Project Report – OS

Name – M Srinivasan

Roll No – IMT2021058

Description - This is a terminal level interface that is used for buying and selling products. This is a small-scale mimic of the online store, where there are buyers selling their products from their inventory and customers buying the products sold by these sellers. In this basic version there is only one seller, and he can have multiple customers. Operating systems concepts such as socket programming, file locking area used in this project to make this interaction as close as to the one experienced in real life.

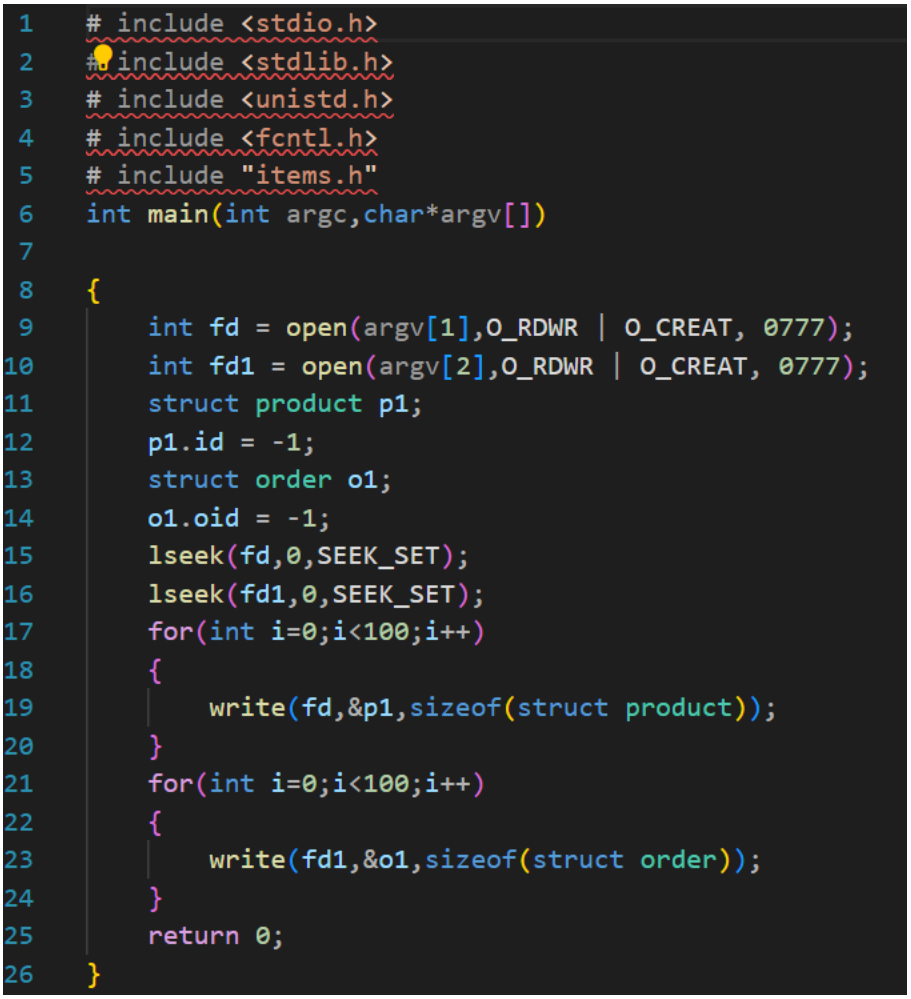
Approach – Here basically we need to make sure that we have a separate list of products that the seller sells and keep track of the quantity left and price of the product. We also need to track orders from each customer with the seller and every order must be independent of other orders.

For maintaining concurrency, I have implemented file locking (record locking mostly) using the FCNTL function, with this we can be sure that changes made by one will be reflected in the other.

Starter Code –

This is one of the simplest codes written in the program. It takes two arguments – the two file names for storing products present in the store and the orders for that store, respectively. It then sets up the initialization conditions for the same.

It initializes the values of the structs (that are to be written in the two files respectively), it sets the value of orderID and productID to be -1 explicitly and then we use the ***read and write sys calls*** to write it to the files. We make use of the ***lseek*** option to move the file descriptor to the correct position (SEEK_SET – start of the file) before we start writing anything in it.

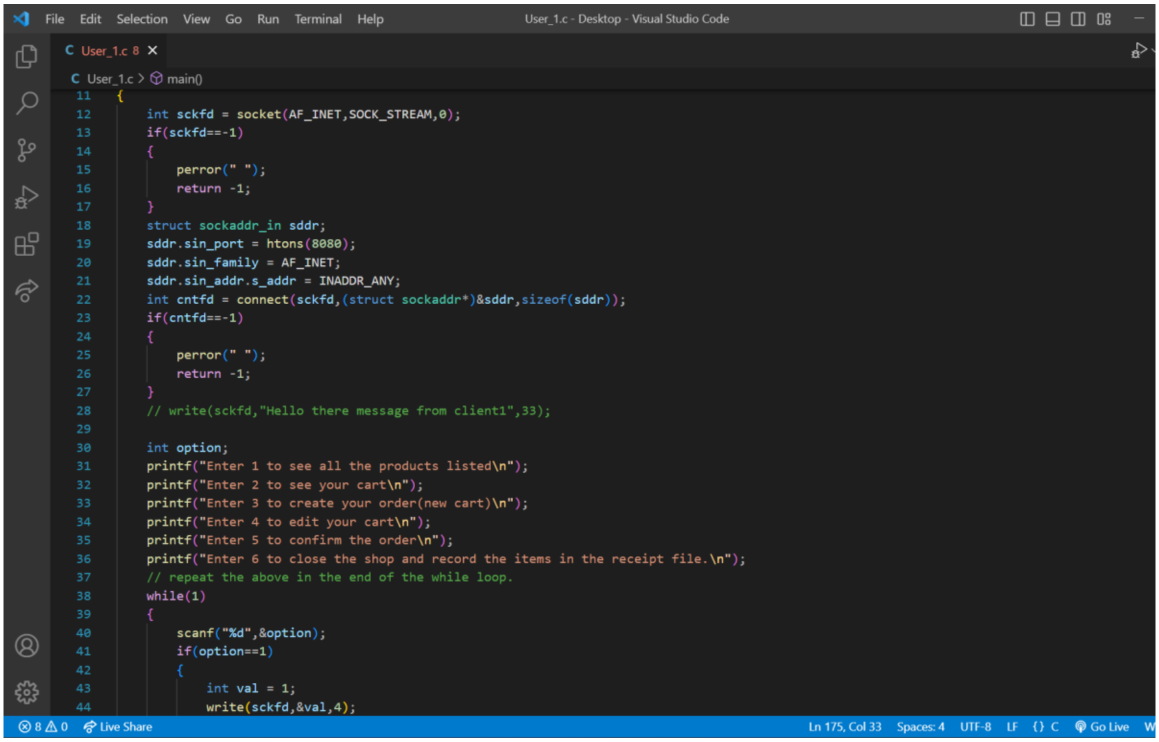


```c
# include <stdio.h>
# include <stdlib.h>
# include <unistd.h>
# include <fcntl.h>
# include "items.h"
int main(int argc,char*argv[])

{
    int fd = open(argv[1],O_RDWR | O_CREAT, 0777);
    int fd1 = open(argv[2],O_RDWR | O_CREAT, 0777);
    struct product p1;
    p1.id = -1;
    struct order o1;
    o1.oid = -1;
    lseek(fd,0,SEEK_SET);
    lseek(fd1,0,SEEK_SET);
    for(int i=0;i<100;i++)
    {
        write(fd,&p1,sizeof(struct product));
    }
    for(int i=0;i<100;i++)
    {
        write(fd1,&o1,sizeof(struct order));
    }
    return 0;
}
```

User Code: This is the user code used to mimic user (customer) options**.** There are diverse options, each of them corresponding to a specific action of the customer.

In this code socket programming is used to connect to the server (the seller) to facilitate two-way communication between the user and seller.



