Минобрнауки России  
ФГБОУ ВПО «НИУ МЭИ»  
ИВТИ  
Кафедра математического и компьютерного моделирования

Сетевые технологии

**Лабораторная работа №3**

«Клиентская подсистема клиент-серверной системы на основе класса TcpClient языка C# »

Работу выполнила:  
 студент гр. А-14-18 Рылов Д.С.  
Работу принял:  
доцент Князев А.В.

Москва  
2022

1. Задание на работу

Разработать клиентскую подсистему клиент-серверной системы, реа- лизующую заданный протокол обмена данными.

Программа должна быть разработана в среде Visual Studio на языке C# с использованием класса TcpClient.

Клиент должен иметь удобный интерфейс (меню, окна и т.д.).

Настройки клиента должны включать задание удалённого адреса и порта.

Клиент должен реализовать дополнительные настройки, обеспечи- вающие гибкость выполнения соответствующего задания.

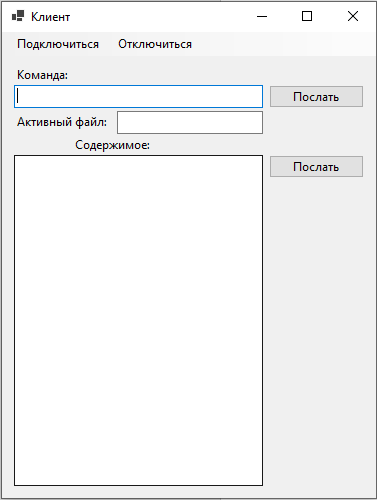
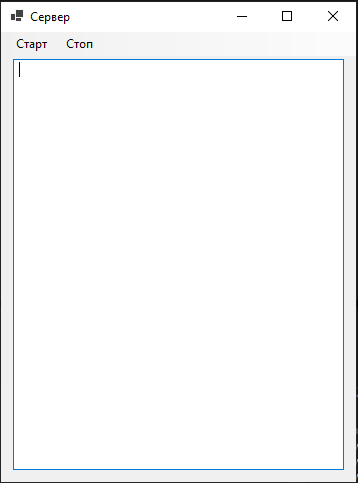
13.Рылов Д. С.  
Клиент посылает команду Put <имя файла>. Сервер создаёт пустой файл в  
заданном каталоге и высылает клиенту подтверждение. Клиент посылает  
содержимое файла. – Сервер записывает информацию в файл и возвращает квитанцию.

1. Описание работы программы

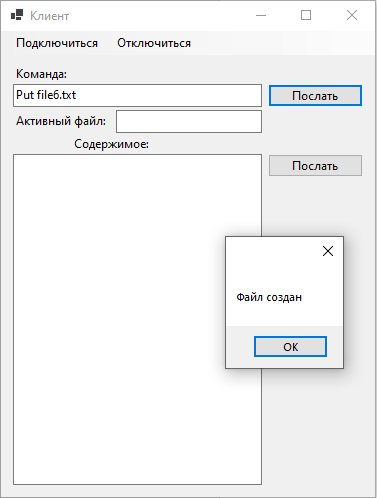
Клиент серверное приложения представляет из себя сервер с файловой системы. Клиент имеет возможность создать файл и заполнить его.

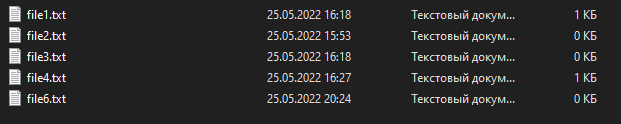
1. Тесты

Изначальный вид.

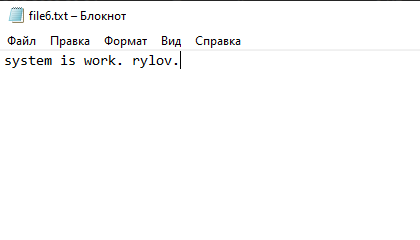
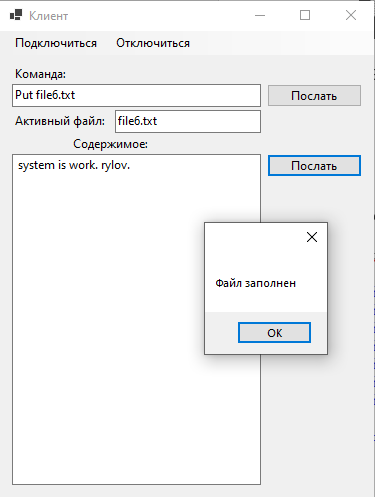


Создание файла:





Запись в файл:



Листинг Программы

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net.Sockets;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ClientLab3

{

class Client

{

TcpClient client;

TextBox currentfile;

TextBox infile;

public Client(TextBox infile, TextBox currentfile)

{

this.infile = infile;

this.currentfile = currentfile;

}

public void Connect(string IP, int port)

{

try

{

client = new TcpClient(IP, port);

}

catch (SocketException e)

{

MessageBox.Show("Не удалось подключиться к серверу");

}

}

public void Disconnect()

{

client.Close();

}

public enum Command

{

PutFile,

FillFile,

}

public void PutFile(string filename)

{

try

{

byte[] buffer = BitConverter.GetBytes((int)Command.PutFile);

var network = client.GetStream();

network.Write(buffer);

SendString(filename);

buffer = new byte[sizeof(bool)];

network.Read(buffer);

if (BitConverter.ToBoolean(buffer))

{

MessageBox.Show("Файл создан");

currentfile.Text = filename;

}

else

{

MessageBox.Show("Произошла ошибка при создании файла");

}

}

catch (Exception)

{

MessageBox.Show("Не подключен к серверу");

}

}

public void FillFile (string currentfile, string infile)

