Системно Програмиране

Курсова работа / Проект за оценка по Системно Програмиране

Тема 21

Технически Университет - София
Факултет Компютърни системи и технологии

Николай Георгиев Станишев

25.05.2021

1. Задание

Когато искате да изтеглите пари от банкомат, се прави трансакция от този банкомат към вашата банкова сметка. Реализирайте две програми, като едната е самият банкомат (потребителски интерфейс) - изпълнява всички операции: избор на език, сума, дали да има касов бон и PIN-код (проверката за него става в банкомата). Скрито за потребителя остава вземането на номера на картата, но той все пак се взима предвид. След като операцията е разрешена, се изпраща заявка до банковата сметка (сума и номер на сметката), където се прави проверка. Ако проверката върне грешка, се извежда съобщение на екрана (с цел улеснение - на банката). Информацията за клиентите се пази във файл и се зарежда в контейнер при всяко стартиране на програмата.

1.1. Анализ

Приложението ще съдържа две програми, които до максимална степен ще напоподобяват действието на реален банкомат. Комуникацията на клиента към тези приложения ще бъде изпълнена чрез конзолен интерфейс. Едното приложение ще бъде самият банкомат, а другото ще бъде банковият сървър. Комуникацията между тях ще бъде осъществена с tcp сървър, който ще си комуникира със сокети.

Приложението за банкомат ще се грижи само за въвеждане на данните от клиента, като ще може да се правят безкрайно много операции с него. Първото нещо, за което ще бъде питан потребителя е езикът, на който да функционира този банкомат. След това ще има въпрос дали потребителя е приключил с неговото използване. Ако не е приключил ще последва въпрос дали е администратор или обикновен потребител.

Ако е администратор, той ще може да добави нов клиент в базата данни на сървъра. За целта, данните които ще бъдат изисквани от него ще бъдат име на банковата карта, номер на тази карта, пин код и сума по

сметката на новият клиент. Името на банковата е карта е ключово, защото това ще представлява името на файла в който ще се пази информацията за картата. След въвеждането на необходимите данни се изпраща заявка към сървъра, който ще добави новият потребител и ще върне резултат съдържащ код и съобщение, които ще бъдат визуализирани на стандартния изход.

Ако е стандартен потребител, той ще може да изпълни две операции това ще бъдат да изтегли пари и да види каква е наличността по сметката му. За целта, данните за които ще бъде попитан са следните - име на картата, пин код, размер на тегленитe пари, дали да има бележка свързана с транзакцията. Тук има няколко ключови неща. Първото е име на картата, това представлява име на файл, намиращ се на файловата система на клиента, който съдържа номера на картата. Следващото ключово нещо е дали да има бележка свързана с транзакцията. Ако клиента избере да има то тогава резултатът от транзакцията ще бъде запаметен във файл на клиентската файлова система. Последното ключово нещо е свързано с размера на тегленитe пари, ако клиента избере 0 за тази опция, то тогава действието на тази операция ще бъде проверка на наличността по неговата сметка. След въвеждането на всички тези данни се изпраща заявка към сървъра, който обработва данните за транзацията и връща резултат съдържащ код и съобщение.

