```
LinksPlatform's Platform Threading Class Library
     ./csharp/Platform.Threading/ConcurrentQueueExtensions.cs
   using System;
   using System.Collections.Concurrent;
2
   using System.Runtime.CompilerServices;
   using System. Threading. Tasks;
4
   using Platform.Collections.Concurrent;
   namespace Platform. Threading
8
        /// <summary>
       /// <para>Provides a set of extension methods for <see cref="ConcurrentQueue{T}"/>
10
           objects.</para>
       /// <para>Предоставляет набор методов расширения для объектов <see
11
           cref="ConcurrentQueue{T}"/>.</para>
       /// </summary>
12
       public static class ConcurrentQueueExtensions
13
14
            /// <summary>
15
            /// <para>Suspends evaluation of the method until all asynchronous operations in the
               queue finish.</para>
            /// <para>Приостановляет выполнение метода до завершения всех асинхронных операций в
               очереди.</para>
            /// </summary>
            /// <param name="queue"><para>The queue of asynchronous operations.</para><para>Очередь
               асинхронных операций.</para></param>
            /// <returns><para>An asynchronous operation representation.</para><para>Представление
            → асинхронной операции.</para></returns>
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
21
           public static async Task AwaitAll(this ConcurrentQueue<Task> queue)
22
23
                foreach (var item in queue.DequeueAll())
24
                {
25
                    await item.ConfigureAwait(continueOnCapturedContext: false);
26
                }
           }
28
29
            /// <summary>
30
            /// <para>Suspends evaluation of the method until the first asynchronous operation in
31
               the queue finishes.</para>
            /// <para>Приостанавливает выполнение метода до завершения первой асинхронной операции в
32
               очереди.</para>
            /// </summary>
            /// <param name="queue"><para>The queue of asynchronous operations.</para><para>Очередь
34
               асинхронных операций.</para></param>
            /// <returns><para>An asynchronous operation representation.</para><para>Представление
35
               асинхронной операции.</para></returns>
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
36
           public static async Task AwaitOne(this ConcurrentQueue<Task> queue)
                  (queue.TryDequeue(out Task item))
39
                {
40
                    await item.ConfigureAwait(continueOnCapturedContext: false);
41
                }
42
           }
43
44
            /// <summary>
45
            /// <para>Adds an <see cref="Action"/> as runned <see cref="Task"/> to the end of the
               <see cref="ConcurrentQueue{T}"/>.</para>
            /// <para>Добавляет <see cref="Action"/> как запущенную <see cref="Task"/> в конец <see
               cref="ConcurrentQueue{T}"/>.</para>
            /// </summary>
            /// <param name="queue"><para>The queue of asynchronous operations.</para><para>Очередь
49
               асинхронных операций.</para></param>
            /// <param name="action"><para>The <see cref="Action"/> delegate.</para><para>Делагат
50
               <see cref="Action"/>.</para></param>
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
           public static void EnqueueAsRunnedTask(this ConcurrentQueue<Task> queue, Action action)
52
            → => queue.Enqueue(Task.Run(action));
       }
53
   }
54
    ./csharp/Platform.Threading/Synchronization/ISynchronization.cs
1.2
   using System;
   using System.Runtime.CompilerServices;
2
   namespace Platform. Threading. Synchronization
```

```
5
        /// <summary>
        /// <para>Represents a synchronization object that supports read and write operations.</para>
        /// <para>Представляет объект синхронизации с поддержкой операций чтения и записи.</para>
        /// </summary>
       public interface ISynchronization
10
11
            /// <summary>
12
            /// <para>Executes action in read access mode.</para>
13
            /// <para>Выполняет действие в режиме доступа для чтения.</para>
14
15
            /// <param name="action"><para>The action.</para><para>Действие.</para></param>
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
17
            void DoRead(Action action);
18
19
            /// <summary>
20
            /// <para>Executes a function in read access mode and returns the function's
               result.</para>
            /// <para>Выполняет функцию в режиме доступа для чтения и возвращает полученный из неё
            → результат.</para>
            /// </summary>
23
            /// <typeparam name="TResult"><para>Туре of function's result.</para><para>Тип
24
            → результата функции.</para></typeparam>
/// <param name="function"><para>The function.</para><para>Функция.</para></para></para>
25
            /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
26
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
27
            TResult DoRead<TResult>(Func<TResult> function);
28
29
            /// <summary>
30
            /// <para>Executes action in write access mode.</para>
            /// <para>Выполняет действие в режиме доступа для записи.</para>
32
            /// </summary>
33
            /// <param name="action"><para>The action.</para><para>Действие.</para></param>
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
            void DoWrite(Action action);
36
37
            /// <summary>
38
            /// <para>Executes a function in write access mode and returns the function's
39
               result.</para>
            /// <para>Выполняет функцию в режиме доступа для записи и возвращает полученный из неё
40

→ результат.
