Public Preview Version

.NET Memory Dump Analysis Accelerated

Version 3.0

Dmitry Vostokov
Software Diagnostics Services

Prerequisites

WinDbg Commands

We use these boxes to introduce some WinDbg commands used in practice exercises

Basic .NET programming and debugging

Training Goals

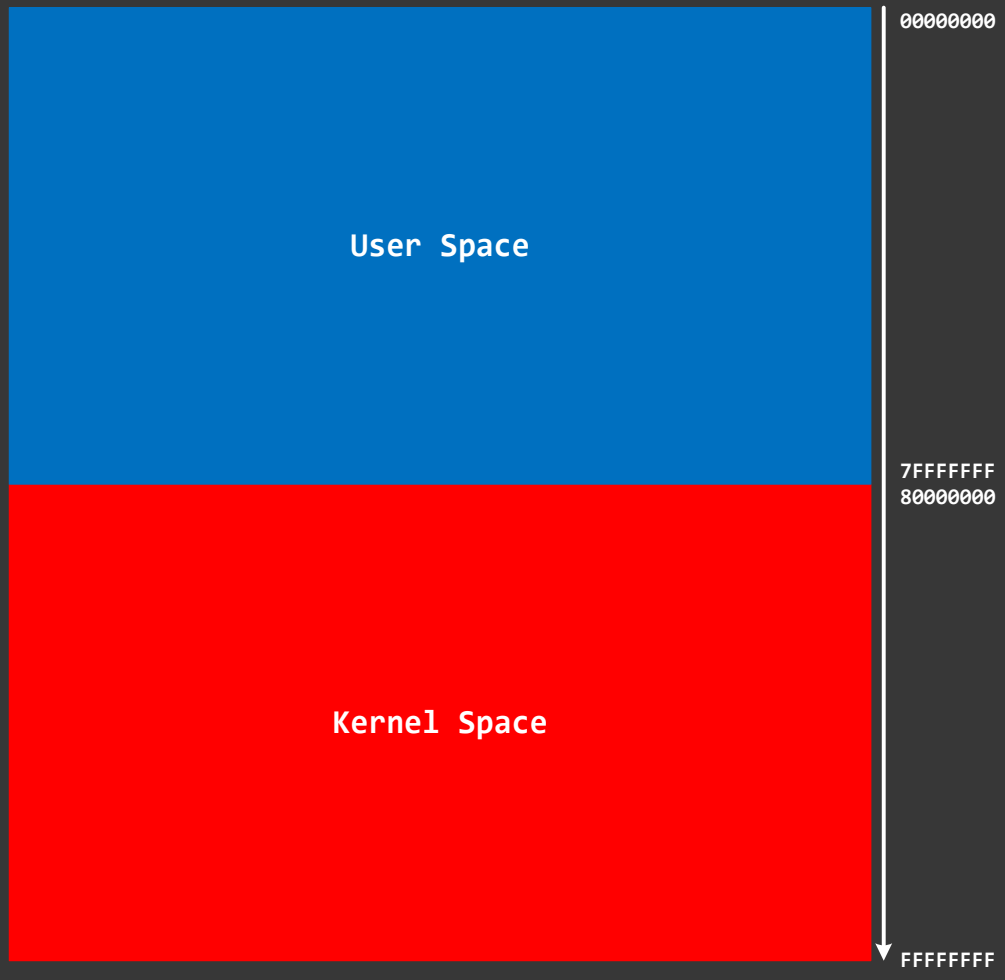
- Review fundamentals
- Learn how to analyze process dumps
- Learn necessary commands in context
- Cover CLR 4 (x86 and x64)

