EG301P-Operating Systems Lab

IMTECH-CSE

4th Semester

Online Retail Store Management System using Linux Systems Programming

---

Author:

Vidhish Trivedi

IMT2021055

April-May, 2023

Contents

1 Introduction

- This project was made as part of the course: Operating Systems - Lab (EG301P), at IIIT-Bangalore.
- This is a client-server application that allows users to browse and purchase products online.
- The system has features such as customer authentication, product browsing, adding products to cart, checkout and order placement.
- The server-side of the application is responsible for handling client requests, managing product and user data, and processing orders.
- The client-side of the application provides a user-friendly interface for users (customers) and admins to interact with the system using the terminal.
- The project uses various programming concepts such as socket programming, file handling, multi-threading and file locking.

2 Project description

- This project makes use of socket programming to simulate an online retail store. It is structured in a client-server architecture. It offers features such as customer login and authentication, browsing products, maintaining a cart for each customer and making a purchase.
- A user can login as an admin to add new products to the platform by specifying a product id, name, price and quantity. They can also remove a selected product from the platform or update its price and quantity.
- A user may choose to login as a customer. If so, they would be able to list all products which are available on the platform along with their relevant information. They can also add products to their cart (separate for each customer), remove products from their cart or update quantity of a product in their cart. Once they are done, they can purchase their entire cart.
- A concurrent server has been implemented to allow multiple clients(customers/admins) to connect simultaneously to the server. Appropriate file locking has been implemented for concurrency control.

3 Features

3.1 General features

- The application provides a user choice menu on startup where the user can choose to login as one of:
  1. Admin (This is a single-admin application, but can easily be extended to support multiple admins)
  2. Customer
- Once the user makes a valid choice, they are asked to login using their credentials:
  1. Id and password for admin (default Id: 1, default password: admin_pass).
  2. Id and password for customers (default Ids: 1 to 25, default password: pass).
- On successful login, a menu is displayed as per the type of user which has logged in.

3.2 Admin features

- Add a new product to the platform.
- Remove an existing product from the platform.
- Update price and/or quantity of an existing product on the platform.
- Logout, which also generates a log of current stock of all products.

3.3 Customer features

- List all products available on the platform.
- List products added to their cart.
- Add a product to their cart.
- Update a product in their cart.
- Purchase products in cart, on successful purchase, a receipt is generated with filename as the current date-time.
- Logout.

4 Project structure

4.1 Directories

Directory	Role
data/	Directory for files which store data about users and products persistently.
logs/admin	Directory for storing log file for Admin.
logs/receipts	Directory for storing receipt files for purchases made by users.
header/	Directory for header files.
src/	Directory for source files.
target/	Directory where the object files and executables are generated.

4.2 Files

File	Role
src/main_server.c	Contains main function for starting the server-side application.
src/main_client.c	Contains main function for starting the client-side application.
src/client_menu.c	Contains functions to display various menus on the client-side application.
src/server.c	Contains functions for server and connection, which are called when a separate thread is created for an incoming client.
src/admin.c	Contains functions for admin-related actions.
src/user.c	Contains functions for user-related (customer) actions.
src/set_up_data.c	Contains main function for setting up the data files and semaphore initially.

5 Concepts used

- **Socket:** Sockets are used to establish a connection between the client and server terminals. The client sends requests to the server using sockets, and the server responds to these requests using the same socket. This allows for real-time communication between the client and server, enabling the user to browse and purchase products seamlessly.
- **Multithreading:** Multithreading is used to handle multiple client connections simultaneously. Each client connection is handled by a separate thread, allowing the server to handle multiple requests at the same time. This improves the performance of the system and ensures that the user experience is not affected by the number of clients connected to the server. (Used to set up a concurrent server).
- **File Locking:** File locking is used to prevent multiple clients from accessing the same file simultaneously. When a client accesses a file, the specific record of the file is locked, preventing other clients from accessing it until the first client has finished. This ensures that the data in the file is not corrupted due to multiple clients accessing it at the same time.
- **File Handling:** For reading from data files and writing to data files. This makes use of read() and write() system calls and allows us to persistently store data.
- **Semaphore:** A semaphore has been used to implement a lock on the payment gateway. The server allows only one customer to enter the payment gateway using binary semaphore.

6 Design choices

- There are 3 data files which are used by the project, these are present in the data/ directory:
  1. **Admin_User:** Stores information about admin(s) such as id, name and password.
  2. **Customer_User:** Stores information about customers such as id, name, password and also stores what items they have added to their cart.
  3. **Product_List:** Stores information about available products and their information, such as id, name, price and quantity.
- When an admin is adding a new product to the platform, no file locking is necessary. There are no concurrency conflicts in this case.
- However, when an admin tries to delete a product or update its price and/or quantity, the Product_List file is locked. The record for the chosen product is write-locked using advisory file locking. This allows other clients to access the remaining products concurrently.
- When a customer tries to login, the server authenticates their credentials from the Customer_User file. Here, the file in question is read-locked using advisory file locking.

- When a query for either all products or a specific product is made, the Product_List file is read-locked using mandatory and advisory file locking respectively.
- When a customer adds a product to their cart, the Customer_User file is write-locked using advisory locking for that particular customer's record. Updating cart by a customer follows a similar approach for updating quantity of a product that has been added to cart, or remove it altogether from the cart.
- The purchase function is locked using semaphore when a customer enters the payment gateway, as specified in the problem statement.
- A concurrent server is set up using threads, which allows multiple clients to connect at the same time.

7 Requirements and running the project

- Source code: https://github.com/Vidhish-Trivedi/Online-Retail-Store
- The project is written in C and uses make for a breezy compilation experience.
- If you do not have make installed, run the following in your terminal:

```
sudo apt install make
```

- To compile the code and generate the executable files, run the following commands in the project root:

```
cd Online-Retail-Store
gcc ./src/set_up_data.c -o target/dataWriter
target/dataWriter
```

- The above step is a one-time step used to set up data files for the project with some default data and creates a semaphore.
- To compile the server-side and client-side programs, run the following:

```
make target/server_exe
make target/client_exe
```

- To run the server-side program, run the following:

```
target/server_exe
```

- To run the client-side program, run the following:

```
target/client_exe
```