Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Курсовая работа по предмету**

**«Операционные системы»**

**по теме**

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

Студент: Кайдалова Александра Андреевна

Группа: М8О–212Б–22

Вариант: 26

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2023.

**Постановка задачи**

Клиент-серверная система для передачи мгновенных сообщений. Базовый функционал должен быть следующим:

1. Клиент может присоединиться к серверу, введя логин

2. Клиент может отправить сообщение другому клиенту по его логину

3. Клиент в реальном времени принимает сообщения от других клиентов

**Вариант 26**

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

**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**

При запуске client.cpp требуется ввести логин, если пользователь с таким логином уже зарегистрирован, то операция не удастся, в ином случае, операция пройдет успешно, и клиент с введенным логином будет занесен в базу (map<string, bool> logged\_in, где string — логин клиента). Кроме того, сервер выведет сообщение о регистрации клиента. Для передачи логинов создается отдельная пара сокета типа Reply – Request на сервере и клиенте соответственно.

При регистрации каждого клиента, создается два сокета типа Push — Pull, с помощью которых сообщения от клиента отправляются на сервер, и клиент получает сообщения от сервера соответственно. В client.cpp: push(сокет 1) — для отправки сообщений серверу, pull(сокет 2) — для получения сообщений от сервера. Аналогично в файле server.cpp: push(сокет 2) — для отправки сообщений клиенту, pull(сокет 1) — для получения сообщений от клиента.

Существует общий набор команд, соответствущий заданным условиям варианта (данные команды принимает client.cpp, обрабатывает и отправляет на дальнейшую обработку server.cpp):

* send- отправляет сообщение другому клиенту. При вводе этой команды, клиент должен ввести никнейм (логин) пользователя, которому он хочет отправить сообщение и само сообщение. Client.cpp формирует строку сообщения (message\_string), которая отправится на сервер: recipient — логин получателя, login — логин отправителя, client\_message — само сообщение. Далее сервер обрабатывает команду и определяет, существует ли логин получателя, если да, то он отправляет сообщение на client.cpp этого получателя, если не существует, то сообщение об этом уходит обратно отправителю.
* history – загружает историю сообщений клиента. Существует структура map<string, map<string, string>> history\_of\_messages, которая хранит историю переписок пользователей. В ней string1 — имя отправителя, string2 — имя получателя, string3 — сообщение. Эта история обновляется с помощью метода history\_save.
* exit – разлогинивает клиента с сервера. При этом закрываются все сокеты, связанные с этим клиентом и посылается сообщение на сервер, который удаляет логин данного клиента из базы map<string, bool> logged\_in и выводит сообщение об этом.

Так же в клиенте создается отдельный поток для получения сообщений с сервера (так называемая очередь сообщений), а на сервере создаются потоки для обслуживания каждого клиента, таким образом в программе не использовались mutex или semaphore.

**Основные файлы программы**

**server.cpp**

#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] = history\_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(), message\_string.size());

if (!socket.send(message\_back, zmq::send\_flags::none)){

cout << "Send message error." << "\n";

}

}

string receive\_message(zmq::socket\_t& socket) {

zmq::message\_t message\_main;

socket.recv(&message\_main);

string answer(static\_cast<char\*>(message\_main.data()), message\_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, history\_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 << "\n";

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;

cout << "User " << nickname << " logged out." << "\n";

}

}

}

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." << "\n";

send\_message("0", socket\_for\_login);

}

else{

logged\_in[nickname] = true;

cout << "User " << nickname << " logged in with id " << id << "\n";

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();

}

}

}

**client.cpp**

#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(), message\_string.size());

if (!socket.send(message\_back, zmq::send\_flags::none)) {

cout << "Send message error." << "\n";

}

}

string receive\_message(zmq::socket\_t &socket) {

zmq::message\_t message\_main;

socket.recv(&message\_main);

string answer(static\_cast<char \*>(message\_main.data()), message\_main.size());

return answer;

