医学信息集成技术实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **学 号：** | **1519640532** | **姓 名：** | **戴维** | **实验成绩：** |  |
| **实验名称：** | **HL7通信程序设计** | | | **实验日期：** |  |

1. **实验目的：**

设计并编码HL7消息接收发送程序。

1. **实验内容：**
2. 使用HAPI TestPanel程序进行HL7消息的发送和接收
3. 启动Testpanel，左侧为控制区，右侧为配置与显示区。控制区从上到下分别为消息控制、发送连接(客户机)控制、接收连接(服务器)控制，单击控制区的任一条目，配置与显示区会联动改变
4. 单击CreateSamples按钮，会自动创建一条ADT-A04消息，并创建一个服务器(端口号随机)和客户机连接(host：localhost， 端口号：同服务器)。并默认选中一条消息在配置与显示区中显示
5. 服务器配置与启动：在接收连接控制区中选择一条连接，或者按+按钮新增一条服务器连接，在配置与显示区中修改其端口号，其他默认，然后单击start按钮开始侦听，等待客户机连接。
6. 客户机配置与启动：在发送连接控制区中选择一条客户机连接，或者按+按钮新增一条客户机连接，在配置与显示区中修改其对应服务器IP地址和端口号，其他默认，然后单击start按钮，连接服务器。
7. 消息发送：在消息控制区中选择一条消息，或者按+按钮新增一条消息，或打开文件按钮从文件中读取消息，在配置与显示窗口顶端send处选择客户机连接，单击send按钮进行发送，可以在对应客户机或服务器的Activity页中看到发送与应答。
8. 用Socket+多线程编写HL7通信程序，增加MLLP的协议处理，以及应答的发送与接收显示，利用Testpanel作为测试对象，正确实现HL7通信。

提示：MLLP格式：0x0b + HL7消息+ 0x1c + 0x0d

1. **程序及注释：**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading;

using System.Net;

using System.Net.Sockets;

namespace Client

{

public class Client

{

/// <summary>

/// 客户端ip地址

/// </summary>

private readonly IPAddress ip;

/// <summary>

/// 客户端端口

/// </summary>

private readonly int port;

/// <summary>

/// 客户端套接字

/// </summary>

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

/// <summary>

/// 接收服务端消息线程

/// </summary>

private readonly Thread receivingData = new Thread(new ParameterizedThreadStart(AcceptMgs));

public event EventHandler<MSGEventArgs> EventMsgReceived;

/// <summary>

/// 客户端

/// </summary>

/// <param name="ip">ip地址</param>

/// <param name="port">端口号</param>

public Client(IPAddress ip, int port)

{

//初始化ip与端口

this.ip = ip;

this.port = port;

}

/// <summary>

/// 开始连接

/// </summary>

public void Connect()

{

try

{

this.socket.Connect(ip, port);

receivingData.Start(this.socket);

}

catch (Exception e)

{

throw e;

}

}

/// <summary>

/// 断开连接

/// </summary>

public void Disconnect()

{

try

{

this.socket.Disconnect(true);

receivingData.Abort();

}

catch (Exception e) { throw e; }

}

/// <summary>

/// 发送消息

/// </summary>

/// <param name="buffer">要发送的字节流</param>

public void Send(byte[] buffer)

{

try

{

int a = buffer.Length;

byte[] b = new byte[a + 3];

b[0] = 0x0b;

for (int i = 1; i < b.Length - 2;i++)

b[i] = buffer[i - 1];

b[b.Length - 1] = 0x0d;

b[b.Length - 2] = 0x1c;

socket.Send(b);

}

catch (Exception e) { throw e; }

}

/// <summary>

/// 自动获取从server传回的消息，收到消息时触发“从服务端接收消息”事件

/// </summary>

private static void AcceptMgs(object \_socket)

{

try

{

while (true)

{

Socket socket = (Socket)\_socket;

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

int r = socket.Receive(buffer);

if (r == 0)

{

break;

}

//接收消息

string strMsg = Encoding.UTF8.GetString(buffer, 0, r);

}

}

catch (Exception e) { throw e; }

}

public class MSGEventArgs : EventArgs

{

public string msg; //HL7消息

public MSGEventArgs(string msg)

{

this.msg = msg;

}

}

}

}

namespace HL7

{

public partial class serverWindow : Form

{

Server serverEnd;

public serverWindow()

{

InitializeComponent();

Control.CheckForIllegalCrossThreadCalls = false;

}

private void Server\_button\_listen\_Click(object sender, EventArgs e)

{

try

{

serverEnd = new Server(IPAddress.Parse(server\_ip.Text), Convert.ToInt32(server\_port.Text));

serverEnd.EventMsgReceived += showMSG;

serverEnd.StartListen();