Training Principles

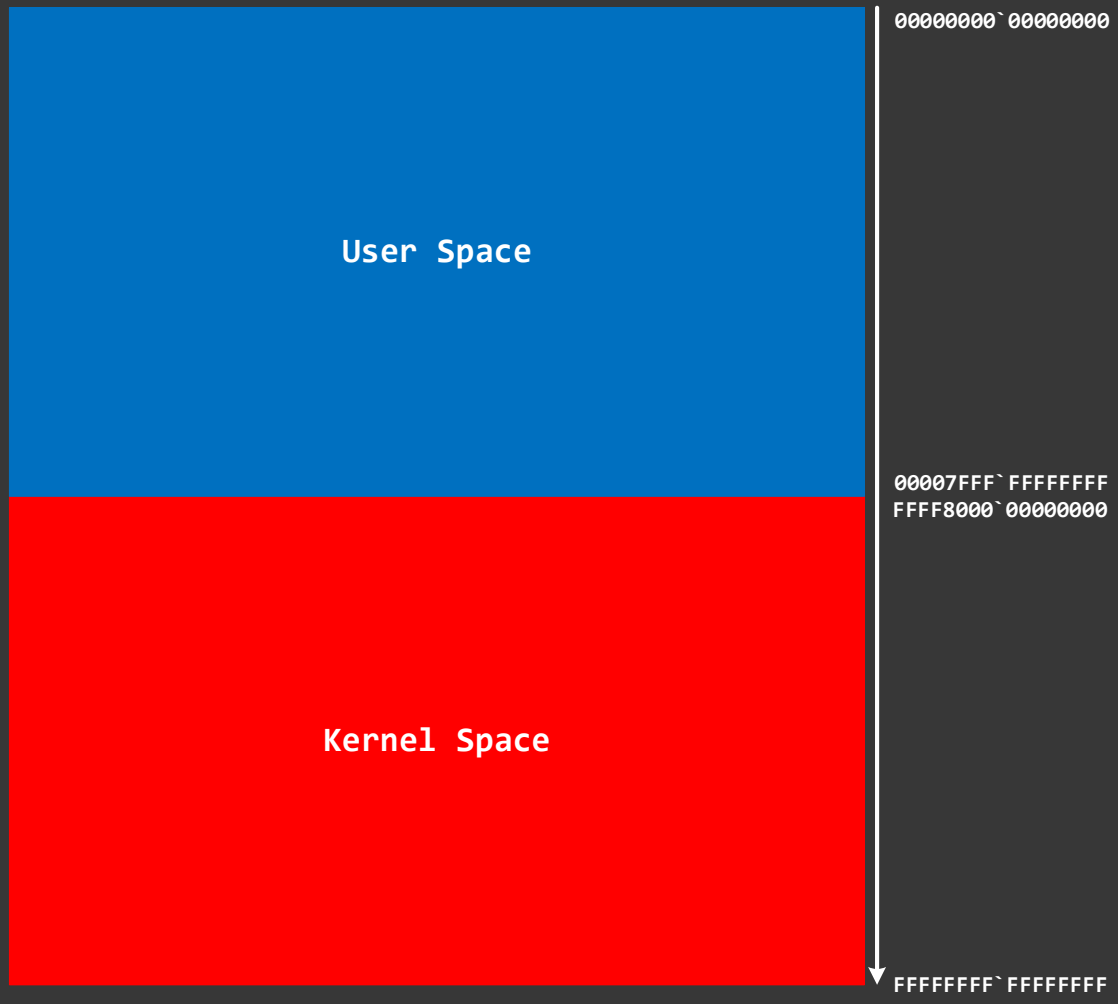
- Talk only about what I can show
- Lots of pictures
- Lots of examples
- Original content and examples

Part 1: Fundamentals

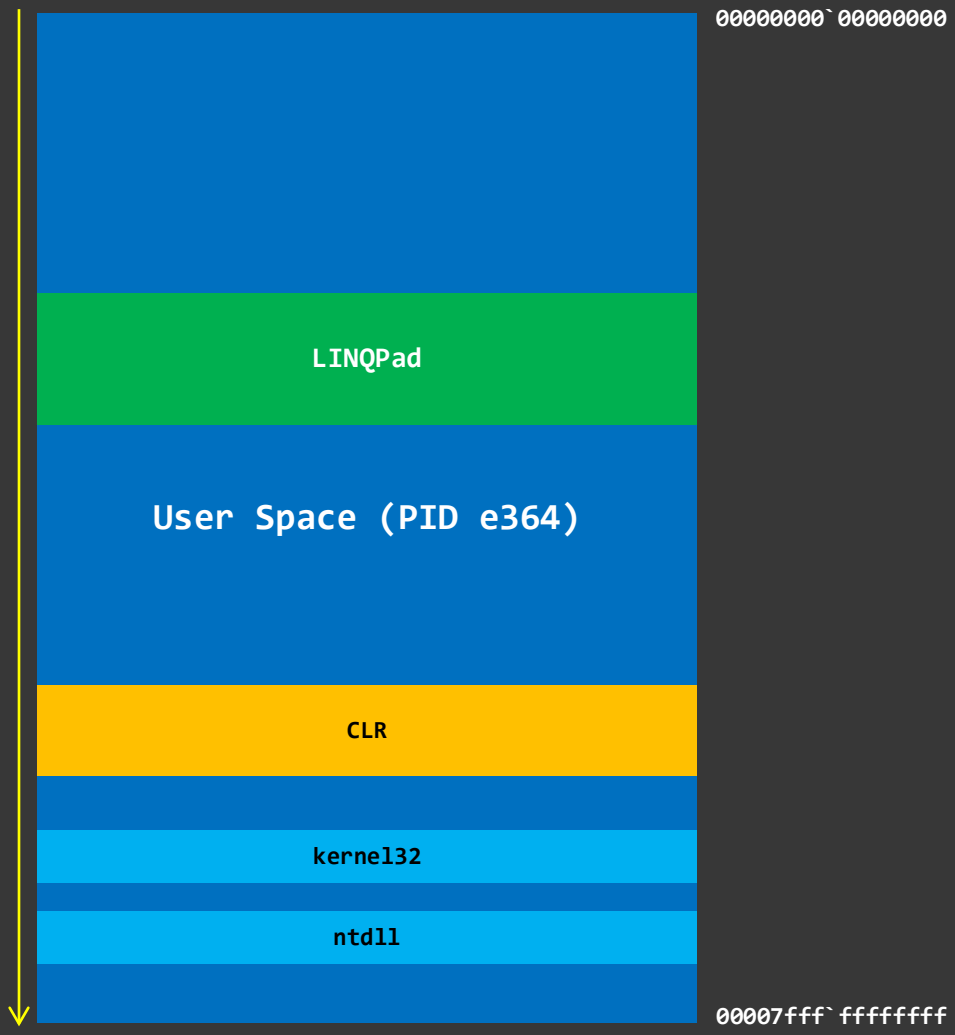
Memory Space (x86)



