

Asynchronous and Parallel Programming in C#

Rasmus Lystrøm
Associate Professor
ITU
rne@itu.dk

The screenshot shows the Visual Studio Code editor with two C# files open: ProgramTests.cs and Program.cs. The left sidebar contains icons for Explorer, Search, Source Control, Run and Debug, Extensions, Test Explorer, Run and Debug, and Output. The bottom status bar shows 1 tests, 0 failures, 0 warnings, and the file path C:\HelloWorld.sln.

```
ProgramTests.cs
1 using System;
2 using System.IO;
3 using Xunit;
4
5 namespace HelloWorld.Tests
6 {
7     0 references | Run All Tests | Debug All Tests
8     public class ProgramTests
9     {
10         [Fact]
11         0 references | Run Test | Debug Test
12         public void Main_given_no_args_p
13         {
14             // Arrange
15             using var writer = new Strin
16             Console.SetOut(writer);
17
18             // Act
19             Program.Main(new string[0]);
20
21             // Assert
22             var output = writer.GetStrin
23             Assert.Equal("Hello World!",
24         }
25     }

```

```
Program.cs
1 using System;
2
3 namespace HelloWorld
4 {
5     1 reference
6     public class Program
7     {
8         1 reference
9         public static void Main(string[]
10         {
11             Console.WriteLine("Hello Wor
12         }
13     }

```

TERMINAL

```
1: pwsh
Loading personal and system profiles took 1008ms.
C:\HelloWorld> dotnet test
Test run for C:\HelloWorld\HelloWorld.Tests\bin\Debug\netcoreapp3.0\HelloWorld.Tests.dll(.NETCoreApp, Vers
ion=v3.0)
Microsoft (R) Test Execution Command Line Tool Version 16.3.0
Copyright (c) Microsoft Corporation. All rights reserved.

Starting test execution, please wait...

A total of 1 test files matched the specified pattern.

Test Run Successful.
Total tests: 1
Passed: 1
Total time: 3,4618 Seconds
C:\HelloWorld>

```

Agenda

Multithreading

Concurrency

Threads

Task Parallel Library

Asynchronous Programming

Async \neq Parallel \neq Threads

Multithreading

Multithreading

Enables executing several pieces of code simultaneously

Leverage multicore CPUs

Speed

The operating system decides the order

Concurrency

Concurrency

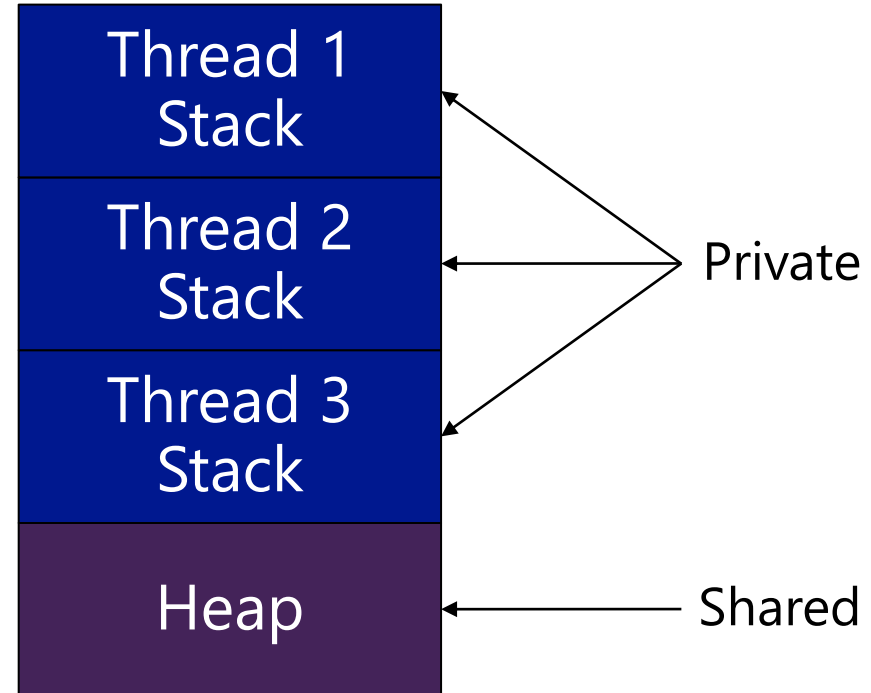
A property of systems in which several computations are executing **simultaneously**, and potentially interacting with each other. The computations may be executing on multiple cores in the same chip, preemptively time-shared threads on the same processor, or executed on physically separated processors.

