МОСКОВСКИЙ ЭНЕРГЕТИЧЕСКИЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Кафедра математического моделирования

Сетевые технологии

Лабораторная работа №2

“Клиент-серверная система на основе класса Socket языка C#”

Студент: Чухнина П.

Группа: А-14-16

Преподаватель: Князев А.В.

Москва

2020

**Задание.**

Разработать клиент-серверную систему, реализующую заданный протокол обмена данными.

Сервер и клиент должны иметь удобный интерфейс (меню, окна и т.д.).

Сервер должен вести учет соединений.

Настройки сервера должны включать задание прослушиваемого порта, имя журнала учета, рабочую директорию.

Настройки клиента должны включать задание удаленного адреса и порта.

Модель Web-сервера.

Клиент посылает запрос Get. – Сервер в ответ передаёт файл с заданным именем, и клиент отображает его на экране. В файле есть ссылки на другие файлы (тоже со ссылками). Клиент может выбрать ссылку и запросить новый файл – сервер его передаёт клиенту для отображения.

**Схема, отражающая протокол работы программы.**

Сервер Клиент

socket() socket()

bind()

listen()

accept()

connect()

Ожидание запроса

от клиента

send()

Данные от клиента

recv()

Обработка данных

Ответ от сервера

send() recv()

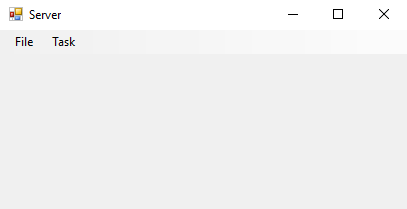
Данные от клиента – имя файла

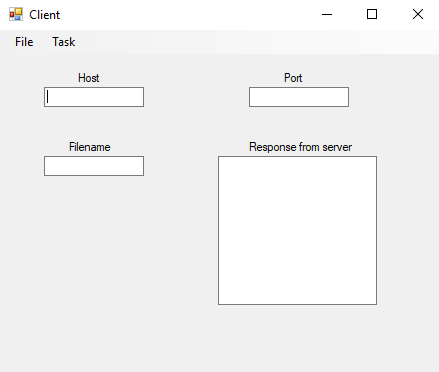
Обработка данных – открытие файла, запись содержимого файла для отправки

Ответ от сервера – содержимое файла

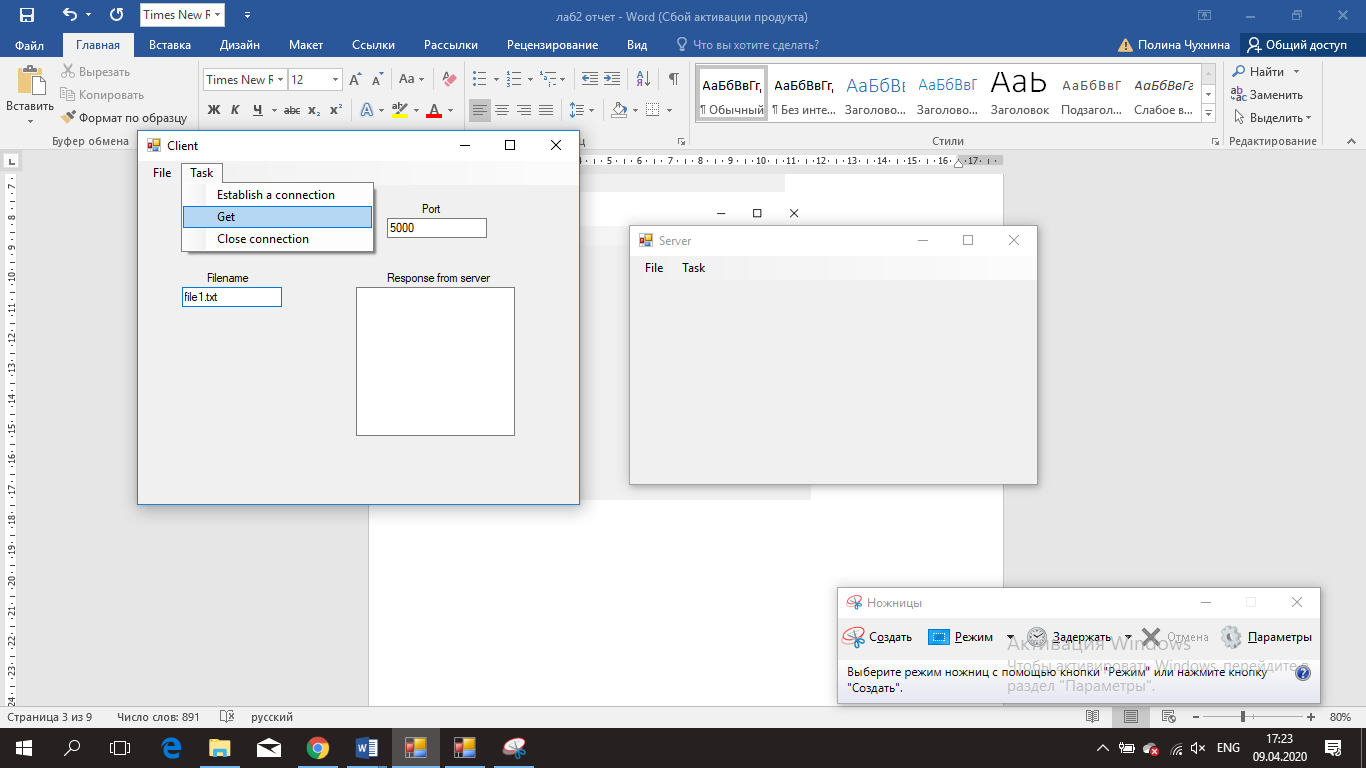
**Тесты. Распечатки экранов при работе программы.**

