

# **OS MINI PROJECT -REPORT**

**BY: NAME:YASH GUPTA(Yash.Gupta514@iiitb.ac.in)**

**ROLL NO:IMT2021514**

Description:This is a server-client based project where the server and client communicates through socket programming.On client side there are two types of users:  
1.Admin(super user) 2.Normal user  
Admin can add,delete or update products.  
Whereas,normal user can buy products,view its carts etc.

Total 7 files .

- 1.server.c-contains server side code
- 2.client.c-contains client side code
- 3.CUSTOMERS.txt
- 4.ORDERS.txt
- 5.REC.txt
- 6.receipt.txt-contains receipt of normal user
- 7.Areceipt.txt-contais all doing of admin

## PROCEDURE TO RUN THE CODE:

use two terminals:

on 1<sup>st</sup> terminal:

type: gcc server.c -o server

on 2<sup>nd</sup> terminal:

type: gcc client.c -o client

then again on 1<sup>st</sup> terminal:

type: ./server

again on second terminal:

type: ./client

## Admin functions:

1. Add/Delete a product
2. Update the quantity/price of an existing product
3. Display all the products in the format: P\_ID, P\_Name, Cost

## User functions:

1. Display all the products in the format: P\_ID, P\_Name, Cost
2. Add product to cart, it can buy multiple products
2. Display its Cart.
3. Edit its cart.
4. Payment of products in cart

### **Functions on client side:**

*sd-socket file descriptor of socket formed on client side*

`void loginadmin(int sd);` WHEN admin logs in, this function comes into play. This function gives menu to admin and performs functions of admin like adding a product, deleting or updating a product.

`void loginuser(int sd);` When normal user login, this function comes into play. This function gives menu to user and performs functions like adding product to cart, viewing cart of user, editing cart of user, doing payment of items in cart etc.

`void printProduct(struct Product p);` - This function prints product list with their ids, names, prices and quantities

## Functions on server side:

```
void setLockCust(int fd_custs, struct flock lock_cust); //setting locks on customers list
```

```
void unlock(int fd, struct flock lock);
```

```
void productReadLock(int fd, struct flock lock); //puts read lock on products
```

```
void productWriteLock(int fd, struct flock lock); //puts write lock on products
```

These functions puts locks on product ,customer list.

```
void listProducts(int fd, int new_fd); //lists the products
```

```
void addCustomer(int fd_cart, int fd_custs, int new_fd); //adding customer to customer list
```

```
void viewCart(int fd_cart, int new_fd, int fd_custs); //fcn to view cart for user
```

```
void addProductToCart(int fd, int fd_cart, int fd_custs, int new_fd); //adding product to the cart of user
```

```
void editProductInCart(int fd, int fd_cart, int fd_custs, int new_fd); //editing cart of user
```

```
void payment(int fd, int fd_cart, int fd_custs, int new_fd); //function performing payment for user
```

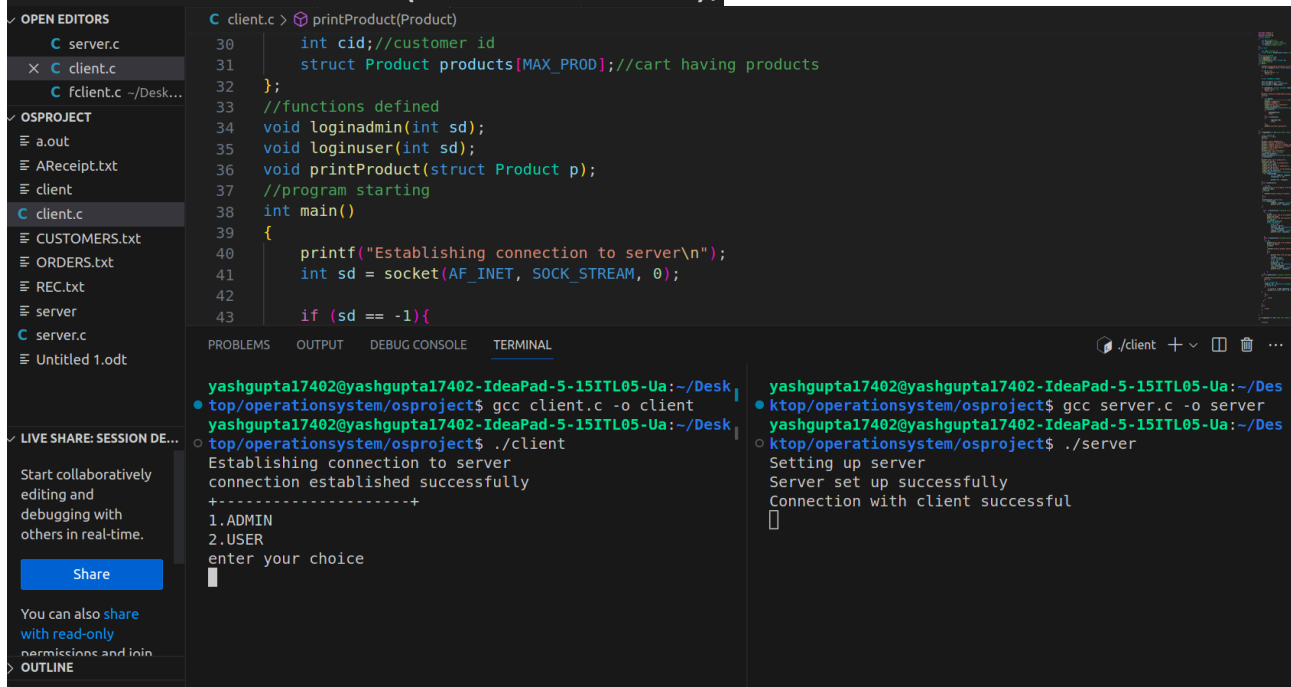
```
void generateAdminReceipt(int fd_admin, int fd); //function to generate admin receipt
```

```
void addProducts(int fd, int new_fd, int fd_admin); //function to add product
```