File Edit Selection View Go Run Terminal Help
User_1.c - Desktop - Visual Studio Code
User_1.c 8
User_1.c > main()
8 0 Live Share
Ln 175, Col 33 Spaces: 4 UTF-8 LF {} C Go Live

```c
{
    int sckfd = socket(AF_INET,SOCK_STREAM,0);
    if(sckfd==-1)
    {
        perror(" ");
        return -1;
    }
    struct sockaddr_in sddr;
    sddr.sin_port = htons(8080);
    sddr.sin_family = AF_INET;
    sddr.sin_addr.s_addr = INADDR_ANY;
    int cntfd = connect(sckfd,(struct sockaddr*)&sddr,sizeof(sddr));
    if(cntfd==-1)
    {
        perror(" ");
        return -1;
    }
    // write(sckfd,"Hello there message from client1",33);

    int option;
    printf("Enter 1 to see all the products listed\n");
    printf("Enter 2 to see your cart\n");
    printf("Enter 3 to create your order(new cart)\n");
    printf("Enter 4 to edit your cart\n");
    printf("Enter 5 to confirm the order\n");
    printf("Enter 6 to close the shop and record the items in the receipt file.\n");
    // repeat the above in the end of the while loop.
    while(1)
    {
        scanf("%d",&option);
        if(option==1)
        {
            int val = 1;
            write(sckfd,&val,4);
```

As usual we first get the socket file descriptor and then initialize the struct sockaddr_in .

We then connect the user to the server for communication between the server and the client. I have created a menu-based application for doing the specific actions –

1) A user can see all the products listed.
2) A user can view his cart.
3) A user can create a new cart.
4) A user can edit his cart.
5) A user can confirm his/her order.
6) He can finish shopping and exit.

1) **See All the products listed** – read and write sys calls are used to communicate between the server and the client. We send a message from client to server telling it to display the products it has in its inventory and the server in return sends the info using the read-write sys calls and then the info is displayed in the user's terminal.

2) **A user can view his cart** – For this functionality we also use read-write sys calls, and we send the customerID of the customer to the server and then in return the server sends the products listed in that cart.

3) **A user can create new cart** – For this functionality also we use read-write sys all and the user first enters his/her customerID and then specifies the products needed for placing the order. If there is already a non-empty cart for that customer, then that will be added to that existing cart. The server will do the needful.

4) **A user can edit his cart –** A user can edit the quantity present in the cart (deleting the product involves making the quantity = 0). Adding a new product comes into the function of creating a new cart. This function is used when the user wants to change the quantity of an existing product in the cart.

5) **A user can confirm and place the order (for his cart) -** For this functionality we also use read-write sys calls for sending and receiving information. The user will first send orderID and then the list of all products will be shown to the customer for final viewing before confirming the order. The total price will also be shown to the customer. The customer shall press '1' for confirming the order and placing (analogous to typing the OTP for paying the money), once the server receives this '1' then it will place the order and give the confirmation to the user.

Server/Admin Code –

This is a major part of the code that is responsible for running the server (or the shop) which interacts with multiple customers and then does what is necessary for each of the query it receives. Here **FILE LOCKING, READ-WRITE sys calls, SOCKET Programming, File related sys calls** etc. are used for the functioning of this program. All the orders and products are stored in a file and then read into and modified by the server application whenever necessary. While running the executable for this program, we shall also have to pass the names of the files that store the products present in the shop and the file that stores the orders, respectively. We should also pass two other files apart from the ones mentioned above, these files shall be used in writing back the data after and the shop is shut (log file) and the other will be used in generating a receipt for the user's order.

There are two modes in this code –

1) Server-side Functionalities
2) Client-side interactions

1)Server-Side Functionalities – These functionalities refer to those that are used to stock up existing products, add new products, remove few of the products. These are completely shop oriented functions and the users have no role in these functionalities. The functions are –

a) ***Add a product*** - Here the shopkeeper can add a new product by giving the appropriate details. First, we use the file that has the products in it, then go through it and check if the to be added product is present in the products file or not. We deny adding if the product is a redundant one. If it's a new one, then we add it to the place in the file which contains the first dummy value (id==-1 if possible) or else we go to the last place in the file where data is written and then add it to the file (we use **lseek() with SEEK_SET).** OfCourse file locking is implemented whenever we are making changes to the file.

b) ***Delete a product*** – Here the orderID is given as input into the terminal and then we search for that product in the file using the read sys call and once we find the necessary product, we lock that area in that file and then change the productID to -1 and unlock it. Also, we go to the orders page and then go through all the orders and check If the deleted product is part of any of them or not. If it is part of any of the order, then necessary changes are made to it and the written back to the file containing the orders. The file is locked while making the changes and then unlocked when the changes are completely done and written back to.

c) ***Update the product price in the inventory -*** We first go to the product file, then check for the product whose price must be updated, update the price for the same, lock the file and unlock it. We also go to the orders page and then we check for orders that contain this product once we get them, we lock that portion of the file and then update the price of the product so that the data present everywhere is coherent, and then unlock it whenever the update is written back to the file.

d) ***Update the product quantity in the inventory -*** We first go to the products file, then check for the product whose quantity must be updated, and we update the quantity for the same, lock the file and make changes to it and then unlock it.

e) ***Exit seller side functionalities and record changes in the log file*** – The third argument along with the executable is the file that will be used to record the items present in the shop. It shall write the number of products present in the shop and other details.

2) Client- Server communications -

Here we use socket programming to interact with the client. We used the concept of concurrent server instead of iterative server to cater to multiple requests simultaneously. Every time a new client new connection comes to the server, a new process is created using the **fork() function** and then further execution is done in the child process, the parent process will just be used accept new connection requests and create a child process for the same. Now coming to the important part of the execution, all the available options for the user mentioned earlier (in the user code) must be catered by the server so here are the functionalities that the server handles –

A) **A User can view the products listed in the inventory –** When the server receives this request from the client it first runs through the products file (argv[1]) and then it writes every valid and available product to the client using the **write sys call .**

B) **A User can view his cart -** When the server receives this request from the client then it first runs through the orders file(argv[2]) and then it will check the customerID and if it matches with the customerID it has received then it will send that order through the write sys call.

C) **A user can create a new order -** As mentioned earlier here two possibilities can occur, one is when there is no existing cart for the customer and second when there is a existing cart for the customer. When there is an existing cart for the customer then it will check if the number of products that are to be added can be added to the cart or not (as there is a cap of 10 distinct productss) in the cart. If yes it will proceed forward, if not then it will decline the request. If there is no existing cart, then it will create a new cart for this customer then assign it a new order number and then do the necessary work and writes it to the order file (locks it and then writes it).

D) **The user will modify his/her cart –** In this functionality the user shall update the quantity, first he/she shall give you his/her ID and then will give you the orderID and the productID for which the quantity must be modified. He/She will also give you the new quantity for that particular product. Now the server will run through after receiving the correct details it will first run though the orders file and then check for this order, it will then modify the order and then write it back to the order file **(after locking the file - always record locking).**

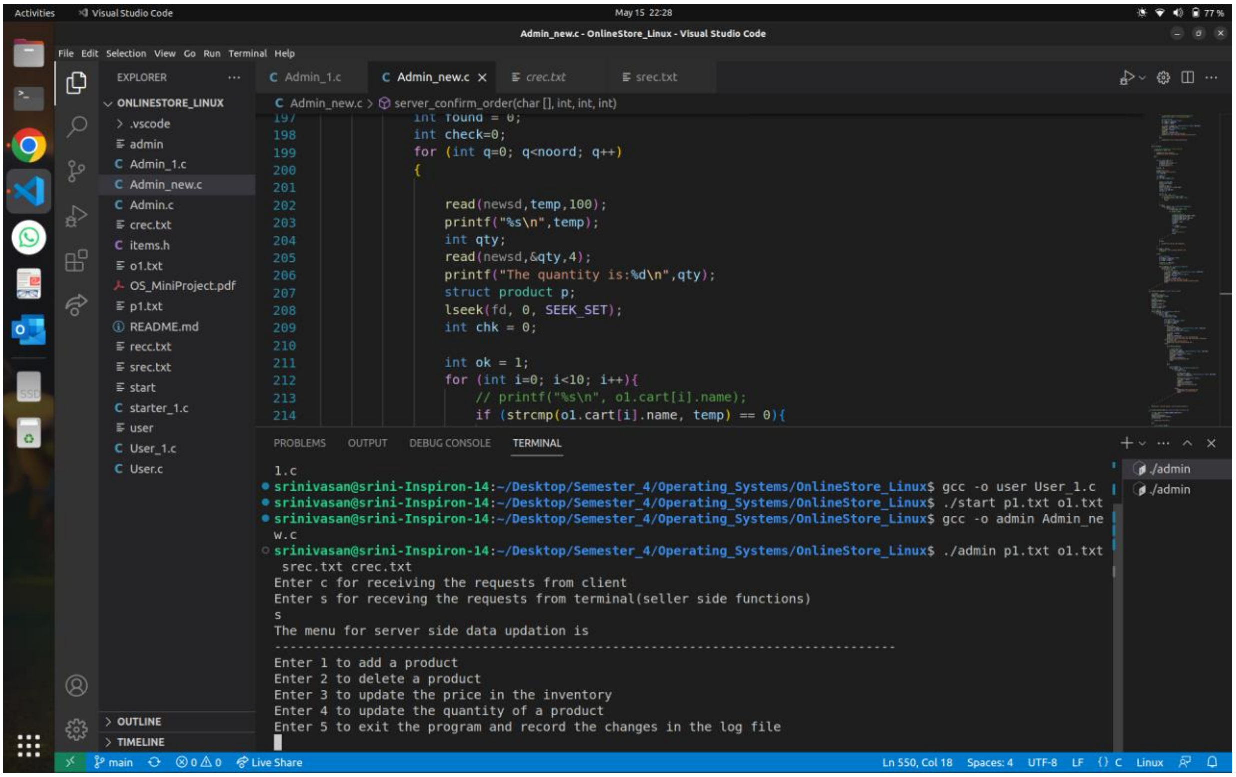
E) **The User will confirm the details and place the order –** This is the most important part of the program where the customer will buy the products present in the order. Once the server gets the orderID, it will first check if the orderID is valid or not, if yes then it will keep track of all the products in the cart and then it will proceed further, then it will iterate through the product list from the order and finds its match in the products file, once the data is matched the quantity is checked, if everything is correct then the order details will be shown to the customer once for review, if the customer is happy with all of that then, the he/she will give a confirmation to proceed then all the updates are written in the file and the order is placed and that order is removed from the file. Also, the receipt for the order placed is written in a new file that is passed as the 4th argument with the executable. In case of any unavailability of products then the order will not be placed.

