

Diagnosing async C# code

Sergey Teplyakov

Principal software engineer

Microsoft

Debugging is easy! Right?

I DON'T KNOW WHERE YOU ARE, I DON'T KNOW HOW YOU WORK, BUT I WILL FIND YOU, AND

DEBUGGING

I WILL FIX YOU

Stack traces for the rescue!



```
□public class Service
     internal void Initialize()
         Thread.Sleep(42 * 1000);
     public static Service Create()
         var service = new Service();
         service.Initialize();
         return service;
🖃 class Program
     static void Main(string[] args)
         var service = Service.Create();
         Console.WriteLine(
             $"Service {service} is constructed.");
```

01 -	BlockingS.	hronous.csproj	Program.cs	WaitHandle.cs	Thread.CoreCLR.cs 🗗 🗙 M	anualResetEventSlim	.CS	÷ 🌣
C#	Miscellaneo	us Files	ح 🖓 Sys	tem.Threading.Thread	👻 🕥 Sleep(int i	millisecondsTimeout))	→ ‡
	245 246 247 248 249 250 251 252	// a Th [Method private /// <su /// Sus /// for /// ==</su 	nread instance dImpl(MethodIm e static extern ummary> spends the curr rces the thread Timeout.Infin:). plOptions.Interna n IntPtr Interna rent thread for 1 d to give up the ite, no timeout w	alCall)] .GetCurrentThread(); cimeout milliseconds. If remainder of its timesli vill occur.	timeout == 0, ce. If timeout	÷	
	253 254 255 256	/// [Method private	summary> dImpl(MethodIm e static exter	olOptions.Interna n void <mark>SleepInte</mark>	alCall)] nal(int millisecondsTime 	out);		
•	257 258 259 260 261 262 263 264 265 266 267 268	E /// <su /// Wai /// Onl /// a e /// [Methoc private public</su 	static void S ummary> it for a lengt ly take a few n explicit busy summary> dImpl(MethodIm e static extern static void Sp	leep(int millised n of time proport machine instruct: loop because the plOptions.Interna n void SpinWaitIn pinWait(int itera	<pre>condsTimeout) => SleepInt cional to 'iterations'. ions. Calling this API i hardware can be informed alCall)] nternal(int iterations); ations) => SpinWaitIntern</pre>	ernal(millisecons Each iteration s preferable to that it is bus al(iterations);	is should coding sy waiting	it) g.
99 %	269 5 🔹 🖓	│ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │	oort(RuntimeHe 🐳 🔻	lpers.QCall, Chai ◀	rSet = CharSet.Unicode)]	▶ Ln: 257	Ch: 62 SPC	LF
Call	Stack						-	ч ×
N ➡ S 0 0	Jame ystem.Priva 1 - Blocking 1 - Blocking 1 - Blocking	te.CoreLib.dll!System.T gSynchronous.dll!_01 gSynchronous.dll!_01 gSynchronous.dll!_01	Threading.Thread.Sle _BlockingSynchrono _BlockingSynchrono _BlockingSynchrono	eep(int millisecondsTim us.Service.Initialize() Lir us.Service.Create() Line us Program Main(string	eout) Line 257 ne 10 16 10 aros) Line 25			Langi 🔶 C# C# C#

01 - BlockingShronous.csproj	Program.cs 🛥 🗙 WaitHand	dle.cs Thread.Cor	eCLR.cs	ManualRe	setEventS	lim.cs		₹ \$		
🖽 01 - BlockingSynchronous	🚽 ॡ_01BlockingS	Synchronous.Service	🝷 😪 Initializ	ze()				- +		
2 [using System.Th 3 4 namespace _01 5 6 □ public class Se 7 { 8 □ internal vo 9 { 10 Ø { 11 } 12 1 13 □ public stat 14 { 15 var ser 16 Service 17 k 19] } 20	reading; _BlockingSynchronous; rvice id Initialize() Sleep(42 * 1000); ic Service Create() vice = new Service(); Initialize(); service;									
21 □ class Program 22 ↓ { 23 □ static void 24 ↓ { 25 ↓ var ser 26 ♥ No issues found Call Stack	Main(string[] args) vice = Service.Create() .WriteLine(∛ ▼ ◀);		•	Ln: 10	Ch: 9	SPC	CRLF		
Name							L	ang		
System Private CoreLib.dll!System Threading Thread.Sleep(int millisecondsTimeout) Line 257										
M 01 BlockingSynchronous dll 01	Placking Synchronous Service In	itialize() Line 10						-#		

01 - BlockingSynchronous.dll!_01___BlockingSynchronous.Service.Initial2e() Line 16 01 - BlockingSynchronous.dll!_01___BlockingSynchronous.Service.Create() Line 16 01 - BlockingSynchronous.dll!_01___BlockingSynchronous.Program.Main(string[] args) Line 25

C#

C#

01 - BlockingShronous.csproj	Program.cs 中 🗙	WaitHandle.cs	Thread.CoreCLR.cs	ManualRe	esetEventS	lim.cs		₹ 1	₽
🖼 01 - BlockingSynchronous	✓ ^A ² ³ _01	BlockingSynchronous.Se	ervice 👻 😚 Creat	te()				- ÷	=
2 [using System.Th: 3 4 namespace _01 5 6 □ public class Se: 7 { 8 □ internal vo: 9 { 10 Thread. 11 } 12 13 □ public stat: 14 { 15 var service 17 18 } 19 [}	reading; _BlockingSynchro rvice id Initialize() Sleep(42 * 1000) ic Service Creat vice = new Serv: .Initialize(); service;	onous;); te() ice();							
21 E class Program 22 { 23 E static void 24 { 25 Var servent 26 Voissues found	Main(string[] a vice = Service.(.WriteLine(∛▼	args) Create();			Ln: 16	Ch: 9	SPC	CRLF	
								- 11 \	
System Private Corel ib dll	reading Thread Sleep	(int millisecondsTimeou	t) Line 257					C#	
01 - BlockingSynchronous.dll! 01	Blocking Synchro <u>nous.</u>	Service.Initialize() Line 1	0					C#	

O1 - BlockingSynchronous.dll!_01___BlockingSynchronous.Service.Create() Line 16
 O1 - BlockingSynchronous.dll!_01___BlockingSynchronous.Program.Main(string[] args) Line 25

C#

C#

01 - BlockingShronous.csproj	Program.cs 🛥 🗙 WaitHandle.cs	Thread.CoreCLR.cs	ManualResetEven	tSlim.cs	⇒ \$
🖽 01 - BlockingSynchronous	✓ ^A S_01BlockingSynchronous.Pr	ogram 👻 🖓 Main(st	tring[] args)		
2 Lusing System.Thr 3	reading;				Â
4 namespace _01 5 6 6 public class Ser 7 { 8 internal voi 9 { 10 Thread.S 11 } 12 13 13 public stati 14 { 15 var serv 16 ; 18 } 19	_BlockingSynchronous; rvice id Initialize() Sleep(42 * 1000); ic Service Create() vice = new Service(); .Initialize(); service;				
21	Main(string[] args) /ice = <mark>Service</mark> .Create(); .WriteLine(🖋 🔹 🔺		► Ln: 25	Ch: 9 SPC	CRLF
Call Stack					→ ₽ X
Name					Langi 🔺
System.Private.CoreLib.dll!System.Th	reading.Thread.Sleep(int millisecondsTimeou	t) Line 257			C#
01 - BlockingSynchronous.dll!_01B	BlockingSynchronous.Service.Initialize() Line 1	0			C#
01 - BlockingSynchronous.dll! 01 B	BlockingSynchronous.Service.Create() Line 16				C#

🍽 01 - BlockingSynchronous.dll!_01___BlockingSynchronous.Program.Main(string[] args) Line 25 🛛

C#

THE APPLICATION GOT STUCK ON PROD STACKTRACES OF THE ISSUE

imgflip.com

WE DO HAVE STACKTRACES. Right? It was a Synchronous code that stuck!?