{

try

{

byte[] buffer = BitConverter.GetBytes((int)Command.FillFile);

var network = client.GetStream();

network.Write(buffer);

SendString(currentfile);

SendString(infile);

buffer = new byte[sizeof(bool)];

network.Read(buffer);

if (BitConverter.ToBoolean(buffer))

{

MessageBox.Show("Файл заполнен");

this.infile.Text = "";

}

else

{

MessageBox.Show("Произошла ошибка при заполнении файла");

}

}

catch (Exception)

{

MessageBox.Show("Не подключен к серверу");

}

}

private void SendString(string str)

{

var network = client.GetStream();

network.Write(BitConverter.GetBytes(str.Length));

network.Write(Encoding.UTF8.GetBytes(str));

}

}

}

namespace ClientLab3

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

client = new Client(infile, currentfile);

}

Client client;

private void подключитьсяToolStripMenuItem\_Click(object sender, EventArgs e)

{

client.Connect("127.0.0.1", 5000);

}

private void отключитьсяToolStripMenuItem\_Click(object sender, EventArgs e)

{

client.Disconnect();

}

private void button1\_Click(object sender, EventArgs e)

{

switch (command.Text.Split(new char[] { ' ' })[0])

{

case "Put":

client.PutFile(command.Text.Split(new char[] { ' ' })[1]);

break;

default:

MessageBox.Show("Неправильная команда");

break;

}

}

private void button2\_Click(object sender, EventArgs e)

{

if (currentfile.Text == "")

{

MessageBox.Show("Какой файл заполнять???");

}

else

{

client.FillFile(currentfile.Text, infile.Text);

}

}

}

}

using System.Net;

using System.Net.Sockets;

using System.Text;

using System.IO;

namespace ServerLab3

{

class Server

{

TcpListener listener;

TextBox log;

SortedSet<Socket> connections;

public Server(TextBox \_log)

{

log = \_log;

connections = new SortedSet<Socket>();

}

public void Start(int port)

{

//IPEndPoint localEndPoint = new IPEndPoint(IPAddress.Any, port);

//listener = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

IPAddress localAddr = IPAddress.Parse("127.0.0.1");

listener = new TcpListener(localAddr, 5000);

try

{

listener.Start();

log.Invoke((MethodInvoker)delegate { log.AppendText("Server start" + Environment.NewLine); });

while (true)

{

// Set the event to nonsignaled state.

Socket client = listener.AcceptSocket();

connections.Add(client);

LogClient(client);

Task t = new Task(() => ClientHandle(client));

t.Start();

}

}

catch (Exception e)

{

MessageBox.Show("Server stopped listening " + e.Message);

}

}

private void LogClient(Socket client)

{

var ip = (IPEndPoint)client.RemoteEndPoint;

log.Invoke((MethodInvoker)delegate { log.AppendText(ip.Address.ToString() + ":" + ip.Port.ToString() + " connected to server" + Environment.NewLine); });

}

private void LogDisconnectClient(Socket client)

{

var ip = (IPEndPoint)client.RemoteEndPoint;

log.Invoke((MethodInvoker)delegate { log.AppendText(ip.Address.ToString() + ":" + ip.Port.ToString() + " disconnected from server" + Environment.NewLine); });

}

public void Stop()

{

listener.Stop();

connections.Clear();

}

public void ClientHandle(Socket client)

{

byte[] buffer = new byte[sizeof(int)];

while (client.Receive(buffer, buffer.Length, SocketFlags.None) > 0)

{

switch ((Command)BitConverter.ToInt32(buffer, 0))

{

case Command.Putfile:

Putfile(client);

break;

case Command.FillFile:

FillFile(client);

break;

}

}

LogDisconnectClient(client);

client.Shutdown(SocketShutdown.Both);

client.Close();

connections.Remove(client);

}

public enum Command

{

Putfile,

FillFile,

}

private void Putfile(Socket handler)

{

string filename = RecvString(handler);

bool result;

string path = "C:/Users/Dima/Desktop/Учеба/8 семестр/Сетевые технологии/лабы/лаба3/proba/Server/ServerLab3/bin/Debug/net6.0-windows/filesystem/";

try

{

FileStream fstream = new FileStream(path + filename, FileMode.Create);

fstream.Close();

result = true;

}

catch (Exception)

{

result = false;

}

handler.Send(BitConverter.GetBytes(result));

}

private void FillFile(Socket handler)

{

string filename = RecvString(handler);

string infile = RecvString(handler);

string path = "C:/Users/Dima/Desktop/Учеба/8 семестр/Сетевые технологии/лабы/лаба3/proba/Server/ServerLab3/bin/Debug/net6.0-windows/filesystem/";

bool result;

try

{

FileStream fstream = new FileStream(path + filename, FileMode.Create, FileAccess.Write);

// преобразуем строку в байты

byte[] buffer = Encoding.Default.GetBytes(infile);

// запись массива байтов в файл

fstream.Write(buffer, 0, buffer.Length);

fstream.Close();

result = true;

}

catch (Exception)

{

result = false;

}

handler.Send(BitConverter.GetBytes(result));

}

private void SendString(Socket handler, string str)

{

handler.Send(BitConverter.GetBytes(str.Length));

handler.Send(Encoding.UTF8.GetBytes(str));

}

private string RecvString(Socket handler)

{

byte[] buffer = new byte[sizeof(int)];

handler.Receive(buffer);

if (BitConverter.ToInt32(buffer) > 0)

{

buffer = new byte[BitConverter.ToInt32(buffer)];

handler.Receive(buffer);

return Encoding.UTF8.GetString(buffer);

}

else

return "";

}

private int RecvInt(Socket handler)

{

byte[] buffer = new byte[sizeof(int)];

handler.Receive(buffer);

return BitConverter.ToInt32(buffer);

}

}

}