МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И

ТЕЛЕКОММУНИКАЦИЙ

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

**Отчёт**

**по лабораторной работе №3**

**по дисциплине**

**«Сетевые технологии»**

Выполнил: студент гр. пм-51 Бакулин А.В. \_\_\_\_\_\_\_\_\_\_\_

Проверил: преподаватель Прозорова Т.Г. \_\_\_\_\_\_\_\_\_\_\_

Киров 2014

**Задание 1.**

Разработать два приложения клиент и сервер для передачи файла от клиента к серверу. Имя файла клиент должен принимать в качестве параметра. IP адрес и порт могут быть параметрами или задаваться в конфигурационном файле. Сервер должен поддерживать одновременную работу с несколькими клиентами. Обратить внимание на то, что файл с таким именем уже может существовать на сервере.

**Задание 2.**

Разработать приложение – транслятор портов. Трансляция осуществляется в соответствии с набором правил трансляции, заданных в конфигурационном файле. Каждое правило должно указывать, с какого порта на какие IP адрес и порт транслировать. При изменении конфигурационного файла новые правила должны вступать в действие, но установленные соединения не должны разрываться.

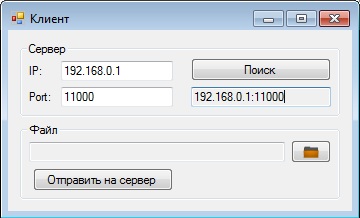
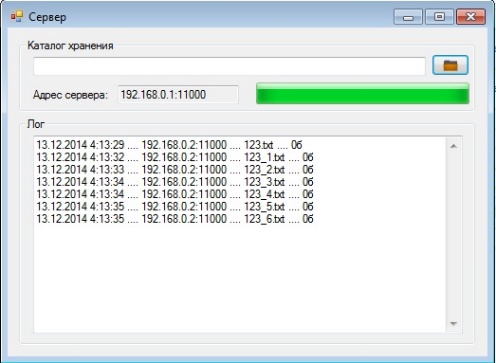
**Задание 3.**

Разработать приложение – почтовый клиент. Приложение должно позволять получать почту с почтового сервера с использованием протокола POP3 и отправлять почту с помощью протокола SMTP. Полученные сообщения должны храниться на клиенте до их удаления.

**Задание 4.**

Разработать два приложения: чат-сервер и чат-клиент. Клиент должен уметь получать сообщения с сервера и оставлять сообщения на сервере. Получение новых сообщений с сервера должно происходить в режиме реального времени. Сервер должен поддерживать работу с несколькими клиентами одновременно.

**Задание 1**

****

Сервер

namespace task31s

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private byte[] PingMsg;

private IPAddress myip;

private UdpClient udpclient, udpc;

private TcpListener tcpl;

private Thread listener, receiver;

private Stream stream;

private NetworkStream nstream;

private delegate void UpdateProgressCallback(Int64 BytesRead, Int64 TotalBytes);

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

{

myip = Dns.GetHostByName(Dns.GetHostName()).AddressList[1];

textBox2.Text = myip.ToString() + ":11000";

PingMsg = Encoding.ASCII.GetBytes(textBox2.Text);

udpclient = new UdpClient(100);

listener = new Thread(new ThreadStart(Listener));

listener.Start();

receiver = new Thread(new ThreadStart(Receiver));

receiver.Start();

}

private void Listener()

{

IPEndPoint point = new IPEndPoint(IPAddress.Any, 100);

Byte[] temp;

string recstr;

while (true)

{

try

{

temp = udpclient.Receive(ref point);

recstr = Encoding.ASCII.GetString(temp);

if (recstr == "Ping")

{

udpclient.Send(PingMsg, PingMsg.Length, point);

}

}

catch (Exception)

{

continue;

}

}

}

private string CheckName(string name)

{

bool checkname;

int addname;

string path;

if (textBox3.Text.Length > 0) path = textBox3.Text;

else path = Path.GetFullPath(@".\");

string[] files = Directory.GetFiles(path);

addname = 1;

checkname = false;

while (!checkname)

{

checkname = true;

foreach (string s in files)

{

if (new FileInfo(s).Name == name)