Let's look at a very simple async case

```
□public class Service
□public class Service
                                                                  internal async Task InitializeAsync()
     internal void Initialize()
                                                                      await Task.Delay(42 * 1000);
         Thread.Sleep(42 \times 1000);
                                                                  public static async Task<Service> CreateAsync()
     public static Service Create()
                                                                      var service = new Service();
         var service = new Service();
                                                                      await service.InitializeAsync();
         service.Initialize();
                                                                      return service;
         return service;
                                              Sync 2 async
                                                            □class Program
🖃 class Program
                                                                  static async Task Main(string[] args)
     static void Main(string[] args)
                                                                      var service = await Service.CreateAsync();
         var service = Service.Create();
                                                                      Console.WriteLine(
         Console.WriteLine(
                                                                          $"Service {service} is constructed.");
             $"Service {service} is constructed.");
```

Debugging async code

Source Not Avai	ble Program.cs ManualResetEventSlim.cs + × Task.cs	•	¢ Sel
💷 Miscellaneou	Files 🔹 🕫 System. Threading. Manual ResetEventSlim 🔹 😚 Wait (int milliseconds Timeout, Cancellation Token cancellation Token)	- =	ution
572 573 574	<pre>// ** the actual wait ** if (!Monitor.Wait(m_lock, realMillisecondsTimeout)) return false; // return immediately if the timeout has expired.</pre>		Explorer
575 576 577 578 579 580 581 582 583 583 584 585 586	<pre> } finally { // Clean up: we're done waiting. Waiters; } // Now just loop back around, and the right thing will happen. Either: // Now just loop back around, and the right thing will happen. Either: // 1. We had a spurious wake-up due to some other wait being canceled via a different cancellationToken (rewait) // or 2. the wait was successful. (the loop will break) } </pre>		Git Changes Diagnostic Tools Diag
586 587 & 588 590 591 592 E 593 594 595 595 596	<pre>// automatically disposes (and unregisters) the callback return true; // done. The wait was satisfied. /// <summary> /// <summary> /// Releases all resources used by the current instance of <see cref="ManualResetEventSlim"></see>. /// </summary> /// <remarks> /// Unlike most of the members of <see cref="ManualResetEventSlim"></see>, <see cref="Dispose()"></see> is not</remarks></summary></pre>		gnostic Analysis
100 % 🝷 🥋	🛛 No issues found 🛛 😽 🔹 🔸 👘 👘 🕹 Ln: 587 Ch: 71 SP	C LF	
Call Stack Name System.Privat System.Privat System.Privat System.Privat (Waiting on A 02 - Blocking		- ₽ > ■ ►	< lin
Call Stack Brea	points Exception Settings Command Window Immediate Window Output Threads Autos Locals Watch 1		

Where is our code?!?!?

Source Not Available 🛥 🗙 Program.cs ManualResetEver

ManualResetEventSlim.cs Task.cs

Source Not Available

Source information is missing from the debug information for this module

You can view disassembly in the Disassembly window. To always view disassembly for missing source files, change the setting in the Options dialog.

C	all Stack	₹ ₽
	Name	Lang
	System.Private.CoreLib.dll!System.Threading.ManualResetEventSlim.Wait(int millisecondsTimeo	C#
	System.Private.CoreLib.dll!System.Threading.Tasks.Task.SpinThenBlockingWait(int millisecondsTi	C#
	System.Private.CoreLib.dll!System.Threading.Tasks.Task.InternalWaitCore(int millisecondsTimeou	C#
	System.Private.CoreLib.dll!System.Runtime.CompilerServices.TaskAwaiter.HandleNonSuccessAn	C#
	System.Private.CoreLib.dll!System.Runtime.CompilerServices.TaskAwaiter.GetResult() Line 105	C#
	[Maiting on Asyne Operation, double click or proceedator to view Asyne Call Stacks]	
R	02 - BlockingAsync.dll_02BlockingAsync.Program. <main>(string[] args)</main>	Un

Solution Explorer

Git Changes

Diagnos

Tools

Diagnostic Analysis

÷.

Async Main: the new feature in C# 7

- Async entry points are very common
- Starting with C# 7.1 the main method can be asynchronous

```
Class Program
Async main

Static async Task Main(string[] args)

Var service = await Service.CreateAsync();
Console.WriteLine($"Service {service} is constructed.");
}
```

```
Class Program Translated code
{
    static void Main(string[] args)
    {
        MainAsync(args).GetAwaiter().GetResult();
    }

    static async Task MainAsync(string[] args)
    {
        var service = await Service.CreateAsync();
        Console.WriteLine($"Service {service} is constructed.");
    }
```

TLDR; the async flow: TODO: fix

");

public class Service
{
internal async Task InitializeAsync()
var delavTask = Task Delav(42 * 1000)
await delayTask:
Concolo Writeline("InitializeAcure is done ");
Console.writeLine("initiatizeAsync is done.");
public static async Task <service> CreateAsync()</service>
<pre>var service = new Service();</pre>
<pre>var initializeTask = service InitializeAsync();</pre>
await initializeTask;
return service:
ICLASS Program
static async Task Main(string[] args)
{
<pre>var createServiceTask = Service.CreateAsync();</pre>
<pre>var service = await createServiceTask;</pre>
Console.WriteLine(\$"Service {service} is constructed



14

And how we should debug it?



"Call Stacks" to "Async Call Stacks"

Program.cs	ManualResetEventSlim.cs 🛥 🗙 Task.cs					÷	¢a Soli
C# Miscellaneous F	Files	🗕 🔩 System. Threading. Manual	ResetEventSlim	🗕 😙 Wait(int mil	lisecondsTimeout, CancellationToken cancellationToken)) - =	÷ tion
572 573 574 575 576		<pre>// ** the actual wai if (!Monitor.Wait(m_</pre>	t ** lock, realMillisecor return immediately	ndsTimeout)) if the timeout has	expired.		Explorer Git Char
577 578 579 580 581 582 583 584 585 586 586 587 588	} } } // automatical	<pre>{ // Clean up: we're d Waiters; } // Now just loop back ar // 1. We had a spuri // or 2. the wait was s ly disposes (and unregis</pre>	one waiting. ound, and the right ous wake-up due to s uccessful. (the loop ters) the callback	thing will happen. some other wait bein b will break)	Either: g canceled via a different cance	llatio	iges Diagnostic Tools Diagnostic Analysis
589 ⇒ 590 591 592 [121% • @ •	return true; //	done. The wait was satis	fied.		▶ Ln: 586 Ch: 18	SPC LE	Ŧ
Call Stack		▼ ₽ ×	Threads			- ₽ 3	×
Name System.Private.(System.Private.(System.Private.(System.Private.(System.Private.(UZ - BiockingAs; Call Stack Breakn	CoreLib.dll!System.Threading.ManualResetEventSlin CoreLib.dll!System.Threading.Tasks.Task.SpinThenBl CoreLib.dll!System.Threading.Tasks.Task.InternalWai CoreLib.dll!System.Runtime.CompilerServices.TaskA Corel ib.dll!System.Runtime.CompilerServices.TaskA ync Operation, double-click or press enter to view A sync.dll!_U2BlockingAsync.Program.Main(string[]	Lang n.Wait(int millisecondsTimeo C# ockingWait(int millisecondsTi C# tCore(int millisecondsTimeou C# waiter.HandleNonSuccessAn C# waiter.GetResult0 Line 105 C# sync Call Stacks] args) Line 29 C# pmediate Window: Output	Search ID Managed ID ▲ Process ID: 32212 (1 thread) P ▲ 37232 1 ▲ Threads Autor Locals Watch	 □ □ □ · □ · □ Group by: Category Name ∞ Main Thread Main Thread 	Process ID Columns B Columns B Columns B Columns Col	IResetEventS	Slin

