

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: " << his-
tory[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;
    }
}

```

```

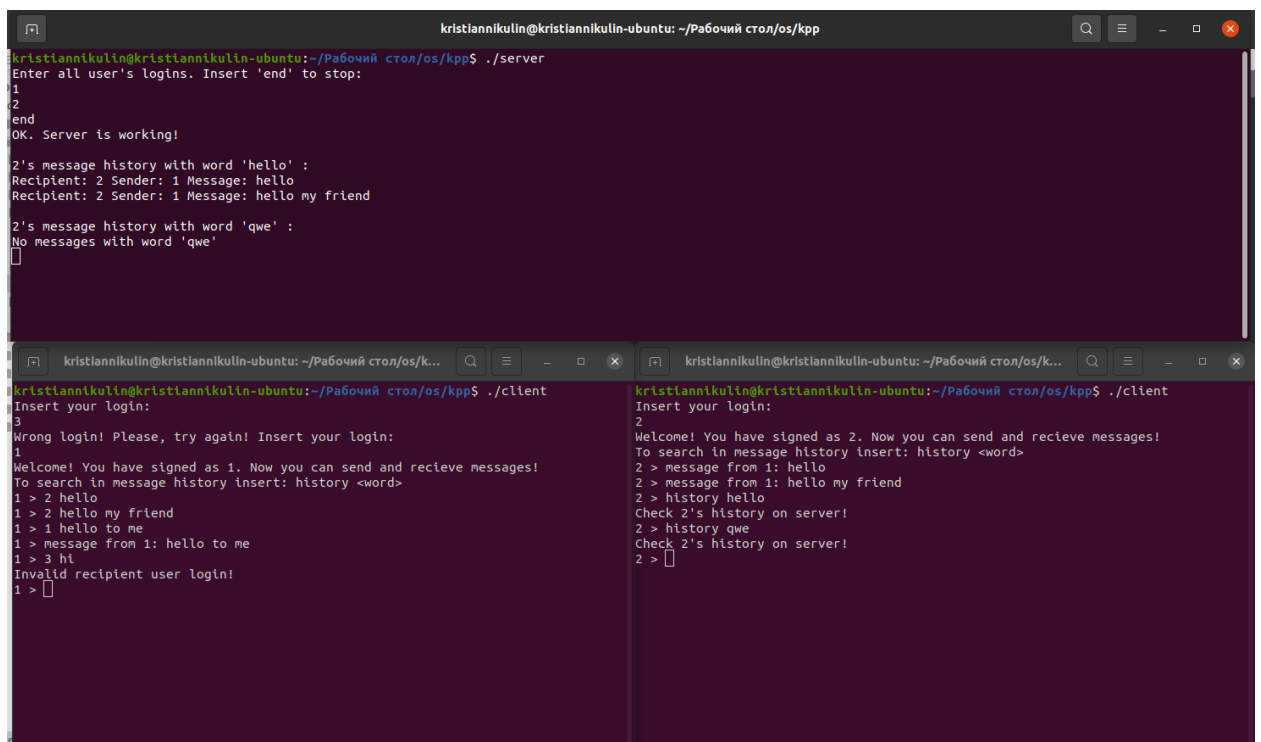
    }
    return -1;
}

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



```

kristiannikulin@kristiannikulin-ubuntu: ~/Рабочий стол/os/kpp
kristiannikulin@kristiannikulin-ubuntu:~/Рабочий стол/os/kpp$ ./server
Enter all user's logins. Insert 'end' to stop:
1
2
end
OK. Server is working!

2's message history with word 'hello' :
Recipient: 2 Sender: 1 Message: hello
Recipient: 2 Sender: 1 Message: hello my friend

2's message history with word 'qwe' :
No messages with word 'qwe'
[]

kristiannikulin@kristiannikulin-ubuntu:~/Рабочий стол/os/k...
kristiannikulin@kristiannikulin-ubuntu:~/Рабочий стол/os/kpp$ ./client
Insert your login:
3
Wrong login! Please, try again! Insert your login:
1
Welcome! You have signed as 1. Now you can send and recieve messages!
To search in message history insert: history <word>
1 > 2 hello
1 > 2 hello my friend
1 > 1 hello to me
1 > message from 1: hello to me
1 > 3 hi
Invalid recipient user login!
1 >

kristiannikulin@kristiannikulin-ubuntu:~/Рабочий стол/os/kpp$ ./client
Insert your login:
2
Welcome! You have signed as 2. Now you can send and recieve messages!
To search in message history insert: history <word>
2 > message from 1: hello
2 > message from 1: hello my friend
2 > history hello
Check 2's history on server!
2 > history qwe
Check 2's history on server!
2 >

```

Вывод

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