Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовой проект

по курсу

«Операционные системы»

Студентка: Варламова Анна Борисовна
Группа: М8О–207Б–20
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021

Цель курсового проекта

1. Приобретение практических навыков в использовании знаний, полученных в течение курса

2. Проведение исследования в выбранной предметной области

Постановка задачи

Задача: Создание клиента для передачи мгновенных личных сообщений.
Создать собственный клиент быстрых сообщений (возможно, и сервер – зависит от выбранной архитектуры), который бы работали в рамках сети.
Клиент-серверная система для передачи мгновенных сообщений. 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

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)

Общие сведения о программе

Для работы используются очереди сообщений, программа собирается при помощи Makefile. Есть 2 исполняемых файла – client, server. Сервер постоянно принимает сообщения от клиентов, которые подключаются к нему по ip. Каждый клиент имеет nickname. Клиент также может посмотреть историю сообщений при помощи команды dialog, найти сообщение по подстроке – find, отправить сообщение – send.

Устройство проекта

Запускается сервер, подключаются клиенты. Клиент вводит свой логин, после чего по общему для всех клиентов сокету передаётся на сервер его логин и id. Если в структуре map logged_in такой логин уже есть со значением true, это значит, что такой клиент уже подключился к серверу. По тому же сокету, который был типом REQ-REP передаётся 0 или 1 обратно клиенту. Если 0, то клиент выходит из программы, если 1, то клиент подключается к сокетам PULL PUSH, порт первго оканчивается на id клиента, второго на id+1. В то же время на сервере биндятся аналогичные сокеты, мапы logged_in и registered принимают значение 0, а в мап ports добавляется умный указатель на сокет PUSH со стороны сервера. В zeromq нет перегрузки оператора присваивания, поэтому просто так сокеты ни в одну структуру добавить нельзя. На сервере при аутентификации происходят ещё действия, связанные с получением сообщений, но о них позже.

У клиента создаётся дополнительный поток для приёма сообщений с сервера, он выполняет функцию process_server. После открепления потока запускается функция process_terminal, которая считывает данные с терминала, отправляет сообщения серверу.
На сервере при входе клиента запускается поток, который обрабатывает данного пользователя. В нём создаётся сокет PULL, принимающий сообщения от клиента.
Общий для всех команд механизм: в клиенте process_terminal считывает данные, узнаёт команду, отправляет сообщение серверу. Там сервер распознаёт команду, считывает необходимые данные, производит манипуляции согласно команде, по порту из ports передаёт сообщение нужному пользователю. В клиенте process_server распознаёт команду и выводит сообщение от сервера или исполяет другое действие согласно описанию команды.

Команды:

send

В клиенте process_terminal считывает получателя и сообщение, отправляет строчку серверу. Сервер считывает получателя, отправителя, письмо, сохраняет в вектор кортежей эти данные, отправляет полученную строчку получателю. Если получатель онлайн, в кортеж ставится занчение булевой переменной 1, если оффлайн, 0. Это нужно, чтобы при входе клиенту приходили пропущенные сообщения. process_server на стороне клиента распознаёт команду, считывает данные, выводит сообщение.

dialog

Сервер в цикле находит кортежи, в которых получатель или отправитель будет клиентом, который отправил запрос, каждое сообщение передаёт через сокет, клиент считывает сообщения и выводит.

find

Похожая на диалог команда, но она ищет в каждом сообщении вхождение подстроки в строку функцией string::find.

exit

Эта команда на стороне клиента отправляет сообщение серверу, завершает цикл while в process_terminal и, соответственно, саму эту функцию. Сервер отмечает данного клиента offline в map logged_in, отправляет сообщение этому же пользователю, чтобы выйти из цикла while в process_server. Так клиент завершит свою работу. Но здесь появлениется ошибка, проблема либо в закрытии потока, либо в сокетах.
На сервере тоже есть команда **exit,** для обработки терминала сервера был создан отдельный поток, она просто завершает программу, не закрывая никакие сокеты, что очень печально.

Основные файлы программы

Makefile:

```
all:
```

server
client

```
server: server.cpp
	g++ server.cpp -lzmq -pthread -o server -w
client: client.cpp
	g++ client.cpp -lzmq -pthread -o client -w
clean:
	rm server client
```

