

1 System.Threading.Interlocked Class

```
2 [ILAsm]  
3 .class public sealed Interlocked extends System.Object  
4 [C#]  
5 public sealed class Interlocked
```

6 Assembly Info:

- 7 • *Name:* mscorlib
- 8 • *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- 9 • *Version:* 2.0.x.x
- 10 • *Attributes:*
 - 11 ○ CLSCompliantAttribute(true)

12 Summary

13 The System.Threading.Interlocked class provides atomic operations for variables that
14 are shared by multiple threads.

15 Inherits From: System.Object

16

17 **Library:** BCL

18

19 **Thread Safety:** All public static members of this type are safe for multithreaded operations.
20 No instance members are guaranteed to be thread safe.

21

22 Description

23 The System.Threading.Interlocked methods protect against errors that can occur
24 when the scheduler switches contexts while a thread is updating a variable that can be
25 accessed by other threads. The members of this class do not throw exceptions.

26

27 [*Note:* The System.Threading.Interlocked.Increment method and its counterpart,
28 System.Threading.Interlocked.Decrement, increment or decrement a variable and
29 store the resulting value, as an atomic operation.

30

31 The System.Threading.Interlocked.Exchange method atomically exchanges the
32 values of the specified variables. The
33 System.Threading.Interlocked.CompareExchange method provides an atomic
34 operation that compares two values and stores a third value in one of the variables,
35 based on the outcome of the comparison.

36

37]

38

1
2 **Interlocked.CompareExchange(System.Int32**
3 **&, System.Int32, System.Int32) Method**

```
4 [ILAsm]  
5 .method public hidebysig static int32 CompareExchange(int32& location1,  
6 int32 value, int32 comparand)  
  
7 [C#]  
8 public static int CompareExchange(ref int location1, int value, int  
9 comparand)
```

10 **Summary**

11 Compares two *System.Int32* values for equality and stores a specified value if they are
12 equal.

13 **Parameters**

Parameter	Description
<i>location1</i>	A <i>System.Int32</i> reference whose value is updated with <i>value</i> if the original value of <i>location1</i> is equal to <i>comparand</i> .
<i>value</i>	A <i>System.Int32</i> whose value will replace the value of <i>location1</i> if <i>location1</i> and <i>comparand</i> are equal.
<i>comparand</i>	A <i>System.Int32</i> to be compared to <i>location1</i> .

14
15 **Return Value**

16 The original value of *location1*.

17 **Description**

18 The compare and store operations are performed as an atomic operation.

19

The following member must be implemented if the ExtendedNumerics library is present in the implementation.

Interlocked.CompareExchange(System.Single &, System.Single, System.Single) Method

```
[ILAsm]
.method public hidebysig static float32 CompareExchange(float32&
location1, float32 value, float32 comparand)

[C#]
public static float CompareExchange(ref float location1, float value,
float comparand)
```

Summary

Compares two `System.Single` values for equality and stores a specified value if they are equal.

Parameters

Parameter	Description
<i>location1</i>	A <code>System.Single</code> whose value is updated with <i>value</i> if its original value is equal to <i>comparand</i> .
<i>value</i>	The <code>System.Single</code> value that will replace value of <i>location1</i> if <i>location1</i> and <i>comparand</i> are equal.
<i>comparand</i>	A <code>System.Single</code> to be compared to <i>location1</i> .

Return Value

A `System.Single` containing the original value of *location1*.

Description

The compare and store operations are performed as an atomic operation.

Exceptions

Exception	Condition
<code>System.ArgumentNullException</code>	The address of <i>location1</i> is null.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21

Interlocked.CompareExchange(System.Object &, System.Object, System.Object) Method

```
[ILAsm]  
.method public hidebysig static object CompareExchange(object& location1,  
object value, object comparand)  
  
[C#]  
public static object CompareExchange(ref object location1, object value,  
object comparand)
```

Summary

Compares two `System.Object` variables for equality and stores a specified object if they are equal.

Parameters

Parameter	Description
<i>location1</i>	A <code>System.Object</code> reference that is set to <i>value</i> if the object to which it refers is equal to <i>comparand</i> .
<i>value</i>	The reference that will replace the value of <i>location1</i> if <i>location1</i> and <i>comparand</i> are equal.
<i>comparand</i>	An object to be compared to that referred to by <i>location1</i> .

Return Value

A `System.Object` containing the original value of *location1*.

Description

The compare and store operations are performed as an atomic operation.

Exceptions

Exception	Condition
<code>System.ArgumentNullException</code>	The address of <i>location1</i> is null.

1 Interlocked.Decrement(System.Int32&) 2 Method

```
3 [ILAsm]  
4 .method public hidebysig static int32 Decrement(int32& location)  
  
5 [C#]  
6 public static int Decrement(ref int location)
```

7 Summary

8 Decrements the specified variable and stores the result as an atomic operation.

9 Parameters

Parameter	Description
<i>location</i>	A <code>System.Int32</code> containing the variable whose value is to be decremented.

