Windows Debugging[4]

Accelerated

Version 3.0

Dmitry Vostokov
Software Diagnostics Services

Prerequisites

- Debugging in Visual Studio

or

- Basic crash dump analysis

© 2022 Software Diagnostics Services

Why WinDbg/WinDbg Preview?

- Easy to install (WinDbg Preview)
- Production debugging
- Redistributable (WinDbg)
- Kernel mode debugging
- Time Travel Debugging (WinDbg Preview)

© 2022 Software Diagnostics Services

Training Goals

- Review fundamentals
- Learn live debugging techniques
- See how software diagnostics is used during debugging

© 2022 Software Diagnostics Services

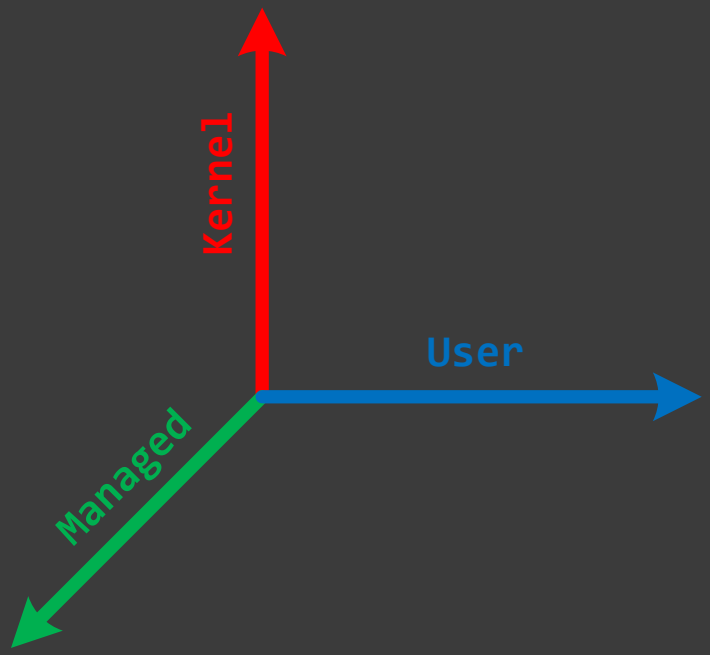
Training Principles

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

© 2022 Software Diagnostics Services

Course Idea

Chemistry³: Introducing Inorganic, Organic, and Physical Chemistry book



© 2022 Software Diagnostics Services

Debugging TV

- www.debugging.tv (more than 40 episodes)
- PDB Symbols: episodes 0x01 – 0x04
- Kernel debugging setup: episode 0x25

© 2022 Software Diagnostics Services

Schedule Summary

Day 1

- Debugging Fundamentals (30 minutes)
- User Mode Debugging (1 hour and 30 minutes)

Day 2

- User Mode Debugging (1 hour and 30 minutes)
- Kernel Debugging (30 minutes)

Day 3

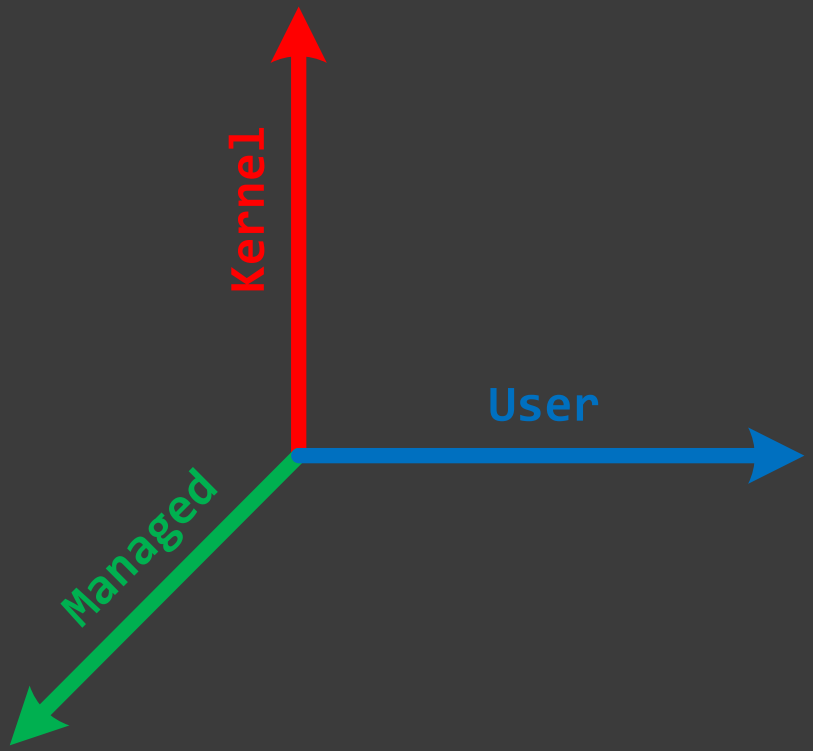
- Kernel Debugging (1 hour)
- Managed Debugging (30 minutes)
- Time-Travel Debugging (30 minutes)