}

catch (Exception ex) { MessageBox.Show(ex.ToString()); }

}

private void showMSG(object sender, MSGEventArgs e)

{

richTextBox2.AppendText(e.msg);

}

}

public class Server

{

/// <summary>

/// 服务端ip地址

/// </summary>

private readonly IPAddress ip;

/// <summary>

/// 服务端端口

/// </summary>

private readonly int port;

public Dictionary<string, Socket> connections = new Dictionary<string, Socket>();

private readonly Socket socketListen = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

public event EventHandler<MSGEventArgs> EventMsgReceived;

public Server(IPAddress ip, int port)

{

this.ip = ip;

this.port = port;

}

public void StartListen()

{

try

{

IPEndPoint endPoint = new IPEndPoint(ip, port);

this.socketListen.Bind(endPoint);

this.socketListen.Listen(5);

Thread td = new Thread(AcceptMgs);

td.Start(this);

}

catch (Exception e) { throw e; }

}

public void StopListen()

{

try

{

this.socketListen.Close();

}

catch (Exception e) { throw e; }

}

private static void AcceptMgs(object o)

{

try

{

Server server = (Server)o;

while (true)

{

//新建线程循环接收客户端发来的信息

Thread td = new Thread(Recive);

td.IsBackground = true;

td.Start(server);

}

}

catch (Exception e) { throw e; }

}

/// <summary>

/// 接收客户端发来的数据，并显示出来

/// </summary>

private static void Recive(object o)

{

Server server = (Server)o;

Socket socketSend = server.socketListen.Accept();

try

{

while (true)

{

//客户端连接成功后，服务器应该接受客户端发来的消息

if (socketSend == null)

{

continue;

}

byte[] buffer = new byte[1024 \* 1024 \* 2];

//实际接受到的有效字节数

int r = socketSend.Receive(buffer);

//如果客户端关闭，发送的数据就为空，然后就跳出循环

if (r == 0)

{

break;

}

//接收客户端信息

string strMsg = Encoding.UTF8.GetString(buffer, 0, r);

MSGEventArgs e = new MSGEventArgs(strMsg);

server.EventMsgReceived(server,e);

byte[] acceptbuffer = Encoding.UTF8.GetBytes("服务器端接收成功");

//将字节数组传递给客户端

socketSend.Send(acceptbuffer);

}

}

catch (Exception e) { throw e; }

}

}

public class MSGEventArgs : EventArgs

{

public string msg; //HL7消息

public MSGEventArgs(string msg)

{

this.msg = msg;

}

}

}

namespace HL7

{

public partial class clientWindow : Form

{

Client.Client clientEnd;

public clientWindow()

{

InitializeComponent();

Control.CheckForIllegalCrossThreadCalls = false;

}

private void GroupBox1\_Enter(object sender, EventArgs e)

{

}

private void Client\_button\_connect\_Click(object sender, EventArgs e)

{

//创建新客户端socket连接

try

{

clientEnd = new Client.Client(IPAddress.Parse(client\_ip.Text),Convert.ToInt32(client\_port.Text));

clientEnd.EventMsgReceived += showMSG;

clientEnd.Connect();

}

catch(Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

private void showMSG(object sender, Client.Client.MSGEventArgs e)

{

richTextBox1.AppendText(e.msg);

}

private void Client\_button\_send\_Click(object sender, EventArgs e)

{

try

{

clientEnd.Send(Encoding.UTF8.GetBytes(client\_message.Text));

}

catch(Exception ex) { MessageBox.Show(ex.ToString()); }

}

private void Client\_button\_disconnect\_Click(object sender, EventArgs e)

{

try { clientEnd.Disconnect(); }

catch(Exception ex) { MessageBox.Show(ex.ToString()); }

}

private void Button1\_Click(object sender, EventArgs e)

{

serverWindow server = new serverWindow();