            /// </summary>
            /// <typeparam name="TResult"><para>Туре of function's result.</para><para>Тип
42
            \rightarrow результата функции.</para></typeparam>
            /// <param name="function"><para>The function.</para><para>Функция.</para></param>
43
            /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
44
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
            TResult DoWrite<TResult>(Func<TResult> function);
46
        }
47
    ./csharp/Platform.Threading/Synchronization/ISynchronizationExtensions.cs
   using System;
   using System.Runtime.CompilerServices;
   namespace Platform. Threading. Synchronization
4
        /// <summary>
        /// <para>Contains extension methods for the <see cref="ISynchronization"/> interface.</para>
        /// <para>Содержит методы расширения для интерфейса <see cref="ISynchronization"/>.</para>
        /// </summary>
       public static class ISynchronizationExtensions
10
11
            /// <summary>
            /// <para>Executes a function in read access mode and returns the function's
13
               result.</para>
            /// <para>Выполняет функцию в режиме доступа для чтения и возвращает полученный из неё
14

    peзультат.
/// </summary>
15
            /// <typeparam name="TResult"><para>Туре of function's result.</para><para>Тип
            _{\hookrightarrow} результата функции.</para></typeparam> /// <typeparam name="TParam"><para>The parameter type.</para><para>Тип
            → параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
               object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter"><para>The parameter</para><para>Параметр.</para></param>
            /// <param name="function"><para>The function.</para><para>Функция.</para></param>
```

```
/// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
           public static TResult DoRead<TResult, TParam>(this ISynchronization synchronization,
23
               TParam parameter, Func<TParam, TResult> function) => synchronization.DoRead(() =>
               function(parameter));
            /// <summary>
25
            /// <para>Executes action in read access mode.</para>
26
            /// <para>Выполняет действие в режиме доступа для чтения.</para>
27
            /// </summary>
            /// <typeparam name="TParam"><para>The parameter type.</para><para>Тип
29
               параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
30
               object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter"><para>The parameter</para><para>Параметр.</para></param>
            /// <param name="action"><para>The action.</para><para>Действие.</para></param>
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
33
           public static void DoRead<TParam>(this ISynchronization synchronization, TParam
34
            parameter, Action<TParam> action) => synchronization.DoRead(() => action(parameter));
            /// <summary>
36
            /// <para>Executes a function in write access mode and returns the function's
               result.</para>
            /// <para>Выполняет функцию в режиме доступа для записи и возвращает полученный из неё
               результат.</para>
            /// </summary>
            /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
40
               результата функции.</para></typeparam>
               <typeparam name="TParam"><para>The parameter type.</para><para>Тип
               параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
42
               object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter"><para>The parameter</para><para>Параметр </para></para>
            /// <param name="function"><para>The function.</para><para>Функция.</para></param>
44
            /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
45
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
46
           public static TResult DoWrite<TResult, TParam>(this ISynchronization synchronization,
47
               TParam parameter, Func<TParam, TResult> function) => synchronization.DoWrite(() =>
               function(parameter));
            /// <summary>
49
            /// <para>Executes action in write access mode.</para>
50
            /// <para>Выполняет действие в режиме доступа для записи.</para>
5.1
            /// </summary>
            /// <typeparam name="TParam"><para>The parameter type.</para><para>Тип
53
            → параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
54
               object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter"><para>The parameter</para><para>Параметр.</para></para>
55
           /// <param name="action"><para>The action.</para><para>Действие.</para></param>
[MethodImpl(MethodImplOptions.AggressiveInlining)]
57
           public static void DoWrite<TParam>(this ISynchronization synchronization, TParam
58
               parameter, Action<TParam> action) => synchronization.DoWrite(() =>
               action(parameter));
59
            /// <summary>
60
            /// <para>Executes a function in read access mode and returns the function's
61
               result.</para>
            /// <para>Выполняет функцию в режиме доступа для чтения и возвращает полученный из неё
               результат.</para>
            /// </summary>
            /// <typeparam name="TResult"><para>Туре of function's result.</para><para>Тип
               результата функции.</para></typeparam>
            /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
               параметра.</para></typeparam>
            /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
               параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
               object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter1"><para>The first parameter</para><para>Первый
               параметр.</para></param>
            /// <param name="parameter2"><para>The second parameter</para><para>Второй
69
               параметр.</para></param>
               <param name="function"><para>The function.</para><para>Функция.</para></param>
            /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
7.1
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
```

```
public static TResult DoRead<TResult, TParam1, TParam2>(this ISynchronization
7.3
                synchronization, TParam1 parameter1, TParam2 parameter2, Func<TParam1, TParam2,
                TResult> function) => synchronization.DoRead(() => function(parameter1, parameter2));
            /// <summary>
            /// <para>Executes action in read access mode.</para>
76
            /// <para>Выполняет действие в режиме доступа для чтения.</para>
77
            /// </summary>
78
            /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
               параметра.</para></typeparam>
            /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
80
               параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
               object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter1"><para>The first parameter</para><para>Первый
               параметр.</para></param>
            /// <param name="parameter2"><para>The second parameter</para><para>Второй
               параметр.</para></param>
            /// <param name="action"><para>The action.</para><para>Действие.</para></param>
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
85
            public static void DoRead<TParam1, TParam2>(this ISynchronization synchronization,
86
                TParam1 parameter1, TParam2 parameter2, Action<TParam1, TParam2> action) =>
                synchronization.DoRead(() => action(parameter1, parameter2));
            /// <summary>
88
            /// <para>Executes a function in write access mode and returns the function's
               result.</para>
            /// <para>Выполняет функцию в режиме доступа для записи и возвращает полученный из неё
            → результат.</para>
            /// </summary>
91
            /// <typeparam name="TResult"><para>Туре of function's result.</para><para>Тип
92
            → результата функции.</para></typeparam>
/// <typeparam name="TParam1"><para>The first parameter type.</para>Tип первого
               параметра.</para></typeparam>
            /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
                параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
                object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter1"><para>The first parameter</para><para>Первый
               параметр.</para></param>
            /// <param name="parameter2"><para>The second parameter</para><para>Второй
               параметр.</para></param>
            /// <param name="function"><para>The function.</para><para>Функция.</para></param>
            /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
99
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
100
            public static TResult DoWrite<TResult, TParam1, TParam2>(this ISynchronization
101
                synchronization, TParam1 parameter1, TParam2 parameter2, Func<TParam1, TParam2,
                TResult> function) => synchronization.DoWrite(() => function(parameter1,
                parameter2));
            /// <summary>
103
            /// <para>Executes action in write access mode.</para>
104
            /// <para>Выполняет действие в режиме доступа для записи.</para>
105
            /// </summary>
            /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
107
               параметра.</para></typeparam>
            /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
108
               параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
109
                object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter1"><para>The first parameter</para><para>Первый
               параметр.</para></param>
            /// <param name="parameter2"><para>The second parameter</para><para>Второй
111
               параметр.</para></param>
            /// <param name="action"><para>The action.</para><para>Действие.</para></param>
112
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
113
            public static void DoWrite<TParam1, TParam2>(this ISynchronization synchronization,
                TParam1 parameter1, TParam2 parameter2, Action<TParam1, TParam2> action) =>
                synchronization.DoWrite(() => action(parameter1, parameter2));
            /// <summary>
            /// <para>Executes a function in read access mode and returns the function's

→ result.</para>

            /// <para>Выполняет функцию в режиме доступа для чтения и возвращает полученный из неё
                результат.</para>
```

```
/// </summary>
119
            /// <typeparam name="TResult"><para>Туре of function's result.</para><para>Тип
                результата функции.</para></typeparam>
            /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
                параметра.</para></typeparam>
            /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
122
                параметра.</para></typeparam>
            /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
123
                параметра. </para></typeparam>
            /// <param name="synchronization"><para>Synchronization
                object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter1"><para>The first parameter</para><para>Первый
                параметр.</para></param>
            /// <param name="parameter2"><para>The second parameter</para><para>Второй
                параметр.</para></param>
            /// <param name="parameter3"><para>The third parameter</para><para>Третий
127
                параметр.</para></param>
            /// <param name="function"><para>The function.</para><para>Функция.</para></param>
128
            /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
130
            public static TResult DoRead<TResult, TParam1, TParam2, TParam3>(this ISynchronization
131
                synchronization, TParam1 parameter1, TParam2 parameter2, TParam3 parameter3,
                Func<TParam1, TParam2, TParam3, TResult> function) => synchronization.DoRead(() =>
                function(parameter1, parameter2, parameter3));
132
            /// <summary>
            /// <para>Executes action in read access mode.</para>
            /// <para>Выполняет действие в режиме доступа для чтения.</para>
135
            /// </summary>
136
            /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
               параметра.</para></typeparam>
            /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
138
                параметра.</para></typeparam>
            /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
139
                параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
140
                object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter1"><para>The first parameter</para><para>Первый
               параметр.</para></param>
            /// <param name="parameter2"><para>The second parameter</para><para>Второй
               параметр.</para></param>
            /// <param name="parameter3"><para>The third parameter</para><para>Третий
143
                параметр.</para></param>
            /// <param name="action"><para>The action.</para>Zpara>Действие.</para></param>
144
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
145
            public static void DoRead<TParam1, TParam2, TParam3>(this ISynchronization
                synchronization, TParam1 parameter1, TParam2 parameter2, TParam3 parameter3,
                Action<TParam1, TParam2, TParam3> action) => synchronization.DoRead(() =>
                action(parameter1, parameter2, parameter3));
            /// <summary>
148