{

if (name.IndexOf('\_') < 0) name = Path.GetFileNameWithoutExtension(name) + "\_" + addname + Path.GetExtension(name);

else name = name.Substring(0, name.LastIndexOf('\_') + 1) + addname + Path.GetExtension(name);

addname++;

checkname = false;

break;

}

}

}

return name;

}

private void Receiver()

{

try

{

tcpl = new TcpListener(myip, 11000);

tcpl.Start();

udpc = new UdpClient(11000);

IPEndPoint endp = new IPEndPoint(IPAddress.Any, 11000);

TcpClient tcpk = tcpl.AcceptTcpClient();

nstream = tcpk.GetStream();

byte[] Buffer = new byte[2048];

Buffer = udpc.Receive(ref endp);

int bytesSize = Buffer.Length;

string fi = System.Text.Encoding.ASCII.GetString(Buffer, 0, bytesSize);

string FileName = CheckName(fi.Split(' ')[0]);

long FileSize = Convert.ToInt64(fi.Split(' ')[1]);

stream = new FileStream(FileName, FileMode.Create);

while ((bytesSize = nstream.Read(Buffer, 0, Buffer.Length)) > 0)

{

stream.Write(Buffer, 0, bytesSize);

this.Invoke(new UpdateProgressCallback(this.UpdateProgress), new object[] { stream.Length, FileSize });

}

textBox1.Text += DateTime.Now.ToString() + " .... " + endp.ToString() + " .... " + FileName + " .... " + (new FileInfo(FileName)).Length.ToString() + "б\r\n";

}

finally

{

if (nstream != null) nstream.Close();

if (stream != null) stream.Close();

udpc.Close();

tcpl.Stop();

Receiver();

}

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

System.Diagnostics.Process.GetCurrentProcess().Kill();

udpclient.Close();

if (nstream != null) nstream.Close();

if (stream != null) stream.Close();

receiver.Abort();

listener.Abort();

}

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

{

if (folderBrowserDialog1.ShowDialog() == DialogResult.OK) textBox3.Text = folderBrowserDialog1.SelectedPath;

}

private void UpdateProgress(Int64 BytesRead, Int64 TotalBytes)

{

if (TotalBytes > 0)

{

progressBar1.Value = Convert.ToInt32((BytesRead \* 100) / TotalBytes);

}

}

}

}

Клиент

namespace task31k

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

byte[] PingMsg;

IPAddress myip;

UdpClient udpclient;

Thread listener;

TcpClient client;

NetworkStream nstream;

FileStream fs;

Socket \_socket;

IPEndPoint \_point;

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

{

PingMsg = Encoding.ASCII.GetBytes("Ping");

myip = Dns.GetHostByName(Dns.GetHostName()).AddressList[1];

udpclient = new UdpClient(100);

}

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

{

\_socket = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);

\_socket.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.Broadcast, 1);

\_socket.Bind(new IPEndPoint(myip, 100));

\_point = new IPEndPoint(IPAddress.Broadcast, 100);

\_socket.SendTo(PingMsg, \_point);

\_socket.Close();

listener = new Thread(new ThreadStart(Listener));

listener.Start();

}

private void Listener()

{

IPEndPoint point = new IPEndPoint(IPAddress.Any, 100);

Byte[] temp;

bool check = true;

string s;

while (check)

{

try

{

temp = udpclient.Receive(ref point);

if ((s = Encoding.ASCII.GetString(temp)) != "Ping")

{

textBox4.Text = s;

check = false;

}

}

catch (Exception)

{

MessageBox.Show("123");

}

}

udpclient.Close();

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

udpclient.Close();

if (listener != null) listener.Abort();

}

private void textBox4\_DoubleClick(object sender, EventArgs e)

{

string[] s = textBox4.Text.Split(':');

textBox1.Text = s[0];

textBox2.Text = s[1];

}

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

{

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

textBox3.Text = openFileDialog1.FileName;

}

}

private void button3\_Click(object sender, EventArgs e)

{

if (textBox3.Text.Length == 0) return;

client = new TcpClient();

try

{

client.Connect(textBox1.Text, Convert.ToInt32(textBox2.Text));

}

catch (Exception)

{

MessageBox.Show(this, "Сервер недоступен");

return;

}

nstream = client.GetStream();

fs = new FileStream(textBox3.Text, FileMode.Open, FileAccess.Read);