Начало работы

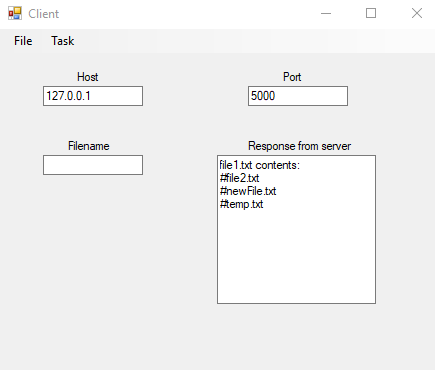




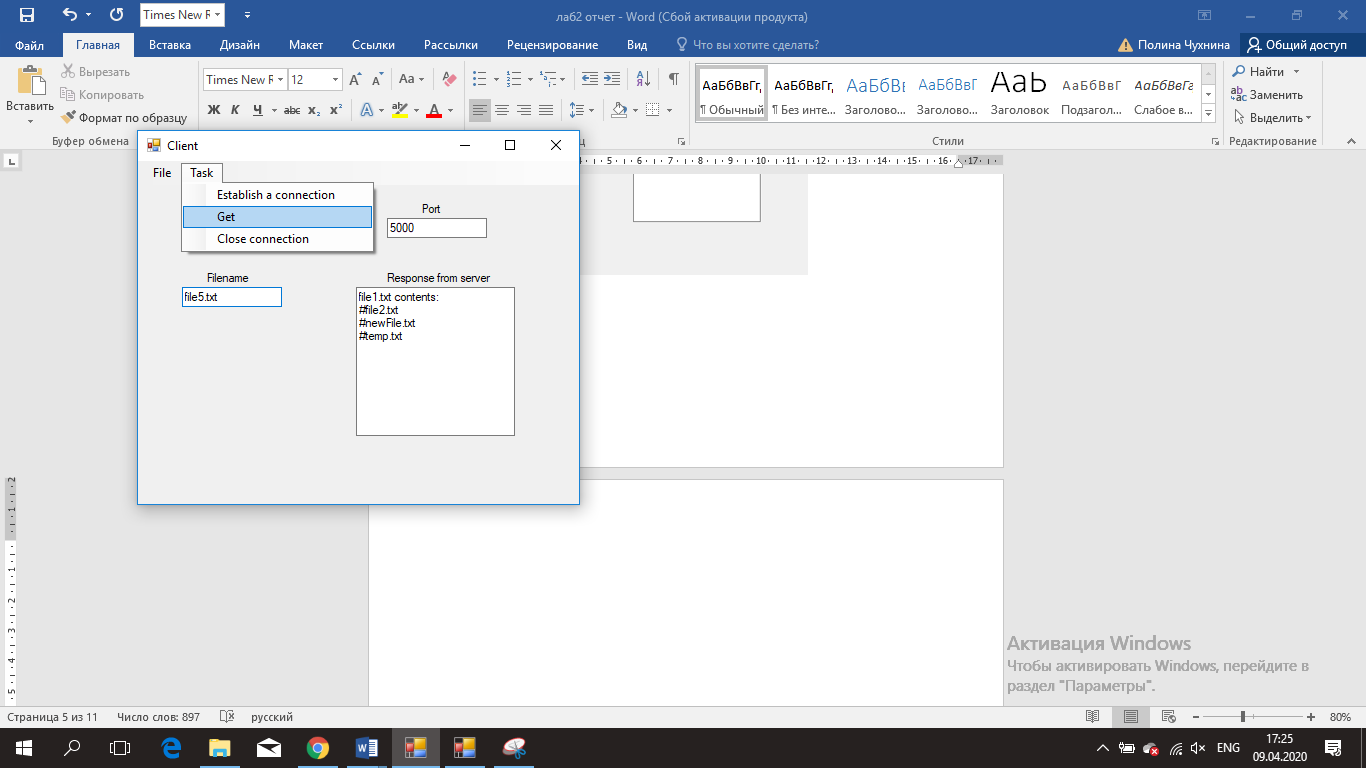
Отправка запроса на сервер



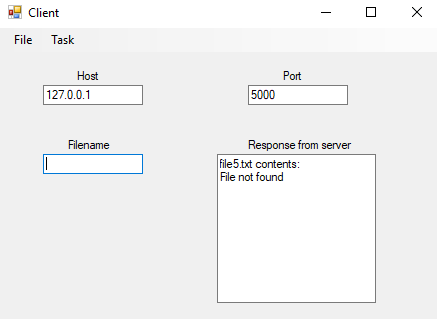
Полученный ответ



Отправка запроса с именем несуществующего файла



Полученный ответ



Содержимое лог-файла:

--------------------------------------------------------------------

Server is running 09.04.2020 17:22:41

Listening mode 09.04.2020 17:22:41

--------------------------------------------------------------------

New connection 09.04.2020 17:22:57

Received data 9 bytes 09.04.2020 17:24:41

Data sent 37 bytes 09.04.2020 17:24:41

Received data 9 bytes 09.04.2020 17:26:43

Data sent 14 bytes 09.04.2020 17:26:43

Number of connections 1

Server is stopped 09.04.2020 17:27:18

**Листинг.**

***Сервер***

*myLogger.cs* – класс, для логирования событий сервера

using System;

using System.IO;

namespace server {

public class myLogger{

private string logPath;

public myLogger(string path)

{

logPath = path;

}

public void logBind()

{

using (StreamWriter sw = new StreamWriter(logPath, true, System.Text.Encoding.Default))

{

sw.WriteLine("--------------------------------------------------------------------");

sw.WriteLine("Server is running " + DateTime.Now);

}

}

public void logListen()

{

