Особые исключения в .NET

DOTNEXT

Евгений Пешков

e-mail: peshkov@kontur.ru
telegram/twitter/vk: @epeshk

ПЛАН

Поведение исключений в .NET

- AccessViolationException
- ThreadAbortException
- OutOfMemoryException
- StackOverflowException

Обработка исключений в Windows и хаки

- SEH/VEH
- Обработка StackOverflowException

ACCESS VIOLATION

```
try {
   Marshal.WriteByte((IntPtr) 1000, 42);
}
catch (AccessViolationException) {
   ...
}
```

ACCESS VIOLATION

```
try {
   var bytes = new byte[] {42};
   Marshal.Copy(bytes, 0, (IntPtr) 1000, bytes.Length);
}
catch (AccessViolationException) {
   ...
}
```

ACCESS VIOLATION

```
Marshal.Copy(bytes, 0, (IntPtr) 1000, bytes.Length);

Marshal.WriteByte((IntPtr) 1000, 42);
```

ACCESS VIOLATION

```
static void Copy(...) {
  Marshal.CopyToNative(
    (object) source, startIndex, destination, length);
}

[MethodImpl(MethodImplOptions.InternalCall)]
static extern void CopyToNative(
    object source,
    int startIndex,
    IntPtr destination,
    int length);
```

ACCESS VIOLATION

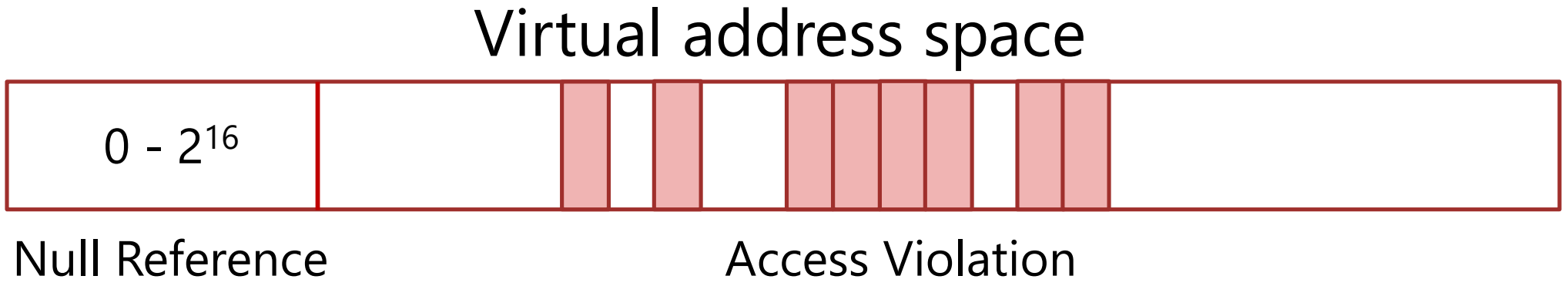
```
unsafe static void WriteByte(IntPtr ptr, byte val) {
  try {
    *(byte*) ptr = val;
  }
  catch (NullReferenceException) {
    throw new AccessViolationException();
  }
}
```

Все исключения, выброшенные `throw` - обрабатываемые

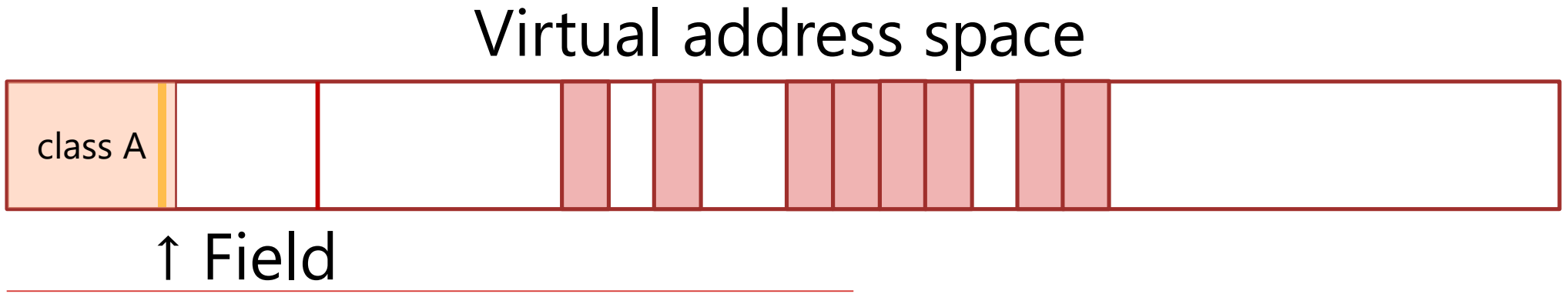
ACCESS VIOLATION

```
unsafe static void WriteByte(IntPtr ptr, byte val) {
  try {
    *(byte*) ptr = val;
  }
  catch (NullReferenceException) {
    throw new AccessViolationException();
  }
}
*(byte*) 1000 = 42;
```