FileInfo finfo = new FileInfo(textBox3.Text);

udpclient = new UdpClient(Convert.ToInt32(textBox2.Text));

\_point = new IPEndPoint(IPAddress.Parse(textBox1.Text), Convert.ToInt32(textBox2.Text));

string FileName = finfo.Name;

long FileSize = finfo.Length;

byte[] fi = new byte[2048];

fi = System.Text.Encoding.ASCII.GetBytes((FileName + " " + FileSize.ToString()).ToCharArray());

udpclient.Send(fi, fi.Length, \_point);

int bytesSize = 0;

byte[] Buffer = new byte[2048];

while ((bytesSize = fs.Read(Buffer, 0, Buffer.Length)) > 0)

{

nstream.Write(Buffer, 0, bytesSize);

}

udpclient.Close();

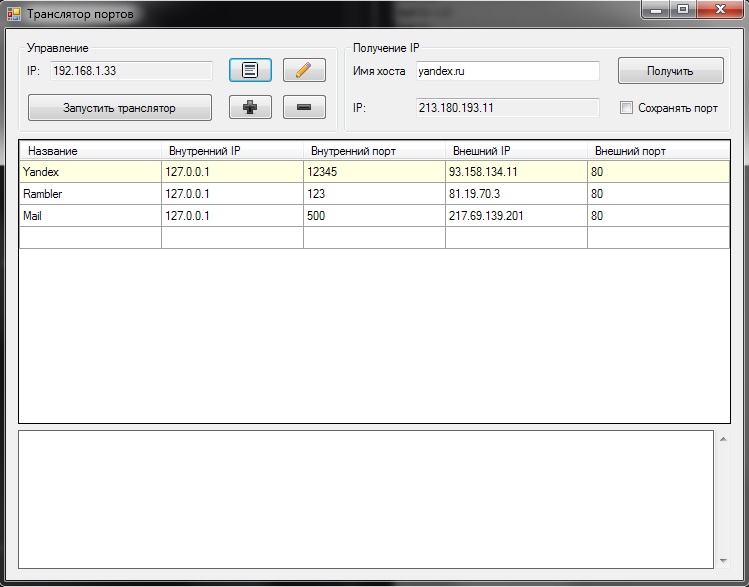
client.Close();

nstream.Close();

fs.Close();

}

}

**Задание 2**

namespace task32

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

public static Rule tRule;

IPAddress myip;

List<Rule> RulesList;

bool isRunnig;

Thread[] tlist;

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

{

myip = Dns.GetHostByName(Dns.GetHostName()).AddressList[0];

textBox1.Text = myip.ToString();

RulesList = new List<Rule>();

string path = Application.StartupPath + "\\task32.cfg";

try

{

LoadCfg(path);

}

catch (Exception)

{

MessageBox.Show(this, "Конфигурационный файл не найден");

}

isRunnig = false;

}

private void button3\_Click(object sender, EventArgs e) //получить имя хоста

{

IPHostEntry host = Dns.GetHostEntry(textBox3.Text);

textBox4.Text = host.AddressList[0].ToString();

}

private void button5\_Click(object sender, EventArgs e) //добавить правило

{

IPAddress ip = null;

int port = 0;

try

{

ip = IPAddress.Parse(textBox4.Text);

port = 80;

}

catch (Exception)

{

}

Rule r = new Rule(null, null, 0, ip, port, true);

Form2 add = new Form2(r);

add.ShowDialog();

if (tRule.success)

{

RulesList.Add(tRule);

dataGridView1.Rows.Add(new object [] { tRule.name, tRule.inip, tRule.inport, tRule.extip, tRule.extport });

}

}

private void button6\_Click(object sender, EventArgs e) //удалить правило

{

if (dataGridView1.CurrentRow == null || dataGridView1.CurrentRow.Index == dataGridView1.Rows.Count - 1) return;

DataGridViewRow row = dataGridView1.CurrentRow;

Rule r = new Rule((string)row.Cells[0].Value, (IPAddress)row.Cells[1].Value, (int)row.Cells[2].Value, (IPAddress)row.Cells[3].Value, (int)row.Cells[4].Value, true);

RulesList.Remove(r);

dataGridView1.Rows.Remove(row);

