

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА  
(САМАРСКИЙ УНИВЕРСИТЕТ)»

Факультет Информатики   
Кафедра Программных систем

**ОТЧЕТ**  
  
по лабораторному практикуму по дисциплине

«Компьютерные сети»

Лабораторная работа № 3

Вариант 3

Студент В.Д. Гижевская

Руководитель А. С. Пигусов

Самара 2020

**Задание**

Разработать клиентское приложение, позволяющее пользователю получать структуру каталогов FTP-сервера.

Адрес FTP-сервера, имя пользователя и пароль вводятся пользователем.

**Листинг программы**

**Program.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace FRp

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

**Form1.cs**

using System.Windows.Forms;

using System.Collections.Generic;

using System;

using System.IO;

using System.Net.Sockets;

using System.Text;

namespace FRp

{

public partial class Form1 : Form

{

FTP\_requests ftp = new FTP\_requests();

public Form1()

{

InitializeComponent();

}

private void btnGet\_Click(object sender, System.EventArgs e)

{

rtbStatus.Text = "";

treeView.Nodes.Clear();

var host = tbHost.Text;

FTP\_Client client = new FTP\_Client(host, tbLogin.Text, tbPassword.Text);

//MakeTree\_Lib(client);

MakeTree\_MY(client);

}

private void CreateCatalogue(DirectoryElement directory, TreeNode inputDir)

{

foreach (var subelement in directory.subdirectories)

{

inputDir.Nodes.Add(subelement.Name);

if (subelement.isFolder)

{

CreateCatalogue(subelement, inputDir.Nodes[inputDir.Nodes.Count - 1]);

}

}

}

//test

private void MakeTree\_Lib(FTP\_Client client)

{

DirectoryElement root = new DirectoryElement(client.Host, true);

List<Status> logs = FTP\_Helper.ProcessServer(client, ref root);

foreach (var log in logs)

{

rtbStatus.Text += log.ConnectionStatus + " " + log.Message + "\n";

}

if (logs[logs.Count - 1].ConnectionStatus != "ERROR")

{

treeView.Nodes.Add(root.Name);

CreateCatalogue(root, treeView.Nodes[0]);

treeView.ExpandAll();

}

}

private void MakeTree\_MY(FTP\_Client client)

{

DirectoryElement root = new DirectoryElement(client.Host, true);

//Создание сокета

Status connectionStatus = ftp.getConnectedSocket(client.Host, FTP\_requests.FTP\_PORT, out Socket socket);

printLog(connectionStatus);

if(connectionStatus.ConnectionStatus == "ERROR")

{

return;

}

//Login

//USER \_\_\_\_ PASS\_\_\_\_

var request = ftp.GetUserRequest(client.Login) + ftp.GetPassRequest(client.Password);

connectionStatus = ftp.GetResponse(socket,request);

printLog(connectionStatus);

if (connectionStatus.ConnectionStatus == "ERROR")

{

return;

}

GetCatalogue(socket,client.Host,ref root, false);

connectionStatus = ftp.QuitSection(socket);

printLog(connectionStatus);

socket.Shutdown(SocketShutdown.Send);

socket.Close();

//Вывод каталога

treeView.Nodes.Add(root.Name);

CreateCatalogue(root, treeView.Nodes[0]);

treeView.ExpandAll();

}

private void GetCatalogue(Socket socket ,string path, ref DirectoryElement root, bool isChildDir)

{

Status connectStatus;

if(isChildDir)

{

//Переход в директорию если дочерний

connectStatus = ftp.GetResponse(socket, ftp.GetCwdrequest(path));

printLog(connectStatus);

if (connectStatus.ConnectionStatus == "ERROR")

{

return;

}

}

//PASV - переход в пассивный режим,получение dataSocket(по нему получаем данные с сервака)

var connectionStatus = ftp.GetPassiveSocket(socket, out Socket dataSocket);

printLog(connectionStatus);

if (connectionStatus.ConnectionStatus == "ERROR")

{

return;

}

//LIST запрос на список каталога

byte[] req = Encoding.ASCII.GetBytes(ftp.GetListrequest());

var bytesSent = socket.Send(req);//отправка

var response = "";

byte[] resp = new byte[socket.ReceiveBufferSize];

int bytes = 0;

using (MemoryStream m = new MemoryStream())

{

while (dataSocket.Poll(1000000, SelectMode.SelectRead) &&

(bytes = dataSocket.Receive(resp, dataSocket.ReceiveBufferSize, SocketFlags.None)) > 0)

{

m.Write(resp, 0, bytes);

}//получение

response = Encoding.ASCII.GetString(m.ToArray());

}

rtbStatus.Text += response + "\n";

//разбиение строк о каждом файле/папке

//-rwxr-x--- 1 1227 1000 18 Nov 20 2015 .bash\_logout

//-rwxr - x-- - 1 1227 1000 193 Nov 20 2015.bash\_profile

string[] split\_response = response.Split(

new char[] { '\r', '\n' },

StringSplitOptions.RemoveEmptyEntries);

foreach (var dirData in split\_response)

