

ParameterizedThreadStart delegate

using System;

using System.Threading;

namespace Threads1

{

    class Program

    {

        static void printme(object n)

        {

            int num = 0;

            if(int.TryParse(n.ToString(),out num))

            {

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

                {

                    Console.WriteLine(i);

                }

            }

        }

        static void Main(string[] args)

        {

            Console.WriteLine("Plaese enter the number ");

            object num = Console.ReadLine();

            ParameterizedThreadStart parmthrd = new ParameterizedThreadStart(Program.printme);

            Thread t1 = new Thread(parmthrd);

            t1.Start(num);

        }

    }

}

TYPESAFE THREADSTART WITH PARAM

using System;

using System.Collections.Generic;

using System.Text;

namespace Threads1

{

    class Myclass1

    {

        int target;

        public Myclass1(int num)

        {

            target = num;

        }

        public void printnum()

        {

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

            {

                Console.WriteLine(i);

            }

        }

    }

}

using System;

using System.Collections.Generic;

using System.Text;

using System.Threading;

namespace Threads1

{

    class Class1

    {

        static void Main(string[] args)

        {

            Console.WriteLine("---Plaese enter the number ");

            int num = int.Parse(Console.ReadLine());

            Myclass1 ob = new Myclass1(num);

            Thread t1 = new Thread(new ThreadStart(ob.printnum));

            //            Thread t1 = new Thread((ob.printnum));

            t1.Start();

        }

    }

}

RETRIEVE DATA FROM THE THREAD

using System;

using System.Collections.Generic;

using System.Text;

using System.Threading;

namespace Threads1

{

    public delegate void factorialcallback(int fact);

    class retrieve

    {

        public static void printfact(int res)

        {

            Console.WriteLine("Factorial of the number = "+res);

        }

        static void Main(string[] args)

        {

            Console.WriteLine("Plaese enter the number ");

            int num = int.Parse(Console.ReadLine());

            factorialcallback callback = new factorialcallback(printfact);

            Factorial ob = new Factorial(num, callback);

            Thread t1 = new Thread(new ThreadStart(ob.getfactorial));

            t1.Start();

        }

    }

    class Factorial

    {

        int num;

        //factcal will be used to call the delegaate function

        factorialcallback factcal;

        public Factorial(int n, factorialcallback callbackfun )

        {

            num = n;

            factcal = callbackfun;

        }

        public void getfactorial()

        {

            int res=1;

            for (int i = 1; i < num; i++)

            {

                res = res \* i;

            }

            if (factcal != null)

                factcal(res);

        }

    }

}

JOIN

using System;

using System.Collections.Generic;

using System.Text;

using System.Threading;

namespace Threads1

{

    class Join\_multi

    {

        static void Thread1fun()

        {

            Console.WriteLine("Thread1 started ");

        }

        static void Thread2fun()

        {

            Console.WriteLine("Thread2 started ");

        }

        static void Main(string[] args)

        {

            Console.WriteLine("main thread started");

            Thread t1 = new Thread(Join\_multi.Thread1fun);

            t1.Start();

            Thread t2 = new Thread(Join\_multi.Thread2fun);

            t2.Start();

            //to force thread 1 to complete and then make the other thread to join

            t1.Join();

            Console.WriteLine("Thread 1 completed");

            t2.Join();

            Console.WriteLine("Thread 2 completed");

            Console.WriteLine("Main thread completed");

        }

    }

}

using System;

using System.Collections.Generic;

using System.Text;

using System.Threading;

namespace Threads1

{

    class Join\_multi

    {

        static void Thread1fun()

        {

            Console.WriteLine("Thread1 started ");

            Thread.Sleep(5000);

            Console.WriteLine("function is about to be retrieved");

        }

        static void Thread2fun()

        {

            Console.WriteLine("Thread2 started ");

        }

        static void Main(string[] args)

        {

            Console.WriteLine("main thread started");