NULL REFERENCE



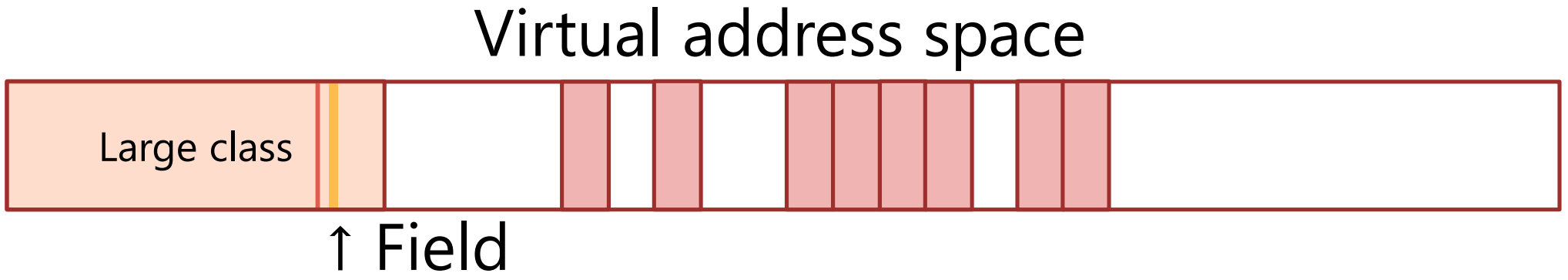
NULL REFERENCE




```
A a = null;
Console.WriteLine(a.Field);
```

```
movzx   eax,byte ptr [rdx+48h]
```

NULL REFERENCE



ACCESS VIOLATION

```
[StructLayout(LayoutKind.Explicit)]
public class LargeClass {
  public byte A;
  [FieldOffset(64 * 1024)] public byte B;
}
```

```
void GetA(LargeClass c) {
  return c.A;
}
```

```
movzx   eax,byte ptr [rdx+8h]
```

```
void GetB(LargeClass c) {
  return c.B;
}
```

```
cmp     dword ptr [rdx],edx
movzx   eax,byte ptr [rdx+10008h]
```

ACCESS VIOLATION

Code/Address	< 64 KB	>= 64 KB
Managed	Null Reference	Access Violation
Native	Access Violation	Access Violation

ACCESS VIOLATION

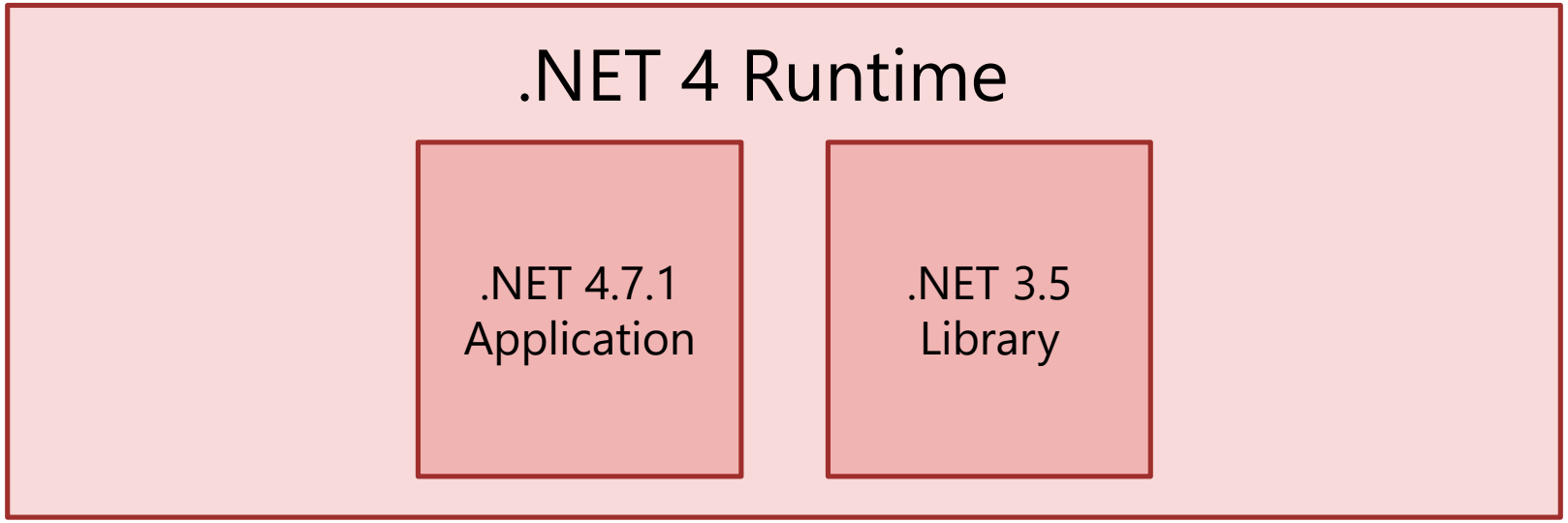
Версия .NET	Поведение Native Access Violation
1.0	NullReferenceException
2.0, 3.5	Обрабатываемый AV
4.0 и выше	Обрабатываемый AV*
.NET Core	Необрабатываемый AV

- `[HandleProcessCorruptedStateExceptions]`

`App.config`

- `legacyNullReferenceExceptionPolicy`
- `legacyCorruptedStateExceptionsPolicy`

ACCESS VIOLATION



ACCESS VIOLATION

```
while (isRunning) {
   try {
      action();
   }
   catch (Exception e) {
      log.Error(e);
   }
   WaitForNextExecution(...);
}
```

ACCESS VIOLATION

- Native/managed
- Обработка зависит от Target Framework и настроек рантайма

THREAD ABORT

```
var thread = new Thread(() => {
   try {
      ...
   }
   catch (ThreadAbortException e) {
      ...
      throw;
   }
});
...
thread.Abort();
```

