ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАЕТЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Кафедра ПМиК

Курсовая работа

по дисциплине «Операционные системы»

«Сетевой чат на основе сокетов»

Выполнил: студент гр. ИП-013

Копытина Т.А.

Проверил: ассистент кафедры ПМиК

Нужнов А.В.

Новосибирск 2022г.

Оглавление

Задание:

Разработать сетевое приложение, реализующего сетевую игру/чат на основе сокетов.

Программная реализация

Программа была написана на языке C++ с использованием Qt5.10.

Проект включает две основные части в виде сервера (Server) и клиента (Client).

Имеется возможность множественного подключения клиентов к серверу (общий чат). Для подключения клиентов должен быть запущен сам сервер. Клиент имеет возможность подключаться. Также выводятся сообщения о подключении других пользователей.

Результаты работы программы

Сначала запускаем сервер. Видим сообщение о том, что хост стартовал.

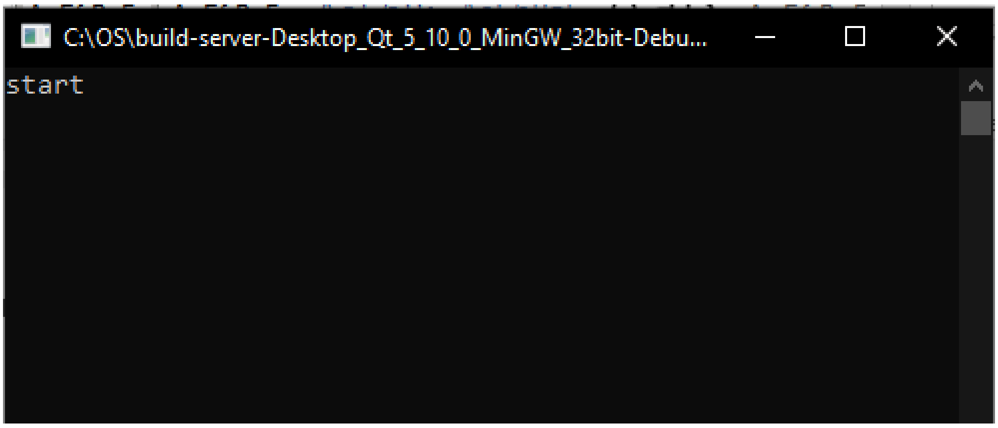

Рисунок 1. Сервер

Далее запускаем клиентскую часть.