            Thread t1 = new Thread(Join\_multi.Thread1fun);

            t1.Start();

            Thread t2 = new Thread(Join\_multi.Thread2fun);

            t2.Start();

            //to force thread 1 to complete and then make the other thread to join

            t1.Join();

            Console.WriteLine("Thread 1 completed");

            t2.Join();

            Console.WriteLine("Thread 2 completed");

            Console.WriteLine("Main thread completed");

        }

    }

}

using System;

using System.Collections.Generic;

using System.Text;

using System.Threading;

namespace Threads1

{

    class Join\_multi

    {

        static void Thread1fun()

        {

            Console.WriteLine("Thread1 started ");

            Thread.Sleep(5000);

            Console.WriteLine("function is about to be retrieved");

        }

        static void Thread2fun()

        {

            Console.WriteLine("Thread2 started ");

        }

        static void Main(string[] args)

        {

            Console.WriteLine("main thread started");

            Thread t1 = new Thread(Join\_multi.Thread1fun);

            t1.Start();

            Thread t2 = new Thread(Join\_multi.Thread2fun);

            t2.Start();

            //to force thread 1 to complete and then make the other thread to join

            if(t1.Join(1000))

                Console.WriteLine("Thread 1 completed in 1 second ");

            else

                Console.WriteLine("Thread function has not been completed in 1 sec ");

            t2.Join();

            Console.WriteLine("Thread 2 completed");

            if(t1.IsAlive)

                Console.WriteLine("thread1 is still alive");

            else

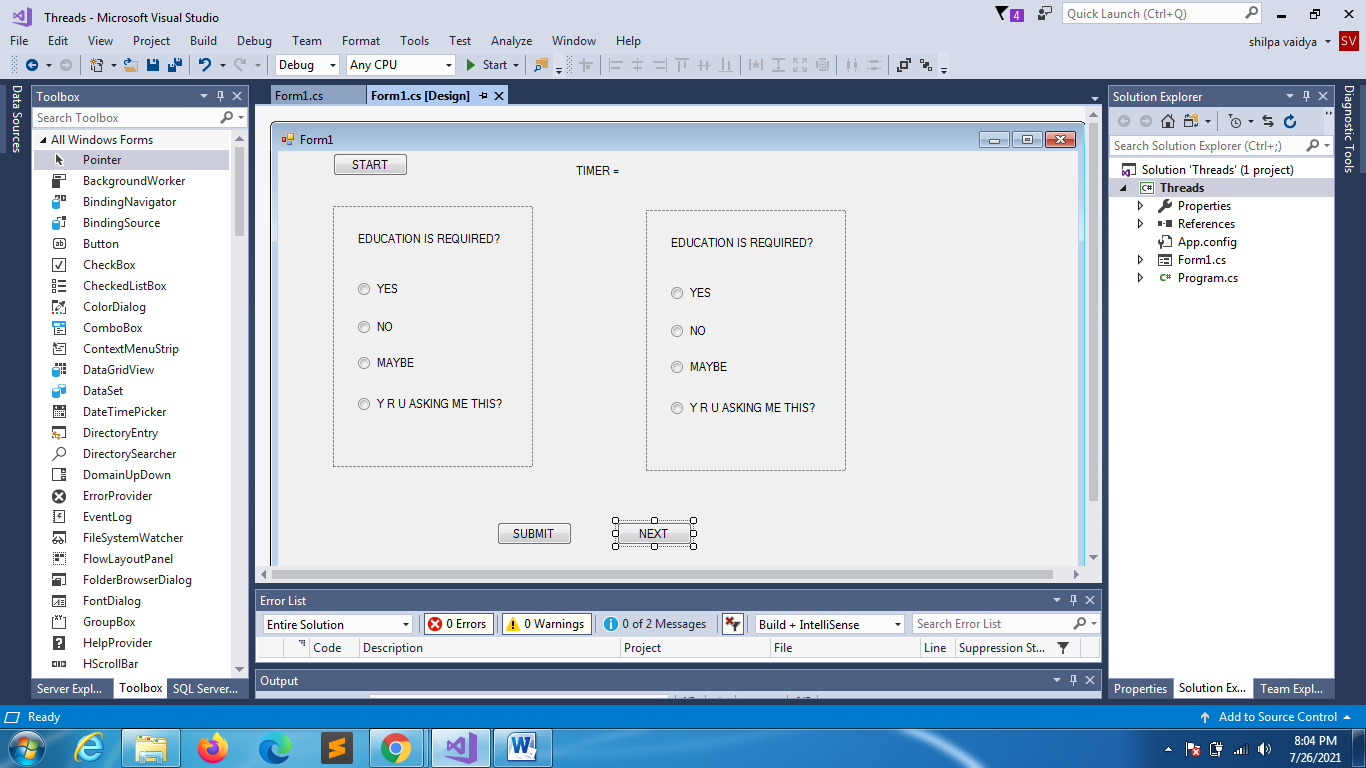
                Console.WriteLine("thread1 is not alive");