server.cpp:

```cpp
#include <iostream>
#include <map>
#include "zmq.hpp"
#include <vector>
#include <cstring>
#include <memory>
#include <thread>
#include <tuple>
#include <zconf.h>

// g++ server.cpp -lzmq -pthread -o server -w

//хранение логинов и сокетов PULL (client) - PUSH (server)
std::map<std::string, std::shared_ptr<zmq::socket_t>> ports;

//сейчас онлайн
std::map<std::string, bool> logged_in;

//когда-либо заходили на сервер
std::map<std::string, bool> registered;

//вектор кортежей из получателя, отправителя, сообщения
std::vector<std::tuple<std::string, std::string, std::string, bool>> tuples;

zmq::context_t context1(1);

void send_message(std::string message_string, zmq::socket_t &socket)

{
   zmq::message_t message_back(message_string.size());
   memcpy(message_back.data(), message_string.c_str(), message_string.size());
   if (!socket.send(message_back))
   {
      std::cout << "Error" << std::endl;
   }
}

void kill_me()
{
   std::string word;
   while (std::cin >> word)
   {
      if (word == "exit") {
         _exit(0);
      } else {
         std::cout << "Enter exit to stop server" << std::endl;
      }
```

```
    }
}

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

void process_client(int id)
{
    zmq::context_t context2(1);
    zmq::socket_t puller(context2, ZMQ_PULL);
    puller.bind("tcp://*:3" + std::to_string(id + 1));
    bool alive = true;
    while (alive)
    {
        std::string command = "";
        std::string client_mes = receive_message(puller);
        for (char i : client_mes) {
            if (i != ' ') {
                command += i;
            } else {
                break;
            }
        }
        int i;
        if (command == "send") {
            std::string recipient = "";
            for(i = 5; i < client_mes.size(); ++i){
                if(client_mes[i] != ' '){
                    recipient += client_mes[i];
                } else{
                    break;
                }
            }
            ++i;
            std::string sender = "";
            for(i; i < client_mes.size(); ++i){
                if(client_mes[i] != ' '){
                    sender += client_mes[i];
                } else {
                    break;
                }
            }
            ++i;
            std::string user_mes;
            for(i; i < client_mes.size(); ++i){
                user_mes += client_mes[i];
            }
            if(logged_in[recipient]) {
                tuples.push_back(std::make_tuple(recipient, sender, user_mes, 1));
                send_message(client_mes, *ports[recipient]);
                std::cout << sender << " sent message to " << recipient << std::endl;
            } else if (registered[recipient]) {
                tuples.push_back(std::make_tuple(recipient, sender, user_mes, 0));
```

```
            send_message("out of server", *ports[sender]);
            std::cout << sender << " sent message to offline " << recipient << std::endl;
        } else {
            send_message("no client", *ports[sender]);
            std::cout << sender << "tried to sent message to unknown " << recipient << std::endl;
        }
    } else if (command == "exit") {
        std::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;
        alive = false;
    } else if (command == "dialog") {
        std::string us1 = "", us2 = "";
        int i = 7, count_mess = 0;
        for(i; i < client_mes.size(); ++i){
            if(client_mes[i] != ' '){
                us1 += client_mes[i];
            } else {
                break;
            }
        }
        ++i;
        for(i; i < client_mes.size(); ++i){
            if(client_mes[i] != ' '){
                us2 += client_mes[i];
            } else {
                break;
            }
        }
        if (registered[us2]) {
            std::cout << us1 << " asked history with " << us2 << std::endl;
            for (auto&& tuple: tuples) {
                std::string rec, sen, mes;
                bool read;
                std::tie(rec, sen, mes, read) = tuple;
                if (((rec == us1) && (sen == us2)) || ((rec == us2) && (sen == us1))) {
                    send_message("dialog " + rec + " " + sen + " " + mes, *ports[us1]);
                    ++count_mess;
                }
            }
            if (count_mess == 0)
                send_message("zero", *ports[us1]);
        } else {
            std::cout << us1 << " asked history with unknown " << us2 << std::endl;
            send_message("no client", *ports[us1]);
        }
    } else if (command == "find") {
        std::string sender = "", str = "";
        int i = 5, count_str = 0;
        for(i; i < client_mes.size(); ++i){
```

```
            if(client_mes[i] != ' '){
                sender += client_mes[i];
            } else {
                break;
            }
        }
        ++i;
        for(i; i < client_mes.size(); ++i){
            str += client_mes[i];
        }
        std::cout << sender << " finds " << str << std::endl;
        for (auto&& tuple: tuples) {
            std::string rec, sen, mes;
            bool read;
            std::tie(rec, sen, mes, read) = tuple;
            std::size_t found = mes.find(str);
            if (((rec == sender) || (sen == sender)) && (found!=std::string::npos)) {
                send_message("find " + rec + " " + sen + " " + mes, *ports[sender]);
                ++count_str;
            }
        }
        if (count_str == 0)
            send_message("empt y", *ports[sender]);
    }
  }
}

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

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

  int uzers_number = 0;

  std::thread to_die = std::thread(kill_me);
  to_die.detach();

  while (1) {
    std::string recieved_message = receive_message(socket_for_login);
    std::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 = std::stoi(id_s);
    std::string nickname;
    ++i;
    for(i; i < recieved_message.size(); ++i){
        if(recieved_message[i] != ' '){
            nickname += recieved_message[i];
        } else{
            break;
```

```
            }
        }
        if(logged_in[nickname] == true) {
            std::cout << "This user already logged in..." << std::endl;
            send_message("0", socket_for_login);
        }
        else{
            logged_in[nickname] = true;
            std::cout << "User " << nickname << " logged in with id " << id << std::endl;
            send_message("1", socket_for_login);

            std::shared_ptr<zmq::socket_t> socket_client = std::make_shared<zmq::socket_t>(context1,
ZMQ_PUSH);
            socket_client->bind("tcp://*:3" + id_s);
            ports[nickname] = socket_client;
            //
            if (registered[nickname]) {
                for (auto&& tuple: tuples) {
                    std::string rec, sen, mes;
                    bool read;
                    std::tie(rec, sen, mes, read) = tuple;
                    if ((rec == nickname) && (read == false)) {
                        send_message("dialog " + rec + " " + sen + " " + mes, *socket_client);
                        read = true;
                    }
                }
            }
            //
            std::thread worker = std::thread(std::ref(process_client), id);
            worker.detach();
            if(!registered[nickname]) {
                registered[nickname] = true;
            }
        }
    }
}
```

