custId = -1;
97     while (1){
98         printf("Enter customer id\n");
99         scanf("%d", &custId);
100
101         if (custId < 0){
102             printf("try again, id cannot be negative\n");
103         }
104     }
105 }
```

The bottom panel shows the TERMINAL tab with the following output:

```
cd "/Users/shreyangupta/Downloads/OnlineStore-main 2/" && gcc Client.c -o Client && "/Users/shreyangupta/Downloads/OnlineStore-main 2/"Client
(base) shreyangupta@Shreyans-MacBook-Air OnlineStore-main 2 % cd "/Users/shreyangupta/Downloads/OnlineStore-main 2/" && gcc Client.c -o Client && "/Users/shreyangupta/Downloads/OnlineStore-main 2/"Client
Establishing connection to server
Success
Are you user / admin? 1 - user, 2 - admin
s1
Exiting program
(base) shreyangupta@Shreyans-MacBook-Air OnlineStore-main 2 %
```

The screenshot shows the VS Code interface with the Explorer sidebar on the left displaying a project named 'OnlineStore-main 2'. The Explorer contains files: Server.c (selected). The main editor displays the code for Server.c, which includes a editprodcard function. The code is as follows:

```
Server.c > editprodcard(int, int, int, int)
296
297     int offset = getOffset(cust_id, fd_custs);
298     struct cart c;
299
300     if (offset == -1){
301
302         struct cart c;
303         c.custid = -1;
304         write(new_fd, &c, sizeof(struct cart));
305
306     }else{
307         struct cart c;
308         struct flock lock_cart;
```

The bottom panel shows the TERMINAL tab with the following output:

```
cd "/Users/shreyangupta/Downloads/OnlineStore-main 2/" && gcc Server.c -o Server && "/Users/shreyangupta/Downloads/OnlineStore-main 2/"Server
(base) shreyangupta@Shreyans-MacBook-Air ~ % cd "/Users/shreyangupta/Downloads/OnlineStore-main 2/" && gcc Server.c -o Server && "/Users/shreyangupta/Downloads/OnlineStore-main 2/"Server
Server.c:659:62: warning: passing 'int *' to parameter of type 'socklen_t *' (aka 'unsigned int *') converts between pointers to integer types with different sign [-Wpointer-sign]
    int new_fd = accept(sockfd, (struct sockaddr *)&cli, &size);
                                                              ^
/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/include/sys/socket.h:724:73: note: passing argument to parameter here
    int accept(int, struct sockaddr * __restrict, socklen_t * __restrict)
                                                                    ^
1 warning generated.
Setting up server
Server set up successfully
Connection with client successful
[]
```

EXPLORER

- ONLINESTORE-MAIN 2
 - adminReceipt.txt
 - Client
 - Client.c
 - customers.txt
 - orders.txt
 - Project_Report.pdf
 - README.md
 - receipt.txt
 - records.txt
 - Server
 - Server.c

Client.c > takecustid()

```
90     write(sockfd, &c, sizeof(struct cart));
91
92 }
93
94 //input functions
95 int takecustid(){
96     int custId = -1;
97     while (1){
98         printf("Enter customer id\n");
99         scanf("%d", &custId);
100
101         if (custId < 0){
102             printf("try again, id cannot be negative\n");
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Code + -

Establishing connection to server
Success
Are you user / admin? 1 - user, 2 - admin
1

```
-----Menu to choose from-----
|a| To exit
|b| To see all the products
|c| View your cart
|d| To add products to your cart
|e| To edit an existing product in your cart
|f| To proceed for payment
|g| To register a new customer
Please enter your choice
```

> OUTLINE
> TIMELINE

Client.c — OnlineStore-main 2

EXPLORER

- ONLINESTORE-MAIN 2
 - adminReceipt.txt
 - Client
 - Client.c
 - customers.txt
 - orders.txt
 - Project_Report.pdf
 - README.md
 - receipt.txt
 - records.txt
 - Server
 - Server.c

Client.c > takecustid()

```
90     write(sockfd, &c, sizeof(struct cart));
91
92 }
93
94 //input functions
95 int takecustid(){
96     int custId = -1;
97     while (1){
98         printf("Enter customer id\n");
99         scanf("%d", &custId);
100
101         if (custId < 0){
102             printf("try again, id cannot be negative\n");
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Code + -

shreyangupta/Downloads/OnlineStore-main 2/"Client
Establishing connection to server
Success
Are you user / admin? 1 - user, 2 - admin
2

```
-----Menu to choose from-----
|a| To add a product
|b| To delete a product
|c| To update the price of an existing product
|d| To update the quantity of an existing product
|e| To see your inventory
|f| To exit the program
Please enter your choice
```

> OUTLINE
> TIMELINE


```

[d] To update the quantity of an existing product
[e] To see your inventory
[f] To exit the program
Please enter your choice
1

a
Enter product name
1
Enter product id
1
Enter quantity
1
Enter price
1
Duplicate product

-----Menu to choose from-----
[a] To add a product
[b] To delete a product
[c] To update the price of an existing product
[d] To update the quantity of an existing product
[e] To see your inventory
[f] To exit the program
Please enter your choice

b
Enter product id
1
Delete successful

-----Menu to choose from-----
[a] To add a product
[b] To delete a product
[c] To update the price of an existing product
[d] To update the quantity of an existing product
[e] To see your inventory
[f] To exit the program
Please enter your choice

c
Enter product id
2
Enter price
1
Product id invalid

-----Menu to choose from-----
[a] To add a product
[b] To delete a product
[c] To update the price of an existing product
[d] To update the quantity of an existing product
[e] To see your inventory
[f] To exit the program
Please enter your choice

```

```

-----Menu to choose from-----
[a] To add a product
[b] To delete a product
[c] To update the price of an existing product
[d] To update the quantity of an existing product
[e] To see your inventory
[f] To exit the program
Please enter your choice

d
Enter product id
3
Enter quantity
3
Quantity modified successfully

-----Menu to choose from-----
[a] To add a product
[b] To delete a product
[c] To update the price of an existing product
[d] To update the quantity of an existing product
[e] To see your inventory
[f] To exit the program
Please enter your choice

e
Fetching data
ProductID      ProductName      QuantityInStock Price
3              3              4
-----Menu to choose from-----
[a] To add a product
[b] To delete a product
[c] To update the price of an existing product
[d] To update the quantity of an existing product
[e] To see your inventory
[f] To exit the program
Please enter your choice

f
Exiting program
(base) shreyangupta@Shreyans-MacBook-Air OnlineStore-main 2 %

```

Ln 107, Col 19 Spaces: 4 UTF-8 LF () C Mac

Shreyan gupta
IMT2021039