```
LinksPlatform's Platform. Unsafe Class Library
./Platform.Unsafe/ByteArrayExtensions.cs
   using Platform.Exceptions;
using Platform.Collections
2
   using System.Runtime.CompilerServices;
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
   namespace Platform. Unsafe
8
        public unsafe static class ByteArrayExtensions
9
10
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
11
            public static TStruct ToStructure<TStruct>(this byte[] bytes)
                where TStruct : struct
13
14
                Ensure.OnDebug.ArgumentNotEmpty(bytes, nameof(bytes));
                var structureSize = System.Runtime.CompilerServices.Unsafe.SizeOf<TStruct>();
16
                Ensure.OnDebug.ArgumentMeetsCriteria(bytes, array => array.Length == structureSize,
17

ightharpoonup nameof(bytes), "Bytes array should be the same length as struct size.");
                TStruct structure = default;
                fixed (byte* pointer = bytes)
19
20
21
                     System.Runtime.CompilerServices.Unsafe.Copy(ref structure, pointer);
22
                return structure;
            }
24
        }
25
26
./Platform.Unsafe/IntPtr.cs
   using System;
using System.Reflection;
1
   using System.Runtime.InteropServices;
   using Platform. Reflection;
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
   namespace Platform. Unsafe
8
9
        /// <remarks>
10
        ///\ {\tt Please\ use\ System.Runtime.CompilerServices.Unsafe\ instead.}
11
        /// </remarks>
12
        public static class IntPtr<T>
14
            public static readonly Func<IntPtr, T> GetValue;
            public static readonly Action<IntPtr, T> SetValue;
16
17
            static IntPtr()
18
19
                GetValue = CompileGetValueDelegate();
20
                SetValue = CompileSetValueDelegate();
21
23
            static private Func<IntPtr, T> CompileGetValueDelegate()
24
25
26
                return DelegateHelpers.Compile<Func<IntPtr, T>>(emiter =>
27
                        (NumericType<T>.IsNumeric)
                         emiter.LoadArgument(0)
30
                         emiter.LoadIndirect<T>();
31
                         emiter.Return();
                     }
33
                     else
                     {
35
                         emiter.LoadArguments(0);
36
                         emiter.Call(typeof(Marshal).GetGenericMethod(nameof(Marshal.PtrToStructure),
37
                             Types<T>.Array, Types<IntPtr, Type, bool>.Array));
                         emiter.Return();
                     }
39
                });
40
            }
41
42
            static private Action<IntPtr, T> CompileSetValueDelegate()
43
                return DelegateHelpers.Compile<Action<IntPtr, T>>(emiter =>
45
46
                     if (NumericType<T>.IsNumeric)
```

```
emiter.LoadArguments(0, 1);
                        emiter.StoreIndirect<T>();
50
                        emiter.Return();
51
                    else
53
54
                        emiter.LoadArguments(0, 1);
                        emiter.LoadConstant(true);
56
                        emiter.Call(typeof(Marshal).GetTypeInfo().GetMethod(nameof(Marshal.Structure
                            ToPtr), Types<object, IntPtr,
                           bool>.Array));
                        emiter.Return();
                    }
59
               });
60
           }
61
       }
62
63
./Platform.Unsafe/IntPtrExtensions.cs
   using System;
using System.Runtime.CompilerServices;
   using Platform. Numbers;
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
6
   namespace Platform. Unsafe
   {
        /// <remarks>
9
        /// Please use System.Runtime.CompilerServices.Unsafe instead.
10
        /// </remarks>
11
       public unsafe static class IntPtrExtensions
12
13
            [Obsolete("GetValue method is deprecated, please use
       System.Runtime.CompilerServices.Unsafe.Read method instead.")]
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
1.5
            public static TElement GetValue<TElement>(this IntPtr pointer) =>
16
               IntPtr<TElement>.GetValue(pointer);
17
            [Obsolete("SetValue method is deprecated, please use
18
       System.Runtime.CompilerServices.Unsafe.Write method instead.")]
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
19
            public static void SetValue<TElement>(this IntPtr pointer, TElement value) =>
               IntPtr<TElement>.SetValue(pointer, value);
21
            [Obsolete("GetElement method is deprecated, please use
22
       System.Runtime.CompilerServices.Unsafe.Add method instead.")]
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
23
