using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Threading;

using System.Net.Sockets;

using System.Net;

using System.Collections;

namespace chat1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

StartPosition = FormStartPosition.CenterScreen;

//关闭对文本框的非线程操作检查

TextBox.CheckForIllegalCrossThreadCalls = false;

}

string RemoteEndPoint; //客户端的网络结点

Thread threadwatch = null;//负责监听客户端的线程

Socket socketwatch = null;//负责监听客户端的套接字

//创建一个和客户端通信的套接字

Dictionary<string, Socket> dic = new Dictionary<string, Socket> { }; //定义一个集合，存储客户端信息

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

{

this.button1.Enabled = false;

//定义一个套接字用于监听客户端发来的消息，包含三个参数（IP4寻址协议，流式连接，Tcp协议）

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

//服务端发送信息需要一个IP地址和端口号

IPAddress address = IPAddress.Parse(textBox1.Text.Trim());//获取文本框输入的IP地址

//将IP地址和端口号绑定到网络节点point上

IPEndPoint point = new IPEndPoint(address,int.Parse(textBox2.Text.Trim()));//获取文本框上输入的端口号

//此端口专门用来监听的

//监听绑定的网络节点

socketwatch.Bind(point);

//将套接字的监听队列长度限制为20

socketwatch.Listen(20);

//创建一个监听线程

threadwatch = new Thread(watchconnecting);

//将窗体线程设置为与后台同步，随着主线程结束而结束

threadwatch.IsBackground = true;

//启动线程

threadwatch.Start();

//启动线程后 textBox3文本框显示相应提示

textBox3.AppendText("开始监听客户端传来的信息!" + "\r\n");

}

void OnlineList\_Disp(string Info)

{

listBoxOnlineList.Items.Add(Info); //在线列表中显示连接的客户端套接字

}

//监听客户端发来的请求

private void watchconnecting()

{

Socket connection = null;

while (true) //持续不断监听客户端发来的请求

{

try

{

connection = socketwatch.Accept();

}

catch (Exception ex)

{

textBox3.AppendText(ex.Message); //提示套接字监听异常

break;

}

//获取客户端的IP和端口号

IPAddress clientIP = (connection.RemoteEndPoint as IPEndPoint).Address;

int clientPort = (connection.RemoteEndPoint as IPEndPoint).Port;

//让客户显示"连接成功的"的信息

string sendmsg = "连接服务端成功！\r\n" + "本地IP:" + clientIP + "，本地端口" + clientPort.ToString();

byte[] arrSendMsg = Encoding.UTF8.GetBytes(sendmsg);

connection.Send(arrSendMsg);

RemoteEndPoint = connection.RemoteEndPoint.ToString(); //客户端网络结点号

textBox3.AppendText("成功与" + RemoteEndPoint + "客户端建立连接！\t\n"); //显示与客户端连接情况

dic.Add(RemoteEndPoint, connection); //添加客户端信息

OnlineList\_Disp(RemoteEndPoint); //显示在线客户端

//IPEndPoint netpoint = new IPEndPoint(clientIP,clientPort);

IPEndPoint netpoint = connection.RemoteEndPoint as IPEndPoint;

//创建一个通信线程

ParameterizedThreadStart pts = new ParameterizedThreadStart(recv);

Thread thread = new Thread(pts);

thread.IsBackground = true;//设置为后台线程，随着主线程退出而退出

//启动线程

thread.Start(connection);

}

}

///

/// 接收客户端发来的信息

///

///客户端套接字对象

private void recv(object socketclientpara)

{

Socket socketServer = socketclientpara as Socket;

while (true)

{

//创建一个内存缓冲区 其大小为1024\*1024字节 即1M

byte[] arrServerRecMsg = new byte[1024 \* 1024];

//将接收到的信息存入到内存缓冲区,并返回其字节数组的长度

try

{

int length = socketServer.Receive(arrServerRecMsg);

//将机器接受到的字节数组转换为人可以读懂的字符串

string strSRecMsg = Encoding.UTF8.GetString(arrServerRecMsg, 0, length);

//将发送的字符串信息附加到文本框txtMsg上

textBox3.AppendText("客户端:" + socketServer.RemoteEndPoint+",time:" + GetCurrentTime() + "\r\n" + strSRecMsg + "\r\n\n");

}

catch (Exception ex)

{

textBox3.AppendText("客户端"+socketServer.RemoteEndPoint+"已经中断连接"+"\r\n"); //提示套接字监听异常

listBoxOnlineList.Items.Remove(socketServer.RemoteEndPoint.ToString());//从listbox中移除断开连接的客户端

socketServer.Close();//关闭之前accept出来的和客户端进行通信的套接字

break;

}

}

}

///

/// 获取当前系统时间的方法

///

/// 当前时间

private DateTime GetCurrentTime()

