REPORT FOR OS MINI PROJECT

DONE BY: MAYANK CHAMARTHI(IMT2021073)

Overview: The Program has 3 parts server, client and admin. The server maintains the database of products. The client or user can view the products, add to cart, update products in cart, and pay for items in his cart. Admin can add a new product to the database, update the quantity and price of already existing products. The admin and the client use socket programming to connect to the server when they want to make a transaction I the database.

SERVER:

- Maintains the database and handles the transactions from the admin and the client.
- Uses two files:
    - product.dat: stores the data of all the products, in sequential order of their IDs i.e., ordering of the product objects in the file is

      {{id = 0, …}, {id = 1, ...}, …}

    - store.dat: stores the number of products that are in the database, it is updated every time the admin adds a new product.
- When the server boots up, it reads the store.dat file for the number of products in the database from the previous time the server was active.
- When a new client request is accepted the server forks a new process, now this process responds to the client's request while the parent server process listens for more connections.

Client: Can make the following requests.

- **View all products:**
    - Client sends a request to the server to send all the products in the database.
    - In response the server locks the file and reads its contents and stores them in an array of products, which is then sent to the client.
    - The client reads the array and prints products in the array.

- **Add to Cart:**
    - Client sends the ID and Quantity of the product to the server.
    - Server locks the file and reads the requested product into a product object.
    - If the available stock of the product can satisfy the client's request, database entry of the corresponding product's quantity in stock is updated and the product object is sent to the client. If add to cart fails error is sent to the client.
    - Client adds the product object to the cart.

NOTE: If x units of a product are added to the cart of a user, the database will now store that there are q-x units of the product available (q is the quantity available before the add to cart operation) even though the user did not buy the product. These x units are restored if the user exits without buying them.

- **Update Cart:**
    - Client sends the ID and the change in product quantity to the server. If the new quantity required by the user is more than the previous one change is positive, else change is negative.
    - Server locks the file and reads the corresponding product object from the DB.
    - If the change in quantity can be made, it is made, and server sends the client a successful response. Else the server sends the client a failure response.
    - If the change is successful, the client updates the cart.
- **PAY for items in cart:**
    - Client sends the Cart to the server.
    - For each item in the cart server checks whether its price has been changed in the database and sends an array storing change in price of cart items to the client. (Server locks the file while checking)
    - The client updates the cart according to the price changes in the array and displays the updated cart to the user who can now choose to pay or go back to the menu.
    - If the user pays, the cart is emptied.
- **EXIT:**
    - Client sends the cart to the server.

    - For each item in the cart, the server adds quantity in cart to the corresponding database entry.
    - Client then exits.

Admin: admin can do the following

- **ADD a new product:**
    - Admin creates a product object with id = -1 and sends it to the server.
    - Server assigns the object a new id and stores it in the database and returns the new id to the admin.
    - Admin adds an entry into the log file.
- **UPDATE product quantity/price:**
    - Admin sends the id and quantity/price to the server.
    - Server locks the file and makes changes.
    - Admin adds an entry to the log.

RUNNING THE PROGRAM:

- Run the **setup.exe** file first, this will set up the necessary files and populate the database.
- Compile and run the **server.c** file.
    - **gcc server.c -o server.exe**
    - **./server.exe**
- Compile and run the **client.c** file for client access.
    - **gcc client.c -o client.exe**
    - **./client.exe**
- Compile and run the **client_admin.c** file for admin access.
    - **gcc client_admin.c -o admin.exe**
    - **./admin.exe**