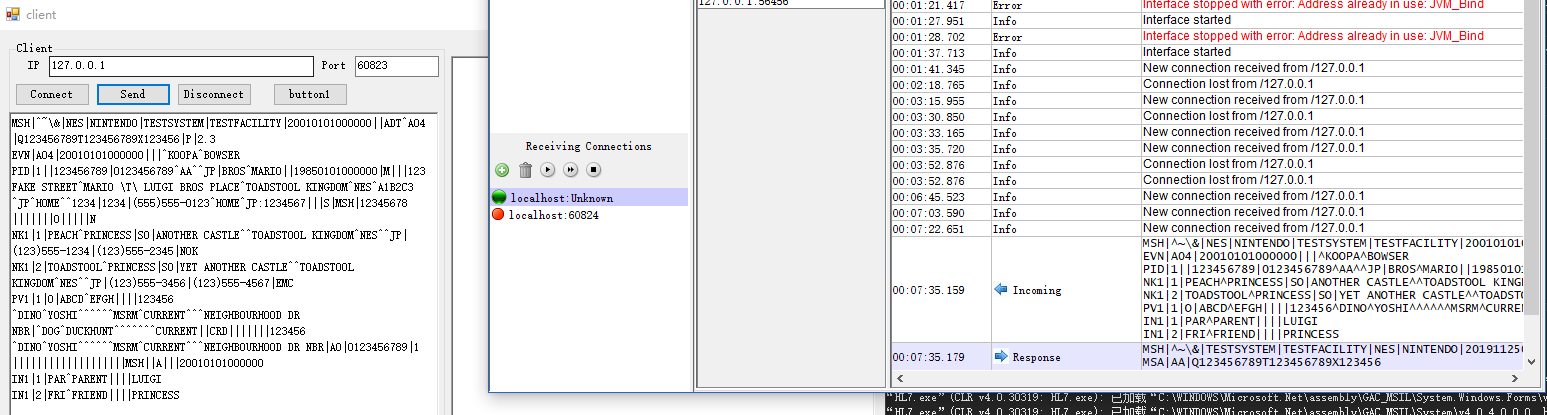
server.Show();

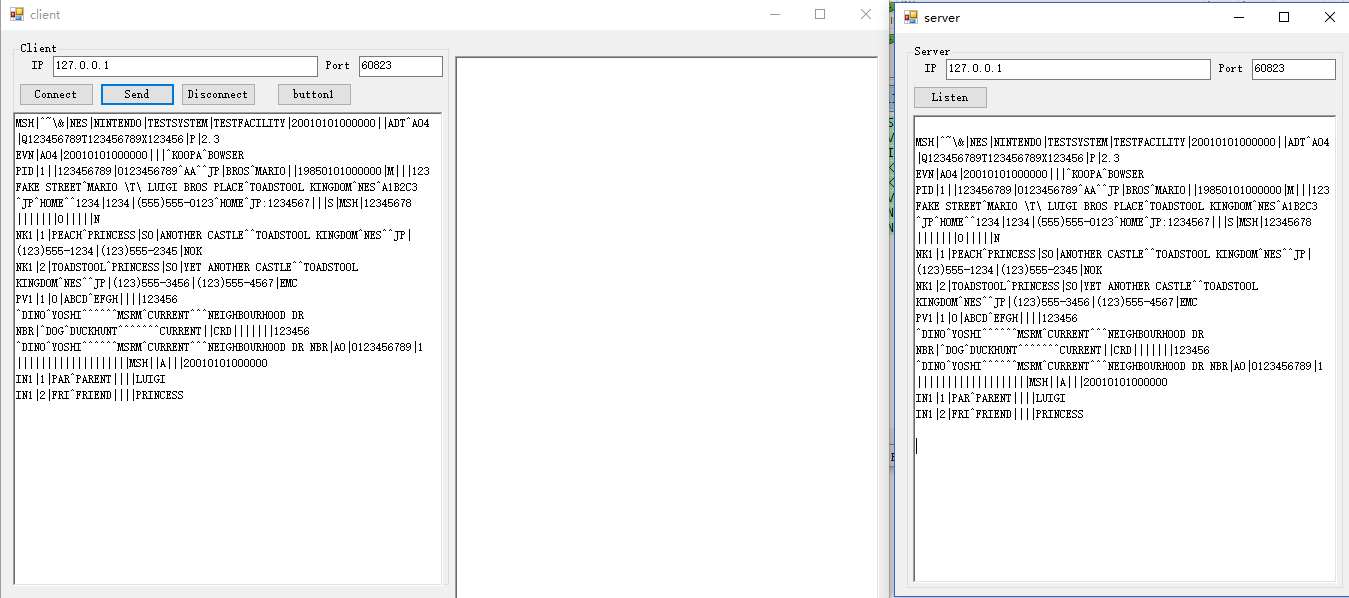
}

}

}

**四.运行结果**





**五.小结(不少于200字)**

本次实验复习了多线程、socket的编程，采用测试软件，用MLLP协议进行了简单的HL7消息传输测试。其中利用了事件的触发、多线程监听消息、MLLP处理消息边界等各种方法，对HL7进行了简单的文本处理。

本次实验中，对于事件和委托的部分处理尚不甚完善，窗体中跨线程调用委托修改控件内容（比如Append接收到的消息至文本框），在本次实验中仅采用开放跨线程的模式实现，在更严谨的情况下，应该进行更为详尽的处理。

另外，对于server和client类，应当更加规范编程格式，本次试验中编码较为零散，注释不全，应予以改善。