Memory Space (x64)



User / Managed Space



WinDbg Commands

lmv command lists all loaded modules (EXE and DLLs)

Types/Assemblies/Modules

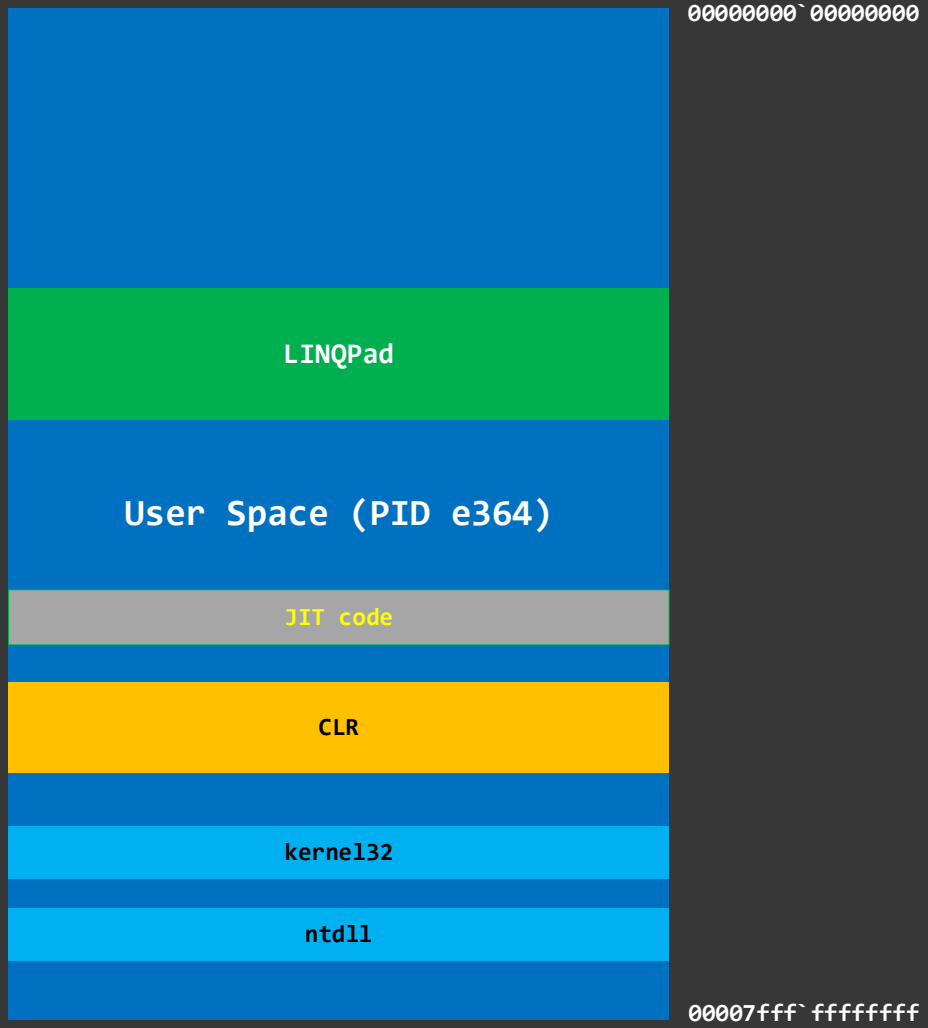
WinDbg Commands

lmv command lists all loaded modules (EXE and DLLs)