}

private void button4\_Click(object sender, EventArgs e) //редактировать правило

{

if (dataGridView1.CurrentRow == null || dataGridView1.CurrentRow.Index == dataGridView1.Rows.Count - 1) return;

DataGridViewRow row = dataGridView1.CurrentRow;

Rule r = new Rule((string)row.Cells[0].Value, (IPAddress)row.Cells[1].Value, (int)row.Cells[2].Value, (IPAddress)row.Cells[3].Value, (int)row.Cells[4].Value, true);

RulesList.Remove(r);

Form2 edit = new Form2(r);

edit.ShowDialog();

int i = dataGridView1.CurrentRow.Index;

dataGridView1.Rows[i].Cells[0].Value = tRule.name;

dataGridView1.Rows[i].Cells[1].Value = tRule.inip;

dataGridView1.Rows[i].Cells[2].Value = tRule.inport;

dataGridView1.Rows[i].Cells[3].Value = tRule.extip;

dataGridView1.Rows[i].Cells[4].Value = tRule.extport;

RulesList.Add(tRule);

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

FileStream fs = File.Open("task32.cfg", FileMode.Create, FileAccess.Write);

StreamWriter stream = new StreamWriter(fs);

try

{

foreach (Rule r in RulesList)

{

string s = r.name + " " + r.inip.ToString() + " " + r.inport.ToString() + " " + r.extip.ToString() + " " + r.extport.ToString();

stream.WriteLine(s);

}

}

finally

{

stream.Close();

fs.Close();

}

}

private void LoadCfg (string path)

{

using (StreamReader stream = new StreamReader(path))

{

dataGridView1.Rows.Clear();

while (stream.Peek() > -1)

{

string[] data;

data = stream.ReadLine().Split(' ');

tRule = new Rule(data[0], IPAddress.Parse(data[1]), Convert.ToInt32(data[2]), IPAddress.Parse(data[3]), Convert.ToInt32(data[4]), true);

RulesList.Add(tRule);

dataGridView1.Rows.Add(new object[] { tRule.name, tRule.inip, tRule.inport, tRule.extip, tRule.extport });

}

}

}

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

{

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

LoadCfg(openFileDialog1.FileName);

}

}

private void Log(string str)

{

if (textBox2.InvokeRequired) textBox2.Invoke(new Action<string>((s) => textBox2.Text += " - " + DateTime.Now.ToString() + " " + str + "\r\n"), str);

else textBox2.Text += " - " + DateTime.Now.ToString() + " " + str;

}

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

{

if (!isRunnig)

{

tlist = new Thread[RulesList.Count];

Log("Транслятор запущен\r\n");

isRunnig = true;

button1.Text = "Остановить транслятор";

for (int i = 0; i < RulesList.Count; i++)

{

tlist[i] = new Thread(Listening);

tlist[i].IsBackground = true;

tlist[i].Start(RulesList[i]);

}

}

else

{

Log("Транслятор остановлен\r\n");

isRunnig = false;

button1.Text = "Запустить транслятор";

for (int i = 0; i < RulesList.Count; i++) tlist[i].Abort();

}

}

private void Listening(object o)

{

Rule rule = (Rule)o;

TcpListener TCP = new TcpListener(rule.inip, rule.inport);

ThreadData td;

TCP.Start();

while (true)

{

if (TCP.Pending())

{

Thread t = new Thread(Execute);

t.IsBackground = true;

td = new ThreadData(TCP.AcceptSocket(), rule.extport);

t.Start(td);

}

}

TCP.Stop();

}

private void Execute(object o)

{

ThreadData td = (ThreadData)o;

if (td.socket.Connected)

{

byte[] httpRequest = Receive(td.socket);

Regex myReg = new Regex(@"Host: (((?<host>.+?):(?<port>\d+?))|(?<host>.+?))\s+", RegexOptions.Multiline | RegexOptions.IgnoreCase);

Match m = myReg.Match(System.Text.Encoding.ASCII.GetString(httpRequest));

string host = m.Groups["host"].Value;

int port = 0;

if (checkBox1.Checked)

{

if (!int.TryParse(m.Groups["port"].Value, out port)) { port = 80; }

}

else port = td.port;

IPHostEntry entry = Dns.GetHostEntry(host);

IPEndPoint point = new IPEndPoint(entry.AddressList[0], port);

