Отчет по лабораторной работе №6-8

По курсу «Операционные системы»

Работу выполнила

студентка группы 8О-208Б

Понагайбо А.О.

Сдано: _____

**Тема:** Управление серверами сообщений, применение отложенных вычислений.

**Цель работы:** Реализовать клиент-серверную систему по асинхронной обработке запросов. Необходимо составить программы сервера и клиента. При запуске сервер и клиент должны быть настраиваемы. Все общение между процессами сервера и клиентов должно осуществляться через сервер сообщений.

**Задание:** Вариант № 15.
*Сервер сообщений:* ZeroMQ.
*Внутреннее хранилище сервера:* вектор.
*Тип ключа клиента:* целое число 32 бита.
*Дополнительные возможности сервера:* сервера (банки) могут передавать средства клиентов друг другу. Считать, что у клиентов одинаковые идентификаторы между банками.

## Текст программы:

**vect.h**

```
#ifndef _VECT_H
#define _VECT_H
#define VECT_SIZE 20;
#define MULT 2;
typedef unsigned long long ull;

struct TNode {
    int id;
    ull sum;
};
typedef struct TNode* TNode;

struct TVector {
    int size;
    int top;
    TNode data;
    int first;
};
typedef struct TVector* TVector;

TVector CreateVector();
void DeleteVector(TVector* v);
TVector ResizeVector(TVector v);
int SizeOfVector(TVector v);
void PrintVector(TVector v);
void PushVector(TVector v, int curID, ull curSum);
TNode TopVector(TVector v);
TNode PopVector(TVector v);
```

```
TNode FindNode(TVector v, int name);
#endif
```

**vect.c**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "vect.h"

TVector CreateVector() {
    TVector out = NULL;
    out = malloc(sizeof(struct TVector));
    out->size = VECT_SIZE;
    out->top = 0;
    out->first = 0;
    out->data = (TNode)malloc(out->size * sizeof(struct TNode));
    return out;
}

void DeleteVector(TVector* v) {
    if (v == NULL) return;
    free((*v)->data);
    free(*v);
    *v = NULL;
}

TVector ResizeVector(TVector v) {
    v->size *= MULT;
    v->data = realloc(v->data, v->size * sizeof(struct TNode));
}

void PrintVector(TVector v) {
    if (SizeOfVector(v)) {
        int i;
        for (i = v->first; i < v->top; i++) printf("id: %d\tsum: %llu\n", v-
>data[i].id, v->data[i].sum);
    } else printf("Vector is empty.\n");
}

int SizeOfVector(TVector v) {
    return ((v->top) - (v->first));
}

void PushVector(TVector v, int curID, ull curSum) {
    if (v->top >= (v->size) - 1) ResizeVector(v);
    v->data[v->top].id = curID;
    v->data[v->top].sum = curSum;
    v->top++;
}

TNode TopVector(TVector v) {
    if (SizeOfVector(v)) return &(v->data[v->first]);
    else printf("Vector is empty.\n");
}

TNode PopVector(TVector v) {
    if (SizeOfVector(v)) {
        TNode el;
        el = &(v->data[v->first]);
        v->first++;
        return el;
    } else printf("Vector is empty.\n");
}

TNode FindNode(TVector v, int name) {
    if (SizeOfVector(v)) {
        int i;
        for (i = v->first; i < v->top; i++)
            if (v->data[i].id == name)
```

```
                return &v->data[i];
    }
    return NULL;
}
```

**client.c**

```
#include <string.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio_ext.h>
#include "zmq.h"

void Help() {
    printf("\tAvailable commands:\n>deposit SID SUM >withdraw SUM ");
    printf(">transfer BANK_ADR SUM >balance\n");
}

