**Отчет**

**по лабораторной работе №6-8.**

**3 семестр**

|  |  |
| --- | --- |
| **Студент:** | **Фирфаров А.С.** |
| **Группа:** | **8O-208Б** |
| **Преподаватель** | **Миронов Е.С.** |

**Цель работы**

Целью является приобретение практических навыков в:

* Управлении серверами сообщений (№6)
* Применение отложенных вычислений (№7)
* Интеграция программных систем друг с другом (№8)

**Задание**

Реализовать клиент-серверную систему по асинхронной обработке запросов. Необходимо составить программы сервера и клиента. При запуске сервер и клиент должны быть настраиваемы, то есть должна быть возможность поднятия на одной ЭВМ нескольких серверов по обработке данных и нескольких клиентов, которые к ним относятся. Все общение между процессами сервера и клиентов должно осуществляться через сервер сообщений.

Серверное приложение – банк. Клиентское приложение клиент банка. Клиент может отправить какую-то денежную сумму в банк на хранение. Клиент также может запросить из банка произвольную сумму. Клиенты могут посылать суммы на счета других клиентов. Запросить собственный счет. При снятии должна производиться проверка на то, что у клиента достаточно денег для снятия денежных средств. Идентификатор клиента задается во время запуска клиентского приложения, как и адрес банка. Считать, что идентификаторы при запуске клиентов будут уникальными.

**Вариант задания**

**Вариант 8**: **Внутреннее хранилище сервера** – список, **тип ключа клиента** – строка, **дополнительные возможности сервера** - каждый клиент обладает двумя счетами. Дебетовым и кредитным. Когда заканчивается дебетовый счет, клиент может получать суммы из кредитного счета. При этом в «минус» можно уйти только до определенного порога. Важно, что клиент может, как пополнять средства, так и брать с разных счетов.

**Описание алгоритма**

Для взаимодействия между клиентом и сервером был выбран паттерн Request-Response. Клиент подключается к банку по адресу, и далее может делать запросы на зачисление денег на дебетовый счет, погашения кредита, перевода денег другому клиенту банка, просмотра состояния счетов, снятия средств. Клиент формирует сообщение - запрос, отправляет его серверу-банку и ждет ответа. Банк принимает запрос от клиента, совершает необходимые операции и отправляет клиенту ответ.

**Текст программы**

**client.c**

#include <time.h>

#include <string.h>

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include "zmq.h"

typedef struct MD {

char clientId[400];

int action;

int sum;

int credit;

char anotherClientId[400];

char recvMessage[256];

void \*client;

} MessageData;

int SendRecv(MessageData\* ms) {

int check = 0;

zmq\_msg\_t zmqMessage;

zmq\_msg\_init\_size(&zmqMessage, sizeof(MessageData));

memcpy(zmq\_msg\_data(&zmqMessage), ms, sizeof(MessageData));

zmq\_msg\_send(&zmqMessage, ms->client, 0);

zmq\_msg\_close(&zmqMessage);

zmq\_msg\_init(&zmqMessage);

zmq\_msg\_recv(&zmqMessage, ms->client, 0);

ms = (MessageData\*)zmq\_msg\_data(&zmqMessage);

if ((check = strlen(ms->recvMessage)) < 2) {

zmq\_msg\_close(&zmqMessage);

return -1;

}

printf("%s\n", ms->recvMessage);

zmq\_msg\_close(&zmqMessage);

return 0;

}

int main() {

char clientId[400];

int bankId;

char bankAdress[256];

void \*context = zmq\_ctx\_new();

printf("Введите логин клиента\n");

scanf("%s",clientId);

printf("Введите адрес банка\n");

scanf("%d",&bankId);

sprintf(bankAdress, "%s%d", "tcp://localhost:", bankId);

void \*socket = zmq\_socket(context, ZMQ\_REQ);

zmq\_connect(socket, bankAdress);

int action = 0;

puts("---------------------MENU-------------------\n");

puts(" 1 - Положить деньги на счет \n");

puts(" 2 - Погасить кредит \n");

puts(" 3 - Перевести другому клиенту \n");

puts(" 4 - Запросить дебетовый счет \n");

puts(" 5 - Запросить задолженность по кредиту \n");

puts(" 6 - Снять деньги \n");

puts(" 7 - Выход \n");

puts("--------------------------------------------\n");

do {

printf("\n");

scanf("%d",&action);

MessageData\* ms = (MessageData\*)malloc(sizeof(MessageData));

ms->action = action;

ms->client = socket;

strncpy(ms->clientId , clientId, 400);

switch(action) {

case 1: {

int check = 0;

printf("Введите сумму: ");

scanf("%d", &(ms->sum));

if ((check = SendRecv(ms)) == -1) {

SendRecv(ms);