            /// <para>Executes a function in write access mode and returns the function's
149
                result.</para>
            /// <para>Выполняет функцию в режиме доступа для записи и возвращает полученный из неё
               результат.</para>
                </summary>
151
            /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
152
            → результата функции.</para></typeparam>
/// <typeparam name="TParam1"><para>The first parameter type.</para>Tип первого
153
               параметра.</para></typeparam>
            /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
                параметра.</para></typeparam>
            /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
                параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
                object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter1"><para>The first parameter</para><para>Первый
               параметр.</para></param>
            /// <param name="parameter2"><para>The second parameter</para><para>Второй
                параметр.</para></param>
            /// <param name="parameter3"><para>The third parameter</para><para>Третий
                параметр. </para></param>
            /// <param name="function"><para>The function.</para><para>Функция.</para></param>
160
            /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
161
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
```

```
public static TResult DoWrite<TResult, TParam1, TParam2, TParam3>(this ISynchronization
163
                synchronization, TParam1 parameter1, TParam2 parameter2, TParam3 parameter3,
                Func<TParam1, TParam2, TParam3, TResult> function) => synchronization.DoWrite(() =>
                function(parameter1, parameter2, parameter3));
            /// <summary>
            /// <para>Executes action in write access mode.</para>
166
            /// <para>Выполняет действие в режиме доступа для записи.</para>
167
            /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
169
            → параметра.</para></typeparam>
/// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
170
                параметра.</para></typeparam>
            /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
                параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
                object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter1"><para>The first parameter</para><para>Первый
               параметр.</para></param>
            /// <param name="parameter2"><para>The second parameter</para><para>Второй
               параметр.</para></param>
            /// <param name="parameter3"><para>The third parameter</para><para>Третий
175
               параметр.</para></param>
            /// <param name="action"><para>The action.</para>Zpara>Действие.</para></param>
176
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
            public static void DoWrite<TParam1, TParam2, TParam3>(this ISynchronization
178
                synchronization, TParam1 parameter1, TParam2 parameter2, TParam3 parameter3,
                Action<TParam1, TParam2, TParam3> action) => synchronization.DoWrite(() =>
                action(parameter1, parameter2, parameter3));
179
            /// <summary>
180
            /// <para>Executes a function in read access mode and returns the function's
181
               result.</para>
            /// <para>Выполняет функцию в режиме доступа для чтения и возвращает полученный из неё
            → результат.</para>
/// </summary>
183
            /// <typeparam name="TResult"><para>Туре of function's result.</para><para>Тип
184
               результата функции.</para></typeparam>
            /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
              параметра.</para></typeparam>
            /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
                параметра.</para></typeparam>
            /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
187
                параметра.</para></typeparam>
            /// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
188
                параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
189
                object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter1"><para>The first parameter</para><para>Первый
                параметр.</para></param>
            /// <param name="parameter2"><para>The second parameter</para><para>Второй
                параметр.</para></param>
            /// <param name="parameter3"><para>The third parameter</para><para>Третий
192
                параметр.</para></param>
            /// <param name="parameter4"><para>The forth parameter</para><para>Чертвёртый
193
                параметр.</para></param>
            /// <param name="function"><para>The function.</para><para>Функция.</para></param>
            /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
196
            197
                function) => synchronization.DoRead(() => function(parameter1, parameter2,
                parameter3, parameter4));
198
            /// <summary>
199
            /// <para>Executes action in read access mode.</para>
            /// <para>Выполняет действие в режиме доступа для чтения.</para>
            /// </summary>
202
            /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
203
                параметра.</para></typeparam>
            /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
                параметра.</para></typeparam>
            /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
205
                параметра.</para></typeparam>
```

```
/// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
206
                параметра. </para></typeparam>
            /// <param name="synchronization"><para>Synchronization
207
                object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter1"><para>The first parameter</para><para>Первый
                параметр.</para></param>
            /// <param name="parameter2"><para>The second parameter</para><para>Второй
209
                параметр.</para></param>
            /// <param name="parameter3"><para>The third parameter</para><para>Третий
210
                параметр.</para></param>
            /// <param name="parameter4"><para>The forth parameter</para><para>Чертвёртый
211
                параметр.</para></param>
             /// <param name="action"><para>The action.</para><para>Действие.</para></param>
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
213
            public static void DoRead<TParam1, TParam2, TParam3, TParam4>(this ISynchronization
214
                synchronization, TParam1 parameter1, TParam2 parameter2, TParam3 parameter3, TParam4 parameter4, Action<TParam1, TParam2, TParam3, TParam4> action) =>
                synchronization.DoRead(() => action(parameter1, parameter2, parameter3, parameter4));
215
            /// <summary>