using (StreamWriter sw = new StreamWriter(logPath, true, System.Text.Encoding.Default))

{

sw.WriteLine("Listening mode " + DateTime.Now);

}

}

public void logNewConnection()

{

using (StreamWriter sw = new StreamWriter(logPath, true, System.Text.Encoding.Default))

{

sw.WriteLine("--------------------------------------------------------------------");

sw.WriteLine("New connection " + DateTime.Now);

}

}

public void logReceive(int count)

{

using (StreamWriter sw = new StreamWriter(logPath, true, System.Text.Encoding.Default))

{

sw.WriteLine("Received data " + count + " bytes " + DateTime.Now);

}

}

public void logSend(int count)

{

using (StreamWriter sw = new StreamWriter(logPath, true, System.Text.Encoding.Default))

{

sw.WriteLine("Data sent " + count + " bytes " + DateTime.Now);

}

}

public void logCloseConnection(int count)

{

using (StreamWriter sw = new StreamWriter(logPath, true, System.Text.Encoding.Default))

{

sw.WriteLine("Number of connections " + count);

sw.WriteLine("Server is stopped " + DateTime.Now);

sw.WriteLine("--------------------------------------------------------------------");

}

}

public void logDisconnect()

{

using (StreamWriter sw = new StreamWriter(logPath, true, System.Text.Encoding.Default))

{

sw.WriteLine("Client disconnected " + DateTime.Now);