            Console.WriteLine("Main thread completed");

        }

    }

}



using System;

using System.Threading;

using System.Windows.Forms;

namespace Threads

{

    public partial class Form1 : Form

    {

        public int i;

        System.Threading.Timer threadtimer;

        public Form1()

        {

            InitializeComponent();

            i = 0;

        }

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

        {

            panel1.Visible = false;

            panel2.Visible = false;

        }

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

        {

        }

        private void waitForASec(object args)

        {

                label3.Text = i.ToString();

                i++;

            if (i == 5)

            {

                threadtimer.Dispose();

                BTNSUB.Visible = false;

            }

        }

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

        {

            panel1.Visible = true;

            threadtimer = new System.Threading.Timer(waitForASec,0,1,1000);

        }

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

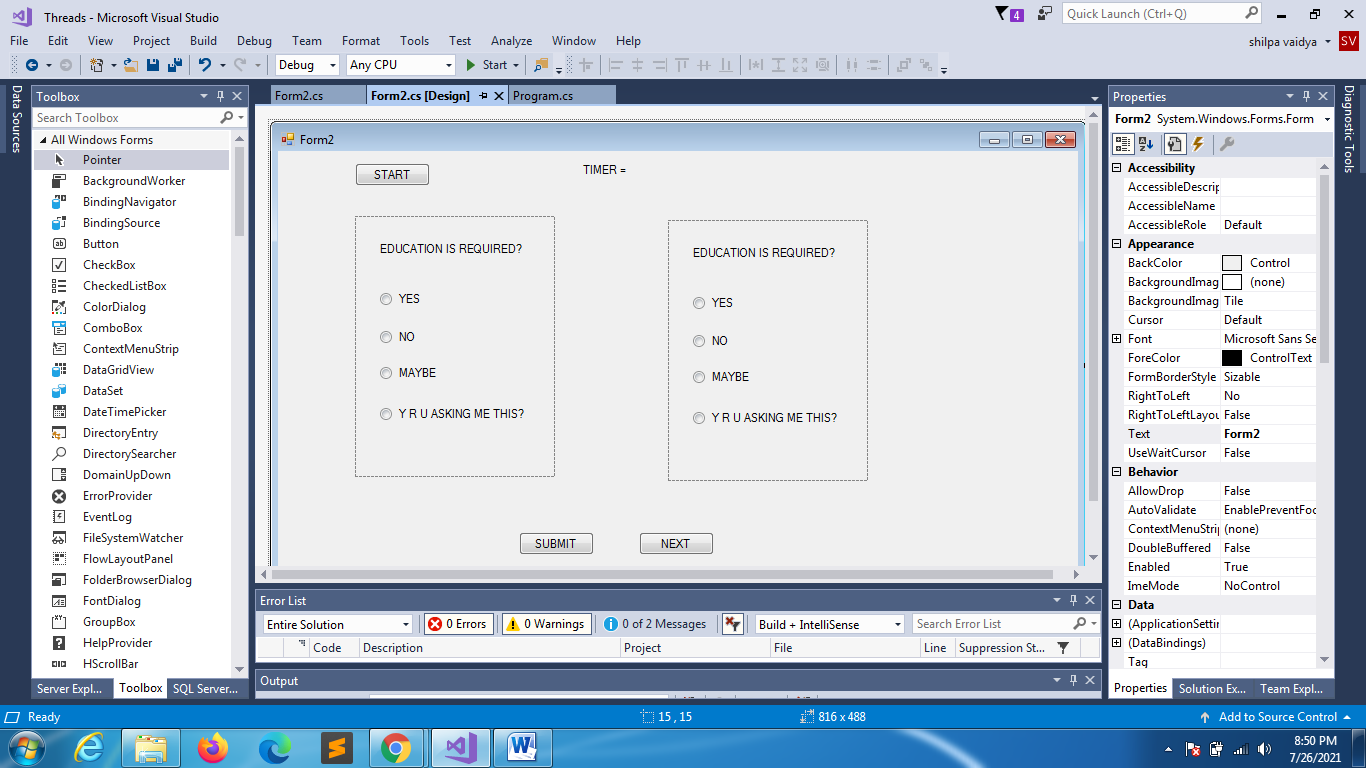
        {

            panel2.Visible = true;

        }

    }

}



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Threads

{

    public partial class Form2 : Form

    {

        int i = 0;

        public Thread thread;

        public Form2()

        {

            InitializeComponent();

        }

        private void Form2\_Load(object sender, EventArgs e)

        {

            panel1.Visible = false;

            panel2.Visible = false;

        }

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

        {

            BTNNEXT.Enabled = false;

            thread = new Thread(waitForASec);

            thread.Start();

        }

        private void waitForASec()

        {

            //panel1.Visible = true;

            panel1.Invoke((MethodInvoker)delegate { panel1.Visible = true; });

            for (;i<=5;i++)

            {

                lbltimer.Invoke((MethodInvoker)(() => lbltimer.Text = i.ToString()));

                Thread.Sleep(1000);

            }

            BTNSUB.Invoke((MethodInvoker)delegate { BTNSUB.Enabled = false; });

            BTNNEXT.Invoke((MethodInvoker)delegate { BTNNEXT.Enabled = true; });

            panel1.Invoke((MethodInvoker)delegate { panel1.Visible = false; });

