```
LinksPlatform's Platform. Unsafe Class Library
     ./Platform.Unsafe/ByteArrayExtensions.cs
   using Platform.Exceptions;
   using Platform.Collections
2
   using System.Runtime.CompilerServices;
   using static System.Runtime.CompilerServices.Unsafe;
4
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
6
   namespace Platform.Unsafe
8
        public unsafe static class ByteArrayExtensions
10
11
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
12
            public static TStruct ToStructure<TStruct>(this byte[] bytes)
13
                where TStruct : struct
15
                Ensure.OnDebug.ArgumentNotEmpty(bytes, nameof(bytes));
16
                var structureSize = SizeOf<TStruct>();
                Ensure.OnDebug.ArgumentMeetsCriteria(bytes, array => array.Length == structureSize,
                → nameof(bytes), "Bytes array should be the same length as struct size."); TStruct structure = default;
19
                fixed (byte* pointer = bytes)
21
                    Copy(ref structure, pointer);
22
24
                return structure:
            }
25
       }
26
   }
27
    ./Platform.Unsafe/IntPtr.cs
   using System;
   using System. Reflection;
   using System.Runtime.CompilerServices;
   using
         System.Runtime.InteropServices;
   using Platform.Reflection;
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
7
9
   namespace Platform.Unsafe
10
        /// <remarks>
11
        /// Please use System.Runtime.CompilerServices.Unsafe instead.
12
13
         // </remarks>
        [Obsolete("Please use System.Runtime.CompilerServices.Unsafe instead.")]
14
       public static class IntPtr<T>
15
            public static readonly Func<IntPtr, T> GetValue = CompileGetValueDelegate();
17
            public static readonly Action<IntPtr, T> SetValue = CompileSetValueDelegate();
18
19
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
20
            static private Func<IntPtr, T> CompileGetValueDelegate()
22
                return DelegateHelpers.Compile<Func<IntPtr, T>>(emiter =>
23
24
                       (NumericType<T>.IsNumeric)
25
26
                         emiter.LoadArgument(0);
27
                         emiter.LoadIndirect<T>();
                         emiter.Return();
30
                    else
32
                         emiter.LoadArguments(0);
33
                         emiter.Call(typeof(Marshal).GetGenericMethod(nameof(Marshal.PtrToStructure),
34
                         → Types<T>.Array, Types<IntPtr, Type, bool>.Array));
                         emiter.Return();
                    }
36
                });
37
            }
39
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
40
41
            static private Action<IntPtr, T> CompileSetValueDelegate()
42
                return DelegateHelpers.Compile<Action<IntPtr, T>>(emiter =>
43
45
                    if (NumericType<T>.IsNumeric)
46
```

```
emiter.LoadArguments(0, 1);
                        emiter.StoreIndirect<T>();
                        emiter.Return();
49
50
                    else
52
                        emiter.LoadArguments(0, 1);
53
                        emiter.LoadConstant(true);
                        emiter.Call(typeof(Marshal).GetTypeInfo().GetMethod(nameof(Marshal.Structure)
55
                            ToPtr), Types<object, IntPtr,
                            bool>.Array));
                        emiter.Return();
56
                    }
57
               });
58
           }
59
       }
60
61
     ./Platform.Unsafe/IntPtrExtensions.cs
1.3
   using System;
   using System.Runtime.CompilerServices;
   using Platform. Numbers;
3
   using static System.Runtime.CompilerServices.Unsafe;
4
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
   namespace Platform. Unsafe
   {
        /// <remarks>
10
        /// Please use System.Runtime.CompilerServices.Unsafe instead.
11
       /// </remarks>
12
       public unsafe static class IntPtrExtensions
13
14
            [Obsolete("GetValue method is deprecated, please use
15
       System.Runtime.CompilerServices.Unsafe.Read method instead.")]
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
16
            public static TElement GetValue<TElement>(this IntPtr pointer) =>
17
               IntPtr<TElement>.GetValue(pointer);
18
            [Obsolete("SetValue method is deprecated, please use
19
       System.Runtime.CompilerServices.Unsafe.Write method instead.")]
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
2.0
            public static void SetValue<TElement>(this IntPtr pointer, TElement value) =>
21
               IntPtr<TElement>.SetValue(pointer, value);
22
            [Obsolete("GetElement method is deprecated, please use
23
       System.Runtime.CompilerServices.Unsafe.Add method instead.")]