}

break;

}

case 2: {

int check = 0;

printf("Введите сумму: ");

scanf("%d", &(ms->sum));

if ((check = SendRecv(ms)) == -1) {

SendRecv(ms);

}

break;

}

case 3: {

int check = 0;

printf("Введите сумму: ");

scanf("%d", &(ms->sum));

printf("Введите ID клиента: ");

scanf("%s", (ms->anotherClientId));

if ((check = SendRecv(ms)) == -1) {

SendRecv(ms);

}

break;

}

case 4: {

int check = 0;

if ((check = SendRecv(ms)) == -1) {

SendRecv(ms);

}

break;

}

case 5: {

int check = 0;

if ((check = SendRecv(ms)) == -1) {

SendRecv(ms);

}

break;

}

case 6: {

int check = 0;

printf("Введите сумму: ");

scanf("%d", &(ms->sum));

if ((check = SendRecv(ms)) == -1) {

SendRecv(ms);

}

break;

}

case 7: {

break;

}

default: {

printf("Нет вариантов\n");

}

}

} while(action != 7);

zmq\_close(socket);

zmq\_ctx\_destroy(context);

return 0;

}

**bank.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <assert.h>

#include "zmq.h"

int limit = 10000;

typedef struct MD {

char clientId[400];

int action;

int sum;

int credit;

char anotherClientId[400];

char recvMessage[256];

void \*client;

} MessageData;

typedef struct \_client {

char clientId[400];

int sum;

int credit;

struct \_client \*next;

} Client;

typedef struct client\_list {

Client\* head;

Client\* tail;

int size;

} ClientList;

void Add(ClientList \*list, char\* clientId);

Client\* Find(ClientList \*list, char\* clientId);

void Destroy(ClientList \*list);

char\* ClientIncreaseDeb(ClientList \*list, char\* cl, int sum);

char\* ClientIncreaseCred(ClientList \*list, char\* cl, int sum);

char\* ClientDecreaseDeb(ClientList \*list, char\* cl, int sum);

int ClientDecreaseCred(Client\* cur, int sum);

char\* SendMoney(ClientList \*list, char\* anotherClientId, char\* client, int sum);

int CheckDeb(ClientList \*list, char\* cl);

int CheckCred(ClientList \*list, char\* cl);

void SendRecv(MessageData\* ms);

int main() {

ClientList\* data = (ClientList\*)malloc(sizeof(ClientList));

data->head = NULL;

data->tail = NULL;

data->size = 0;

int action = 0;

void \*context = zmq\_ctx\_new();

void \*socket = zmq\_socket(context, ZMQ\_REP);

int bank;

char adress[256];

printf("Введите адрес банка\n");

scanf("%d",&bank);

sprintf(adress, "%s%d", "tcp://\*:", bank);

zmq\_bind(socket, adress);

while(1) {

char info[256];

zmq\_msg\_t message;

zmq\_msg\_init(&message);

zmq\_msg\_recv(&message, socket, 0);

MessageData \*md = (MessageData\*)zmq\_msg\_data(&message);

zmq\_msg\_close(&message);

switch (md->action) {

case 1: {

strncpy(info, ClientIncreaseDeb(data, md->clientId, md->sum), 256);

memcpy(md->recvMessage, info, 256);

zmq\_msg\_init\_size(&message, sizeof(MessageData));

memcpy(zmq\_msg\_data(&message), md, sizeof(MessageData));

zmq\_msg\_send(&message, socket, 0);

zmq\_msg\_close(&message);

break;

}

case 2: {

strncpy(info, ClientIncreaseCred(data, md->clientId, md->sum), 256);

memcpy(md->recvMessage, info, 256);

zmq\_msg\_init\_size(&message, sizeof(MessageData));

memcpy(zmq\_msg\_data(&message), md, sizeof(MessageData));

zmq\_msg\_send(&message, socket, 0);

zmq\_msg\_close(&message);

break;

}

case 3: {

strncpy(info, SendMoney(data,md->anotherClientId ,md->clientId, md->sum), 256);

memcpy(md->recvMessage, info, 256);

zmq\_msg\_init\_size(&message, sizeof(MessageData));

memcpy(zmq\_msg\_data(&message), md, sizeof(MessageData));

zmq\_msg\_send(&message, socket, 0);

zmq\_msg\_close(&message);

break;

}

case 4: {

int info;

char inf\_m[256];

if ((info = CheckDeb(data, md->clientId)) == -1) {

strcpy(inf\_m, "Вы не клиент банка\n");

}

else {

sprintf(inf\_m, "%s%d", "Дебетовый счет: ", info);

