Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра: 806 «Вычислительная математика и программирование»

Факультет: «Прикладная математика и физика»

Дисциплина: «Операционные системы»

Лабораторная работа №6-8.

Тема:

«Сервера сообщений»

Группа: 8О-204Б

Студент: Утенкков

Вариант: №3-3-3

Оценка:

Подпись:

Москва

2016

Программа сервера

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <string.h>

#include <stdio.h>

#include "zmq.h"

#include <windows.h>

typedef struct node{

int choise;

double key;

double frid;

float balance;

float sum;

node\* left;

node\* right;

}node;

struct node \*root=0;

void insert(double key, struct node \*\*leaf)

{

if (\*leaf == 0)

{

\*leaf = (struct node\*) malloc(sizeof(struct node));

(\*leaf)->key = key;

(\*leaf)->balance = 0;

(\*leaf)->left = 0;

(\*leaf)->right = 0;

}

else if (key < (\*leaf)->key)

{

insert(key, &(\*leaf)->left);

}

else if (key >(\*leaf)->key)

{

insert(key, &(\*leaf)->right);

}

}

struct node \*search(double z, struct node \*leaf)

{

if (leaf != 0)

{

if (z == leaf->key)

{

return leaf;

}

else if (z<leaf->key)

{

return search(z, leaf->left);

}

else

{

return search(z, leaf->right);

}

}

else return 0;

}

void add\_money(float money, double key)

{

struct node \*leaf = search(key, root);

leaf->balance += money;

}

void min\_money(float money, double key)

{

struct node \*leaf = search(key, root);

if (leaf->balance < money)

{

printf("Are you broken?\n");

}

else{

leaf->balance -= money;}

}

void send\_client(double key)

{

int d;

node \*leaf = search(key, root);

node x;

x.key = leaf->key;

x.balance = leaf->balance;

x.left = 0;

x.right = 0;

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

void\* serSocket = zmq\_socket(contex, ZMQ\_REQ);

char string[30];

printf("enter last digit of ip address ");

scanf("%d", &d);

sprintf(string, "tcp://127.0.0.%d:4400\n",d);

zmq\_connect(serSocket, string); zmq\_msg\_t reply;

zmq\_msg\_init(&reply);

zmq\_msg\_recv(&reply, serSocket, 0);

size\_t repSize = zmq\_msg\_size(&reply);

//printf("Received: - %d %s\n", repSize, zmq\_msg\_data(&reply));

zmq\_msg\_close(&reply);

zmq\_msg\_t zmqMess;

zmq\_msg\_init\_size(&zmqMess, sizeof(node));

memcpy(zmq\_msg\_data(&zmqMess), &x, sizeof(node));

int send = zmq\_msg\_send(&zmqMess, serSocket, 0);

printf("mess data %f",x.key);

zmq\_msg\_close(&zmqMess);

Sleep(5000);

}

void send\_sum(float sum, double cur\_id, double friend\_id)

{

node \*leaf = search(cur\_id, root);

if (leaf->balance >= sum)

{

min\_money(sum, cur\_id);

add\_money(sum, friend\_id);

}

else

printf("Are you broken?\n");

}

void money(double key)

{

struct node \*leaf = search(key, root);

printf("balance = %f\n", leaf->balance);

}

int main(int argc, char const \*argv[])

{

int k=0;

float sum=0;

int d;

char string[30];

printf("enter last digit of ip address ");

scanf("%d", &d);

sprintf(string, "tcp://127.0.0.%d:4400\n", d);

printf("%s", string);

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

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

zmq\_bind(serverSocket, string);

printf("Starting...\n");

zmq\_msg\_t message;

zmq\_msg\_init(&message);

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

node \*m = (node \*)zmq\_msg\_data(&message);

printf("id: %f balance: %f \n", m->key, m->balance);

insert(m->key, &root);

zmq\_msg\_close(&message);

Sleep(1000);

zmq\_msg\_t reply;

zmq\_msg\_init\_size(&reply, strlen("ok") + 1);

memcpy(zmq\_msg\_data(&reply), "ok\0", 3);

zmq\_msg\_send(&reply, serverSocket, 0);

zmq\_msg\_close(&reply);

while (1)

{

zmq\_msg\_t mess;

zmq\_msg\_init(&mess);

zmq\_msg\_recv(&mess, serverSocket, 0);

node \*md = (node \*)zmq\_msg\_data(&mess);

if (!search(md->key,root))

{

insert(md->key, &root);

printf("new client added\n");

printf("%lf", md->key);

}

zmq\_msg\_close(&mess);

switch (md->choise)

{

case NULL:

{

break;

}

case 1:

{

printf("id %f\n", md->key);

add\_money(md->sum, md->key);

money(md->key);

printf("success\n");

break;

}

case 2:

{

printf("id %f\n", md->key);

min\_money(md->sum, md->key);

money(md->key);

printf("success\n");

break;