!IP2MD command shows type method and module address

!DumpModule command shows module name

Process Threads



WinDbg Commands

.load *<a path to SOS>*
Loads SOS WinDbg extension

~<n>s command switches between threads

k command shows unmanaged stack trace

!Threads command shows managed threads

!CLRStack command shows managed stack trace

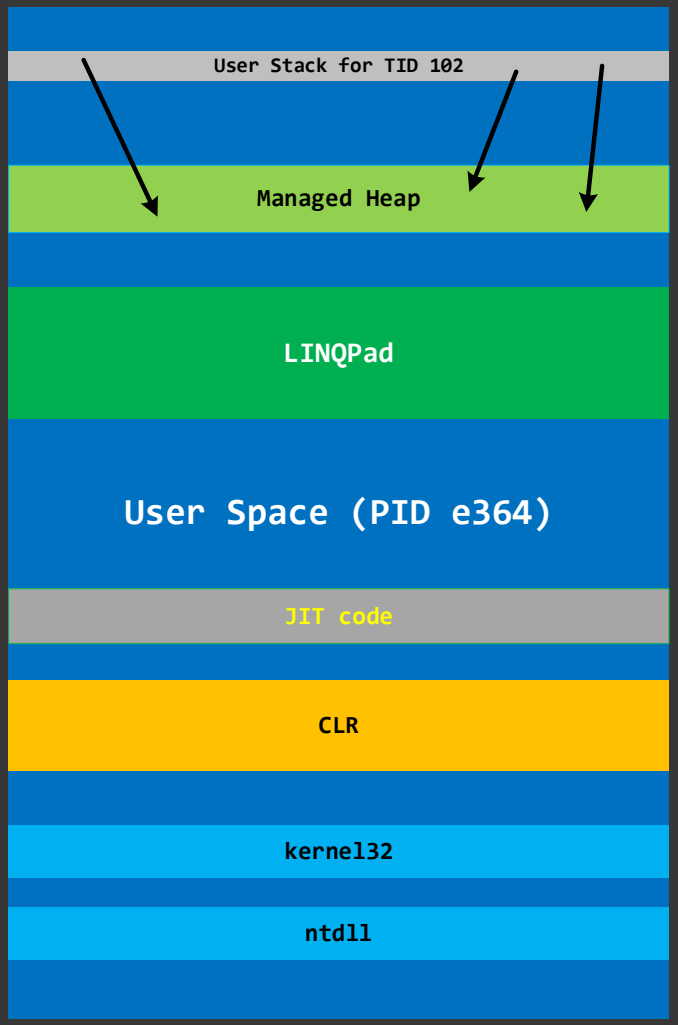
Example

```
0:000> k
 # Child-SP          RetAddr           Call Site
00 000000dc`219be178 00007ffb`41ddddb8 win32u!NtUserWaitMessage+0x14
01 000000dc`219be180 00007ffb`41d71535 System_Windows_Forms_ni+0x2cddb8
02 000000dc`219be230 00007ffb`41d70c88 System_Windows_Forms_ni+0x261535
03 000000dc`219be330 00007ffb`41d70a8f System_Windows_Forms_ni+0x260c88
04 000000dc`219be3d0 00007ffb`12f4dbc0 System_Windows_Forms_ni+0x260a8f
05 000000dc`219be430 00007ffb`12f45a94 0x00007ffb`12f4dbc0
06 000000dc`219be5e0 00007ffb`12f41c9d 0x00007ffb`12f45a94
07 000000dc`219bed20 00007ffb`12f420b6 0x00007ffb`12f41c9d
08 000000dc`219bee00 00007ffb`12f406e4 0x00007ffb`12f420b6
09 000000dc`219bee30 00007ffb`724d5863 0x00007ffb`12f406e4
0a 000000dc`219bef20 00007ffb`724d5702 clr!CallDescrWorkerInternal+0x83
0b 000000dc`219bef60 00007ffb`724d5e05 clr!CallDescrWorkerWithHandler+0x4e
0c 000000dc`219befa0 00007ffb`72633598 clr!MethodDescCallSite::CallTargetWorker+0xf8
0d 000000dc`219bf0a0 00007ffb`72633ede clr!RunMain+0x1e7
0e 000000dc`219bf280 00007ffb`72633d9d clr!Assembly::ExecuteMainMethod+0xb6
0f 000000dc`219bf570 00007ffb`72633750 clr!SystemDomain::ExecuteMainMethod+0x639
10 000000dc`219bfb90 00007ffb`726336ce clr!ExecuteEXE+0x3f
11 000000dc`219bfc00 00007ffb`72634234 clr!_CorExeMainInternal+0xb2
12 000000dc`219bfc90 00007ffb`72ec7a6d clr!CorExeMain+0x14
13 000000dc`219bfcd0 00007ffb`7303a44c mscoreei!CorExeMain+0x112
14 000000dc`219bfd30 00007ffb`85591fe4 MSCOREE!CorExeMain_Exported+0x6c
15 000000dc`219bfd60 00007ffb`8797f061 KERNEL32!BaseThreadInitThunk+0x14
16 000000dc`219bfd90 00000000`00000000 ntdll!RtlUserThreadStart+0x21
```

```
0:000> !IP2MD 0x00007ffb`12f45a94
MethodDesc:   00007ffb12f81c40
Method Name:  LINQPad.Program.Go(System.String[])
Class:        00007ffb12f322e0
MethodTable:  00007ffb12f81e10
mdToken:      0000000006000579
Module:       00007ffb12dc4110
IsJitted:     yes
CodeAddr:     00007ffb12f436f0
Transparency: Critical

0:000> !DumpModule 00007ffb12dc4110
Name:       C:\Program Files\LINQPad5\LINQPad.exe
Attributes: PEFile
Assembly:   000001f80c3cf620
LoaderHeap:                   0000000000000000
TypeDefToMethodTableMap: 00007ffb12ed0020
TypeRefToMethodTableMap: 00007ffb12ed3a88
MethodDefToDescMap:      00007ffb12ed7000
FieldDefToDescMap:       00007ffb12eefcc8
MemberRefToDescMap:      0000000000000000
FileReferencesMap:       00007ffb12f036c8
AssemblyReferencesMap:   00007ffb12f036d0
MetaData start address:  000001f808cffb04 (1498300
bytes)
```