            /// <para>Executes a function in write access mode and returns the function's
                result.</para>
            /// <para>Выполняет функцию в режиме доступа для записи и возвращает полученный из неё
218
            → результат.</para>
/// </summa==:
219
            /// <typeparam name="TResult"><para>Туре of function's result.</para><para>Тип
                результата функции.</para></typeparam>
            /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
                параметра.</para></typeparam>
            /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
222
                параметра.</para></typeparam>
            /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
223
                параметра.</para></typeparam>
            /// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
                параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
                object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter1"><para>The first parameter</para><para>Первый
226
                параметр.</para></param>
            /// <param name="parameter2"><para>The second parameter</para><para>Второй
227
                параметр.</para></param>
            /// <param name="parameter3"><para>The third parameter</para><para>Третий
228
                параметр.</para></param>
            /// <param name="parameter4"><para>The forth parameter</para><para>Чертвёртый
                параметр.</para></param>
            /// <param name="function"><para>The function.</para><para>Функция.</para></param>
230
            /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
231
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
232
            public static TResult DoWrite<TResult, TParam1, TParam2, TParam3, TParam4>(this
                ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2, TParam3
                parameter3, TParam4 parameter4, Func<TParam1, TParam2, TParam3, TParam4, TResult>
                function) => synchronization.DoWrite(() => function(parameter1, parameter2,
                parameter3, parameter4));
234
            /// <summary>
            /// <para>Executes action in write access mode.</para>
236
            /// <para>Выполняет действие в режиме доступа для записи.</para>
237
            /// </summary>
238
            /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
239
                параметра.</para></typeparam>
            /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
240
                параметра.</para></typeparam>
            /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
241
                параметра.</para></typeparam>
            /// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
242
                параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
                object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter1"><para>The first parameter</para><para>Первый
                параметр.</para></param>
            /// <param name="parameter2"><para>The second parameter</para><para>Второй
245
                параметр.</para></param>
            /// <param name="parameter3"><para>The third parameter</para><para>Третий
246
                параметр.</para></param>
```

```
/// <param name="parameter4"><para>The forth parameter</para><para>Чертвёртый
247
                параметр.</para></param>
             /// <param name="action"><para>The action.</para><para>Действие.</para></param>
248
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
249
            public static void DoWrite<TParam1, TParam2, TParam3, TParam4>(this ISynchronization
250
                synchronization, TParam1 parameter1, TParam2 parameter2, TParam3 parameter3, TParam4
                parameter4, Action<TParam1, TParam2, TParam3, TParam4> action) =>
                synchronization.DoWrite(() => action(parameter1, parameter2, parameter3,
                parameter4));
        }
251
252
1.4
     ./csharp/Platform.Threading/Synchronization/ISynchronized.cs
   using System.Runtime.CompilerServices;
    namespace Platform. Threading. Synchronization
 3
 4
        /// <summary>
        /// <para>Represents extendable synchronized interface access gate.</para>
 6
        /// <para>Представляет расширяемый интерфейс шлюза синхронизированного доступа.</para>
        /// </summary>
        /// <typeparam name="TInterface"><para>Synchronized interface.</para><para>Синхронизируемый
 9
            интерфейс.</para></typeparam>
        public interface ISynchronized<out TInterface>
10
             /// <summary>
12
             /// <para>Gets sychronization method.</para>
13
             /// <para>Возвращает способ синхронизации.</para>
14
             /// </summary>
15
            ISynchronization SyncRoot
16
                 [MethodImpl(MethodImplOptions.AggressiveInlining)]
18
19
            }
20
             /// <summary>
22
             /// <para>Get source version of <typeparamref name="TInterface"/>, that does not
23
            garantee thread safe access synchronization.</para>
/// <para>Возвращает исходную версию <typeparamref name="TInterface"/>, которая не
24
                гарантирует потокобезопасную синхронизацию доступа.</para>
             /// </summary>
            /// <remarks>
26
            /// <para>It is unsafe to use it directly, unless compound context using SyncRoot is
                created.</para>
             /// <para>Использовать напрямую небезопасно, за исключением ситуации когда создаётся
                составной контекст с использованием SyncRoot.</para>
             /// </remarks>
             TInterface Unsync
30
31
                 [MethodImpl(MethodImplOptions.AggressiveInlining)]
32
                 get;
33
            }
35
             /// <summary>
36
             /// <para>Get wrapped/decorated version of <typeparamref name="TInterface"/>, that does
37
                garantee thread safe access synchronization.</para>
             /// <para>Возвращает обернутую/декорированную версию <typeparamref name="TInterface"/>,
38
                которая гарантирует потокобезопасную синхронизацию доступа.</para>
             /// </summary>
39
             /// <remarks>
             /// <para>It is safe to use it directly, because it must be thread safe
41
                implementation.</para>
             /// <para>Безопасно использовать напрямую, так как реализация должна быть
42
                потокобезопасной.</para>
             /// </remarks>
43
            TInterface Sync
44
             {
45
                 [MethodImpl(MethodImplOptions.AggressiveInlining)]
46
47
                 get;
            }
48
        }
49
    }
50
     ./csharp/Platform.Threading/Synchronization/ReaderWriterLockSynchronization.cs
   using System;
```

using System.Runtime.CompilerServices;

using System. Threading;

```
namespace Platform. Threading. Synchronization