{

DateTime currentTime = new DateTime();

currentTime = DateTime.Now;

return currentTime;

}

//发送信息到客户端

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

{

string sendMsg = textBox4.Text.Trim(); //要发送的信息

byte[] bytes = System.Text.Encoding.UTF8.GetBytes(sendMsg); //将要发送的信息转化为字节数组，因为Socket发送数据时是以字节的形式发送的

if (listBoxOnlineList.SelectedIndex == -1)

{

MessageBox.Show("请选择要发送的客户端！", "提示", MessageBoxButtons.OK, MessageBoxIcon.Stop);

}

else

{

string selectClient = listBoxOnlineList.Text; //选择要发送的客户端

dic[selectClient].Send(bytes); //发送数据

textBox4.Clear();

textBox3.AppendText(label4.Text + GetCurrentTime() + "\r\n" + sendMsg + "\r\n");

}

}

//快捷键 Enter 发送信息

private void textBox4\_KeyDown(object sender, KeyEventArgs e)

{

//如果用户按下了Enter键

if (e.KeyCode == Keys.Enter)

{

string sendMsg = textBox4.Text.Trim(); //要发送的信息

byte[] bytes = System.Text.Encoding.UTF8.GetBytes(sendMsg);

if (listBoxOnlineList.SelectedIndex == -1)

{

MessageBox.Show("请选择要发送的客户端！", "提示", MessageBoxButtons.OK, MessageBoxIcon.Stop);

}

else

{

string selectClient = listBoxOnlineList.Text; //选择要发送的客户端

dic[selectClient].Send(bytes); //发送数据

textBox4.Clear();

textBox3.AppendText(label4.Text + GetCurrentTime() + "\r\n" + sendMsg + "\r\n");

}

}

}

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

{

}

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

{

DialogResult result = MessageBox.Show("是否退出？选否,最小化到托盘", "操作提示", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

this.Dispose();

}

else if (result == DialogResult.Cancel)

{

e.Cancel = true;

}

else

{

e.Cancel = true;

this.WindowState = FormWindowState.Minimized;

this.Visible = false;

this.notifyIcon1.Visible = true;

this.ShowInTaskbar = false;

}

}

private void notifyIcon1\_MouseDoubleClick(object sender, MouseEventArgs e)

{

base.Visible = true;

this.notifyIcon1.Visible = false;

this.ShowInTaskbar = true;

//base.Show();

base.WindowState = FormWindowState.Normal;

}

}

}

[复制代码](javascript:void(0);)

 以下是客户端的代码：

[复制代码](javascript:void(0);)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Threading;

using System.Net.Sockets;

using System.Net;

namespace chat2

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

StartPosition = FormStartPosition.CenterScreen;

//关闭对文本框的非法线程操作检查

TextBox.CheckForIllegalCrossThreadCalls = false;

}

//创建 1个客户端套接字 和1个负责监听服务端请求的线程

Thread threadclient = null;

Socket socketclient = null;

List<IPEndPoint> mlist =new List<IPEndPoint>();

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

{

//SocketException exception;

this.button1.Enabled = false;

//定义一个套接字监听

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

//获取文本框中的IP地址

IPAddress address = IPAddress.Parse(textBox1.Text.Trim());

//将获取的IP地址和端口号绑定在网络节点上

IPEndPoint point = new IPEndPoint(address,int.Parse(textBox2.Text.Trim()));

try

{

//客户端套接字连接到网络节点上，用的是Connect

socketclient.Connect(point);

}

catch (Exception )

{

//MessageBox.

MessageBox.Show("连接失败\r\n");

this.button1.Enabled = true;

return;

}

threadclient=new Thread(recv);

threadclient.IsBackground=true;

threadclient.Start();

}

// 接收服务端发来信息的方法

private void recv()//

{

int x = 0;

while (true)//持续监听服务端发来的消息

{

try

{

//定义一个1M的内存缓冲区，用于临时性存储接收到的消息

byte[] arrRecvmsg = new byte[1024 \* 1024];

//将客户端套接字接收到的数据存入内存缓冲区，并获取长度

int length = socketclient.Receive(arrRecvmsg);

//将套接字获取到的字符数组转换为人可以看懂的字符串

string strRevMsg = Encoding.UTF8.GetString(arrRecvmsg, 0, length);

if (x == 1)

{

textBox3.AppendText("服务器:" + GetCurrentTime() + "\r\n" + strRevMsg + "\r\n\n");

}

else

{

textBox3.AppendText(strRevMsg + "\r\n\n");

x = 1;

}

}

catch(Exception ex)

{

textBox3.AppendText("远程服务器已经中断连接"+"\r\n");

this.button1.Enabled = true;

break;

}

}

}

//获取当前系统时间

private DateTime GetCurrentTime()

