

Forum:[Hack Forums](#)

Thread: [Single Thread > Multi Thread](#)

bmr03 - April 22, 2013, 4:14 pm

Is there an easy way to quickly make my TCP Client\Server multi threaded? I have a basic understanding of what it is, but it would be easier if someone could explain it, and maybe tell me how I could take the code for a single threaded TCP Client\Server and make it multi threaded. Thanks in advanced! ?

flameswor10 - April 22, 2013, 4:26 pm

This is an in-depth explanation from MSDN.

Remember to google your problems before asking!

Code using System;
using System.Threading;

```
// Simple threading scenario: Start a static method running
// on a second thread.
public class ThreadExample {
    // The ThreadProc method is called when the thread starts.
    // It loops ten times, writing to the console and yielding
    // the rest of its time slice each time, and then ends.
    public static void ThreadProc() {
        for (int i = 0; i < 10; i++) {
            Console.WriteLine("ThreadProc: {0}", i);
            // Yield the rest of the time slice.
            Thread.Sleep(0);
        }
    }

    public static void Main() {
        Console.WriteLine("Main thread: Start a second thread.");
        // The constructor for the Thread class requires a ThreadStart
        // delegate that represents the method to be executed on the
        // thread. C# simplifies the creation of this delegate.
        Thread t = new Thread(new ThreadStart(ThreadProc));

        // Start ThreadProc. Note that on a uniprocessor, the new
        // thread does not get any processor time until the main thread
        // is preempted or yields. Uncomment the Thread.Sleep that
        // follows t.Start() to see the difference.
        t.Start();
        //Thread.Sleep(0);

        for (int i = 0; i < 4; i++) {
            Console.WriteLine("Main thread: Do some work.");
            Thread.Sleep(0);
        }
    }
}
```

```

Console.WriteLine("Main thread: Call Join(), to wait until ThreadProc ends.");
t.Join();
Console.WriteLine("Main thread: ThreadProc.Join has returned. Press Enter to end program.");
Console.ReadLine();
}
}

```

Output:

Code [VB, C++, C#]

```

Main thread: Start a second thread.
Main thread: Do some work.
ThreadProc: 0
Main thread: Do some work.
ThreadProc: 1
Main thread: Do some work.
ThreadProc: 2
Main thread: Do some work.
ThreadProc: 3
Main thread: Call Join(), to wait until ThreadProc ends.
ThreadProc: 4
ThreadProc: 5
ThreadProc: 6
ThreadProc: 7
ThreadProc: 8
ThreadProc: 9
Main thread: ThreadProc.Join has returned. Press Enter to end program.
Reference: http://msdn.microsoft.com/en-AU/library/system.threading.thread.aspx

```

I think this should explain it fully.

If not, post what you're unsure about and I'll help you figure it out.

ismellike - April 22, 2013, 5:04 pm

I dont know if this is what you were asking, but this is what I do for multi threading.

```

Code Thread threadName = new Thread(new ThreadStart(delegate
{
    //The stuff to be threaded goes here
}));
threadName.Start();

```

```

Thread threadName2 = new Thread(new ThreadStart(delegate
{
    //The stuff to be threaded goes here
}));
threadName2.Start();

```

What the delegate does is that it pretty much just starts the threading and then continues the rest of the code.

bmr03 - April 22, 2013, 6:46 pm

Thanks for both of your guy's help, this was for my RAT so if you want PM me and I will give you a free copy when I finish it.

BaussHacker - April 22, 2013, 10:36 pm

Yes. Using asynchronous sockets would be your best shot.

<https://hackforums.net/showthread.php?tid=2962387>