{

var fileName = getFileName(dirData);

var newPath = path + "/" + fileName;

if (dirData.StartsWith("d"))

{

//Если директорий запускаем рекурсивно

//drwxr-xr-x 2 1227 1000 4096 Nov 22 2016 123

var subdir = new DirectoryElement(fileName, true);

GetCatalogue(socket,fileName, ref subdir, true);

root.addElem(subdir);

rtbStatus.Text += $"OK Директорий по адресу {newPath} обработан успешно\n";

}

/\*else

{

//если папка

//-rw-r--r-- 1 1227 1000 21 Nov 22 2016 info.php

root.addElem(new DirectoryElement(fileName, false));

rtbStatus.Text += $"OK Файл по адресу {newPath} обработан успешно\n";

}\*/

}

//Переход в родительский директорий

connectStatus = ftp.GetResponse(socket, ftp.GetCdupRequest());

printLog(connectionStatus);

dataSocket.Shutdown(SocketShutdown.Receive);

dataSocket.Close();

}

//-rwxr-x--- 1 1227 1000 193 Nov 20 2015 -->.bash\_profile<--

private string getFileName(string dirData)

{

//

string[] dirAttributes = dirData.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

return dirAttributes[dirAttributes.Length - 1];

}

//Вывод статуса в лог

private void printLog(Status status)

{

rtbStatus.Text += status.ConnectionStatus + " " + status.Message + "\n";

}

}

}

**FTP\_requests.cs**

using System.Net;

using System.IO;

using System.Net.Sockets;

using System.Text;

namespace FRp

{

class FTP\_requests

{

public const int FTP\_PORT = 21;

public FTP\_requests ()

{

}

//Requests

#region

public string GetPassRequest(string password)

{

return "PASS " + password + "\r\n";

}

public string GetUserRequest(string username)

{

return "USER " + username + "\r\n";

}

//Завершение сеанса

public string GetQuitRequest()

{

return "QUIT\r\n";

}

//Переход в пассивный режим

public string GetPasvRequest()

{

return "PASV\r\n";

}

//Переход в активный режим

public string GetPostRequest()

{

return "POST\r\n";

}

//Сменить директорию

public string GetCwdrequest(string curdir)

{

return $"CWD {curdir}\r\n";

}

//Переход в родительский каталог

public string GetCdupRequest()

{

return "CDUP\r\n";

}

public string GetListrequest()

{

return "LIST\r\n";

}

#endregion

public Status getConnectedSocket(string host, int port, out Socket socket)

{

socket = new Socket(

AddressFamily.InterNetwork,

SocketType.Stream,

ProtocolType.Tcp);

try

{

IPEndPoint ipPoint = new IPEndPoint(IPAddress.Parse(host), port);

socket.Connect(ipPoint);

}

catch (WebException ex)

{

return new Status("ERROR",ex.Message);

}

if (socket.Connected)

{

return new Status("OK",$"Соединение с {host} по порту {port}");

}

return new Status("ERROR","Not Connected");

}

public Status GetResponse(Socket socket,string request)

{

byte[] req = Encoding.ASCII.GetBytes(request);

var response = "";

byte[] resp = new byte[socket.ReceiveBufferSize];

try

{

var bytesSent = socket.Send(req);//отправка

int bytes = 0;

using (MemoryStream m = new MemoryStream())//получение

{

while (socket.Poll(1000000, SelectMode.SelectRead) &&

(bytes = socket.Receive(resp, socket.ReceiveBufferSize, SocketFlags.None)) > 0)

{

m.Write(resp, 0, bytes);

}

response = Encoding.ASCII.GetString(m.ToArray());

}

}

catch (WebException ex)

{

return new Status("ERROR", ex.Message);

}

return new Status("OK", response);

}

public Status GetPassiveSocket(Socket socket, out Socket dataSocket)

{

dataSocket = null;

Status connectStatus = GetResponse(socket,GetPasvRequest());

if (connectStatus.ConnectionStatus == "ERROR")

{

return connectStatus;

}

var response = connectStatus.Message;

// получаем <h1,h2,h3,h4,p1,p2>

var start = response.IndexOf('(') + 1;

var end = response.IndexOf(')') ;

response = response.Substring(start,end - start);

string[] IP\_Port = response.Split(',');

// ip adress xxx.xxx.xxx.xxx

var ipAddress = $"{IP\_Port[0]}.{IP\_Port[1]}.{IP\_Port[2]}.{IP\_Port[3]}";

//(p1 \* 256) + p2 = data port

int port = (int.Parse(IP\_Port[4]) \* 256) + int.Parse(IP\_Port[5]);

Status connectionStatus = getConnectedSocket(ipAddress, port, out dataSocket);

if (connectionStatus.ConnectionStatus != "OK")

{

return connectionStatus;