© 2022 Software Diagnostics Services

Part 1: Fundamentals

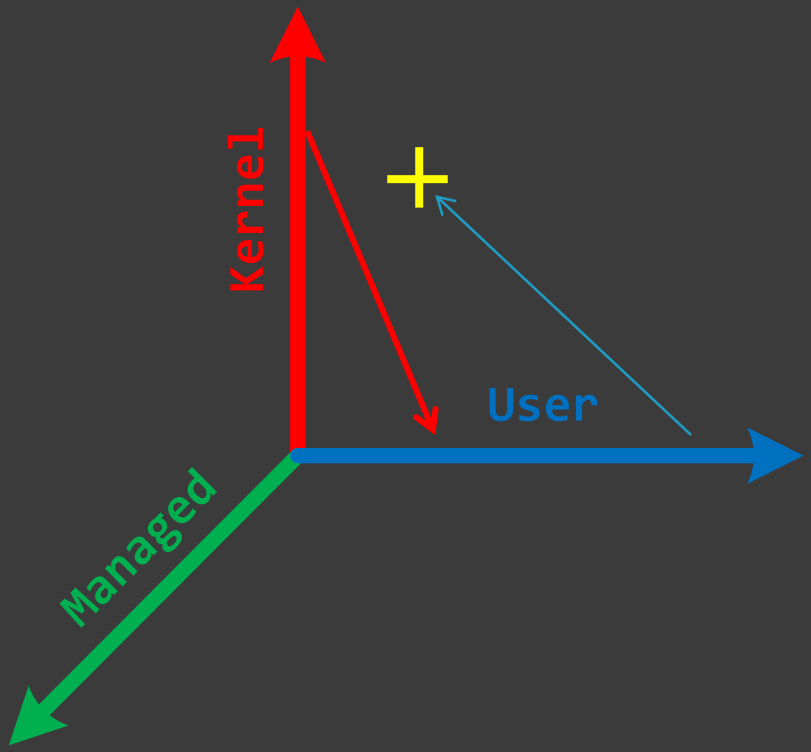
© 2022 Software Diagnostics Services

Memory Space^3



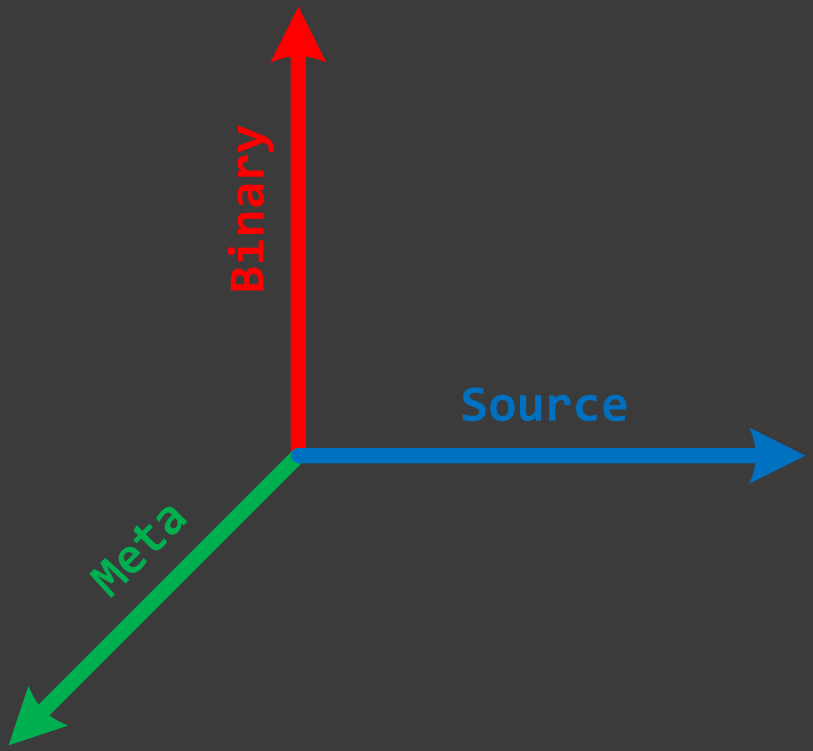
© 2022 Software Diagnostics Services

Execution Mode^3



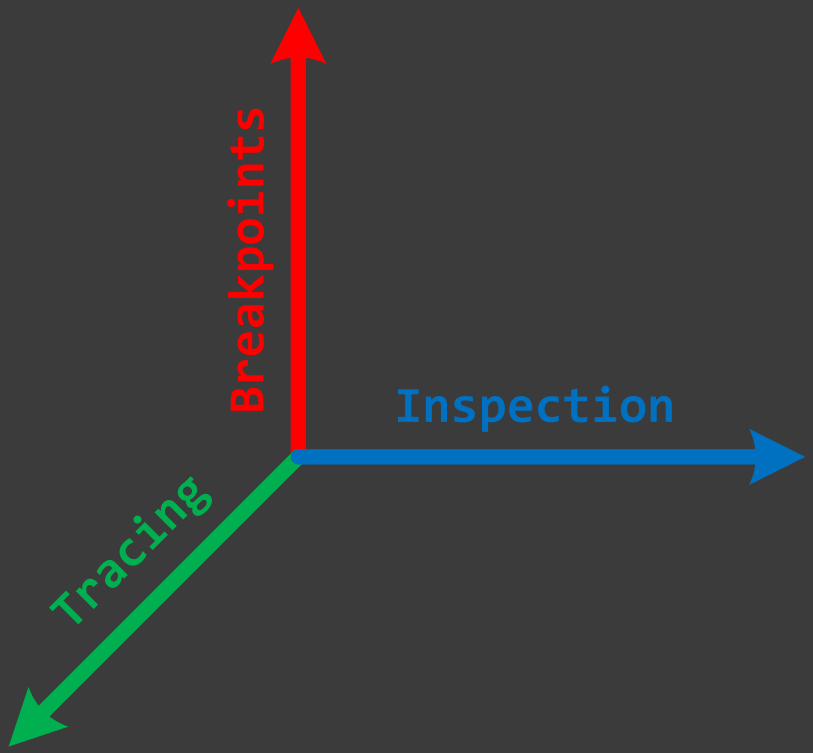
© 2022 Software Diagnostics Services

Code^3



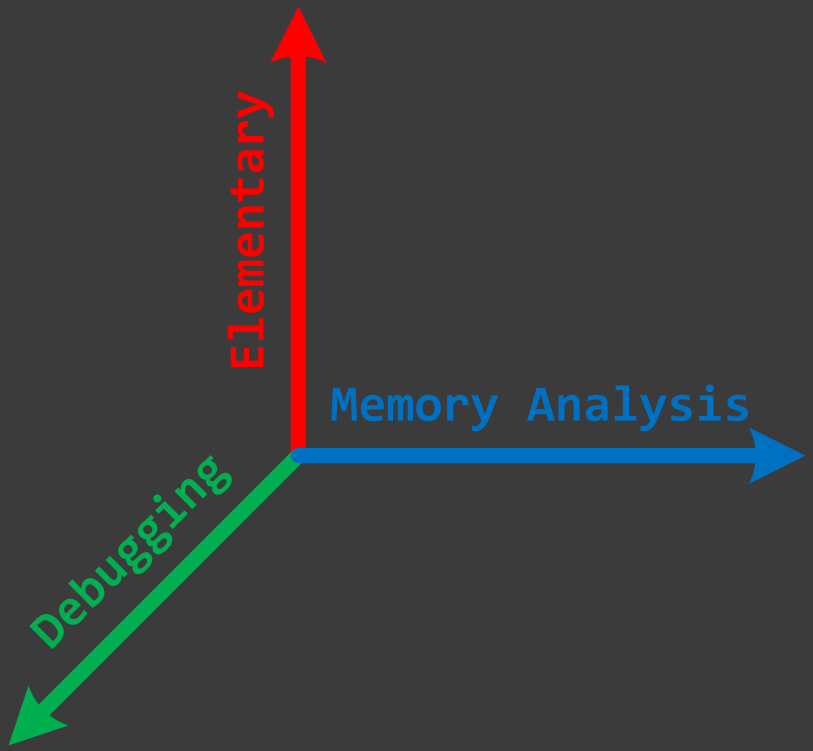
© 2022 Software Diagnostics Services

Live Debugging Technique[3]