client.cpp:

```
#include <iostream>
#include <cstring>
#include "zmq.hpp"
#include <string>
#include <zconf.h>
#include <thread>
#include <string>

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

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

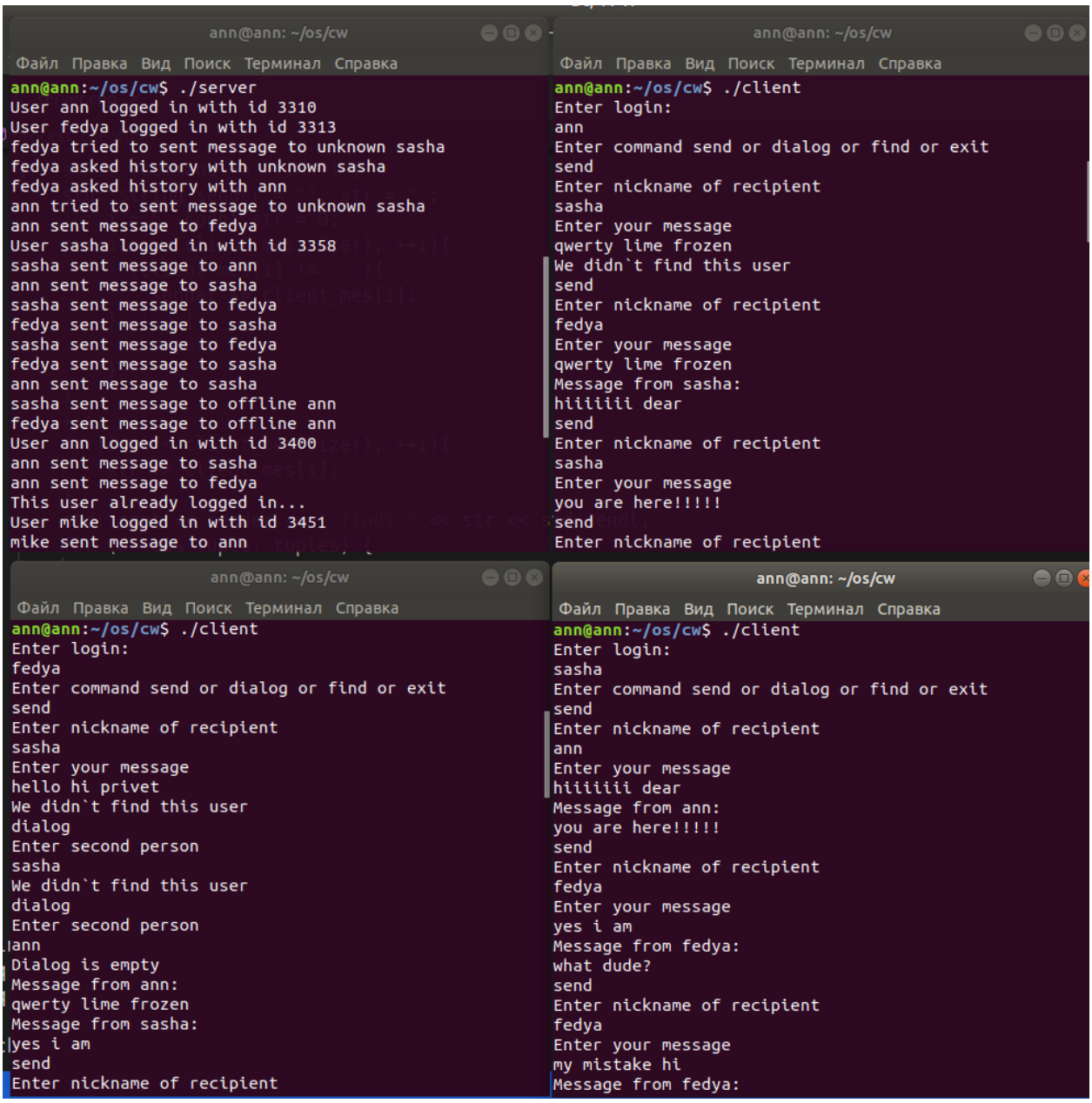
```
std::string receive_message(zmq::socket_t& socket){
    zmq::message_t message_main;
    socket.recv(&message_main);
    std::string answer(static_cast<char*>(message_main.data()), message_main.size());
    return answer;
}