Few Notes -

1) Most of the file locking that we did in our report uses, record locking(it only locks a part of the page and not the entire page).
2) I used functions wherever possible in the admin side to make the code look cleaner.
3) I used concurrent server using fork() instead of the iterative to simulate real life server-client operations.

Screenshots of working of code -

1) Starting the admin code for seller side operations



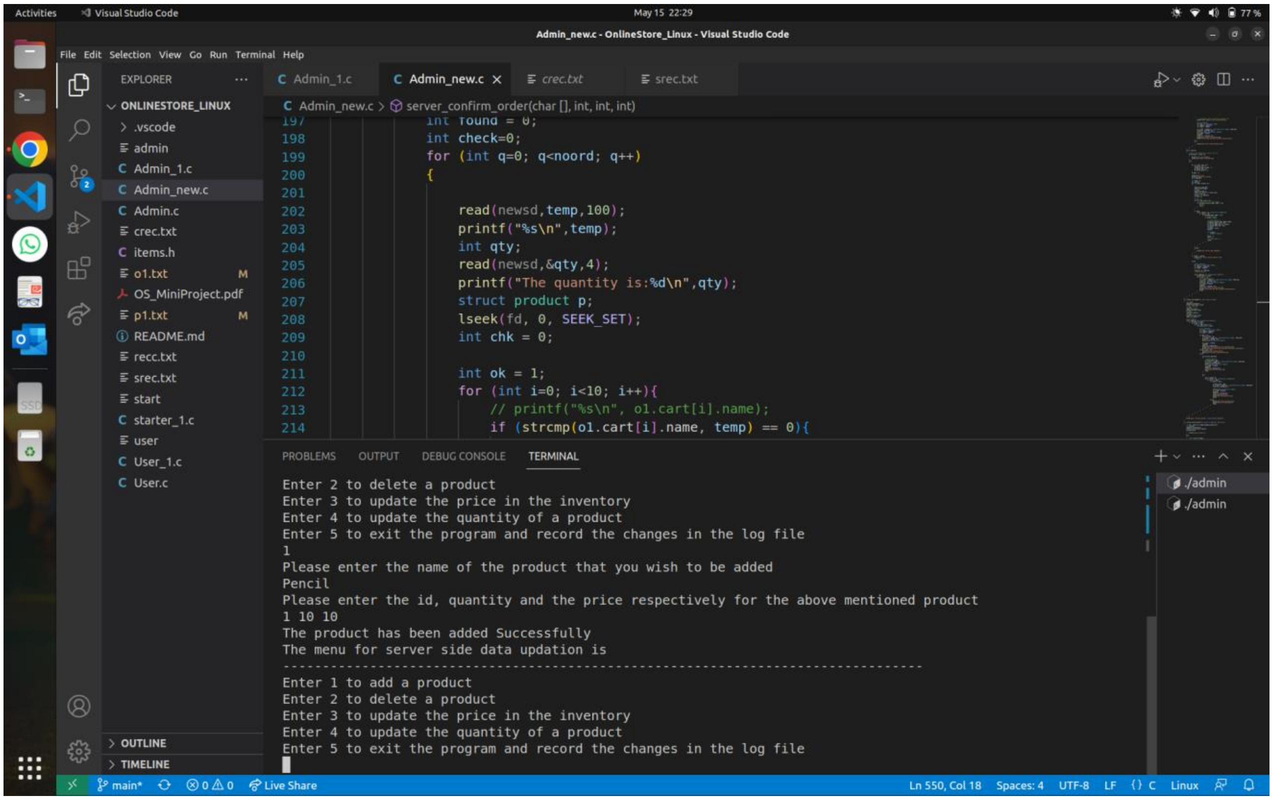
```c
            int found = 0;
            int check=0;
            for (int q=0; q<noord; q++)
            {

                read(newsd,temp,100);
                printf("%s\n",temp);
                int qty;
                read(newsd,&qty,4);
                printf("The quantity is:%d\n",qty);
                struct product p;
                lseek(fd, 0, SEEK_SET);
                int chk = 0;

                int ok = 1;
                for (int i=0; i<10; i++){
                    // printf("%s\n", o1.cart[i].name);
                    if (strcmp(o1.cart[i].name, temp) == 0){
```

```
1.c
srinivasan@srini-Inspiron-14:~/Desktop/Semester_4/Operating_Systems/OnlineStore_Linux$ gcc -o user User_1.c
srinivasan@srini-Inspiron-14:~/Desktop/Semester_4/Operating_Systems/OnlineStore_Linux$ ./start p1.txt o1.txt
srinivasan@srini-Inspiron-14:~/Desktop/Semester_4/Operating_Systems/OnlineStore_Linux$ gcc -o admin Admin_ne
w.c
srinivasan@srini-Inspiron-14:~/Desktop/Semester_4/Operating_Systems/OnlineStore_Linux$ ./admin p1.txt o1.txt
 srec.txt crec.txt
Enter c for receiving the requests from client
Enter s for receving the requests from terminal(seller side functions)
s
The menu for server side data updation is
--------------------------------------------------------------------------------
Enter 1 to add a product
Enter 2 to delete a product
Enter 3 to update the price in the inventory
Enter 4 to update the quantity of a product
Enter 5 to exit the program and record the changes in the log file
```

2) Adding a new product


Admin_new.c - OnlineStore_Linux - Visual Studio Code
Admin_1.c
Admin_new.c
crec.txt
srec.txt
Admin_new.c > server_confirm_order(char [], int, int, int)
ONLINESTORE_LINUX
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Ln 550, Col 18