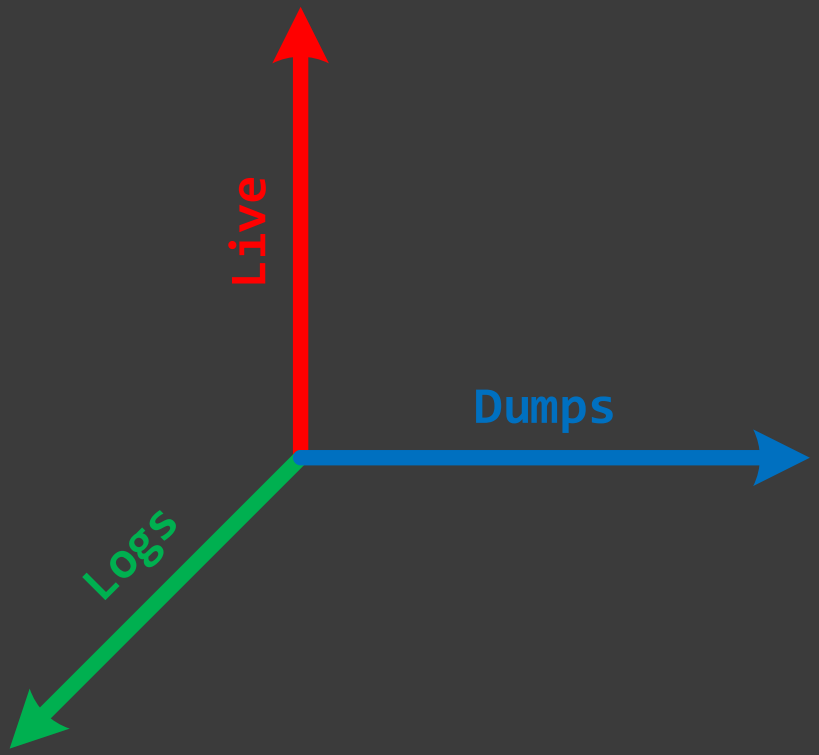
© 2022 Software Diagnostics Services

Pattern^3



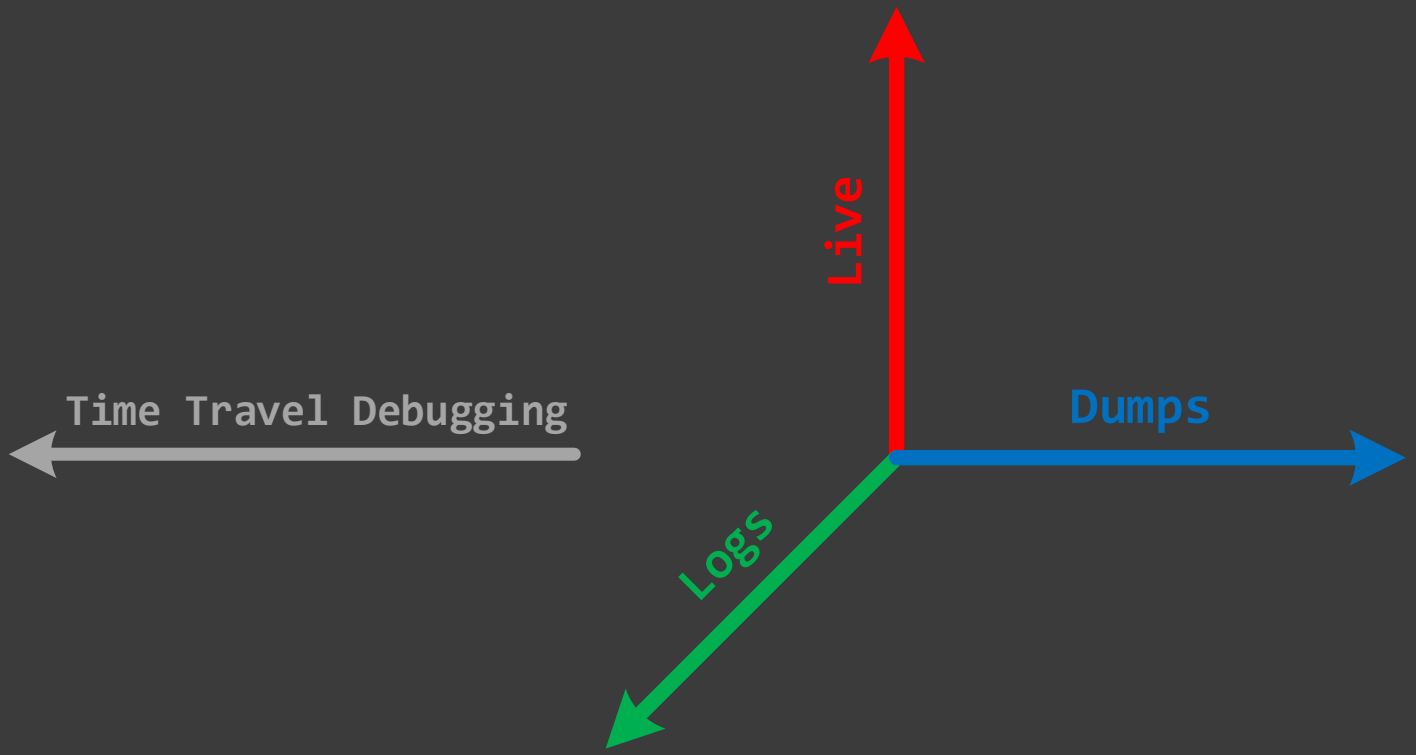