16

Async call stack

AsyncService.cs	Program.cs	ManualResetEventSlim.cs	Task.cs 🗗 🗙		~	¢	Parallel Stac	cks ዋ	×						
C# Miscellaneous Files		 System.Threading.Tasks.1 	ask	- 🏠 InnerInvoke()	-	∳	Search				- م		View:	Tasks 👻	
2356 2357 2358 2359 2360 2361 2362 2363 2364 2365 2366 2367 2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379 00	<pre> Debug./ // Only Unsafe }; /// <summax (m_a="" (m_a<="" <="" acf="" debug.="" if="" internal="" invo="" summax="" td="" the="" v:="" {=""><td>Assert(obj is Task); y used privately to pass d .As<task>(obj).InnerInvoke ry> tual code which invokes th ary> irtual void InnerInvoke() oke the delegate Assert(m_action != null, " action is Action action) tion(); turn; action is Action<object?> tionWithState(m_stateObject turn; Fail("Invalid m_action in</object?></task></td><th><pre>irectly to EC. (); e body of the Null action in actionWithStat t); Task");</pre></th><td>Run task. This can be ov InnerInvoke()"); e)</td><td>/erridden in deriv€</td><td>↓</td><td></td><td>1 Asyr → Task Serv Prog → Prog</td><td>nc Logica k.Delay vice.Initia gram.Ma gram.<m< td=""><td>al Stack lizeAsync in lain></td><td></td><td></td><td>view:</td><td>IdSKS</td><td></td></m<></td></summax></pre>	Assert(obj is Task); y used privately to pass d .As <task>(obj).InnerInvoke ry> tual code which invokes th ary> irtual void InnerInvoke() oke the delegate Assert(m_action != null, " action is Action action) tion(); turn; action is Action<object?> tionWithState(m_stateObject turn; Fail("Invalid m_action in</object?></task>	<pre>irectly to EC. (); e body of the Null action in actionWithStat t); Task");</pre>	Run task. This can be ov InnerInvoke()"); e)	/erridden in deriv€	↓		1 Asyr → Task Serv Prog → Prog	nc Logica k.Delay vice.Initia gram.Ma gram. <m< td=""><td>al Stack lizeAsync in lain></td><td></td><td></td><td>view:</td><td>IdSKS</td><td></td></m<>	al Stack lizeAsync in lain>			view:	IdSKS	
Call Stack					▼Д Х	×	Threads								
Name					Langi	^	Search	ړ) - @		Group	ov: Pr	ocess ID		•
 [Async] System.Privat [Async] 02 - Blocking [Async] 02 - Blocking [Async] 02 - Blocking 	e.CoreLib.dll!Systen Async.dll!_02Bloc Async.dll!_02Bloc	n.Threading.Tasks.Task.Delay :kingAsync.Diagram.Service.InitializeA :kingAsync.Diagram.Service.CreateAs	Async() Line 12 ync() Line 18		C# C# C#		Process	D N	Managed 80 (1 thr	l ID ead)	Category	N	lame	Location	
02 - BlockingAsync.d	High and the second s	nc.Diagram.Program. <main>(string]</main>	args)		Un		면 다 45	76 1			🖂 Main Thre	ad Ma	ain Thread	System.	Pr

Async call stack



Async call stack



Sync vs. Async Call stacks

6	⊡public class Service	Sync version		7 8	⊡ pub {	blic c	lass Service		Async version
, 8 9 \$≻ 10 <i>⊗</i>	<pre>internal void Initialize() { Thread.Sleep(42 * 1000);</pre>			9 10 11		inte: {	ernal async Ta await Task.De	ask InitializeA elay(42 * 1000)	sync() ;
11 12 13 14 15 16 17 18 19	<pre>} public static Service Create() { var service = new Service(); service.Initialize(); return service; } }</pre>		\$\$ ₽	120° 13 14 15 16 17 18 19 20 21	E [}	publ	ic static asy. var service = await service return service	ync Task <servic = new Service() e.InitializeAsy ce;</servic 	e> CreateAsync() ; nc();
20 21 22 23 24 25 26 27 28	<pre>class Program { static void Main(string[] args) { var service = Service.Create() Console.WriteLine(\$"Service {service } } }</pre>); service} is constructed.");		22 23 24 25 26 27 28 29	□ cla { 	ass Prostat. { }	rogram ic async Task var service = Console.Write	< Main(string[] = await Service eLine(\$"Service	<pre>args) .CreateAsync(); {service} is constructed.");</pre>
29 99 % •	🖗 🗢 ivo issues touna 🛛 🗳 🔹		99 % Call	Stack	~	No issu	ies found		
Call Stack			N	lame					
Name System.F 01 - Bloc	rivate.CoreLib.dll!System.Threading.Thread.Sleep(int millise kingSynchronous.dll!_01BlockingSynchronous.Service.Ini kingSynchronous.dlll_01BlockingSynchronous.Service.Cr	condsTimeout) Line 257 tialize() Line 10	/ب / المح / المح / المح	Async] Sys Async] 02 - Async] 02 - Async] 02 -	tem.Pri - Blocki - Blocki - Blocki	ivate.Con ingAsyn ingAsyn ingAsyn	nc.dll!_O2Blockin nc.dll!_O2Blockin nc.dll!_O2Blockin nc.dll!_O2Blockin	nreading.Tasks.Task.D gAsync.Diagram.Serv gAsync.Diagram.Serv gAsync.Diagram.Prod	Delay vice.InitializeAsync() Line 12 vice.CreateAsync() Line 18 gram Main(string[] args) Line 27
01 - Bloc	kingSynchronous.dll!_01BlockingSynchronous.Program.N	Aain(string[] args) Line 25	0	2 - Blockir	ngAsyn	nc.dll!_02	2BlockingAsync.	Diagram.Program. <n< td=""><td>Main>(string[] args)</td></n<>	Main>(string[] args)

Async code is viral!



Is sync over async dangerous?

- A simple file system cache
- Initialized lazily to avoid unnecessary work
- Used in many synchronous paths
- Used only on the backend

```
public class DirectoryCache
{
    private readonly Lazy<List<string>> _files;
    public List<string> Files => _files.Value;
    public DirectoryCache()
    {
        _files = new Lazy<List<string>>(() => ReadFilesAsync().GetAwaiter().GetResult());
    }
    private static async Task<List<string>> ReadFilesAsync()
    {
        await Task.Delay(3 * 1000);
        return new List<string> { "42.txt" };
    }
}
```

Let's try it out!

DirectoryCache.cs + X	AsyncTaskMethodBuilderT.cs	TaskContinuation.cs	WaitHandle.cs	± ⇔	Test Explorer 👎 🗙	
고 03 - SyncOverAsync	- ��\$_03Sy	ncOverAsync.Cache.FileNameT	Tests 🝷 😚 FirstFileIs42txt()	▼ ‡	📕 🕨 – 🧭 🖄 🖾 3 🔗 2 😣 0 争	1 📓 🕶 🍾 [= 🕀
8				^	Test	Duration Traits
9 Lassemi	bly: CollectionBehavior((vDarallelThreads = 2)]	ollectionBehavior.Col	lectionPerClass,		▲ 3 - SyncOverAsync (3)	6 sec
10 11					↓ ① _03SyncOverAsync (1)	
12 namespa	ace _03SyncOverAsync.C	Cache;			▲	6 sec
13					FileCountTests (1)	3 sec
14 ⊡public	class DirectoryCache				▷ 🎲 FileNameTests (1)	3 sec
					Group Summary	
aller to	A THE	State of the second second	1 American		Group Summary	
THE R.	A MARCHAR	States and a state	The second		_03SyncOverAsync.Cache	
		A handler	35 11/2		Tests In group: 2	
			CAR VOA		C Total Duration: 6 Sec	
		1.50	A		Outcomes	
NUMBER OF THE OWNER OF THE OWNER	Self - C - The Self of the self	and the second se	CARE REPORTS OF CARES	CONTRACTOR OF	🗖 🤉 Dacced	

Regular stack trace is not very helpful!

TaskContinuation.cs	Task.cs	ManualResetEventSlim.c	cs 🛥 🗙 DirectoryCache.cs	WaitHandle.cs	TaskAwaiter.cs	ThreadPoolExhaustion	Test.cs	Lazy.cs	AsyncTestSyncContext.cs		;	to S
CI Miscellaneous Files			👻 🖓 System. Thread	ding.ManualResetEve	ntSlim	- 0	🗇 Wait(int mil	lisecondsTimeo	ut, CancellationToken cancel	lationToken)		r ∯ lion
578 579 580 581 582 583 583 584 585		// Cl Waite // Now ju // 1. // or 2. }	lean up: we're done waitin ars; ust loop back around, and . We had a spurious wake-u . the wait was successful.	g. the right thin p due to some (the loop wil	g will happen. Eit other wait being ca l break)	cher: anceled via a diffe	erent canc	ellationTok	en (rewait)			Explorer Git Changes
586 587 588	} } // au	itomatically dispos	ses (and unregisters) the	callback								Diagnostic
589 590 591 592 593 594 595 596 597 598 11 599 600 601 602 100 % ♥ ♥ ♥ No iss	<pre>/// summar /// Release /// /// <remark /// Unlike /// Unlike /// thread- /// public void { Dispose GC.Supp Les found</remark </pre>	true; // done. The cy> es all resources us ary> <s> most of the member -safe and may not b cks> d Dispose() e(true); oressFinalize(this) 중 ▼ ◀</s>	<u>e wait was satisfied.</u> sed by the current instanc rs of <see <b="">cref="ManualRes be used concurrently with);</see>	e of <see <b="">cref etEventSlim"/> other members o</see>	= "ManualResetEvents , <see <b="">cref="Dispoe of this instance.</see>	Glim"/>. se()"/> is not				Ln: 589 Ch: :	58 SPC	fools Diagnostic Analysis ► ц
Call Stack				Three	ıds						↓ ↓	١x
 Name System.Private.CoreLib.d System.Private.CoreLib.d System.Private.CoreLib.d System.Private.CoreLib.d System.Private.CoreLib.d 	II:System.Threadi II:System.Threadi II:System.Runtim II:System.Runtim tion, double-clic 13SyncOverAs II:System.Lazy < S II:System.Lazy < S cention. Settings	ing. Manual Reset Event Slim ing. Tasks. Task. Spin Then Blo ing. Tasks. Task. Internal Wait e. Compiler Services. Task Av e. Compiler Services. Task Av k or press enter to view As ync. Cache. Directory Cache ystem. Collections. Generic ystem. Collections. Generic s. Command Window. In	Wait(int millisecondsTimeout, Syster ockingWait(int millisecondsTimeout, Syster vaiter.HandleNonSuccessAndDebugg vaiter.SystemCanon > GetResult() sync Call Stacks] c.ctor.AnonymousMethod_3_0() Line .List <string> .ViaFactory(System.Thr .List<string> .CreateValue() Line 433 nmediate WindowOutput</string></string>	Langu ▲ Searc n C# Searc sy C# P ger C# P i C# P Three	h ID Managed ID 15488 10 16408 0 8664 16 11812 17 6636 14 16480 15 Managed ID 15488 10 16408 0 16636 14 16480 15	Category N @ Worker Thread <1	Name No Name> No Name> No Name> No Name> Vorker Thread Vorker Thread	roup by: Proce Location <not available<br=""><not available<br="">System.Priv System.Priv <not available<br=""><not available<="" td=""><td>ss ID Columnation Columnation</td><td>mns ▼</td><td>e.WaitOneNc</td><td>oc ₩</td></not></not></not></not>	ss ID Columnation	mns ▼	e.WaitOneNc	oc ₩