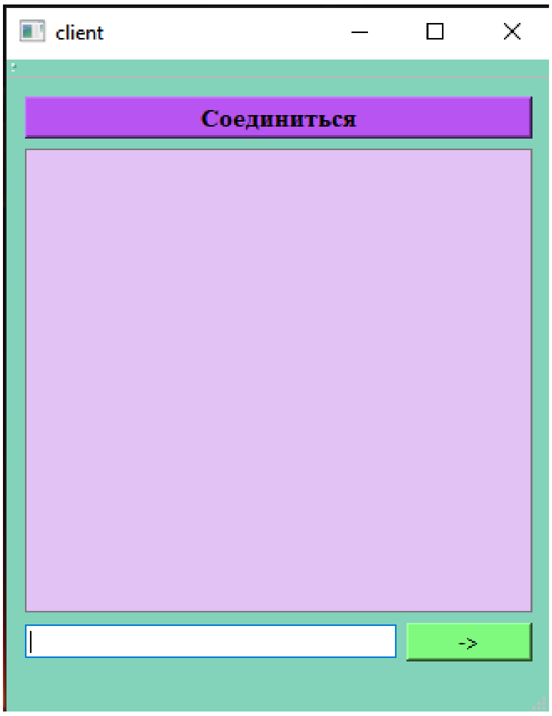

Рисунок 2. Клиент номер 1

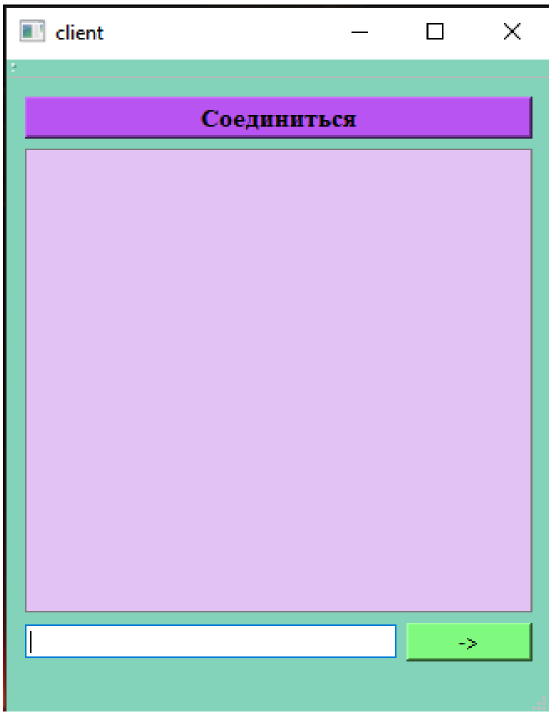

Рисунок 3. Клиент номер 2

Подключаем клиентов к серверу

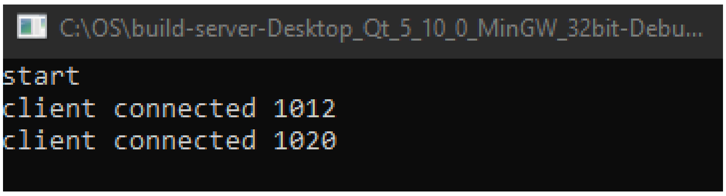

Рисунок 4. Подключение клиентов

Отправка сообщений в чате и трансляция об этих сообщениях на сервере.

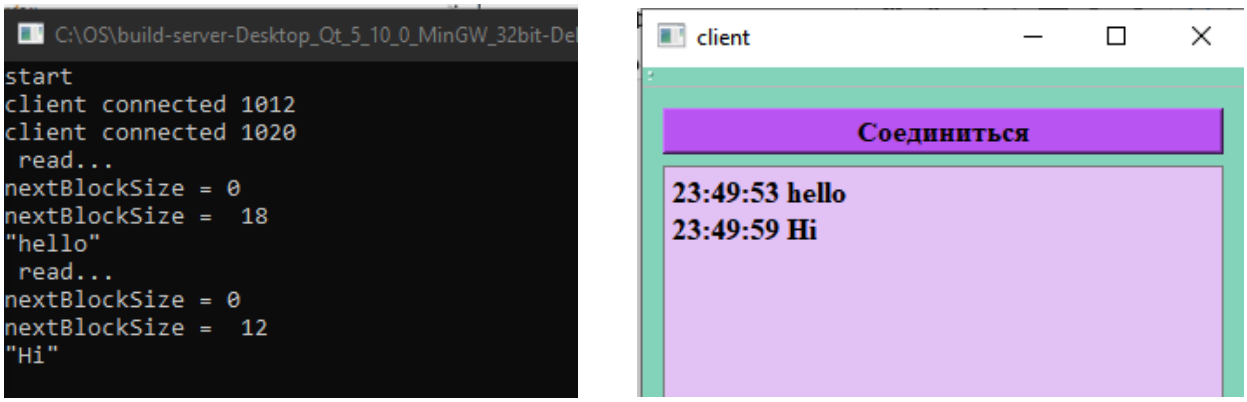

Рисунок 5-6. Передача сообщений от клиента серверу и от сервера клиентам.

Листинг

файл server.h

```
#ifndef SERVER_H
#define SERVER_H
#include <QTcpServer>
#include <QTcpSocket>
#include <QVector>
#include <QDataStream>
#include <QTime>

class Server : public QTcpServer
{
    Q_OBJECT
public:
    Server();
    QTcpSocket *socket;
private:
    QVector <QTcpSocket*> Socket;
    QByteArray Data;
    void SendToClient(QString str);
    quint16 nextBlockSize;
public slots:
    void incomingConnection(qintptr socketDescriptor);
    void slotReadyRead();
};
#endif // SERVER_H
```

