

# Лабораторная работа №4.

## Чат



ОГУ  
КАФЕДРА ВТИЗИ  
ГАЛИМОВ Р.Р.

# Асинхронный ввод-вывод

```
using System.Net;
using System.Net.Sockets;
using System.Windows.Forms;
namespace Sockets
{
    class CAsynchronousIoServer
    {
        private Socket _serversocket;
        private Form1 form1;
        int port;
        private void AddMess(string mess)
        {
            form1.Invoke(form1.myDelegate,mess);
        }
        public CAsynchronousIoServer(int _port,Form1 _form1)
        {
            port = _port;
            form1 = _form1;
        }
    }
}
```

# Асинхронный ввод-вывод

```
void SetupServer()
```

```
{
```

```
    IPHostEntry localMachineInfo =  
        Dns.GetHostEntry(Dns.GetHostName());  
    IPEndPoint myEndPoint = new  
        IPEndPoint(localMachineInfo.AddressList[3],  
        port);  
  
    serversocket = new  
        Socket(myEndPoint.Address.AddressFamily,  
        SocketType.Stream, ProtocolType.Tcp);  
  
    serversocket.Bind(myEndPoint);  
  
    serversocket.Listen((int)SocketOptionName.MaxCo  
nnections);  
}
```

# Асинхронный ввод-вывод

```
private class ConnectionInfo
{
    public Socket Socket;
    public byte[] Buffer;
}

private List<ConnectionInfo>
_connections = new
List<ConnectionInfo>();
```

# Асинхронный ввод-вывод

```
private class ConnectionInfo
{
    public Socket Socket;
    public byte[] Buffer;
}

private List<ConnectionInfo>
_connections = new
List<ConnectionInfo>();
```

# Асинхронный ввод-вывод

```
private class ConnectionInfo
{
    public Socket Socket;
    public byte[] Buffer;
}

private List<ConnectionInfo>
_connections = new
List<ConnectionInfo>();
public void Start()
{
    SetupServer();
    _serversocket.BeginAccept(new
 AsyncCallback(AcceptCallback),
_serversocket);
}
```

# Асинхронный ввод-вывод

```
private void AcceptCallback(IAsyncResult result)
{
    ConnectionInfo connection = new ConnectionInfo();
    try
    {
        // Завершение операции Accept
        Socket s = (Socket)result.AsyncState;
        connection.Socket = s.EndAccept(result);
        connection.Buffer = new byte[255];
        lock (_connections) _connections.Add(connection);
        // Начало операции Receive и новой операции Accept
        connection.Socket.BeginReceive(connection.Buffer,
            0, connection.Buffer.Length, SocketFlags.None,
            new AsyncCallback(ReceiveCallback),
            connection);
        _serversocket.BeginAccept(new AsyncCallback(
            AcceptCallback), result.AsyncState);
        AddMess("Соединение установлено");
    }
    catch (SocketException exc)
    {
        CloseConnection(connection);
        AddMess("Socket exception: " + exc.SocketErrorCode);
    }
    catch (Exception exc)
    {
        CloseConnection(connection);
        AddMess("Exception: " + exc);
    }
}
```

```
private void ReceiveCallback(IAsyncResult result)
{
    ConnectionInfo connection = (ConnectionInfo)result.AsyncState;
    try
    {
        int bytesRead = connection.Socket.EndReceive(result);
        AddMess(String.Format("Получено {0} байт", bytesRead));
        AddMess("Текст:" + Encoding.ASCII.GetString(connection.Buffer, 0, bytesRead));
        if (0 != bytesRead)
        {
            lock (_connections)
            {
                foreach (ConnectionInfo conn in _connections)
                {
                    if (connection != conn)
                        conn.Socket.Send(connection.Buffer, bytesRead,
SocketFlags.None);
                }
            }
            connection.Socket.BeginReceive( connection.Buffer, 0,
connection.Buffer.Length, SocketFlags.None,
new AsyncCallback(ReceiveCallback), connection);
        }
        else CloseConnection(connection);
    }
    catch (SocketException exc)
    {
        CloseConnection(connection);
        AddMess("Socket exception: " + exc.SocketErrorCode);
    }
    catch (Exception exc)
    {
        CloseConnection(connection);
        AddMess("Exception: " + exc);
    }
}
```