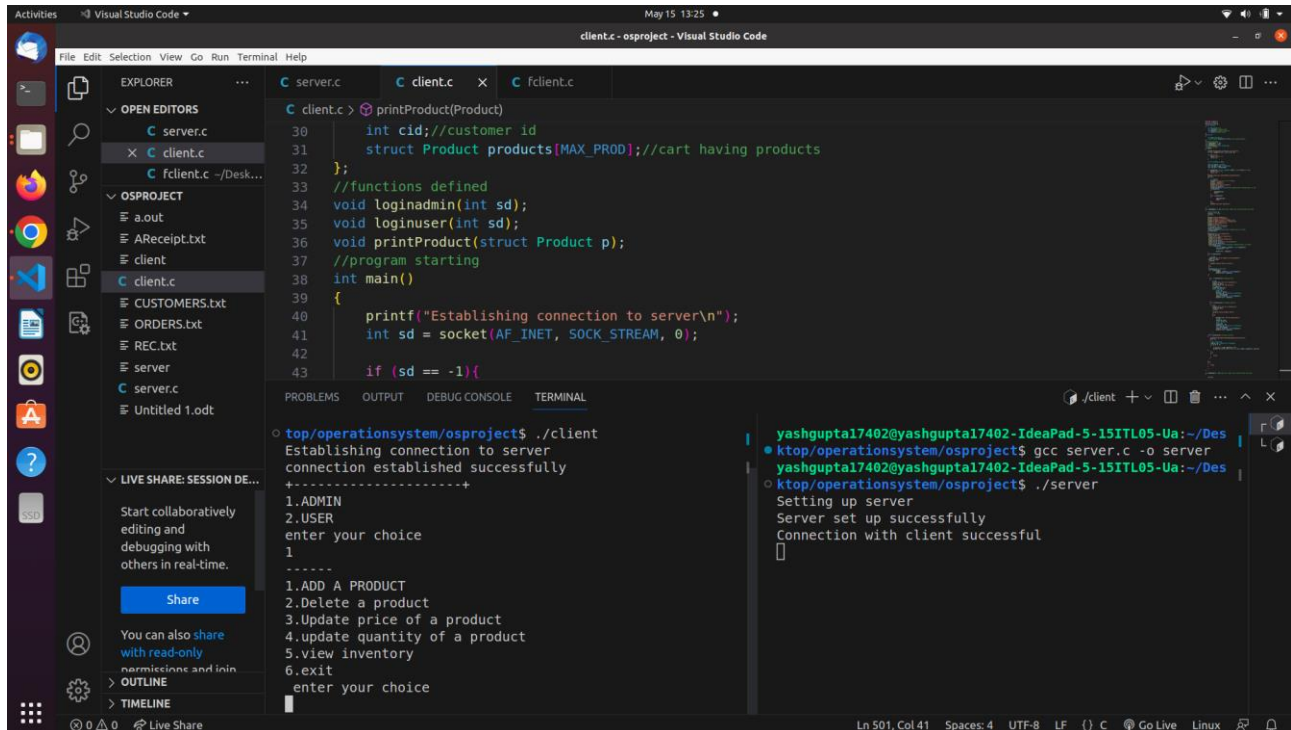
```
void deleteProduct(int fd, int new_fd, int id, int fd_admin); //function to delete product
```

```
void updateProduct(int fd, int new_fd, int ch, int fd_admin); //function to change price/quantity of product
```

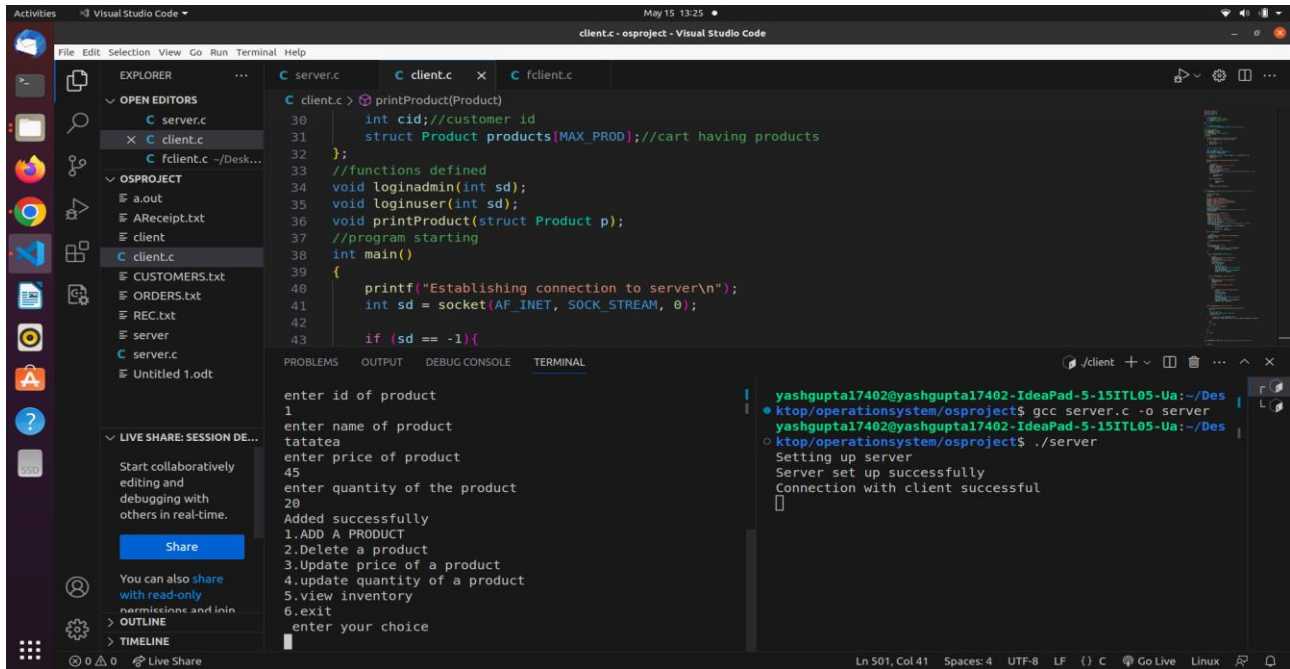
## WORKING OF CODE (SCREENSHOTS);