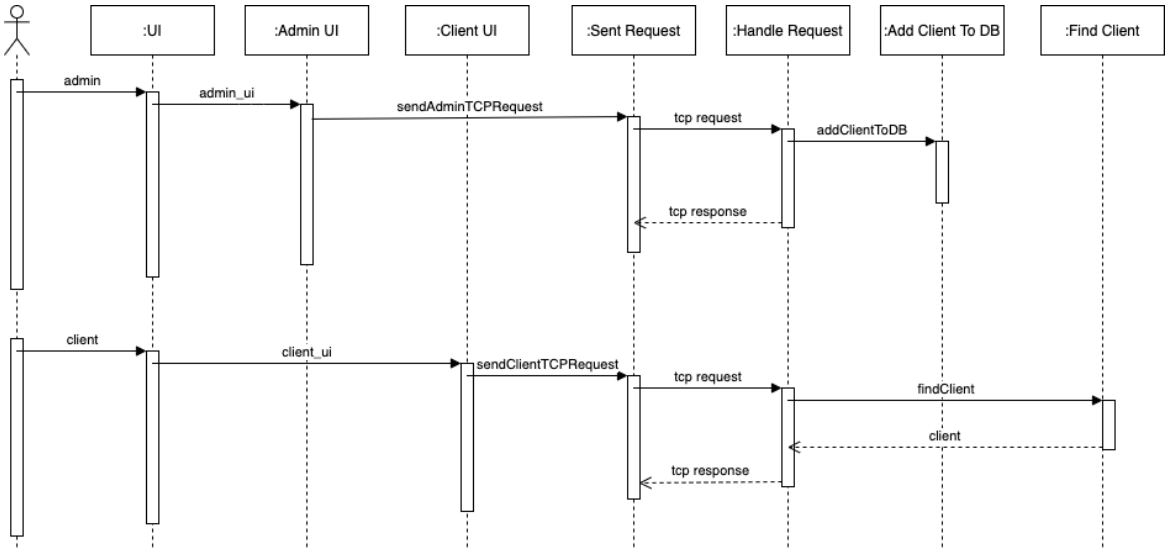
Приложението за сървър ще се грижи за обработване на заяките изпратени му от банкомата. При неговото стартиране, се зарежда базата данни от файловата система, а при приключването на неговият процес се извършва запазване на базата данни на файловата система. Форматът в който се пази е бинарен файл съдържащ структури от данни. Подържаните заявки са две - добавяне на потребител и извършване на парично теглене. Тези две заяки са осъществени с помоща на две нишки, които чакат заяки за двете операции.

Първата подържана заявка е за добавяне на потребител. При нея след получаване на данните за нов потребител се извършва проверка дали този потребител вече е наличен в базата данни, ако е наличен се връща код и съобщение че такъв потребител същесвува на системата, ако ли не то той се добавя в зареденатa база от данни.

Втората подържана заявка е за изпълнение на транзакция свързана с теглене от банковата сметка. След получаване на такаква заявка от банкомата, се извъшва проверка дали има такъв потребител, ако не е наличен се връща код и съобщение за грешка, следващата проверка е дали клиента има достатъчно налични финанси, за да покрият търсенето от потребителя, ако няма се връща съобщение за грешка ако има се осъществява транзакцията, като се връща информация за изтеглените пари и наличните по сметката му.

2. Функционално описание

2.1. Sequence diagram



3. Изпълнение на функционалностите

3.1. Споделени ресурси

```
typedef enum {false, true} bool;
```

Енъм използван за репрезентация на липсващият в езикът с boolean тип.

```
typedef struct Transaction {
```

```
    int withdraw_amount;
    bool cash_receipt;
    char pin[5];
    char card_number[17];
} Transaction;
```

Структура използвана репрезентация на операция за транзакция. Съдържанието на тази структура е размерът на теглене, дали да има бележка с информация за транзакцията, четири символен пин код и шестнадесет символен номер на картата.

```
typedef struct Client {
    int amount;
    char pin[5];
    char card_number[17];
} Client;
```

Структура използвана за репрезентация на банков клиент. Описанието на клиента става чрез наличност по сметката му, четири символен пин код и шестнадесен символен номер на картата му.

```
unsigned short CLIENT_PORT = 33333;
unsigned short ADMIN_PORT = 44444;
```

Портовете за комуникация между банкомата и банковият сървър. Двата порта са за двете възможни операции създаване на клиент и извършване на транзакция.

```
typedef enum ResponseCode {
    MISSING_USER,
    NOT_ENOUGH_MONEY,
    SUCCESSFULL_WITHDRAW,
    CLIENT_ALREADY_EXISTS,
    SUCCESSFULL_CLIENT_CREATION
} ResponseCode;

typedef struct Response {
    ResponseCode code;
    char message[1000];
} Response;
```

Структура съдържаща информация за резултата от извършената транзакция. Данните, които съдържа са резултатния код и съобщение. Резултатния код е репрезентиран чрез enum, чиито стойности са възможните изходи от различните операции.