<u>Async stack trace & Parallel Tasks</u>

Source Not Found Di	irectoryCache.cs +	× AsyncTaskMethod	BuilderT.cs TaskContin	uation.cs	WaitH	landle.cs	₹ ¢	Para	allel Stacks 🕒	×						Solu
코 03 - SyncOverAsync	-	℃ _03SyncOverAsyr	nc.Cache.DirectoryCache 🝷 😚	DirectoryCach	e()		- ÷	Sear	rch		م) - \leftarrow \rightarrow	View: Tasks	- - -	,	NQ 🖺 👘
6 using Xun	nit;			6] 1			▲ 	Son	ne tasks may b	e missing b	ecause the de	ebugger atta	ached to an already	y running pro	cess.	Explore X
8 [assembly 9 10 namespace 11 12 □public cl. 13 { 14 priva: 15 publi. 17 18 √ □ ▶ publi. 19 { 20	<pre>: CollectionBe : _03SyncOve .ass DirectoryC .te readonly La .c List<string> .c DirectoryCac .files = new La .te static asyn .wait Task.Dela .return new List .ass FileCountT s found</string></pre>	<pre>havior(Collection rAsync.Cache; ache zy<list<string>> Files => _files he() c Task<list<string>> c Task<list<string>> %</list<string></list<string></list<string></pre>	<pre>_files; _files; Value; (() => ReadFilesAsync() ng>> ReadFilesAsync()</pre>	rClass, Ma).GetAwait	er().Ge	elThread tResult(Ch: 28	spc CRLF		 AsyncTaskM TestCollect XunitTestA: XunitTestA: XunitTestA: Task.Run XunitTestA: TestAssemi XunitTestFr 	1 Async Lo MethodBuild ionRunner < ssemblyRunn ssemblyRunn 1 Async Log ssemblyRunner <> ssemblyRunner <> rameworkExe	gical Stack er < >.Start < > >.RunAsync her.RunTestColle her.RunTestColle ical Stack her.RunTestColle .RunAsync ecutor.RunTestColle	ectionAsy ectionsAs ections ases	1 As Svnchronizatio MaxConcurren ExecutionConto MaxConcurren XunitWorkerTh	aync Logical Sta nContextTaskSo cySyncContext. cySyncContext. ext.RunInternal cySyncContext. rread.QueueUso	ck theduler_ccto RunOnSyncC WorkerThrea erWorkitem.A	r Git Changes Diagnostic Tools Diagnostic Analysis
Call Stack				→ I ×	Threads											• 1 ×
Name	c.dll!_03SyncOverA SyncOverAsync.Ca System.Lazy <system. System Lazy<system< th=""><td>Async.Cache.DirectoryCa ache.DirectoryCachectc .Collections.Generic.List .Collections.Generic.List</td><td>che.ReadFilesAsync() Line 26 r.AnonymousMethod_3_0() Lin <string>>.ViaFactory(System.Th <string>>.SecutionAndPublica</string></string></td><td>Langi ▲ C# e 20 C# rea C#</td><td>Search P</td><td>ID 16700 6</td><td>Managed ID</td><td>Cate ক্রী V</td><td>。 egory Vorker Thread</td><td>Name Worker Th</td><td>Group by: Locatic read V Syst</td><td>Process IE on tem.Net.Soc</td><td>ckets.dll!System.Ne</td><td>Columns - et.Sockets.Soc</td><td>⊞ 🗆 🖞</td><td></td></system<></system. 	Async.Cache.DirectoryCa ache.DirectoryCachectc .Collections.Generic.List .Collections.Generic.List	che.ReadFilesAsync() Line 26 r.AnonymousMethod_3_0() Lin <string>>.ViaFactory(System.Th <string>>.SecutionAndPublica</string></string>	Langi ▲ C# e 20 C# rea C#	Search P	ID 16700 6	Managed ID	Cate ক্রী V	。 egory Vorker Thread	Name Worker Th	Group by: Locatic read V Syst	Process IE on tem.Net.Soc	ckets.dll!System.Ne	Columns - et.Sockets.Soc	⊞ 🗆 🖞	
System.Private.CoreLib.dlllS System.Private.CoreLib.dlllS System.Private.CoreLib.dlllS 03 - SyncOverAsync.dll!_03_ 03 - SyncOverAsync.dll!_03_ [Native to Managed Transiti IManaged to Native Transiti	system.Lazy <system. System.Lazy<system. System.Lazy<system. SyncOverAsync.Ca SyncOverAsync.Ca ion]</system. </system. </system. 	Collections.Generic.List Canon>.Value.get() Li ache.DirectoryCache.File ache.FileCountTests.Hase	<pre>sting>>.createValue() Line 43 ne 505 s.get() Line 16 OneFile() Line 33</pre>	3 C# C# C# C#	日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日日	1032 0 10596 0 1268 1 6040 1 7372 1	3 4 5	۷ ۵ ۵ ۵ ۵ ۵ ۵ ۹ ۵ ۹ ۹ ۵ ۹ ۹ ۹ ۹ ۹ ۹ ۹ ۹	Vorker Thread Vorker Thread Vorker Thread Vorker Thread Vorker Thread	<no name<br=""><no name<br="">Worker Th <no name<br=""><no name<="" td=""><td>e> <not a<br="">e> <not a<br="">read <not a<br="">e> < Syst e> < Syst</not></not></not></td><td>vailable> vailable> vailable> tem.Private. tem.Private.</td><td>CoreLib.dll!System CoreLib.dll!System</td><td>.Threading.W .Threading.M</td><td>aitHandle.W anualResetE</td><td>/aitOneNoC ventSlim.W</td></no></no></no></no>	e> <not a<br="">e> <not a<br="">read <not a<br="">e> < Syst e> < Syst</not></not></not>	vailable> vailable> vailable> tem.Private. tem.Private.	CoreLib.dll!System CoreLib.dll!System	.Threading.W .Threading.M	aitHandle.W anualResetE	/aitOneNoC ventSlim.W
Call Stack Breakpoints Exce	eption Settings Cor	mmand Window Imme	diate Window Output		Threads	a Autos	Locals Watch	1								

25

SynchronizationContext is not a purely UI thing...

