OS MINI-PROJECT REPORT

SAI MADHAVAN G – IMT2021101

GitHub link: https://github.com/SaiMadhavanG/os_miniproj

APPROACH TO THE PROBLEM

ARCHITECTURE

- A **client-server** architecture has been used for this project.
- The clients will interact with the application using a client window which will only act as an interface to the server and will not do any data storage or computation on its own.
- The client and server communicate through the means of sockets.
- A bulk of the data storage, computation and business logic is being handled on the server side.
- The admin can also access the server side for adding, deleting, or updating the inventory.

SERVER

- A concurrent server has been used with each new client being handled by a new process with the use of the `fork()` system call.
- A child process has also been created as the menu interface for the admin to make changes to the inventory and to shut down the server.
- The data of the inventory is being stored in a datafile dubbed as `product.dat`, where the list of products in the store is being stored as an array of structs.
- While shutting down the server, the child process (admin interface) sends a `SIGKILL` signal to the parent process (the server).
- A log file is being maintained which always writes the current state of products along with the timestamp to the end of the file.

CLIENT

- Each client is uniquely identified by the process id of the interface application.
- The client has a simple menu-based interface for interacting with the application.
- Each of the commands of the user is communicated to the server by sending a `struct product` with the necessary data loaded through the socket and the server, in turn, sends a response to the requested data.
- Each client has a cart of desired items which is maintained on the server.
- The client can choose to add, remove, or edit items from their cart.

- Once, the client decides to buy the items in their cart, they are led to the payment gateway, where they must enter the correct amount required for the purchase of the items in the cart.
- Upon successful payments, bills are generated, detailing the contents of items purchased along with the process id and timestamp.

CONCURRENCY

- Concurrency and avoidance of race condition has been achieved using System V Semaphores.
- During initialisation, a semaphore set containing the same number of semaphores as the maximum number of products permissible, is created. Each of these semaphores is a binary semaphore.
- Each such semaphore corresponds to one product in the datafile and is responsible for locking that record whenever it is being written on.
- The records are locked in two occasions –
    1. Whenever the admin is updating or deleting a record, that record gets locked.
    2. Whenever a client is proceeding to check out, the items in their cart are locked until the payment is complete.

INITIALISATION

- Initialisation is done by executing the **`init.out`** file, which creates the semaphores and datafile if they do not exist already.

CONCEPTS OF OS USED

SEMAPHORES

Semaphores were used for locking and unlocking records in the datafile. They helped with concurrency and avoiding race condition.

SOCKET PROGRAMMING

The server and client used sockets to communicate with each other.

PROCESS CREATION

New processes were created every time a client connected to the server.

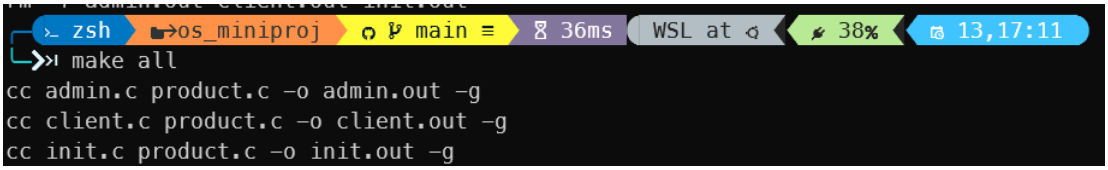
SIGNALS

Signals were used to terminate related processes.

INSTALLATION AND USER GUIDE (WITH SCREENSHOTS)

INITIALISATION

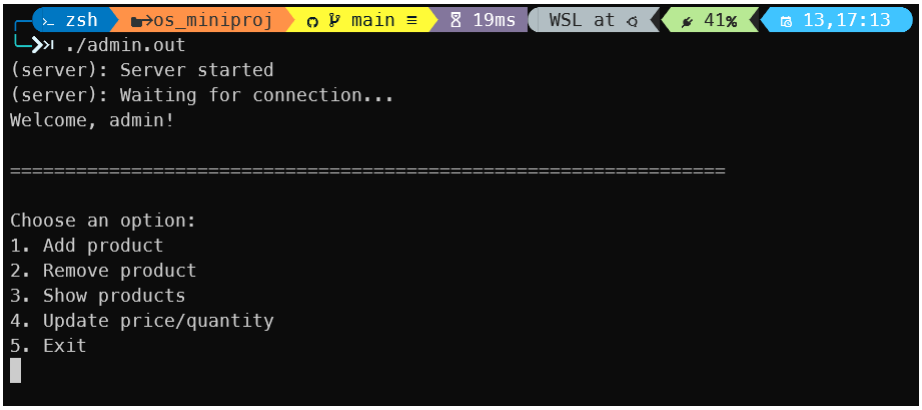
1. On the terminal, run the **`make all`** command to make the Makefile compile all the required files.



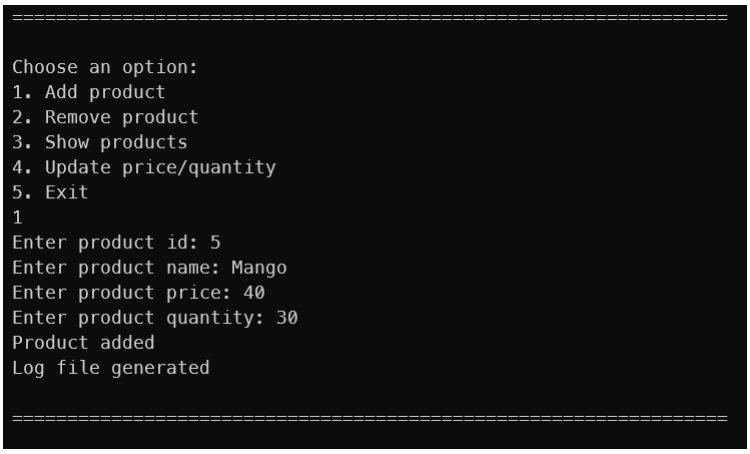
2. Execute the **`init.out`** file to create the semaphores and datafile.

