```
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
9
       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
                Ensure.OnDebug.ArgumentMeetsCriteria(bytes, HasSameSizeAs<TStruct>, nameof(bytes),
                → "Bytes array should be the same length as struct size.");
                TStruct structure = default;
18
                fixed (byte* pointer = bytes)
19
                    Copy(ref structure, pointer);
                }
22
                return structure;
24
25
            private static bool HasSameSizeAs<TStruct>(byte[] array) where TStruct : struct =>
26
            → array.Length == Structure<TStruct>.Size;
   }
28
     ./Platform.Unsafe/IntPtrExtensions.cs
1.2
   using System;
using System.Runtime.CompilerServices;
   using static System.Runtime.CompilerServices.Unsafe;
4
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
5
   namespace Platform.Unsafe
       public unsafe static class IntPtrExtensions
10
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
11
            public static void WriteElementValue<TValue>(this IntPtr pointer, long index, TValue
12
               value) => Write((byte*)pointer + (SizeOf<TValue>() * index), value);
13
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
14
            public static TValue ReadElementValue<TValue>(this IntPtr pointer, long index) =>
15
               Read<TValue>((byte*)pointer + (SizeOf<TValue>() * index));
       }
16
   }
17
     ./Platform.Unsafe/MemoryBlock.cs
1.3
   using System;
   using System.Collections.Concurrent;
2
   using System.Runtime.CompilerServices;
   using System. Threading. Tasks;
4
   using static System.Runtime.CompilerServices.Unsafe;
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
   namespace Platform.Unsafe
9
   {
10
       public static unsafe class MemoryBlock
11
12
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
13
            public static void Zero(void* pointer, long capacity)
15
                // A way to prevent wasting resources due to Hyper-Threading.
16
                var threads = Environment.ProcessorCount / 2;
17
                if (threads <= 1)</pre>
                {
19
                    ZeroBlock(pointer, 0, capacity);
20
                }
                else
22
23
                    // Using 2 threads because two-channel memory architecture is the most available
24

    type.
```

```
// CPUs mostly just wait for memory here.
25
26
                    threads = 2;
                    Parallel.ForEach(Partitioner.Create(OL, capacity), new ParallelOptions {
27
                        MaxDegreeOfParallelism = threads }, range => ZeroBlock(pointer, range.Item1,
                        range.Item2));
                }
            }
20
30
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
            private static void ZeroBlock(void* pointer, long from, long to)
32
33
                var offset = (byte*)pointer + from;
                var length = to - from;
35
                var uintMaxValue = uint.MaxValue;
36
                while (length > uintMaxValue)
37
38
                    InitBlock(offset, 0, uintMaxValue);
39
                    length -= uintMaxValue;
40
                    offset += uintMaxValue;
41
42
                InitBlock(offset, 0, unchecked((uint)length));
43
            }
        }
45
46
1.4
     ./Platform.Unsafe/Structure.cs
   using System;
1
   using System.Runtime.CompilerServices;
2
   using System.Runtime.InteropServices;
3
   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
9
        public static class Structure<TStruct>
10
            where TStruct : struct
12
            /// <summary>
13
            /// <para>
14
            /// Returns the size of an unmanaged type in bytes.
            /// This property do this without throwing exceptions for generic types as <see
16
               cref="Marshal.SizeOf{T}()"/> and <see cref="Marshal.SizeOf(Type)"/> do.
            /// </para>
17
            /// <para>
18
            /// Возвращает размер неуправляемого типа в байтах.
19
            /// Этот свойство делает это без выбрасывания исключений для универсальных типов, как
20
               это делают <see cref="Marshal.SizeOf{T}()"/> и <see cref="Marshal.SizeOf(Type)"/>.