5
6
        /// <summary>
        /// <para>Implementation of <see cref="ISynchronization"/> based on <see
           cref="ReaderWriterLockSlim"/>.</para>
        /// <para>Реализация <see cref="ISynchronization"/> на основе <see
9
           cref="ReaderWriterLockSlim"/>.</para>
        /// </summary>
10
        public class ReaderWriterLockSynchronization : ISynchronization
11
12
            private readonly ReaderWriterLockSlim _rwLock = new
            ReaderWriterLockSlim(LockRecursionPolicy.SupportsRecursion);
14
            /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml
15
            \rightarrow path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronization]
                n.DoRead(System.Action)"]/*'/>
            /// <inheritdoc/>
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
17
            public void DoRead(Action action)
18
20
                _rwLock.EnterReadLock();
                try
21
22
                     action();
23
24
                finally
25
26
27
                     _rwLock.ExitReadLock();
                }
28
            }
29
            /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
31
            \rightarrow path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronization]
                n.DoRead``1(System.Func{``0})"]/*'/>
            /// <inheritdoc/>
32
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
33
            public TResult DoRead<TResult>(Func<TResult> function)
35
                 _rwLock.EnterReadLock();
36
                try
37
                {
38
39
                    return function();
40
                finally
41
                {
                     _rwLock.ExitReadLock();
43
                }
44
            }
46
            /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
47
            \rightarrow path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronization]
                n.DoWrite(System.Action)"]/*'/>
            /// <inheritdoc/>
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
49
            public void DoWrite(Action action)
50
51
                 _rwLock.EnterWriteLock();
                try
53
                {
                     action();
55
56
                finally
57
                {
58
                     _rwLock.ExitWriteLock();
59
                }
            }
61
62
            /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
63
                path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
                n.DoWrite``1(System.Func{``0})"]/*'/>
            /// <inheritdoc/>
64
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
65
            public TResult DoWrite<TResult>(Func<TResult> function)
67
                 _rwLock.EnterWriteLock();
68
                try
69
7.0
```

```
return function();
                           finally
73
                                   _rwLock.ExitWriteLock();
7.5
                           }
76
                    }
77
             }
78
79
        ./csharp/Platform.Threading/Synchronization/Unsynchronization.cs
1.6
     using System;
1
      using System.Runtime.CompilerServices;
 3
      namespace Platform. Threading. Synchronization
 4
 5
              /// <summary>
             /// <para>Implementation of <see cref="ISynchronization"/> that makes no actual
                   synchronization.</para>
             /// /// cref="ISynchronization"/>, которая не выполняет фактическую
                   синхронизацию.</para>
             /// </summary>
             public class Unsynchronization : ISynchronization
10
11
12
                    /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
                     _{\rightarrow} \quad \texttt{path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronization.} \\
                          n.DoRead(System.Action)"]/*'/>
                    /// <inheritdoc/>
                    [MethodImpl(MethodImplOptions.AggressiveInlining)]
                    public void DoRead(Action action) => action();
15
                    /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
17
                     _{\rightarrow} \quad \text{path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronization.} \\
                          n.DoRead``1(System.Func{``0})"]/*'/>
                    /// <inheritdoc/>
18
                    [MethodImpl(MethodImplOptions.AggressiveInlining)]
19
                    public TResult DoRead<TResult>(Func<TResult> function) => function();
20
                    /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
                     path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
                          n.DoWrite(System.Action)"]/*'/>
                    /// <inheritdoc/>
                    [MethodImpl(MethodImplOptions.AggressiveInlining)]
24
                    public void DoWrite(Action action) => action();
25
26
                    /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
2.7
                     _{\rightarrow} \quad \text{path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronization.} \\ In the path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronization.] \\ In the path='doc/members/members/members/members/members. \\ In the path='doc/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/members/mem
                          n.DoWrite``1(System.Func{``0})"]/*'/>
                    /// <inheritdoc/>
                    [MethodImpl(MethodImplOptions.AggressiveInlining)]
29
                    public TResult DoWrite<TResult>(Func<TResult> function) => function();
30
             }
31
      }
32
       ./csharp/Platform.Threading/TaskExtensions.cs
      using System.Runtime.CompilerServices;
      using System. Threading. Tasks;
 3
      namespace Platform. Threading
      {
 5
             /// <summary>
 6
             /// <para>Provides a set of extension methods for <see cref="Task{TReturn}"/> objects.</para>
             /// <para>Предоставляет набор методов расширения для объектов <see
                   cref="Task{TReturn}"/>.</para>
             /// </summary>
             public static class TaskExtensions
11
                    /// <summary>
12
                    /// <para>Waits for completion of the asynchronous <see cref="Task{TReturn}"/> and
13
                           returns its result.</para>
                    /// <para>Ожидает завершения асинхронной <see cref="Task{TReturn}"/> и возвращает её

→ pезультат.</para>
/// </summary>

                    /// <typeparam name="TReturn"><para>The return value type.</para><para>Тип возвращаемого
16
                          значения.</para></typeparam>
                    /// <param name="task"><para>The asynchronous <see
                          cref="Task{TReturn}"/>.</para><para>Aссинхронная <see
                          cref="Task{TReturn}"/>.</para></param>
```

```
/// <returns><para>The result of completed <see
