After you have seen your grades; have recuperated well from 3rd term; are ready to move forward; and have some time available during the break, it will be good to take a preliminary look at some of the technologies that we will be focussing on during the next term such as C#, .NET, [UWP / Classic Desktop / WPF] applications, SQLServer, Oracle, ASP.NET Web Forms, ASP.NET MVC, Entity Framework, Signal R and Angular.

If you are keen on mastering at least the basic syntax of C# during the break, the quickest way to do so would be to implement in C# something that you have implemented recently using competing languages such as java and C/C++. So (for one last time) try implementing Multithreaded Directory server and the companion console based client application but this time using C#. Developing this app in C# would require the knowledge of mostly the basic syntax. I will cover more uncommon/advanced C# syntax/concepts such as Events and Delegates, and other support for asynchronous programming when the semester starts.

Please find enclosed VisualStudioConfiguration.doc that contains instructions on installing the needed project templates. Some links to Microsoft docs describing the C# language are also given.

Given below is the reference code for a simple TCPClient, and TCPServer. If you are able to compile them without any problem then start TCPServer on either Visual Studio Developer or regular “Command Window” and start TCP Client on another window. If they are able to run and connect correctly, you will see the client printing out “THIS IS A TEST”. Basically server receives “This is a Test” from the client and returns it after capitalizing it.

After successfully completing the above step, integrate the reference code of DirReader given below into your TCPServer code and alter the TCPClient appropriately to transform this application into a very simple directory server and client.

Finally, integrate Thread class into the Directory Server to make it multithreaded. A TwoThreadTest example similar to the ones in java and C/C++ is also given below.

//TCPServer

//========

using System;

using System.IO;

using System.Net;

using System.Net.Sockets;

using System.Text;

class MyTcpListener

{

static void Main(string[] args)

{

try

{

int port = 13000;

IPAddress address = IPAddress.Parse("127.0.0.1");

// IPAddress address = IPAddress.Parse("142.232.246.23");

IPEndPoint ipe = new IPEndPoint(address, port);

Socket s = new Socket(ipe.AddressFamily, SocketType.Stream, ProtocolType.Tcp);

s.Bind(ipe);

s.Listen(10);

Socket cls = s.Accept();

Byte[] byteSent = new Byte[256];

Byte[] bytesReceived = new Byte[256];

int bytes = cls.Receive(bytesReceived, bytesReceived.Length, 0);

Console.WriteLine(Encoding.ASCII.GetString(bytesReceived, 0, bytes));

String data = System.Text.Encoding.ASCII.GetString(bytesReceived, 0, bytes);

//Process the data sent by the client.

data = data.ToUpper();

byte[] msg = System.Text.Encoding.ASCII.GetBytes(data);

//Semd request to the server

cls.Send(msg, msg.Length, 0);

cls.Close();

s.Close();

}

catch (SocketException e)

{

Console.WriteLine("Socket exception: {0}", e);

}

Console.Read();

}

}

//TCPClient

//=========

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Net;

using System.Net.Sockets;

namespace TCPClient

{

class Program

{

static void Main(string[] args)

{

try

{

int port = 13000;

IPAddress address = IPAddress.Parse("127.0.0.1");

// IPAddress address = IPAddress.Parse("142.232.246.22");

IPEndPoint ipe = new IPEndPoint(address, port);

Socket s = new Socket(ipe.AddressFamily, SocketType.Stream, ProtocolType.Tcp);

s.Connect(ipe);

if (s.Connected)

{

string request = "This is a test";

Byte[] bytesSent = Encoding.ASCII.GetBytes(request);

Byte[] bytesReceived = new Byte[256];

s.Send(bytesSent, bytesSent.Length, 0);

int bytes = 0;

bytes = s.Receive(bytesReceived, bytesReceived.Length, 0);

Console.WriteLine(Encoding.ASCII.GetString(bytesReceived, 0, bytes));

s.Close();

}

}

catch (ArgumentNullException e)

{

Console.WriteLine("Argument Null Exception: {0}", e);

}

catch (SocketException e)

{

Console.WriteLine("SocketException: {0}", e);

}

Console.Read();

}

}

}

//DirRead

//========

using System;

using System.IO;

using System.Net;

using System.Net.Sockets;

using System.Text;

using System.Threading;

class DirReader

{

static void Main(string[] args) {

DirectoryInfo di = new DirectoryInfo("C:");

FileInfo[] fiArr = di.GetFiles();

string files = "";

foreach (FileInfo fri in fiArr)

{

Console.WriteLine(fri.Name);

}

}

}

//TwoThreadsTest

//==============

using System;

using System.Threading;

public class Countries

{

public void fiji()

{

for (int i = 0; i < 10; i++)

{

Thread.Sleep(10);

Console.WriteLine("Fiji");

}

}

public void jamaica()

{

for (int i = 0; i < 10; i++)

{

Thread.Sleep(10);

Console.WriteLine("Jamaica");

}

}

public static int Main()

{

Countries countries = new Countries();

Thread thread = new Thread(new ThreadStart(countries.fiji));

thread.Start();

thread = new Thread(new ThreadStart(countries.jamaica));

thread.Start();

Console.ReadLine();

return 0;

}

}