3.2. Банкомат

```
typedef enum Language {
    BG,
    EN,
    ERR
} Language;
```

Enum съдържаща информация за избрания език.

```
typedef enum UIString {
    ADMIN_OR_CLIENT,
    AMOUNT,
    CARD_NAME,
    CARD_NUMBER,
    WITHDRAW,
    CASH_RECEIPT,
    PIN,
    CARD,
    EXIT
} UIString;
```

Enum съдържащ информация свързана с локализацията на приложението. Този enum се използва при избора на какъв език да се визуализира на потребителя.

```
void ui();
```

Функция показваща програмния интерфейс на потребителя, нейната роля е да избере езикът на банкомата и след това да навигира до интерфейса за клиент или за администратор, докато потребителя не реши да излезе от банкомата.

```
void client_ui(Language);
```

Функция показваща програмният интерфейс за клиент на банкомата. Чрез нея потребителя въвежда данните за транзакцията, които в последствие биват изпратени до банковият сървър.

Параметри:

- Language - Езикът на който да оперира банкомата.

```
void admin_ui(Language);
```

Функция показваща програмният интерфейс за администратор на банкомата. Чрез нея потребителя въвежда данните за нов клиент, които в последствие биват изпратени до банковият сървър.

Параметри:

- Language - Езикът на който да оперира банкомата.

```
Language choose_language();
```

Функция за избор на езикът на който да оперира банкомата.

Резултат:

- Language - Езикът на който да оперира банкомата.

```
char* get_ui_string(Language, UIString);
```

Функция за вземане на репрезентацията показвана на потребителският интерфейс.

Параметри:

- Language - Езикът на който да оперира банкомата.
- UIString - Ключът на низа, който ще бъде визуализиран на потребителя.

Резултат:

- char* - Низовата репрезентация, която ще бъде визуализирана на потребителя.

```
bool choose_yes_no(Language, UIString);
```

Функция за показване на въпрос на потребителя, чиито резултат е от булев тип.

Параметри:

- Language - Езикът на който да оперира банкомата.

- UIString - Ключът на низа, който ще бъде визуализиран на потребителя.

Резултат:

- bool - Въведеният от потребителя отговор на зададения въпрос.

```
char* read_card_number(Language);
```

Функция за чете на номер на карта от файл.

Параметри:

- Language - Езикът на който да оперира банкомата.

Резултат:

- char* - Номер на потребителската карта, прочетен от файл.

```
int initializeTCPConnection(unsigned short);
```

Функция за инициализиране на tcp връзка по зададен порт.

Параметри:

- unsigned short - порт по който ще бъде инициализирана връзката.

Резултат:

- int - информация за сокета, към който е инициализирана връзка.

```
void closeTCPConnection();
```

Функция за затваряне на tcp връзката.

```
void sendClientTCPRequest(Transaction);
```

Функция за изпращане на заявка от нормален потребител.

Параметри:

- Transaction - Информация за операцията на транзакцията.

```
void sendAdminTCPRequest(Client);
```

Функция за изпращане на заявка от потребител, които е администратор.

Параметри:

- Client - Информация за клиента, които да бъде създаден.

```
int osocket_client;
int osocket_admin;
```

Глобални променливи, съдържащи информация за свързаните с банковият сървър сокети.

3.3. Банков сървър

```
void fillDB();
```
Функция имаща ролята да инициализира базата данни с началните данни в нея.

```
void loadDB();
```
Функция имаща ролята да зареди базата данни от файл.

```
void storeDB();
```
Функция имаща ролята да запази базата данни във файл.

```
int initializeTCPConnection(unsigned short);
```
Функция за инициализиране на tcp връзка по зададен порт.

Параметри:

- unsigned short - порт по който ще бъде инициализирана връзката.