void process_terminal(zmq::socket_t &pusher, std::string login)
{
    std::string command = "";
    std::cout << "Enter command send or dialog or find or exit" << std::endl;
    while (std::cin >> command)
    {
        if (command == "send") {
            std::cout << "Enter nickname of recipient" << std::endl;
            std::string recipient = "";
            std::cin >> recipient;
            std::cout << "Enter your message" << std::endl;
            std::string client_message = "";
            char a; std::cin >> a;
            std::getline (std::cin, client_message);
            std::string message_string = "send " + recipient + " " + login + " " + a + client_message;
            send_message(message_string, pusher);
        }
        else if (command == "exit") {
            send_message("exit " + login, pusher);
            break;
        } else if (command == "dialog") {
            std::cout << "Enter second person" << std::endl;
            std::string man = "";
            std::cin >> man;
            send_message("dialog " + login + " " + man, pusher);
        } else if (command == "find") {
            std::cout << "Enter finding string" << std::endl;
            std::string str = "";
            char a; std::cin >> a;
            std::getline (std::cin, str);
            send_message("find " + login + " " + a + str, pusher);
        } else {
            std::cout << "Enter command send or dialog or find or exit" << std::endl;
        }
    }
}

void process_server(zmq::socket_t &puller, std::string login)
{
    bool a = true;
    while (a)
    {
        std::string command = "";
        std::string recieved_message = receive_message(puller);
        for (char i : recieved_message) {
            if (i != ' ') {
                command += i;
            } else {
                break;
```

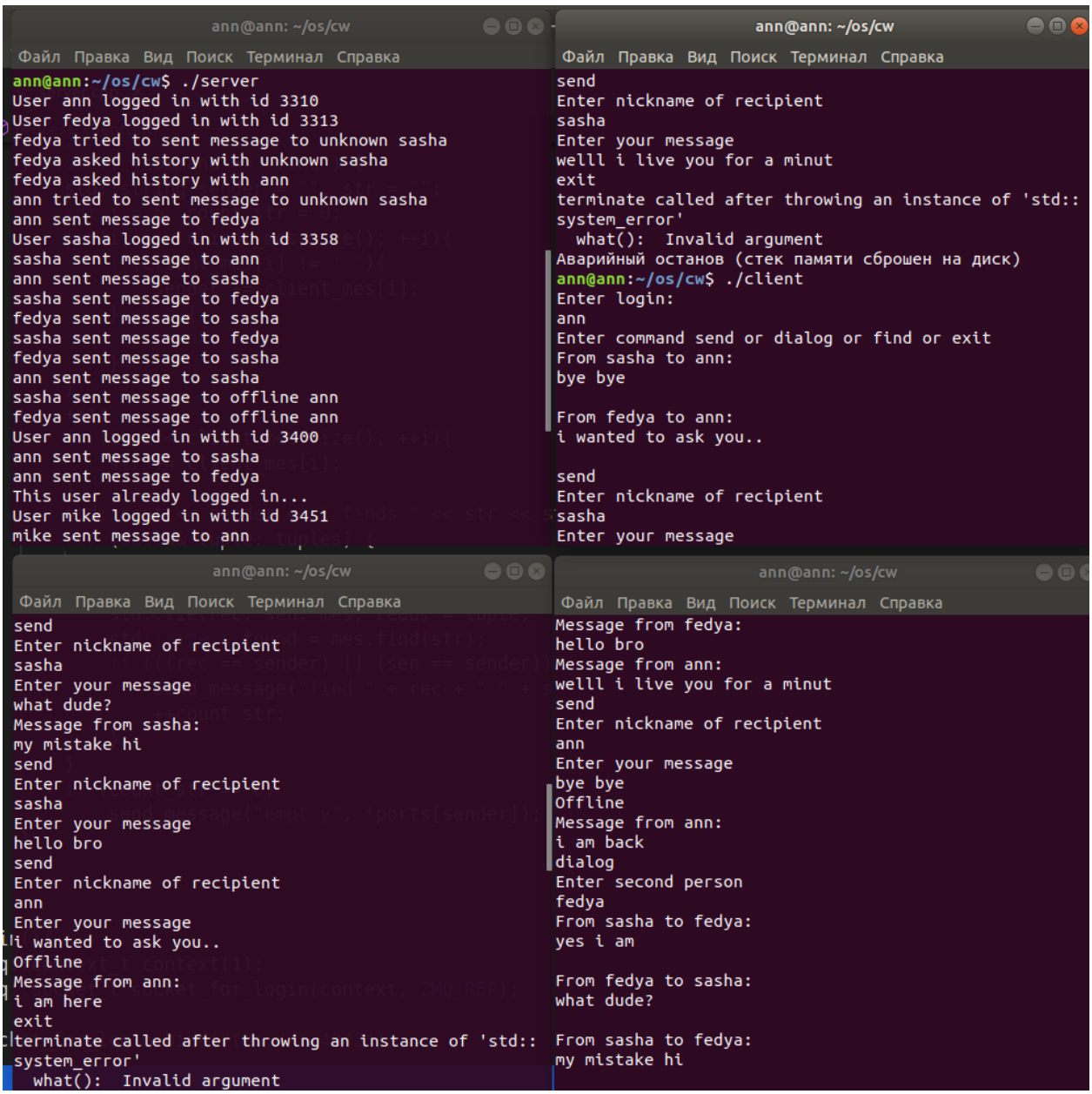
```
        }
    }
    if (command == "send") {
        int i;
        std::string recipient = "", sender = "", mes_to_me = "";
        for(i = 5; i < recieved_message.size(); ++i){
            if(recieved_message[i] != ' '){
                recipient += recieved_message[i];
            } else{
                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];
        }