/\* попытка жестко перенаправлять трафик \*/

/\*

Regex myReg = new Regex(@"Host: (((?<host>.+?):(?<port>\d+?))|(?<host>.+?))\s+", RegexOptions.Multiline | RegexOptions.IgnoreCase);

Match m = myReg.Match(System.Text.Encoding.ASCII.GetString(httpRequest));

string host = m.Groups["host"].Value;

int port = 80;

string before = System.Text.Encoding.ASCII.GetString(httpRequest);

string after = before.Replace(host, "81.19.70.3");

int cook;

if ((cook = after.IndexOf("Cookie")) > 0) after = after.Remove(cook) + "\r\n";

httpRequest = Encoding.ASCII.GetBytes(after);

IPEndPoint point = new IPEndPoint(IPAddress.Parse("81.19.70.3"), 80);

FileStream fs = File.Open("ttt.txt", FileMode.Append, FileAccess.Write);

StreamWriter stream = new StreamWriter(fs);

try

{

string s = System.Text.Encoding.ASCII.GetString(httpRequest);

stream.WriteLine(s);

}

finally

{

stream.Close();

fs.Close();

}\*/

using (Socket rerouting = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp))

{

try

{

rerouting.Connect(point);

}

catch (Exception)

{

return;

}

if (rerouting.Send(httpRequest, httpRequest.Length, SocketFlags.None) != httpRequest.Length)

{

Log("Ошибка при отправке данных\r\n");

}

else

{

Log("Отправлено данных: " + httpRequest.Length);

byte[] httpResponse = Receive(rerouting);

/\* FileStream fs1 = File.Open("qqq.txt", FileMode.Append, FileAccess.Write);

StreamWriter stream1 = new StreamWriter(fs1);

try

{

string s = System.Text.Encoding.ASCII.GetString(httpResponse);

stream1.WriteLine(s);

}

finally

{

stream1.Close();

fs1.Close();

}\*/

if (httpResponse != null && httpResponse.Length > 0)

{

td.socket.Send(httpResponse, httpResponse.Length, SocketFlags.None);

Log("Получено данных: " + httpResponse.Length);

}

}

}

td.socket.Close();

}

}

private static byte[] Receive(Socket socket)

{

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

int len = 0;

using (MemoryStream m = new MemoryStream())

{

try

{

while (socket.Poll(1000000, SelectMode.SelectRead) && (len = socket.Receive(b, socket.ReceiveBufferSize, SocketFlags.None)) > 0)

{

m.Write(b, 0, len);

}

return m.ToArray();

}

catch (Exception)

{

return null;

}

}

}

}

public struct Rule

{

public string name;

public IPAddress inip, extip;

public int inport, extport;

public bool success;

public Rule(string s, IPAddress ip1, int p1, IPAddress ip2, int p2, bool suc)

{

name = s;

inip = ip1;

inport = p1;

extip = ip2;

extport = p2;

success = suc;

}

}

public struct ThreadData

{

public Socket socket;

public int port;

public ThreadData (Socket s, int i)

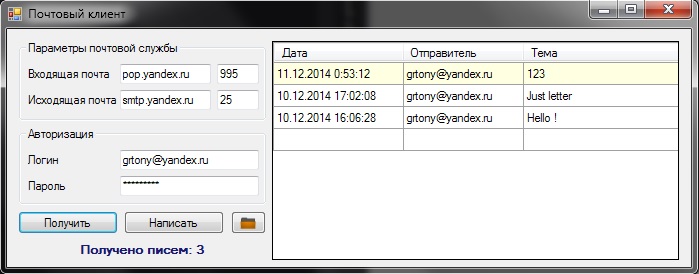
{

socket = s;

port = i;

}

}

**Задание 3**

namespace task33

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private List<Message> myLetters;

private void Log (string s)

{

label5.Text = s;

label5.Refresh();

}

private List<Message> GetMessages(string hostname, int port, string username, string password)

{

using (Pop3Client client = new Pop3Client())

{

client.Connect(hostname, port, true);

client.Authenticate(username, password);

int messageCount = client.GetMessageCount();

List<Message> allMessages = new List<Message>(messageCount);

for (int i = messageCount; i > 0; i--)

{

allMessages.Add(client.GetMessage(i));