## Admin's menu:



# Adding of product by admin:



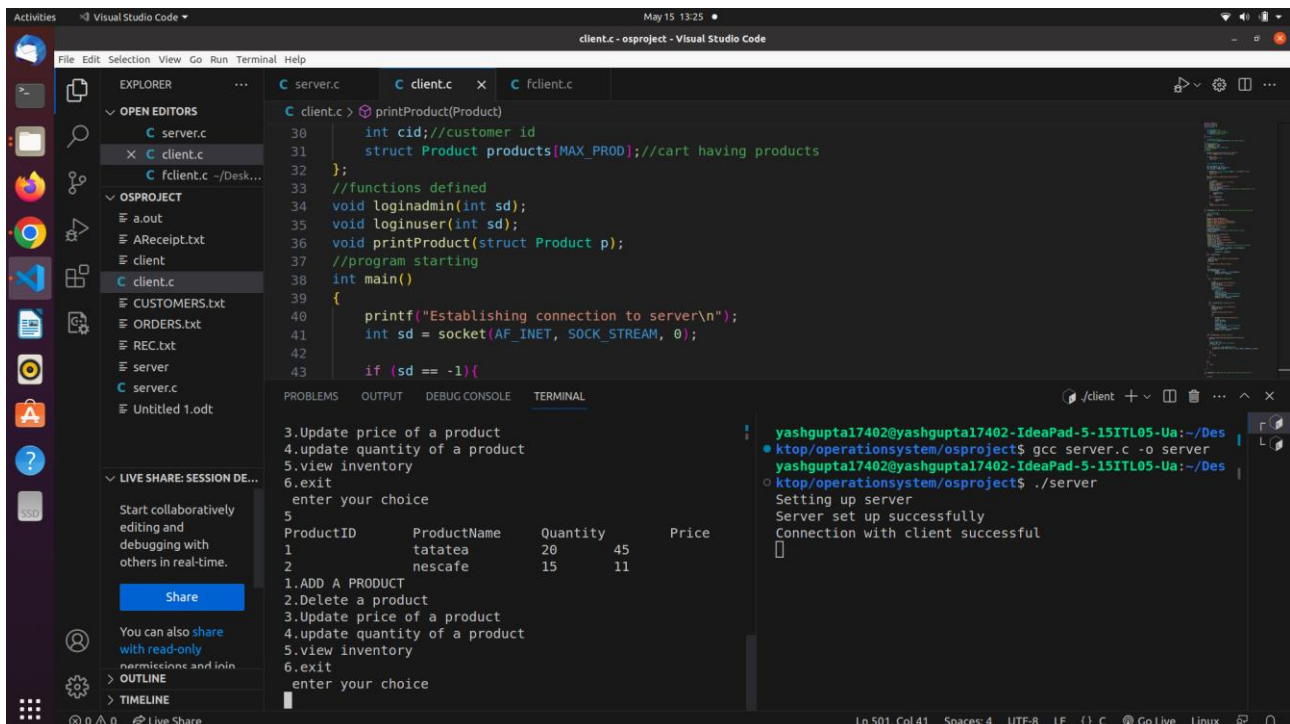
The screenshot shows the Visual Studio Code interface with the `client.c` file open. The code defines a `Product` struct and functions for login and printing products. The terminal shows the execution of the server and client programs, with the client adding a product.

```
client.c: printProduct(Product)
30 int cid;//customer id
31 struct Product products[MAX_PROD];//cart having products
32 };
33 //functions defined
34 void loginadmin(int sd);
35 void loginuser(int sd);
36 void printProduct(struct Product p);
37 //program starting
38 int main()
39 {
40     printf("Establishing connection to server\n");
41     int sd = socket(AF_INET, SOCK_STREAM, 0);
42
43     if (sd == -1){
44         perror("socket creation failed");
45         return -1;
46     }
47
48     while(1)
49     {
50         printf("enter id of product\n");
51         int id;
52         scanf("%d", &id);
53         printf("enter name of product\n");
54         char name[50];
55         scanf("%s", name);
56         printf("enter price of product\n");
57         float price;
58         scanf("%f", &price);
59         printf("enter quantity of the product\n");
60         int quantity;
61         scanf("%d", &quantity);
62         Product p;
63         p.cid = id;
64         strcpy(p.name, name);
65         p.price = price;
66         p.quantity = quantity;
67         products[id] = p;
68         printf("Added successfully\n");
69         printf("1.ADD A PRODUCT\n");
70         printf("2.Delete a product\n");
71         printf("3.Update price of a product\n");
72         printf("4.update quantity of a product\n");
73         printf("5.view inventory\n");
74         printf("6.exit\n");
75         printf("enter your choice\n");
76         int choice;
77         scanf("%d", &choice);
78         if(choice == 1)
79             printProduct(products[id]);
80         else if(choice == 2)
81             deleteProduct(id);
82         else if(choice == 3)
83             updatePrice(id, price);
84         else if(choice == 4)
85             updateQuantity(id, quantity);
86         else if(choice == 5)
87             viewInventory();
88         else if(choice == 6)
89             break;
90     }
91 }
```

```
enter id of product
1
enter name of product
tatatea
enter price of product
45
enter quantity of the product
20
Added successfully
1.ADD A PRODUCT
2.Delete a product
3.Update price of a product
4.update quantity of a product
5.view inventory
6.exit
enter your choice
```

```
yashguptal7402@yashguptal7402-IdeaPad-5-15ITL05-Ua:~/Des
ktop/operationssystem/osproject$ gcc server.c -o server
yashguptal7402@yashguptal7402-IdeaPad-5-15ITL05-Ua:~/Des
ktop/operationssystem/osproject$ ./server
Setting up server
Server set up successfully
Connection with client successful
```