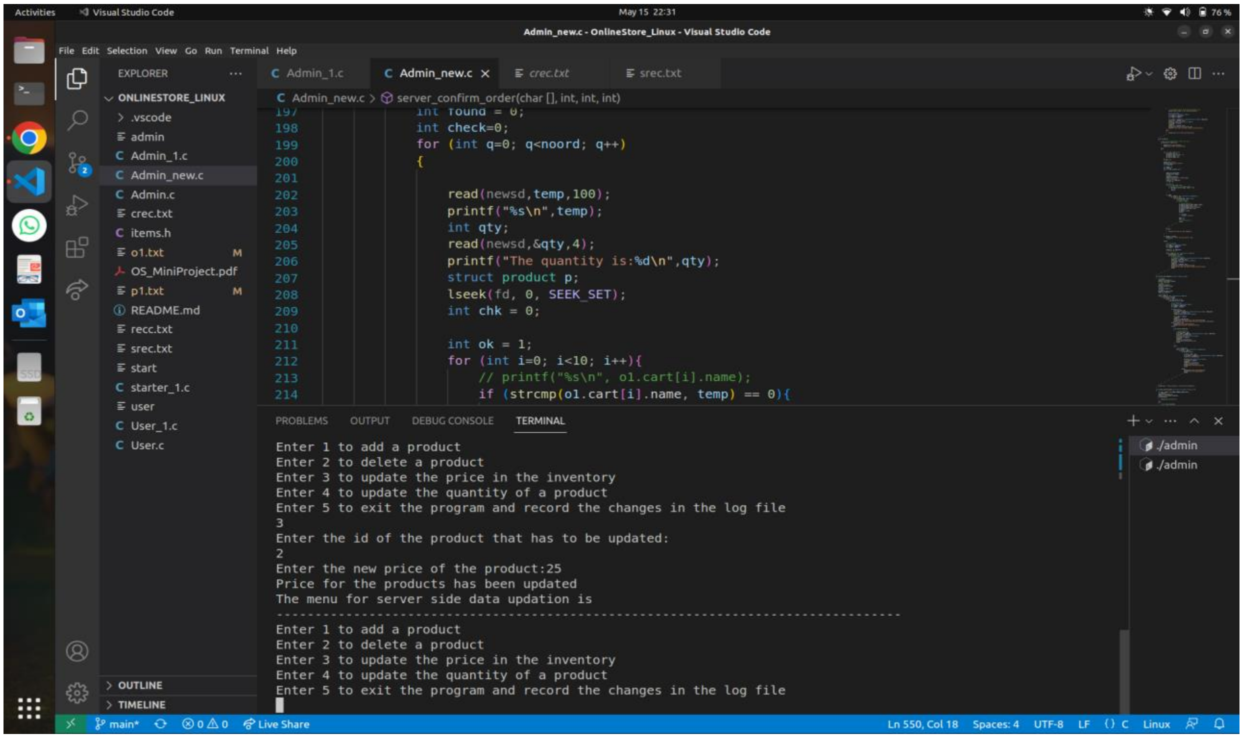
```c
            int found = 0;
            int check=0;
            for (int q=0; q<noord; q++)
            {

                read(newsd,temp,100);
                printf("%s\n",temp);
                int qty;
                read(newsd,&qty,4);
                printf("The quantity is:%d\n",qty);
                struct product p;
                lseek(fd, 0, SEEK_SET);
                int chk = 0;

                int ok = 1;
                for (int i=0; i<10; i++){
                    // printf("%s\n", o1.cart[i].name);
                    if (strcmp(o1.cart[i].name, temp) == 0){
```

```
Enter 2 to delete a product
Enter 3 to update the price in the inventory
Enter 4 to update the quantity of a product
Enter 5 to exit the program and record the changes in the log file
1
Please enter the name of the product that you wish to be added
Pencil
Please enter the id, quantity and the price respectively for the above mentioned product
1 10 10
The product has been added Successfully
The menu for server side data updation is
--------------------------------------------------------------------------------
Enter 1 to add a product
Enter 2 to delete a product
Enter 3 to update the price in the inventory
Enter 4 to update the quantity of a product
Enter 5 to exit the program and record the changes in the log file
```

3) Updating price of a product



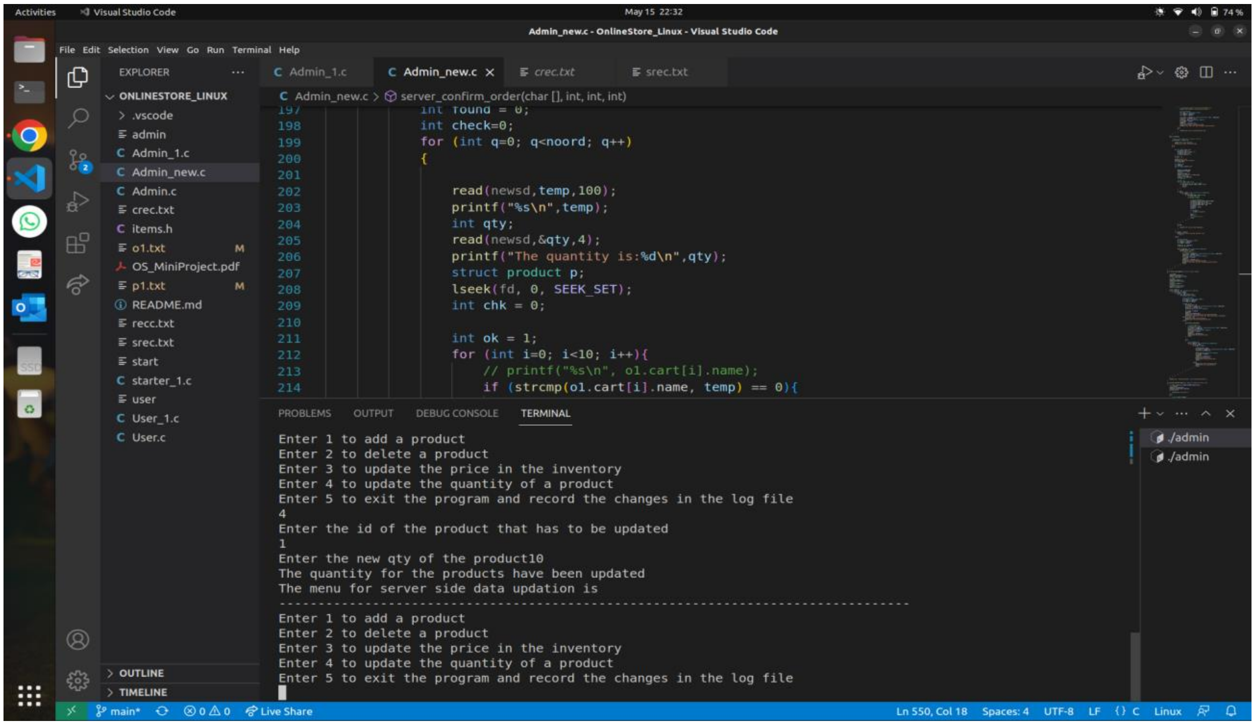
```
            int found = 0;
            int check=0;
            for (int q=0; q<noord; q++)
            {

                read(newsd,temp,100);
                printf("%s\n",temp);
                int qty;
                read(newsd,&qty,4);
                printf("The quantity is:%d\n",qty);
                struct product p;
                lseek(fd, 0, SEEK_SET);
                int chk = 0;

                int ok = 1;
                for (int i=0; i<10; i++){
                    // printf("%s\n", o1.cart[i].name);
                    if (strcmp(o1.cart[i].name, temp) == 0){
```

```
Enter 1 to add a product
Enter 2 to delete a product
Enter 3 to update the price in the inventory
Enter 4 to update the quantity of a product
Enter 5 to exit the program and record the changes in the log file
3
Enter the id of the product that has to be updated:
2
Enter the new price of the product:25
Price for the products has been updated
The menu for server side data updation is
--------------------------------------------------------------------------------
Enter 1 to add a product
Enter 2 to delete a product
Enter 3 to update the price in the inventory
Enter 4 to update the quantity of a product
Enter 5 to exit the program and record the changes in the log file
```

4) Updating the quantity of an existing product in the inventory



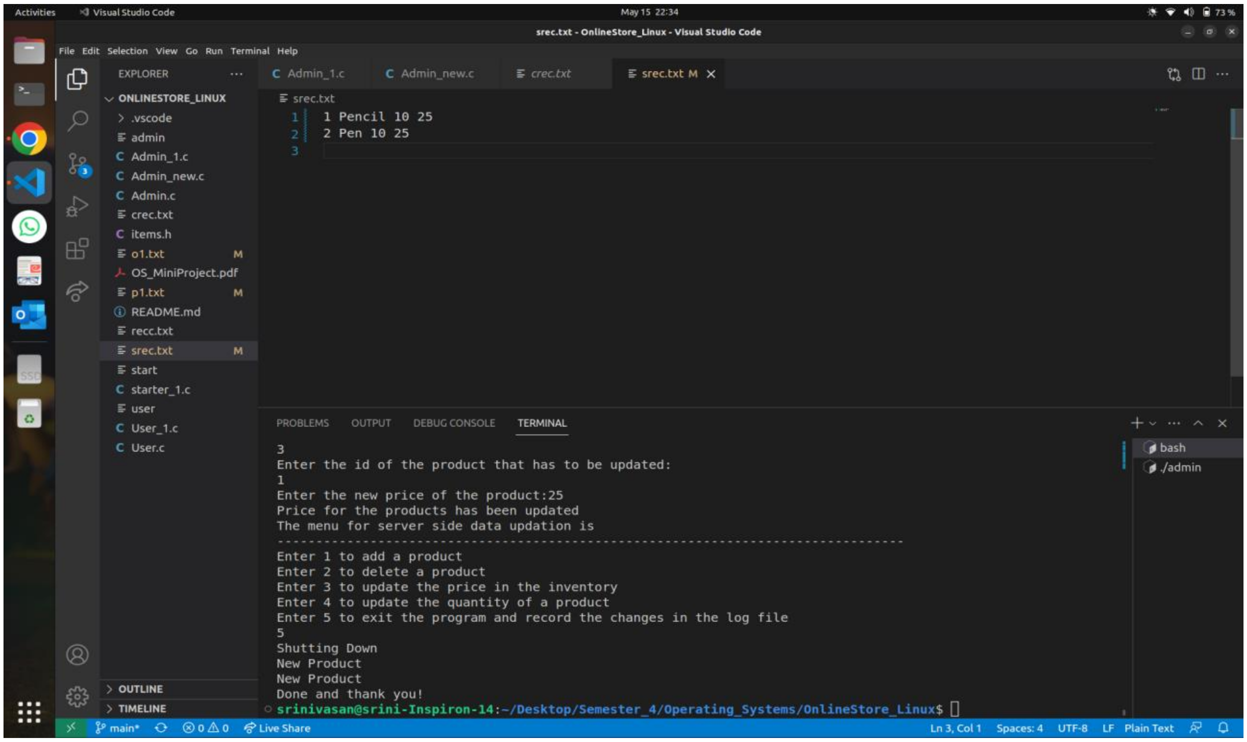
```
            int found = 0;
            int check=0;
            for (int q=0; q<noord; q++)
            {

                read(newsd,temp,100);
                printf("%s\n",temp);
                int qty;
                read(newsd,&qty,4);
                printf("The quantity is:%d\n",qty);
                struct product p;
                lseek(fd, 0, SEEK_SET);
                int chk = 0;

                int ok = 1;
                for (int i=0; i<10; i++){
                    // printf("%s\n", o1.cart[i].name);
                    if (strcmp(o1.cart[i].name, temp) == 0){
```

```
Enter 1 to add a product
Enter 2 to delete a product
Enter 3 to update the price in the inventory
Enter 4 to update the quantity of a product
Enter 5 to exit the program and record the changes in the log file
4
Enter the id of the product that has to be updated
1
Enter the new qty of the product10
The quantity for the products have been updated
The menu for server side data updation is
--------------------------------------------------------------------------------
Enter 1 to add a product
Enter 2 to delete a product
Enter 3 to update the price in the inventory
Enter 4 to update the quantity of a product
Enter 5 to exit the program and record the changes in the log file
```

