

1 System.Threading.Interlocked Class

2
3

```
4 [ILASM]  
5 .class public sealed Interlocked extends System.Object  
6 [C#]  
7 public sealed class Interlocked
```

8 Assembly Info:

- 9 • Name: mscorlib
- 10 • Public Key: [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00]
- 11 • Version: 1.0.x.x
- 12 • Attributes:
 - 13 ○ CLSCompliantAttribute(true)

14 Summary

15

16 The **System.Threading.Interlocked** class provides atomic
17 operations for variables that are shared by multiple threads.

18 Inherits From: System.Object

19

20 **Library:** BCL

21

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

24

25 Description

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

30

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

35

36 The **System.Threading.Interlocked.Exchange** method atomically
37 exchanges the values of the specified variables. The
38 **System.Threading.Interlocked.CompareExchange** method
39 provides an atomic operation that compares two values and stores a
40 third value in one of the variables, based on the outcome of the
41 comparison.]

42

1 Interlocked.CompareExchange(System.Int32&, System.Int32, System.Int32)

3 Method

```
4 [ILASM]  
5 .method public hidebysig static int32 CompareExchange(class  
6 System.Int32& location1, int32 value, int32 comparand)  
  
7 [C#]  
8 public static int CompareExchange(ref int location1, int  
9 value, int comparand)
```

10 Summary

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

13 Parameters

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

16 17 Return Value

18
19 The original value of *location1*.

20 Description

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

23

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

3 Interlocked.CompareExchange(System.Si 4 ngle&, System.Single, System.Single) 5 Method

```
6 [ILASM]  
7 .method public hidebysig static float32  
8 CompareExchange(class System.Single& location1, float32  
9 value, float32 comparand)  
10  
11 [C#]  
12 public static float CompareExchange(ref float location1,  
13 float value, float comparand)
```

13 Summary

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

16 Parameters

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

19 Return Value

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

21 Description

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

24 Exceptions

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

1
2
3

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

```
4 [ILASM]  
5 .method public hidebysig static object  
6 CompareExchange(class System.Object& location1, object  
7 value, object comparand)  
  
8 [C#]  
9 public static object CompareExchange(ref object location1,  
10 object value, object comparand)
```

11 Summary

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

14 Parameters

15
16

Parameter	Description
<i>location1</i>	A System.Object 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> .

17
18
19

18 Return Value

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

21 Description

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

24 Exceptions

25
26

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

27
28
29

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

```
3 [ILASM]  
4 .method public hidebysig static int32 Decrement(class  
5 System.Int32& location)  
  
6 [C#]  
7 public static int Decrement(ref int location)
```

8 Summary

9 Decrement the specified variable and stores the result as an atomic
10 operation.

11 Parameters

12
13

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

14
15
16

Return Value

17 A **System.Int32** containing the decremented value.

18 Description

19 This method handles an overflow condition by wrapping: if *location* =
20 **System.Int32.MinValue**, *location* - 1 = **System.Int32.MaxValue**.
21 No exception is thrown.

22

1 Interlocked.Decrement(System.Int64& 2 Method

```
3 [ILASM]  
4 .method public hidebysig static int64 Decrement(class  
5 System.Int64& location)  
  
6 [C#]  
7 public static long Decrement(ref long location)
```

8 Summary

9 Decrement the specified variable and stores the result as an atomic
10 operation.

11 Parameters

12
13

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

14
15
16

Return Value

17 A **System.Int64** containing the decremented value.

18 Description

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

22
23 The 64-bit versions of **System.Threading.Interlocked.Increment**
24 and **System.Threading.Interlocked.Decrement** are truly atomic
25 only on systems where a **System.IntPtr** is 64-bits long. On other
26 systems, these methods are atomic with respect to each other, but not
27 with respect to other means of accessing the data.

28

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

```
3 [ILASM]  
4 .method public hidebysig static int32 Exchange(class  
5 System.Int32& location1, int32 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
10 operation and returns the original value.

11 Parameters

12
13

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

14
15
16

Return Value

17 A **System.Int32** containing the value of *location1* before the
18 exchange.

19

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

3 Interlocked.Exchange(System.Single& 4 System.Single) Method

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

11 Summary

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

14 Parameters

15
16

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.

17
18
19

Return Value

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

22

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

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

9 Summary

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

12 Parameters

13
14

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

15
16
17

16 Return Value

18 The original value of *location1*.

19 Exceptions

20
21

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

22
23
24

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

```
3 [ILASM]  
4 .method public hidebysig static int32 Increment(class  
5 System.Int32& location)  
  
6 [C#]  
7 public static int Increment(ref int location)
```

8 Summary

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

11 Parameters

12
13

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

14
15
16

Return Value

17 A **System.Int32** containing the incremented value.

18 Description

19 This method handles an overflow condition by wrapping: if *location* =
20 **System.Int32.MaxValue**, *location* + 1 = **System.Int32.MinValue**.
21 No exception is thrown.

22

1 Interlocked.Increment(System.Int64&) 2 Method

```
3 [ILASM]  
4 .method public hidebysig static int64 Increment(class  
5 System.Int64& location)  
  
6 [C#]  
7 public static long Increment(ref long location)
```

8 Summary

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

11 Parameters

12
13

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

14
15
16

Return Value

17 A **System.Int64** containing the incremented value.

18 Description

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

22
23
24
25
26
27

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

28