

LinksPlatform's Platform.IO Class Library

1.1 ./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 ./ConsoleHelpers.cs

```
1 using System;
2 using System.Diagnostics;
3 using System.Runtime.CompilerServices;
4 using Platform.Collections;
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 ConsoleHelpers
11     {
12         [MethodImpl(MethodImplOptions.AggressiveInlining)]
13         public static void PressAnyKeyToContinue()
14         {
15             Console.WriteLine("Press any key to continue.");
16             Console.ReadKey();
17         }
18
19         [MethodImpl(MethodImplOptions.AggressiveInlining)]
20         public static string GetOrReadArgument(int index, params string[] args) =>
21             ↪ GetOrReadArgument(index, $"{index + 1} argument", args);
22
23         [MethodImpl(MethodImplOptions.AggressiveInlining)]
24         public static string GetOrReadArgument(int index, string readMessage, params string[]
25             ↪ args)
26         {
27             string result;
28             if (args != null && args.Length > index)
29             {
30                 result = args[index];
31             }
32             else
33             {
34                 Console.Write($"{readMessage}: ");
35                 result = Console.ReadLine();
36             }
37             result = (result ?? "").Trim().TrimSingle(' ').Trim();
38             return result;
39         }
40
41         [Conditional("DEBUG")]
42         public static void Debug(string format, params object[] args) =>
43             ↪ Console.WriteLine(format, args);
44     }
45 }
```

1.3 ./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             {
21                 return reader.ReadAll<T>();
22             }
23         }
24
25         [MethodImpl(MethodImplOptions.AggressiveInlining)]
26         public static T ReadFirstOrDefault<T>(string path)
27             where T : struct
28         {
29             using (var fileStream = GetValidFileStreamOrDefault<T>(path))
30             {
31                 return fileStream?.ReadOrDefault<T>() ?? default;
32             }
33         }
34     }
35 }
```

```

32     }
33 }
34
35 [MethodImpl(MethodImplOptions.AggressiveInlining)]
36 private static FileStream GetValidFileStreamOrDefault<TStruct>(string path) where
    ↳ TStruct : struct => GetValidFileStreamOrDefault(path, Structure<TStruct>.Size);
37
38 [MethodImpl(MethodImplOptions.AggressiveInlining)]
39 private static FileStream GetValidFileStreamOrDefault(string path, int elementSize)
40 {
41     if (!File.Exists(path))
42     {
43         return null;
44     }
45     var fileSize = GetSize(path);
46     if (fileSize % elementSize != 0)
47     {
48         throw new InvalidOperationException($"File is not aligned to elements with size
    ↳ {elementSize}.");
49     }
50     return fileSize > 0 ? File.OpenRead(path) : null;
51 }
52
53 [MethodImpl(MethodImplOptions.AggressiveInlining)]
54 public static T ReadLastOrDefault<T>(string path)
55     where T : struct
56 {
57     var elementSize = Structure<T>.Size;
58     using (var reader = GetValidFileStreamOrDefault(path, elementSize))
59     {
60         if (reader == null)
61         {
62             return default;
63         }
64         var totalElements = reader.Length / elementSize;
65         reader.Position = (totalElements - 1) * elementSize; // Set to last element
66         return reader.ReadOrDefault<T>();
67     }
68 }
69
70 [MethodImpl(MethodImplOptions.AggressiveInlining)]
71 public static void WriteFirst<T>(string path, T value)
72     where T : struct
73 {
74     using (var writer = File.OpenWrite(path))
75     {
76         writer.Position = 0;
77         writer.Write(value);
78     }
79 }
80
81 [MethodImpl(MethodImplOptions.AggressiveInlining)]
82 public static FileStream Append(string path) => File.Open(path, FileMode.Append,
    ↳ FileAccess.Write);
83
84 [MethodImpl(MethodImplOptions.AggressiveInlining)]
85 public static long GetSize(string path) => File.Exists(path) ? new FileInfo(path).Length
    ↳ : 0;
86
87 [MethodImpl(MethodImplOptions.AggressiveInlining)]
88 public static void SetSize(string path, long size)
89 {
90     using (var fileStream = File.Open(path, FileMode.OpenOrCreate))
91     {
92         if (fileStream.Length != size)
93         {
94             fileStream.SetLength(size);
95         }
96     }
97 }
98 }
99 }

```

1.4 ./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 }

```

Index

./ConsoleCancellation.cs, 1
./ConsoleHelpers.cs, 1
./FileHelpers.cs, 2
./StreamExtensions.cs, 3