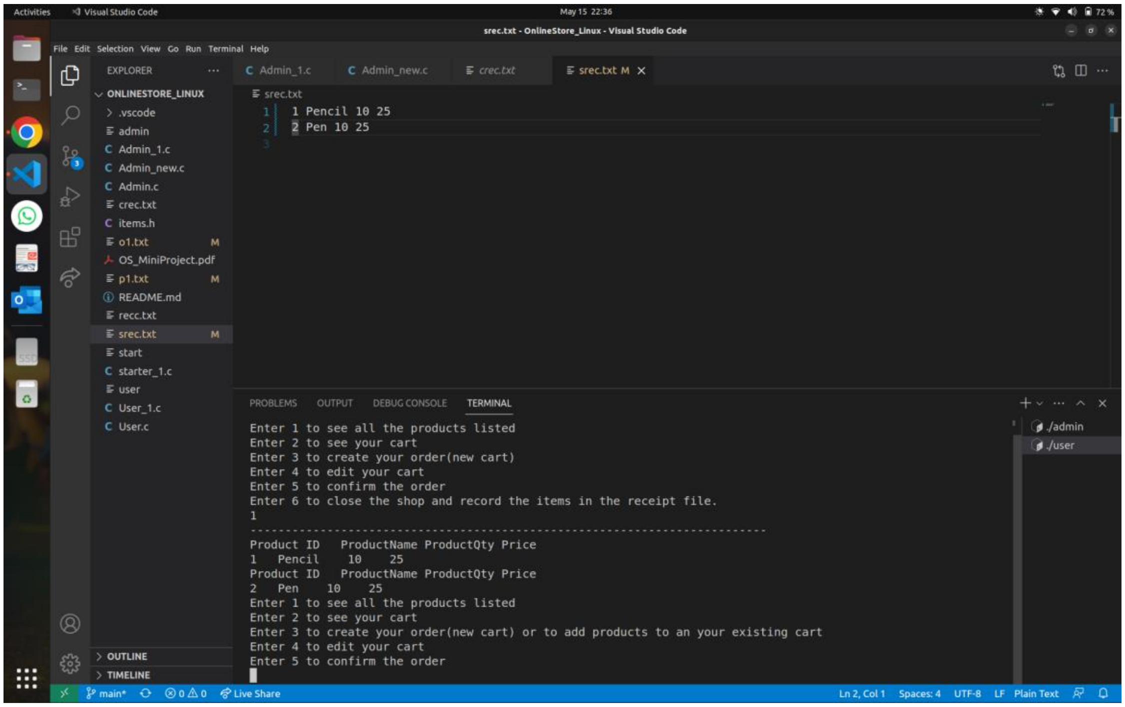
5) Finishing the seller side and creatin a log file



```
1 Pencil 10 25
2 Pen 10 25
```

```
3
Enter the id of the product that has to be updated:
1
Enter the new price of the product:25
Price for the products has been updated
The menu for server side data updation is
--------------------------------------------------------------------------------
Enter 1 to add a product
Enter 2 to delete a product
Enter 3 to update the price in the inventory
Enter 4 to update the quantity of a product
Enter 5 to exit the program and record the changes in the log file
5
Shutting Down
New Product
New Product
Done and thank you!
srinivasan@srini-Inspiron-14:~/Desktop/Semester_4/Operating_Systems/OnlineStore_Linux$ 
```

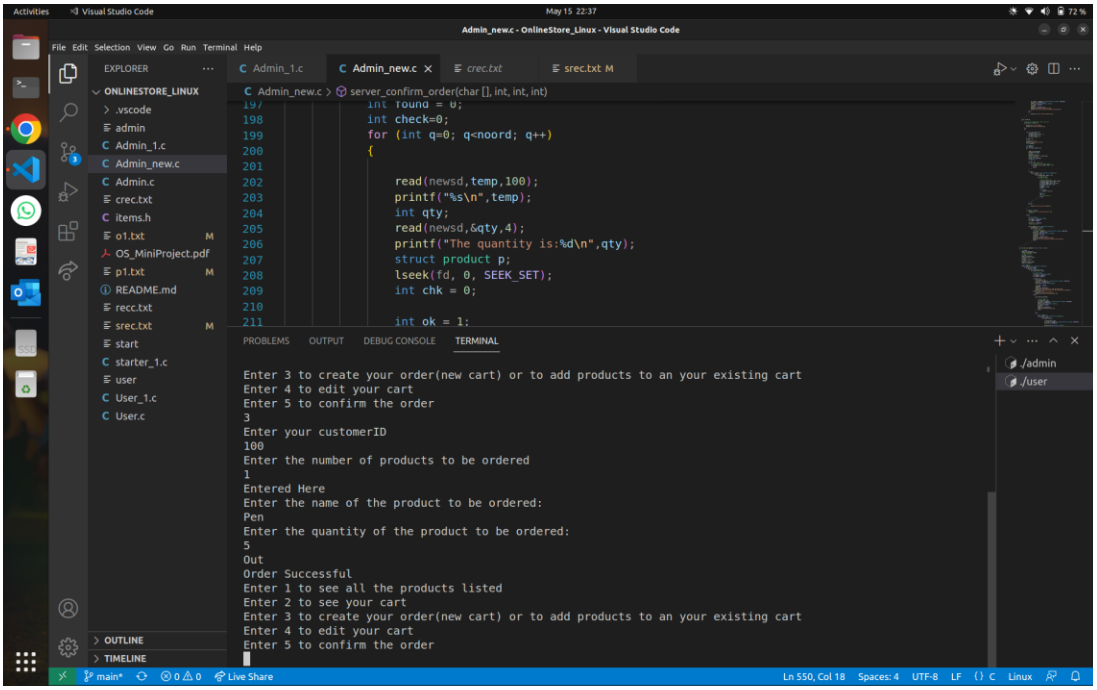
6) Client asking seller to send all the products in the inventory



```
1 Pencil 10 25
2 Pen 10 25
```

```
Enter 1 to see all the products listed
Enter 2 to see your cart
Enter 3 to create your order(new cart)
Enter 4 to edit your cart
Enter 5 to confirm the order
Enter 6 to close the shop and record the items in the receipt file.
1
-------------------------------------------------------------------------
Product ID   ProductName ProductQty Price
1   Pencil    10    25
Product ID   ProductName ProductQty Price
2   Pen    10    25
Enter 1 to see all the products listed
Enter 2 to see your cart
Enter 3 to create your order(new cart) or to add products to an your existing cart
Enter 4 to edit your cart
Enter 5 to confirm the order
```