int main (int argc, char* argv[]) {
    if (argc != 3) {
        printf("Error! You should set client ID and bank adress.\n");
        return -1;
    }
    char* adr = (char*)malloc(32 * sizeof(char));
    strcpy(adr, "tcp://localhost:");
    strcat(adr, argv[2]);
    int sz = strlen(argv[1]);
    char* curID = (char*)malloc(sz);
    strcpy(curID, argv[1]);
    void* context = zmq_ctx_new();
    if (context == NULL) {
        printf("Can not create context.\n");
        return -1;
    }
    void* request = zmq_socket(context, ZMQ_REQ);
    if (request == NULL) {
        printf("Can not create socket (request).\n");
        return -1;
    }
    if (zmq_connect(request, adr) == -1) {
        printf("Can not connect.\n");
        return -1;
    }
    char tmp[100];
    Help();
    char msg[100];
    while (!feof(stdin)) {
        __fpurge(stdin);
        printf("\nEnter your command:\n>");
        fgets(tmp, 100, stdin);
        strcpy(msg, curID);
        strcat(msg, " ");
        strcat(msg, tmp);
        if (zmq_send(request, msg, 100, 0) == -1) {
            printf("Error! Can not send message.\n");
            return -1;
        }
        memset(tmp, 0, 100);
        if (zmq_recv(request, tmp, 100, 0) == -1) {
            printf("Error! Can not receive message.\n");
            return -1;
        }
        printf("%s\n", tmp);
    }
    zmq_close(request);
    zmq_ctx_destroy(context);
    return 0;
}
```

**banc.c**

```
#include <string.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include "zmq.h"
#include "vect.h"

int mainID;
void Deposit(void* socket, TVector vect, int clientID, ull value) {
    TNode cur = FindNode(vect, clientID);
    if (cur) cur->sum += value;
    else PushVector(vect, clientID, value);
    printf("Deposit: operation is done.\n");
    char msg[100];
    sprintf(msg, "Deposit: done. +%llu, id: %d", value, clientID);
    if (zmq_send(socket, msg, strlen(msg), 0) == -1)
        printf("Error! Can not send message (deposit).\n");
}

int Withdraw(void* socket, TVector vect, ull value) {
    TNode cur = FindNode(vect, mainID);
    if (cur) {
        if (cur->sum < value) {
            printf("Insufficient funds.\n");
            if (zmq_send(socket, "Insufficient funds", 18, 0) == -1)
                printf("Error! Can not send message (deposit).\n");
            return 0;
        } else {
            cur->sum -= value;
            char msg[100];
            sprintf(msg, "Withdraw: done. -%llu, id: %d", value, mainID);
            if (zmq_send(socket, msg, strlen(msg), 0) == -1) {
                printf("Error! Can not send message (deposit).\n");
                return 0;
            }
            printf("Withdraw: operation is done.\n");
            return 1;
        }
    } else {
        printf("Error! Can not find an account of the client (withdraw).\n");
        if (zmq_send(socket, "Can not find the client's account (withdraw)", 44, 0)== -1)
            printf("Error! Can not send message (withdraw).\n");
        return 0;
    }
}

void Transfer(void* socket1, void* socket2, TVector vect, int adress, ull value) {
    char adr[32];
    char msg[100];
    sprintf(adr, "tcp://localhost:%d", adress);
    sprintf(msg, "%d deposit %d %llu", mainID, mainID, value);
    if (Withdraw(socket1, vect, value)) {
        zmq_connect (socket2, adr);
        zmq_send(socket2, msg, strlen(msg), 0);
        zmq_disconnect (socket2, adr);
        sprintf(msg, "Transfer: done. -%llu, id: %d", value, mainID);
        printf("Transfer: operation is done.\n");
        if (zmq_send(socket1, msg, strlen(msg), 0) == -1)
            printf("Error! Can not send message (transfer).\n");
    } else {
        printf("Error! Not enough money to transfer.\n");
        if (zmq_send(socket1, "Not enough money to transfer", 28, 0) == -1)
            printf("Error! Can not send message (transfer).\n");
    }
}

void Balance(void* socket, TVector vect) {
    TNode cur = FindNode(vect, mainID);
```

```
    if (cur) {
        char msg[30];
        sprintf(msg, "Balance: %llu", cur->sum);
        printf("Balance: operation is done. ID: %d, balance: %llu\n", mainID, cur->sum);
        if (zmq_send(socket, msg, strlen(msg), 0) == -1)
            printf("Error! Can not send message (balance).\n");
    }
    else {
        printf("Error! Can not find an account of the client (balance).\n");
        if (zmq_send(socket, "Can not find the client's account (balance)", 43, 0) == -1)
            printf("Error! Can not send message (balance).\n");
    }
}