© 2022 Software Diagnostics Services

Debugging Paradigm^3



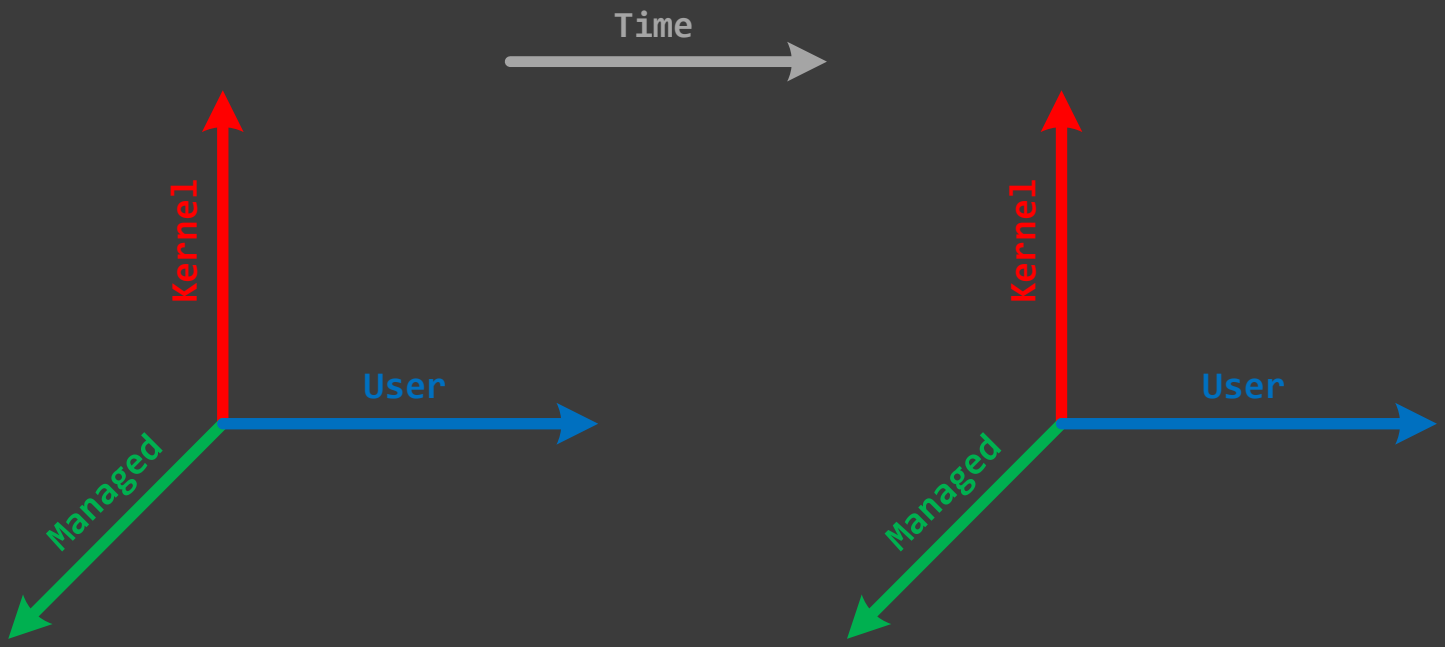
© 2022 Software Diagnostics Services

Debugging Paradigm[4]



