Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Курсовой проект по курсу**

**«Операционные системы»**

**Клиент-серверная система для передачи мгновенных сообщений**

Студент: Никулин Кристиан Ильич

Группа: М8О-208Б-21

Вариант: 26

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

**Содержание**

1. Постановка задачи
2. Общие сведения о программе
3. Общий метод и алгоритм решения
4. Основные файлы программы
5. Демонстрация работы программы
6. Вывод

**Постановка задачи**

**Цель работы**

1. Приобретение практических навыков в использовании знаний, полученных в течении курса
2. Проведение исследования в выбранной предметной области

**Задание**

Необходимо спроектировать и реализовать программный прототип в соответствии с выбранным вариантом. Произвести анализ и сделать вывод на основании данных, полученных при работе программного прототипа.

Необходимо спроектировать и реализовать программный прототип в соответствии с выбранным вариантом. Произвести анализ и сделать вывод на основании данных, полученных при работе программного прототипа.

**Вариант 26**

Необходимо предусмотреть возможность хранения истории переписок (на сервере) и поиска по ним. Связь между сервером и клиентом должна быть реализована при помощи очередей сообщений (например, ZeroMQ).

Базовый функционал должен быть следующим:

• Клиент может присоединиться к серверу, введя логин

• Клиент может отправить сообщение другому клиенту по его логину

• Клиент в реальном времени принимает сообщения от других клиентов

**Общие сведения о программе**

Программа состоит из трёх файлов – server.cpp, client.cpp, functions.cpp, в которых расположены код сервера, код клиента и реализация вспомогательный функций и структур. Для удобства также был создан Makefile.

**Общий метод и алгоритм решения**

Общение между клиентом и сервером осуществляется по схеме:

- сервер принимает сообщения от клиентов с помощью ZMQ\_PULL сокета, а потом отправляет их с помощью ZMQ\_PUB

- клиент отсылает сообщение серверу через ZMQ\_PUSH сокет, а в отдельном потоке принимает сообщения от сервера с помощью ZMQ\_SUB

Для начала необходимо запустить сервер и зарегистрировать пользователей (ввести список допустимых логинов, которые сервер сохранит для дальнейшего использования в файл logins.txt).

После запускаются клиенты, которые могут отправлять сообщения друг другу через сервер.

Чтобы посмотреть свою историю сообщений, клиент вводит команду history <word>, где <word> - слово, которое мы ищем в сообщениях.

**Основные файлы программы**

**Makefile :**

all:

    g++ server.cpp -o server -lzmq

    g++ client.cpp -o client -lzmq -pthread

server:

    g++ server.cpp -o server -lzmq

client:

    g++ client.cpp -o client -lzmq -pthread

clean:

    rm -rf client server logins.txt

**server.cpp :**

#include "functions.cpp"

int main()

{

    cout << "Enter all user's logins. Insert 'end' to stop:\n";

    fflush(stdout);

    vector<vector<string>> history(HISTORY\_SIZE, vector<string>(3)); // 0 - пол, 1 - отпр, 2 - сообщение

    int index = 0;

    vector<string> logins;

    string login;

    ofstream out;

    out.open("logins.txt");

    while (login != "end")

    {

        cin >> login;

        if (in(logins, login) == -1)

        {

            if (login != "end")

            {

                logins.push\_back(login);

                out << login << "\n";

            }

        }

        else

            cout << "Already exists! Write new login!\n";

        fflush(stdout);

    }

    out.close();

    // ZEROMQ

    void \*context = zmq\_ctx\_new();

    void \*requester = zmq\_socket(context, ZMQ\_PULL); // для приема сообщений сервером

    char conn\_pull[] = "tcp://127.0.0.1:1234";

    int rc = zmq\_bind(requester, conn\_pull); // привязка сокета

    assert(rc == 0);

    void \*publisher = zmq\_socket(context, ZMQ\_PUB); // для отправления сообщений сервером

    char conn\_pub[] = "tcp://127.0.0.1:4321";

    rc = zmq\_bind(publisher, conn\_pub);

    assert(rc == 0);

    //

    MessageData buf; // сообщение (логин отправителя, логин получателя, сообщение)

    cout << "OK. Server is working!\n";

    while (1)

    {

        receive\_struct(requester, &buf);

        int pos = h.find(buf.recipient\_login);

        if (pos != -1) // history

        {

            pos = a.find(buf.msg\_txt);

            if (pos != -1)

            {

                cout << "\n"

                     << "Message history of all users:\n";

                for (int i = 0; i < index; ++i)

                {

                    cout << "Recipient: " << history[i][0] << " Sender: " << history[i][1];

                    cout << " Message: " << history[i][2] << "\n";

                    fflush(stdout);

                }

            }

            else

            {

                cout << "\n"

                     << buf.sender\_login << "'s message history with word '" << buf.msg\_txt << "' :\n";

                fflush(stdout);

                char send[SIZE];

                char recp[SIZE];

                int s, r, q = 0;

                char m[SIZE];

                strcpy(m, buf.msg\_txt);

