LinksPlatform's Platform.IO Class Library

./ConsoleCancellation.cs

```csharp
using System;
using System.Threading;
using Platform.Disposables;
using Platform.Threading;

#pragma warning disable CS1591 // Missing XML comment for publicly visible type or member

namespace Platform.IO
{
    public class ConsoleCancellation : DisposableBase
    {
        public CancellationTokenSource Source { get; }

        public CancellationToken Token { get; }

        public bool IsRequested => Source.IsCancellationRequested;

        public bool NotRequested => !Source.IsCancellationRequested;

        public ConsoleCancellation()
        {
            Source = new CancellationTokenSource();
            Token = Source.Token;
            Console.CancelKeyPress += OnCancelKeyPress;
        }

        public void ForceCancellation() => Source.Cancel();

        public void Wait()
        {
            while (NotRequested)
            {
                ThreadHelpers.Sleep();
            }
        }

        protected override void Dispose(bool manual, bool wasDisposed)
        {
            if (!wasDisposed)
            {
                Console.CancelKeyPress -= OnCancelKeyPress;
                Source.DisposeIfPossible();
            }
        }

        private void OnCancelKeyPress(object sender, ConsoleCancelEventArgs e)
        {
            e.Cancel = true;
            if (NotRequested)
            {
                Source.Cancel();
            }
        }
    }
}
```

./ConsoleHelpers.cs

```csharp
using System;
using System.Diagnostics;
using Platform.Collections;

#pragma warning disable CS1591 // Missing XML comment for publicly visible type or member

namespace Platform.IO
{
    public static class ConsoleHelpers
    {
        public static void PressAnyKeyToContinue()
        {
            Console.WriteLine("Press any key to continue.");
            Console.ReadKey();
        }

        public static string GetOrReadArgument(int index, params string[] args) =>
            GetOrReadArgument(index, $"{index + 1} argument", args);

        public static string GetOrReadArgument(int index, string readMessage, params string[]
            args)
```

```csharp
20          {
21              string result;
22              if (args != null && args.Length > index)
23              {
24                  result = args[index];
25              }
26              else
27              {
28                  Console.Write($"{readMessage}: ");
29                  result = Console.ReadLine();
30              }
31              result = (result ?? "").Trim().TrimSingle('"').Trim();
32              return result;
33          }
34
35          [Conditional("DEBUG")]
36          public static void Debug(string format, params object[] args) =>
   ↪    Console.WriteLine(format, args);
37      }
38  }
```

./FileHelpers.cs

```csharp
1   using System;
2   using System.IO;
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 FileHelpers
10      {
11          public static char[] ReadAllChars(string path) => File.ReadAllText(path).ToCharArray();
12
13          public static T[] ReadAll<T>(string path)
14              where T : struct
15          {
16              using (var reader = File.OpenRead(path))
17              {
18                  return reader.ReadAll<T>();
19              }
20          }
21
22          public static T ReadFirstOrDefault<T>(string path)
23              where T : struct
24          {
25              using (var fileStream = GetValidFileStreamOrDefault<T>(path))
26              {
27                  return fileStream?.ReadOrDefault<T>() ?? default;
28              }
29          }
30
31          private static FileStream GetValidFileStreamOrDefault<TStruct>(string path) where
   ↪    TStruct : struct => GetValidFileStreamOrDefault(path, Structure<TStruct>.Size);
32
33          private static FileStream GetValidFileStreamOrDefault(string path, int elementSize)
34          {
35              if (!File.Exists(path))
36              {
37                  return null;
38              }
39              var fileSize = GetSize(path);
40              if (fileSize % elementSize != 0)
41              {
42                  throw new InvalidOperationException($"File is not aligned to elements with size
   ↪    {elementSize}.");
43              }
44              return fileSize > 0 ? File.OpenRead(path) : null;
45          }
46
47          public static T ReadLastOrDefault<T>(string path)
48              where T : struct
49          {
50              var elementSize = Structure<T>.Size;
51              using (var reader = GetValidFileStreamOrDefault(path, elementSize))
52              {
53                  if (reader == null)
54                  {
55                      return default;
```

```csharp
56                  }
57                  var totalElements = reader.Length / elementSize;
58                  reader.Position = (totalElements - 1) * elementSize; // Set to last element
59                  return reader.ReadOrDefault<T>();
60              }
61          }
62
63          public static void WriteFirst<T>(string path, T value)
64              where T : struct
65          {
66              using (var writer = File.OpenWrite(path))
67              {
68                  writer.Position = 0;
69                  writer.Write(value);
70              }
71          }
72
73          public static FileStream Append(string path) => File.Open(path, FileMode.Append,
        ↪  FileAccess.Write);
74
75          public static long GetSize(string path) => File.Exists(path) ? new FileInfo(path).Length
        ↪  : 0;
76
77          public static void SetSize(string path, long size)
78          {
79              using (var fileStream = File.Open(path, FileMode.OpenOrCreate))
80              {
81                  if (fileStream.Length != size)
82                  {
83                      fileStream.SetLength(size);
84                  }
85              }
86          }
87      }
88  }
```

./StreamExtensions.cs

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

Index