}

void process\_terminal(zmq::socket\_t &pusher, string login) {

string command = "";

cout << "Enter command:" << "\n";

while (cin >> command)

{

if (command == "send") {

cout << "Enter nickname of recipient:" << "\n";

string recipient = "";

cin >> recipient;

cout << "Enter your message:" << "\n";

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);

} else if (command == "history") {

string message\_string = "history";

send\_message(message\_string, pusher);

} else if (command == "exit") {

send\_message("exit " + login, pusher);

break;

} else {

cout << "There is no such command." << "\n";

}

cout << "Enter command:" << "\n";

}

}

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 << ":" << "\n" << mes\_to\_me << "\n";

} else if (command == "history") {

string history;

for (int i = 8; i < recieved\_message.size(); ++i) {

history += recieved\_message[i];

}

cout << history << "\n";

} else if (command == "no") {

cout << "There is no such user." << "\n";

} 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: " << "\n";

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" << "\n";

\_exit(0);

} else if (recieved\_message == "1") {

cout << "========= All commands ========== \n send \n history \n exit \n";

cout << "================================= \n";

zmq::context\_t context1(1);

zmq::socket\_t puller(context1, ZMQ\_PULL); // прием сообщений от сервера

puller.connect("tcp://localhost:3" + to\_string(getpid()));

zmq::context\_t context2(1);

zmq::socket\_t pusher(context2, ZMQ\_PUSH); // отправка сообщений клиенту

pusher.connect("tcp://localhost:3" + to\_string(getpid() + 1));

thread thr;

thr = thread(process\_server, ref(puller));

thr.detach();

process\_terminal(pusher, login);

thr.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;

}

**Пример работы**

alexandra@alexandra-HVY-WXX9:~/Рабочийстол/mai\_os\_labs/cp/src/build$ ./server

User zxcv logged in with id 9064

User ghj logged in with id 9241

zxcv

1234567

ghj

2

zxcv

3

ghj

2

zxcv

3

4

ghj

2

5

zxcv

3

4

User qwe logged in with id 10135

User ghj logged in with id 10343

^C

alexandra@alexandra-HVY-WXX9:~/Рабочийстол/mai\_os\_labs/cp/src/build$ ./client

Enter login:

zxcv

========= All commands ==========

send

history

exit

=================================

Enter command:

Message from ghj:

1234567

Message from ghj:

1

send

Enter nickname of recipient:

ghj

Enter your message:

2

Enter command:

send

Enter nickname of recipient:

zxcv

Enter your message:

3

Enter command:

Message from zxcv:

3

history

Enter command:

ghj

2

zxcv

3

send

Enter nickname of recipient:

zxcv

Enter your message:

4

Enter command:

Message from zxcv:

4

history

Enter command:

ghj

2

zxcv

3

4

send

Enter nickname of recipient:

ghj

Enter your message:

5

Enter command:

history

Enter command:

ghj

2

5

zxcv

3

4

exit

alexandra@alexandra-HVY-WXX9:~/Рабочий стол/mai\_os\_labs/cp/src/bu

ild$ ./client

Enter login:

ghj

========= All commands ==========

send

history

exit

=================================

Enter command:

send zxcv

Enter nickname of recipient:

Enter your message:

1234567

Enter command:

history

Enter command:

zxcv

1234567

send

Enter nickname of recipient:

zxcv

Enter your message:

1

Enter command:

Message from zxcv:

2

Message from zxcv:

5

exit

**Вывод**

Благодаря данной курсовой работе я подробнее изучила механизмы работы с ZeroMQ, выполнила поставленное задание, а именно: написала клиент-серверную систему для передачи мгновенных сообщений. Кроме того, я закрепила навыки работы с потоками и очередями сообщений.

Полученные в ходе выполения курсовой работы навыки могут помочь мне в реализации дальнейших проектов.