Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Курсовой проект по:

«Операционные системы»

Студент: Казанцев Данила Игоревич
Группа: М8О-207Б-21
Вариант: 26
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

https://github.com/thgdanilaya/mai_os_labs_cp

Постановка задачи

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

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 двух файлов – server.cpp, client.cpp, в которых расположены код сервера, код клиента соответственно.

Общий метод и алгоритм решения

Для передачи логинов и id процессов создается отдельная пара сокета типа Reply – Request на сервере и клиенте. При регистрации каждого клиента, создаются сокеты push и pull, с помощью которых сообщения от клиента отправляются на сервер, и клиент получает сообщения от сервера соответственно. Существует набор команд: send <login> <message> - отправляет сообщение другому клиенту, history – загружает историю сообщений клиента, exit – разлогинивает клиента с сервера. Так же в клиенте создается отдельный поток для получения сообщений с сервера, а на сервере потоки для обслуживания каждого клиента.

Исходный код

```
#include <iostream>
#include <map>
#include "zmq.hpp"
#include <vector>
#include <cstring>
#include <memory>
#include <thread>
```

```
using namespace std;


map<string, shared_ptr<zmq::socket_t>> ports;

map<string, bool> logged_in;

zmq::context_t context1(1);

void history_save(string login_sender, string login_accepter, string message,
map<string, map<string, string>> &history_of_messages)
{
    history_of_messages[login_sender][login_accepter] = histo-
ry_of_messages[login_sender][login_accepter].append("\n" + message);
}

void send_message(string message_string, zmq::socket_t &socket)
{
    zmq::message_t message_back(message_string.size());
    memcpy(message_back.data(), message_string.c_str(), mes-
sage_string.size());
    if (!socket.send(message_back))
    {
        cout << "Error" << endl;
    }
}

string receive_message(zmq::socket_t& socket) {
    zmq::message_t message_main;
    socket.recv(&message_main);
    string answer(static_cast<char*>(message_main.data()), mes-
sage_main.size());
    return answer;
}

void process_client(int id, map<string, map<string, string>>
&history_of_messages, string nickname)
{
    zmq::context_t context2(1);
    zmq::socket_t puller(context2, ZMQ_PULL);
    puller.bind("tcp://*:3" + to_string(id + 1));
    while (1)
    {
        string command = "";
        string client_mes = receive_message(puller);
        for (char i : client_mes) {
            if (i != ' ') {
                command += i;
            } else {
                break;
            }
        }
        int i;
        if (command == "send") {
            string recipient = "";
            for(i = 5; i < client_mes.size(); ++i){
                if(client_mes[i] != ' '){
                    recipient += client_mes[i];
                } else{
                    break;
                }
            }
            if(logged_in[recipient]) {
```

```
                string message;
                ++i;
                while(client_mes[i] != ' ') ++i;
                ++i;
                for(i; i < client_mes.size(); ++i){
                    message += client_mes[i];
                }
                send_message(client_mes, *ports[recipient]);
                history_save(nickname, recipient, message, histo-
ry_of_messages);
            } else {
                ++i;
                string sender = "";
                for(i; i < client_mes.size(); ++i){
                    if(client_mes[i] != ' '){
                        sender += client_mes[i];
                    } else {
                        break;
                    }
                }
                send_message("no client", *ports[sender]);
            }
        }else if (command == "history") {
            string history;
            for(auto sender : history_of_messages){
                if(sender.first == nickname){
                    for(auto accepter : sender.second){
                        history += accepter.first + " " + accepter.second +=
'\n';
                    }
                }
            }
            cout << history << endl;
            send_message("history " + history, *ports[nickname]);
        }else if (command == "exit") {
            string sender = "";
            for(i = 5; i < client_mes.size(); ++i){
                if(client_mes[i] != ' '){
                    sender += client_mes[i];
                } else{
                    break;
                }
            }
            send_message("exit", *ports[sender]);
            logged_in[sender] = false;
        }
    }

}

int main(){
    zmq::context_t context(1);
    zmq::socket_t socket_for_login(context, ZMQ_REP);

    socket_for_login.bind("tcp://*:4042");

    map<string, map<string, string>> history_of_messages;

    while (1) {
        string recieved_message = receive_message(socket_for_login);
        string id_s = "";
        int i;
        for(i = 0; i < recieved_message.size(); ++i){
```