        std::cout << "Message from " << sender << ":" << std::endl << mes_to_me << std::endl;
    } else if (command == "no") {
        std::cout << "We didn`t find this user" << std::endl;
    } else if (command == "exit") {
        a = false;
    } else if (command == "out") {
        std::cout << "Offline" << std::endl;
    } else if (command == "empt") {
        std::cout << "We didn`t find messages with this string" << std::endl;
    } else if (command == "zero") {
        std::cout << "Dialog is empty" << std::endl;
    } else if (command == "dialog") {
        int i = 7;
        std::string getter = "", setter = "", mes = "";
        for(i; i < recieved_message.size(); ++i){
            if (recieved_message[i] != ' ') {
                getter += recieved_message[i];
            } else{
                break;
            }
        }
        ++i;
        for(i; i < recieved_message.size(); ++i){
            if(recieved_message[i] != ' '){
                setter += recieved_message[i];
            } else{
                break;
            }
        }
        ++i;
        for(i; i < recieved_message.size(); ++i){
            mes += recieved_message[i];
        }
```

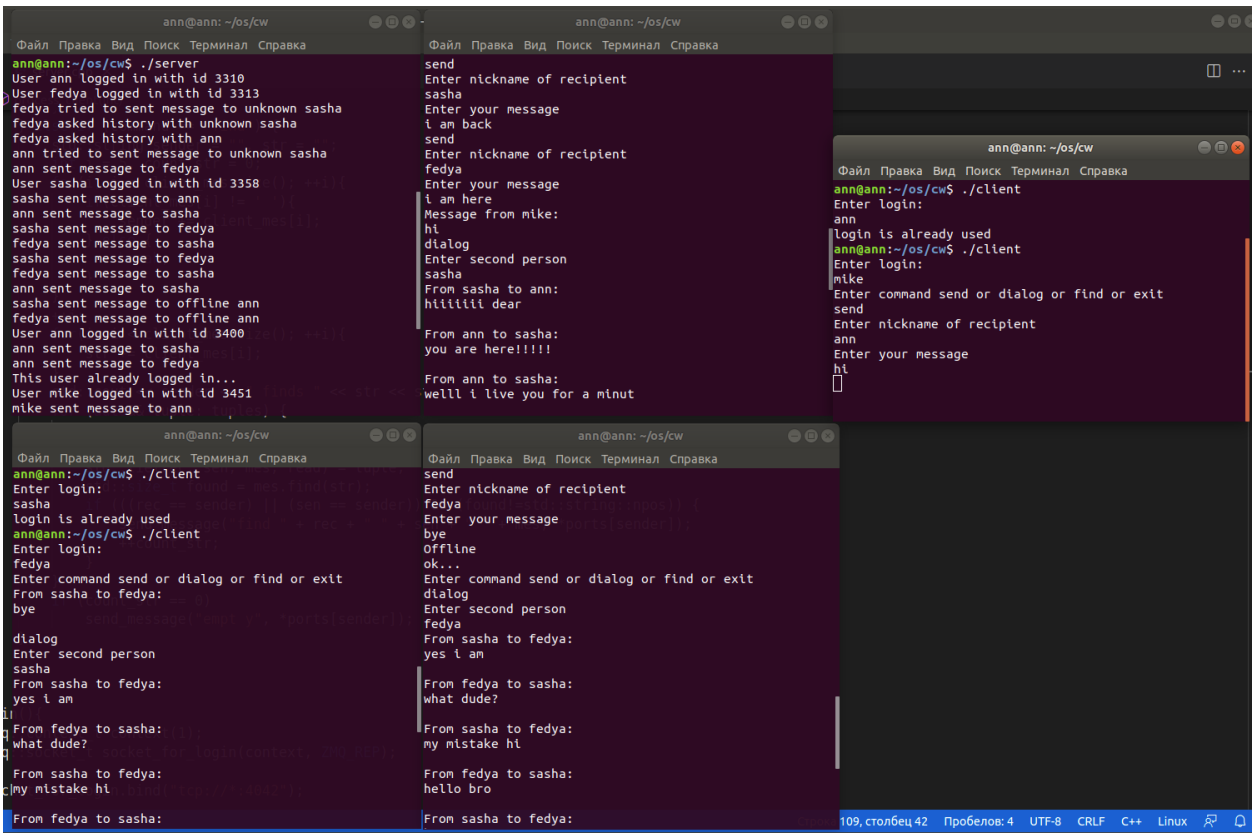
```
            std::cout << "From " << setter << " to " << getter << ":" << std::endl;
            std::cout << mes << std::endl << std::endl;
        } else if (command == "find") {
            int i = 5;
            std::string getter = "", setter = "", mes = "";
            for(i; i < recieved_message.size(); ++i){
                if (recieved_message[i] != ' ') {
                    getter += recieved_message[i];
                } else{
                    break;
                }
            }
            ++i;
            for(i; i < recieved_message.size(); ++i){
                if(recieved_message[i] != ' '){
                    setter += recieved_message[i];
                } else{
                    break;
                }
            }
            ++i;
            for(i; i < recieved_message.size(); ++i){
                mes += recieved_message[i];
            }
            std::cout << "From " << setter << " to " << getter << ":" << std::endl;
            std::cout << mes << std::endl << std::endl;
        }
    }
}