}

memcpy(md->recvMessage, inf\_m, 256);

zmq\_msg\_init\_size(&message, sizeof(MessageData));

memcpy(zmq\_msg\_data(&message), md, sizeof(MessageData));

zmq\_msg\_send(&message, socket, 0);

zmq\_msg\_close(&message);

break;

}

case 5: {

int info;

char inf\_m[256];

if ((info = CheckCred(data, md->clientId)) == -1) {

strcpy(inf\_m, "Вы не клиент банка\n");

}

else {

sprintf(inf\_m, "%s%d", "Задолженность: ", info);

}

memcpy(md->recvMessage, inf\_m, 256);

zmq\_msg\_init\_size(&message, sizeof(MessageData));

memcpy(zmq\_msg\_data(&message), md, sizeof(MessageData));

zmq\_msg\_send(&message, socket, 0);

zmq\_msg\_close(&message);

break;

}

case 6: {

strncpy(info, ClientDecreaseDeb(data, md->clientId, md->sum), 256);

memcpy(md->recvMessage, info, 256);

zmq\_msg\_init\_size(&message, sizeof(MessageData));

memcpy(zmq\_msg\_data(&message), md, sizeof(MessageData));

zmq\_msg\_send(&message, socket, 0);

zmq\_msg\_close(&message);

break;

}

}

}

}

void Add(ClientList \*list, char\* clientId) {

Client \*cl = (Client\*)malloc(sizeof(Client));

strncpy(cl->clientId, clientId, 400);

cl->sum = 0;

cl->credit = 0;

cl->next = NULL;

if (list->size == 0) {

list->head = cl;

list->tail = cl;

}

else {

list->tail->next = cl;

list->tail = cl;

}

++(list->size);

}

Client\* Find(ClientList \*list, char\* clientId) {

Client \*cur = list->head;

int check = 0;

while(cur != NULL) {

if ((check = strcmp(cur->clientId, clientId)) == 0) {

return cur;

}

cur = cur->next;

}

return NULL;

}

char\* ClientIncreaseDeb(ClientList \*list, char\* cl, int sum) {

int add = 0;

Client \*cur = Find(list, cl);

if (cur == NULL) {

Add(list, cl);

add = 1;

}

cur = Find(list, cl);

cur->sum = cur->sum + sum;

if (add == 1) {

return "Клиент добавлен. Деньги зачислены на дебетовый счет\n";

}

else {

return "Деньги зачислены на дебетовый счет\n";

}

}

char\* ClientIncreaseCred(ClientList \*list, char\* cl, int sum) {

Client \*cur = Find(list, cl);

if (cur == NULL) {

return "Не является клиентом банка\n";

}

if (cur->credit == 0) {

return "Нет кредита для погашения\n";

}

if (sum > cur->credit) {

return "Сумма больше текущего кредита\n";

}

cur->credit = cur->credit - sum;

if (cur->credit == 0) {

return "Кредит погашен\n";

}

return "Часть кредита погашена\n";

}

char\* SendMoney(ClientList \*list, char\* anotherClientId, char\* client, int sum) {

char info[256];

Client \*client\_an = Find(list, anotherClientId);

if (client\_an == NULL) {

return "Получатель не является клиентом банка\n";

}

Client \*cur = Find(list, client);

if (cur == NULL) {

return "Отправитель не является клиентом банка\n";

}

if (cur->sum < sum) {

return "Недостаточно средств для перевода\n";

}

cur->sum = cur->sum - sum;

client\_an->sum = client\_an->sum + sum;

return "Деньги переведены\n";

}

int CheckDeb(ClientList \*list, char\* cl) {

Client \*cur = Find(list, cl);

if (cur == NULL) {

return -1;

}

return cur->sum;

}

int CheckCred(ClientList \*list, char\* cl) {

Client \*cur = Find(list, cl);

if (cur == NULL) {

return -1;

}

return cur->credit;

}

char\* ClientDecreaseDeb(ClientList \*list, char\* cl, int sum) {

Client \*cur = Find(list, cl);

if (cur == NULL) {

return "Вы не являетесь клиентом банка\n";

}

if (cur->sum < sum) {

int check = 0;

int credit = sum - cur->sum;

cur->sum = 0;

check = ClientDecreaseCred(cur, credit);

if (check == -1) {

return "Превышен лимит\n";

}

if (check == 0) {

return "Деньги сняты. Был взят кредит\n";

}

}

else {

cur->sum = cur->sum - sum;

return "Деньги сняты\n";

}

}

int ClientDecreaseCred(Client\* cur, int sum) {

if ((cur->credit + sum) > limit) {

return -1;

}

cur->credit = cur->credit + sum;

return 0;