10 11 Return Value

12 A `System.Int32` containing the decremented value.

13 Description

14 This method handles an overflow condition by wrapping: if *location* =
15 `System.Int32.MinValue`, *location* - 1 = `System.Int32.MaxValue`. No exception is
16 thrown.

17

1 Interlocked.Decrement(System.Int64&)

2 Method

```
3 [ILAsm]  
4 .method public hidebysig static int64 Decrement(int64& location)  
5 [C#]  
6 public static long Decrement(ref long location)
```

7 Summary

8 Decrements the specified variable and stores the result as an atomic operation.

9 Parameters

Parameter	Description
<i>location</i>	A System.Int64 containing the variable whose value is to be decremented.

10

11 Return Value

12 A System.Int64 containing the decremented value.

13 Description

14 This method handles an overflow condition by wrapping: if *location* =
15 System.Int64.MinValue, *location* - 1 = System.Int64.MaxValue. No exception is
16 thrown.

17

18 The 64-bit versions of System.Threading.Interlocked.Increment and
19 System.Threading.Interlocked.Decrement are truly atomic only on systems where a
20 System.IntPtr is 64-bits long. On other systems, these methods are atomic with
21 respect to each other, but not with respect to other means of accessing the data.

22

1 Interlocked.Exchange(System.Int32&, System.Int32) Method

```
3 [ILAsm]  
4 .method public hidebysig static int32 Exchange(int32& location1, int32  
5 value)  
  
6 [C#]  
7 public static int Exchange(ref int location1, int value)
```

8 Summary

9 Sets a `System.Int32` variable to a specified value as an atomic operation and returns
10 the original value.

11 Parameters

Parameter	Description
<i>location1</i>	A <code>System.Int32</code> variable to set to the supplied value as an atomic operation.
<i>value</i>	The <code>System.Int32</code> value to which <i>location1</i> is set.

12 13 Return Value

14 A `System.Int32` containing the value of *location1* before the exchange.

15

1 **The following member must be implemented if the ExtendedNumerics library is present in**
2 **the implementation.**

3 Interlocked.Exchange(System.Single&, System.Single) Method

```
5 [ILAsm]  
6 .method public hidebysig static float32 Exchange(float32& location1,  
7 float32 value)  
8 [C#]  
9 public static float Exchange(ref float location1, float value)
```

10 Summary

11 Sets a System.Single variable to a specified value as an atomic operation and returns
12 the original value.

13 Parameters

Parameter	Description
<i>location1</i>	A System.Single variable to set to the supplied value as an atomic operation.
<i>value</i>	The System.Single value to which <i>location1</i> is set.

15 Return Value

16 A System.Single containing the value of *location1* before the exchange.

1 Interlocked.Exchange(System.Object&, System.Object) Method

```
3 [ILAsm]  
4 .method public hidebysig static object Exchange(object& location1, object  
5 value)  
  
6 [C#]  
7 public static object Exchange(ref object location1, object value)
```

8 Summary

9 Sets a `System.Object` reference to refer to a specified object as an atomic operation
10 and returns a reference to the original object.

11 Parameters

Parameter	Description
<i>location1</i>	The variable to set.
<i>value</i>	The reference to which <i>location1</i> is set.

12 Return Value

14 The original value of *location1*.

15 Exceptions

Exception	Condition
System.ArgumentNullException	The address of <i>location1</i> is null.

16
17

1 Interlocked.Increment(System.Int32& 2 Method

```
3 [ILAsm]  
4 .method public hidebysig static int32 Increment(int32& location)  
5 [C#]  
6 public static int Increment(ref int location)
```

7 Summary

8 Increments the specified variable and stores the result as an atomic operation.

9 Parameters

Parameter	Description
<i>location</i>	A <code>System.Int32</code> containing the variable whose value is to be incremented.

10

11 Return Value

12 A `System.Int32` containing the incremented value.

13 Description

14 This method handles an overflow condition by wrapping: if *location* =
15 `System.Int32.MaxValue`, *location* + 1 = `System.Int32.MinValue`. No exception is
16 thrown.

17

1 Interlocked.Increment(System.Int64&)

2 Method

```
3 [ILAsm]  
4 .method public hidebysig static int64 Increment(int64& location)  
5 [C#]  
6 public static long Increment(ref long location)
```

7 Summary

8 Increments the specified variable and stores the result as an atomic operation.

9 Parameters

Parameter	Description
<i>location</i>	A System.Int64 containing the variable whose value is to be incremented.

10

11 Return Value

12 A System.Int64 containing the incremented value.

13 Description

14 This method handles an overflow condition by wrapping: if *location* =
15 System.Int64.MaxValue, *location* + 1 = System.Int64.MinValue. No exception is
16 thrown.

17

18 The 64-bit versions of System.Threading.Interlocked.Increment and
19 System.Threading.Interlocked.Decrement are truly atomic only on systems where a
20 System.IntPtr is 64-bits long. On other systems, these methods are atomic with
21 respect to each other, but not with respect to other means of accessing the data.

22