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

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

    std::string recieved_message = receive_message(socket_for_login);
    if (recieved_message == "0") {
        std::cout << "login is already used" << std::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" + std::to_string(getpid()));
        zmq::context_t context2(1);
        zmq::socket_t pusher(context2, ZMQ_PUSH);
        pusher.connect("tcp://localhost:3" + std::to_string(getpid() + 1));
        std::thread to_obtain = std::thread(process_server, std::ref(puller), login);
        to_obtain.detach();
        process_terminal(pusher, login);
        to_obtain.join();
        context1.close();
        context2.close();
```

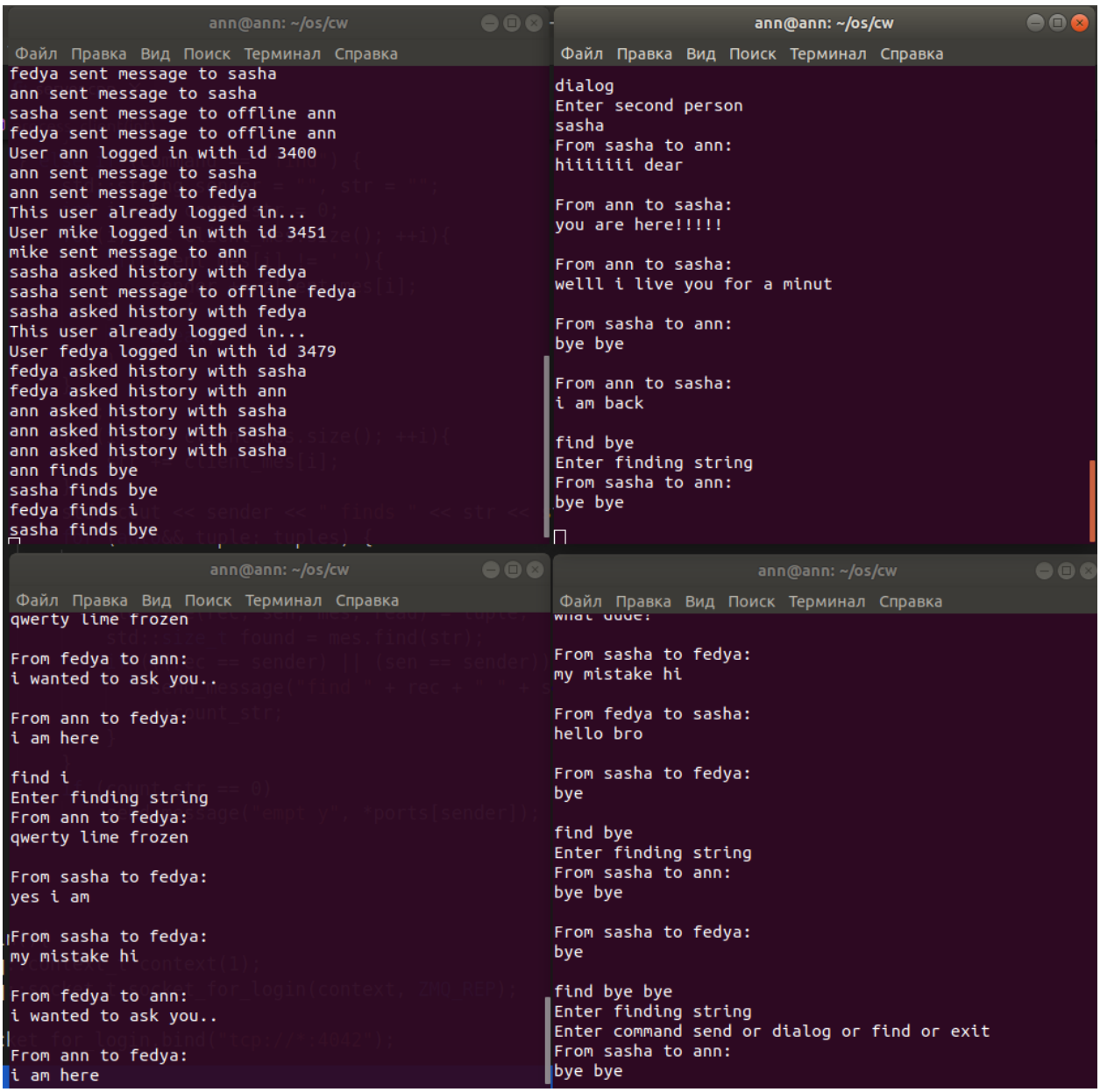
```
        puller.disconnect("tcp://localhost:3" + std::to_string(getpid()));
        pusher.disconnect("tcp://localhost:3" + std::to_string(getpid() + 1));
    }
    context.close();
    socket_for_login.disconnect("tcp://localhost:4042");
    return 0;
}
```

Пример работы


ann@ann: ~/os/cw
Файл Правка Вид Поиск Терминал Справка
ann@ann:~/os/cw$ ./server
User ann logged in with id 3310
User fedya logged in with id 3313
fedya tried to sent message to unknown sasha
fedya asked history with unknown sasha
fedya asked history with ann
ann tried to sent message to unknown sasha
ann sent message to fedya
User sasha logged in with id 3358
sasha sent message to ann
ann sent message to sasha
sasha sent message to fedya
fedya sent message to sasha
sasha sent message to fedya
fedya sent message to sasha
ann sent message to sasha
sasha sent message to offline ann
fedya sent message to offline ann
User ann logged in with id 3400
ann sent message to sasha
ann sent message to fedya
This user already logged in...
User mike logged in with id 3451
mike sent message to ann
ann@ann: ~/os/cw
Файл Правка Вид Поиск Терминал Справка
ann@ann:~/os/cw$ ./client
Enter login:
ann
Enter command send or dialog or find or exit
send
Enter nickname of recipient
sasha
Enter your message
qwerty lime frozen
We didn`t find this user
send
Enter nickname of recipient
fedya
Enter your message
qwerty lime frozen
Message from sasha:
hiiiiiii dear
send
Enter nickname of recipient
sasha
Enter your message
you are here!!!!!
send
Enter nickname of recipient
ann@ann: ~/os/cw
Файл Правка Вид Поиск Терминал Справка
ann@ann:~/os/cw$ ./client
Enter login:
fedya
Enter command send or dialog or find or exit
send
Enter nickname of recipient
sasha
Enter your message
hello hi privet
We didn`t find this user
dialog
Enter second person
sasha
We didn`t find this user
dialog
Enter second person
ann
Dialog is empty
Message from ann:
qwerty lime frozen
Message from sasha:
yes i am
send
Enter nickname of recipient
ann@ann: ~/os/cw
Файл Правка Вид Поиск Терминал Справка
ann@ann:~/os/cw$ ./client
Enter login:
sasha