}

**Тестирование**

user@lubuntu:~$ cd OC

user@lubuntu:~/OC$ cd lab

bash: cd: lab: Нет такого файла или каталога

user@lubuntu:~/OC$ cd lab6

user@lubuntu:~/OC/lab6$ ./client.exe

Введите логин клиента

per1

Введите адрес банка

4040

---------------------MENU-------------------

1 - Положить деньги на счет

2 - Погасить кредит

3 - Перевести другому клиенту

4 - Запросить дебетовый счет

5 - Запросить задолженность по кредиту

6 - Снять деньги

7 - Выход

--------------------------------------------

1

Введите сумму: 5000

Клиент добавлен. Деньги зачислены на дебетовый счет

^C

user@lubuntu:~/OC/lab6$ clear

[3;J

user@lubuntu:~/OC/lab6$ ./client.exe

Введите логин клиента

per1

Введите адрес банка

4040

---------------------MENU-------------------

1 - Положить деньги на счет

2 - Погасить кредит

3 - Перевести другому клиенту

4 - Запросить дебетовый счет

5 - Запросить задолженность по кредиту

6 - Снять деньги

7 - Выход

--------------------------------------------

1

Введите сумму: 5000

Клиент добавлен. Деньги зачислены на дебетовый счет

4

Дебетовый счет: 5000

6

Введите сумму: 6000

Деньги сняты. Был взят кредит

4

Дебетовый счет: 0

5

Задолженность: 1000

2

Введите сумму: 1500

Сумма больше текущего кредита

2

Введите сумму: 1000

Кредит погашен

1

Введите сумму: 7500

Деньги зачислены на дебетовый счет

3

Введите сумму: 8000

Введите ID клиента: person

Получатель не является клиентом банка

3

Введите сумму: 5000

Введите ID клиента: person1

Деньги переведены

7

user@lubuntu:~/OC/lab6$ ./client.exe

Введите логин клиента

per1

Введите адрес банка

4040

---------------------MENU-------------------

1 - Положить деньги на счет

2 - Погасить кредит

3 - Перевести другому клиенту

4 - Запросить дебетовый счет

5 - Запросить задолженность по кредиту

6 - Снять деньги

7 - Выход

--------------------------------------------

4

Дебетовый счет: 2500

6

Введите сумму: 2500

Деньги сняты

4

Дебетовый счет: 0

5

Задолженность: 0

7

user@lubuntu:~/OC/lab6$

**ltrace**

user@lubuntu:~/OC/lab6$ ltrace ./client.exe

\_\_libc\_start\_main(0x400e09, 1, 0x7fff73261d58, 0x401290 <unfinished ...>

zmq\_ctx\_new(1, 0x7fff73261d58, 0x7fff73261d68, 224) = 0x191dc20

puts("\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\273\320\276\320\263\320\270\320\275 \320\272\320\273\320\270"...Введите логин клиента

) = 41

\_\_isoc99\_scanf(0x401341, 0x7fff73261ad0, 0x7fd6242d3780, 0x7fd6240042c0person1

) = 1

puts("\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\260\320\264\321\200\320\265\321\201 \320\261\320\260\320\275"...Введите адрес банка

) = 37

\_\_isoc99\_scanf(0x40136d, 0x7fff73261998, 0x7fd6242d3780, 0x7fd6240042c04040

) = 1

sprintf("tcp://localhost:4040", "%s%d", "tcp://localhost:", 4040) = 20

zmq\_socket(0x191dc20, 3, 0x401385, 0x7fffffeb) = 0x1921120

zmq\_connect(0x1921120, 0x7fff732619d0, 0x7fff732619d0, 0) = 0

puts("---------------------MENU-------"...---------------------MENU-------------------

) = 46

puts(" 1 - \320\237\320\276\320\273\320\276\320\266\320\270\321\202\321\214 \320\264\320"... 1 - Положить деньги на счет

) = 66

puts(" 2 - \320\237\320\276\320\263\320\260\321\201\320\270\321\202\321"... 2 - Погасить кредит

) = 60

puts(" 3 - \320\237\320\265\321\200\320\265\320\262\320\265\321\201\321\202\320\270 \320"... 3 - Перевести другому клиенту

) = 69

puts(" 4 - \320\227\320\260\320\277\321\200\320\276\321\201\320\270\321\202\321\214"... 4 - Запросить дебетовый счет

) = 68

puts(" 5 - \320\227\320\260\320\277\321\200\320\276\321\201\320\270\321\202\321\214 \320\267\320\260"... 5 - Запросить задолженность по кредиту

) = 77

puts(" 6 - \320\241\320\275\321\217\321\202\321\214 \320\264"... 6 - Снять деньги

) = 57

puts(" 7 - \320\222\321\213\321\205\320\276\320\264"... 7 - Выход

) = 51

puts("--------------------------------"...--------------------------------------------

