Специальность: 09.02.07 Информационные системы и программирование

Проверил:

\_\_\_\_\_\_\_\_\_\_\_ Антонов М.Б.

Дата: «28» марта 2020 г.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

(Университет ИТМО)

факультет среднего профессионального образования

Отчет

о Лабораторной работе №14

по теме: Работа с сокетами

по дисциплине: Разработка программных модулей

Выполнил:

студент группы 2333

\_\_\_\_\_\_\_\_\_\_\_ Агапова Е.А.

Санкт-Петербург 2020

Содержание

[1 Постановка задачи 3](#_Toc35962118)

[1.1 Цель 3](#_Toc35962119)

[1.2 Задачи 3](#_Toc35962120)

[1.3 Задание 3](#_Toc35962121)

[2 Исходный код 5](#_Toc35962122)

[3 ЛИСТИНГ ВЫполнения программы 11](#_Toc35962123)

[4 Вывод 12](#_Toc35962124)

1. Постановка задачи
   1. Цель

Познакомиться с основами работы QTcpSocket в приложениях Qt.

* 1. Задачи

1. изучить основы работы с сокетными приложениями;
2. создать простейшее приложение-чат.
   1. Задание

Допускается выполнение лабораторной работы по группам 2 человека. Если студент работает один, то функции помеченные (+) являются не обязательными.

Написать клиент и сервер программы чата.

**Требования к серверу:**

* сервер должен хранить последние 10 сообщений;
* при подключении нового пользователя оповестить существующих о подключении нового пользователя;
* новому пользователю отправить последние 10 сообщений чата;
* обработать команду от пользователя: отключение от чата (Qt::disconnect);
* (+) при закрытии программы последние 10 сообщений должны быть сохранены в файл;
* (+) при открытии программы последние 10 сообщений при их доступности должны быть прочитаны из файла.

**Требования к клиенту**

* клиент должен задать ник перед началом общения с чатом;
* сообщения на клиенте должны отображаться в виде: <%nick-name%> %message%;
* клиент может отключиться от сервера в любой момент;
* (+) предусмотреть недоступность сервера;
* (+) ники пользователей должны выделяться цветом.

Общее ограничение на лабораторную работу:

* использовать разбитие на файлы основной программы (в main.cpp только функция main.cpp);
* документировать весь код используя аннотации Doxygen.

Требования к отчёту:

* цели лабораторной работы;
* задачи лабораторной работы;
* выполняемое задание;
* диаграмма(ы) классов;
* диаграммы деятельности (для методов и/или функций больше 10 строк кода (исключая форматирования и пустые строки));
* исходный код программы с комментариями (2 пробела вместо 4, 10pt, моноширинный шрифт);
* листинг выполнения программы;
* исходный код в виде файлов для запуска на произвольном ПК.

1. Исходный код

Листинг 2.1 – Код программы server

/\*\*

\* @file main.cpp

\* @author Agapova Ekaterina

\*/

#include <QCoreApplication>

#include "server.h"

int main(int argc, char \*argv[])

{

QCoreApplication a(argc, argv);

Server serv;

return a.exec();

}

Листинг 2.2 – Заголовочный файл server.h

/\*\*

\* @file server.h

\* @brief server

\* @author Agapova Ekaterina

\*/

#ifndef SERVER\_H

#define SERVER\_H

#include <QObject>

#include <QtNetwork>

class Server : public QObject

{

Q\_OBJECT

private:

/\*\*

\* @brief connecting to the tcpServer

\*/

QTcpServer\* tcpS;

/\*\*

\* @brief connecting to the client

\*/

QTcpSocket\* client;

/\*\*

\* @brief list of clients

\*/

QList <QTcpSocket\*> clients;

QFile file;

QStringList list;

int k = 0;

public:

/\*\*

\* @brief Server Constructor

\* @param parent

\*/

explicit Server(QObject \*parent = nullptr);