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
24
            public static IntPtr GetElement(this IntPtr pointer, int elementSize, int index) =>
               pointer + (elementSize * index);
26
            [Obsolete("GetElement method is deprecated, please use
27
       System.Runtime.CompilerServices.Unsafe.Add method instead.")]
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
28
            public static IntPtr GetElement(this IntPtr pointer, long elementSize, long index) =>
29
            → new IntPtr((byte*)pointer.ToPointer() + (elementSize * index));
30
            [Obsolete("GetElement method is deprecated, please use
31
       System.Runtime.CompilerServices.Unsafe.Add method instead.")]
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
32
            public static IntPtr GetElement<TIndex>(this IntPtr pointer, int elementSize, TIndex
33
               index) => pointer.GetElement((long)elementSize, (Integer)(Integer<TIndex>)index);
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
35
            public static void WriteElementValue<TValue>(this IntPtr pointer, long index, TValue
36
            value) => Write((byte*)pointer + (SizeOf<TValue>() * index), value);
37
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
38
           public static TValue ReadElementValue<TValue>(this IntPtr pointer, long index) =>
            Read<TValue>((byte*)pointer + (SizeOf<TValue>() * index));
41
    ./Platform.Unsafe/MemoryBlock.cs
   using System.Collections.Concurrent;
   using System.Runtime.CompilerServices;
   using System. Threading. Tasks;
   using static System.Runtime.CompilerServices.Unsafe;
```

```
#pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
   namespace Platform. Unsafe
   {
9
       public static unsafe class MemoryBlock
10
11
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
12
           public static void Zero(void* pointer, long capacity) => InitBlock(pointer, 0,
            14
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
15
           public static void ParallelZero(void* pointer, long capacity) =>
               Parallel.ForEach(Partitioner.Create(0, capacity), range => ZeroBlock(pointer,
               range.Item1, range.Item2));
17
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
18
           private static void ZeroBlock(void* pointer, long from, long to)
19
20
                var offset = (void*)((byte*)pointer + from);
                var length = (uint)(to - from);
                InitBlock(offset, 0, length);
23
            }
       }
   }
26
    ./Platform.Unsafe/Structure.cs
1.5
   using System;
   using System.Runtime.CompilerServices;
   using System.Runtime.InteropServices;
3
   using static System.Runtime.CompilerServices.Unsafe;
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
   namespace Platform.Unsafe
9
       public static class Structure<TStruct>
10
           where TStruct : struct
11
            /// <summary>
13
            /// <para>
14
            /// Returns the size of an unmanaged type in bytes.
15
            /// This property do this without throwing exceptions for generic types as <see
               cref="Marshal.SizeOf(T)()"/> and <see cref="Marshal.SizeOf(Type)"/> do.
            /// </para>
            /// <para>
18
            /// Возвращает размер неуправляемого типа в байтах.
19
            /// 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>
21
            /// </summary>
22
           public static int Size
23
2.4
                [MethodImpl(MethodImplOptions.AggressiveInlining)]
26
            } = SizeOf<TStruct>();
27
       }
28
   }
29
     ./Platform.Unsafe/StructureExtensions.cs
   using System.Runtime.CompilerServices;
   using Platform.Hardware.Cpu;
   using static System.Runtime.CompilerServices.Unsafe;
3
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
   namespace Platform.Unsafe
7
       public unsafe static class StructureExtensions
9
10
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
           public static byte[] ToBytes<TStruct>(this ref TStruct obj)
12
                where TStruct : struct
13
14
                var structureSize = SizeOf<TStruct>();
15
                var bytes = new byte[structureSize];
16
                fixed (byte* pointer = bytes)
18
                    obj.CopyTo(pointer, structureSize);
```

```
20
                 return bytes;
21
22
23
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
24
            public static void CopyTo<TStruct>(this ref TStruct source, void* destination)
25
                 where TStruct : struct
26
27
                 var size = SizeOf<TStruct>();