Enter command send or dialog or find or exit
send
Enter nickname of recipient
ann
Enter your message
hiiiiiii dear
Message from ann:
you are here!!!!!
send
Enter nickname of recipient
fedya
Enter your message
yes i am
Message from fedya:
what dude?
send
Enter nickname of recipient
fedya
Enter your message
my mistake hi
Message from fedya:


ann@ann: ~/os/cw
Файл Правка Вид Поиск Терминал Справка
ann@ann:~/os/cw$ ./server
User ann logged in with id 3310
User fedya logged in with id 3313
fedya tried to sent message to unknown sasha
fedya asked history with unknown sasha
fedya asked history with ann
ann tried to sent message to unknown sasha
ann sent message to fedya
User sasha logged in with id 3358
sasha sent message to ann
ann sent message to sasha
sasha sent message to fedya
fedya sent message to sasha
sasha sent message to fedya
fedya sent message to sasha
ann sent message to sasha
sasha sent message to offline ann
fedya sent message to offline ann
User ann logged in with id 3400
ann sent message to sasha
ann sent message to fedya
This user already logged in...
User mike logged in with id 3451
mike sent message to ann
ann@ann: ~/os/cw
Файл Правка Вид Поиск Терминал Справка
send
Enter nickname of recipient
sasha
Enter your message
welll i live you for a minut
exit
terminate called after throwing an instance of 'std::
system_error'
what(): Invalid argument
Аварийный останов (стек памяти сброшен на диск)
ann@ann:~/os/cw$ ./client
Enter login:
ann
Enter command send or dialog or find or exit
From sasha to ann:
bye bye
From fedya to ann:
i wanted to ask you..
send
Enter nickname of recipient
sasha
Enter your message
ann@ann: ~/os/cw
Файл Правка Вид Поиск Терминал Справка
send
Enter nickname of recipient
sasha
Enter your message
what dude?
Message from sasha:
my mistake hi
send
Enter nickname of recipient
sasha
Enter your message
hello bro
send
Enter nickname of recipient
ann
Enter your message
i wanted to ask you..
Offline
Message from ann:
i am here
exit
terminate called after throwing an instance of 'std::
system_error'
what(): Invalid argument
ann@ann: ~/os/cw
Файл Правка Вид Поиск Терминал Справка
Message from fedya:
hello bro
Message from ann:
welll i live you for a minut
send
Enter nickname of recipient
ann
Enter your message
bye bye
Offline
Message from ann:
i am back
dialog
Enter second person
fedya
From sasha to fedya:
yes i am
From fedya to sasha:
what dude?
From sasha to fedya:
my mistake hi


ann@ann: ~/os/cw
Файл Правка Вид Поиск Терминал Справка
ann@ann:~/os/cw$ ./server
User ann logged in with id 3310
User fedya logged in with id 3313
fedya tried to sent message to unknown sasha
fedya asked history with unknown sasha
fedya asked history with ann
ann tried to sent message to unknown sasha
ann sent message to fedya
User sasha logged in with id 3358
sasha sent message to ann
ann sent message to sasha
sasha sent message to fedya