}

client.Disconnect();

return allMessages;

}

}

private void LoadMessages()

{

myLetters.Clear();

}

private void SaveMessages()

{

Letters data = new Letters();

data.Data = myLetters;

/\*

FileStream stream = new FileStream(textBox6.Text, FileMode.Create);

IFormatter formatter = new BinaryFormatter();

formatter.Serialize(stream, data);

stream.Close();\*/

/\*

XmlSerializer ser = new XmlSerializer(data.GetType());

FileStream fs = new FileStream(textBox6.Text, FileMode.Open);

ser.Serialize(fs, data);

fs.Close();\*/

}

private void LoadTable()

{

dataGridView1.Rows.Clear();

foreach (Message M in myLetters)

{

dataGridView1.Rows.Add(new object[] { M.Headers.DateSent, M.Headers.ReturnPath, M.Headers.Subject });

}

}

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

{

if (textBox6.Text.Length == 0 || textBox5.Text.Length == 0 || textBox2.Text.Length == 0 || textBox1.Text.Length == 0)

{

MessageBox.Show(this, "Заполнены не все поля");

}

else

{

Log("Получение писем ...");

label5.Refresh();

myLetters = GetMessages(textBox1.Text, Convert.ToInt32(textBox2.Text), textBox6.Text, textBox5.Text);

Log("Получено писем: " + myLetters.Count);

LoadTable();

SaveMessages();

}

}

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

{

if (textBox6.Text.Length == 0 || textBox5.Text.Length == 0 || textBox4.Text.Length == 0 || textBox3.Text.Length == 0)

{

MessageBox.Show(this, "Заполнены не все поля");

}

else

{

Form3 sendform = new Form3(textBox6.Text, textBox5.Text, textBox4.Text, textBox3.Text, null);

sendform.ShowDialog();

}

}

private void button3\_Click(object sender, EventArgs e)

{

LoadMessages();

LoadTable();

}

public static bool isanswer;

private void dataGridView1\_DoubleClick(object sender, EventArgs e)

{

Form2 letter = new Form2(myLetters[dataGridView1.CurrentRow.Index]);

letter.ShowDialog();

if (isanswer)

{

string to = myLetters[dataGridView1.CurrentRow.Index].Headers.ReturnPath.ToString();

Form3 sendform = new Form3(textBox6.Text, textBox5.Text, textBox4.Text, textBox3.Text, to);

sendform.ShowDialog();

}

}

}

public class Letters

{

public Letters()

{

}

/\*

public void GetObjectData(SerializationInfo info, StreamingContext context)

{

info.AddValue("props", mL, typeof(List<Message>));

}

public Letters(SerializationInfo info, StreamingContext context)

{

mL = (List<Message>)info.GetValue("props", typeof(List<Message>));

}\*/

private List<Message> mL;

public List<Message> Data

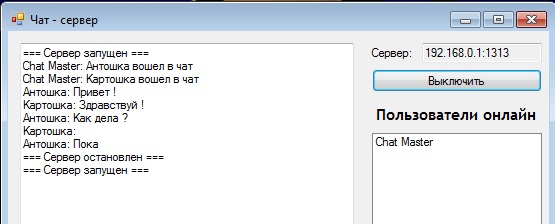
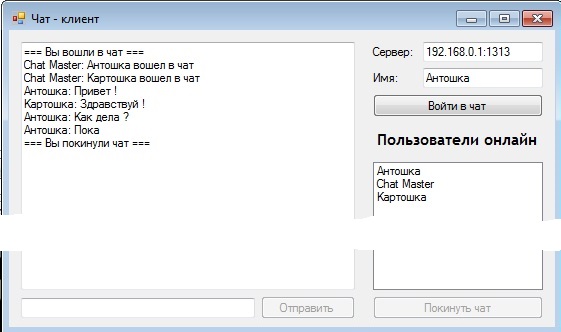
{

get { return mL; }

set { mL = value; }

}

}

**Задание 4**

Сервер

namespace task34s

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

Thread listener;

Thread talk;

IPAddress myip;

UdpClient udpclient;

private byte[] PingMsg;

Thread talkListener;

TcpListener TCP;

bool isRunning = false;

StreamReader srReceiver;

