

Forum:[Hack Forums](#)

Thread: [How to create a simple multithreaded program](#)

Verizur - March 15, 2016, 4:05 pm

In this tutorial I'll be teaching you the basics in multithreading.
This is very useful if you want to speed-up your programs actions.

First we'll need to add this reference to our project:

Code using System.Threading;

Now we have to decide how many threads we want to start:

Code static void Main(string[] args)

```
{
    int Threads = 5; // Number of threads
}
```

Now its time for the actual thread creation part:

Code static void Main(string[] args)

```
{
    int Threads = 5; // Number of threads
    for (int i = 1; i < Threads; i++)
    {
        Thread T = new Thread(new ThreadStart(Work)); // Declare thread
        T.Name = "Thread " + i; // Set a thread name
        T.Start(); // Start the thread
    }
}
```

And last but not least - the sub worker:

Code static void Work()

```
{
    Console.WriteLine(Thread.CurrentThread.Name + ": Hello world!");
    Console.ReadKey();
}
```

OUTPUT:

This is a very basic tutorial to help you understand how multithreading actually works.

Cecilia - March 15, 2016, 5:24 pm

What about tasks/parallels/threadpools?

Verizur - March 15, 2016, 6:40 pm

(Mar 15, 2016 05:24 PM)LaPanthere Wrote: [\[CITE\]](#) What about tasks/parallels/threadpools?

This is just a basic tutorial for beginners.?

+Haze - March 15, 2016, 11:37 pm

You can't cover a tutorial on threads and not talk about unsafe and safe thread calls. Like what is the proper way to handle cross thread calls. Also creating threads like this is ehheh, atleast use a thread pool if you don't intend n using Tasks/Parallel

Cecilia - March 15, 2016, 11:45 pm

(Mar 15, 2016 11:37 PM)+Haze Wrote: [\[CITE\]](#) You can't cover a tutorial on threads and not talk about unsafe and safe thread calls. Like what is the proper way to handle cross thread calls. Also creating threads like this is ehheh, atleast use a thread pool if you don't intend n using Tasks/Parallel
Its not terribly hard to invoke the ui thread from a different thread, or similar...

+Haze - March 16, 2016, 12:11 am

(Mar 15, 2016 11:45 PM)LaPanthere Wrote: [\[CITE\]](#) Its not terribly hard to invoke the ui thread from a different thread, or similar...

That's not the point, it should be covered imho.

Odus - March 16, 2016, 1:31 am

(Mar 16, 2016 12:11 AM)+Haze Wrote: [\[CITE\]](#) That's not the point, it should be covered imho.
A hell of a lot more should be covered and explained, even as a beginner tutorial. Instructing to paste code teaches nothing.

Assault - March 16, 2016, 5:01 am

```
Code int Threads = 5; // Number of threads
    for (int i = 1; i < Threads; i++)
fyi: i == 1 means 4 iterations, not 5 in your case
```

Rhez - March 16, 2016, 7:22 am

```
Code int Threads = 5; // Number of threads
    for (int i = 1; i < Threads; i++)
1<5 = True {Do the code here} - increment
2<5 = True {Do the code here} - increment
3<5 = True {Do the code here} - increment
4<5 = True {Do the code here} - increment
5<5 = False {Nothing} - nothing
```

It means that it will only have 4 iteration not 5 lol.

unSatisfied - March 16, 2016, 9:54 am

Why not introduce tasks? Most of the time Task.Run (with lambda expression) is a much more elegant solution.

mrnikbobjeff - March 16, 2016, 12:35 pm

(Mar 16, 2016 09:54 AM)unSatisfied Wrote: [\[CITE\]](#) Why not introduce tasks? Most of the time Task.Run (with lambda expression) is a much more elegant solution.
Especially because Thread is obsolete, Task should be used when available!

Verizur - March 16, 2016, 2:35 pm

Thanks for all the feedback guys. I know that there's a lot that I'm not talking about in the thread.
This is because I just made a quick VERY BASIC thread on how multithreading works for users who are making their first bots etc.

