

Forum: [Hack Forums](#)

Thread: [Stopping thread pool](#)

VUEDUE - May 26, 2021, 9:30 am

I've been learning more and more about threading and multi-threading in VB. I wrote this code below:

```
Code Dim max As Integer = NumericalUpDown1.Value - 1
    For X As Integer = 0 To max
        System.Threading.ThreadPool.QueueUserWorkItem(Sub() myTHREAD())
    Next
```

However, lets say the threads do something infinite; a loop maybe. How do I kill all threads in the threadpool?

I've searched around but I didn't manage to find a clean solution.

HFDone - May 26, 2021, 10:45 am

its not up to threadpool choose when to stop it in your loop

VUEDUE - May 26, 2021, 12:06 pm

(May 26, 2021 10:45 AM)HFDone Wrote: [\[CITE\]](#) its not up to threadpool choose when to stop it in your loop

Well I realized I could set a parameter. I could do something like:

```
Code Public var As Integer = 0
```

'Change var to equal '1' somewhere along the line.

```
Public Sub myThread()
Do
    If var == 1 Then
        Exit Do
    End If

    Console.WriteLine("Loop is still going!")
Loop
```

```
Console.WriteLine("Done!")
End Sub
```

Just as a quick example yenko. But I figured there would be a systematic way to do it.

Shadow_Cloud - May 26, 2021, 11:11 pm

Is there any specific reason you aren't using a cancellation token?

<https://docs.microsoft.com/en-us/dotnet/api/system.threading.cancellationtoken?view=net-5.0>

Your example won't allow thread safe access to the variable you are checking.

VUEDUE - May 27, 2021, 4:20 am

(May 26, 2021 11:11 PM)Shadow_Cloud Wrote: [\[CITE\]](#) Is there any specific reason you aren't using

a cancellationToken?

<https://docs.microsoft.com/en-us/dotnet/api/system.threading.cancellationtoken?view=net-5.0>

I had no idea they existed up until now, but I'll look into it.

(May 26, 2021 11:11 PM)Shadow_Cloud Wrote: [\[CITE\]](#) Your example won't allow thread safe access to the variable you are checking.

How so? I'm not sure what you mean by that.

Shadow_Cloud - May 27, 2021, 5:14 am

(May 27, 2021 04:20 AM)VUEDUE Wrote: [\[CITE\]](#) I had no idea they existed up until now, but I'll look into it.

How so? I'm not sure what you mean by that.

So imagine you had 10 threads and 2 of them need to stop. Based on your implementation below, you wouldn't be able to tell which ones need to stop and which ones need to carry on. This will actually just kill all of them.

Code Public var As Integer = 0

'Change var to equal '1' somewhere along the line.

```
Public Sub myThread()
```

```
Do
```

```
    If var == 1 Then
```

```
        Exit Do
```

```
    End If
```

```
    Console.WriteLine("Loop is still going!")
```

```
Loop
```

You could have something like a concurrentDictionary keep track of the threads and their running state so you only need to have the thread do a lookup on its own ID to determine if it should be cancelled.