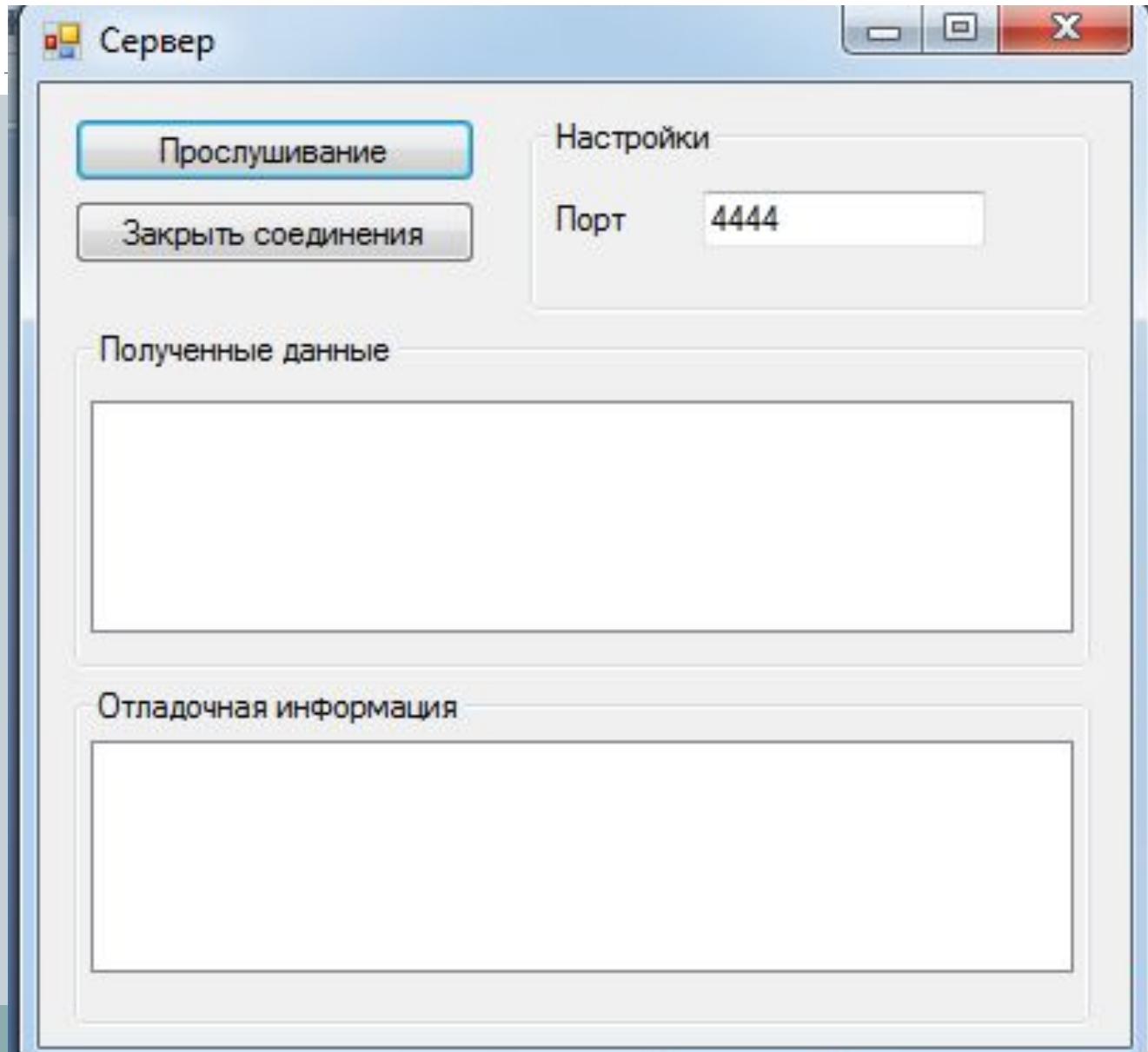
```
private void CloseConnection(ConnectionString  
ci)  
{  
    ci.Socket.Close();  
    lock (_connections)  
_connections.Remove(ci);  
}
```

```
namespace ConsoleApplication1 {  
    // Создадим делегат  
    delegate int IntOperation (int i, int j);  
    class Program { // Организуем ряд методов  
        static int Sum(int x, int y)  
        { return x + y; }  
        static int Prz(int x, int y)  
        { return x * y; }  
        static int Del(int x, int y)  
        { return x / y; }  
        static void Main() { // Сконструируем делегат  
            IntOperation op1 = new IntOperation(Sum);  
            int result = op1(5, 10);  
            Console.WriteLine("Сумма: " + result);  
            // Изменим ссылку на метод  
            op1 = new IntOperation(Prz);  
            result = op1(5, 10);  
            Console.WriteLine("Произведение: " + result);  
            Console.ReadLine();  
        }  
    }  
}
```