                // цикл по кол-ву сообщений в истории

                for (int i = 0; i < index + 1; ++i)

                {

                    r = history[i][0].find(buf.sender\_login);

                    s = history[i][1].find(buf.sender\_login);

                    pos = history[i][2].find(m);

                    if ((r != -1 || s != -1) && pos != -1)

                    {

                        q++;

                        cout << "Recipient: " << history[i][0] << " Sender: " << history[i][1];

                        cout << " Message: " << history[i][2] << "\n";

                        fflush(stdout);

                    }

                }

                if (q == 0)

                {

                    cout << "No messages with word '" << buf.msg\_txt << "'\n";

                    fflush(stdout);

                }

            }

        }

        else

        {

            // добавляем в историю

            char buffer[SIZE];

            string q = "";

            strcpy(buffer, buf.recipient\_login);

            q = "";

            q += buf.recipient\_login;

            history[index][0] = q;

            strcpy(buffer, buf.sender\_login);

            q = "";

            q += buf.sender\_login;

            history[index][1] = q;

            strcpy(buffer, buf.msg\_txt);

            q = "";

            q += buf.msg\_txt;

            history[index][2] = q;

            index++;

            // отправляем сообщение

            send\_struct(publisher, buf);

        }

    }

    zmq\_close(requester);

    zmq\_term(context);

    remove("logins.txt");

    return 0;

}

**client.cpp :**

#include "functions.cpp"

typedef struct args\_thrd\_

{

    void \*socket;

    char login[SIZE];

} Args\_thrd\_t;

void \*accept\_message(Args\_thrd\_t \*args)

{

    while (1)

    {

        MessageData buf;

        receive\_struct(args->socket, &buf);

        if (!strcmp(buf.recipient\_login, args->login))

        {

            cout << "message from " << buf.sender\_login << ": " << buf.msg\_txt << "\n";

            fflush(stdout);

            cout << args->login << " > ";

            fflush(stdout);

        }

    }

}

int main()

{

    // ZEROMQ

    void \*context = zmq\_ctx\_new();

    void \*requester = zmq\_socket(context, ZMQ\_PUSH); // отсылает

    char conn\_push[] = "tcp://127.0.0.1:1234";

    int rc = zmq\_connect(requester, conn\_push);

    assert(rc == 0);

    void \*subscriber = zmq\_socket(context, ZMQ\_SUB); // принимает

    char conn\_sub[] = "tcp://127.0.0.1:4321";

    rc = zmq\_connect(subscriber, conn\_sub);

    assert(rc == 0);

    rc = zmq\_setsockopt(subscriber, ZMQ\_SUBSCRIBE, "", 0);

    assert(rc == 0);

    //

    ifstream in;

    in.open("logins.txt");

    string s;

    char c;

    while (!in.eof())

    {

        in.get(c);

        s.push\_back(c);

    }

    in.close();

    cout << "Insert your login:\n";

    char login[SIZE];

    while (1)

    {

        cin >> login;

        int pos = s.find(login);

        if (pos != -1)

        { // логин есть

            cout << "Welcome! You have signed as " << login;

            cout << ". Now you can send and recieve messages! ";

            cout << "\nTo search in message history insert: history <word>\n";

            fflush(stdout);

            break;

        }

        else

            cout << "Wrong login! Please, try again! Insert your login:\n";

        fflush(stdout);

    }

    MessageData buf;

    Args\_thrd\_t args;

    strcpy(args.login, login);

    args.socket = subscriber;

    thread thrd(accept\_message, &args);

    while (1)

    {

        char rec[SIZE] = {};

        char mes[SIZE] = {};

        int pos;

        cout << login << " > ";

        fflush(stdout);

        strcpy(buf.sender\_login, login);

        cin >> rec;

        strcpy(buf.recipient\_login, rec);

        string mm = "";

        getline(cin, mm);

        copy(mm.begin() + 1, mm.end(), mes);

        mes[mm.size()] = 0;

        strcpy(buf.msg\_txt, mes);

        pos = h.find(rec);

        if (pos != -1) // history

        {

            cout << "Check " << login << "'s history on server!\n";

            fflush(stdout);

            send\_struct(requester, buf);

        }

        else

        {

            pos = s.find(rec);

            if (pos != -1) // логин есть

            {

                send\_struct(requester, buf);

            }

            else

            {

                cout << "Invalid recipient user login!\n";

                fflush(stdout);

            }

        }

    }

    thrd.detach();

    zmq\_close(requester);

    zmq\_close(subscriber);

    zmq\_term(context);

    return 0;

}

**functions.cpp :**

#include <iostream>

#include <stdlib.h>

#include <unistd.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <errno.h>

#include <fcntl.h>

#include <map>

#include <vector>

#include <string>

#include <thread>

#include <fstream>

#include "zmq.hpp"

using namespace std;

#define SIZE 128

#define HISTORY\_SIZE 30

typedef struct message\_

{

    char sender\_login[SIZE];

    char recipient\_login[SIZE];

    char msg\_txt[SIZE];