7) Client creating a new order



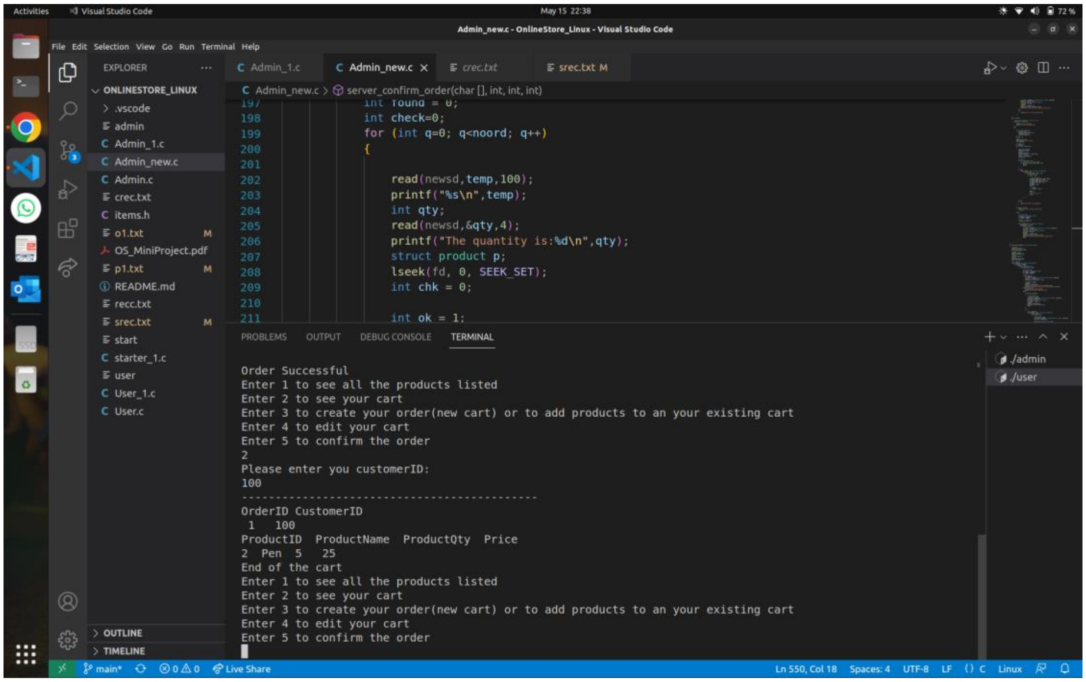
```c
            int found = 0;
            int check=0;
            for (int q=0; q<noord; q++)
            {

                read(newsd,temp,100);
                printf("%s\n",temp);
                int qty;
                read(newsd,&qty,4);
                printf("The quantity is:%d\n",qty);
                struct product p;
                lseek(fd, 0, SEEK_SET);
                int chk = 0;

                int ok = 1;
```

```
Enter 3 to create your order(new cart) or to add products to an your existing cart
Enter 4 to edit your cart
Enter 5 to confirm the order
3
Enter your customerID
100
Enter the number of products to be ordered
1
Entered Here
Enter the name of the product to be ordered:
Pen
Enter the quantity of the product to be ordered:
5
Out
Order Successful
Enter 1 to see all the products listed
Enter 2 to see your cart
Enter 3 to create your order(new cart) or to add products to an your existing cart
Enter 4 to edit your cart
Enter 5 to confirm the order
█
```

8) Client checking his cart



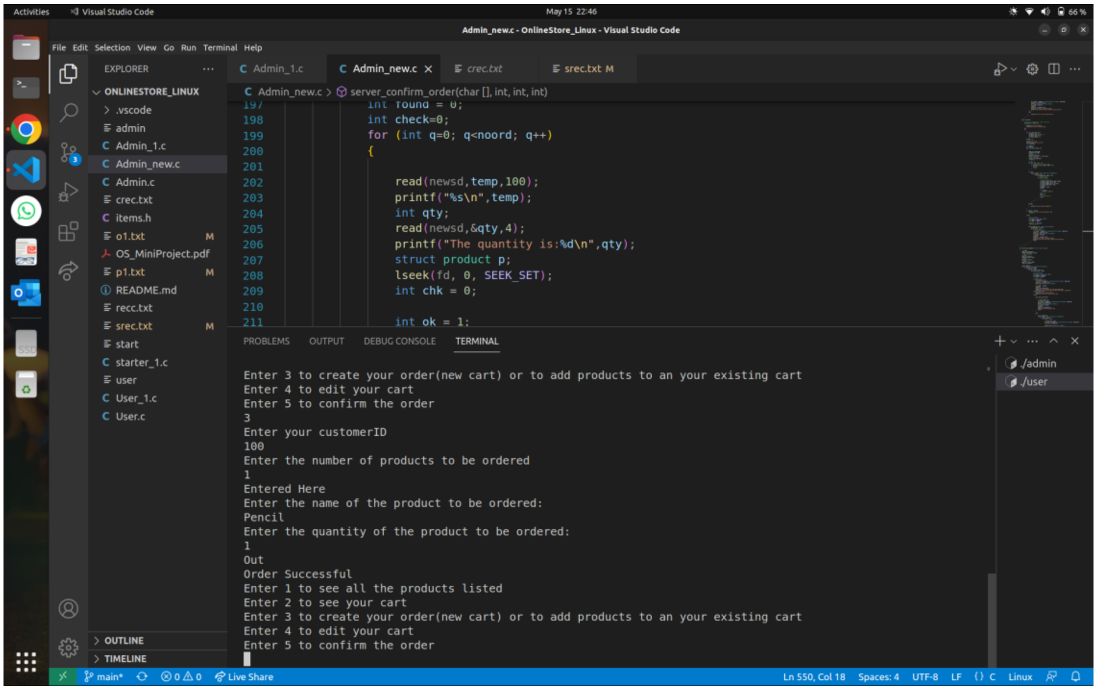
```c
            int found = 0;
            int check=0;
            for (int q=0; q<noord; q++)
            {

                read(newsd,temp,100);
                printf("%s\n",temp);
                int qty;
                read(newsd,&qty,4);
                printf("The quantity is:%d\n",qty);
                struct product p;
                lseek(fd, 0, SEEK_SET);
                int chk = 0;

                int ok = 1;
```

```
Order Successful
Enter 1 to see all the products listed
Enter 2 to see your cart
Enter 3 to create your order(new cart) or to add products to an your existing cart
Enter 4 to edit your cart
Enter 5 to confirm the order
2
Please enter you customerID:
100
---------------------------------------------
OrderID CustomerID
 1   100
ProductID  ProductName  ProductQty  Price
2  Pen  5   25
End of the cart
Enter 1 to see all the products listed
Enter 2 to see your cart
Enter 3 to create your order(new cart) or to add products to an your existing cart
Enter 4 to edit your cart
Enter 5 to confirm the order
█
```

9) Client adding new products to the existing cart

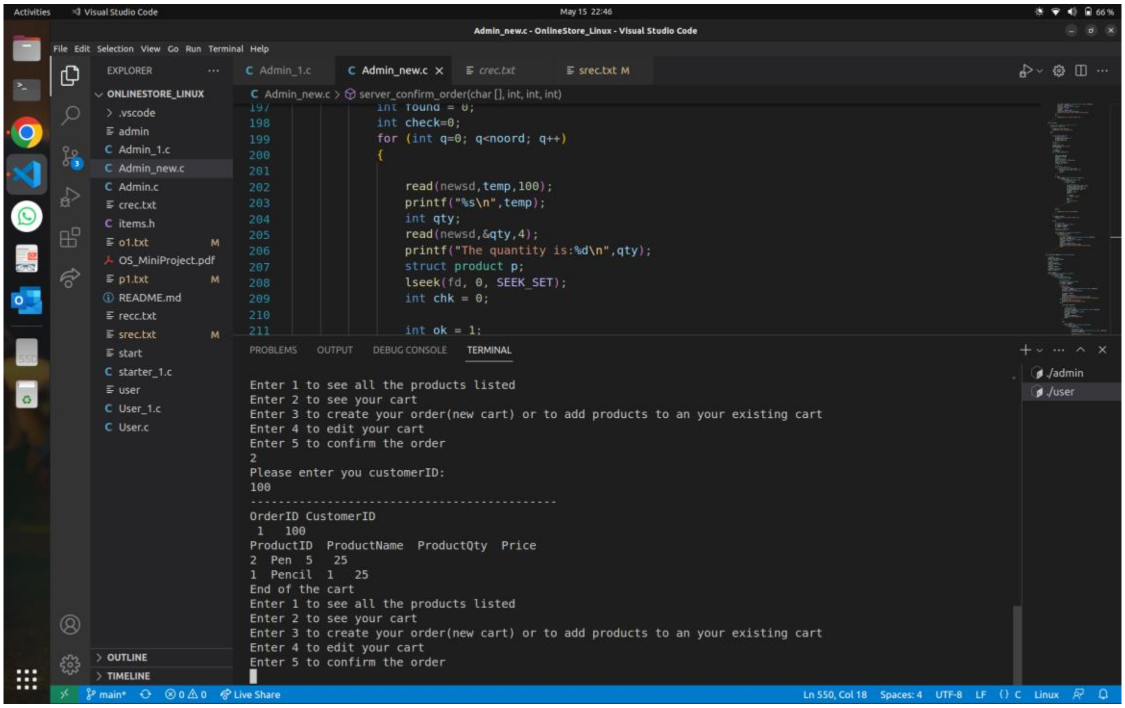


```c
int found = 0;
int check=0;
for (int q=0; q<noord; q++)
{

    read(newsd,temp,100);
    printf("%s\n",temp);
    int qty;
    read(newsd,&qty,4);
    printf("The quantity is:%d\n",qty);
    struct product p;
    lseek(fd, 0, SEEK_SET);
    int chk = 0;

    int ok = 1;
```

```
Enter 3 to create your order(new cart) or to add products to an your existing cart
Enter 4 to edit your cart
Enter 5 to confirm the order
3
Enter your customerID
100
Enter the number of products to be ordered
1
Entered Here
Enter the name of the product to be ordered:
Pencil
Enter the quantity of the product to be ordered:
1
Out
Order Successful
Enter 1 to see all the products listed
Enter 2 to see your cart
Enter 3 to create your order(new cart) or to add products to an your existing cart
Enter 4 to edit your cart
Enter 5 to confirm the order
```