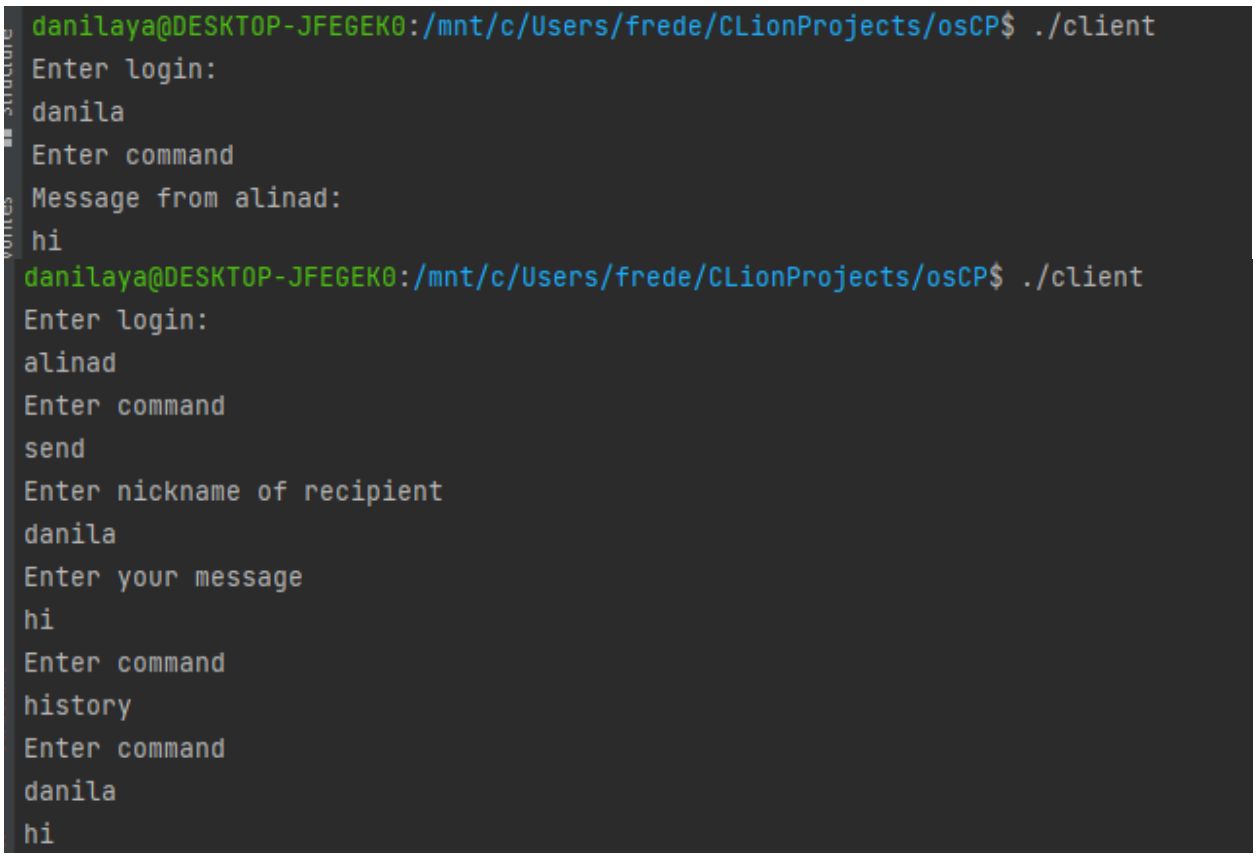
```
            if(recieved_message[i] != ' '){
                id_s += recieved_message[i];
            } else{
                break;
            }
        }
        int id = stoi(id_s);
        string nickname;
        ++i;
        for(i; i < recieved_message.size(); ++i){
            if(recieved_message[i] != ' '){
                nickname += recieved_message[i];
            } else{
                break;
            }
        }
        if(logged_in[nickname]) {
            cout << "This user already logged in..." << endl;
            send_message("0", socket_for_login);
        }
        else{
            logged_in[nickname] = true;
            cout << "User " << nickname << " logged in with id " << id <<
endl;
            send_message("1", socket_for_login);

            shared_ptr<zmq::socket_t> socket_client =
make_shared<zmq::socket_t>(context1, ZMQ_PUSH);
            socket_client->bind("tcp://*:3" + id_s);
            ports[nickname] = socket_client;
            thread worker = thread(ref(process_client), id,
ref(history_of_messages), nickname);
            worker.detach();
        }
    }
}
#include <iostream>
#include <cstring>
#include "zmq.hpp"
#include <string>
#include <thread>
#include <string>
#include <unistd.h>


using namespace std;

void send_message(string message_string, zmq::socket_t &socket) {
    zmq::message_t message_back(message_string.size());
    memcpy(message_back.data(), message_string.c_str(), mes-
sage_string.size());
    if (!socket.send(message_back)) {
        cout << "Error" << endl;
    }
}

string receive_message(zmq::socket_t &socket) {
    zmq::message_t message_main;
    socket.recv(&message_main);
    string answer(static_cast<char *>(message_main.data()), mes-
sage_main.size());
    return answer;
}
```

```
void process_terminal(zmq::socket_t &pusher, string login) {
    string command = "";
    cout << "Enter command" << endl;
    while (cin >> command) {
        if (command == "send") {
            cout << "Enter nickname of recipient" << endl;
            string recipient = "";
            cin >> recipient;
            cout << "Enter your message" << endl;
            string client_message = "";
            char a;
            cin >> a;
            getline(cin, client_message);
            string message_string = "send " + recipient + " " + login + " " +
a + client_message;
            send_message(message_string, pusher);
        }
        if (command == "history") {
            string message_string = "history";
            send_message(message_string, pusher);
        } else if (command == "exit") {
            send_message("exit " + login, pusher);
            break;
        }
        cout << "Enter command" << endl;
    }
}

void process_server(zmq::socket_t &puller) {
    while (1) {
        string command = "";
        string recieved_message = receive_message(puller);
        for (char i: recieved_message) {
            if (i != ' ') {