}

return connectStatus;

}

public Status QuitSection(Socket socket)

{

Status connectStatus = GetResponse(socket,GetQuitRequest());

return connectStatus;

}

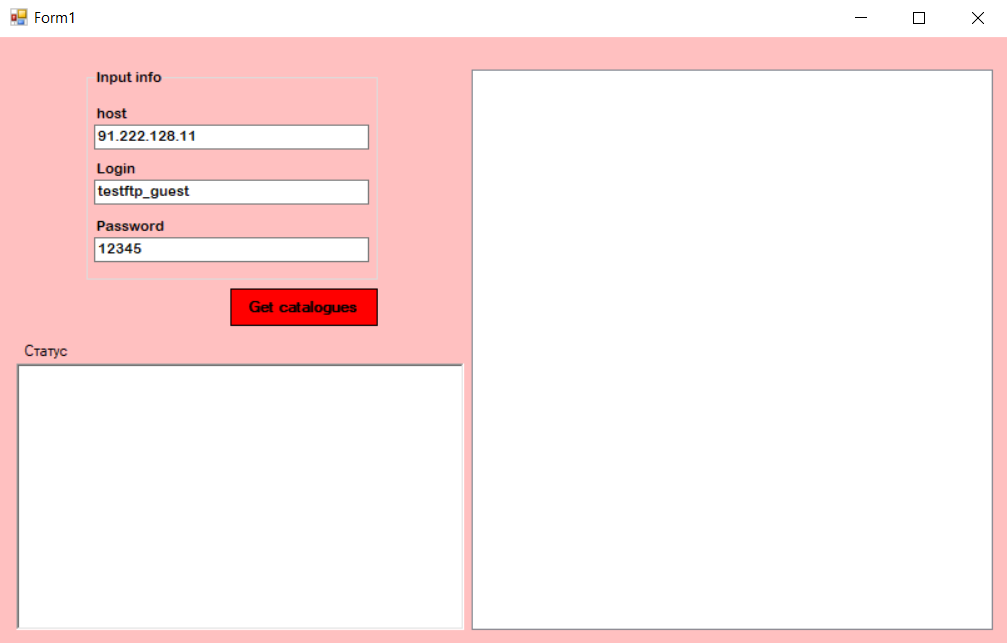
}

}

**Результаты работы программы**

Данная программа позволяет пользователю получать структуру каталогов FTP-сервера.

На рисунке 1 представлен интерфейс программы.

  
Рисунок 1 – Интерфейс программы

На рисунке 2 представлен результат работы программы. В качестве входных параметров были заданы:

* Адрес хоста: 91.222.128.11 (по умолчанию);
* Логин: testftp\_guest;
* Пароль: 12345;

Найденные каталоги показаны на рисунке 2.

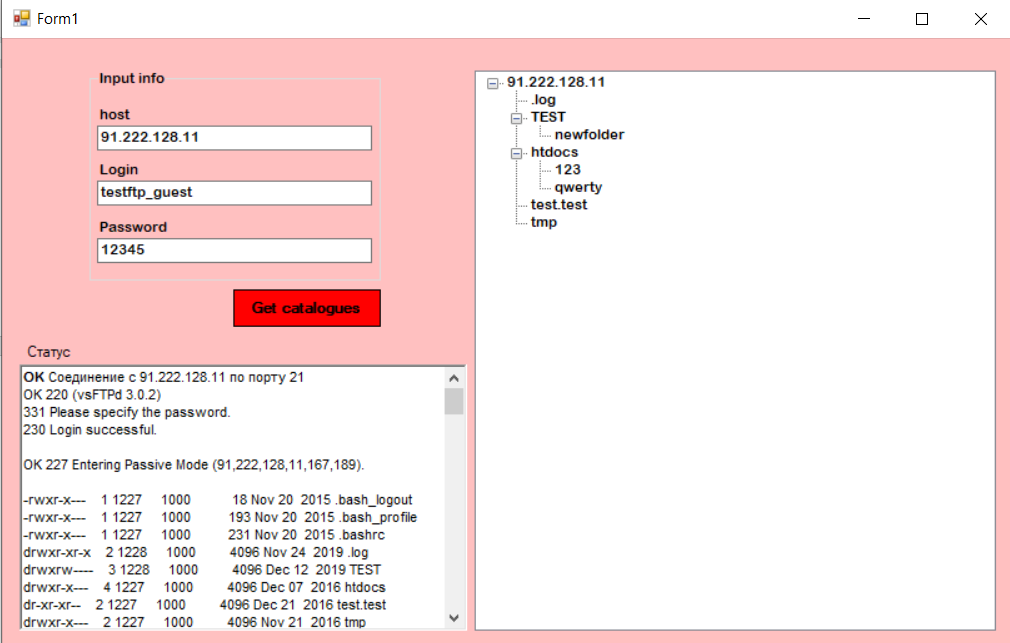


Рисунок 2 – Результат работы программы