}

case 3:

{

printf("id %f\n", md->key);

send\_sum(md->sum, md->key, md->frid);

money(md->key);

printf("id %f\n", md->frid);

money(md->frid);

printf("success\n");

break;

}

case 4:

{

printf("%f", md->key);

send\_client(md->key);

break;

}

case -1:

{

printf(“Are you sure?”);

scanf(“%d”, &k);

if(k==1){

zmq\_close(serverSocket);

zmq\_ctx\_destroy(context);

return 0;}

else break;

}

}

zmq\_msg\_t reply;

zmq\_msg\_init\_size(&reply, strlen("ok") + 1);

memcpy(zmq\_msg\_data(&reply), "ok\0", 3);

zmq\_msg\_send(&reply, serverSocket, 0);

Sleep(1000);

}

}

Программа клиента

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <time.h>

#include <string.h>

#include <stdio.h>

#include "zmq.h"

#include <Windows.h>

typedef struct node{

int choise;

double key;

double frid;

float balance;

float sum;

node\* left;

node\* right;

};

int main(int argc, char const \*argv[])

{

srand(time(0));

double id = (double)rand();

int i;

int d;

char string[30];

printf("enter last digit of ip address ");

scanf("%d", &d);

sprintf(string, "tcp://127.0.0.%d:4400\n", d);

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

printf("Publisher %d Starting...\n", id);

float sum;

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

zmq\_connect(publishSocket, string);

int count = 0;

int choise;

node md;

md.key = id;

md.balance = 0;

md.left = 0;

md.right = 0;

zmq\_msg\_t zmqMessage;

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

memcpy(zmq\_msg\_data(&zmqMessage), &md, sizeof(node));

printf("Sending: - %d\n", count);

int send = zmq\_msg\_send(&zmqMessage, publishSocket, 0);

zmq\_msg\_close(&zmqMessage);

Sleep(1000);

zmq\_msg\_t reply;

zmq\_msg\_init(&reply);

zmq\_msg\_recv(&reply, publishSocket, 0);

size\_t repSize = zmq\_msg\_size(&reply);

printf("Received: - %d %s\n", repSize, zmq\_msg\_data(&reply));

zmq\_msg\_close(&reply);

while (1){

zmq\_msg\_t reply;

zmq\_msg\_init(&reply);

zmq\_msg\_recv(&reply, publishSocket, 0);

size\_t repSize = zmq\_msg\_size(&reply);

//printf("Received: - %d %s\n", repSize, zmq\_msg\_data(&reply));

zmq\_msg\_close(&reply);

printf("enter action\n");

scanf\_s("%d", &choise);

switch (choise)

{

case 1:

{

printf("enter sum ");

scanf\_s("%f", &sum);

md.choise = 1;

md.sum = sum;

break;

}

case 2:

{

printf("enter sum ");

scanf\_s("%f", &sum);

md.choise = 2;

md.sum = sum;

break;

}

case 3:

{

double k;

printf("enter friend ");

scanf\_s("%lf", &k);

printf("enter sum ");

scanf\_s("%f", &sum);

md.choise = 3;

md.frid = k;

md.sum = sum;

break;

}

case 4:

{

md.key = id;

md.choise = 4;

break;

}

case -1:

{

md.choise = -1;

zmq\_msg\_t zmqMess;

zmq\_msg\_init\_size(&zmqMess, sizeof(node));

memcpy(zmq\_msg\_data(&zmqMess), &md, sizeof(node));

int send = zmq\_msg\_send(&zmqMess, publishSocket, 0);

zmq\_msg\_close(&zmqMess);

zmq\_close(publishSocket);

zmq\_ctx\_destroy(context);

return 0;

}

}

zmq\_msg\_t zmqMess;

zmq\_msg\_init\_size(&zmqMess, sizeof(node));

memcpy(zmq\_msg\_data(&zmqMess), &md, sizeof(node));

int send = zmq\_msg\_send(&zmqMess, publishSocket, 0);

zmq\_msg\_close(&zmqMess);

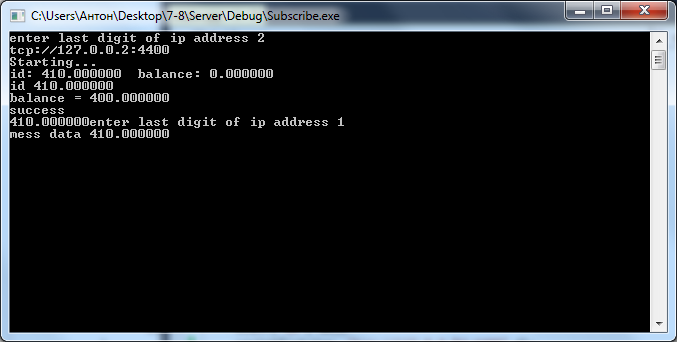
Sleep(100);