                command += i;
            } else {
                break;
            }
        }
        if (command == "send") {
            int i;
            string recipient = "", sender = "", mes_to_me = "";
            for (i = 5; i < recieved_message.size(); ++i) {
                if (recieved_message[i] != ' ') {
                    recipient += recieved_message[i];
                } else {
                    break;
                }
            }
            ++i;
            for (i; i < recieved_message.size(); ++i) {
                if (recieved_message[i] != ' ') {
                    sender += recieved_message[i];
                } else {
                    break;
                }
            }
            ++i;
            for (i; i < recieved_message.size(); ++i) {
                mes_to_me += recieved_message[i];
            }
            cout << "Message from " << sender << ":" << endl << mes_to_me <<
```

```
endl;
        } else if (command == "history") {
            string history;
            for (int i = 8; i < recieved_message.size(); ++i) {
                history += recieved_message[i];
            }
            cout << history << endl;
        } else if (command == "no") {
            cout << "We didn`t find this user" << endl;
        } else if (command == "exit") {
            break;
        }
    }
}

int main() {
    zmq::context_t context(1);
    zmq::socket_t socket_for_login(context, ZMQ_REQ);

    socket_for_login.connect("tcp://localhost:4042");
    cout << "Enter login: " << endl;
    string login = "";
    cin >> login;
    send_message(to_string(getpid()) + " " + login, socket_for_login);

    string recieved_message = receive_message(socket_for_login);
    if (recieved_message == "0") {
        cout << "login is already used" << endl;
        _exit(0);
    } else if (recieved_message == "1") {
        zmq::context_t context1(1);
        zmq::socket_t puller(context1, ZMQ_PULL);
        puller.connect("tcp://localhost:3" + to_string(getpid()));
        zmq::context_t context2(1);
        zmq::socket_t pusher(context2, ZMQ_PUSH);
        pusher.connect("tcp://localhost:3" + to_string(getpid() + 1));
        thread thr[1];
        thr[0] = thread(process_server, ref(puller));
        thr[0].detach();
        process_terminal(pusher, login);
        thr[0].join();
        context1.close();
        context2.close();
        puller.disconnect("tcp://localhost:3" + to_string(getpid()));
        pusher.disconnect("tcp://localhost:3" + to_string(getpid() + 1));
    }
    context.close();
    socket_for_login.disconnect("tcp://localhost:4042");
    return 0;
}
```

Демонстрация работы программы

```
danilaya@DESKTOP-JFEGEK0:/mnt/c/Users/frede/CLionProjects/osCP$ ./server
User danila logged in with id 1979
This user already logged in...
User alinad logged in with id 2010
```



Выводы

В ходе выполнения курсового проекта, я закрепил навыки работы с потоками и с очередями сообщений.