        }

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

        {

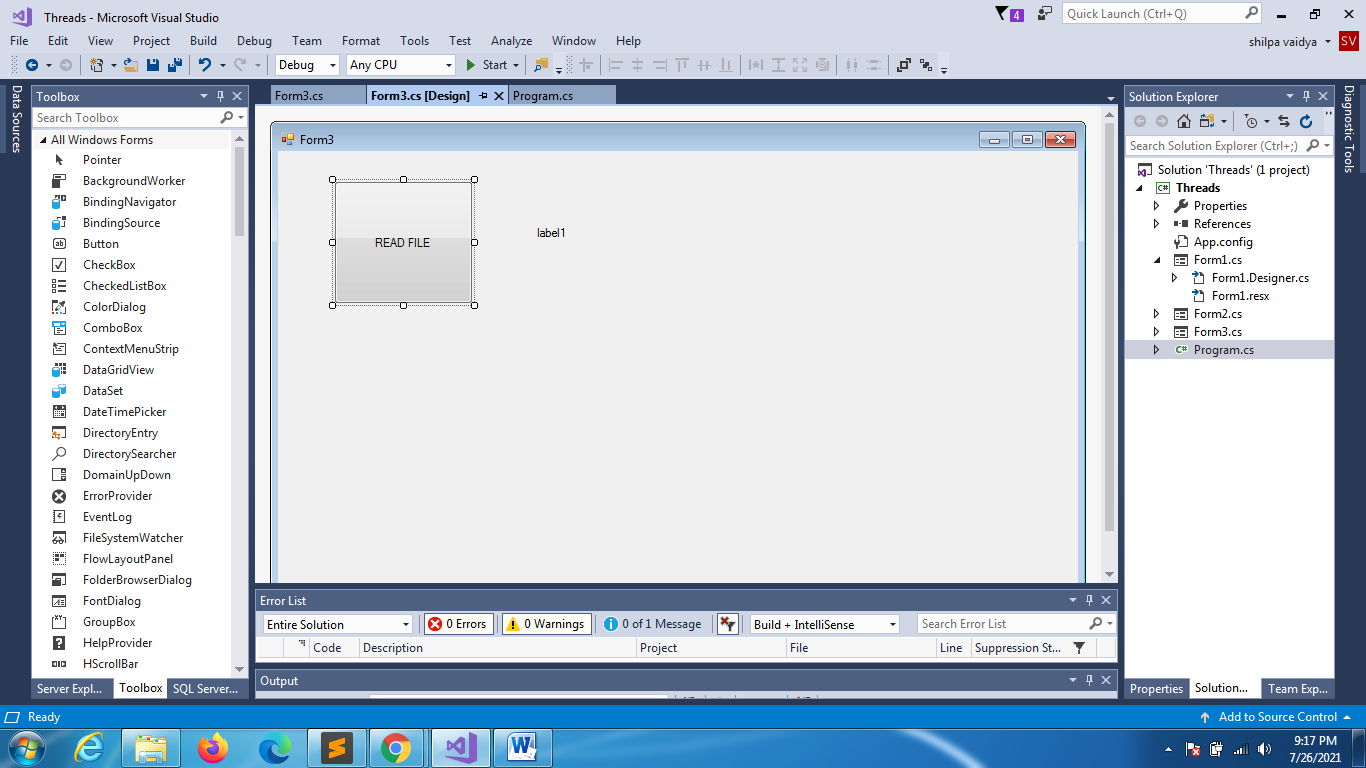
            panel2.Visible = true;

        }

    }

}

JOIN



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Threads

{

    public partial class Form3 : Form

    {

        public Form3()

        {

            InitializeComponent();

        }

        private int Countchar()

        {

            int count = 0;

            string file = @"C:\Users\User\Desktop\myfile.txt";

            using (StreamReader reader = new StreamReader(file))

            {

                string filecont = reader.ReadToEnd();

                count = filecont.Length;

                Thread.Sleep(5000);

            }

            return count;

        }

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

        {

            int count = 0;

            Thread thread = new Thread(() => { count = Countchar(); });

            thread.Start();

            lblcountchar.Text = "Processing the files please wait ";

            //we should make our main thread wait till we finish the reading of file

            thread.Join();