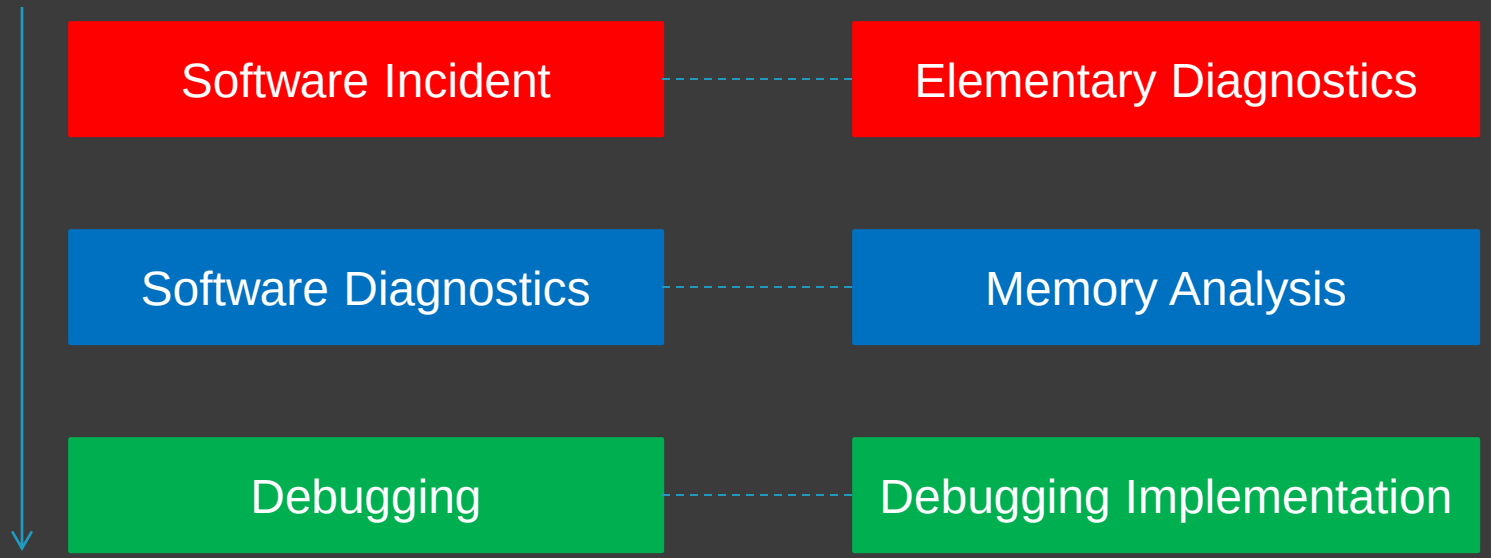
© 2022 Software Diagnostics Services

Memory Spacetime



© 2022 Software Diagnostics Services

Pattern Mapping



© 2022 Software Diagnostics Services

Elementary Diagnostics

- Functional
  - Use-case Deviation
- Non-functional
  - Crash
  - Hang (includes delays)
  - Counter Value (includes resource leaks, CPU spikes)
  - Error Message

© 2022 Software Diagnostics Services

Analysis Patterns

- Memory Analysis catalog
- Software Trace and Log Analysis catalog

© 2022 Software Diagnostics Services

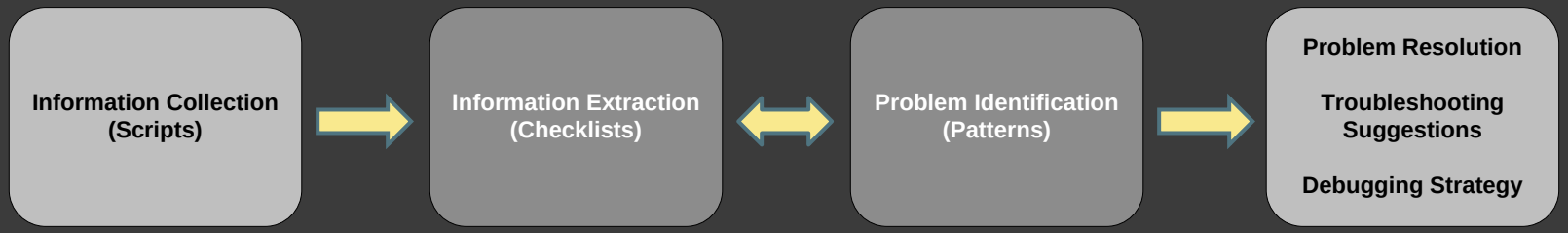
Pattern-Oriented Diagnostic Analysis

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

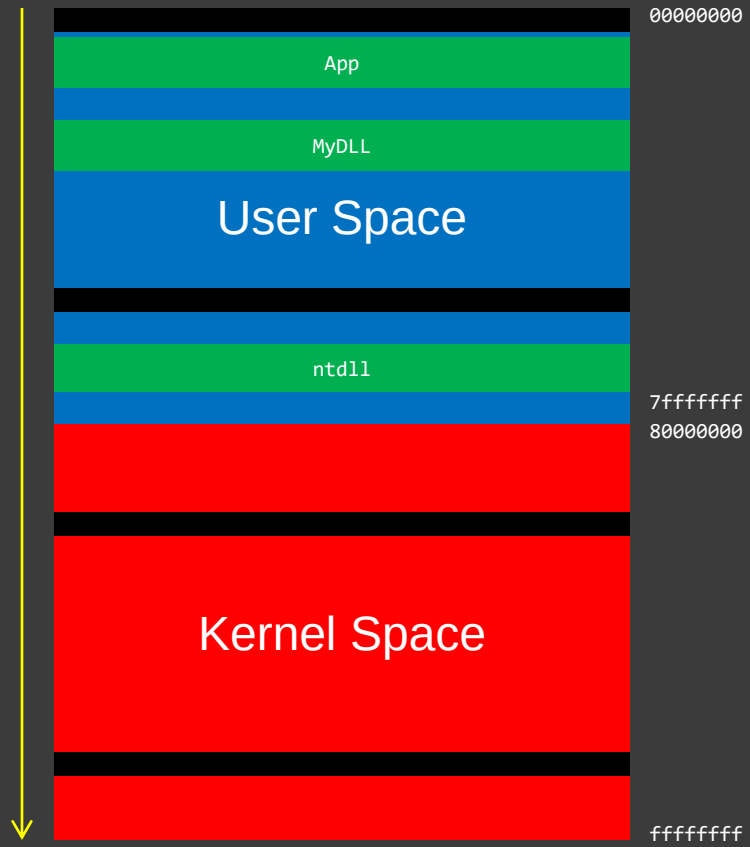
© 2022 Software Diagnostics Services

Unified Debugging Patterns

- Analysis (software diagnostics)
- Architecture/Design of debugging
- Implementation of debugging
- Usage/presentation of debugging (for example, Watch dialog)