THREAD ABORT

```
var thread = new Thread(() => {
   try {
      ...
   }
   catch (ThreadAbortException e) {
      ...
      Thread.ResetAbort();
   }
});
...
thread.Abort();
```

FINALLY

```
var thread = new Thread(() =>
{
  try { }
  catch { } // <-- No ThreadAbortException in catch
  finally { // <-- No ThreadAbortException in finally
    Thread.Sleep(-1);
  }
});
thread.Start();
...
thread.Abort(); // ← Never returns
```

THREAD ABORT – NET CORE

```
thread.Abort()
```

- throws PlatformNotSupportedException

OUT OF MEMORY

```
var arr4gb = new int[int.MaxValue/2];

App.config: gcAllowVeryLargeObjects

var largeArr = new int[int.MaxValue];
```

Max array index:

- byte arrays – 0x7FFFFFC7
- other arrays– 0X7FEFFFFF

OUT OF MEMORY

```
static string Concat(params string[] values)
{
  int totalLength = 0;
  string[] sArgs = new string[values.Length];
  for (int index = 0; index < values.Length; ++index)
  {
    string str = values[index];
    sArgs[index] = str == null ? string.Empty : str;
    totalLength += sArgs[index].Length;
    if (totalLength < 0)
      throw new OutOfMemoryException();
  }
  return string.ConcatArray(sArgs, totalLength);
}
```

OUT OF MEMORY

```
LimitMemory(64.Mb());

try {
  while (true)
    list.Add(new byte[size]);
}
catch (OutOfMemoryException e) {
  Console.WriteLine(e);
}
```

OUT OF MEMORY

- Исключение обработается
- Процесс упадёт
- Зайдём в catch, но исключение вылетит снова
- Зайдём в catch, но вылетит StackOverflow

OUT OF MEMORY

Ответ...

OUT OF MEMORY

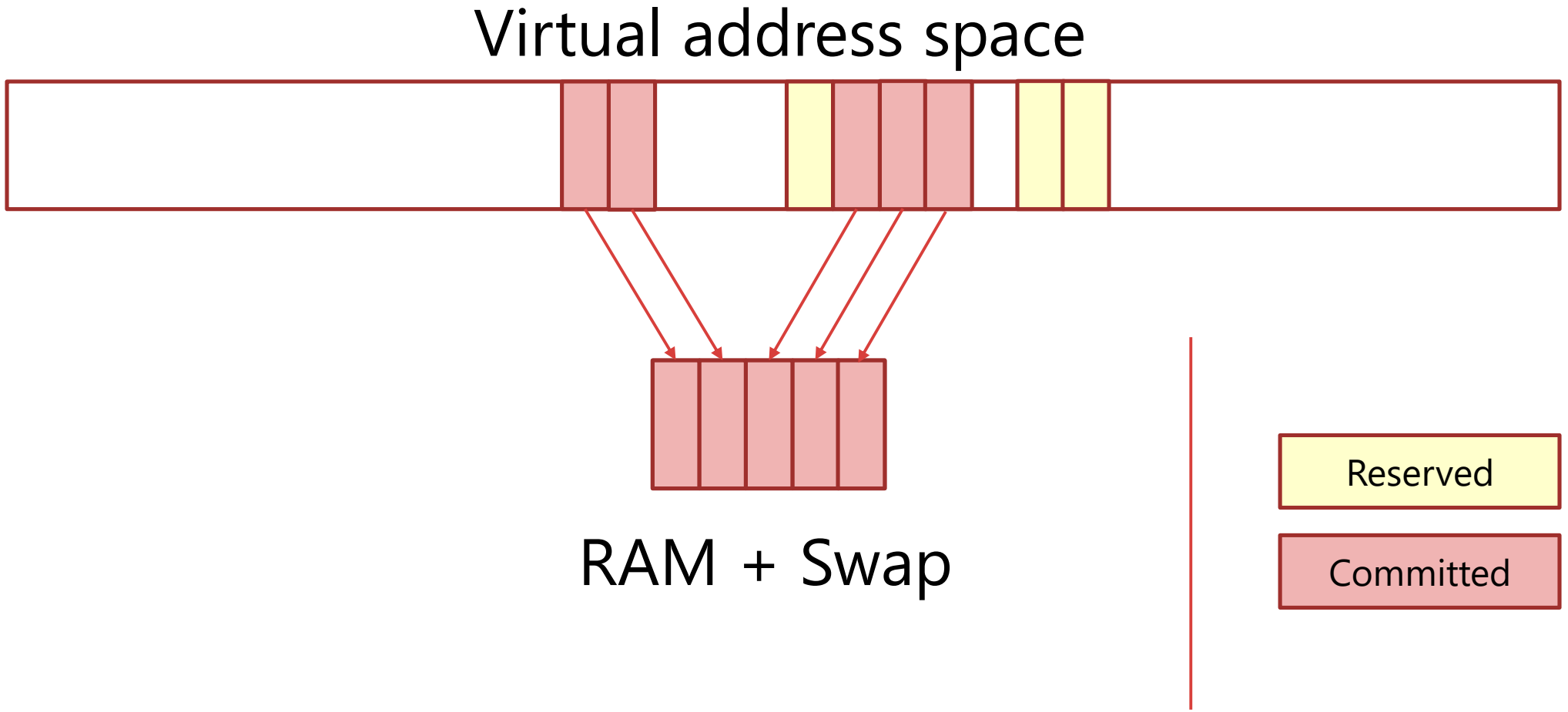
- ✓ Исключение обработается
- ✓ Процесс упадёт
- ✓ Зайдём в catch, но исключение вылетит снова
- ✓ Зайдём в catch, но вылетит StackOverflow