} MessageData;

string h = "history-";

string a = "ALL-";

int in(vector<string> logins, string str)

{

    for (int i = 0; i < logins.size(); ++i)

    {

        if (logins[i] == str)

            return i;

    }

    return -1;

}

void receive\_struct(void \*socket, MessageData \*md)

{

    zmq\_recv(socket, md->sender\_login, sizeof(md->sender\_login), 0);

    zmq\_recv(socket, md->recipient\_login, sizeof(md->recipient\_login), 0);

    zmq\_recv(socket, md->msg\_txt, sizeof(md->msg\_txt), 0);

}

void send\_struct(void \*socket, MessageData md)

{

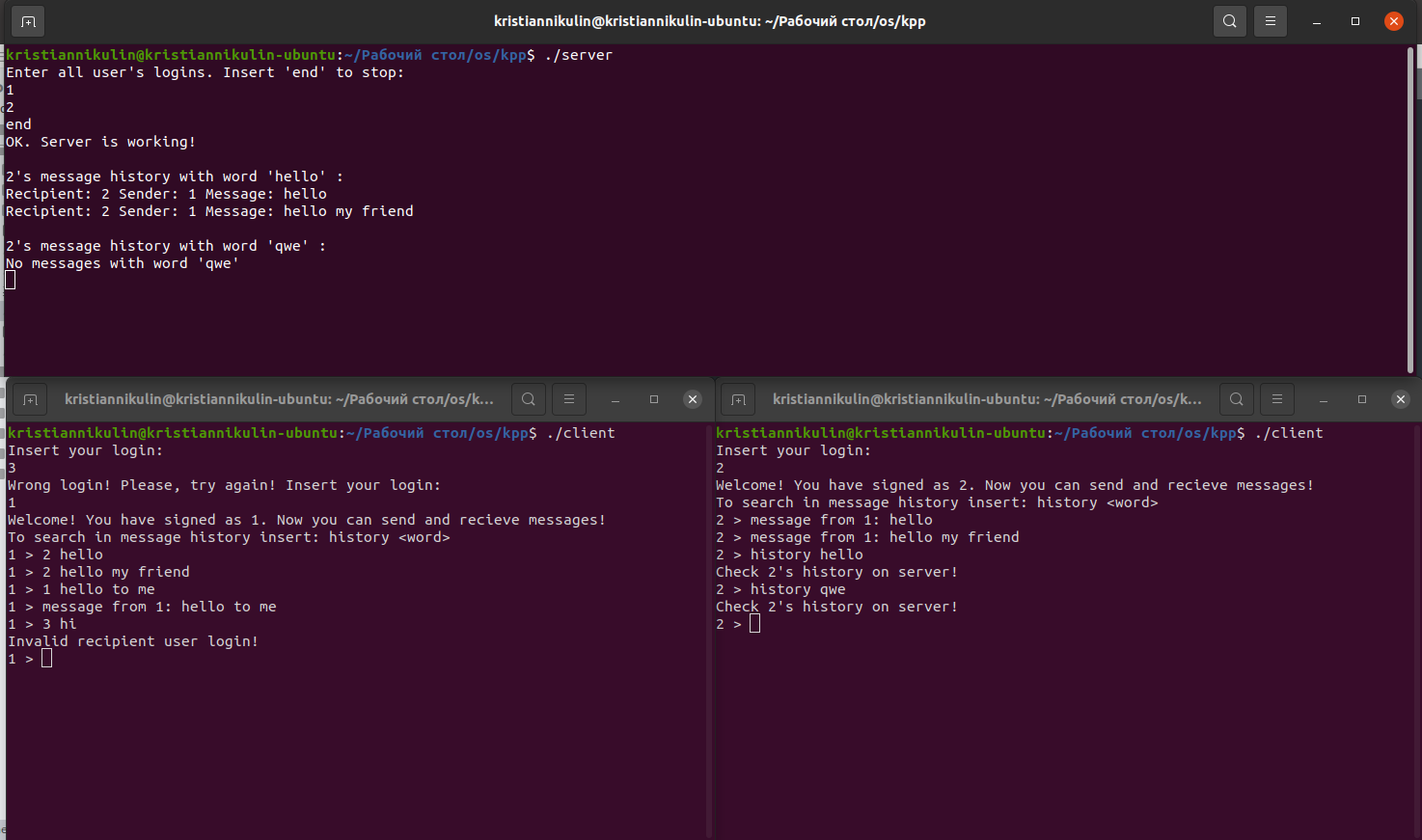
    zmq\_send(socket, md.sender\_login, sizeof(md.sender\_login), 0);

    zmq\_send(socket, md.recipient\_login, sizeof(md.recipient\_login), 0);

    zmq\_send(socket, md.msg\_txt, sizeof(md.msg\_txt), 0);

}

**Демонстрация работы программы**

****

**Вывод**

Сделав данный курсовой проект, я закрепил навыки в реализации консольного клиент-серверного взаимодействия с помощью очередей сообщений (zeromq), работы со строками в С++ и потоками.