fedya sent message to sasha
sasha sent message to fedya
fedya sent message to sasha
ann sent message to sasha
sasha sent message to offline ann
fedya sent message to offline ann
User ann logged in with id 3400
ann sent message to sasha
ann sent message to fedya
This user already logged in...
User mike logged in with id 3451
mike sent message to ann
send
Enter nickname of recipient
sasha
Enter your message
i am back
send
Enter nickname of recipient
fedya
Enter your message
i am here
Message from mike:
hi
dialog
Enter second person
sasha
From sasha to ann:
hiiiiiii dear
From ann to sasha:
you are here!!!!!
From ann to sasha:
welll i live you for a minut
ann@ann: ~/os/cw
Файл Правка Вид Поиск Терминал Справка
ann@ann:~/os/cw$ ./client
Enter login:
ann
login is already used
ann@ann:~/os/cw$ ./client
Enter login:
mike
Enter command send or dialog or find or exit
send
Enter nickname of recipient
ann
Enter your message
hi
ann@ann: ~/os/cw
Файл Правка Вид Поиск Терминал Справка
ann@ann:~/os/cw$ ./client
Enter login:
sasha
login is already used
ann@ann:~/os/cw$ ./client
Enter login:
fedya
Enter command send or dialog or find or exit
From sasha to fedya:
bye
dialog
Enter second person
sasha
From sasha to fedya:
yes i am
From fedya to sasha:
what dude?
From sasha to fedya:
my mistake hi
From fedya to sasha:
ann@ann: ~/os/cw
Файл Правка Вид Поиск Терминал Справка
send
Enter nickname of recipient
fedya
Enter your message
bye
Offline
ok...
Enter command send or dialog or find or exit
dialog
Enter second person
fedya
From sasha to fedya:
yes i am
From fedya to sasha:
what dude?
From sasha to fedya:
my mistake hi
From fedya to sasha:
hello bro
From sasha to fedya:
109, столбец 42
Пробелов: 4
UTF-8
CRLF
C++
Linux



Вывод

В ходе выполнения курсового проекта я закрепила навык работы с очередью сообщений ZeroMQ, у неё есть свои плюсы и минусы. Из плюсов - высокий уровень абстракции, передача сообщений может проходить по разным уровням, что не заметно в использовании, несколько типов сокетов, которые позволяют реализовать свои задачи. Но этого всё равно не достаточно, мне пришлось использовать потоки, так как иначе нельзя добиться параллельных приёма и отправки сообщений. Также нет оператора присваивания сокетов, что добавляет хлопот при хранении, я для этого использовала умные указатели. Из проблем, которые решить не удалось: ошибка при завершении

клиента, ещё не получилось при завершении сервера убить всех клиентов, с использованием итераторов в map ports сообщения не передавались по сокету. Но это всё не мешает корректной работе программы, она выполняет всё так, как и задумывалось. Ещё при повторном входе в программу клиент получает сообщения, которые ему отправлялись, когда он выходил. Итак, это был полезный опыт работы с очередью сообщений, больше полезный тем, что я сделала свой сервер и применила потоки, нежели работала с очередью сообщений.