            стеf="Task{TReturn}"/>.</para>Результат завершённой <see
               cref="Task{TReturn}"/>.</para></returns>
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
           public static TReturn AwaitResult<TReturn>(this Task<TReturn> task) =>
20
               task.GetAwaiter().GetResult();
21
   }
22
     ./csharp/Platform. Threading/Thread Helpers. cs\\
1.8
   using System;
using System.Runtime.CompilerServices;
   using System. Threading;
3
   namespace Platform. Threading
6
       /// <summary>
/// <para>Provides a set of helper methods for <see cref="Thread"/> objects.</para>
7
       /// <para>Предоставляет набор вспомогательных методов для объектов <see
9
           cref="Thread"/>.</para>
       /// </summary>
10
       public static class ThreadHelpers
11
12
            /// <summary>
13
            /// <para>Gets the maximum stack size in bytes by default.</para>
14
            /// <para>Возвращает размер максимальный стека в байтах по умолчанию.</para>
           /// </summarv>
16
           public static readonly int DefaultMaxStackSize;
18
19
            /// <summary>
           /// <para>Gets the extended maximum stack size in bytes by default.</para>
20
           /// <para>Возвращает расширенный максимальный размер стека в байтах по умолчанию.</para>
21
           /// </summary>
           public static readonly int DefaultExtendedMaxStackSize = 256 * 1024 * 1024;
23
           /// <summary>
25
           /// <para>Returns the default time interval for transferring control to other threads in
26
               milliseconds</para>
            /// <para>Возвращает интервал времени для передачи управления другим потокам в
               миллисекундах по умолчанию.</para>
            /// </summary>
           public static readonly int DefaultSleepInterval = 1;
29
30
            /// <summary>
31
            /// <para>Invokes the <see cref="Action{T}"/> with modified maximum stack size.</para>
32
            /// <para>Вызывает <see cref="Action{T}"/> с изменённым максимальным размером
               стека.</para>
            /// </summary>
3.4
            /// <typeparam name="T"><para>The type of the <see cref="Action{T}"/>
35
               argument.</para><para>Тип аргумента <see cref="Action{T}"/>.</para></typeparam>
            /// <param name="param"><para>The object containing data to be used by the invoked <see
36
               cref="Action{T}"/> delegate.</para><para>Объект, содержащий данные, которые будут
               использоваться вызваемым делегатом <see cref="Action{T}"/>.</para></param>
            /// <param name="action"><para>The <see cref="Action{T}"/> delegate.</para><para>Делагат
               <see cref="Action{T}"/>.</para></param>
            /// <param name="maxStackSize"><para>The maximum stack size in
               bytes.</para><para>Mаксимальный размер стека в байтах.</para></param>
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
39
           public static void InvokeWithModifiedMaxStackSize<T>(T param, Action<object> action, int
40
            maxStackSize) => StartNew(param, action, maxStackSize).Join();
41
            /// <summary>
42
            /// <para>Invokes the <see cref="Action{T}"/> with extend maximum stack size.</para>
            /// <para>Вызывает <see cref="Action{T}"/> с расширенным максимальным размером
44
               стека.</para>
            /// </summary>
45
            /// <typeparam name="T"><para>The type of the <see cref="Action{T}"/>
46
               argument.</para><para>Тип аргумента <see cref="Action{T}"/>.</para></typeparam>
            /// <param name="param"><para>The object containing data to be used by the invoked <see
            cref="Action{T}"/> delegate.</para>Объект, содержащий данные, которые будут
               использоваться вызваемым делегатом <see cref="Action{T}"/>.</para></param>
            /// <param name="action"><para>The <see cref="Action{T}"/> delegate.</para><para>Делагат
               <see cref="Action{T}"/>.</para></param>
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
49
           public static void InvokeWithExtendedMaxStackSize<T>(T param, Action<object> action) =>
50
               InvokeWithModifiedMaxStackSize(param, action, DefaultExtendedMaxStackSize);
```

```
/// <summary>
52
           /// <para>Invokes the <see cref="Action"/> with modified maximum stack size.</para>
           /// <para>Вызывает <see cref="Action"/> с изменённым максимальным размером cтека.</para>
54
           /// </summary>
55
           /// <param name="action"><para>The <see cref="Action"/> delegate.</para><para>Делагат
               <see cref="Action"/>.</para></param>
           /// <param name="maxStackSize"><para>The maximum stack size in
            → bytes.</para><para>Mаксимальный размер стека в байтах.</para></param>
           [MethodImpl(MethodImplOptions.AggressiveInlining)]
           public static void InvokeWithModifiedMaxStackSize(Action action, int maxStackSize) =>
59
               StartNew(action, maxStackSize).Join();
60
           /// <summary>
61
           /// <para>Invokes the <see cref="Action"/> with extend maximum stack size.</para>
62
           /// <para>Вызывает <see cref="Action"/> с расширенным максимальным размером стека.</para>
           /// </summary>
64
           /// <param name="action"><para>The <see cref="Action"/> delegate.</para><para>Делагат
65
               <see cref="Action"/>.</para></param>
           [MethodImpl(MethodImplOptions.AggressiveInlining)]
66
           public static void InvokeWithExtendedMaxStackSize(Action action) =>
               InvokeWithModifiedMaxStackSize(action, DefaultExtendedMaxStackSize);
68
           /// <summary>
           /// <para>Initializes a new instance of the <see cref="Thread"/> class, causes the
70
               operating system to change the state of that instance to <see
               cref="ThreadState.Running"/> and supplies an object containing data to be used by
               the method that thread executes.</para>
           /// <para>Инициализирует новый экземпляр класса <see cref="Thread"/>, просит
               операционную систему изменить состояние этого экземпляра на <see
               cref="ThreadState.Running"/> и предоставляет объект, содержащий данные, которые
               будут использоваться в методе, который выполняет этот поток.</para>
           /// </summary>
           /// <typeparam name="T"><para>The type of the <see cref="Action{T}"/>
73
               argument.</para><para>Тип аргумента <see cref="Action{T}"/>.</para></typeparam>
           /// <param name="param"><para>The object containing data to be used by the method that
              thread executes.</para><para>Объект, содержащий данные, которые будут использоваться
               методом, выполняемым потоком.</para></param>