© 2022 Software Diagnostics Services

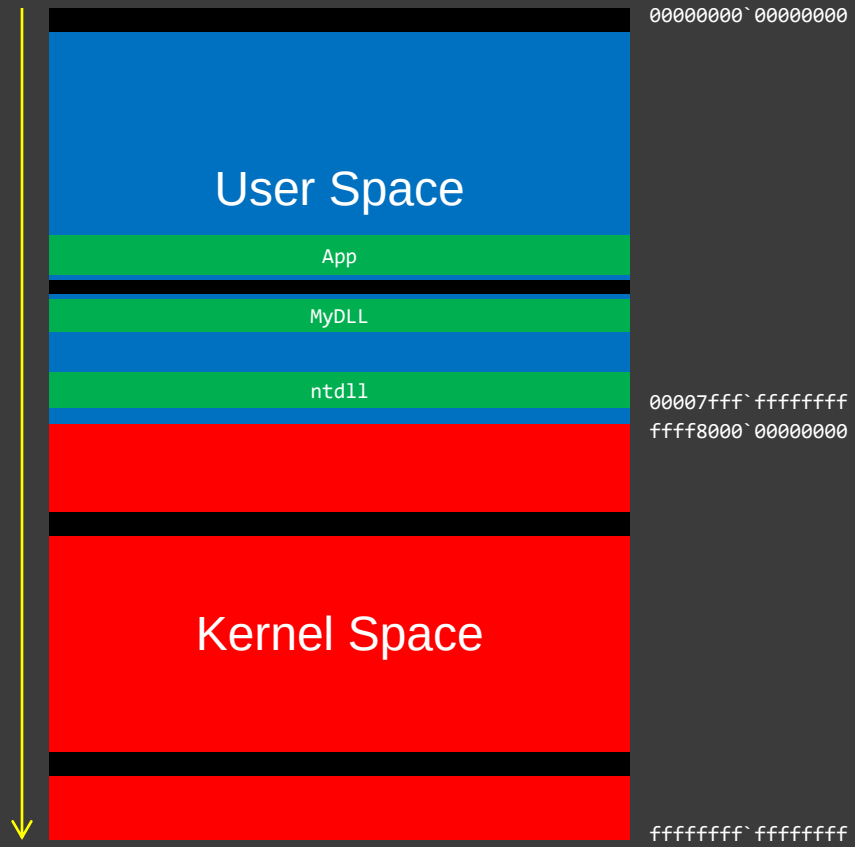
Space Review (x86)




```
0:000> lm
start    end        module name
00ed0000 00f0e000   App
0f450000 0f530000   MyDLL
638d0000 63b2d000   CoreUIComponents
63b30000 63bbb000   CoreMessaging
642d0000 64435000   twinapi_appcore
6beb0000 6c0b4000   comctl32
6d7e0000 6d803000   dwmapi
6d930000 6d9ac000   uxtheme
6d9c0000 6da96000   WinTypes
6dac0000 6dce8000   iertutil
6dcf0000 6de8c000   urlmon
700b0000 70230000   propsys
70230000 7029c000   winspool
702a0000 702f6000   oleacc
70530000 70559000   ntmarta
70af0000 70b08000   mpr
74080000 74099000   bcrypt
74280000 742b0000   IPHLPAPI
746a0000 746aa000   CRYPTBASE
746b0000 746d0000   sspicli
74870000 74892000   gdi32
749a0000 74b2d000   user32
74b60000 74c1f000   msvcrt
74c20000 74c98000   advapi32
74ca0000 74d28000   SHCore
74d30000 74f14000   KERNELBASE
74f20000 7517c000   combase
75180000 75260000   kernel32
75260000 752e3000   clbcatq
752f0000 75434000   msctf
75440000 7678a000   shell32
767a0000 767f8000   bcryptPrimitives
76800000 7691e000   ucrtbase
76920000 769e0000   rpcrt4
769e0000 769f7000   win32u
76a00000 76a39000   cfgmgr32
76ee0000 76ee8000   fltLib
76f00000 76f45000   powrprof
76f50000 76f95000   shlwapi
76fa0000 77036000   oleaut32
77200000 77218000   profapi
77310000 77354000   sechost
773c0000 77524000   gdi32full
77590000 775b6000   imm32
775e0000 776b6000   comdlg32
776c0000 77c7a000   windows_storage
77c80000 77cfd000   msvcp_win
77df0000 77f80000   ntdll
```

