

LinksPlatform's Platform.IO Class Library

1.1 ./csharp/Platform.IO/ConsoleCancellation.cs

```
1  using System;
2  using System.Runtime.CompilerServices;
3  using System.Threading;
4  using Platform.Disposables;
5  using Platform.Threading;
6
7  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
8
9  namespace Platform.IO
10 {
11     public class ConsoleCancellation : DisposableBase
12     {
13         public CancellationTokenSource Source
14         {
15             [MethodImpl(MethodImplOptions.AggressiveInlining)]
16             get;
17         }
18
19         public CancellationToken Token
20         {
21             [MethodImpl(MethodImplOptions.AggressiveInlining)]
22             get;
23         }
24
25         public bool IsRequested
26         {
27             [MethodImpl(MethodImplOptions.AggressiveInlining)]
28             get => Source.IsCancellationRequested;
29         }
30
31         public bool NotRequested
32         {
33             [MethodImpl(MethodImplOptions.AggressiveInlining)]
34             get => !Source.IsCancellationRequested;
35         }
36
37         [MethodImpl(MethodImplOptions.AggressiveInlining)]
38         public ConsoleCancellation()
39         {
40             Source = new CancellationTokenSource();
41             Token = Source.Token;
42             Console.CancelKeyPress += OnCancelKeyPress;
43         }
44
45         [MethodImpl(MethodImplOptions.AggressiveInlining)]
46         public void ForceCancellation() => Source.Cancel();
47
48         [MethodImpl(MethodImplOptions.AggressiveInlining)]
49         public void Wait()
50         {
51             while (NotRequested)
52             {
53                 ThreadHelpers.Sleep();
54             }
55         }
56
57         [MethodImpl(MethodImplOptions.AggressiveInlining)]
58         protected override void Dispose(bool manual, bool wasDisposed)
59         {
60             if (!wasDisposed)
61             {
62                 Console.CancelKeyPress -= OnCancelKeyPress;
63                 Source.DisposeIfPossible();
64             }
65         }
66
67         [MethodImpl(MethodImplOptions.AggressiveInlining)]
68         private void OnCancelKeyPress(object sender, ConsoleCancelEventArgs e)
69         {
70             e.Cancel = true;
71             if (NotRequested)
72             {
73                 Source.Cancel();
74             }
75         }
76     }
77 }
```

1.2 ./csharp/Platform.IO/ConsoleHelpers.cs

```
1 using System;
2 using System.Diagnostics;
3 using System.Runtime.CompilerServices;
4 using Platform.Collections;
5 using Platform.Collections.Arrays;
6
7 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
8
9 namespace Platform.IO
10 {
11     public static class ConsoleHelpers
12     {
13         [MethodImpl(MethodImplOptions.AggressiveInlining)]
14         public static void PressAnyKeyToContinue()
15         {
16             Console.WriteLine("Press any key to continue.");
17             Console.ReadKey();
18         }
19
20         [MethodImpl(MethodImplOptions.AggressiveInlining)]
21         public static string GetOrReadArgument(int index, params string[] args) =>
22             ↪ GetOrReadArgument(index, $"{index + 1} argument", args);
23
24         [MethodImpl(MethodImplOptions.AggressiveInlining)]
25         public static string GetOrReadArgument(int index, string readMessage, params string[]
26             ↪ args)
27         {
28             if (!args.TryGetElement(index, out string result))
29             {
30                 Console.Write($"{readMessage}: ");
31                 result = Console.ReadLine();
32             }
33             if (string.IsNullOrEmpty(result))
34             {
35                 return "";
36             }
37             else
38             {
39                 return result.Trim().TrimSingle(' ').Trim();
40             }
41         }
42
43         [Conditional("DEBUG")]
44         public static void Debug(string @string) => Console.WriteLine(@string);
45
46         [Conditional("DEBUG")]
47         public static void Debug(string format, params object[] args) =>
48             ↪ Console.WriteLine(format, args);
49     }
50 }
```

