Contents

[What is Multitasking? 3](#_Toc145884194)

[What is a thread? 3](#_Toc145884195)

[Thread class 3](#_Toc145884196)

[Current thread 3](#_Toc145884197)

[IsAlive 3](#_Toc145884198)

[Name 3](#_Toc145884199)

[Priority 3](#_Toc145884200)

[Interrupt 3](#_Toc145884201)

[Join 3](#_Toc145884202)

[Resume 4](#_Toc145884203)

[Sleep 4](#_Toc145884204)

[SpinWait 4](#_Toc145884205)

[Start 4](#_Toc145884206)

[Suspend 4](#_Toc145884207)

[Thread live cycle 4](#_Toc145884208)

[Protecting Shared resources 4](#_Toc145884209)

[Monitor.Wait 5](#_Toc145884210)

[Monitor.Pulse 5](#_Toc145884211)

[Monitor.PulseAll 5](#_Toc145884212)

[ManualResetEvent 5](#_Toc145884213)

[AutoResetEvent 5](#_Toc145884214)

[Mutex in multithreading 6](#_Toc145884215)

[Semaphore 6](#_Toc145884216)

[Dead Lock in Multithreads 7](#_Toc145884217)

[Thread Pool 7](#_Toc145884218)

[Asynchronous Programing 8](#_Toc145884219)

[Task based Asynchronous Pattern 8](#_Toc145884220)

[Thread vs Task 8](#_Toc145884221)

[Task ways to start 9](#_Toc145884222)

[Wait 9](#_Toc145884223)

[Continuation task 9](#_Toc145884224)

[Async and Await 10](#_Toc145884225)

[Parallel Programing 11](#_Toc145884226)

[Parallel for loop 11](#_Toc145884227)

[ParallelOptions 11](#_Toc145884228)

[Termination a Parallel Loop 12](#_Toc145884229)

Learn Multithreading/asynchronous and Parallel Programming in C# and .Net

# What is Multitasking?

Is the concept of running multiple applications at a time.

# What is a thread?

a thread is a lightweight process, thread by default does not have name.

# Thread class

Help us to perform tasks such as creating and setting the priority of a thread. We can use this class to control a thread and obtain status.

## Current thread

Retrieve the name of the thread which is currently running.

## IsAlive

Retrieve the thread status.

## Name

Retrieve the thread name.

## Priority

Indicates the scheduling priority of a thread. By default, is **normal**. We can assign **Highest**, **Above normal**, **normal**, **Below normal**, or **lowest** value to the priority property.

## Interrupt

To interrupt the thread which is in the **WaitSleepJoin** status.

## Join

To block a thread until another thread has terminated. The main thread await until the child thread finish.

## Resume

To resume a thread which has been suspended earlier.

## Sleep

To sleep a thread for a specific period of time.

## SpinWait

To make a tread wait the number of times specified in the iteration parameter.

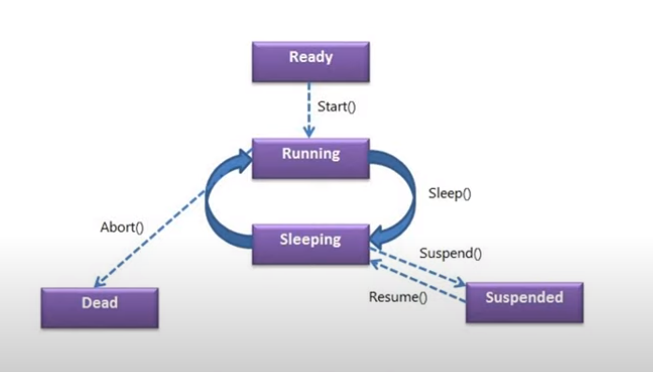
## Start

To start a tread.

## Suspend

To suspend a tread.