OUT OF MEMORY

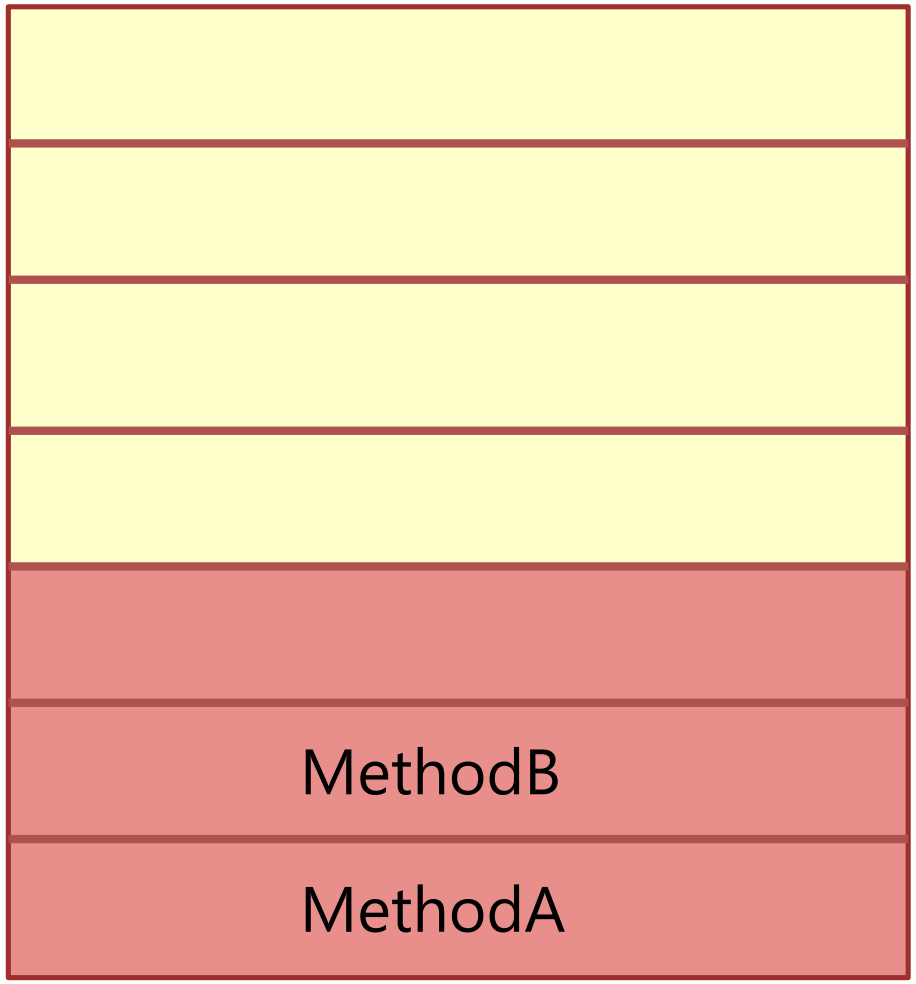
```
LimitMemory(64.Mb());

try {
  while (true)
    list.Add(new byte[size]);
}
catch (OutOfMemoryException e) {
  Console.WriteLine(e);
}
```

ОТКУДА STACKOVERFLOW



ОТКУДА STACKOVERFLOW



Reserved pages

Committed pages

ОТКУДА STACKOVERFLOW

```
new Thread(() => F(), 4*1024*1024)
   .Start();
```

ОТКУДА STACKOVERFLOW

<disableCommitThreadStack> Element

03/30/2017 • 2 minutes to read • Contributors

Specifies whether the full thread stack is committed when a thread is started.

```
<disableCommitThreadStack enabled="false"/>
```

«The default behavior of the common language runtime is to commit the full thread stack when a thread is started» - MSDN

https://docs.microsoft.com/en-us/dotnet/framework/configure-apps/file-schema/runtime/disablecommitthreadstack-element

STACK OVERFLOW

```
try {
  InfiniteRecursion();
}
catch (Exception) {
  ...
}
```

```
try {
  throw new StackOverflowException();
}
catch (Exception) {
  ...
}
```

STACK OVERFLOW

«You **cannot catch** stack overflow exceptions, because the exception-handling code may require the stack. Instead, when a stack overflow occurs in a normal application, the Common Language Runtime (CLR) terminates the process.» - MSDN

https://msdn.microsoft.com/en-us/library/w6sxk224.aspx

КАК ИЗБЕЖАТЬ STACK OVERFLOW

- RuntimeHelpers.EnsureSufficientExecutionStack();
  - «Ensures that the remaining stack space is large enough to execute the average .NET Framework function.» - MSDN
  - InsufficientExecutionStackException
  - 512 KB – x86, AnyCPU, 2 MB – x64 (half of stack size)
  - 64/128 KB - .NET Core
  - Check only stack address space
- RuntimeHelpers.TryEnsureSufficientExecutionStack();

https://docs.microsoft.com/en-us/dotnet/api/system.runtime.compilerservices.runtimehelpers.ensuresufficientexecutionstack
https://github.com/dotnet/coreclr/issues/9167

КАК ИЗБЕЖАТЬ STACK OVERFLOW

```
Span<byte> span;
if (CanAllocateOnStack(size))
  span = stackalloc byte[size];
else
  span = new byte[size];
```