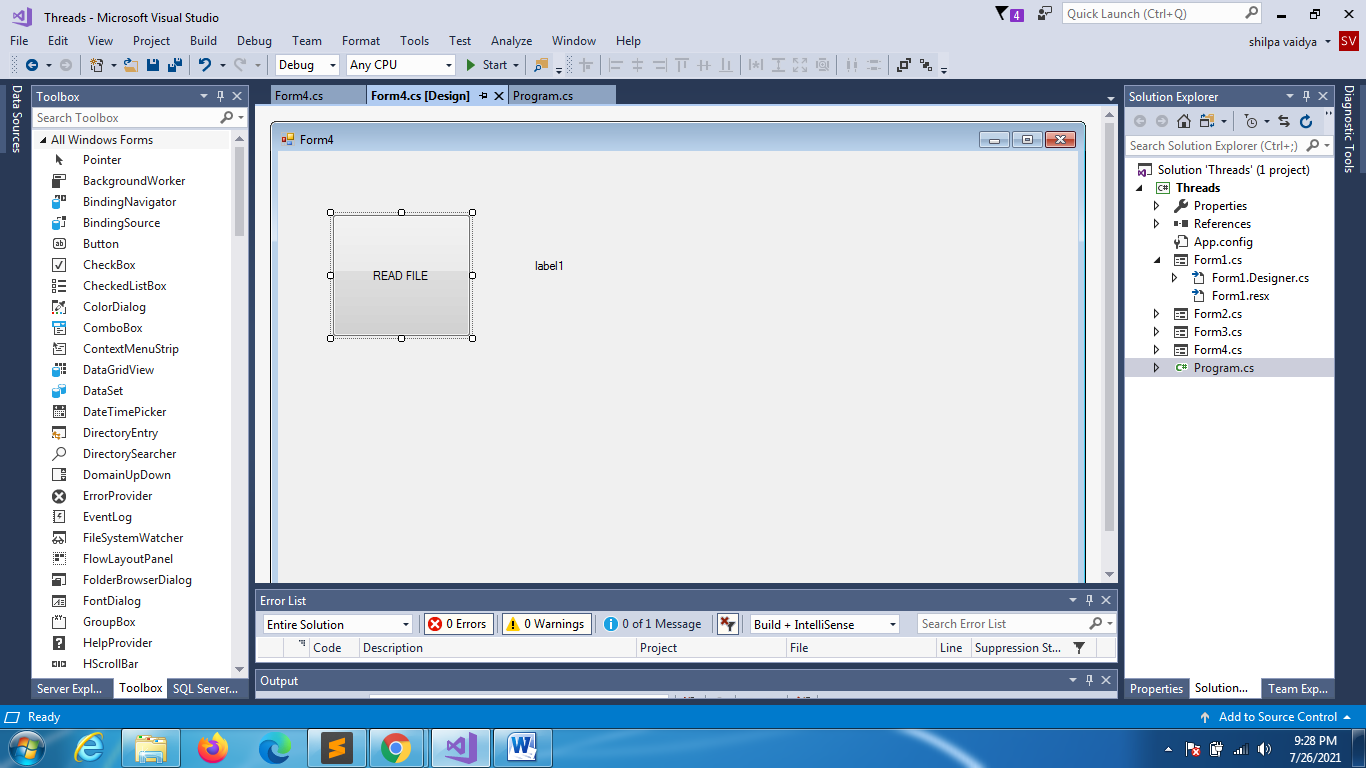
            lblcountchar.Text = "the no of chars in file = " + count.ToString();

        }

    }

}

Asynchronising equvivalent



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Threads

{

    public partial class Form4 : Form

    {

        int charcnt;

        public Form4()

        {

            InitializeComponent();

        }

        private int Countchar()

        {

            int count = 0;

            string file = @"C:\Users\User\Desktop\myfile.txt";

            using (StreamReader reader = new StreamReader(file))

            {

                string filecont = reader.ReadToEnd();

                count = filecont.Length;

                Thread.Sleep(5000);

            }

            return count;

        }

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

        {

            Thread thread = new Thread(() =>

                {

                    charcnt = Countchar();

                    Action action = new Action(Settext);

                    this.BeginInvoke(action);

                });

            thread.Start();

            lblcountchar.Text = "Processing ....";

        }

        private void Settext()

        {

            lblcountchar.Text = "File has " + charcnt.ToString() + " chars";

        }

    }

}

Task (asynchronous function)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Threads

{

    public partial class Form5 : Form

    {

        public Form5()

        {

            InitializeComponent();

        }

        private int Countchar()

        {

            int count = 0;

            string file = @"C:\Users\User\Desktop\myfile.txt";

            using (StreamReader reader = new StreamReader(file))

            {

                string filecont = reader.ReadToEnd();

                count = filecont.Length;

                Thread.Sleep(5000);

            }

            return count;

        }

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

        {

            Task<int> task = new Task<int>(Countchar);

            task.Start();

            label1.Text = "Processing ...";

            int count = await task;

            label1.Text = "no of chars = " + count.ToString();

        }

    }

}