/\*\*

\* @brief Server Destructor

\*/

~Server();

private slots:

/\*\*

\* @brief connect User

\*/

void connectUser();

/\*\*

\* @brief message from client

\*/

void runClientAction();

/\*\*

\* @brief disconnect User

\*/

void disconnectUser();

/\*\*

\* @brief send 10 messages for new user

\*/

void send10Message();

signals:

/\*\*

\* @brief server close

\*/

void serverClose();

};

#endif // SERVER\_H

Листинг 2.3 – Заголовочный файл server.cpp

/\*\*

\* @file server.cpp

\* @brief server

\* @author Agapova Ekaterina

\*/

#include "server.h"

#include <QDebug>

Server::Server(QObject \*parent) : QObject(parent)

{

tcpS = new QTcpServer(this);

connect(tcpS, &QTcpServer::newConnection,

this, &Server::connectUser);

if (tcpS->listen(QHostAddress::Any, 1337)) {

qInfo() << "Server started";

} else {

qWarning() << "Unable to start server";

}

connect(this, &Server::serverClose,

this, &QCoreApplication::quit);

clients.clear();

list.clear();

}

Server::~Server()

{

emit serverClose();

}

void Server::connectUser()

{

qInfo() << "We have new user!";

foreach (client, clients) {

QTcpSocket\* clientSock = (QTcpSocket\*) client;

QTextStream str (clientSock);

str << " 1 new user";

}

client = tcpS->nextPendingConnection();

clients.push\_back(client);

this->send10Message();

connect(client, &QTcpSocket::readyRead,

this, &Server::runClientAction);

connect(client, &QTcpSocket::disconnected,

this, &Server::disconnectUser);

}

void Server::runClientAction()

{

k++;

QTcpSocket\* clientSocket = (QTcpSocket\*) sender();

QTextStream stream (clientSocket);

QString nickname;

QString message;

for (int i = 0;i < 2;i++){

if (i == 0) stream >> nickname;

if (i == 1) stream >> message;

}

qInfo() << "Get message from "<< nickname << ": " << message;

foreach (client, clients){

QTcpSocket\* clSock = (QTcpSocket\*) client;

QTextStream str (clSock);

str << nickname + ": " + message;

}

file.setFileName("file.txt");

list.push\_back(nickname + " " + message);

if(!file.open(QIODevice::WriteOnly)){

qInfo() << "File isn't open.";

}

if (k > 10){

list.removeFirst();

}

QTextStream \_stream(&file);

foreach(QString str, list){

if (str != ""){

\_stream << str << endl;

}

}

file.close();

}

void Server::disconnectUser() {

qInfo() << "We lost user!";

foreach (client, clients) {

QTcpSocket\* clientSock = (QTcpSocket\*) client;

QTextStream str (clientSock);

str << " 1 user left.";

}

clients.removeOne(client);

client->disconnect();

if(clients.size() == 0)

emit serverClose();

}

void Server::send10Message() {

file.setFileName("file.txt");

if(!file.open(QIODevice::ReadOnly)){

qInfo() << "File isn't open.";

}

QTcpSocket\* clSock = (QTcpSocket\*) client;

QTextStream str (clSock);

QTextStream \_stream(&file);

int i = 0;

while(!\_stream.atEnd()){

QString st;

i++;

\_stream >> st;

if (((i % 2) == 1)&&(st != ""))

str << st + ": ";

else {

str << st;

if(!\_stream.atEnd())

str<<endl;

}

}

if(k == 0)

k += i/2;

file.close();

}

Листинг 2.4 – Код программы client

/\*\*

\* @file main.cpp

\* @author Agapova Ekaterina

\*/

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

Листинг 2.5 – Заголовочный файл mainwindow.h

/\*\*

\* @file mainwindow.h

\* @brief client

\* @author Agapova Ekaterina

\*/

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QtNetwork>

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

private:

/\*\*

\* @brief socket of connection

\*/

QTcpSocket \*sockCl;