            public static IntPtr GetElement(this IntPtr pointer, int elementSize, int index) =>
24
            → pointer + (elementSize * index);
            [Obsolete("GetElement method is deprecated, please use
26
       System.Runtime.CompilerServices.Unsafe.Add method instead.")]
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
27
            public static IntPtr GetElement(this IntPtr pointer, long elementSize, long index) =>
28
               new IntPtr((byte*)pointer.ToPointer() + (elementSize * index));
29
            [Obsolete("GetElement method is deprecated, please use
30
       System.Runtime.CompilerServices.Unsafe.Add method instead.")]
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
            public static IntPtr GetElement<TIndex>(this IntPtr pointer, int elementSize, TIndex
               index) => pointer.GetElement((long)elementSize, (Integer)(Integer<TIndex>)index);
33
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
34
            public static void WriteElementValue<TValue>(this IntPtr pointer, long index, TValue
                value) => System.Runtime.CompilerServices.Unsafe.Write((byte*)pointer +
                (System.Runtime.CompilerServices.Unsafe.SizeOf<TValue>() * index), value);
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
37
            public static TValue ReadElementValue<TValue>(this IntPtr pointer, long index) =>
38
               System.Runtime.CompilerServices.Unsafe.Read<TValue>((byte*)pointer +
               (System.Runtime.CompilerServices.Unsafe.SizeOf<TValue>() * index));
       }
39
./Platform.Unsafe/MemoryBlock.cs
   using System.Collections.Concurrent;
   using System. Threading. Tasks;
```

```
#pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
5
   namespace Platform. Unsafe
   {
7
       public static unsafe class MemoryBlock
9
            public static void Zero(void* pointer, long capacity)
10
                Parallel.ForEach(Partitioner.Create(0, capacity), range =>
12
                {
13
                    var from = range.Item1;
14
                    var offset = (void*)((byte*)pointer + from);
15
                    var length = (uint)(range.Item2 - from);
16
                    System.Runtime.CompilerServices.Unsafe.InitBlock(offset, 0, length);
17
                });
18
            }
19
       }
20
21
./Platform.Unsafe/Structure.cs
   using System;
   using System.Runtime.InteropServices;
3
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
4
   namespace Platform. Unsafe
       public static class Structure<TStruct>
            where TStruct : struct
9
10
            /// <summary>
11
            /// <para>
12
            /// Returns the size of an unmanaged type in bytes.
13
            /// This property do this without throwing exceptions for generic types as <see
14
               cref="Marshal.SizeOf{T}()"/> and <see cref="Marshal.SizeOf(Type)"/> do.
            /// </para>
15
            /// <para>
16
            /// Возвращает размер неуправляемого типа в байтах.
17
            /// Этот свойство делает это без выбрасывания исключений для универсальных типов, как
               это делают <see cref="Marshal.SizeOf{T}()"/> и <see cref="Marshal.SizeOf(Type)"/>.
            /// </para>
            /// </summary>
20
            public static int Size { get; } =
21
            System.Runtime.CompilerServices.Unsafe.SizeOf<TStruct>();
       }
22
   }
23
./Platform.Unsafe/StructureExtensions.cs
   using System.Runtime.CompilerServices;
   using Platform. Hardware. Cpu;
3
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
   namespace Platform. Unsafe
       public unsafe static class StructureExtensions
9
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
10
            public static byte[] ToBytes<TStruct>(this ref TStruct obj)
1.1
                where TStruct : struct
12
            {
13
                var structureSize = System.Runtime.CompilerServices.Unsafe.SizeOf<TStruct>();
14
                var bytes = new byte[structureSize];
                fixed (byte* pointer = bytes)
16
                {
17
                    obj.CopyTo(pointer, structureSize);
19
                return bytes;
20
            }
21
22
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
            public static void CopyTo<TStruct>(this ref TStruct source, void* destination)
24
                where TStruct : struct
25
            {
26
                var size = System.Runtime.CompilerServices.Unsafe.SizeOf<TStruct>();
27
                CopyTo(ref source, destination, size);
28
            }
```

```
[MethodImpl(MethodImplOptions.AggressiveInlining)]
31
            public static void CopyTo<TStruct>(this ref TStruct source, void* destination, int size)
                 where TStruct : struct