© 2022 Software Diagnostics Services

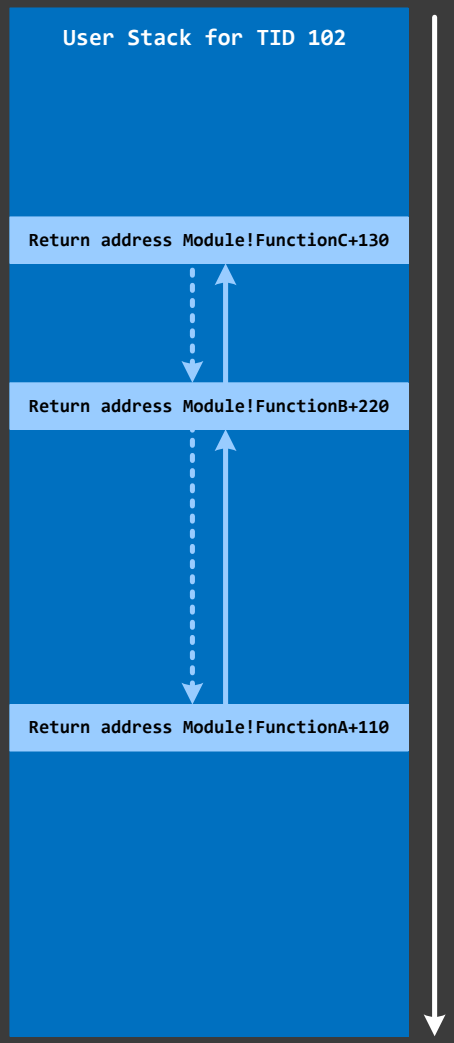
Space Review (x64)




```
0:000> lm
start             end                 module name
00007ff7`73080000 00007ff7`7330f000   App
00007ff8`44990000 00007ff8`44a27000   MyDLL
00007ff8`46ad0000 00007ff8`46d39000   comctl32
00007ff8`56d70000 00007ff8`56ddb000   oleacc
00007ff8`56de0000 00007ff8`56f7a000   GdiPlus
00007ff8`58dd0000 00007ff8`58e54000   winspool
00007ff8`5c6b0000 00007ff8`5c9ce000   CoreUIComponents
00007ff8`5ca10000 00007ff8`5ca75000   ninput
00007ff8`5ce40000 00007ff8`5cf8d000   WinTypes
00007ff8`5db50000 00007ff8`5dd04000   propsys
00007ff8`5ec10000 00007ff8`5ec3a000   winmmbase
00007ff8`5ec40000 00007ff8`5ec63000   winmm
00007ff8`5ee10000 00007ff8`5ee17000   msimg32
00007ff8`5f210000 00007ff8`5f2ea000   CoreMessaging
00007ff8`5f590000 00007ff8`5f628000   uxtheme
00007ff8`5f820000 00007ff8`5f849000   dwmapi
00007ff8`601c0000 00007ff8`601f1000   ntmarta
00007ff8`606d0000 00007ff8`60708000   IPHLPAPI
00007ff8`60c30000 00007ff8`60c55000   bcrypt
00007ff8`610d0000 00007ff8`6111c000   powrprof
00007ff8`61120000 00007ff8`6112a000   fltLib
00007ff8`61150000 00007ff8`61161000   kernel_appcore
00007ff8`61170000 00007ff8`6118f000   profapi
00007ff8`61190000 00007ff8`6128a000   ucrtbase
00007ff8`61290000 00007ff8`612d9000   cfgmgr32
00007ff8`612e0000 00007ff8`61472000   gdi32full
00007ff8`61480000 00007ff8`614a0000   win32u
00007ff8`616f0000 00007ff8`61dfd000   windows_storage
00007ff8`61e00000 00007ff8`61e7a000   bcryptPrimitives
00007ff8`61e80000 00007ff8`61f1f000   msvcp_win
00007ff8`61f20000 00007ff8`62193000   KERNELBASE
00007ff8`62250000 00007ff8`622ab000   sechost
00007ff8`622d0000 00007ff8`63710000   shell32
00007ff8`63710000 00007ff8`637d2000   oleaut32
00007ff8`637e0000 00007ff8`63881000   advapi32
00007ff8`63890000 00007ff8`638bd000   imm32
00007ff8`638c0000 00007ff8`63a50000   user32
00007ff8`63ae0000 00007ff8`63c55000   msctf
00007ff8`63ce0000 00007ff8`63d08000   gdi32
00007ff8`63d20000 00007ff8`63dbe000   msvcrt
00007ff8`63fc0000 00007ff8`64069000   SHCore
00007ff8`64070000 00007ff8`64194000   rpcrt4
00007ff8`641a0000 00007ff8`641f1000   shlwapi
00007ff8`642f0000 00007ff8`64613000   combase
00007ff8`64620000 00007ff8`64771000   ole32
00007ff8`64820000 00007ff8`648d2000   kernel32
00007ff8`64dc0000 00007ff8`64fa1000   ntdll
```

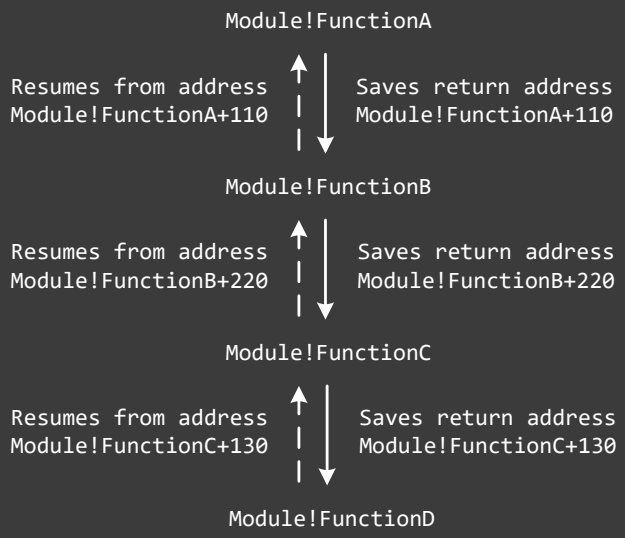
© 2022 Software Diagnostics Services

Thread Stack Trace



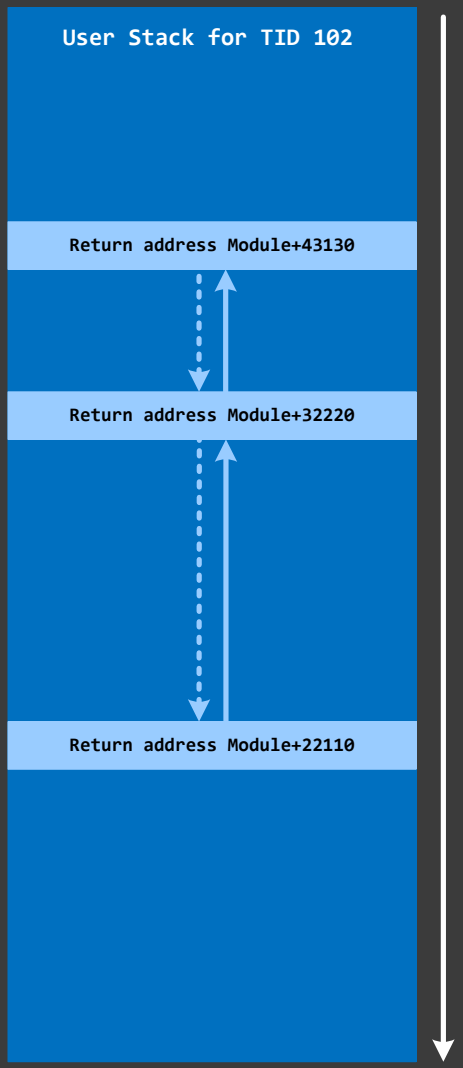

```
FunctionA()
{
  ...
  FunctionB();
  ...
}

FunctionB()
{
  ...
  FunctionC();
  ...
}

FunctionC()
{
  ...
  FunctionD();
  ...
}
```

```
0:000> k
Module!FunctionD
Module!FunctionC+130
Module!FunctionB+220
Module!FunctionA+110
```