Thread Stack Raw Data



WinDbg Commands

Get stack range:
!teb

Dump raw data:
dc / **dps** / **dpp** / **dpa** / **dpu**

Dump managed references:
!DumpStackObjects

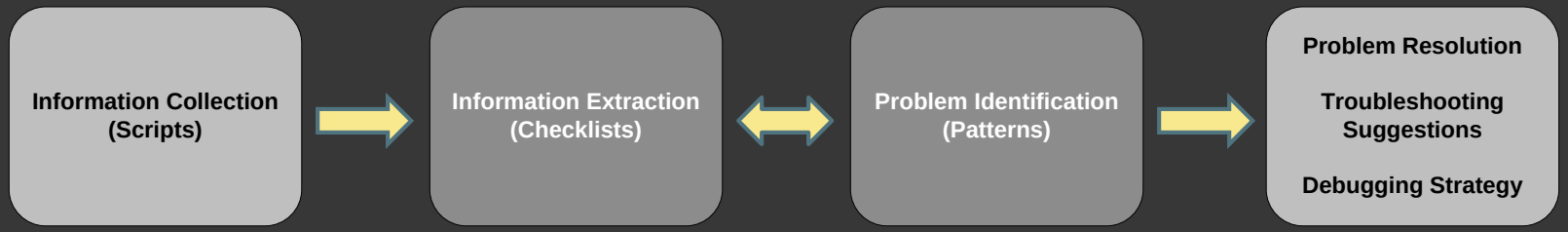
Pattern-Oriented Diagnostic Analysis

Diagnostic Pattern: a common recurrent identifiable problem together with a set of recommendations and possible solutions to apply in a specific context.

Diagnostic Problem: a set of indicators (symptoms, signs) describing a problem.

Diagnostic Analysis Pattern: a common recurrent analysis technique and method of diagnostic pattern identification in a specific context.

Diagnostics Pattern Language: common names of diagnostic and diagnostic analysis patterns. The same language for any operating system: Windows, Mac OS X, Linux, ...



Checklist: http://www.dumpanalysis.org/windows-memory-analysis-checklist

Part 2: Practice Exercises

Links

- Memory Dumps:

Not available in preview version

- Exercise Transcripts:

Not available in preview version

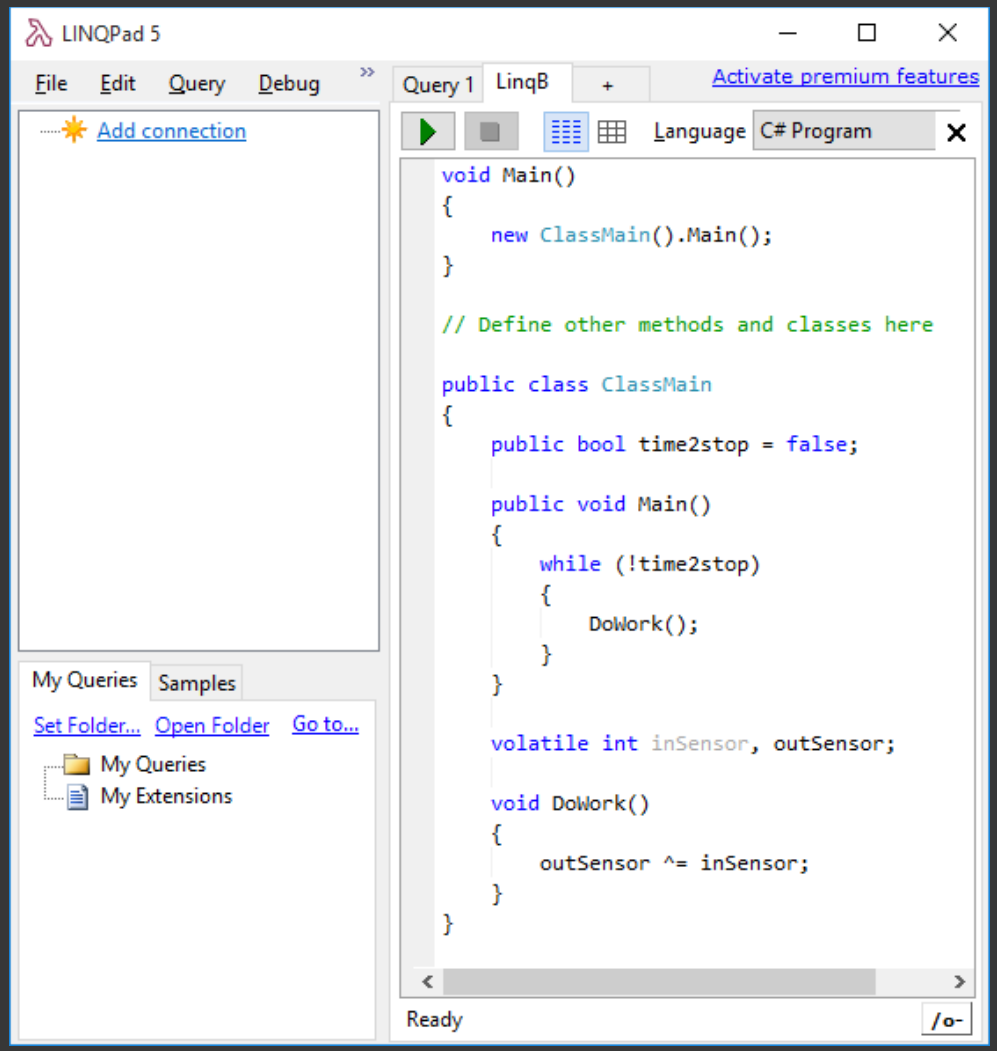
Exercise 0

- **Goal:** Install Debugging Tools for Windows and learn how to set up symbols correctly
- **Patterns:** Incorrect Stack Trace
- **Commands:** .symfix, .reload, k
- \ANETMDA-Dumps\Exercise-0-Download-Setup-WinDbg.pdf

Process Memory Dumps

Exercises PN1 – PN12

Modeling with LINQPad



http://www.linqpad.net/

Exercise PN1

- **Goal:** Learn how to load the correct .NET SOS WinDbg extension and analyze managed space
- **Patterns:** Stack Trace Collection; CLR Thread; Version-Specific Extension; Software Exception; Exception Stack Trace; Managed Code Exception; Managed Stack Trace
- **Commands:** .logopen, .symfix, .reload, ~*k, .load, !pe, ~*e, lmv, .chain, .unload, !analyze -v, !CLRStack, .logclose
- \ANETMDA-Dumps\Exercise-PN1-Analysis-process-dump-ApplicationA.pdf

Exercise PN2

- **Goal:** Compare 64-bit process memory dump from exercise PN1 with 32-bit process memory dump
- **Patterns:** Platform-Specific Debugger
- \ANETMDA-Dumps\Exercise-PN2-Analysis-process-dump-ApplicationA-32.pdf

Exercise PN3

- **Goal:** Learn how to find problem assemblies, modules, classes and methods, disassemble code, analyze CPU spikes
- **Patterns:** Active Thread; Technology-Specific Subtrace; JIT Code; Spiking Thread; Annotated Disassembly
- **Commands:** !analyze -v -hang, !IP2MD, !runaway, .prompt_allow, ~<>s, ~<>k, !U, !DumpMD, !DumpClass, !DumpMT, !DumpModule, !DumpAssembly, !DumpDomain
- \ANETMDA-Dumps\Exercise-PN3-Analysis-process-dump-LINQPadB.pdf