# Основная форма

```
public partial class Form1 : Form
{
    public delegate void AddListItem(string smess);
    public AddListItem myDelegate;
    private void AddDebugText(string smess)
    {
        listBox1.Items.Add(smess);
    }
    public Form1()
    {
        InitializeComponent();
        myDelegate = new AddListItem(AddDebugText);
    }
    private void button1_Click(object sender, EventArgs e)
    {
        CAsynchronousIoServer sercver = new
CAsynchronousIoServer(Convert.ToInt32(textBoxPort.Text),this);
        sercver.Start();
    }
}
```

# Основная форма



# Основная форма

с4. Telnet 192.168.1.4

124 до пожаловать в программу-клиент Microsoft Telnet

Символ переключения режима: ' ]'

Microsoft Telnet> open 192.168.1.3 4444

Подключение к 192.168.1.3...Не удалось открыть подключе  
4444: Сбой подключения

Microsoft Telnet> open 192.168.1.4 4444

Подключение к 192.168.1.4...

# Основная форма

```
public class StateObject
```

```
{
```

```
    // Client socket.
```

```
    public Socket workSocket = null;
```

```
    // Size of receive buffer.
```

```
    public const int BufferSize = 256;
```

```
    // Receive buffer.
```

```
    public byte[] buffer = new byte[BufferSize];
```

```
    // Received data string.
```

```
    public StringBuilder sb = new StringBuilder();
```

```
    public Form1 frm;
```

```
}
```

# Основная форма

```
public partial class Form1 : Form
```

```
{
```

```
    Socket client=null;
```

```
    public delegate void AddItem(string str);
```

```
    public AddItem addRes;
```

```
    public void AddResponse(string str)
```

```
{
```

```
        listBox1.Items.Add(str);
```

```
}
```

```
    public Form1()
```

```
{
```

```
        InitializeComponent();
```

```
        addRes = new AddItem(AddResponse);
```

```
}
```

```
private void button1_Click(object sender, EventArgs e)
{
    IPAddress ipAddress = IPAddress.Parse(textBox1.Text);
    IPEndPoint remoteEP = new IPEndPoint(ipAddress,
Convert.ToInt32(textBox2.Text));
    client = new Socket(AddressFamily.InterNetwork,
        SocketType.Stream, ProtocolType.Tcp);
    try
    {
        client.Connect(remoteEP);
    }
    catch(Exception ex)
    {
        MessageBox.Show("Не удалось подключиться к
серверу\n"+ex.Message);
        client = null;
    }
}
```

```
try
{
    // Create the state object.
    StateObject state = new StateObject();
    state.workSocket = client;
    state.frm = this;

    // Begin receiving the data from the remote device.
    client.BeginReceive(state.buffer, 0,
StateObject.BufferSize, 0,
        new AsyncCallback(ReceiveCallback), state);
}
catch (Exception ex)
{
    MessageBox.Show("Ошибка определения
обработчика получения данных\n" + ex.Message);
}
}
```

```
private void button2_Click(object sender, EventArgs  
e)
```

```
{
```

```
    if (client == null)
```

```
        return;
```

```
    byte[] byteData =
```

```
Encoding.Unicode.GetBytes(textBox3.Text);
```

```
    client.Send(byteData);
```

```
}
```

```
private static void ReceiveCallback(IAsyncResult ar)
{
    try
    {
        StateObject state = (StateObject)ar.AsyncState;
        Socket client = state.workSocket;
        int bytesRead = client.EndReceive(ar);
        if (bytesRead > 0)
        {
            string response = Encoding.Unicode.GetString(state.buffer, 0,
bytesRead);
            state.frm.Invoke(state.frm.addRes, response);
            client.BeginReceive(state.buffer, 0, StateObject.BufferSize, 0,
                new AsyncCallback(ReceiveCallback), state);
        }
    }
    catch (Exception e)
    {
        MessageBox.Show("Error\n" + e);
    }
}
```

# Литература

1. Дэрин Кили. Winsock.

<https://msdn.microsoft.com/ru-ru/library/dd335942.aspx>