StreamWriter swSender;

public static Hashtable Users = new Hashtable(20);

public static Hashtable Connections = new Hashtable(20);

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

{

myip = Dns.GetHostByName(Dns.GetHostName()).AddressList[1];

textBox1.Text = myip.ToString() + ":1313";

PingMsg = Encoding.ASCII.GetBytes(textBox1.Text);

udpclient = new UdpClient(100);

listener = new Thread(new ThreadStart(Listener));

listener.Start();

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

listener.Abort();

System.Diagnostics.Process.GetCurrentProcess().Kill();

}

private void Log(string str)

{

if (textBox4.InvokeRequired) textBox4.Invoke(new Action<string>((s) => textBox4.Text += str + "\r\n"), str);

else textBox4.Text += str + "\r\n";

}

private void Listener()

{

IPEndPoint point = new IPEndPoint(IPAddress.Any, 100);

Byte[] temp;

string recstr;

while (true)

{

try

{

temp = udpclient.Receive(ref point);

recstr = Encoding.ASCII.GetString(temp);

if (recstr == "Ping")

{

udpclient.Send(PingMsg, PingMsg.Length, point);

}

}

catch (Exception)

{

continue;

}

}

}

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

{

if (!isRunning)

{

isRunning = true;

TCP = new TcpListener(1313);

TCP.Start();

talkListener = new Thread(Listening);

talkListener.Start();

textBox4.AppendText("=== Сервер запущен ===\r\n");

button1.Text = "Выключить";

listBox1.Items.Add("Chat Master");

}

else

{

TCP.Stop();

talkListener.Abort();

textBox4.AppendText("=== Сервер остановлен ===\r\n");

isRunning = false;

listBox1.Items.Clear();

button1.Text = "Включить";

}

}

private void Listening()

{

while (isRunning)

{

TcpClient tcpClient = TCP.AcceptTcpClient();

talk = new Thread(AcceptUser);

talk.Start(tcpClient);

}

}

private void AddUser(TcpClient tcpUser, string name)

{

Users.Add(name, tcpUser);

Connections.Add(tcpUser, name);

SendAdminMessage(name + " вошел в чат");

if (listBox1.Items.IndexOf(name) == -1) listBox1.Items.Add(name);

}

private void RemoveUser(TcpClient tcpUser)

{

if (Connections[tcpUser] != null)

{

SendAdminMessage(Connections[tcpUser] + " покинул чат");

listBox1.Items.Remove(Connections[tcpUser]);

Users.Remove(Connections[tcpUser]);

Connections.Remove(tcpUser);

}

}

private void SendAdminMessage(string Message)

{

StreamWriter Sender;

Log("Chat Master: " + Message);

TcpClient[] tcpClients = new TcpClient[Users.Count];

Users.Values.CopyTo(tcpClients, 0);

for (int i = 0; i < tcpClients.Length; i++)

{

try

{

if (Message.Trim() == "" || tcpClients[i] == null) continue;

Sender = new StreamWriter(tcpClients[i].GetStream());

Sender.WriteLine("Chat Master: " + Message);

Sender.Flush();

Sender = null;

}

catch

{

RemoveUser(tcpClients[i]);

}

}

}

private void SendMessage(string From, string Message)

{

StreamWriter Sender;

Log(From + ": " + Message);

TcpClient[] tcpClients = new TcpClient[Users.Count];

Users.Values.CopyTo(tcpClients, 0);

for (int i = 0; i < tcpClients.Length; i++)

{

try

{

if (Message.Trim() == "" || tcpClients[i] == null) continue;

Sender = new StreamWriter(tcpClients[i].GetStream());

Sender.WriteLine(From + ": " + Message);

Sender.Flush();

Sender = null;

}

catch

{

RemoveUser(tcpClients[i]);

}

}

}

private void AcceptUser(object o)

{

TcpClient tcpClient = (TcpClient)o;

srReceiver = new StreamReader(tcpClient.GetStream());

swSender = new StreamWriter(tcpClient.GetStream());

string newUser = srReceiver.ReadLine();

string str;

if (newUser == "") return;

if (Users.Contains(newUser))

{

swSender.WriteLine("0|== Такое имя занято ===\r\n");

swSender.Flush();

tcpClient.Close();

srReceiver.Close();

swSender.Close();

return;