КАК ИЗБЕЖАТЬ STACK OVERFLOW

- `RuntimeHelpers.TryForSufficientStack(size);`
- `trystackalloc`

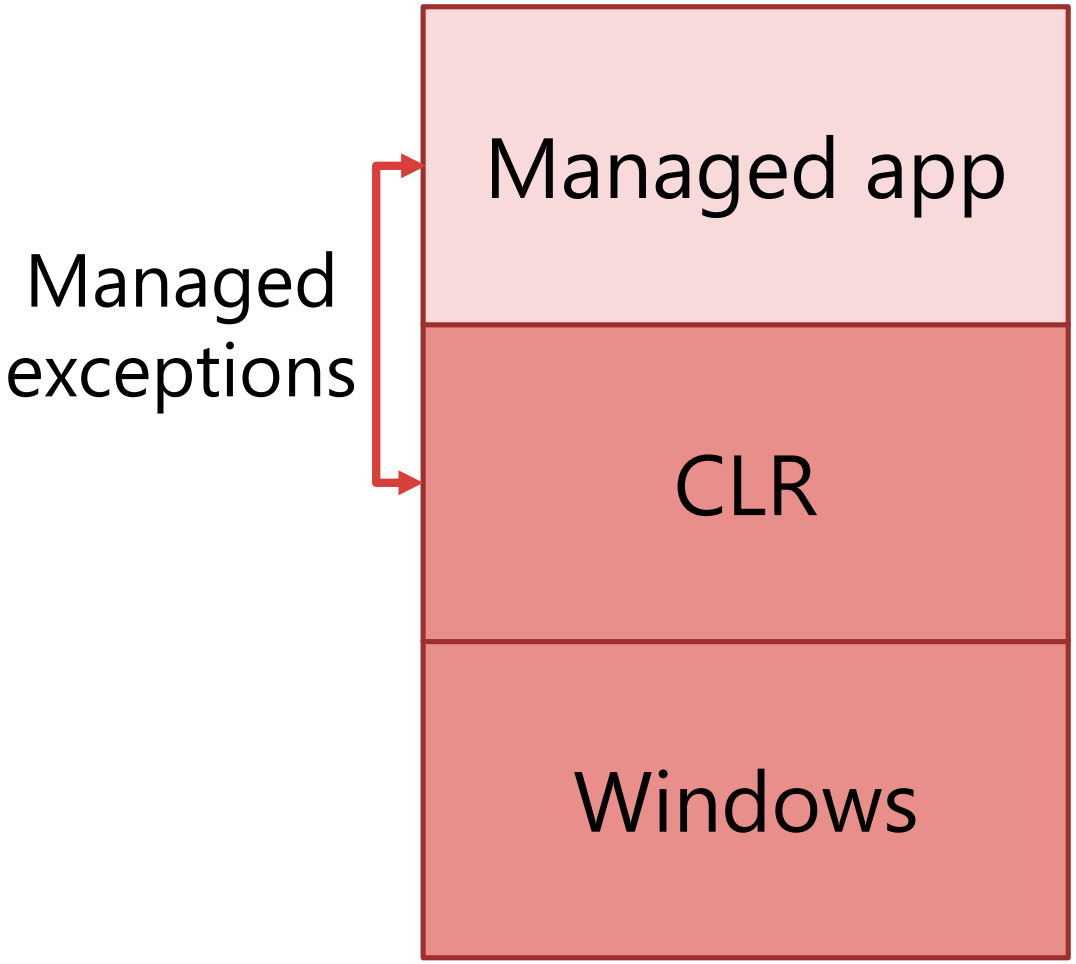
https://github.com/dotnet/corefx/issues/26954

STACK OVERFLOW

«You **cannot catch** stack overflow exceptions, because the exception-handling code may require the stack. Instead, when a stack overflow occurs in **a normal application**, the Common Language Runtime (CLR) terminates the process.» - MSDN

https://msdn.microsoft.com/en-us/library/w6sxk224.aspx

НА УРОВЕНЬ НИЖЕ



CLR HOSTING

«An application that **hosts the CLR** can change the default behavior and specify that the CLR **unload the application domain** where the exception occurs, but lets the process continue.» - MSDN

`StackOverflowException` → `AppDomainUnloadedException`

https://msdn.microsoft.com/en-us/library/w6sxk224.aspx

CLR HOSTING

```
ICLRPolicyManager *policyMgr;
pCLRControl->GetCLRManager(
    IID_ICLRPolicyManager, (void**)(&policyMgr));
policyMgr->SetActionOnFailure(
    FAIL_StackOverflow, eRudeUnloadAppDomain);
```

- eRudeExitProcess
- eRudeUnloadAppDomain

CLR HOSTING

```
try {
  var appd = AppDomain.CreateDomain("...");
  appd.DoCallBack(() =>
  {
    var thread = new Thread(() => InfiniteRecursion());
    thread.Start();
    thread.Join();
  });
  AppDomain.Unload(appd);
}
catch (AppDomainUnloadedException) { }
```

APP DOMAINS

```
public class CustomException : Exception {}

var appd = AppDomain.CreateDomain("...");
appd.DoCallBack(() => throw new CustomException());
```