) = 46

putchar(10, 0x191dfd0, 0x7fd6242d3780, 0x7fd6240042dd

) = 10

\_\_isoc99\_scanf(0x40136d, 0x7fff7326199c, 10, 0x7fd6240042dd1

) = 1

malloc(1080) = 0x192b260

strncpy(0x192b260, "person1", 400) = 0x192b260

printf("\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \321\201\321\203\320\274\320\274\321\203: ") = 27

\_\_isoc99\_scanf(0x40136d, 0x192b3f4, 0, 27Введите сумму: 5000

) = 1

zmq\_msg\_init\_size(0x7fff73261930, 1080, 0x7fd6242d3790, 16) = 0

zmq\_msg\_data(0x7fff73261930, 0x192bb00, 0x192b6c8, 0x7fd6242d1b20) = 0x192b6c8

memcpy(0x192b6c8, "person1\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 1080) = 0x192b6c8

zmq\_msg\_send(0x7fff73261930, 0x1921120, 0, 0x1921120) = 1080

zmq\_msg\_close(0x7fff73261930, 0x7fff73261750, 0, 0) = 0

zmq\_msg\_init(0x7fff73261930, 0x7fff73261750, 0, 0) = 0

zmq\_msg\_recv(0x7fff73261930, 0x1921120, 0, 0x1921120) = 1080

zmq\_msg\_data(0x7fff73261930, 0x7fff73261930, 1080, 0) = 0x7fd61c006648

strlen("\320\232\320\273\320\270\320\265\320\275\321\202 \320\264\320\276\320\261\320\260\320\262\320\273\320\265\320\275. \320"...) = 96

puts("\320\232\320\273\320\270\320\265\320\275\321\202 \320\264\320\276\320\261\320\260\320\262\320\273\320\265\320\275. \320"...Клиент добавлен. Деньги зачислены на дебетовый счет

) = 97

zmq\_msg\_close(0x7fff73261930, 0x191dfd0, 0x7fd6242d3780, 0x7fd6240042dd) = 0

putchar(10, 0, 0, 0x7fd6240042dd

) = 10

\_\_isoc99\_scanf(0x40136d, 0x7fff7326199c, 10, 0x7fd6240042dd4

) = 1

malloc(1080) = 0x192b6a0

strncpy(0x192b6a0, "person1", 400) = 0x192b6a0

zmq\_msg\_init\_size(0x7fff73261930, 1080, 0, 0x7fd623fb3cf0) = 0

zmq\_msg\_data(0x7fff73261930, 0x192bfa0, 0x192bb68, 0x7fd6242d1b20) = 0x192bb68

memcpy(0x192bb68, "person1\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 1080) = 0x192bb68

zmq\_msg\_send(0x7fff73261930, 0x1921120, 0, 0x1921120) = 1080

zmq\_msg\_close(0x7fff73261930, 0x7fff73261750, 0, 0) = 0

zmq\_msg\_init(0x7fff73261930, 0x7fff73261750, 0, 0) = 0

zmq\_msg\_recv(0x7fff73261930, 0x1921120, 0, 0x1921120) = 1080

zmq\_msg\_data(0x7fff73261930, 0x7fff73261930, 1080, 0) = 0x7fd61c006648

strlen("\320\224\320\265\320\261\320\265\321\202\320\276\320\262\321\213\320\271 \321\201\321\207\320\265\321\202: 500"...) = 33

puts("\320\224\320\265\320\261\320\265\321\202\320\276\320\262\321\213\320\271 \321\201\321\207\320\265\321\202: 500"...Дебетовый счет: 5000

) = 34

zmq\_msg\_close(0x7fff73261930, 0x191dfd0, 0x7fd6242d3780, 0x7fd6240042dd) = 0

putchar(10, 0, 0, 0x7fd6240042dd

) = 10

\_\_isoc99\_scanf(0x40136d, 0x7fff7326199c, 10, 0x7fd6240042dd5

) = 1

malloc(1080) = 0x192bb40

strncpy(0x192bb40, "person1", 400) = 0x192bb40

zmq\_msg\_init\_size(0x7fff73261930, 1080, 0, 0x7fd623fb3cf0) = 0

zmq\_msg\_data(0x7fff73261930, 0x192c3e0, 0x192bfa8, 0x7fd6242d1b20) = 0x192bfa8

memcpy(0x192bfa8, "person1\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 1080) = 0x192bfa8

zmq\_msg\_send(0x7fff73261930, 0x1921120, 0, 0x1921120) = 1080

zmq\_msg\_close(0x7fff73261930, 0x7fff73261750, 0, 0) = 0

zmq\_msg\_init(0x7fff73261930, 0x7fff73261750, 0, 0) = 0

zmq\_msg\_recv(0x7fff73261930, 0x1921120, 0, 0x1921120) = 1080

zmq\_msg\_data(0x7fff73261930, 0x7fff73261930, 1080, 0) = 0x7fd61c006648

strlen("\320\227\320\260\320\264\320\276\320\273\320\266\320\265\320\275\320\275\320\276\321\201\321\202\321\214: 0") = 29

puts("\320\227\320\260\320\264\320\276\320\273\320\266\320\265\320\275\320\275\320\276\321\201\321\202\321\214: 0"Задолженность: 0