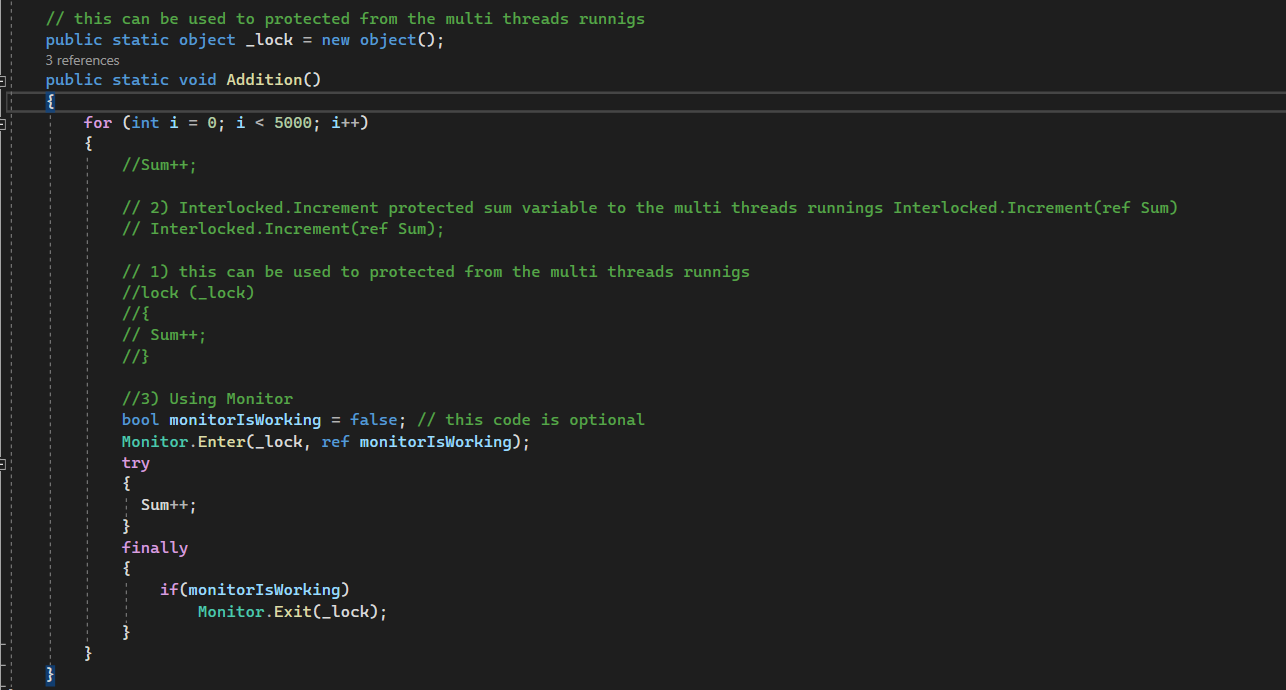
## Thread live cycle



## Protecting Shared resources

By using

Interlocked.Increment(), lock Method, monitor Method



### Monitor.Wait

Wait for other threads to notify.

### Monitor.Pulse

Notify to another thread.

### Monitor.PulseAll

Notifies all others threads within a process

## ManualResetEvent

Wait until uno tread write o get something to read or get something. **Is set order in treads.**

A screenshot of a computer

Description automatically generated

## AutoResetEvent

Is used for sending signals between two threads. Both Thread share the same autoResetEventObject. Work in order inside the threads. When one thread is complete another thread is allowed.

**Mutex is better than AutoResetEvent**

A screen shot of a computer

Description automatically generated

## Mutex in multithreading

Is a Synchronization primitive that grants exclusive access to the shared resource to only on thread.

If a Thread acquires a Mutex, the second thread that wants to acquire that Mutex is suspended until the first thread release the Mutex.

A screen shot of a black background

Description automatically generated

## Semaphore

Is used to limit the number of threads that can have access to a share resource concurrently. Allows one or more threads to enter into the critical section and execute the task concurrently with thread safety.

A screen shot of a computer

Description automatically generated

## Dead Lock in Multithreads

Is where two or more threads are unmoving or frozen in their execution because they are waiting for each other to finish.