33
                 if (CacheLine.Size >= size)
35
                 {
36
                     System.Runtime.CompilerServices.Unsafe.Copy(destination, ref source);
37
                 }
                 else
39
                 {
40
                     System.Runtime.CompilerServices.Unsafe.CopyBlock(destination,
41

→ System.Runtime.CompilerServices.Unsafe.AsPointer(ref source), (uint)size);

                 }
42
            }
43
        }
44
   }
45
./Platform.Unsafe.Tests/IntPtrExtensionsTests.cs\\
   using System;
   using System.Runtime.InteropServices;
   using Xunit;
using Xunit.Abstractions;
4
   using Platform.Diagnostics;
   namespace Platform. Unsafe. Tests
9
        public unsafe class IntPtrExtensionsTests
10
            private const int N = 10000000;
11
12
            private readonly ITestOutputHelper _output;
13
14
            public IntPtrExtensionsTests(ITestOutputHelper output)
15
16
                 _output = output;
18
19
            [Fact]
20
            public void ReadAndWriteOperationsForPointerValuesDelegatesTest()
21
                 var pointer = Marshal.AllocHGlobal(sizeof(ulong));
23
                 ulong result = default;
24
                 for (var i = 0; i < N; i++)</pre>
25
                 {
26
                     result = Delegates(pointer);
27
                 Assert.Equal(42UL, result);
29
                Marshal.FreeHGlobal(pointer);
30
31
32
            private static ulong Delegates(IntPtr pointer)
33
                 ulong result;
35
                 IntPtr<ulong>.SetValue(pointer, 42UL);
36
                 result = IntPtr<ulong>.GetValue(pointer);
37
                 return result;
38
            }
39
40
            [Fact]
41
            public void ReadAndWriteOperationsForPointerValuesExtensionMethodsTest()
42
43
                 var pointer = Marshal.AllocHGlobal(sizeof(ulong));
                ulong result = default;
45
                 for (var i = 0; i < N; i++)</pre>
46
                 {
47
                     result = ExtensionMethods(pointer);
48
49
                 Assert.Equal(42UL, result);
                Marshal.FreeHGlobal(pointer);
51
52
53
            private static ulong ExtensionMethods(IntPtr pointer)
54
55
                ulong result;
56
                pointer.SetValue(42UL);
57
                 result = pointer.GetValue<ulong>();
                 return result;
59
            }
60
61
            [Fact]
```

```
public void ReadAndWriteOperationsForPointerValuesUnsafeClassMethodsTest()
    void* pointer = (void*)Marshal.AllocHGlobal(sizeof(ulong));
    ulong result = default;
    for (var i = 0; i < N; i++)</pre>
        result = ReadAndWriteMethods(pointer);
    Assert.Equal(42UL, result);
    Marshal.FreeHGlobal((IntPtr)pointer);
private static ulong ReadAndWriteMethods(void* pointer)
    ulong result;
    System.Runtime.CompilerServices.Unsafe.Write(pointer, 42UL);
    result = System.Runtime.CompilerServices.Unsafe.Read<ulong>(pointer);
    return result;
}
[Fact]
public void ReadAndWriteOperationsComparisionTest()
    var t1 = Performance.Measure(ReadAndWriteOperationsForPointerValuesDelegatesTest);
    var t2 =
    Performance.Measure(ReadAndWriteOperationsForPointerValuesExtensionMethodsTest);
    var t3 = Performance.Measure(ReadAndWriteOperationsForPointerValuesUnsafeClassMethod)

    sTest):

    var message = $\"\{t1\} \{t2\} \{t3\}\";
    _output.WriteLine(message);
}
[Fact]
public void ElementOffsetOperationsForPointerValuesExtensionMethods()
    var pointer = Marshal.AllocHGlobal(sizeof(ulong) * 10);
    ulong result = default;
    for (var i = 0; i < N; i++)</pre>
        result = GetElementExtensionMethods(pointer);
    Assert.Equal(5UL * 8UL, result - (ulong)pointer);
    Marshal.FreeHGlobal(pointer);
private static ulong GetElementExtensionMethods(IntPtr pointer)
    ulong result;
    result = (ulong)pointer.GetElement(8, 5);
    return result;
}
[Fact]
public void ElementOffsetOperationsForPointerValuesUnsafeClassMethodsTest()
    void* pointer = (void*)Marshal.AllocHGlobal(sizeof(ulong) * 10);
    ulong result = default;
    for (var i = 0; i < N; i++)</pre>
        result = GetElementMethods(pointer);
    Assert.Equal(5UL * 8UL, result - (ulong)pointer);
    Marshal.FreeHGlobal((IntPtr)pointer);
}
private static ulong GetElementMethods(void* pointer)
    ulong result;
    result = (ulong)System.Runtime.CompilerServices.Unsafe.Add<ulong>(pointer, 5);
    return result;
}
[Fact]
public void GetElementOperationsComparisionTest()
    var t1 =
    Performance.Measure(ElementOffsetOperationsForPointerValuesExtensionMethods);
    var t2 = Performance.Measure(ElementOffsetOperationsForPointerValuesUnsafeClassMetho |

    dsTest):
```