Ui::MainWindow \*ui;

QString nick = " ";

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

/\*\*

\* @brief send messages

\*/

void execButtonAction();

/\*\*

\* @brief read Socket

\*/

void readSocket();

/\*\*

\* @brief set nickname of user

\*/

void setNick();

/\*\*

\* @brief serverDisconnect

\*/

void serverDisconnect();

};

#endif // MAINWINDOW\_H

Листинг 2.6 – Заголовочный файл mainwindow.cpp

/\*\*

\* @file mainwindow.cpp

\* @brief client

\* @author Agapova Ekaterina

\*/

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QDebug>

#include <QMessageBox>

#include <QWidget>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent), ui(new Ui::MainWindow)

{

ui->setupUi(this);

sockCl = new QTcpSocket(this);

sockCl->connectToHost("localhost", 1337, QTcpSocket::ReadWrite);

if(sockCl->waitForConnected(30000)) {

qInfo() << "Connected!";

qInfo() << sockCl->peerAddress();

} else{

QMessageBox::information(NULL,QObject::tr("Информация"),tr("Нет подключения к серверу."));

}

connect(sockCl, &QTcpSocket::readyRead,

this, &MainWindow::readSocket);

connect(ui->pushButton, &QPushButton::clicked,

this, &MainWindow::execButtonAction);

connect(ui->setNick, &QPushButton::clicked,

this, &MainWindow::setNick);

connect(sockCl, &QTcpSocket::disconnected,

this, &MainWindow::serverDisconnect);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::execButtonAction()

{

if(ui->nickname->text().toUtf8() == "" || nick == " "){

QMessageBox::information(NULL,QObject::tr("Ошибка"),tr("Введите ник"));

return;

} else if (ui->message->text().toUtf8() == ""){

QMessageBox::information(NULL,QObject::tr("Ошибка"),tr("Введите сообщение"));

return;

}else

sockCl->write(ui->nickname->text().toUtf8()

+ " " + ui->message->text().toUtf8());

}

void MainWindow::readSocket()

{

QString data = QString(sockCl->readAll());

ui->textEdit->append(data);

}

void MainWindow::setNick(){

nick = ui->nickname->displayText();

ui->nickname->setReadOnly("True");

}

void MainWindow::serverDisconnect()

{

QMessageBox::information(NULL,QObject::tr("Информация"),tr("Сервер был отключен"));

}

1. ЛИСТИНГ ВЫполнения программы

На рисунках 5-6 представлены интерфейсы работы программы.

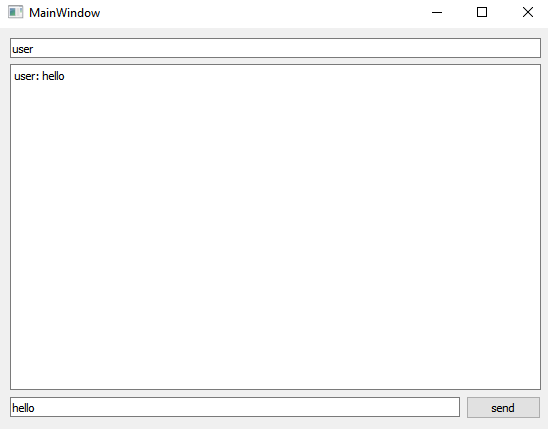


Рисунок 5 – Интерфейс чата

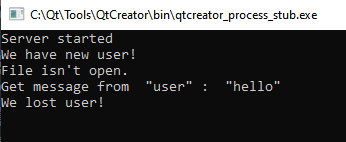


Рисунок 6 – Консоль сервера

1. Вывод

В ходе выполнения лабораторной работы была написана программа, которая реализует работу простейшего чата. При подключении/удалении пользователя всем существующим отправляется сообщение. Последние 10 сообщений сохраняются в файл и восстанавливаются при новом запуске.