```
LinksPlatform's Platform.IO Class Library
    ./ConsoleCancellation.cs
   using System;
   using System.Runtime.CompilerServices;
2
   using System. Threading; using Platform. Disposables;
4
   using Platform. Threading;
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
   namespace Platform. IO
9
   {
10
        public class ConsoleCancellation : DisposableBase
11
12
            public CancellationTokenSource Source
13
14
                 [MethodImpl(MethodImplOptions.AggressiveInlining)]
15
17
            public CancellationToken Token
19
                 [MethodImpl(MethodImplOptions.AggressiveInlining)]
2.1
                 get;
            }
23
            public bool IsRequested
25
26
                 [MethodImpl(MethodImplOptions.AggressiveInlining)]
27
                 get => Source.IsCancellationRequested;
28
30
            public bool NotRequested
31
32
                 [MethodImpl(MethodImplOptions.AggressiveInlining)]
33
                 get => !Source.IsCancellationRequested;
34
35
36
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
37
            public ConsoleCancellation()
38
39
                 Source = new CancellationTokenSource();
40
                 Token = Source.Token;
41
                 Console.CancelKeyPress += OnCancelKeyPress;
42
43
44
45
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
            public void ForceCancellation() => Source.Cancel();
46
47
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
            public void Wait()
49
50
                 while (NotRequested)
                 {
52
                     ThreadHelpers.Sleep();
53
                 }
54
            }
56
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
            protected override void Dispose(bool manual, bool wasDisposed)
58
59
                 if (!wasDisposed)
60
61
                     Console.CancelKeyPress -= OnCancelKeyPress;
62
                     Source.DisposeIfPossible();
63
                 }
64
            }
65
66
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
67
            private void OnCancelKeyPress(object sender, ConsoleCancelEventArgs e)
68
                 e.Cancel = true;
70
                 if (NotRequested)
                 {
72
                     Source.Cancel();
73
                 }
74
            }
75
        }
76
   }
77
```

```
./ConsoleHelpers.cs
   using System;
   using System.Diagnostics;
using System.Runtime.CompilerServices;
   using Platform.Collections;
4
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
   namespace Platform. IO
        public static class ConsoleHelpers
10
11
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
12
            public static void PressAnyKeyToContinue()
13
                Console.WriteLine("Press any key to continue.");
15
                Console.ReadKey();
16
            }
18
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
19
            public static string GetOrReadArgument(int index, params string[] args) =>
                GetOrReadArgument(index, $\bigs\bigs\cdot\frac{1}{\text{argument}}\) args);
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
22
            public static string GetOrReadArgument(int index, string readMessage, params string[]
                args)
                string result;
25
                if (args != null && args.Length > index)
27
                     result = args[index];
2.8
                }
                else
30
31
                     Console.Write($\frac{\$\"{readMessage}: \");
                     result = Console.ReadLine();
34
                result = (result ?? "").Trim().TrimSingle('"').Trim();
35
                return result;
36
37
            [Conditional("DEBUG")]
39
            public static void Debug(string format, params object[] args) =>
40

→ Console.WriteLine(format, args);
        }
41
   }
42
    ./FileHelpers.cs
   using System;
   using System.IO;
using System.Runtime.CompilerServices;
   using Platform.Unsafe;
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
   namespace Platform.IO
9
        public static class FileHelpers
10
11
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
12
            public static char[] ReadAllChars(string path) => File.ReadAllText(path).ToCharArray();
13
14
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
15
            public static T[] ReadAll<T>(string path)
16
                where T : struct
17
18
                using var reader = File.OpenRead(path);
                return reader.ReadAll<T>();
20
            }
21
22
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
23
            public static T ReadFirstOrDefault<T>(string path)
                where T : struct
25
26
                using var fileStream = GetValidFileStreamOrDefault<T>(path);
                return fileStream?.ReadOrDefault<T>() ?? default;
29
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
```

```
private static FileStream GetValidFileStreamOrDefault<TStruct>(string path) where
32
               TStruct : struct => GetValidFileStreamOrDefault(path, Structure<TStruct>.Size);
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
           private static FileStream GetValidFileStreamOrDefault(string path, int elementSize)
3.5
36
                if (!File.Exists(path))
37
                {
38
                    return null;
39
                }
40
                var fileSize = GetSize(path);
41
                if (fileSize % elementSize != 0)
42
43
                    throw new InvalidOperationException($\B\"File is not aligned to elements with size
44
                    45
                return fileSize > 0 ? File.OpenRead(path) : null;
46
            }
47
48
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
49
           public static T ReadLastOrDefault<T>(string path)
50
                where T : struct
52
                var elementSize = Structure<T>.Size;
                using var reader = GetValidFileStreamOrDefault(path, elementSize);
                if (reader == null)
55
                {
56
                    return default;
57
                }
58
                var totalElements = reader.Length / elementSize;
                reader.Position = (totalElements - 1) * elementSize; // Set to last element
60
                return reader.ReadOrDefault<T>();
61
62
63
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
64
           public static void WriteFirst<T>(string path, T value)
                where T : struct
66
                using var writer = File.OpenWrite(path);
                writer.Position = 0;
69
                writer.Write(value);
70
71
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
73
           public static FileStream Append(string path) => File.Open(path, FileMode.Append,

→ FileAccess.Write);

            [MethodImpl(MethodImplOptions.AggressiveInlining)]
76
           public static long GetSize(string path) => File.Exists(path) ? new FileInfo(path).Length
77
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
            public static void SetSize(string path, long size)
80
81
                using var fileStream = File.Open(path, FileMode.OpenOrCreate);
82
                if (fileStream.Length != size)
                {
84
                    fileStream.SetLength(size);
85
                }
           }
87
       }
88
   }
89
    ./StreamExtensions.cs
1.4
   using System.IO;
1
   using System.Runtime.CompilerServices;
   using Platform.Unsafe;
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
5
   namespace Platform.IO
       public static class StreamExtensions
9
10
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
11
           public static void Write<T>(this Stream stream, T value)
12
13
                where T : struct
            {
14
                var bytes = value.ToBytes();
```

```
stream.Write(bytes, 0, bytes.Length);
16
             }
17
18
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
            public static T ReadOrDefault<T>(this Stream stream)
    where T : struct
20
21
22
                 var size = Structure<T>.Size;
23
                 var buffer = new byte[size];
                 return stream.Read(buffer, 0, size) == size ? buffer.ToStructure<T>() : default;
25
26
^{27}
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
28
            public static T[] ReadAll<T>(this Stream stream)
   where T : struct
29
30
31
                 var size = Structure<T>.Size;
32
                 var buffer = new byte[size];
33
                 var elementsLength = stream.Length / size;
                 var elements = new T[elementsLength];
                 for (var i = 0; i < elementsLength; i++)</pre>
36
37
                      stream.Read(buffer, 0, size);
                     elements[i] = buffer.ToStructure<T>();
39
40
41
                 return elements;
            }
42
        }
43
   }
44
```

Index

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