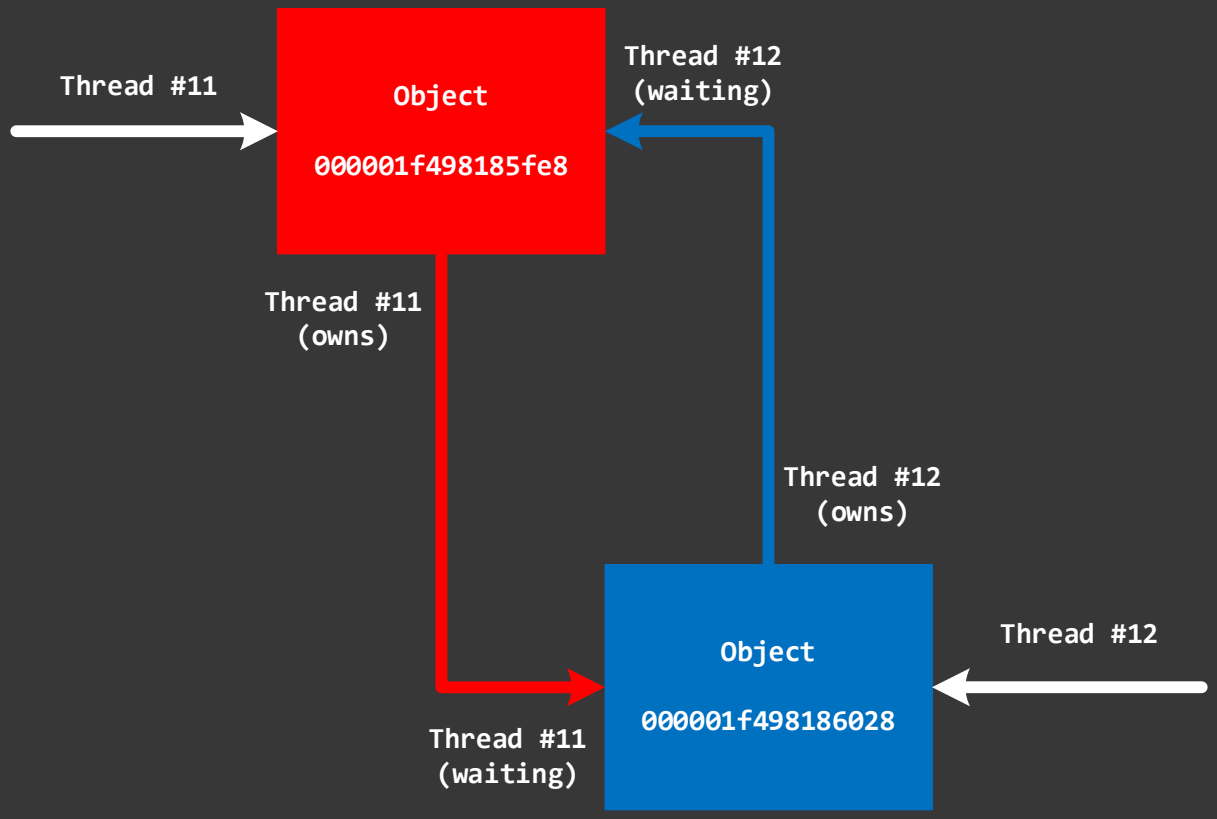
Exercise PN4

- **Goal:** Compare 64-bit process memory dump from exercise PN3 with 32-bit process memory dump
- \ANETMDA-Dumps\Exercise-PN4-Analysis-process-dump-LINQPadB-32.pdf

Exercise PN5

- **Goal:** Learn how to recognize and analyze deadlocks using SOS(EX), execution residue, handled exceptions, dump object references
- **Patterns:** Special Thread; Wait Chain; Deadlock; Execution Residue (user space); Hidden Exception (user space); Coincidental Symbolic Information; Caller-n-Callee
- **Commands:** ~*kL, !Threads, !syncblk, !DumpObj, ub, dp, !dlk, !DumpStack, !teb, dpS
- \ANETMDA-Dumps\Exercise-PN5-Analysis-process-dump-LINQPadC.pdf