Резултат:

- int - информация за сокета, към който е инициализирана връзка.

```
void closeTCPConnection();
```
Функция за затваряне на tcp връзката, запазване на данните в базата и унищожаване на мутекса за операции свързани с промяна на базата от данни.

```
void handleTCPConnection();
```
Функция имаща ролята да създаде нишките, които ще се грижат за приемане на данни от клиентският банкомат.

```
void *handleClientTCPConnection(void *);
```
Функция имаща ролята да чака и обработва заявка от обикновен потребител на системата.

Параметри:

- void* - Идентификатор на нишката.

```
void *handleAdminTCPConnection(void *);
```
Функция имаща ролята да чака и обработва заявка от потребител, който е администратор на системата.

Параметри:

- void* - Идентификатор на нишката.

```
void addClientToDB(Client);
```

Функция за добавяне на потребител на системата към базата от данни.

Параметри:

- Client - потребител на системата, който да бъде добавен към базата от данни.

```
int findClient(Transaction);
```

Функция за намиране на клиент по транзация.

Параметри:

- Transaction - Информация за операцията на транзакцията.

Резултат:

- int - индекс на потребителя в базата от данни.

```
int findClientByCard(char*);
```

Функция за намиране на клиент по номер на банковата карта.

Параметри:

- char* - номер на банковата карта.

Резултат:

- int - индекс на потребителя в базата от данни.

```
bool isClientNull(int);
```

Функция за проверка дали клиента съществува в базата от данни.

Параметри:

- int - индекс на клиент в базата от данни.

Резултат:

- bool - false или true на база дали клиента се намира в базата от данни.

```
Client *db;
int db_count = 0;
```

Глобални променливи съдържащи информация за базата от данни - самите данни и размерът на тази база.

```
int osocket[2];
int nsocket_client;
int nsocket_admin;
```

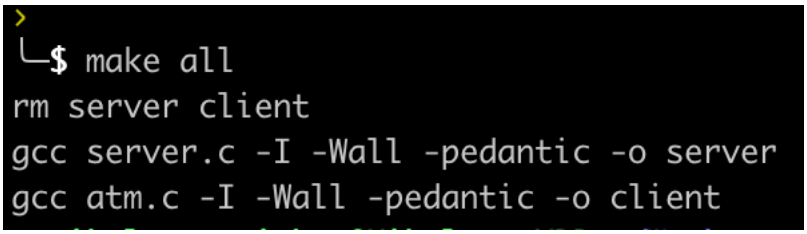
Глобални променливи, съдържащи информация за свързаните с банковият сървър сокети.

```
pthread_mutex_t client_lock;
```

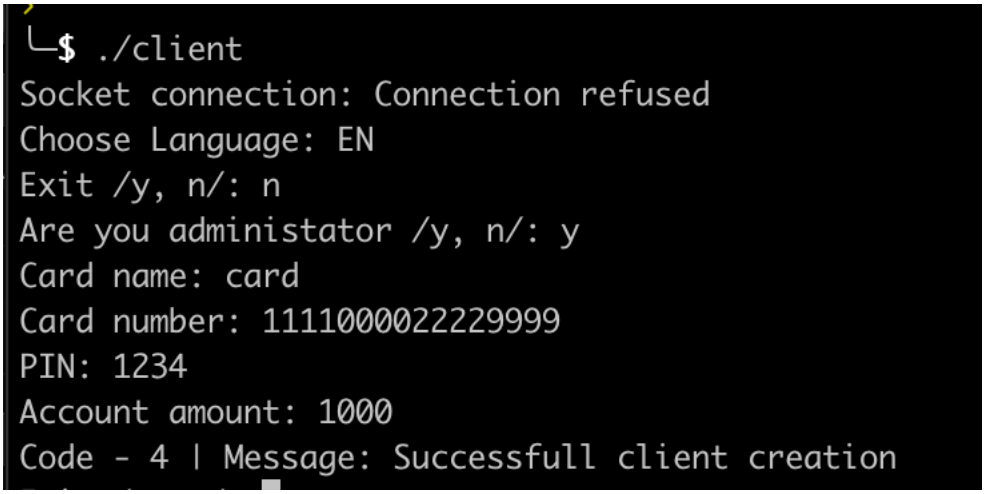
Глобална променлива съдържаща информация за състоянието на мютекса използван за извършване на операции по промяна на базата от данни.