           /// <param name="action"><para>The <see cref="Action{T}"/> delegate.</para><para>Делагат
               <see cref="Action{T}"/>.</para></param>
           /// <param name="maxStackSize"><para>The maximum stack size in
               bytes.</para><para>Максимальный размер стека в байтах.</para></param>
           /// <returns><para>A new started <see cref="Thread"/> instance.</para><para>Новый
               запущенный экземпляр <see cref="Thread"/>.</para></returns>
           [MethodImpl(MethodImplOptions.AggressiveInlining)]
           public static Thread StartNew<T>(T param, Action<object> action, int maxStackSize)
               var thread = new Thread(new ParameterizedThreadStart(action), maxStackSize);
81
               thread.Start(param);
82
               return thread;
83
           }
84
           /// <summary>
86
           /// <para>Initializes a new instance of the <see cref="Thread"/> class, causes the
87
               operating system to change the state of that instance to <see
               cref="ThreadState.Running"/> and supplies an object containing data to be used by
            \hookrightarrow
               the method that thread executes.</para>
           /// <para>Инициализирует новый экземпляр класса <see cref="Thread"/>, просит
               операционную систему изменить состояние этого экземпляра на <see
               cref="ThreadState.Running"/> и предоставляет объект, содержащий данные, которые
            \hookrightarrow
               будут использоваться в методе, который выполняет этот поток.</para>
           /// </summary>
89
           /// <typeparam name="T"><para>The type of the <see cref="Action{T}"/>
90
               argument.</para><para>Тип аргумента <see cref="Action{T}"/>.</para></typeparam>
           /// <param name="param"><para>The object containing data to be used by the method that
               thread executes.</para><para>Объект, содержащий данные, которые будут использоваться
               методом, выполняемым потоком.</para></param>
           /// <param name="action"><para>The <see cref="Action{T}"/> delegate.</para><para>Делагат
               <see cref="Action{T}"/>.</para></param>
           /// <returns><para>A new started <see cref="Thread"/> instance.</para><para>Новый
               запущенный экземпляр <see cref="Thread"/>.</para></returns>
           [MethodImpl(MethodImplOptions.AggressiveInlining)]
           public static Thread StartNew<T>(T param, Action<object> action) => StartNew(param,
               action, DefaultMaxStackSize);
96
```

/// <summary>

```
/// <para>Initializes a new instance of the <see cref="Thread"/> class, causes the
                operating system to change the state of that instance to <see
                cref="ThreadState.Running"/> and supplies the method executed by that thread.</para>
            /// <para>Инициализирует новый экземпляр класса <see cref="Thread"/>, просит
                операционную систему изменить состояние этого экземпляра на <see
                cref="ThreadState.Running"/> и предоставляет метод, который выполняется этим
                потоком.</para>
            /// </summary>
100
            /// <param name="action"><para>The <see cref="Action"/> delegate.</para><para>Делагат
101
                <see cref="Action"/>.</para></param>
            /// <param name="maxStackSize"><para>The maximum stack size in \footnote{Max.eq}
                bytes.</para><para>Максимальный размер стека в байтах.</para></param>
            /// <returns><para>A new started <see cref="Thread"/> instance.</para><para>Новый
103
                запущенный экземпляр <see cref="Thread"/>.</para></returns>
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
104
            public static Thread StartNew(Action action, int maxStackSize)
105
                var thread = new Thread(new ThreadStart(action), maxStackSize);
107
                thread.Start():
108
109
                return thread;
            }
110
            /// <summary>
112
            /// <para>Initializes a new instance of the <see cref="Thread"/> class, causes the
113
                operating system to change the state of that instance to <see
                cref="ThreadState.Running"/> and supplies the method executed by that thread.</para>
            /// <para>Инициализирует новый экземпляр класса <see cref="Thread"/>, просит
                операционную систему изменить состояние этого экземпляра на <see
               cref="ThreadState.Running"/> и предоставляет метод, который выполняется этим
                потоком.</para>
            /// </summary>
115
            /// <param name="action"><para>The <see cref="Action"/> delegate.</para><para>Делагат
116
                <see cref="Action"/>.</para></param>
            /// <returns><para>A new started <see cref="Thread"/> instance.</para><para>Новый
117
                запущенный экземпляр <see cref="Thread"/>.</para></returns>
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
            public static Thread StartNew(Action action) => StartNew(action, DefaultMaxStackSize);
119
            /// <summary>
121
            /// Suspends the current thread for the <see cref="DefaultSleepInterval"/>.
122
            /// </summary>
123
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
124
            public static void Sleep() => Thread.Sleep(DefaultSleepInterval);
125
        }
126
127
1.9
     ./csharp/Platform.Threading.Tests/ThreadHelpersTests.cs
    using Xunit;
 1
 2
 3
    namespace Platform. Threading. Tests
 4
        public class ThreadHelpersTests
 5
 6
            [Fact]
            public void InvokeTest()
                var number = 0;
10
                ThreadHelpers.InvokeWithExtendedMaxStackSize(() => number = 1);
11
                Assert.Equal(1, number);
                ThreadHelpers.InvokeWithExtendedMaxStackSize(2, (param) => number = (int)param);
13
                Assert.Equal(2, number);
14
                ThreadHelpers.InvokeWithModifiedMaxStackSize(() => number = 1, maxStackSize: 512);
15
                Assert.Equal(1, number);
16
                ThreadHelpers.InvokeWithModifiedMaxStackSize(2, (param) => number = (int)param,
17

→ maxStackSize: 512);

                Assert.Equal(2, number);
18
            }
        }
20
```

21 }

Index

```
./csharp/Platform.Threading.Tests/ThreadHelpersTests.cs, 13
./csharp/Platform.Threading/ConcurrentQueueExtensions.cs, 1
./csharp/Platform.Threading/Synchronization/ISynchronization.cs, 1
./csharp/Platform.Threading/Synchronization/ISynchronizationExtensions.cs, 2
./csharp/Platform.Threading/Synchronization/ISynchronized.cs, 8
./csharp/Platform.Threading/Synchronization/ReaderWriterLockSynchronization.cs, 8
./csharp/Platform.Threading/Synchronization/Unsynchronization.cs, 10
./csharp/Platform.Threading/TaskExtensions.cs, 10
./csharp/Platform.Threading/ThreadHelpers.cs, 11
```