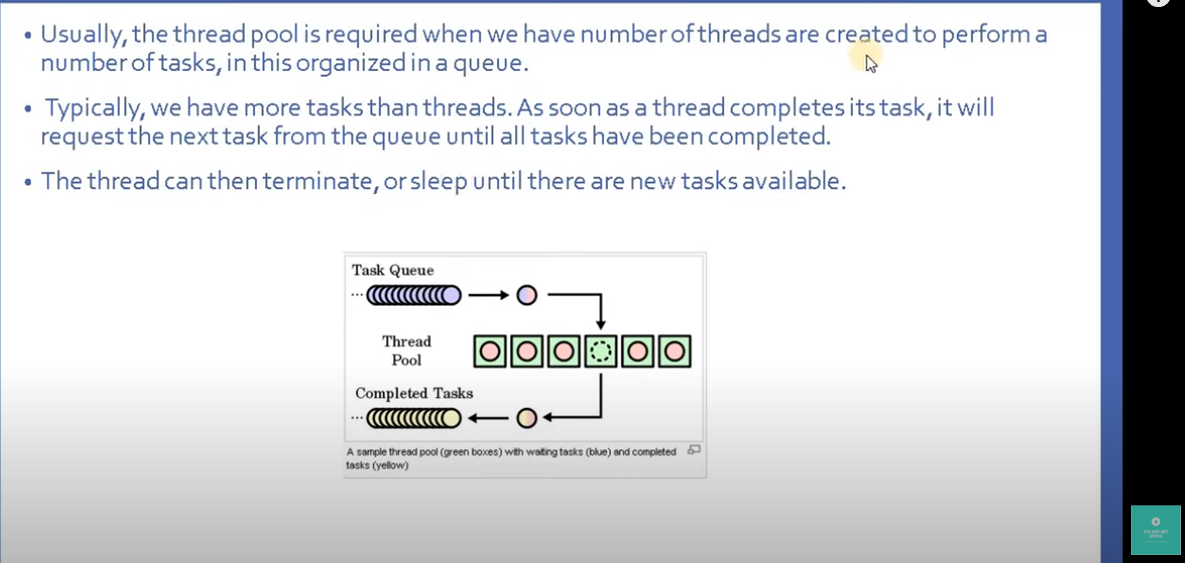
We can resolve Deadlock in multithreading by many ways, using by acquiring locks in a specific order.

Using monitor.TryEnter() or Mutex

# Thread Pool

Is the process of creating a collection of threads during the initialization of a multithread application, and then reusing those threads for new tasks as and when required, instead of creating new threads.

**This is the best way to create Threads because use computer resource better**



# Asynchronous Programing

Asynchronous programing is a process that starts happening together at the same time. Call waits for the method to complete before continuing with the program flow. It leaves the users to bad experiences with blocked UI.

## Task based Asynchronous Pattern

Is base on Task<ItaskResult>, it is typically execute asynchronously on a thread pool.

## Thread vs Task

Both are use for parallel programing.

Thread is low-level and Task is higher-level.

Thread should be preferred for any long-running operations while a Task should be preferred for any other asynchronous programming

Task can return result while there is not mechanism to return from thread.