# Viewing inventory:



The screenshot shows the Visual Studio Code interface with the `client.c` file open. The code defines a `Product` struct and functions for login and printing products. The terminal shows the execution of the server and client programs, with the client viewing the inventory.

```
client.c: printProduct(Product)
30 int cid;//customer id
31 struct Product products[MAX_PROD];//cart having products
32 };
33 //functions defined
34 void loginadmin(int sd);
35 void loginuser(int sd);
36 void printProduct(struct Product p);
37 //program starting
38 int main()
39 {
40     printf("Establishing connection to server\n");
41     int sd = socket(AF_INET, SOCK_STREAM, 0);
42
43     if (sd == -1){
44         perror("socket creation failed");
45         return -1;
46     }
47
48     while(1)
49     {
50         printf("enter id of product\n");
51         int id;
52         scanf("%d", &id);
53         printf("enter name of product\n");
54         char name[50];
55         scanf("%s", name);
56         printf("enter price of product\n");
57         float price;
58         scanf("%f", &price);
59         printf("enter quantity of the product\n");
60         int quantity;
61         scanf("%d", &quantity);
62         Product p;
63         p.cid = id;
64         strcpy(p.name, name);
65         p.price = price;
66         p.quantity = quantity;
67         products[id] = p;
68         printf("Added successfully\n");
69         printf("1.ADD A PRODUCT\n");
70         printf("2.Delete a product\n");
71         printf("3.Update price of a product\n");
72         printf("4.update quantity of a product\n");
73         printf("5.view inventory\n");
74         printf("6.exit\n");
75         printf("enter your choice\n");
76         int choice;
77         scanf("%d", &choice);
78         if(choice == 1)
79             printProduct(products[id]);
80         else if(choice == 2)
81             deleteProduct(id);
82         else if(choice == 3)
83             updatePrice(id, price);
84         else if(choice == 4)
85             updateQuantity(id, quantity);
86         else if(choice == 5)
87             viewInventory();
88         else if(choice == 6)
89             break;
90     }
91 }
```

```
3.Update price of a product
4.update quantity of a product
5.view inventory
6.exit
enter your choice
5
ProductID      ProductName    Quantity    Price
1              tatatea        20          45
2              nescafe        15          11
1.ADD A PRODUCT
2.Delete a product
3.Update price of a product
4.update quantity of a product
5.view inventory
6.exit
enter your choice
```

```
yashguptal7402@yashguptal7402-IdeaPad-5-15ITL05-Ua:~/Des
ktop/operationssystem/osproject$ gcc server.c -o server
yashguptal7402@yashguptal7402-IdeaPad-5-15ITL05-Ua:~/Des
ktop/operationssystem/osproject$ ./server
Setting up server
Server set up successfully
Connection with client successful
```

## Changing price of product with id=2

The screenshot shows the Visual Studio Code editor with the `client.c` file open. The code defines a `Product` struct and a `printProduct` function. The `main` function handles user input for changing the price of a product with ID 2. The terminal output shows the program running and the user entering the product ID and the new price.

```
client.c: printProduct(Product)
30 int cid;//customer id
31 struct Product products[MAX_PROD];//cart having products
32 };
33 //functions defined
34 void loginadmin(int sd);
35 void loginuser(int sd);
36 void printProduct(struct Product p);
37 //program starting
38 int main()
39 {
40     printf("Establishing connection to server\n");
41     int sd = socket(AF_INET, SOCK_STREAM, 0);
42
43     if (sd == -1){
44         //error handling
45     }
46
47     while(1)
48     {
49         enter your choice
50         5
51         ProductID      ProductName      Quantity      Price
52         1               tatatea          20            45
53         2               nescafe          15            11
54
55         1.ADD A PRODUCT
56         2.Delete a product
57         3.Update price of a product
58         4.Update quantity of a product
59         5.view inventory
60         6.exit
61
62         enter your choice
63         3
64         enter the id of product
65         2
66         enter the new price
67         10
68     }
69 }
```