Admin_new.c - OnlineStore_Linux - Visual Studio Code
EXPLORER
ONLINESTORE_LINUX
server_confirm_order(char [], int, int, int)
TERMINAL
./admin
./user
Ln 550, Col 18

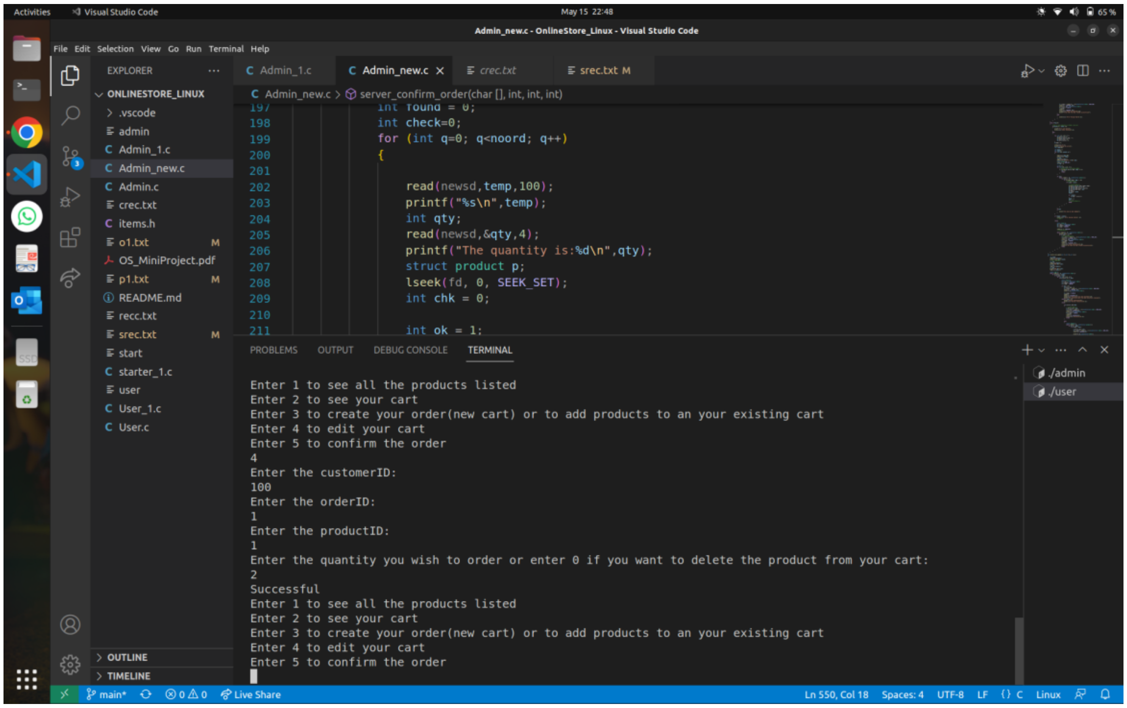
```c
int found = 0;
int check=0;
for (int q=0; q<noord; q++)
{

    read(newsd,temp,100);
    printf("%s\n",temp);
    int qty;
    read(newsd,&qty,4);
    printf("The quantity is:%d\n",qty);
    struct product p;
    lseek(fd, 0, SEEK_SET);
    int chk = 0;

    int ok = 1;
```

```
Enter 1 to see all the products listed
Enter 2 to see your cart
Enter 3 to create your order(new cart) or to add products to an your existing cart
Enter 4 to edit your cart
Enter 5 to confirm the order
2
Please enter you customerID:
100
--------------------------------------------
OrderID CustomerID
 1   100
ProductID  ProductName  ProductQty  Price
2  Pen  5   25
1  Pencil  1   25
End of the cart
Enter 1 to see all the products listed
Enter 2 to see your cart
Enter 3 to create your order(new cart) or to add products to an your existing cart
Enter 4 to edit your cart
Enter 5 to confirm the order
```

10) User edits his cart (changes the qty of items present in the cart-

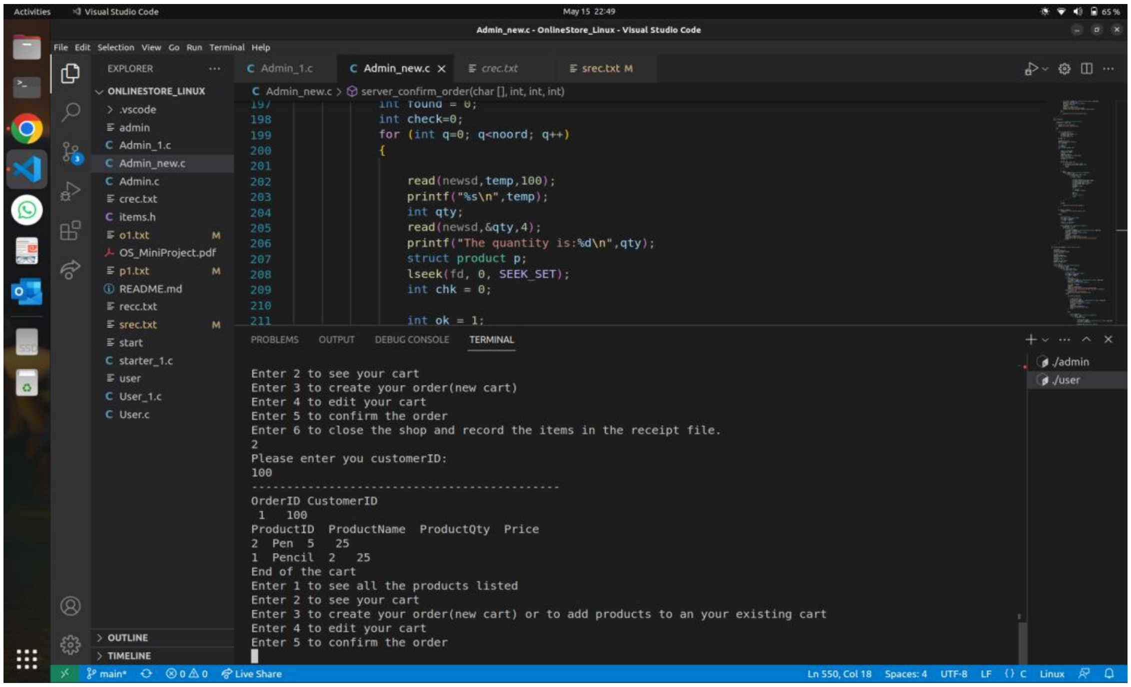


```
        int found = 0;
        int check=0;
        for (int q=0; q<noord; q++)
        {

            read(newsd,temp,100);
            printf("%s\n",temp);
            int qty;
            read(newsd,&qty,4);
            printf("The quantity is:%d\n",qty);
            struct product p;
            lseek(fd, 0, SEEK_SET);
            int chk = 0;

            int ok = 1;
```

```
Enter 1 to see all the products listed
Enter 2 to see your cart
Enter 3 to create your order(new cart) or to add products to an your existing cart
Enter 4 to edit your cart
Enter 5 to confirm the order
4
Enter the customerID:
100
Enter the orderID:
1
Enter the productID:
1
Enter the quantity you wish to order or enter 0 if you want to delete the product from your cart:
2
Successful
Enter 1 to see all the products listed
Enter 2 to see your cart
Enter 3 to create your order(new cart) or to add products to an your existing cart
Enter 4 to edit your cart
Enter 5 to confirm the order
```