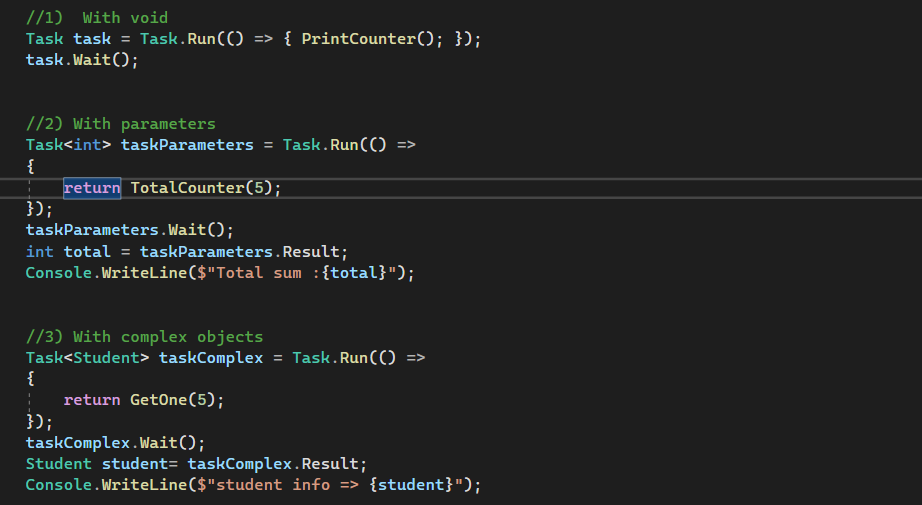
## Task ways to start

A screenshot of a computer program

Description automatically generated

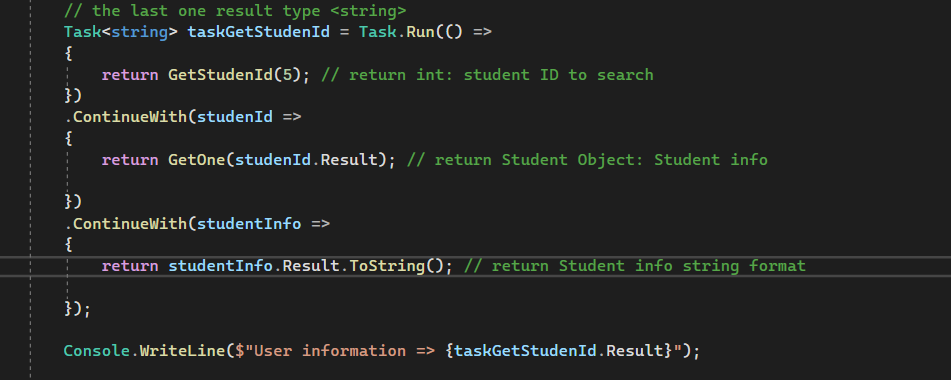
## Wait

The main thread is waiting until the child’s task is finished. Like joins in Threads.



## Continuation task

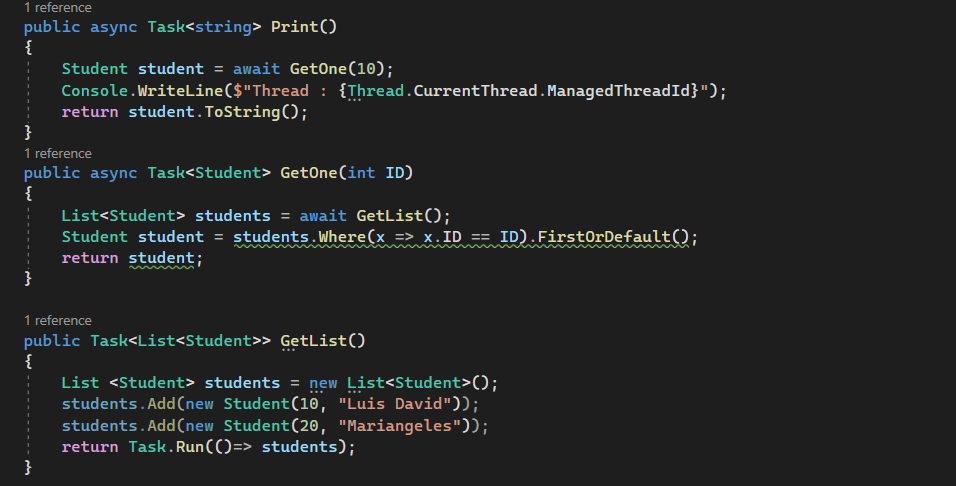
Is a task invoke by another task



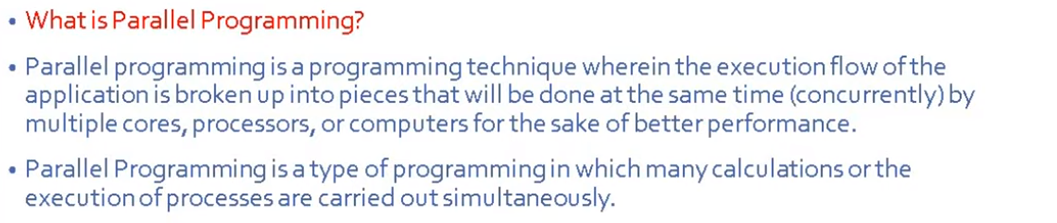
## Async and Await

**Async** turns a method into an asynchronous method, which allows you to use await keyword in its body. It does not freeze the UI.

**Await** suspends the calling method and yield control back to its caller until the await task is complete. It can be used only in async methods.