) = 30

zmq\_msg\_close(0x7fff73261930, 0x191dfd0, 0x7fd6242d3780, 0x7fd6240042dd) = 0

putchar(10, 0, 0, 0x7fd6240042dd

) = 10

\_\_isoc99\_scanf(0x40136d, 0x7fff7326199c, 10, 0x7fd6240042dd6

) = 1

malloc(1080) = 0x192bf80

strncpy(0x192bf80, "person1", 400) = 0x192bf80

printf("\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \321\201\321\203\320\274\320\274\321\203: ") = 27

\_\_isoc99\_scanf(0x40136d, 0x192c114, 0, 27Введите сумму: 7000

) = 1

zmq\_msg\_init\_size(0x7fff73261930, 1080, 0x7fd6242d3790, 16) = 0

zmq\_msg\_data(0x7fff73261930, 0x192c820, 0x192c3e8, 0x7fd6242d1b20) = 0x192c3e8

memcpy(0x192c3e8, "person1\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 1080) = 0x192c3e8

zmq\_msg\_send(0x7fff73261930, 0x1921120, 0, 0x1921120) = 1080

zmq\_msg\_close(0x7fff73261930, 0x7fff73261750, 0, 0) = 0

zmq\_msg\_init(0x7fff73261930, 0x7fff73261750, 0, 0) = 0

zmq\_msg\_recv(0x7fff73261930, 0x1921120, 0, 0x1921120) = 1080

zmq\_msg\_data(0x7fff73261930, 0x7fff73261930, 1080, 0) = 0x7fd61c006648

strlen("\320\224\320\265\320\275\321\214\320\263\320\270 \321\201\320\275\321\217\321\202\321\213. \320\221\321\213\320\273 "...) = 54

puts("\320\224\320\265\320\275\321\214\320\263\320\270 \321\201\320\275\321\217\321\202\321\213. \320\221\321\213\320\273 "...Деньги сняты. Был взят кредит

) = 55

zmq\_msg\_close(0x7fff73261930, 0x191dfd0, 0x7fd6242d3780, 0x7fd6240042dd) = 0

putchar(10, 0, 0, 0x7fd6240042dd

) = 10

\_\_isoc99\_scanf(0x40136d, 0x7fff7326199c, 10, 0x7fd6240042dd5

) = 1

malloc(1080) = 0x192c3c0

strncpy(0x192c3c0, "person1", 400) = 0x192c3c0

zmq\_msg\_init\_size(0x7fff73261930, 1080, 0, 0x7fd623fb3cf0) = 0

zmq\_msg\_data(0x7fff73261930, 0x192cc60, 0x192c828, 0x7fd6242d1b20) = 0x192c828

memcpy(0x192c828, "person1\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 1080) = 0x192c828

zmq\_msg\_send(0x7fff73261930, 0x1921120, 0, 0x1921120) = 1080

zmq\_msg\_close(0x7fff73261930, 0x7fff73261750, 0, 0) = 0

zmq\_msg\_init(0x7fff73261930, 0x7fff73261750, 0, 0) = 0

zmq\_msg\_recv(0x7fff73261930, 0x1921120, 0, 0x1921120) = 1080

zmq\_msg\_data(0x7fff73261930, 0x7fff73261930, 1080, 0) = 0x7fd61c006648

strlen("\320\227\320\260\320\264\320\276\320\273\320\266\320\265\320\275\320\275\320\276\321\201\321\202\321\214: 2000"...) = 32

puts("\320\227\320\260\320\264\320\276\320\273\320\266\320\265\320\275\320\275\320\276\321\201\321\202\321\214: 2000"...Задолженность: 2000