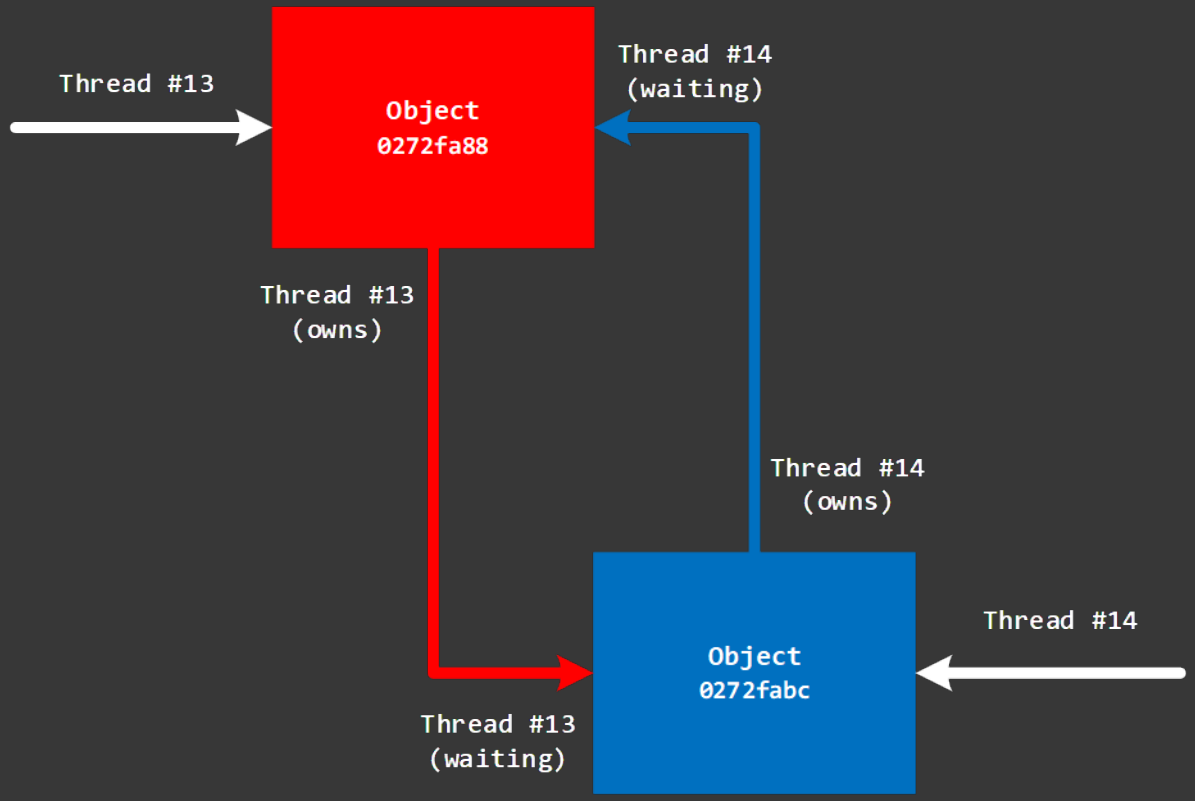
Deadlock



Exercise PN6

- **Goal:** Compare 64-bit process memory dump from exercise PN5 with 32-bit process memory dump
- **Patterns:** Execution Residue (managed space); Hidden Exception (managed space), Handled Exception
- **Commands:** !DumpStackObjects
- \ANETMDA-Dumps\Exercise-PN6-Analysis-process-dump-LINQPadC-32.pdf

Deadlock (x86)



Exercise PN7

- **Goal:** Learn how to analyze multiple managed exceptions
- **Patterns:** Managed Stack Trace Collection; Multiple Exceptions; Nested Exceptions; NULL Pointer
- \ANETMDA-Dumps\Exercise-PN7-Analysis-process-dump-ApplicationD.pdf

Exercise PN8

- **Goal:** Compare 64-bit process memory dump from exercise PN7 with 32-bit process memory dump
- \ANETMDA-Dumps\Exercise-PN8-Analysis-process-dump-ApplicationD-32.pdf

Exercise PN9

- **Goal:** Learn how to diagnose heap and handle leaks
- **Patterns:** Handle Leak; Memory Leak
- **Commands:** !heap, !address, !DumpHeap, ?, !eeheap, !GCHandles, !FinalizeQueue, !handle, .cxr
- \ANETMDA-Dumps\Exercise-PN9-Analysis-process-dump-LINQPadD.pdf

Exercise PN10

- **Goal:** Compare 64-bit process memory dump from exercise PN9 with 32-bit process memory dump
- \ANETMDA-Dumps\Exercise-PN9-Analysis-process-dump-LINQPadD-32.pdf

Exercise PN11

- **Goal:** Learn how to recognize and analyze heap corruption
- **Patterns:** Invalid Pointer; Regular Data; Managed Heap Corruption
- **Commands:** .formats, !VerifyHeap
- \ANETMDA-Dumps\Exercise-PN11-Analysis-process-dump-LINQPadE.pdf

Exercise PN12

- **Goal:** Compare 64-bit process memory dump from exercise PN11 with 32-bit process memory dump
- \ANETMDA-Dumps\Exercise-PN12-Analysis-process-dump-LINQPadE-32.pdf

Pattern Links

CLR Thread
Managed Code Exception
Nested Exceptions
Mixed Exception
Memory Leak
JIT Code
Managed Stack Trace
Multiple Exceptions
Version-Specific Extension
Caller-n-Callee
Hidden Exception

Deadlock
Duplicate Extension
Stack Trace Collection
Dynamic Memory Corruption
Special Thread
Execution Residue
Handled Exception
Annotated Disassembly
Technology-Specific Subtrace
Wait Chain
Object Distribution Anomaly

CLR-related and managed

Incorrect Stack Trace Execution Residue
NULL Pointer Handle Leak Exception Stack Trace
Software Exception Platform-Specific Debugger
Spiking Thread Hidden Exception Regular Data
Coincidental Symbolic Information

Unmanaged user space

SOS Checklist

- CLR module and SOS extension versions (lmv and .chain)
- Managed exceptions (~*e !pe -nested)
- Managed threads (!Threads -special)
- Managed stack traces (~*e !CLRStack)
- Managed execution residue (~*e !DumpStackObjects)
- Managed heap (!VerifyHeap, !DumpHeap -stat and !eeheap -gc)
- GC handles (!GCHandles)
- Finalizer queue (!FinalizeQueue)
- Sync blocks (!syncblk)

Resources

- WinDbg Help / WinDbg.org (quick links) / DumpAnalysis.org
- C# 7.0 Pocket Reference
- CLR via C#, Fourth Edition
- Advanced .NET Debugging
- Debugging Microsoft .NET 2.0 Applications
- Shared Source CLI 2.0 Internals
- Accelerated Windows Memory Dump Analysis, 4th edition
- Memory Dump Analysis Anthology (Volumes 1 – 10)

Q&A

Please send your feedback using the contact form on PatternDiagnostics.com

Thank you for attendance!