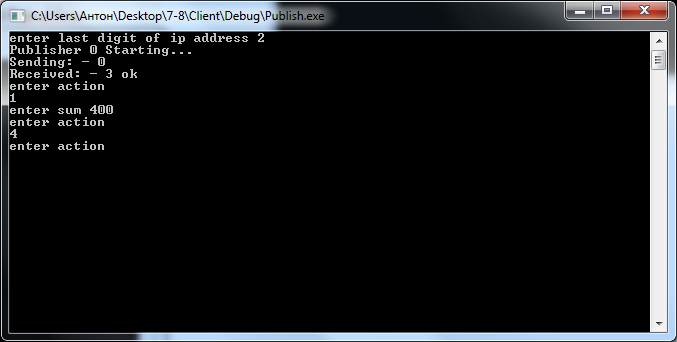
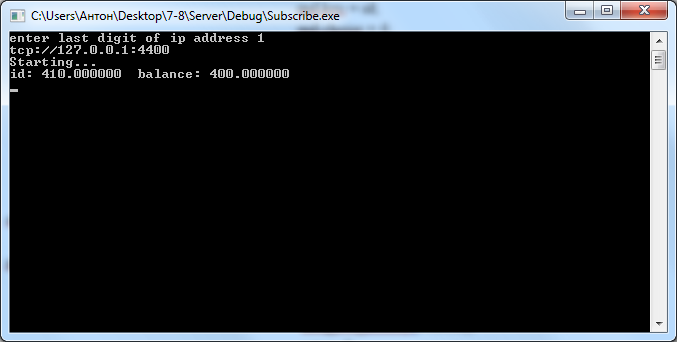
}

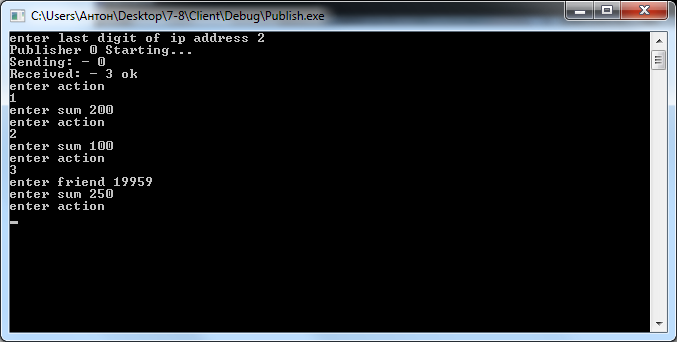
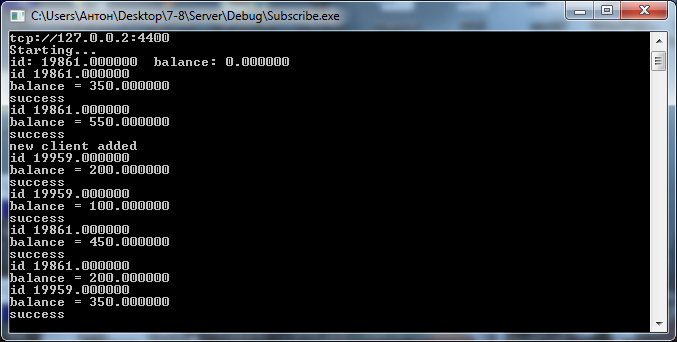
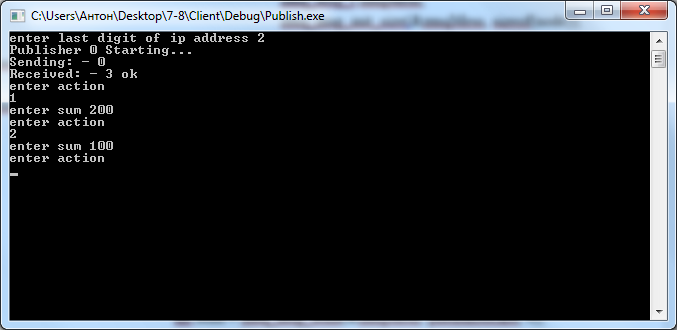
}

Пример работы

Передача данных клиента на другой сервер





Обмен счетами между клиентами  

Вывод

Для данной работы мы использовали минимальный функционал библиотек семейства MQ,

а именно создание/подключение к сокету и передачу/прием сообщений, причем в качестве «текста» сообщений мы можем использовать любую структуру, будь то пользовательскую (элемент дерева), либо системную (строка, число).

И по своей структуре программа в чем-то похожа на связку PHP+SQL, где мы так же подключаемся и принимаем сообщение после запроса.

Но основная проблема заключается в правильной установке библиотеки для VS, потому что требуется учитывать версию статической и динамической, является ли dll откомпилированной и находится ли она в папке с программой. И если не будет выполнено одно из требований установки – нас ждет «веселый» поиск причин ошибки линковщика, в то же время как для Unix компилятора C/C++ нам всего лишь требуется пару ключей для makefile.