) = 33

zmq\_msg\_close(0x7fff73261930, 0x191dfd0, 0x7fd6242d3780, 0x7fd6240042dd) = 0

putchar(10, 0, 0, 0x7fd6240042dd

) = 10

\_\_isoc99\_scanf(0x40136d, 0x7fff7326199c, 10, 0x7fd6240042dd7

) = 1

malloc(1080) = 0x192c800

strncpy(0x192c800, "person1", 400) = 0x192c800

zmq\_close(0x1921120, 8, 0, 0x7fd623fb3cf0) = 0

zmq\_ctx\_destroy(0x191dc20, 0x7fff73261900, 8, 0x7fd6242e74bd) = 0

+++ exited (status 0) +++

user@lubuntu:~/OC/lab6$

user@lubuntu:~/OC/lab6$ ltrace ./bank.exe

\_\_libc\_start\_main(0x400bb6, 1, 0x7ffd0453c038, 0x401910 <unfinished ...>

malloc(24) = 0x143cc20

zmq\_ctx\_new(0, 0x143cc30, 0x143cc20, 0x7f2a9abd5b20) = 0x143cc40

zmq\_socket(0x143cc40, 4, 2510, 0) = 0x143f920

puts("\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\260\320\264\321\200\320\265\321\201 \320\261\320\260\320\275"...Введите адрес банка

) = 37

\_\_isoc99\_scanf(0x4019bd, 0x7ffd0453bcd0, 0x7f2a9abd7780, 0x7f2a9a9082dd4040

) = 1

sprintf("tcp://\*:4040", "%s%d", "tcp://\*:", 4040) = 12

zmq\_bind(0x143f920, 0x7ffd0453bd40, 0x7ffd0453bd40, 0x7ffffff3) = 0

zmq\_msg\_init(0x7ffd0453bd00, 0x7f2a9abd5b28, 0x14408c0, 0x143fd00) = 0

zmq\_msg\_recv(0x7ffd0453bd00, 0x143f920, 0, 0x143f920) = 1080

zmq\_msg\_data(0x7ffd0453bd00, 0x7ffd0453bd00, 1080, 0) = 0x7f2a9400df78

zmq\_msg\_close(0x7ffd0453bd00, 0x7ffd0453bd00, 102, 0) = 0

malloc(416) = 0x1440670

strncpy(0x1440670, "person1", 400) = 0x1440670

strcmp("person1", "person1") = 0

strncpy(0x7ffd0453be40, "\320\232\320\273\320\270\320\265\320\275\321\202 \320\264\320\276\320\261\320\260\320\262\320\273\320\265\320\275. \320"..., 256) = 0x7ffd0453be40

zmq\_msg\_init\_size(0x7ffd0453bd00, 1080, 31, 0) = 0

zmq\_msg\_data(0x7ffd0453bd00, 0x1442690, 0x1442258, 0x7f2a9abd5b20) = 0x1442258

memcpy(0x1442258, "person1\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 1080) = 0x1442258

zmq\_msg\_send(0x7ffd0453bd00, 0x143f920, 0, 0x143f920) = 1080

zmq\_msg\_close(0x7ffd0453bd00, 0x7ffd0453bb70, 8, 0) = 0

zmq\_msg\_init(0x7ffd0453bd00, 0x7ffd0453bb70, 0, 0) = 0

zmq\_msg\_recv(0x7ffd0453bd00, 0x143f920, 0, 0x143f920) = 1080

zmq\_msg\_data(0x7ffd0453bd00, 0x7ffd0453bd00, 1080, 0) = 0x7f2a9400df78

zmq\_msg\_close(0x7ffd0453bd00, 0x7ffd0453bd00, 102, 0) = 0

strcmp("person1", "person1") = 0

sprintf("\320\224\320\265\320\261\320\265\321\202\320\276\320\262\321\213\320\271 \321\201\321\207\320\265\321\202: 500"..., "%s%d", "\320\224\320\265\320\261\320\265\321\202\320\276\320\262\321\213\320\271 \321\201\321\207\320\265\321\202: ", 5000) = 33

zmq\_msg\_init\_size(0x7ffd0453bd00, 1080, 31, 0) = 0

zmq\_msg\_data(0x7ffd0453bd00, 0x1442690, 0x1442258, 0x7f2a9abd5b20) = 0x1442258

memcpy(0x1442258, "person1\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 1080) = 0x1442258

zmq\_msg\_send(0x7ffd0453bd00, 0x143f920, 0, 0x143f920) = 1080

zmq\_msg\_close(0x7ffd0453bd00, 0x7ffd0453bb70, 8, 0) = 0

zmq\_msg\_init(0x7ffd0453bd00, 0x7ffd0453bb70, 0, 0) = 0

zmq\_msg\_recv(0x7ffd0453bd00, 0x143f920, 0, 0x143f920) = 1080

zmq\_msg\_data(0x7ffd0453bd00, 0x7ffd0453bd00, 1080, 0) = 0x7f2a9400df78

zmq\_msg\_close(0x7ffd0453bd00, 0x7ffd0453bd00, 102, 0) = 0

strcmp("person1", "person1") = 0

sprintf("\320\227\320\260\320\264\320\276\320\273\320\266\320\265\320\275\320\275\320\276\321\201\321\202\321\214: 0", "%s%d", "\320\227\320\260\320\264\320\276\320\273\320\266\320\265\320\275\320\275\320\276\321\201\321\202\321\214: ", 0) = 29