Xunit relies on SynchronizationContext for limiting the number of running tests!

```
namespace Xunit.Sdk
{
    /// <summary>
    /// An implementation of <see cref="SynchronizationContext"/> which runs work on custom threads
    /// rather than in the thread pool, and limits the number of in-flight actions.
    /// </summary>
    public class MaxConcurrencySyncContext : SynchronizationContext, IDisposable
    {
        bool disposed = false;
        readonly ManualResetEvent terminate = new ManualResetEvent(false);
        readonly List<XunitWorkerThread> workerThreads;
        readonly ConcurrentQueue<Tuple<SendOrPostCallback, object, object>> workQueue = new Concurrent(
        readonly AutoResetEvent workReady = new AutoResetEvent(false);
    }
}
```

[assembly: CollectionBehavior(CollectionBehavior.CollectionPerClass, MaxParallelThreads = 2)]

Detaching the sync context

DirectoryCache.cs +	X AsyncTaskMethodBuilderT.cs TaskContinuation.cs WaitHandle.cs	⇒ 🌣	🗘 Test Explorer 👎 🗙 📃 👻
🔄 03 - SyncOverAsync	✓ Image: View of the synchronic of the synch	- +	= 🕩 🕨 - 🕞 🌝 🖾 3 🕏 2 😣 0 🗣 1 📓 - 🍾 [語 田 田 ଊ - Search Test Explorer (Ct 🔎 -
19 🛱	public DirectoryCache()	^	Test Duration Traits Error Message
20	{	- I.	▲ ② 03 - SyncOverAsync (3) 6 sec
21	_files = new Lazy <list<string>>(() =></list<string>	- 1 C	▶ ① _03SyncOverAsync (1)
22	ReadFilesAsync().GetAwaiter().GetResult());		▲ ⊘ _03SyncOverAsync.Cache (2) 6 sec
23	}		▷ 🤗 FileCountTests (1) 3 sec
24			▷ 🖉 FileNameTests (1) 3 sec
25 🗉	<pre>private static async lask<list<string>> ReadFilesAsync() </list<string></pre>		
26			
27	await Task.Delay(3 * 1000);		
28	i recurn new List <string> { "42.txt" };</string>		
29	J		
21 SC 1			
22 🗆 n	ublic class FileCountTests		
32 4		- 12	
34	[Fact]		
35	public void HasOneFile()		
36	{		
37	SynchronizationContext.SetSynchronizationContext(null);		
38	Assert.Single(new DirectoryCache().Files);		
39	}		
40 }			
41			
42 ⊟p	ublic class FileNameTests		
43 {			
44	[Fact]		
45 🖻	<pre>public void FirstFileIs42txt()</pre>		
46			
47	SynchronizationContext.SetSynchronizationContext(null);		
48	Assert.Equal("42.txt", new DirectoryCache().Files.First());		
49	}		
50 [}			

How to fix the issue properly?

- Stop using blocking operations.
 - Preferred, but not always possible.
- Stop using xUnit © (not really!) or change its concurrency limits!
 - The same issue may happen elsewhere as well.
- Use ConfigureAwait(false)
 - Won't fix all the possible issues.
 - A reasonable first step.



What is going on at runtime?





Lessons learned

- This is based on a real issue.
- Parallel Tasks is an invaluable tool for figuring out issues in async code.
- Understanding of how async works is very important.
- Always use 'ConfigureAwait(false)' in the library code.
- Always use 'ConfigureAwait(false)' if the code will be used with Sync Contexts.
- Consider using 'ConfigureAwait(false)' for service code as well.
- Sync Contexts can be used outside the UI world.



What is TaskCompletionSource?

- TCS allows controlling Task's lifetime manually
- A tool for implementing async API based on non-task-based implementations
- Core building block for producer-consumer queues, communication protocols etc.



A classical usage of TaskCompletionSource

- A simple work item queue
- Based on BlockingCollection<T> and TaskCompletionSource<T>

```
public class WorkItemQueue
     private record WorkItem(string Command, TaskCompletionSource<string> Result);
     private readonly BlockingCollection<WorkItem> _queue = new ();
     public Task ProcessItemsTask { get; }
     public WorkItemQueue() => ProcessItemsTask = Task.Run(ProcessWorkItems);
     public async Task ProcessCommandAsync(string command)
         var tcs = new TaskCompletionSource<string>();
         _queue.Add(new WorkItem(command, tcs));
         await tcs.Task;
     private async Task ProcessWorkItems()
         foreach (var item in _queue.GetConsumingEnumerable())
             // Processing the command.
             await Task.Delay(42*1000);
             item.Result.SetResult("Ok");
```

Debugging an issue with TaskCompletionSource

ubblic async Task ProcessOneCommand() var facade = new WorkItemQueue(); await facade.ProcessCommandAsync("Command 1") 277 278 279 280 280 290 280 290 280 290 280 290 280 290 290 290 290 290 290 290 290 290 29
<pre>var facade = new WorkItemQueue(); await facade.ProcessCommandAsync("Command 1") # Mixeellaneous Files * * ********************************</pre>
299 I' ' ' ' ' ' ' ' ' ' ' ' ' ' ' ' ' ' '

33

Async stacks to the rescue! (Again!)

WorkItemQueue.cs + X WaitHandle.cs TaskCompletionSource_T.cs SocketPal.Windows.cs	⇒ \$	Parallel Stacks 7 X
🗄 04 - TaskCompletionSource 🔹 🛠_04_TaskCompletionSource.WorkItemC 🝷 😚 ProcessOneCommand()	→ ‡	Search 🖉 - 🔶 View:
<pre>23 var tcs = new TaskCompletionSource<string>();</string></pre>	A	1 Async Logical Stack
24 _queue.Add(new WorkItem(command, tcs));		[Async] Task [Promise]
25 awart CCS.Task;		WorkItemQueue.ProcessCommandAsync
27		➡ WorkItemQueueTests.ProcessOneCommand
28 📄 private async Task ProcessWorkItems()	- 11	iestinvoker<>.invoke iestivietnoaAsync.Anon
	- 11	ExecutionTimer.AggregateAsync
30 E foreach (Var item in _queue.GetConsumingEnumerable())		ExceptionAggregator.RunAsync
32 // Processing the command.		TestInvoker<>.InvokeTestMethodAsync
33 await Task.Delay(42*1000);		TestInvoker<>.RunAsync.AnonymousMethod
34 item.Result.SetResult("Ok");		ExceptionAggregator.RunAsync<>
		XunitTestRunner.InvokeTestAsync
30 5		ExceptionAggregator.RunAsync<>
38		TestRunner<>.RunAsync
39 ⊟public class WorkItemQueueTests		TestCaseRunner<>.RunAsync
		TestMethodRunner<>.RunTestCasesAsync
μ_{1} [ract] μ_{2} μ_{2} [ract]		TestMethodRunner<>.RunAsync
43		TestClassRunner<>.RunTestMethodsAsync
44 var facade = new WorkItemQueue();		TestClassRunner<>.RunAsync
45 await facade.ProcessCommandAsync("Command 1");		TestCollectionRunner<>.RunTestClassesAsync
	—	TestCollectionRunner<>.RunAsync
99 % 👻 🕺 0 🛕 1 ↑ \downarrow 🕉 🕶 🔳 🚺 🕨 🕨 🕨 En: 46 Ch: 5 SP	C CRLF	
Call Stack		- ₽ ×
Name		Langi 📥
➡ [Async] System.Private.CoreLib.dll!Task (string) [Promise]		C#
[Async] 04 - TaskCompletionSource.dll!_04_TaskCompletionSource.WorkItemQueue.ProcessCommandAsync(string command) Li	ne 26	C#
S [Async] 04 - TaskCompletionSource.dll!_04_TaskCompletionSource.WorkItemQueueTests.ProcessOneCommand() Line 46	101: 20	C#
[Async] xunit.execution.dotnet.dll:Xunit.Sdk.lestInvoker <xunit.sdk.ixunit.lestcase>.Invoke.lestMethodAsync.AnonymousMethod</xunit.sdk.ixunit.lestcase>	_1() Line 26 ction) Line 4	68 C#
[Async] xunit.core.dll!Xunit.Sdk.ExceptionAggregator.RunAsync(System.Func <system.threading.tasks.task> code) Line 91</system.threading.tasks.task>		C#
$[Async] \ xunit.execution.dotnet.dll!Xunit.Sdk.TestInvoker < Xunit.Sdk.IXunitTestCase > .InvokeTestMethodAsync(object \ testClassInstance) \ and \ testClassInstance \ testClassInstance$	ance) Line 27	76 C#
[Async] xunit.execution.dotnet.dll!Xunit.Sdk.TestInvoker <xunit.sdk.ixunittestcase>.RunAsync.AnonymousMethod_47_0() Line 20</xunit.sdk.ixunittestcase>	08	C#
[Async] xunit.core.dlll.Xunit.Sdk.ExceptionAggregator.RunAsync <decimal>(System.Func<system.threading.tasks.task<decimal>></system.threading.tasks.task<decimal></decimal>	> code) Line	c#
[Async] xunit.core.dll!Xunit.Sdk.ExceptionAggregator.RunAsync <system.tuple<decimal, string="">>(System.Func<system.threading)< td=""><td>g.Tasks.Task</td><td><<system.tuple<decimal, string="">>> code) Line 109 C#</system.tuple<decimal,></td></system.threading)<></system.tuple<decimal,>	g.Tasks.Task	< <system.tuple<decimal, string="">>> code) Line 109 C#</system.tuple<decimal,>
Call Stack Breakpoints Exception Settings Command Window Immediate Window Output		