Terminal Output:

```
yashgupta17402@yashgupta17402-IdeaPad-5-151TL05-Ua:~/Des
ktop/operationssystem/osproject$ gcc server.c -o server
yashgupta17402@yashgupta17402-IdeaPad-5-151TL05-Ua:~/Des
ktop/operationssystem/osproject$ ./server
Setting up server
Server set up successfully
Connection with client successful
[]
```

## User's menu:

The screenshot shows the Visual Studio Code editor with the `client.c` file open. The code defines a `Product` struct and a `printProduct` function. The `main` function handles user input for the user's menu. The terminal output shows the program running and the user entering the menu options.

```
client.c: printProduct(Product)
30 int cid;//customer id
31 struct Product products[MAX_PROD];//cart having products
32 };
33 //functions defined
34 void loginadmin(int sd);
35 void loginuser(int sd);
36 void printProduct(struct Product p);
37 //program starting
38 int main()
39 {
40     printf("Establishing connection to server\n");
41     int sd = socket(AF_INET, SOCK_STREAM, 0);
42
43     if (sd == -1){
44         //error handling
45     }
46
47     while(1)
48     {
49         connection established successfully
50         +-----+
51         1. ADMIN
52         2. USER
53         enter your choice
54         2
55         ---
56         Menu:
57         1. To see all the products available
58         2. To see your cart
59         3. To add products to your cart
60         4. To edit an existing product in your cart
61         5. To proceed for payment
62         6. To register a new customer
63         7. To exit the menu
64         Please enter your choice
65         1
66     }
67 }
```

Terminal Output:

```
yashgupta17402@yashgupta17402-IdeaPad-5-151TL05-Ua:~/Des
ktop/operationssystem/osproject$ gcc server.c -o server
yashgupta17402@yashgupta17402-IdeaPad-5-151TL05-Ua:~/Des
ktop/operationssystem/osproject$ ./server
Setting up server
Server set up successfully
Connection with client successful
Connection terminated
Connection with client successful
[]
```



The screenshot shows a Visual Studio Code editor window titled "client.c - osproject - Visual Studio Code". The editor displays a C program for a client-server application. The Explorer sidebar on the left shows the project structure with files like "server.c", "client.c", and "fclient.c". The Output window at the bottom shows the program's execution, including menu options and successful connections. The status bar at the bottom indicates the file is at line 501, column 41, with 4 spaces, UTF-8 encoding, and LF line endings.

The C code in the editor is as follows:

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <string.h>
5  #include <sys/socket.h>
6  #include <sys/types.h>
7  #include <pthread.h>
8  #include <time.h>
9  #include <ctype.h>
10 #include <math.h>
11 #include <limits.h>
12 #include <stdbool.h>
13 #include <stdint.h>
14 #include <inttypes.h>
15 #include <float.h>
16 #include <double.h>
17 #include <complex.h>
18 #include <wchar.h>
19 #include <wctype.h>
20 #include <locale.h>
21 #include <iconv.h>
22 #include <fcntl.h>
23 #include <poll.h>
24 #include <sys/stat.h>
25 #include <sys/time.h>
26 #include <sys/uio.h>
27 #include <sys/xattr.h>
28 #include <unistd.h>
29 #include <sys/types.h>
30 #include <sys/socket.h>
31 #include <sys/types.h>
32 #include <unistd.h>
33 #include <string.h>
34 #include <stdio.h>
35 #include <stdlib.h>
36 #include <unistd.h>
37 #include <string.h>
38 #include <sys/socket.h>
39 #include <sys/types.h>
40 #include <unistd.h>
41 #include <string.h>
42 #include <stdio.h>
43 #include <stdlib.h>
44 #include <unistd.h>
45 #include <string.h>
46 #include <sys/socket.h>
47 #include <sys/types.h>
48 #include <unistd.h>
49 #include <string.h>
50 #include <stdio.h>
51 #include <stdlib.h>
52 #include <unistd.h>
53 #include <string.h>
54 #include <sys/socket.h>
55 #include <sys/types.h>
56 #include <unistd.h>
57 #include <string.h>
58 #include <stdio.h>
59 #include <stdlib.h>
60 #include <unistd.h>
61 #include <string.h>
62 #include <sys/socket.h>
63 #include <sys/types.h>
64 #include <unistd.h>
65 #include <string.h>
66 #include <stdio.h>
67 #include <stdlib.h>
68 #include <unistd.h>
69 #include <string.h>
70 #include <sys/socket.h>
71 #include <sys/types.h>
72 #include <unistd.h>
73 #include <string.h>
74 #include <stdio.h>
75 #include <stdlib.h>
76 #include <unistd.h>
77 #include <string.h>
78 #include <sys/socket.h>
79 #include <sys/types.h>
80 #include <unistd.h>
81 #include <string.h>
82 #include <stdio.h>
83 #include <stdlib.h>
84 #include <unistd.h>
85 #include <string.h>
86 #include <sys/socket.h>
87 #include <sys/types.h>
88 #include <unistd.h>
89 #include <string.h>
90 #include <stdio.h>
91 #include <stdlib.h>
92 #include <unistd.h>
93 #include <string.h>
94 #include <sys/socket.h>
95 #include <sys/types.h>
96 #include <unistd.h>
97 #include <string.h>
98 #include <stdio.h>
99 #include <stdlib.h>
100 #include <unistd.h>
101 #include <string.h>
102 #include <sys/socket.h>
103 #include <sys/types.h>
104 #include <unistd.h>
105 #include <string.h>
106 #include <stdio.h>
107 #include <stdlib.h>
108 #include <unistd.h>
109 #include <string.h>
110 #include <sys/socket.h>
111 #include <sys/types.h>
112 #include <unistd.h>
113 #include <string.h>
114 #include <stdio.h>
115 #include <stdlib.h>
116 #include <unistd.h>
117 #include <string.h>
118 #include <sys/socket.h>
119 #include <sys/types.h>
120 #include <unistd.h>
121 #include <string.h>
122 #include <stdio.h>
123 #include <stdlib.h>
124 #include <unistd.h>
125 #include <string.h>
126 #include <sys/socket.h>
127 #include <sys/types.h>
128 #include <unistd.h>
129 #include <string.h>
130 #include <stdio.h>
131 #include <stdlib.h>
132 #include <unistd.h>
133 #include <string.h>
134 #include <sys/socket.h>
135 #include <sys/types.h>
136 #include <unistd.h>
137 #include <string.h>
138 #include <stdio.h>
139 #include <stdlib.h>
140 #include <unistd.h>
141 #include <string.h>
142 #include <sys/socket.h>
143 #include <sys/types.h>
144 #include <unistd.h>
145 #include <string.h>
146 #include <stdio.h>
147 #include <stdlib.h>
148 #include <unistd.h>
149 #include <string.h>
150 #include <sys/socket.h>
151 #include <sys/types.h>
152 #include <unistd.h>
153 #include <string.h>
154 #include <stdio.h>
155 #include <stdlib.h>
156 #include <unistd.h>
157 #include <string.h>
158 #include <sys/socket.h>
159 #include <sys/types.h>
160 #include <unistd.h>
161 #include <string.h>
162 #include <stdio.h>
163 #include <stdlib.h>
164 #include <unistd.h>
165 #include <string.h>
166 #include <sys/socket.h>
167 #include <sys/types.h>
168 #include <unistd.h>
169 #include <string.h>
170 #include <stdio.h>
171 #include <stdlib.h>
172 #include <unistd.h>
173 #include <string.h>
174 #include <sys/socket.h>
175 #include <sys/types.h>
176 #include <unistd.h>
177 #include <string.h>
178 #include <stdio.h>
179 #include <stdlib.h>
180 #include <unistd.h>
181 #include <string.h>
182 #include <sys/socket.h>
183 #include <sys/types.h>
184 #include <unistd.h>
185 #include <string.h>
186 #include <stdio.h>
187 #include <stdlib.h>
188 #include <unistd.h>
189 #include <string.h>
190 #include <sys/socket.h>
191 #include <sys/types.h>
192 #include <unistd.h>
193 #include <string.h>
194 #include <stdio.h>
195 #include <stdlib.h>
196 #include <unistd.h>
197 #include <string.h>
198 #include <sys/socket.h>
199 #include <sys/types.h>
200 #include <unistd.h>
201 #include <string.h>
202 #include <stdio.h>
203 #include <stdlib.h>
204 #include <unistd.h>
205 #include <string.h>
206 #include <sys/socket.h>
207 #include <sys/types.h>
208 #include <unistd.h>
209 #include <string.h>
210 #include <stdio.h>
211 #include <stdlib.h>
212 #include <unistd.h>
213 #include <string.h>
214 #include <sys/socket.h>
215 #include <sys/types.h>
216 #include <unistd.h>
217 #include <string.h>
218 #include <stdio.h>
219 #include <stdlib.h>
220 #include <unistd.h>
221 #include <string.h>
222 #include <sys/socket.h>
223 #include <sys/types.h>
224 #include <unistd.h>
225 #include <string.h>
226 #include <stdio.h>
227 #include <stdlib.h>
228 #include <unistd.h>
229 #include <string.h>
230 #include <sys/socket.h>
231 #include <sys/types.h>
232 #include <unistd.h>
233 #include <string.h>
234 #include <stdio.h>
235 #include <stdlib.h>
236 #include <unistd.h>
237 #include <string.h>
238 #include <sys/socket.h>
239 #include <sys/types.h>
240 #include <unistd.h>
241 #include <string.h>
242 #include <stdio.h>
243 #include <stdlib.h>
244 #include <unistd.h>
245 #include <string.h>
246 #include <sys/socket.h>
247 #include <sys/types.h>
248 #include <unistd.h>
249 #include <string.h>
250 #include <stdio.h>
251 #include <stdlib.h>
252 #include <unistd.h>
253 #include <string.h>
254 #include <sys/socket.h>
255 #include <sys/types.h>
256 #include <unistd.h>
257 #include <string.h>
258 #include <stdio.h>
259 #include <stdlib.h>
260 #include <unistd.h>
261 #include <string.h>
262 #include <sys/socket.h>
263 #include <sys/types.h>
264 #include <unistd.h>
265 #include <string.h>
266 #include <stdio.h>
267 #include <stdlib.h>
268 #include <unistd.h>
269 #include <string.h>
270 #include <sys/socket.h>
271 #include <sys/types.h>
272 #include <unistd.h>
273 #include <string.h>
274 #include <stdio.h>
275 #include <stdlib.h>
276 #include <unistd.h>
277 #include <string.h>
278 #include <sys/socket.h>
279 #include <sys/types.h>
280 #include <unistd.h>
281 #include <string.h>
282 #include <stdio.h>
283 #include <stdlib.h>
284 #include <unistd.h>
285 #include <string.h>
286 #include <sys/socket.h>
287 #include <sys/types.h>
288 #include <unistd.h>
289 #include <string.h>
290 #include <stdio.h>
291 #include <stdlib.h>
292 #include <unistd.h>
293 #include <string.h>
294 #include <sys/socket.h>
295 #include <sys/types.h>
296 #include <unistd.h>
297 #include &
```



# Viewing cart of customer

The screenshot shows the Visual Studio Code interface with the `client.c` file open. The code defines a menu and a function to print the product list. The terminal output shows the program running and displaying the menu and the current cart contents for Customer ID 0.