ADMIN

1. To start the server, execute the **`admin.out`** file.

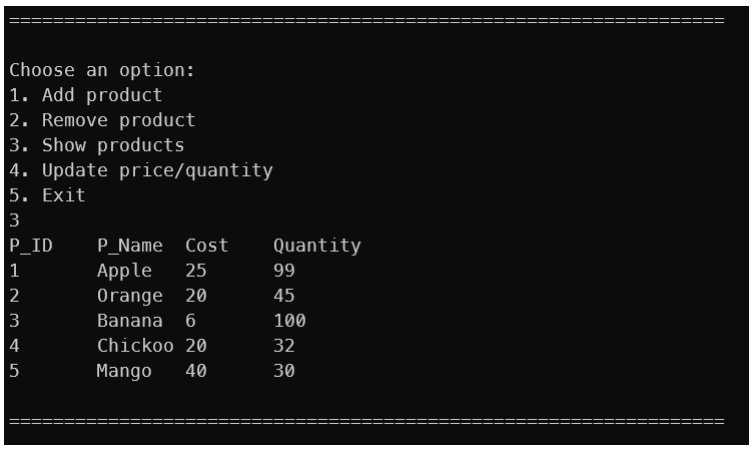


2. To add a product to the store, press 1.



3. To remove a product, press 2.

4. To get a list of all products in the store, press 3.



5. To update the price or quantity of a product, press 4

6. To shut down the server, press 5.

```
================================================================

Choose an option:
1. Add product
2. Remove product
3. Show products
4. Update price/quantity
5. Exit
5
Exiting...
(server): Server down
zsh: killed     ./admin.out
```

7. For the most recent changes by the admin to the store inventory, check the end of the **`update.log`** file.

```
 zsh  os_miniproj  main ≡  0ms  WSL at  51%  13,17:23
 tail update.log
5       Mango   40      30

Log generated at Sat May 13 17:20:45 2023

P_ID    P_Name  Cost    Quantity
1       Apple   25      99
2       Orange  20      45
3       Banana  6       100
4       Chickoo 20      32
5       Mango   45      30
```

CLIENT

1. To start an instance of a client, execute the **`client.out`** file. (Make sure the server is up)

```
 zsh  os_miniproj  main ≡  12ms  WSL at  51%  13,17:23
 ./client.out
Welcome user 28835!
Connected to server

================================================================

Choose an option:
1. Show products
2. Display cart
3. Add to cart
4. Edit cart
5. Payment window
6. Exit
```

2. To get a list of all products in the store, press 1.

```
================================================================

Choose an option:
1. Show products
2. Display cart
3. Add to cart
4. Edit cart
5. Payment window
6. Exit
1
P_ID    P_Name  Cost    Quantity
1       Apple   25      99
2       Orange  20      45
3       Banana  6       100
4       Chickoo 20      32
5       Mango   45      30

================================================================
```

3. To add items to cart, press 3.

```
================================================================

Choose an option:
1. Show products
2. Display cart
3. Add to cart
4. Edit cart
5. Payment window
6. Exit
3
Enter product id: 5
Enter quantity: 2
SUCCESS: Items added to the cart

================================================================
```

4. To display the contents of the cart, press 2.

```
================================================================

Choose an option:
1. Show products
2. Display cart
3. Add to cart
4. Edit cart
5. Payment window
6. Exit
2
P_ID    P_Name  Cost    Quantity
1       Apple   25      5
3       Banana  6       10
5       Mango   45      2

----------------------------------------------------------------
```

5. To edit the contents of the cart, press 4.

```
================================================================

Choose an option:
1. Show products
2. Display cart
3. Add to cart
4. Edit cart
5. Payment window
6. Exit
4
Enter product id: 5
Enter quantity: 4
SUCCESS: Cart items have been edited

================================================================
```

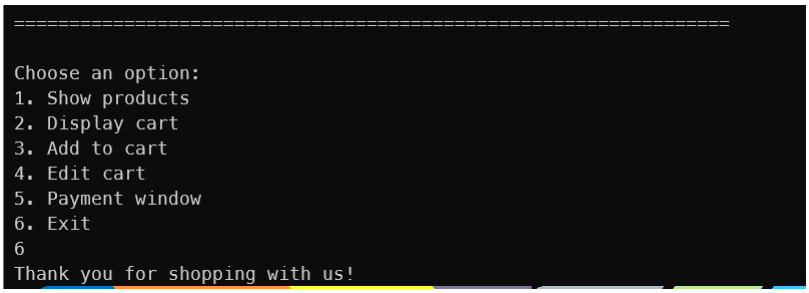
6. To go to the payment gateway, press 5.

```
================================================================

Choose an option:
1. Show products
2. Display cart
3. Add to cart
4. Edit cart
5. Payment window
6. Exit
5
WAIT: Checking out...
P_ID    P_Name  Cost    Quantity
1       Apple   25      5
3       Banana  6       10
5       Mango   45      4
Total amount: 365
Enter amount: 365
SUCCESS: Payment successful
BILL: Bill generated

================================================================
```

7. To exit, press 6.



8. Check for the bill generated for your transaction in the same directory.