Let's debug it! Stack traces are not helpful;)

WorkItemQueue.cs + X WaitHandle.cs TaskCompletionSource_T.cs SocketPal.Windows.cs	⇒ 🌣	Parallel S	Stacks 🕈 🗙				•
🗊 04 - TaskCompletionSource 🔹 🔩 _04TaskCompletionSource.WorkItemC 🝷 😪 ProcessWorkItems()	→ ‡	Search		₽ - ∧ .	View: Tasks	- P 5 82	
<pre>23 var tcs = new TaskCompletionSource<string>(); 24 _queue.Add(new WorkItem(command, tcs)); 25 await tcs.Task; 26 } 27 28 E private async Task ProcessWorkItems() 29 { 30 E foreach (var item in _queue.GetConsumingEnumerable())</string></pre>			©1 A WaitHandle.Wai MessageBus.Rep XunitWorkerThr	sync Logical Stack tOneNoCheck porterWorker ead.QueueUserWork	Item.Anony		
<pre>31 { 32 // Processing the command. await Task.Delay(42*1000); 33 // item.Result.SetResult("0k"); 35 // item.Result.SetResult("0k"); 36 // 37 // 38 39 = public class WorkItemQueueTests 40 { </pre>	 	2 	©2 As WaitHandle.Wai MaxConcurrenc XunitWorkerThr	sync Logical Stacks tMultiple ySyncContext.Worke ead.QueueUserWork	rThreadProc Item.Anony	1 Async Logical S Task.Delay ♥ WorkItemQueue.Proce Task.Run	5tack essWorkItems
41 [Fact] 42 public async Task ProcessOneCommand() 43 { 44 var facade = new WorkItemQueue(); 45 await facade.ProcessCommandAsync("Command 1"); 46 } 99 % ♀ ♀ ▶ Ln: 34 Ch: 13 SPC	CRLF						
Call Stack	- 4 ×	Threads					- ₽ ×
Name	Langi ≜ C#	Search	ب ج	🖬 🗕 🕞 Group by:	Process ID	🝷 Columns 👻 🖽 E	
[Async] 04 - TaskCompletionSource.dll!_04_TaskCompletionSource.WorkItemQueue.ProcessWorkItems() Line 34	C#		ID Managed ID	Category	Name	Location	<u> </u>
[Async] System.Private.CoreLib.dll!System.Threading.Tasks.Task.Run	C#	○日本の1000000000000000000000000000000000000	25996 1 27164 0 52000 13 20972 14 40956 15 40124 16 18272 17	➢ Main Thread Ø Worker Thread Ø Worker Thread Ø Worker Thread Ø Worker Thread Ø Worker Thread Ø Worker Thread	Main Thread <no name=""> Worker Thread Worker Thread <no name=""> <no name=""> <no name=""></no></no></no></no>	 System.Private.CoreLib.dll!Sy <not available=""></not> <not available=""></not> System.Private.CoreLib.dll!Sy System.Private.CoreLib.dll!Sy System.Private.CoreLib.dll!Sy 	ystem.Threading.Mi ystem.Threading.Wi ystem.Threading.Wi ystem.Threading.Wi
Call Stack Breakpoints Exception Settings Command Window Immediate Window Output		Threads	Autos Locals Watch				

Let's add one more test! Shall we?

```
private async Task ProcessWorkItems()
         foreach (var item in _queue.GetConsumingEnumerable())
             // Processing the command.
             await Task.Delay(42);
             item.Result.SetResult("Ok");
public class WorkItemQueueTests
     [Fact]
     public async Task ProcessTwoCommands()
         var queue = new WorkItemQueue();
         await queue.ProcessCommandAsync("Command 1");
         queue.ProcessCommandAsync("Command 2").GetAwaiter().GetResu
```

Test Explorer 👎 🗙	
▶ ▶ • 🕞 🇞 🖾 2 🛇 2 😣 0 月	- ┣4 [☷ 田 田 ଊ -
Test	Duration Traits I
🔺 🥪 04 - TaskCompletionSource (2)	152 ms
🔺 🥪 _04TaskCompletionSource (2)	152 ms
🖌 Ϛ WorkItemQueueTests (2)	152 ms
Ø ProcessOneCommand	54 ms
✓ ProcessTwoCommands	98 ms



Production after deploying a new version



Why the tests were fine?

private async Task ProcessWorkItems()

foreach (var item in _queue.GetConsumingEnumerable())

// Processing the command.
await Task.Delay(42);



- 14 [= 🕂	- 錼 -
Duration	Traits E
159 ms	
159 ms	
159 ms	
54 ms	
105 ms	
	 ► ► ▲ Duration 159 ms 159 ms 159 ms 54 ms 54 ms 105 ms



What is going on at runtime?





40

Parallel Stacks to the rescue!

WorkItemQueue.cs + X MessageHandler.cs WaitHandle.cs ManualResetEventSlim.cs	⇒ \$	🕈 Parallel Stacks 7 🗙	-
🖬 04 - TaskCompletionSource 🔹 🖓_04_TaskCompletionSource.WorkIter 🔹 😚 ProcessTwoCommands()	- +	= Search ・ P 🗊 🎦 🖓 🖓 🗎	
28 📋 private async Task ProcessWorkItems()		1 Async Logical Stack	
29 {		Async] Task [Promise]	🛇 Wai –
30 E foreach (Var item in _queue.GetConsumingEnumerable())		WorkItemQueue ProcessCommandAsync	
32 // Processing the command.		WorklemQueueTests ProcessTwoCommande	
33 await Task.Delay(42);	- E	WorkitemQueueresis.ProcessiwoCommands	
<pre>34 item.Result.SetResult("Ok");</pre>	- E		
35	- HE	1 Async Logical Stack	
		TestInvoker<> InvokeTestMethodAsync Anon	
37 LS 38		ExecutionTimer AggregateAsync	O Wai
39 ∎public class WorkItemQueueTests		Execution Inter-Aggregater Sync	
40 {			
41 [Fact]		Testinvoker<>.invokerestinethodAsync	
42 e public async Task ProcessTwoCommands()		Iestinvoker<>.RunAsync.AnonymousMethod	
43 1 1 1 43 43 44 44 44 44 44 44 44 44 44 44 44	11 A 11	ExceptionAggregator.RunAsync<>	
45 SynchronizationContext.SetSynchronizationContext(null):		XunitTestRunner.InvokeTestAsync	
46		1 Async Logical Stack ExceptionAggregator.RunAsync<>	
47 var queue = new WorkItemQueue();		TestRunner<>.RunAsync	
48 await queue.ProcessCommandAsync("Command 1");		WorkItemQueue ProcessWorkItems	
<pre>49% queue.ProcessCommandAsync("Command 2").GetAwaiter().GetResult();</pre>		TestMethodRunner<>.RunTestCasesAsync	
50 3		TestMethodRunner<>.RunAsync	
52	-	TestClassRunner<>.RunTestMethodsAsync	•
99 % 🔻 🖓 🥝 No issues found 🛛 😽 🔻 🖣 🚺 🕨 🕨 🕨 Ln: 49 Ch: 9 S	SPC CRLF		▶ 🗖
Call Stack		Threads	- ₽ ×
Name	Langı 🔺	Search 🔎 - 📄 🖪 🖬 - 🕛 Group by: Process ID - Columns - 🖽 🗗 🗐	
$System. Private. Core Lib. dll! System. Runtime. Compiler Services. Task Awaiter. Output Wait Etw Events. An onymous Method_12_0 and the service of the se$)(S C#		
System.Private.CoreLib.dll!System.Runtime.CompilerServices.AsyncMethodBuilderCore.ContinuationWrapper.Invoke() Line	e C#	ID Managed ID Category Name Location	Provide A constraints
System Private CoreLib.dll!System. Threading. Tasks Task Pup Continuation. RunOrScheduleAction(System.Action action, boo	ol C#	Image: Private CoreLib.dil:System. Inread Image: Private Private CoreLib.dil:System. Inread Image: Private Pri	ling.Manuali
System Private CoreLib dillSystem Threading Tasks Task FinishContinuations(Object ContinuationObject) Line 5229	C#	P 35860 1 ∞ Main Thread <inot available=""> P 35860 1 ∞ Main Thread Main Thread System Private Corol in dill/System Thread</inot>	ding Manuall
System. Private.CoreLib.dll!System.Threading.Tasks.Task <system.threading.tasks.voidtaskresult>.TrySetResult(System.Threading.Tasks.VoidTaskResult>.TrySetResult(System.Threading.Tasks.Task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.tasks.task<system.tasks.task<system.tasks.task<system.tasks.task<system.tasks.task<system.tasks.task<system.tasks.task<system.tasks.task<system.tasks.tasks.tasks.task<system.tasks.tas< td=""><td>ea C#</td><td>P 34212 6 Ø Worker Thread Worker Thread System Not Sockets dillSystem Not Sockets</td><td></td></system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.threading.tasks.task<system.tasks.task<system.tasks.task<system.tasks.task<system.tasks.task<system.tasks.task<system.tasks.task<system.tasks.task<system.tasks.task<system.tasks.tasks.tasks.task<system.tasks.tas<></system.threading.tasks.voidtaskresult>	ea C#	P 34212 6 Ø Worker Thread Worker Thread System Not Sockets dillSystem Not Sockets	
System.Private.CoreLib.dll!System.Runtime.CompilerServices.AsyncTaskMethodBuilder <system.threading.tasks.voidtaskr< td=""><td>Re C#</td><td>P → 38288 9 Worker Thread Worker Thread Vorker Thread System Private Corol in dll/System Thread</td><td>ting Manual</td></system.threading.tasks.voidtaskr<>	Re C#	P → 38288 9 Worker Thread Worker Thread Vorker Thread System Private Corol in dll/System Thread	ting Manual
System.Private.CoreLib.dll!System.Runtime.CompilerServices.AsyncTaskMethodBuilder.SetResult() Line 109	C#	P 46004 0 @ Worker Thread <no name=""> <not available=""></not></no>	
[Completed] 04 - TaskCompletionSource.dll!_04_TaskCompletionSource.WorkItemQueue.ProcessCommandAsync(string of	co C#	P 28520 0 B Worker Thread <no name=""> <not available=""></not></no>	
System.Private.CoreLib.dll!System.Runtime.CompilerServices.AsynclaskMethodBuilder <system.threading.tasks.voidtaskr< td=""><td>(e C#</td><td>P 16284 14</td><td>ding WaitHar 🔻</td></system.threading.tasks.voidtaskr<>	(e C#	P 16284 14	ding WaitHar 🔻
	C C#		•
Call Stack Breakpoints Exception Settings Command Window Immediate Window Output		Threads Autos Locals Watch 1	