1.3 ./csharp/Platform.IO/FileHelpers.cs

```
1 using System;
2 using System.IO;
3 using System.Runtime.CompilerServices;
4 using Platform.Unsafe;
5
6 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
7
8 namespace Platform.IO
9 {
10     public static class FileHelpers
11     {
12         [MethodImpl(MethodImplOptions.AggressiveInlining)]
13         public static char[] ReadAllChars(string path) => File.ReadAllText(path).ToCharArray();
14
15         [MethodImpl(MethodImplOptions.AggressiveInlining)]
16         public static T[] ReadAll<T>(string path)
17             where T : struct
18         {
19             using var reader = File.OpenRead(path);
20             return reader.ReadAll<T>();
21         }
22
23         [MethodImpl(MethodImplOptions.AggressiveInlining)]
24         public static T ReadFirstOrDefault<T>(string path)
25             where T : struct
26         {
27             using var reader = File.OpenRead(path);
28             return reader.ReadFirstOrDefault<T>();
29         }
30     }
31 }
```

```

27     using var fileStream = GetValidFileStreamOrDefault<T>(path);
28     return fileStream?.ReadOrDefault<T>() ?? default;
29 }
30
31 [MethodImpl(MethodImplOptions.AggressiveInlining)]
32 private static FileStream GetValidFileStreamOrDefault<TStruct>(string path) where
    ↪ TStruct : struct => GetValidFileStreamOrDefault(path, Structure<TStruct>.Size);
33
34 [MethodImpl(MethodImplOptions.AggressiveInlining)]
35 private static FileStream GetValidFileStreamOrDefault(string path, int elementSize)
36 {
37     if (!File.Exists(path))
38     {
39         return null;
40     }
41     var fileSize = GetSize(path);
42     if (fileSize % elementSize != 0)
43     {
44         throw new InvalidOperationException($"File is not aligned to elements with size
            ↪ {elementSize}.");
45     }
46     return fileSize > 0 ? File.OpenRead(path) : null;
47 }
48
49 [MethodImpl(MethodImplOptions.AggressiveInlining)]
50 public static T ReadLastOrDefault<T>(string path)
51     where T : struct
52 {
53     var elementSize = Structure<T>.Size;
54     using var reader = GetValidFileStreamOrDefault(path, elementSize);
55     if (reader == null)
56     {
57         return default;
58     }
59     var totalElements = reader.Length / elementSize;
60     reader.Position = (totalElements - 1) * elementSize; // Set to last element
61     return reader.ReadOrDefault<T>();
62 }
63
64 [MethodImpl(MethodImplOptions.AggressiveInlining)]
65 public static void WriteFirst<T>(string path, T value)
66     where T : struct
67 {
68     using var writer = File.OpenWrite(path);
69     writer.Position = 0;
70     writer.Write(value);
71 }
72
73 [MethodImpl(MethodImplOptions.AggressiveInlining)]
74 public static FileStream Append(string path) => File.Open(path, FileMode.Append,
    ↪ FileAccess.Write);
75
76 [MethodImpl(MethodImplOptions.AggressiveInlining)]
77 public static long GetSize(string path) => File.Exists(path) ? new FileInfo(path).Length
    ↪ : 0;
78
79 [MethodImpl(MethodImplOptions.AggressiveInlining)]
80 public static void SetSize(string path, long size)
81 {
82     using var fileStream = File.Open(path, FileMode.OpenOrCreate);
83     if (fileStream.Length != size)
84     {
85         fileStream.SetLength(size);
86     }
87 }
88
89 [MethodImpl(MethodImplOptions.AggressiveInlining)]
90 public static void DeleteAll(string directory) => DeleteAll(directory, "*");
91
92 [MethodImpl(MethodImplOptions.AggressiveInlining)]
93 public static void DeleteAll(string directory, string searchPattern) =>
    ↪ DeleteAll(directory, searchPattern, SearchOption.TopDirectoryOnly);
94
95 [MethodImpl(MethodImplOptions.AggressiveInlining)]
96 public static void DeleteAll(string directory, string searchPattern, SearchOption
    ↪ searchOption)
97 {
98     foreach (var file in Directory.EnumerateFiles(directory, searchPattern,
    ↪ searchOption))

```

```

99         {
100             File.Delete(file);
101         }
102     }
103 }
104 }

```

1.4 ./csharp/Platform.IO/StreamExtensions.cs

```

1  using System.IO;
2  using System.Runtime.CompilerServices;
3  using Platform.Unsafe;
4
5  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
6
7  namespace Platform.IO
8  {
9      public static class StreamExtensions
10     {
11         [MethodImpl(MethodImplOptions.AggressiveInlining)]
12         public static void Write<T>(this Stream stream, T value)
13             where T : struct
14         {
15             var bytes = value.ToBytes();
16             stream.Write(bytes, 0, bytes.Length);
17         }
18
19         [MethodImpl(MethodImplOptions.AggressiveInlining)]
20         public static T ReadOrDefault<T>(this Stream stream)
21             where T : struct
22         {
23             var size = Structure<T>.Size;
24             var buffer = new byte[size];
25             return stream.Read(buffer, 0, size) == size ? buffer.ToStructure<T>() : default;
26         }
27
28         [MethodImpl(MethodImplOptions.AggressiveInlining)]
29         public static T[] ReadAll<T>(this Stream stream)
30             where T : struct
31         {
32             var size = Structure<T>.Size;
33             var buffer = new byte[size];
34             var elementsLength = stream.Length / size;
35             var elements = new T[elementsLength];
36             for (var i = 0; i < elementsLength; i++)
37             {
38                 stream.Read(buffer, 0, size);
39                 elements[i] = buffer.ToStructure<T>();
40             }
41             return elements;
42         }
43     }
44 }

```

1.5 ./csharp/Platform.IO.Tests/FileHelpersTests.cs

```

1  using System.IO;
2  using Xunit;
3
4  namespace Platform.IO.Tests
5  {
6      public class FileHelpersTests
7      {
8          [Fact]
9          public void WriteReadTest()
10         {
11             var temporaryFile = Path.GetTempFileName();
12             var originalValue = 42UL;
13             FileHelpers.WriteFirst(temporaryFile, originalValue);
14             var readValue = FileHelpers.ReadFirstOrDefault<ulong>(temporaryFile);
15             Assert.Equal(readValue, originalValue);
16             File.Delete(temporaryFile);
17         }
18     }
19 }

```

Index

- ./csharp/Platform.IO.Tests/FileHelpersTests.cs, 4
- ./csharp/Platform.IO/ConsoleCancellation.cs, 1
- ./csharp/Platform.IO/ConsoleHelpers.cs, 1
- ./csharp/Platform.IO/FileHelpers.cs, 2
- ./csharp/Platform.IO/StreamExtensions.cs, 4