Threads

Threads

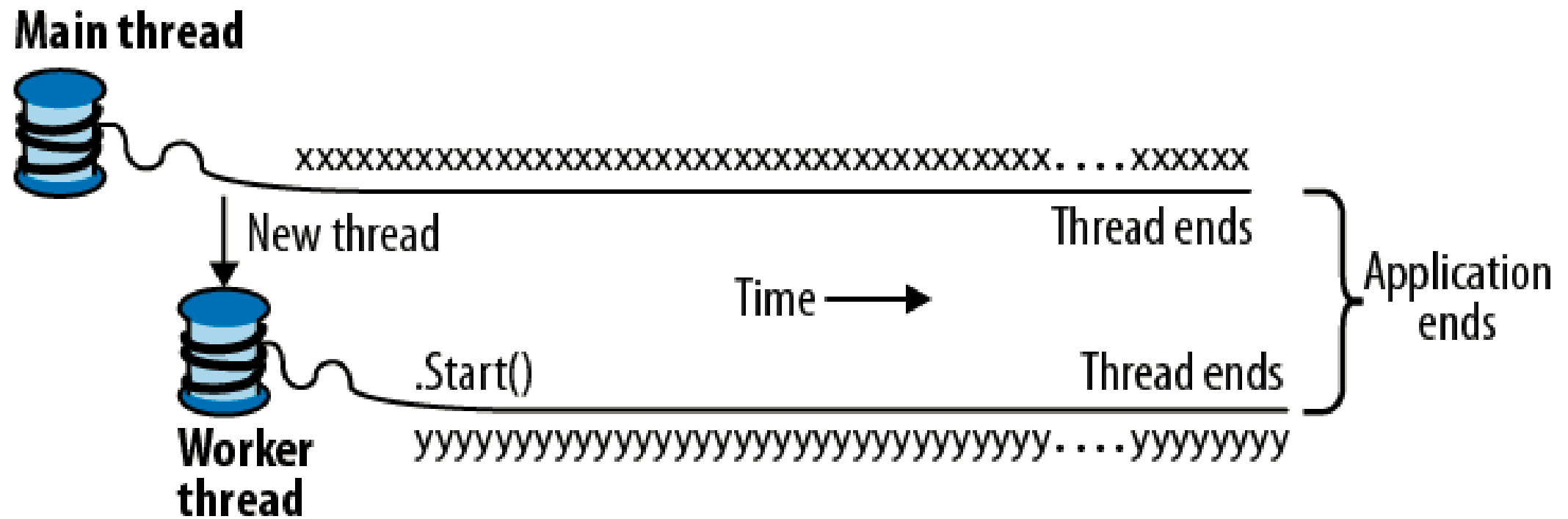


Single Threaded Program



Multithreaded Program

Threads Example



Threads

Demo

Race Condition



Race Condition

Behavior of a program where the output is **dependent** on the **sequence** or **timing** of other **uncontrollable** events.

→ Bug, when events do not happen in the order the programmer intended.

Race Condition

Demo

Deadlock



Deadlock

A situation in which two or more competing actions are each waiting for the other to finish, and thus neither ever does.

Deadlock

Demo

Task Parallel Library

Task.Run

Task.Factory...

Task.Delay

Parallel.For

Parallel.ForEach

Parallel.Invoke

Parallel Linq → .AsParallel()

Task Parallel Library

Demo

System.Collections.Concurrent

ConcurrentQueue<T>

ConcurrentStack<T>

BlockingCollection<T>

ConcurrentDictionary<TKey, TValue>

Asynchronous Programming

Asynchronous Programming

Asynchronous programming is a means of parallel **programming** in which a unit of work runs separately from the main application thread and notifies the calling thread of its completion, failure or progress.

async/await

async →

Method must return `void`, `Task`, `Task<T>`, or a task-like type. Specifically: a type, which satisfy the `async` pattern, meaning a `GetAwaiter` method must be accessible.

await → Await task(s)...

Note: `Main` and *test* methods must return `Task`

async/await

Demo

Speed
Multiprocessor
Parallel execution

Async ≠ Parallel ≠ Threads

Non-blocking UI,
background tasks,
asynchronous

Low-level building
block
Do not use directly!