                 CopyTo(ref source, destination, size);
29
            }
30
31
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
32
            public static void CopyTo<TStruct>(this ref TStruct source, void* destination, int size)
33
                 where TStruct : struct
34
35
                 if (CacheLine.Size >= size)
36
                 {
37
                     Copy(destination, ref source);
38
                 }
39
                 else
40
                 {
41
                     CopyBlock(destination, AsPointer(ref source), (uint)size);
42
                 }
43
            }
44
        }
45
   }
    ./Platform.Unsafe.Tests/IntPtrExtensionsTests.cs
   using System;
   using System.Runtime.InteropServices;
using Xunit;
   using static System.Runtime.CompilerServices.Unsafe;
   namespace Platform. Unsafe. Tests
        public unsafe class IntPtrExtensionsTests
9
            [Fact]
10
            public void ReadAndWriteOperationsForPointerValuesUnsafeClassMethodsTest()
11
12
                 void* pointer = (void*)Marshal.AllocHGlobal(sizeof(ulong));
13
                Write(pointer, 42UL);
14
                 Assert.Equal(42UL, Read<ulong>(pointer));
15
                Marshal.FreeHGlobal((IntPtr)pointer);
17
18
            [Fact]
19
            public void ElementOffsetOperationsForPointerValuesTest()
20
                 void* pointer = (void*)Marshal.AllocHGlobal(sizeof(ulong) * 10);
22
                 ulong result = (ulong)Add<ulong>(pointer, 5);
23
                 Assert.Equal(5UL * 8UL, result - (ulong)pointer);
24
                 Marshal.FreeHGlobal((IntPtr)pointer);
25
            }
26
        }
27
    ./Platform.Unsafe.Tests/SizeOfTests.cs
1.8
   using System.Runtime.InteropServices;
   using Xunit;
2
   namespace Platform. Unsafe. Tests
4
        public static class SizeOfTests
6
7
            public struct X<T>
                public readonly T F1;
public readonly T F2;
10
11
            }
12
13
            [Fact]
            public static void UnsafeClassSizeOfTest()
15
16
                 var size = System.Runtime.CompilerServices.Unsafe.SizeOf<X<int>>();
17
                 Assert.Equal(8, size);
            }
19
```

```
[Fact]
21
            public static void MarshalSizeOfTest()
23
                 var size = Marshal.SizeOf(default(X<int>));
24
                 Assert.Equal(8, size);
            }
26
27
            [Fact]
28
            public static void StructurePropertyTest()
29
30
                 var size = Structure<X<int>>.Size;
31
                 Assert.Equal(8, size);
32
33
            }
        }
34
   }
35
     ./Platform.Unsafe.Tests/StructAndBytesConversionTests.cs
1.9
   using Xunit;
   namespace Platform.Unsafe.Tests
3
4
        public static class StructAndBytesConversionTests
5
6
            [Fact]
            public static void StructToBytesTest()
9
10
                 ulong source = ulong.MaxValue;
                var result = source.ToBytes();
11
                for (int i = 0; i < result.Length; i++)</pre>
12
13
                     Assert.Equal(byte.MaxValue, result[i]);
                 }
15
            }
16
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>();
22
23
                 Assert.Equal(ulong.MaxValue, result);
            }
^{24}
        }
25
   }
1.10
     ./Platform.Unsafe.Tests/ZeroMemoryTests.cs
   using Xunit;
   namespace Platform. Unsafe. Tests
3
4
        public static unsafe class ZeroMemoryTests
5
            [Fact]
            public static void ZeroMemoryTest()
                 var bytes = new byte[1024];
10
                for (int i = 0; i < bytes.Length; i++)</pre>
11
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
                 {
                     Assert.Equal(0, bytes[i]);
21
                 }
22
            }
23
24
            [Fact]
25
            public static void ParallelZeroMemoryTest()
26
27
                 var bytes = new byte[1024];
28
29
                 for (int i = 0; i < bytes.Length; i++)</pre>
30
                     bytes[i] = unchecked((byte)i);
31
                 }
```

```
fixed (byte* pointer = bytes)
33
                     MemoryBlock.ParallelZero(pointer, bytes.Length);
35
36
                for (int i = 0; i < bytes.Length; i++)</pre>
38
                     Assert.Equal(0, bytes[i]);
39
                }
40
           }
        }
^{42}
   }
43
```

Index

```
./Platform.Unsafe.Tests/IntPtrExtensionsTests.cs, 4
./Platform.Unsafe.Tests/SizeOfTests.cs, 4
./Platform.Unsafe.Tests/StructAndBytesConversionTests.cs, 5
./Platform.Unsafe.Tests/ZeroMemoryTests.cs, 5
./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
```