Parallel Stacks to the rescue!

WorkItemQueue.cs + X MessageHandler.cs WaitHandle.cs ManualResetEventSlim.cs	🕸 Parallel Stacks 🕂 🗙	-
🗐 04 - TaskCompletionSource 🔹 🕫 _04_TaskCompletionSource.WorkIter 🔹 😚 ProcessTwoCommands() 🔹 🔹	÷ Search	
28 📄 private async Task ProcessWorkItems()	A 1 Async Logical Stack	
$29 \qquad 1 \qquad 1 \qquad 20 \qquad 1 \qquad $	[Async] Task [Promise]	🛇 Wai 🚽
30 A Foreach (Var item in _queue.secconsumingEnumerable())	WorkItemOueue.ProcessCommandAsvnc	O Met
32 // Processing the command.	Work/ItemQueueTests ProcessTwoCommands	
33 await Task.Delay(42);		
34 item.Result.SetResult("Ok");		
	1 Async Logical Stack	
30 30	TestInvoker<>.InvokeTestMethodAsync.Anon	
38	ExecutionTimer AggregateAsync	🛇 Wai
39 ⊟public class WorkItemQueueTests	ExceptionAggregator RunAsync	O Max
40 {	Testinyoker<> InvokeTestMethodAsvac	🛇 Xun
41 [Fact]	Testinvoker <> Invoker <> Invoker <> Approximation Sync	
44 // Mimic the real world case: removing the sync context	VunitTestPunner InvokeTestAcune	
45 SynchronizationContext.SetSynchronizationContext(null);		
46	1 Async Logical Stack	
47 var queue = new WorkItemQueue();	TaskCompletionSource <>.SetResult	
48 await queue. ProcessCommandAsync("Command 2"), GetAwaiter(), GetResult():	WorkItemQueue.ProcessWorkItems	
50	Task.Run Task.Run	
51	TestMethodRunner<>.RunAsync	
52	▼ TestClassRunner<>.RunTestMethodsAsync	•
99 % 🔻 🥨 🥝 No issues found 🔰 🕅 🔻 🖣 👘 🕨 👘 Ln: 49 Ch: 9 SPC CF	F 4	
Call Stack	K Threads	• ¶ ×
Name	🔺 Search 🖉 🗭 🛱 🖡 🖡 Group by: Process ID 🔹 Columns 🕶 🖽 🕀 🕀	11 🕨
System.Private.CoreLib.dlllSystem.Threading.ManualResetEventSlim.Wait(int millisecondsTimeout, System.Threading.Cancell C#	ID Managed ID Category Name Location	
System. Private.CoreLib.dll:System. Threading.Tasks.Task.Spiritrenblocking.wait(int millisecondsTimeout, System.Threading.Cancell	Image: Provide and the second sec	Manuall
System.Private.CoreLib.dlllSystem.Runtime.CompilerServices.TaskAwaiter.HandleNonSuccessAndDebuggerNotification(Syst C#	P 51496 0	
System.Private.CoreLib.dll!System.Runtime.CompilerServices.TaskAwaiter.GetResult() Line 105 C#	👎 35860 1 📈 Main Thread Main Thread 🗸 System.Private.CoreLib.dll!System.Threading.	Manuall
[waiting on Async Operation, double-click or press enter to view Async Call Stacks]	🖻 34212 6 🚳 Worker Thread Worker Thread 🗸 System.Net.Sockets.dll!System.Net.Sockets.dl	ocketPa
VIA - TaskCompletionSource.dll!_04_TaskCompletionSource.WorkItemQueueTests.ProcessTwoCommands() Line 49 C#	P 💠 38288 9 🛷 Worker Thread Worker Thread 🗸 System.Private.CoreLib.dll!System.Threading.l	Manuall
System Private.CoreLib.dll!System.Runtime.CompilerServices.AsvncTaskMethodBuilder <system.threading.tasks.voidtaskre c#<="" td=""><td>P 46004 0 Ø Worker Thread <no name=""> <not available=""></not></no></td><td></td></system.threading.tasks.voidtaskre>	P 46004 0 Ø Worker Thread <no name=""> <not available=""></not></no>	
System.Private.CoreLib.dll!System.Threading.ExecutionContext.RunInternal(System.Threading.ExecutionContext executionC C#	H 28520 0 Ø Worker Thread <no name=""> <not available=""></not></no>	
$System. Private. Core Lib. dll! System. Runtime. Compiler Services. A syncTask Method Builder < System. Threading. Tasks. Void Task Re\ C\# C = C + C + C + C + C + C + C + C + C +$	▼ 16284 14 @V Worker Thread <no name=""> V System Private Corel ib dllSystem Threading V</no>	WaitHar
Call Stack Breakpoints Exception Settings Command Window Immediate Window Output	Threads Autos Locals Watch 1	

How to solve the problem?

- Don't block async code (maybe easier said then done!)
- Run TaskCompletionSource's continuations asynchronously (.NET Framework 4.6+)!
- Consider using a wrapper or a helper for running continuations asynchronously

```
public readonly struct TaskSourceSlim<TResult>
{
    private readonly TaskCompletionSource<TResult> m_tcs;
    internal TaskSourceSlim(bool dummy)
        : this()
        {
            m_tcs = new TaskCompletionSource<TResult>(
                TaskCreationOptions.RunContinuationsAsynchronously);
        }
        public Task<TResult> Task => m_tcs.Task;
        // SetResult, SetException, SetCanceled and other members
        public static class TaskSourceSlim
        {
            public static TaskSourceSlim
```

⊟public class WorkItemQueue

```
private record WorkItem(string Command, TaskCompletionSource<string> Result);
private readonly BlockingCollection<WorkItem> _queue = new ();
```

```
public Task ProcessItemsTask { get; }
```

```
public WorkItemQueue() => ProcessItemsTask = Task.Run(ProcessWorkItems);
```

```
public async Task ProcessCommandAsync(string command)
```

```
var tcs = new TaskCompletionSource<string>(
    // This is a very important change!
    // It will force the continuations to ran asynchronously!
    TaskCreationOptions.RunContinuationsAsynchronously);
```

```
_queue.Add(new WorkItem(command, tcs));
await tcs.Task;
```

```
private async Task ProcessWorkItems()
```

```
foreach (var item in _queue.GetConsumingEnumerable())
```

```
// Processing the command.
await Task.Delay(42);
item.Result.SetResult("Ok");
```

Let's check it!