4. Експериментални данни

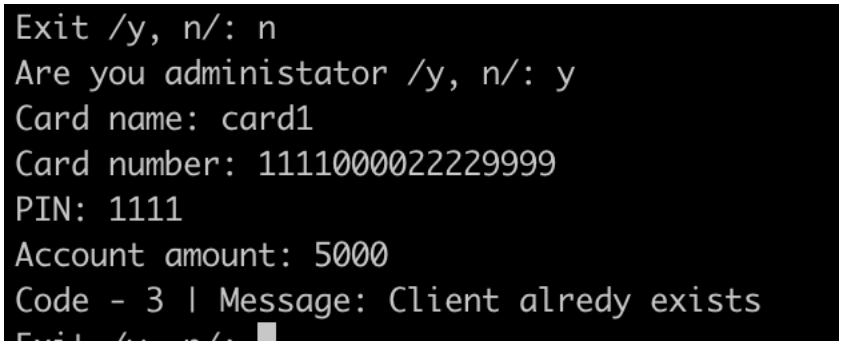
4.1. Компилиране на приложението.



4.1. Създаване на потребител.



4.2. Опит за създаване на съществуващ потребител.



4.3. Транзакция с несъщесвуващ потребител.

```
└─$ cat card1
9999000022229999
```

```
Exit /y, n/: n
Are you administator /y, n/: n
Card: card1
PIN: 1111
Withdraw Amount /by choosing 0 you can observe your account amount/: 50
Do you want cash receipt /y, n/: n
Code - 0 | Message: Missing user
```

4.4. Транзакция с по-голям размер на тегленето от това на потребителя.

```
Exit /y, n/: n
Are you administator /y, n/: n
Card: card
PIN: 1234
Withdraw Amount /by choosing 0 you can observe your account amount/: 2000
Do you want cash receipt /y, n/: n
Code - 1 | Message: Not enough money
```

4.5. Успешна транзакция

```
Exit /y, n/: n
Are you administator /y, n/: n
Card: card
PIN: 1234
Withdraw Amount /by choosing 0 you can observe your account amount/: 50
Do you want cash receipt /y, n/: n
Code - 2 | Message: Money in account - 950 | Withdrawen money - 50
```

4.6. Справка за състоянието на банковата сметка със запазване на резултата.

```
Exit /y, n/: n
Are you administator /y, n/: n
Card: card
PIN: 1234
Withdraw Amount /by choosing 0 you can observe your account amount/: 0
Do you want cash receipt /y, n/: y
Code - 2 | Message: Money in account - 950 | Withdrawen money - 0
```

```
└$ cat cash_receipt
Money in account - 950 | Withdrawen money - 0%
```

4.7. Изход.

```
Exit /y, n/: y
```

```
└$ ./server
DB loaded
Server ready
Socket recieve: Connection reset by peer
Socket recieve: Connection reset by peer
```

4.8. Проверка, че базата данни е запазена успешно.

```
└$ ./server
DB loaded
Server ready
Socket recieve: Connection reset by peer
```

```
└$ ./client
Socket connection: Connection refused
Choose Language: BG
Изход /y, n/: n
Администратор ли сте /y, n/: n
Карта: card
Пин: 1234
Размер на теглене /при избиране на 0 може да видите наличността по сметката си/: 0
Изкаш ли касов бон /y, n/: n
Code - 2 | Message: Money in account - 950 | Withdrawen money - 0
Изход /y, n/: y
```

5. Изходен код

https://github.com/nikolaystanishev/tu-system-programming/tree/main/course-work