Файл server.cpp

```
  #include "server.h"

Server::Server()
{
    if(this->listen(QHostAddress::Any, 2301))//запуск сервера
```

```
    {
        qDebug() << "start";
    }
    else
    {
        qDebug() << "Error";
    }
    nextBlockSize = 0;
}
void Server::incomingConnection(qintptr socketDescriptor)// обработка подключения клиента к серверу
{
    socket = new QTcpSocket;
    socket->setSocketDescriptor(socketDescriptor);
    connect(socket, &QTcpSocket::readyRead, this, &Server::slotReadyRead);
    connect(socket, &QTcpSocket::disconnected,socket, &QTcpSocket::deleteLater); //при откл. клиента сокет сразу удалится


    Socket.push_back(socket);
    qDebug() << "client connected" << socketDescriptor;
}
void Server::slotReadyRead() //обратотка сообщений от клиента
{
    socket = (QTcpSocket*)sender(); //тот сокет с которого пришел запрос
    QDataStream in(socket);
    in.setVersion(QDataStream::Qt_5_10);
    if(in.status() == QDataStream::Ok)
    {
        qDebug() << " read...";
        for(;;)
        {
            if(nextBlockSize == 0)
            {
```

```
                qDebug() << "nextBlockSize = 0";
                if(socket->bytesAvailable() < 2)
                {
                    qDebug() << "Data < 2, break";
                    break;
                }
                in >> nextBlockSize;
                qDebug() << "nextBlockSize = " << nextBlockSize;
            }
            if(socket->bytesAvailable() < nextBlockSize)
            {
                qDebug() << "Data nit full, break";
                break;
            }
            QString str;
            QTime time;
            in >>time>> str;
            nextBlockSize = 0;
            qDebug() << str;
            SendToClient(str);
            break;
        }
    }
    else
    {
        qDebug() << "DataStream error";
    }
}
void Server::SendToClient(QString str) //отправка сообщений клиенту
{
    Data.clear();
    QDataStream out(&Data, QIODevice::WriteOnly);
    out.setVersion(QDataStream::Qt_5_10);
    out <<quint16(0)<< QTime::currentTime() << str;
```

```
    out.device()->seek(0);
    out << quint16(Data.size() - sizeof(quint16));

    for (int i = 0; i < Socket.size(); i++)
    {
        Socket[i]->write(Data);
    }
}
```

Файл Client.h

```
#ifndef CLIENT_H
#define CLIENT_H

#include <QMainWindow>
#include <QTcpSocket>
#include <QTime>

namespace Ui {
class client;
}
class client : public QMainWindow
{
    Q_OBJECT
public:
    explicit client(QWidget *parent = 0);
    ~client();

private slots:
    void on_pushButton_clicked();
    void slotReadyRead();
    void on_pushButton_2_clicked();
    void on_lineEdit_returnPressed();
private:
    Ui::client *ui;
    QTcpSocket *socket;
```

```
    QByteArray Data; //то что переходит от сервера к клиенту
    void SendToServer(QString str);
    quint16 nextBlocksSize;
};
#endif // CLIENT_H
```

Файл Client.cpp

```
#include "client.h"
#include "ui_client.h"

client::client(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::client)
{
    ui->setupUi(this);
    socket = new QTcpSocket(this);

connect(socket,&QTcpSocket::readyRead,this,&client::slotReadyRead);
    connect(socket, &QTcpSocket::disconnected, socket,
&QTcpSocket::deleteLater);
    nextBlocksSize = 0;
}
client::~client()
{
    delete ui;
}
void client::on_pushButton_clicked()
{
    socket->connectToHost("127.0.0.1", 2301); //соединение клиента с
сервером через хост
}
void client::SendToServer(QString str) // отправка сообщений серверу
{
    Data.clear();
    QDataStream out(&Data, QIODevice::WriteOnly);
    out.setVersion(QDataStream::Qt_5_10);
```

```
    out << quint16(0) <<QTime::currentTime()<< str;
    out.device()->seek(0);
    out << quint16(Data.size() - sizeof(quint16));
    socket->write(Data);

    ui->lineEdit->clear();//очищение линии сообщения
}
void client::slotReadyRead()//обработка сообщений от сервера
{
    QDataStream in(socket);
    in.setVersion(QDataStream::Qt_5_10);
    if(in.status() == QDataStream::Ok)
    {
        for(;;)
        {
            if(nextBlocksSize == 0) //считывание размера блока
            {
                if(socket->bytesAvailable() < 2) //для чтения доступно
не меньше 2 байт
                {
                    break;
                }
                in >> nextBlocksSize;
            }
            if(socket->bytesAvailable() < nextBlocksSize) //сравнение
с количеством байт которое пришло от сервера
            {
                break;
            }
            QString str;
            QTime time;
            in >> time >> str;
            nextBlocksSize = 0;
            ui->textBrowser->append(time.toString() + " " + str);
        }
```

```
    }
    else
    {
        ui->textBrowser->append("read error");
    }
}
void client::on_pushButton_2_clicked()
{
    SendToServer(ui->lineEdit->text());
}
void client::on_lineEdit_returnPressed()
{
    SendToServer(ui->lineEdit->text());
}
```