Oliver Sykes - March 16, 2016, 5:28 pm

(Mar 15, 2016 11:45 PM)LaPanthere Wrote: [\[CITE\]](#) Its not terribly hard to invoke the ui thread from a different thread, or similar...

It's not terribly hard to write the following either.

```
Code var thread = new Thread({x => SomeMethod.DoShit(x)});
```

(Mar 16, 2016 02:35 PM)Verizur Wrote: [\[CITE\]](#) Thanks for all the feedback guys. I know that there's a lot that I'm not talking about in the thread.

This is because I just made a quick VERY BASIC thread on how multithreading works for users

who are making their first bots etc.

If they can't figure this much out, they really don't need to be writing bots imo.

unSatisfied - March 16, 2016, 5:55 pm

(Mar 16, 2016 12:35 PM)mrnikbobjeff Wrote: [\[CITE\]](#) Especially because Thread is obsolete, Task should be used when available!

Totally agree, I never use Thread anymore.

Cecilia - March 16, 2016, 6:07 pm

(Mar 16, 2016 05:55 PM)unSatisfied Wrote: [\[CITE\]](#) Totally agree, I never use Thread anymore. LOL.

Thread isn't fucking deprecated/obsolete, its extremely useful in different circumstances. For multi-threading (repeating same action over and over) it is not ideal but still can be used.

Odus - March 17, 2016, 2:00 am

(Mar 16, 2016 12:35 PM)mrnikbobjeff Wrote: [\[CITE\]](#) Especially because Thread is obsolete, Task should be used when available!

Threading is far from obsolete. I agree use tasks when possible but they're used for different purposes.

unSatisfied - March 18, 2016, 11:15 pm

(Mar 16, 2016 06:07 PM)LaPanthere Wrote: [\[CITE\]](#) LOL.

Thread isn't fucking deprecated/obsolete, its extremely useful in different circumstances. For multi-threading (repeating same action over and over) it is not ideal but still can be used.

Of course is can be useful, but in the majority of scenarios, I find that Tasks can handle what I need to do just as good if not better. Maybe I just haven't coded anything that needs Threads explicitly.

Rolex™ - March 20, 2016, 7:48 am

Or make a List<Task> and then add all the Tasks to that list in the loop and use PHP Code
<?php

Task.WhenAll()

to wait for all of them to get done then show the results.

I like Task better, in my honest opinion.

Decim - March 20, 2016, 7:54 am

(Mar 16, 2016 05:55 PM)unSatisfied Wrote: [\[CITE\]](#) Totally agree, I never use Thread anymore.

Task is not an alternative to threading, they're both different things.

Rolex™ - March 20, 2016, 8:00 am

(Mar 20, 2016 07:54 AM)Arcanecfg Wrote: [\[CITE\]](#) Task is not an alternative to threading, they're both different things.

Yes but even though Task is higher level than the low level Thread, they can be used to achieve the same outcome. If you want to do something thing "x" y amount of times why would you use Thread over Task?

unSatisfied - March 20, 2016, 2:03 pm

(Mar 20, 2016 08:00 AM)Rolex™ Wrote: [\[CITE\]](#) Yes but even though Task is higher level than the low level Thread, they can be used to achieve the same outcome. If you want to do something thing "x" y amount of times why would you use Thread over Task?

Or you can simply use Task.Run if you want to perform an action just once:

I totally agree about tasks though, it's a much more elegant and less verbose solution for multi-threading.

Corios - *March 20, 2016, 5:44 pm*

(Mar 20, 2016 07:54 AM)Arcanecfg Wrote: [\[CITE\]](#) Task is not an alternative to threading, they're both different things.

Yes it is, what the fuck. Are they the same thing? No, but it is 100% an alternative to threading.

(Mar 16, 2016 06:07 PM)LaPanthere Wrote: [\[CITE\]](#) its extremely useful in different circumstances. Give an example.