WorkItemQueue.cs + X WaitHandle.cs MessageHandler.cs ManualResetEventSlim.cs	Future.cs	, ⇒	Test Explorer 👎 🗙	
🖅 04 - TaskCompletionSource 🔹 🕫 _04_TaskCompletionSource.WorkItemQı 👻 😪 Process	VorkItems() -	÷	▶ ▶ • 🕑 🗞 🖾 2 🕗 2 😣 0	£] • ™ [≣ 🗖 🗖
			Test	Duration Traits
19 public workitemqueue() => Processitemstask = Task.kun(Proces	SWOIRICEMS);		✓ 04 - TaskCompletionSource (2)	153 ms
20 public async Task ProcessCommandAsync(string command)			🔺 🥝 _04TaskCompletionSource (2)	153 ms
21 {			🔺 🥪 WorkItemQueueTests (2)	153 ms
22 var tcs = new TaskCompletionSource <string>(</string>			ProcessOneCommand	54 ms
23 // This is a very important change: 20 // It will force the continuations to ran asynchrono	us]v!		🖌 ProcessTwoCommands	99 ms
25 TaskCreationOptions.RunContinuationsAsynchronously);	us cy.		ProcessTwoCommands	
26				
27 _queue.Add(new WorkItem(command, tcs));				
28 await tcs.Task;				
30				
31 🖃 private async Task ProcessWorkItems()				
32 {				
33				
34 t				
await Task.Delav(42):				
<pre>37 item.Result.SetResult("0k");</pre>				
38				
39 }				
40 L}				
41 42 Epublic class WorkItemOueueTests		•		
43 {				
44 [Fact]				
45 public async Task ProcessTwoCommands()				
46 [i // Mimic the real world case: remeving the sync context				
47 48 SynchronizationContext.SetSynchronizationContext(null):			Test Detail Summany	
49				
50 var queue = new WorkItemQueue();				ltemQueueTests.ProcessT
51 await queue.ProcessCommandAsync("Command 1");			Source: <u>WorkItemQueue.cs</u>	ine 45
52 queue.ProcessCommandAsync("Command 2").GetAwaiter().GetF	esult();		Duration: 99 ms	

44

Lessons learned

- Synchronous behavior of TaskCompletionSource.SetResult is quite dangerous.
- Blocking can be hidden. 'GetConsumingEnumerable' is a blocking call.
- Always force async continuations when using TaskCompletionSource.
 - This will detach the caller from synchronously calling unknown code.
- Consider using a helper or a wrapper for that.



Thanks a lot, everyone!

[Bonus] Deadlock with two tasks

Deadlocks.cs* - 🗙 DatabaseFacade.cs	ManualResetEventSlim.cs TaskCompletionSource_T.cs	± ⇔	Parallel Sta	cks 7 X	
물 04 - TaskCompletionSource +	🕂 😚_04_TaskCompletionSource.Deadlocks 🛛 🖌 😚 SimpleTaskDeadlock()	- ‡	Search	・ ク・ ↓ View:	Tasks 🔹 🆻 🛱
10 [Fact]					
11 🗉 public async Task	SimpleTaskDeadlock()		11	1 Async Logical	Stack
12 {				Deadlocks.SimpleTaskDea	dlock.Anonymous
13 Synchronizatio	onContext.SetSynchronizationContext(null);		1	Task.Run	
14 15 var bothTasks(Created = new TaskCompletionSource <bool>()</bool>				<u></u>
16					4
17 Task? t2 = nu	u;		11	1 Async	Logical Stack
18 🖨 🛛 Task t1 = Task	k.Run(async () =>			🍽 Deadlocks.Simple	TaskDeadlock.Anonymous
19 i	hTacks(mated Tack)			Task.Run	
21 Console.W	riteLine("t1 waiting for t2");	1		4	
Solution State					
23 Console.W	riteLine(" t1 done.");			1 Async Logical Stack	
24 3);				Task.WhenAll	
26				Deadlocks.SimpleTaskDeadlock	
27 🗉 t2 = Task.Run	(async () =>			TestInvoker<>.InvokeTestMethodAsync.Anon	
28 {				ExecutionTimer.AggregateAsync	
29 await both	hTasksCreated.Task; mitoling("t2 waiting for t1 ");			ExceptionAggregator.RunAsync	
31 await t1:	riteline(tz waiting for ti),			TestInvoker<>.InvokeTestMethodAsync	
32 Console.W	riteLine(" t2 done.");			TestInvoker<>.RunAsync.AnonymousMethod	
33 });				ExceptionAggregator.RunAsync<>	
34 hothTacksCroot	tod SatDacult(true);			XunitTestRunner.InvokeTestAsync	
35 DOCHTASRSCIPAL	iced.secresucc(crue);			ExceptionAggregator.RunAsync<>	
37 await Task.Who	enAll(t1, t2);			TestRunner<>.RunAsync	
38				TestCaseRunner<>.RunAsync	
39				TestMethodRunner<>.RunTestCasesAsync	
40				TestMethodRunner<>.RunAsync	
42				TestClassRunner<>.RunTestMethodsAsync	
43				TestClassRunner<>.RunAsync	
44				TestCollectionRunner<>.RunTestClassesAsync	
45				TestCollectionRunner<>.RunAsync	
47				Task.Run	
48				XunitTestAssemblyRunner.RunTestCollections	
49				TestAssemblyRunner<>.RunAsync	
50				XunitTestFrameworkExecutor.RunTestCases	
52					

[Bonus] Deadlock a semaphore



	1 Async Logical Stack
	SemaphoreSlim.WaitAsync
P	Deadlocks.SemaphoreDeadlock.Anonymousl
	Task.Run
	Deadlocks.SemaphoreDeadlock.Anonymous
	Task.Run
	Deadlocks.SemaphoreDeadlock
	TestInvoker<>.InvokeTestMethodAsync.Anon
	ExecutionTimer.AggregateAsync
	ExceptionAggregator.RunAsync
	TestInvoker<>.InvokeTestMethodAsync
	TestInvoker<>.RunAsync.AnonymousMethod
	ExceptionAggregator.RunAsync<>
	XunitTestRunner.InvokeTestAsync
	ExceptionAggregator.RunAsync<>
	TestRunner<>.RunAsync
	TestCaseRunner<>.RunAsync
	TestMethodRunner<>.RunTestCasesAsync
	TestMethodRunner<>.RunAsync
	TestClassRunner<>.RunTestMethodsAsync
	TestClassRunner<>.RunAsync
	TestCollectionRunner<>.RunTestClassesAsync
	TestCollectionRunner<>.RunAsync
	Task.Run
	XunitTestAssemblyRunner.RunTestCollections
	TestAssemblyRunner<>.RunAsync
	XunitTestFrameworkExecutor.RunTestCases

[Bonus] Deadlock with yield return and lock

```
[Fact]
public async Task YieldReturnDeadlock()
    object _syncObj = new object();
    foreach (var i in GetEnum())
        await UseAsync(i).ConfigureAwait(false);
    IEnumerable<int> GetEnum()
        lock (_syncObj)
            yield return GetValue(0);
            yield return GetValue(1);
    int GetValue(int value)
        lock (_syncObj)
            return value + 1;
    Task UseAsync(int i) => Task.Delay(i + 100);
```

♦ 1 Async Logical Stack
S [Waiting on lock owned by Thread 31824, doub
SieldReturnDeadlockGetValue 1
VieldReturnDeadlockGetEnum 0
O Deadlocks.YieldReturnDeadlock
TestInvoker<>.InvokeTestMethodAsync.Ano
ExecutionTimer.AggregateAsync
ExceptionAggregator.RunAsync
TestInvoker<>.InvokeTestMethodAsync
TestInvoker<>.RunAsync.AnonymousMetho
ExceptionAggregator.RunAsync<>
XunitTestRunner.InvokeTestAsync
ExceptionAggregator.RunAsync<>
TestRunner<>.RunAsync
TestCaseRunner<>.RunAsync
TestMethodRunner<>.RunTestCasesAsync
TestMethodRunner<>.RunAsync
TestClassRunner<>.RunTestMethodsAsync
TestClassRunner<>.RunAsync
TestCollectionRunner<>.RunTestClassesAsync
TestCollectionRunner<>.RunAsync
Task.Run
XunitTestAssemblyRunner.RunTestCollection
TestAssemblyRunner<>.RunAsync
XunitTestFrameworkExecutor.RunTestCases