{

DateTime currentTime = new DateTime();

currentTime = DateTime.Now;

return currentTime;

}

//发送字符信息到服务端的方法

private void ClientSendMsg(string sendMsg)

{

//将输入的内容字符串转换为机器可以识别的字节数组

byte[] arrClientSendMsg = Encoding.UTF8.GetBytes(sendMsg);

//调用客户端套接字发送字节数组

socketclient.Send(arrClientSendMsg);

//将发送的信息追加到聊天内容文本框中

textBox3.AppendText(this.label4.Text+": " + GetCurrentTime() + "\r\n" + sendMsg + "\r\n\n");

}

//点击按钮button2 向服务端发送信息

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

{

//调用ClientSendMsg方法 将文本框中输入的信息发送给服务端

ClientSendMsg(textBox4.Text.Trim());

textBox4.Clear();

}

private void textBox4\_KeyDown\_1(object sender, KeyEventArgs e)

{

//当光标位于文本框时 如果用户按下了键盘上的Enter键

if (e.KeyCode == Keys.Enter)

{

//则调用客户端向服务端发送信息的方法

ClientSendMsg(textBox4.Text.Trim());

textBox4.Clear();

}

}

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

{

}

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

{

DialogResult result = MessageBox.Show("是否退出？选否,最小化到托盘", "操作提示", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

this.Dispose();

}

else if (result == DialogResult.Cancel)

{

e.Cancel = true;

}

else

{

e.Cancel = true;

this.WindowState = FormWindowState.Minimized;

this.Visible = false;

this.notifyIcon1.Visible = true;

this.ShowInTaskbar = false;

}

}

private void notifyIcon1\_MouseDoubleClick(object sender, MouseEventArgs e)

{

base.Visible = true;

this.notifyIcon1.Visible = false;

this.ShowInTaskbar = true;

//base.Show();

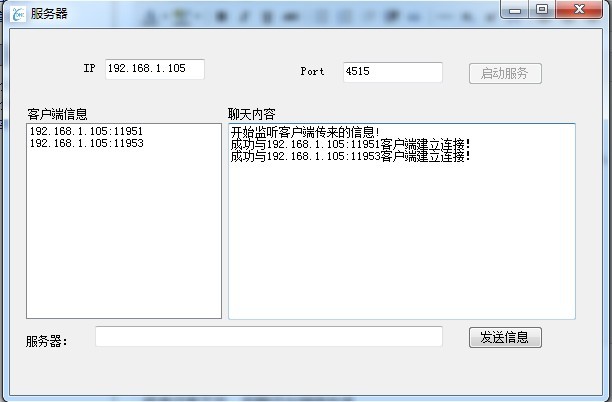
base.WindowState = FormWindowState.Normal;

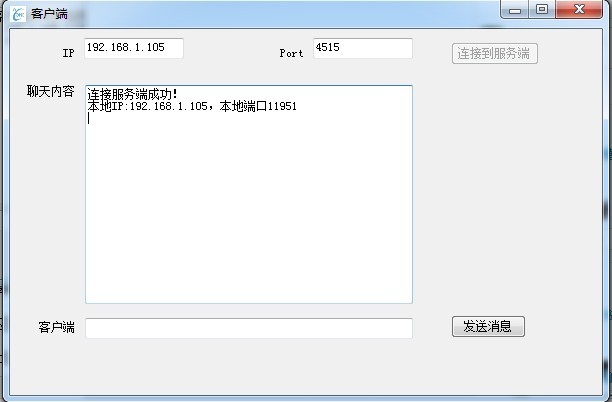
}

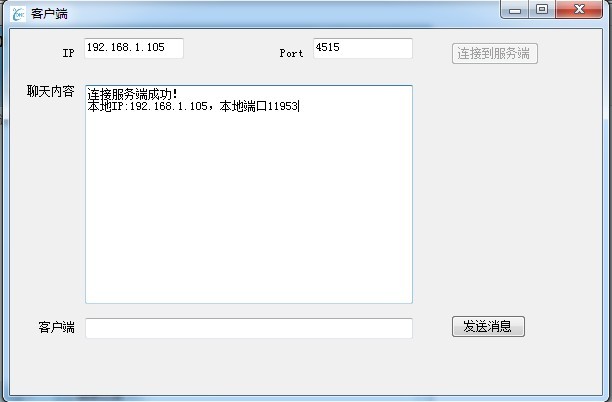
}

}

[复制代码](javascript:void(0);)







程序还有不足，后期还会继续改进

客户端在断开连接后，应该把socket关闭；

if (result == DialogResult.Yes)

 {  
           if (socketclient == null)  
            {  
                    this.Dispose();  
            }  
            else  
            {  
                socketclient.Close();  
                this.Dispose();  
             }

                      }