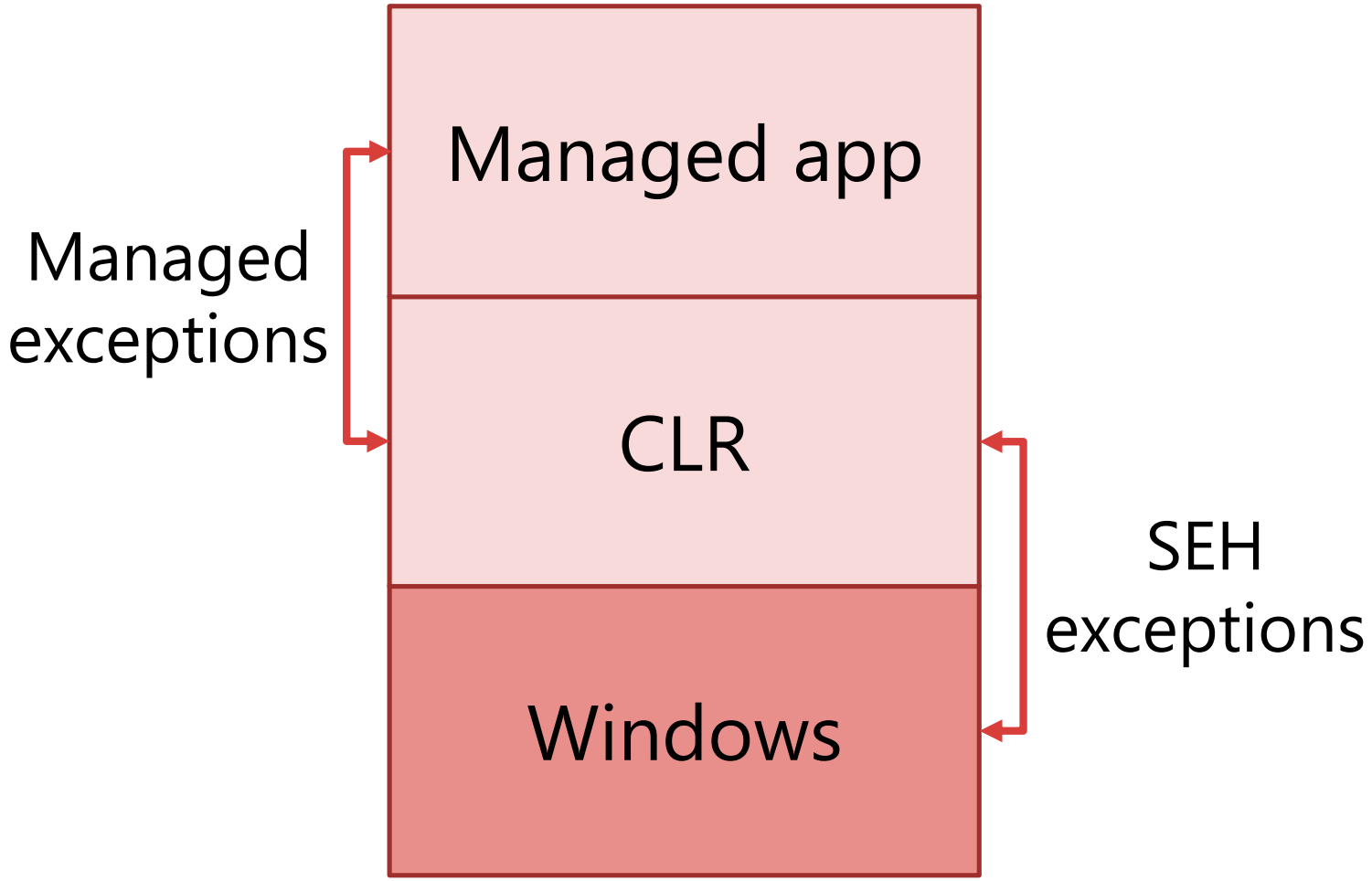
System.Runtime.Serialization.SerializationException:
Type 'CustomException' is not marked as serializable.
at System.AppDomain.DoCallBack(CrossAppDomainDelegate callBackDelegate)

APP DOMAINS (SERIALIZATION EXCEPTION)

```
[Serializable]
public class CustomException : Exception
{
    public CustomException(){}
    public CustomException(
       SerializationInfo info,
       StreamingContext ctx) : base(info, context){}
}

var appd = AppDomain.CreateDomain("...");
appd.DoCallBack(() => throw new CustomException());
```

НА УРОВЕНЬ НИЖЕ



SEH

SEH – Structured Exception Handling

- Механизм обработки исключений в Windows
- Единообразная обработка software и hardware исключений
- C# исключения реализованы поверх SEH

SEH ↔ MANAGED

- EXCEPTION_STACK_OVERFLOW → Crash
- EXCEPTION_ACCESS_VIOLATION → AccessViolationException
- EXCEPTION_ACCESS_VIOLATION → NullReferenceException
- EXCEPTION_INT_DIVIDE_BY_ZERO → DivideByZeroException
- Unknown SEH exceptions → SEHException

RAISE SEH

```
[DllImport("kernel32.dll")]
static extern void RaiseException(
  uint dwExceptionCode,
  uint dwExceptionFlags,
  uint nNumberOfArguments,
  IntPtr lpArguments);
```

RAISE SEH

```
// DivideByZeroException
RaiseException(0xc0000094, 0, 0, IntPtr.Zero);

// Stack overflow
RaiseException(0xc00000fd, 0, 0, IntPtr.Zero);
```