```
Customer ID 0
ProductID  ProductName  Quantity  Price
1          tatatea     15        45
2          nescafe      5         10
----
Menu:
1. To see all the products available
2. To see your cart
3. To add products to your cart
4. To edit an existing product in your cart
5. To proceed for payment
6. To register a new customer
7. To exit the menu
Please enter your choice
```

# Edited quantity of product (id=2)

The screenshot shows the Visual Studio Code interface with the `client.c` file open. The code defines a menu and a function to print the product list. The terminal output shows the program running and displaying the menu and the current cart contents for Customer ID 0. The quantity of product ID 2 has been updated to 4.

```
Customer ID 0
ProductID  ProductName  Quantity  Price
1          tatatea     15        45
2          nescafe      4         10
----
Menu:
1. To see all the products available
2. To see your cart
3. To add products to your cart
4. To edit an existing product in your cart
5. To proceed for payment
6. To register a new customer
7. To exit the menu
Please enter your choice
```

# Payment:

The screenshot shows the Visual Studio Code interface with the 'client.c' file open. The code defines a menu for payment options. The terminal output shows the user selecting option 6 to register a new customer, entering a customer ID of 0, and then adding items to the cart. The cart contains two items: Product ID 1 (Ordered - 15; Incart - 15; Price - 45) and Product ID 2 (Ordered - 4; Incart - 4; Price - 10). The total in the cart is 715. The user is prompted to enter the amount to pay, but the terminal output is cut off at 715.

```
client.c: printProduct(Product)
30 int cid;//customer id
31 struct Product products[MAX_PROD];//cart having products
32 };
33 //functions defined
34 void loginadmin(int sd);
35 void loginuser(int sd);
36 void printProduct(struct Product p);
37 //program starting
38 int main()
39 {
40     printf("Establishing connection to server\n");
41     int sd = socket(AF_INET, SOCK_STREAM, 0);
42
43     if (sd == -1){
44
45         6. To register a new customer
46         7. To exit the menu
47         Please enter your choice
48         5
49         Enter customer id
50         0
51         Product id- 1
52         Ordered - 15; Incart - 15; Price - 45
53         Product id- 2
54         Ordered - 4; Incart - 4; Price - 10
55         Total in your cart
56         715
57         Please enter the amount to pay
58         712
59         Wrong total entered, enter again
60         Please enter the amount to pay
61         715
```

# Products available left :

The screenshot shows the Visual Studio Code interface with the 'server.c' file open. The code defines a menu for product management. The terminal output shows the user selecting option 1 to see all the products available. The products are listed in a table:

ProductID	ProductName	Quantity	Price
1	tatatea	5	
45			
2	nescafe	11	
10			
----			

The menu options are:

1. To see all the products available
2. To see your cart
3. To add products to your cart
4. To edit an existing product in your cart
5. To proceed for payment
6. To register a new customer
7. To exit the menu

The user is prompted to enter their choice.

```
server.c: printProduct(Product)
30 int cid;//customer id
31 struct Product products[MAX_PROD];//cart having products
32 };
33 //functions defined
34 void loginadmin(int sd);
35 void loginuser(int sd);
36 void printProduct(struct Product p);
37 //program starting
38 int main()
39 {
40     printf("Establishing connection to server\n");
41     int sd = socket(AF_INET, SOCK_STREAM, 0);
42
43     if (sd == -1){
44
45         1 ProductID      ProductName      Quantity      Price
46         1      tatatea           5
47         45
48         2      nescafe          11
49         10
50         ----
51         Menu:
52         1. To see all the products available
53         2. To see your cart
54         3. To add products to your cart
55         4. To edit an existing product in your cart
56         5. To proceed for payment
57         6. To register a new customer
58         7. To exit the menu
59         Please enter your choice
```

# Admin's receipt

The screenshot shows the Visual Studio Code interface with the file explorer on the left. The 'OPEN EDITORS' list shows 'AReceipt.txt' as the active file. The 'EXPLORER' sidebar shows the project structure, including 'server.c', 'client.c', 'AReceipt.txt', 'fclient.c', 'a.out', 'client', 'CUSTOMERS.txt', 'ORDERS.txt', 'REC.txt', 'receipt.txt', 'server', 'server.c', and 'Untitled 1.odt'. The 'LIVE SHARE: SESSION DE...' section is also visible. The main editor area displays the content of 'AReceipt.txt':