int main (int argc, char* argv[]) {
    if (argc != 2) {
        printf("Error! You should set the bank adress.\n");
        return -1;
    }
    int bankAdress = atoi(argv[1]);
    char* adr = (char*)malloc(32 * sizeof(char));
    strcpy(adr, "tcp://*:");
    strcat(adr, argv[1]);
    void* context = zmq_ctx_new();
    if (context == NULL) {
        printf("Can not create context.\n");
        return -1;
    }
    void* respond = zmq_socket(context, ZMQ_REP);
    if (respond == NULL) {
        printf("Can not create socket (responder).\n");
        return -1;
    }
    void* interbank = zmq_socket(context, ZMQ_REQ);
    if (interbank == NULL) {
        printf("Can not create socket (interbank).\n");
        return -1;
    }
    if (zmq_bind(respond, adr) == -1) {
        printf("Can not bind socket.\n");
        return -1;
    }
    TVector vect = CreateVector();
    char tmp[100];
    char cmd[100];
    int clientID;
    int adress;
    ull value;
    while (1) {
        strcpy(cmd, "");
        strcpy(tmp, "");
        if (zmq_recv(respond, tmp, 100, 0) == -1) {
            printf("Error! Can not receive message.\n");
            return -1;
        }
        sscanf(tmp, "%d %s", &mainID, cmd);
        if (!strcmp(cmd, "deposit")) {
            sscanf(tmp, "%*d %*s %d %llu", &clientID, &value);
            Deposit(respond, vect, clientID, value);
        } else if (!strcmp(cmd, "balance")) {
            Balance(respond, vect);
        } else if (!strcmp(cmd, "withdraw")) {
            sscanf(tmp, "%*d %*s %llu", &value);
            Withdraw(respond, vect, value);
        } else if (!strcmp(cmd, "transfer")) {
            sscanf(tmp, "%*d %*s %d %llu", &adress, &value);
            Transfer(respond, interbank, vect, adress, value);
        } else {
            printf("Unknown command!\n");
```

```
            if (zmq_send(respond, "Unknown command", 15, 0) == -1) {
                printf("Error! Can not send message (balance).\n");
            }
        }
        clientID = 0;
        value = 0;
        adress = 0;
        sleep(1);
    }
    DeleteVector(&vect);
    zmq_close(respond);
    zmq_close(interbank);
    zmq_ctx_destroy(context);
    return 0;
}
```

## Тесты:

```
OC/lab6-8$ ./client 13 4444
Available commands:
>deposit SID SUM >withdraw
SUM >transfer BANK_ADR SUM
>balance

Enter your command:
>withdraw 100
Can not find the client's
account (withdraw)

Enter your command:
>balance
Can not find the client's
account (balance)

Enter your command:
>transfer 2017 50
Can not find the client's
account (withdraw)

Enter your command:
>deposit 13 600
Deposit: done. +600, id: 13

Enter your command:
>withdraw 750
Insufficient funds

Enter your command:
>balance
Balance: 600
```

```
OC/lab6-8$ ./client 13 2017
Available commands:
>deposit SID SUM >withdraw
SUM >transfer BANK_ADR SUM
>balance

Enter your command:
>deposit 13 950
Deposit: done. +950, id: 13

Enter your command:
>transfer 4444 200
Withdraw: done. -200, id: 13

Enter your command:
>balance
Balance: 750
```

```
                                Enter your command:
                                >balance
                                Balance: 800

                                Enter your command:
                                >withdraw 750
                                Withdraw: done. -750, id:
                                13

                                Enter your command:
                                >balance
                                Balance: 50

                                Enter your command:
                                >deposit 12 900
                                Deposit: done. +900, id: 12
OC/lab6-8$ ./client 12
4444
Available commands:
>deposit SID SUM
>withdraw SUM >transfer
BANK_ADR SUM >balance

Enter your command:
>balance
Balance: 900
```

**Выводы:**

При работе с клиент-серверными системами для обмена запросами между ними удобно использовать серверы сообщений, например, ZeroMQ. Он позволяет отправлять и обрабатывать сообщения асинхронно, а также предоставляет удобную абстракцию над сокетами. При использовании ZeroMQ есть возможность отправлять запросы серверу, даже если он в этот момент не подключен. После подключения гарантируется, что все сообщения будут доставлены в правильном порядке и обработаны.