THROW & SEH

```
throw → RaiseException(0xe0434f4d, ...)
```

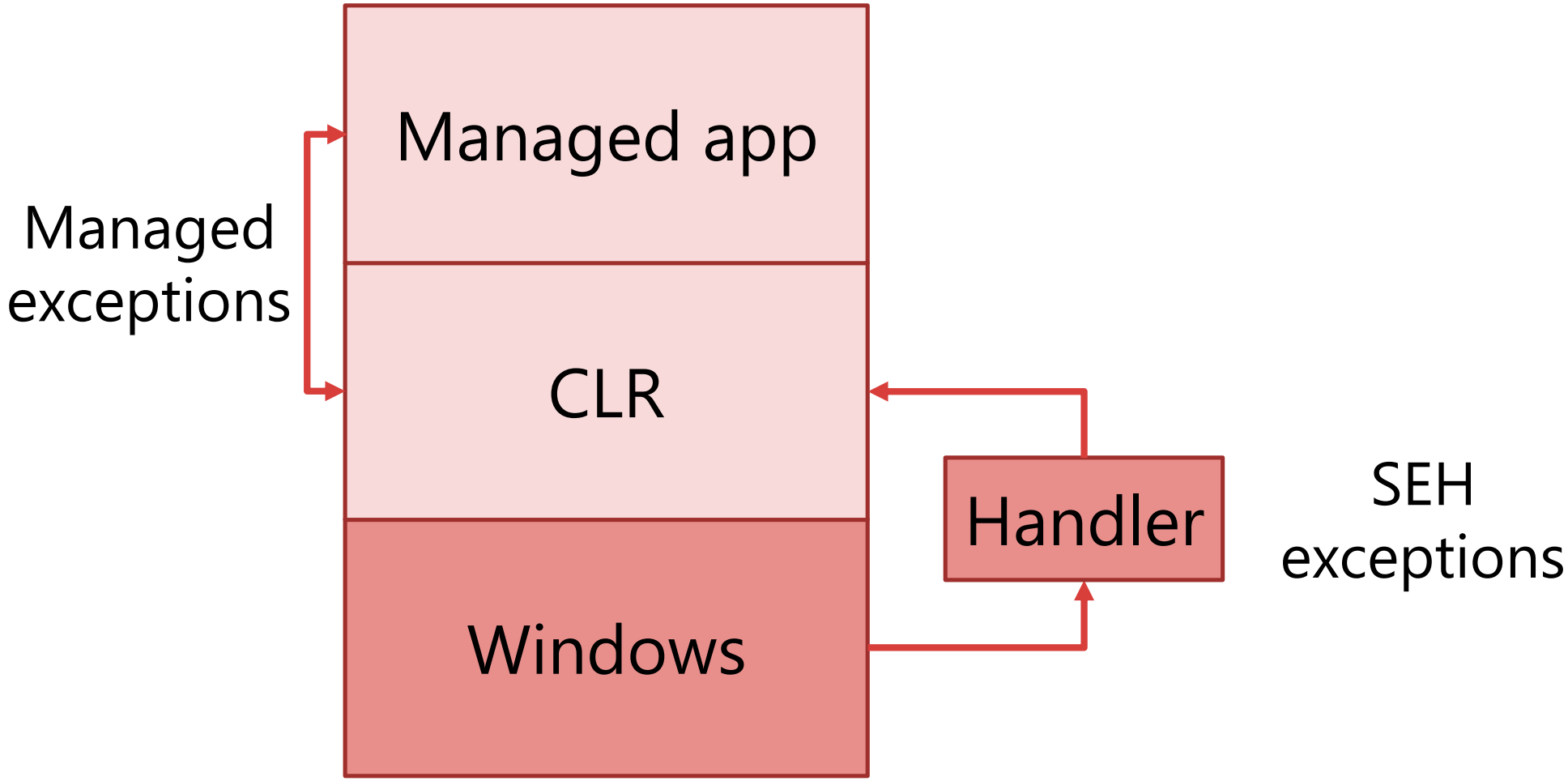
- SEH CLR Exception ← Managed exception
- SEH CLR Exception → Managed exception

VEH

VEH – Vectored Exception Handling

- Расширение SEH
- Работает на уровне процесса
- Позволяет изменить SEH исключение до обработки

ПЕРЕХВАТ STACK OVERFLOW



VEH

```
[DllImport("kernel32.dll", SetLastError = true)]
static extern IntPtr AddVectoredExceptionHandler(
  IntPtr FirstHandler,
  PVECTORED_EXCEPTION_HANDLER VectoredHandler);
```

VEH

```
delegate VEH PVECTORED_EXCEPTION_HANDLER(
   ref EXCEPTION_POINTERS exceptionPointers);

public enum VEH : long
{
   EXCEPTION_CONTINUE_SEARCH = 0,
   EXCEPTION_EXECUTE_HANDLER = 1,
   EXCEPTION_CONTINUE_EXECUTION = -1
}
```

VEH

```
delegate VEH PVECTORED_EXCEPTION_HANDLER(
  ref EXCEPTION_POINTERS exceptionPointers);

[StructLayout(LayoutKind.Sequential)]
unsafe struct EXCEPTION_POINTERS {
  public EXCEPTION_RECORD* ExceptionRecord;
  public IntPtr Context;
}
```

VEH

```
delegate VEH PVECTORED_EXCEPTION_HANDLER(
  ref EXCEPTION_POINTERS exceptionPointers);

[StructLayout(LayoutKind.Sequential)]
unsafe struct EXCEPTION_RECORD {
  public uint ExceptionCode;
  ...
}
```