}

}

}

}

*Form1.cs*

using System;

using System.IO;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Net;

using System.Net.Sockets;

namespace server

{

public partial class Form1 : Form

{

static Socket workSocket;

static bool bCont;

static int numberOfConnections = 0;

static string workPath = Directory.GetCurrentDirectory() + "\\";

static myLogger logger = new myLogger(Directory.GetCurrentDirectory() + "\\log.txt");

Socket listenSocket;

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

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

{

Close();

}

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

{

int bytesReceived, bytesSent;

byte[] buffer = new byte[2000];

byte[] message;

string filename, messageStr = "";

int port = 5000;

IPEndPoint endPoint = new IPEndPoint(IPAddress.Any, port);

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

//устанавливаем соединение

listenSocket.Bind(endPoint);

logger.logBind();

//устанавливаем сокет в режим прослушивания

listenSocket.Listen(10);

logger.logListen();

bCont = true;

Task testTask = Task.Factory.StartNew(() =>

{

while (bCont)

{

//создаем объект сокет для нового соединения

workSocket = listenSocket.Accept();

numberOfConnections++;

logger.logNewConnection();

try

{

//пока сокет на стороне клиента активен и отправляет что-то

while (!(workSocket.Poll(-1, SelectMode.SelectRead) && (workSocket.Available == 0)))

{

//получаем данные из связанного объекта сокет в указанный буфер

bytesReceived = workSocket.Receive(buffer);

logger.logReceive(bytesReceived);

filename = Encoding.UTF8.GetString(buffer).Trim('\0');

string tempPath = String.Concat(workPath, filename);

//содержательная часть: открытие файла с заданным именем, считывание из него и отправка данных

string tempStr;

try

{

using (StreamReader sr = new StreamReader(tempPath))

{

while ((tempStr = sr.ReadLine()) != null)

{

messageStr += String.Concat(tempStr, "\r\n");

}

}

}

catch(FileNotFoundException ex)

{

messageStr = "File not found";

}

message = Encoding.UTF8.GetBytes(messageStr);

bytesSent = workSocket.Send(message);

logger.logSend(bytesSent);

messageStr = "";

filename = "";

}

}

catch (SocketException ex)

{

logger.logDisconnect();

}

workSocket.Shutdown(SocketShutdown.Both);

workSocket.Close();

}

});

}

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

{

bCont = false;

listenSocket.Close();

logger.logCloseConnection(numberOfConnections);

numberOfConnections = 0;

}

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

{

Close();

}

}

}

***Клиент***

*Form1.cs*

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Net;

using System.Net.Sockets;

namespace client

{

public partial class Form1 : Form

{

Socket socket;

public Form1()

{

InitializeComponent();

}

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

{

Close();

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

}

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

{

string host = textBox1.Text;

int port;

if (host.Equals(""))

{

MessageBox.Show("Host is empty", "Error", MessageBoxButtons.OK);

return;

}

try

{

port = Int32.Parse(textBox2.Text);

}

catch(FormatException ex)

{

MessageBox.Show("Port is empty", "Error", MessageBoxButtons.OK);

return;

}

IPAddress[] IPs = Dns.GetHostAddresses(host);

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

try

{

socket.Connect(IPs[0], port);

}

catch (SocketException ex)

{

MessageBox.Show("Connection error", "Error", MessageBoxButtons.OK);

}

}

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

{

socket.Shutdown(SocketShutdown.Both);

socket.Close();

textBox1.Text = String.Empty;

textBox2.Text = String.Empty;

textBox3.Text = String.Empty;

textBox4.Text = String.Empty;

}

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

{

string filename = textBox3.Text;

if (filename.Equals(""))

{

MessageBox.Show("Filename is empty", "Error", MessageBoxButtons.OK);

return;

}

byte[] message = new byte[30];

message = Encoding.UTF8.GetBytes(filename);

byte[] bytes = new byte[2000];

try

{

int bytesSent = socket.Send(message);

int bytesReceived = socket.Receive(bytes);

textBox3.Text = String.Empty;

textBox4.Text = (filename + " contents:\r\n");

textBox4.Text += Encoding.UTF8.GetString(bytes).Trim('\0');

}

catch (SocketException ex)

{

MessageBox.Show("Socket error", "Error", MessageBoxButtons.OK);

}

}

}

}