63

65

67 68

69 70

7.1

72 73 74

75 76

77

79 80

81 82

84 85

87

88

91

93

94 95

97

98

100 101

103 104 105

106 107

108

109 110

111 112

113

114 115

116

118 119

 $\frac{120}{121}$

122

123

125

 $\frac{126}{127}$

129 130

 $\frac{131}{132}$

134 135

136

137

```
var message = $\"\{t1\} \{t2\}\";
138
                 _output.WriteLine(message);
139
             }
140
        }
141
142
./Platform.Unsafe.Tests/SizeOfTests.cs
    using System.Runtime.InteropServices;
    using Xunit;
 2
    namespace Platform. Unsafe. Tests
         public static class SizeOfTests
 6
             public struct X<T>
                 public readonly T F1;
public readonly T F2;
1.0
11
12
             [Fact]
14
             public static void UnsafeClassSizeOfTest()
15
16
                 var size = System.Runtime.CompilerServices.Unsafe.SizeOf<X<int>>();
17
                 Assert.Equal(8, size);
18
             }
19
20
             [Fact]
21
             public static void MarshalSizeOfTest()
22
23
                 var size = Marshal.SizeOf(default(X<int>));
24
                 Assert.Equal(8, size);
25
             }
27
             [Fact]
             public static void StructurePropertyTest()
29
30
                 var size = Structure<X<int>>.Size;
                 Assert.Equal(8, size);
32
33
         }
34
35
./Platform.Unsafe.Tests/StructAndBytesConversionTests.cs
    using Xunit;
 1
    namespace Platform. Unsafe. Tests
 3
 4
         \underline{\textbf{public static class}} \ \texttt{StructAndBytesConversionTests}
 5
             [Fact]
             public static void StructToBytesTest()
                 ulong source = ulong.MaxValue;
10
                 var result = source.ToBytes();
11
                 for (int i = 0; i < result.Length; i++)</pre>
12
                 {
                      Assert.Equal(byte.MaxValue, result[i]);
14
                 }
15
             }
17
             [Fact]
18
             public static void BytesToStructTest()
19
20
                 byte[] bytes = new[] { byte.MaxValue, byte.MaxValue, byte.MaxValue, byte.MaxValue,
21
                  byte.MaxValue, byte.MaxValue, byte.MaxValue };
                 ulong result = bytes.ToStructure<ulong>();
                 Assert.Equal(ulong.MaxValue, result);
23
             }
24
        }
    }
26
./Platform.Unsafe.Tests/ZeroMemoryTests.cs
    using Xunit;
 2
    namespace Platform. Unsafe. Tests
 3
         public static unsafe class ZeroMemoryTests
 5
```

```
public static void ZeroMemoryTest()
{
9
                  var bytes = new byte[1024];
for (int i = 0; i < bytes.Length; i++)</pre>
10
                   {
12
                       bytes[i] = unchecked((byte)i);
13
                   }
14
                  fixed (byte* pointer = bytes)
15
                   {
16
                       MemoryBlock.Zero(pointer, bytes.Length);
17
18
                  for (int i = 0; i < bytes.Length; i++)</pre>
19
20
21
                        Assert.Equal(0, bytes[i]);
                  }
^{22}
             }
23
         }
^{24}
   }
```

Index

```
./Platform.Unsafe.Tests/IntPtrExtensionsTests.cs, 4
./Platform.Unsafe.Tests/SizeOfTests.cs, 6
./Platform.Unsafe.Tests/StructAndBytesConversionTests.cs, 6
./Platform.Unsafe.Tests/ZeroMemoryTests.cs, 6
./Platform.Unsafe/ByteArrayExtensions.cs, 1
./Platform.Unsafe/IntPtr.cs, 1
./Platform.Unsafe/IntPtrExtensions.cs, 2
./Platform.Unsafe/MemoryBlock.cs, 2
./Platform.Unsafe/Structure.cs, 3
./Platform.Unsafe/StructureExtensions.cs, 3
```