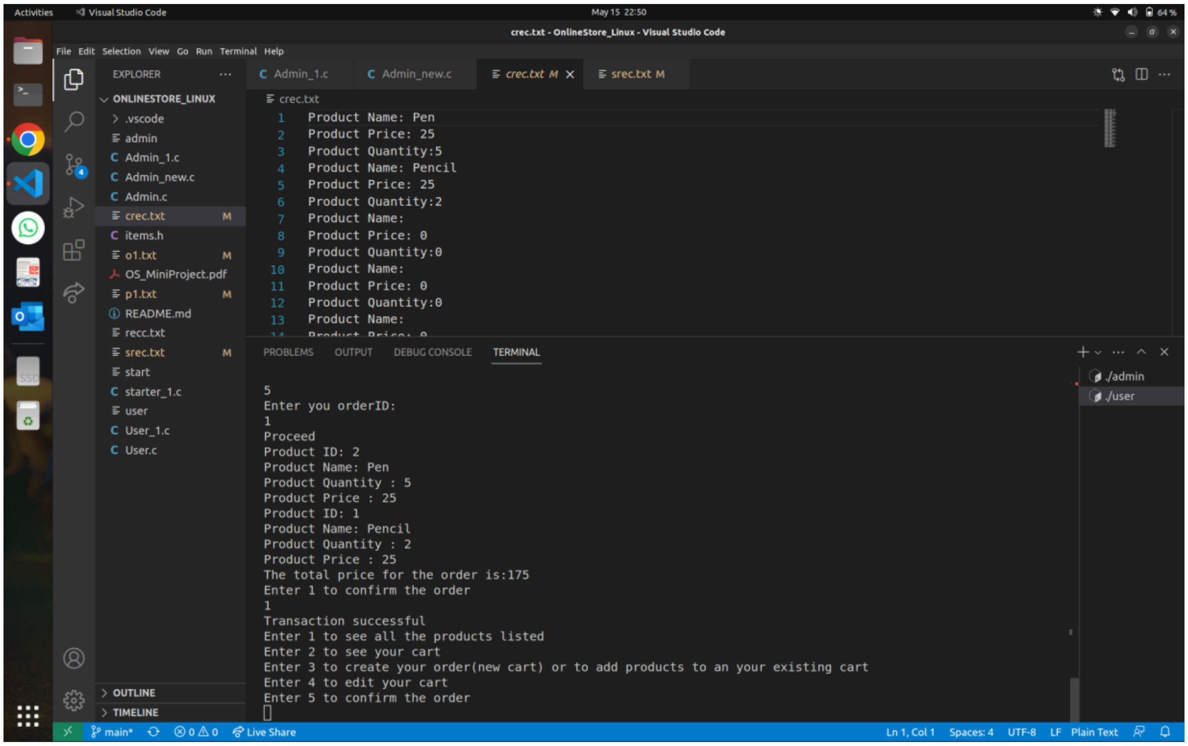
```
        int found = 0;
        int check=0;
        for (int q=0; q<noord; q++)
        {

            read(newsd,temp,100);
            printf("%s\n",temp);
            int qty;
            read(newsd,&qty,4);
            printf("The quantity is:%d\n",qty);
            struct product p;
            lseek(fd, 0, SEEK_SET);
            int chk = 0;

            int ok = 1;
```

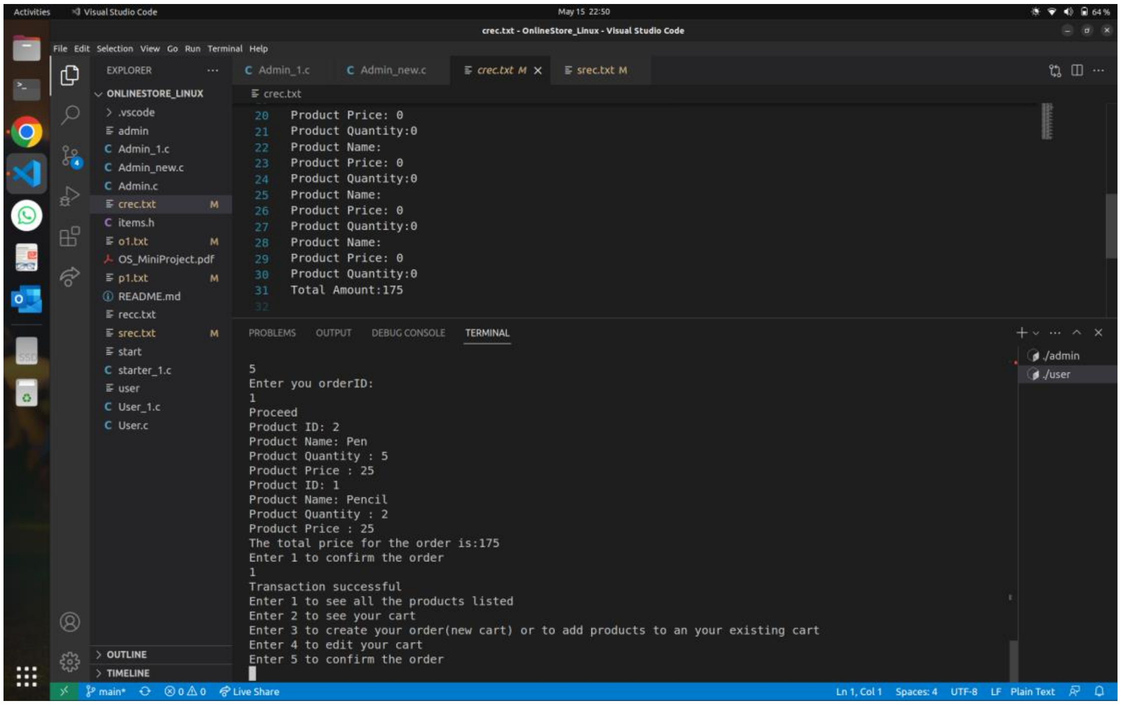
```
Enter 2 to see your cart
Enter 3 to create your order(new cart)
Enter 4 to edit your cart
Enter 5 to confirm the order
Enter 6 to close the shop and record the items in the receipt file.
2
Please enter you customerID:
100
--------------------------------------------
OrderID CustomerID
 1   100
ProductID  ProductName  ProductQty  Price
2  Pen  5   25
1  Pencil  2   25
End of the cart
Enter 1 to see all the products listed
Enter 2 to see your cart
Enter 3 to create your order(new cart) or to add products to an your existing cart
Enter 4 to edit your cart
Enter 5 to confirm the order
```

11) User confirms the order and then proceeds for the payment -



```
Product Name: Pen
Product Price: 25
Product Quantity:5
Product Name: Pencil
Product Price: 25
Product Quantity:2
Product Name:
Product Price: 0
Product Quantity:0
Product Name:
Product Price: 0
Product Quantity:0
Product Name:
```

```
5
Enter you orderID:
1
Proceed
Product ID: 2
Product Name: Pen
Product Quantity : 5
Product Price : 25
Product ID: 1
Product Name: Pencil
Product Quantity : 2
Product Price : 25
The total price for the order is:175
Enter 1 to confirm the order
1
Transaction successful
Enter 1 to see all the products listed
Enter 2 to see your cart
Enter 3 to create your order(new cart) or to add products to an your existing cart
Enter 4 to edit your cart
Enter 5 to confirm the order
```



```
Product Price: 0
Product Quantity:0
Product Name:
Product Price: 0
Product Quantity:0
Product Name:
Product Price: 0
Product Quantity:0
Product Name:
Product Price: 0
Product Quantity:0
Total Amount:175
```

```
5
Enter you orderID:
1
Proceed
Product ID: 2
Product Name: Pen
Product Quantity : 5
Product Price : 25
Product ID: 1
Product Name: Pencil
Product Quantity : 2
Product Price : 25
The total price for the order is:175
Enter 1 to confirm the order
1
Transaction successful
Enter 1 to see all the products listed
Enter 2 to see your cart
Enter 3 to create your order(new cart) or to add products to an your existing cart
Enter 4 to edit your cart
Enter 5 to confirm the order
```

GitHub link - https://github.com/Srini2404/OnlineStore_Linux.git