            /// </para>
21
            /// </summary>
22
            public static int Size
23
                [MethodImpl(MethodImplOptions.AggressiveInlining)]
25
26
            } = SizeOf<TStruct>();
27
        }
28
29
     ./Platform.Unsafe/StructureExtensions.cs
1.5
   using System.Runtime.CompilerServices;
   using static System.Runtime.CompilerServices.Unsafe;
3
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
5
   namespace Platform.Unsafe
   {
7
        public unsafe static class StructureExtensions
9
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
10
            public static byte[] ToBytes<TStruct>(this ref TStruct obj)
                where TStruct : struct
12
13
                var bytes = new byte[Structure<TStruct>.Size];
                fixed (byte* pointer = bytes)
15
                {
16
                    Copy(pointer, ref obj);
17
                return bytes;
19
            }
```

```
}
22
    ./Platform.Unsafe.Tests/IntPtrExtensionsTests.cs
   using System;
   using System.Runtime.InteropServices; using Xunit;
   using static System.Runtime.CompilerServices.Unsafe;
4
   namespace Platform. Unsafe. Tests
        public unsafe class IntPtrExtensionsTests
8
9
            [Fact]
10
            public void ReadAndWriteOperationsForPointerValuesUnsafeClassMethodsTest()
11
                void* pointer = (void*)Marshal.AllocHGlobal(sizeof(ulong));
                Write(pointer, 42UL);
14
                Assert.Equal(42UL, Read<ulong>(pointer));
15
                Marshal.FreeHGlobal((IntPtr)pointer);
            }
            [Fact]
19
            public void ElementOffsetOperationsForPointerValuesTest()
20
21
                void* pointer = (void*)Marshal.AllocHGlobal(sizeof(ulong) * 10);
                ulong result = (ulong)Add<ulong>(pointer, 5);
23
                Assert.Equal(5UL * 8UL, result - (ulong)pointer);
24
                Marshal.FreeHGlobal((IntPtr)pointer);
            }
26
        }
27
   }
28
    ./Platform.Unsafe.Tests/SizeOfTests.cs
   using System.Runtime.InteropServices;
using Xunit;
   namespace Platform. Unsafe. Tests
4
5
        public static class SizeOfTests
6
            public struct X<T>
9
                public readonly T F1
10
                public readonly T F2;
            }
13
            [Fact]
14
            public static void UnsafeClassSizeOfTest()
15
16
                var size = System.Runtime.CompilerServices.Unsafe.SizeOf<X<int>>();
17
                Assert.Equal(8, size);
19
20
            [Fact]
21
            public static void MarshalSizeOfTest()
22
                var size = Marshal.SizeOf(default(X<int>));
                Assert.Equal(8, size);
25
            }
26
27
            [Fact]
28
            public static void StructurePropertyTest()
                var size = Structure<X<int>>.Size;
31
                Assert.Equal(8, size);
32
            }
33
        }
34
    /Platform Unsafe Tests/StructAndBytesConversionTests cs
   using Xunit;
   namespace Platform. Unsafe. Tests
3
4
        public static class StructAndBytesConversionTests
            [Fact]
            public static void StructToBytesTest()
```

```
ulong source = ulong.MaxValue;
10
                var result = source.ToBytes();
11
                for (int i = 0; i < result.Length; i++)</pre>
                {
13
                     Assert.Equal(byte.MaxValue, result[i]);
14
                }
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 };
22
                ulong result = bytes.ToStructure<ulong>();
                Assert.Equal(ulong.MaxValue, result);
23
            }
24
        }
   }
26
    ./Platform.Unsafe.Tests/ZeroMemoryTests.cs
   using Xunit;
   namespace Platform.Unsafe.Tests
3
4
        public static unsafe class ZeroMemoryTests
6
            [Fact]
            public static void ZeroMemoryTest()
9
                var bytes = new byte[1024];
10
                for (int i = 0; i < bytes.Length; i++)</pre>
11
                    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
                     Assert.Equal(0, bytes[i]);
21
            }
23
        }
24
   }
^{25}
```

Index

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