```
1 New product with product id 1 added successfully
2 New product with product id 2 added successfully
3 Price of product with product id 2 modified from 11 to 10
4 Current Inventory:
5 ProductID  ProductName  Quantity  Price
6 1          tatatea     20        45
7 2          nescafe     15        10
8
```

The terminal at the bottom shows the command prompt for 'yashgupta17402@yashgupta17402-IdeaPad-5-15ITL05-Ua:~/Desk' and the directory 'top/operationsystem/osproject\$'.

# Order's receipt

The screenshot shows the Visual Studio Code interface with the file explorer on the left. The 'OPEN EDITORS' list shows 'receipt.txt' as the active file. The 'EXPLORER' sidebar shows the project structure, including 'server.c', 'client.c', 'AReceipt.txt', 'fclient.c', 'a.out', 'client', 'CUSTOMERS.txt', 'ORDERS.txt', 'REC.txt', 'receipt.txt', 'server', 'server.c', and 'Untitled 1.odt'. The 'LIVE SHARE: SESSION DE...' section is also visible. The main editor area displays the content of 'receipt.txt':

```
1 ProductID  ProductName  Quantity  Price
2 1          tatatea     15        45
3 2          nescafe     4         10
4 Total - 715
5
```

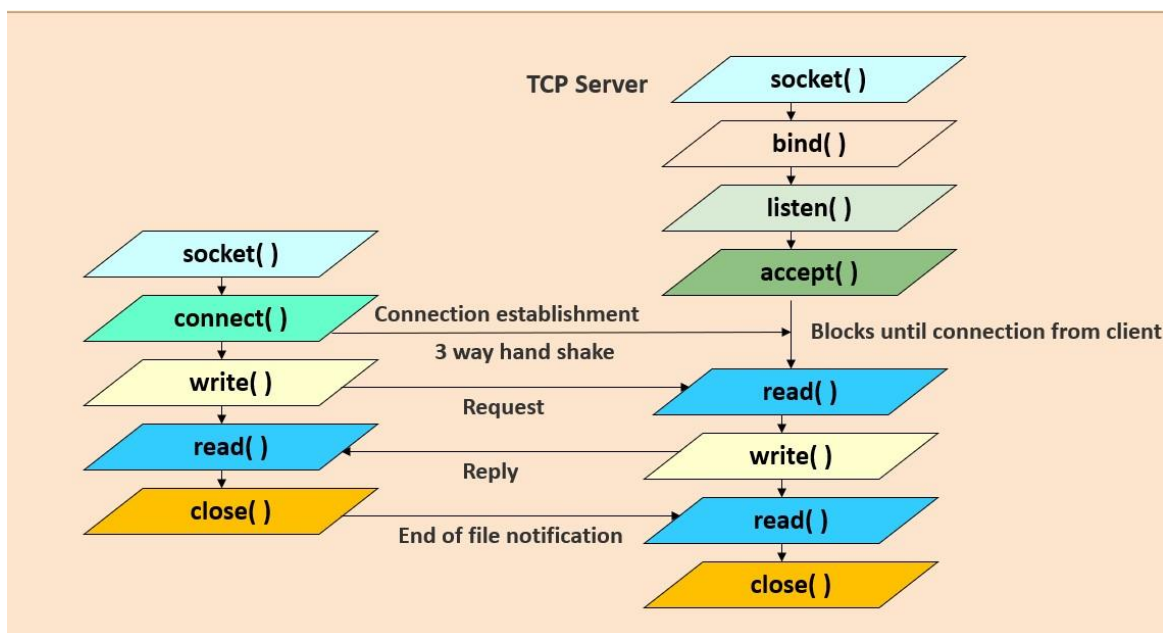
The terminal at the bottom shows the command prompt for 'yashgupta17402@yashgupta17402-IdeaPad-5-15ITL05-Ua:~/Desk' and the directory 'top/operationsystem/osproject\$'.

# CONCEPTS USED:

**1. SOCKET PROGRAMMING:** Socket programming is a way of writing networked applications in which different processes or computers can communicate with each other using the Internet Protocol (IP).

In socket programming, a server process creates a socket and binds it to a specific IP address and port number on the machine. This socket is then set up to listen for incoming connections from client processes. When a client process wants to connect to the server, it creates its own socket and specifies the server's IP address and port number. The server can then accept the connection and create a new socket to handle the communication with that client.

Once a connection is established between a client and server, they can communicate by sending data through their sockets. This data can be in any format, but it is typically sent in small chunks, or packets, to avoid overwhelming the network or the receiving process.



## 2. File Locking:

Mandatory and record-locking techniques are used to manage concurrent access to the data. `fcntl()` is used.

When we try to find the customer id in the `customers.txt` file, we perform a mandatory read lock. Read lock is also used while displaying the inventory. The record write lock is used while adding a new product and deleting a product. The record write lock is used while adding a new customer. Write lock is used in the payment gateway.

fcntl Implementation	flock Implementation
<pre>struct flock lock;  lock.l_type    = F_WRLCK; lock.l_whence  = SEEK_SET; lock.l_start   = nth record; lock.l_len     = sizeof (record); lock.l_pid     = getpid( );  fcntl ( fd, F_SETLKW, &amp;lock );  .....critical section.....  lock.l_type    = F_UNLCK;  fcntl ( fd, F_SETLK, &amp;lock );</pre>	<pre>flock ( fd, LOCK_EX );  .....critical section.....  flock ( fd, LOCK_UN );  -----  flock ( fd, LOCK_SH );  .....critical section.....  flock ( fd, LOCK_UN );</pre>

## 3. File Handling:

Instead of setting up a database, files have been used to store and access data for the ease of implementation.

Hence to store, read and write to the files, file handling is used. Functions like `open()`, `read()`, `write()` are used.