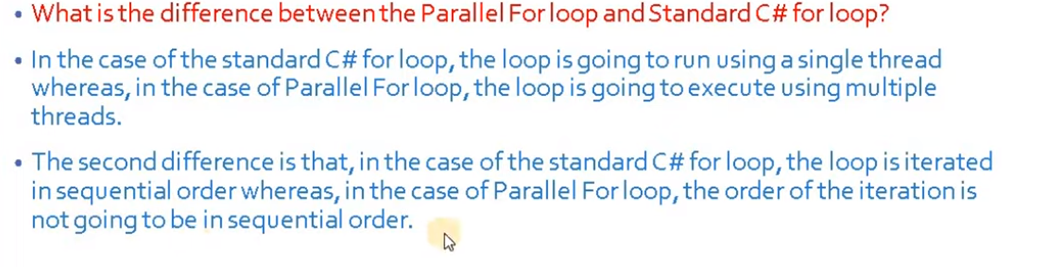


# Parallel Programing

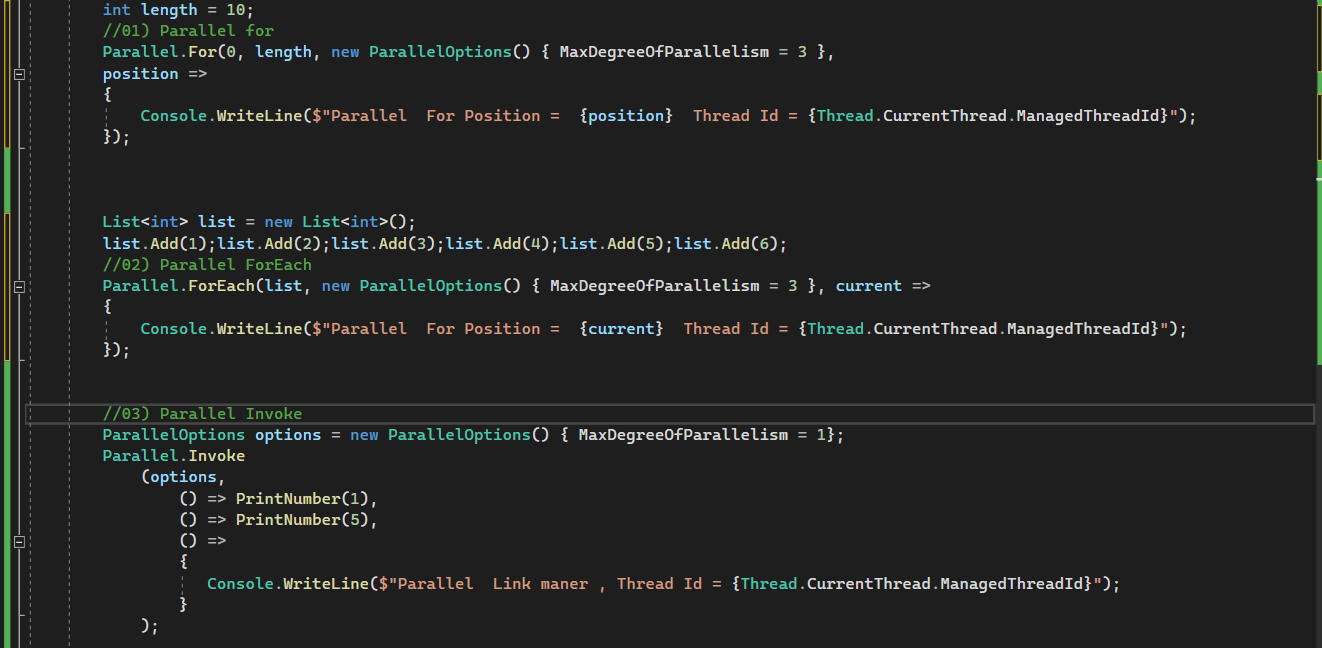


Running application y many computers’ cores.

## Parallel for loop



The main thread waits until the parallel for finish.



### ParallelOptions

This class provide options to limit the number of concurrently executions loops method by **MaxDegreeOfParallelism**.

## Termination a Parallel Loop