ПЕРЕХВАТ STACK OVERFLOW

```
static unsafe VEH Handler(ref EXCEPTION_POINTERS e) {
  if (e.ExceptionRecord == null)
    return VEH.EXCEPTION_CONTINUE_SEARCH;

  var record = e.ExceptionRecord;
  if (record->ExceptionCode != ExceptionStackOverflow)
    return VEH.EXCEPTION_CONTINUE_SEARCH;

  record->ExceptionCode = 0x01234567;
  return VEH.EXCEPTION_EXECUTE_HANDLER;
}
```

ПЕРЕХВАТ STACK OVERFLOW

```
HandleSO(() => InfiniteRecursion());
```

ПЕРЕХВАТ STACK OVERFLOW

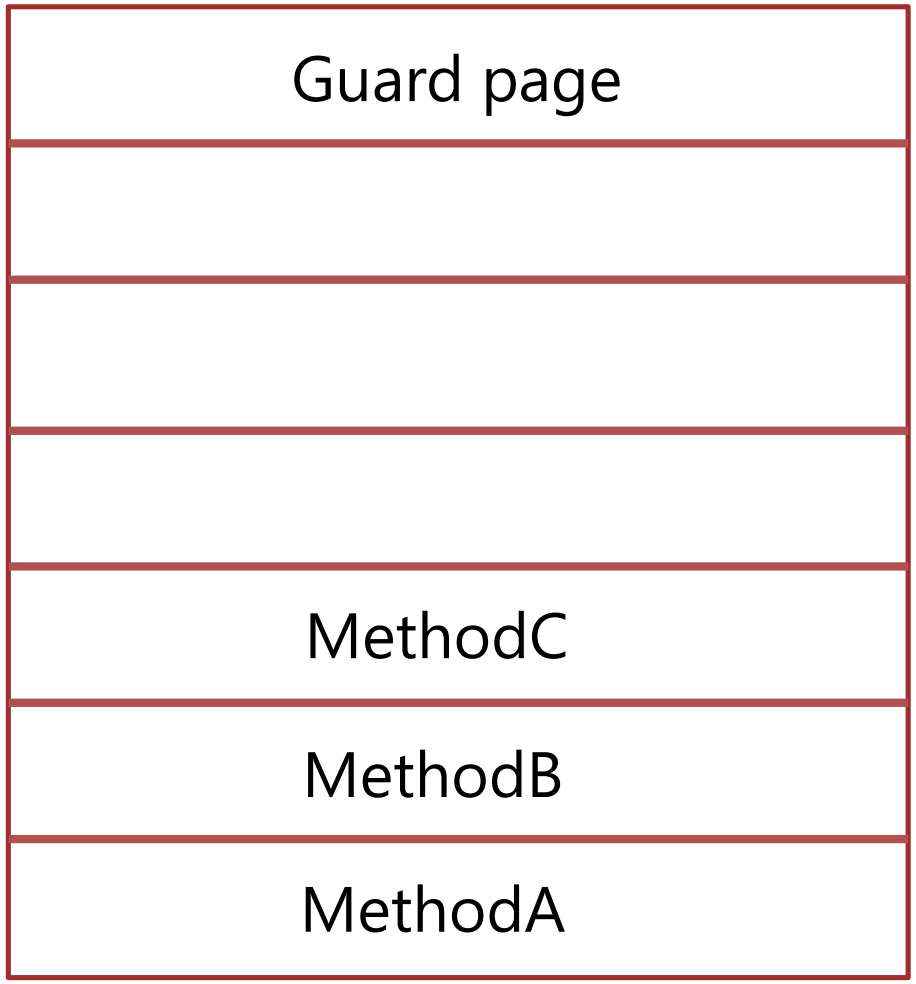
```
static T HandleSO<T>(Func<T> action) {
  Kernel32.AddVectoredExceptionHandler(
    IntPtr.Zero,
    new PVECTORED_EXCEPTION_HANDLER(Handler));
  Kernel32.SetThreadStackGuarantee(ref size);
  try {
    return action();
  }
  catch (Exception e)
    when ((uint) Marshal.GetExceptionCode() == 0x01234567) {}
  return default(T);
}
```

ПЕРЕХВАТ STACK OVERFLOW

```
HandleSO(() => InfiniteRecursion());
HandleSO(() => InfiniteRecursion());
```

AccessViolationException

ПЕРЕХВАТ STACK OVERFLOW



STATUS_GUARD_PAGE_VIOLATION

ВОССТАНОВЛЕНИЕ GUARD PAGE

```
[DllImport("msvcrt.dll")]
static extern int _resetstkoflw();
```

ПЕРЕХВАТ AV В NET CORE

```
Kernel32.AddVectoredExceptionHandler(
  IntPtr.Zero, handler);

Kernel32.AddVectoredExceptionHandler(
  (IntPtr) 1,  handler);
```

ВЫВОДЫ

- Исключения не так просты, как кажутся
- Не все исключения обрабатываются одинаково
- Обработка исключений происходит на разных уровнях абстракции
- Можно вмешаться в процесс обработки исключений

LINKS

- Samples
  https://github.com/epeshk/dotnext-2018-exceptions
- Exceptional Exceptions in .NET
  https://www.youtube.com/watch?v=U92Ts53win4

ВОПРОСЫ

DOTNEXT

Евгений Пешков

e-mail: peshkov@kontur.ru
telegram/twitter/vk: @epeshk