}

else

{

swSender.WriteLine("1");

swSender.Flush();

AddUser(tcpClient, newUser);

}

try

{

while ((str = srReceiver.ReadLine()) != "")

{

if (str == null) RemoveUser(tcpClient);

else SendMessage(newUser, str);

}

}

catch

{

RemoveUser(tcpClient);

}

}

}

Клиент

namespace task34k

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

StreamWriter Sender;

StreamReader Receiver;

TcpClient tcpServer;

Thread talk;

IPAddress serverip;

IPAddress myip;

string myname;

bool isConnected;

UdpClient udpclient;

byte[] PingMsg;

Thread listener;

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

{

udpclient = new UdpClient(100);

PingMsg = Encoding.ASCII.GetBytes("Ping");

myip = Dns.GetHostByName(Dns.GetHostName()).AddressList[1];

myname = textBox2.Text;

Socket socket = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);

socket.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.Broadcast, 1);

socket.Bind(new IPEndPoint(myip, 100));

IPEndPoint point = new IPEndPoint(IPAddress.Broadcast, 100);

socket.SendTo(PingMsg, point);

socket.Close();

listener = new Thread(new ThreadStart(Listener));

listener.Start();

}

private void Listener()

{

IPEndPoint point = new IPEndPoint(IPAddress.Any, 100);

Byte[] temp;

bool check = true;

string s;

while (check)

{

try

{

temp = udpclient.Receive(ref point);

if ((s = Encoding.ASCII.GetString(temp)) != "Ping")

{

textBox1.Text = s;

check = false;

}

}

catch (Exception)

{

}

}

udpclient.Close();

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

listener.Abort();

udpclient.Close();

if (isConnected == true)

{

talk.Abort();

isConnected = false;

Sender.Close();

Receiver.Close();

tcpServer.Close();

//System.Diagnostics.Process.GetCurrentProcess().Kill();

}

}

private void Log(string str)

{

if (textBox4.InvokeRequired) textBox4.Invoke(new Action<string>((s) => textBox4.Text += str + "\r\n"), str);

else textBox4.Text += str + "\r\n";

}

private void CloseConnection(string s)

{

Log(s);

button1.Enabled = true;

button2.Enabled = false;

button3.Enabled = false;

isConnected = false;

talk.Abort();

Sender.Close();

Receiver.Close();

tcpServer.Close();

}

private void ReceiveMessages()

{

Receiver = new StreamReader(tcpServer.GetStream());

string ConResponse = Receiver.ReadLine();

if (ConResponse[0] == '1')

{

Log("=== Вы вошли в чат ===");

}

else

{

CloseConnection("=== Не удалось войти в чат ===\r\n");

return;

}

while (isConnected)

{

string rs = Receiver.ReadLine();

Log(rs);

string from = rs.Substring(0, rs.IndexOf(':'));

if (listBox1.Items.IndexOf(from) == -1) listBox1.Items.Add(from);

}

}

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

{

tcpServer = new TcpClient();

if (textBox1.Text.IndexOf(':') > 0)

{

string[] ss = textBox1.Text.Split(':');

serverip = IPAddress.Parse(ss[0]);

tcpServer.Connect(serverip, Convert.ToInt32(ss[1]));

}

else tcpServer.Connect(IPAddress.Parse(textBox1.Text), 1313);

isConnected = true;

myname = textBox2.Text;

listBox1.Items.Add(myname);

button1.Enabled = false;

button2.Enabled = true;

button3.Enabled = true;

Sender = new StreamWriter(tcpServer.GetStream());

Sender.WriteLine(myname);

Sender.Flush();

talk = new Thread(new ThreadStart(ReceiveMessages));

talk.Start();

}

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

{

CloseConnection("=== Вы покинули чат ===");

}

private void button3\_Click(object sender, EventArgs e)

{

if (textBox3.Text.Length >= 1)

{

Sender.WriteLine(textBox3.Text);

Sender.Flush();

textBox3.Text = "";

}

}

}

**Вывод**

В результате выполнения лабораторной работы исследованы и реализованы технологии передачи файлов по сети; приема и отправки почты по протоколам POP3 и SMTP; трансляции портов; обмена сообщениями. Созданные приложения имеют графический интерфейс и пригодны для повседневного использования.