zmq\_msg\_init\_size(0x7ffd0453bd00, 1080, 31, 0) = 0

zmq\_msg\_data(0x7ffd0453bd00, 0x1442690, 0x1442258, 0x7f2a9abd5b20) = 0x1442258

memcpy(0x1442258, "person1\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 1080) = 0x1442258

zmq\_msg\_send(0x7ffd0453bd00, 0x143f920, 0, 0x143f920) = 1080

zmq\_msg\_close(0x7ffd0453bd00, 0x7ffd0453bb70, 8, 0) = 0

zmq\_msg\_init(0x7ffd0453bd00, 0x7ffd0453bb70, 0, 0) = 0

zmq\_msg\_recv(0x7ffd0453bd00, 0x143f920, 0, 0x143f920) = 1080

zmq\_msg\_data(0x7ffd0453bd00, 0x7ffd0453bd00, 1080, 0) = 0x7f2a9400df78

zmq\_msg\_close(0x7ffd0453bd00, 0x7ffd0453bd00, 102, 0) = 0

strcmp("person1", "person1") = 0

strncpy(0x7ffd0453be40, "\320\224\320\265\320\275\321\214\320\263\320\270 \321\201\320\275\321\217\321\202\321\213. \320\221\321\213\320\273 "..., 256) = 0x7ffd0453be40

zmq\_msg\_init\_size(0x7ffd0453bd00, 1080, 31, 0) = 0

zmq\_msg\_data(0x7ffd0453bd00, 0x1442690, 0x1442258, 0x7f2a9abd5b20) = 0x1442258

memcpy(0x1442258, "person1\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 1080) = 0x1442258

zmq\_msg\_send(0x7ffd0453bd00, 0x143f920, 0, 0x143f920) = 1080

zmq\_msg\_close(0x7ffd0453bd00, 0x7ffd0453bb70, 8, 0) = 0

zmq\_msg\_init(0x7ffd0453bd00, 0x7ffd0453bb70, 0, 0) = 0

zmq\_msg\_recv(0x7ffd0453bd00, 0x143f920, 0, 0x143f920) = 1080

zmq\_msg\_data(0x7ffd0453bd00, 0x7ffd0453bd00, 1080, 0) = 0x7f2a9400df78

zmq\_msg\_close(0x7ffd0453bd00, 0x7ffd0453bd00, 102, 0) = 0

strcmp("person1", "person1") = 0

sprintf("\320\227\320\260\320\264\320\276\320\273\320\266\320\265\320\275\320\275\320\276\321\201\321\202\321\214: 2000"..., "%s%d", "\320\227\320\260\320\264\320\276\320\273\320\266\320\265\320\275\320\275\320\276\321\201\321\202\321\214: ", 2000) = 32

zmq\_msg\_init\_size(0x7ffd0453bd00, 1080, 31, 0) = 0

zmq\_msg\_data(0x7ffd0453bd00, 0x1442690, 0x1442258, 0x7f2a9abd5b20) = 0x1442258

memcpy(0x1442258, "person1\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 1080) = 0x1442258

zmq\_msg\_send(0x7ffd0453bd00, 0x143f920, 0, 0x143f920) = 1080

zmq\_msg\_close(0x7ffd0453bd00, 0x7ffd0453bb70, 8, 0) = 0

zmq\_msg\_init(0x7ffd0453bd00, 0x7ffd0453bb70, 0, 0) = 0

zmq\_msg\_recv(0x7ffd0453bd00, 0x143f920, 0, 0x143f920^C <no return ...>

--- SIGINT (Interrupt) ---

+++ killed by SIGINT +++

user@lubuntu:~/OC/lab6$

**Выводы**

При выполнении данной лабораторной работы была реализована клиент-серверная система на базе очередей сообщений. Благодаря работе с сервером сообщений я узнал о многих новых паттернах передачи сообщений, что может пригодиться в будущем при разработке приложения обмена сообщениями. Я выбрал паттерн REQ-RES, так как он показался мне наиболее простым в реализации. Использование очередей сообщений позволило значительно повысить отказоустойчивость, когда новые сообщения не отбрасываются при загруженном сервере, а хранятся в очереди. Минус в том, что определить точное время, через которое сервер отправит ответ, нельзя. Я считаю, что полученные в ходе данной лабораторной работе навыки важны, так как очереди сообщений широко используются в программировании и позволяют существенно упростить процесс обмена сообщениями между клиентом и сервером.