© 2022 Software Diagnostics Services

Thread Stack Trace (no PDB)

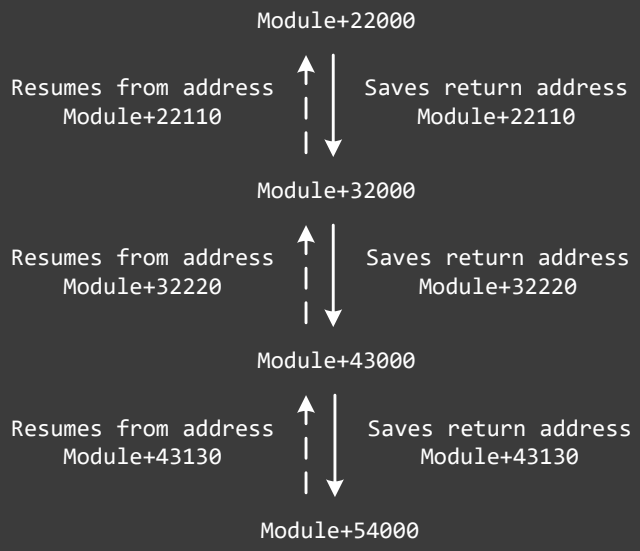



```
FunctionA()
{
  ...
  FunctionB();
  ...
}

FunctionB()
{
  ...
  FunctionC();
  ...
}

FunctionC()
{
  ...
  FunctionD();
  ...
}
```

Symbol file Module.pdb

```
FunctionA 22000 - 23000
FunctionB 32000 - 33000
FunctionC 43000 – 44000
FunctionD 54000 - 55000
```

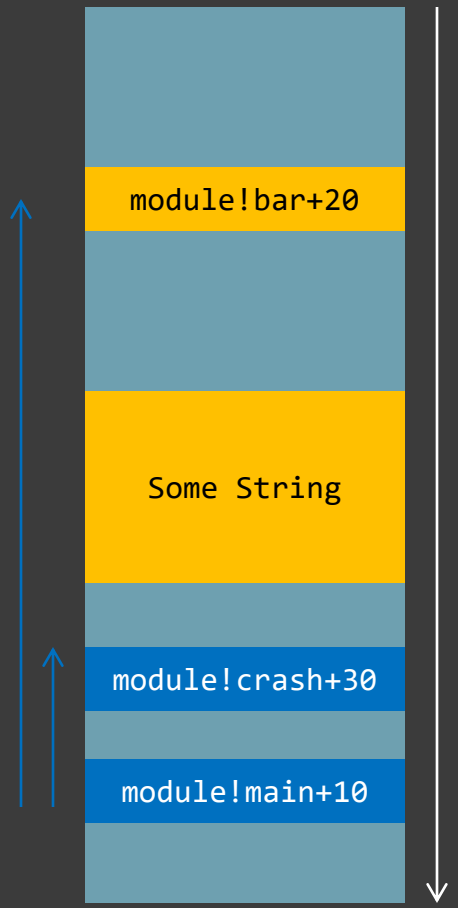
No symbols for Module




```
0:000> k
Module+0
Module+43130
Module+32220
Module+22110
```

© 2022 Software Diagnostics Services

Thread Raw Stack Data




```
void main()
{
     foo();
     crash();
}

void foo()
{
     char sz[256] = “Some String”;
     bar();
}

void bar()
{
     do();
}

void crash()
{
     WER();
}
```

```
0:000> k
module!crash+30
module!main+10
```

© 2022 Software Diagnostics Services

First vs. Second Chance

- First chance exceptions

WinDbg is notified of an exception, you can ignore it

- Second chance exceptions

If the exception wasn't handled (for example, by a catch block) WinDbg is notified again

- Relation to crash dumps

© 2022 Software Diagnostics Services

Part 2: Practice Exercises

© 2022 Software Diagnostics Services

Links

- Applications:

Download links are in exercises UD0 and KD0.

- Exercise Transcripts:

Included in this book.

© 2022 Software Diagnostics Services

Warning

Because of live debugging, due to differences in actual systems and ASLR (Address Space Layout Randomization), when you launch applications, actual addresses in WinDbg command output may be different from those shown in exercise transcripts.

© 2022 Software Diagnostics Services

Exercise UD0

- **Goal:** Download and verify your WinDbg Preview installation
- **Memory Analysis Patterns:** Incorrect Stack Trace
- \AWD4\Exercise-UD0-Download-WinDbg-Preview.pdf

© 2022 Software Diagnostics Services

User Mode Debugging

Exercises UD1 – UD8

© 2022 Software Diagnostics Services

Exercise UD1

- **Goal:** Learn how code generation parameters can influence process execution behavior
- **Elementary Diagnostics Patterns:** Crash
- **Memory Analysis Patterns:** Exception Stack Trace; Constant Subtrace
- **Debugging Implementation Patterns:** Scope; Variable Value; Type Structure; Code Breakpoint
- \AWD4\Exercise-UD1.pdf

© 2022 Software Diagnostics Services

Exercise UD2

- **Goal:** Learn how to use hardware breakpoints to catch data corruption
- **Elementary Diagnostics Patterns:** Counter Value
- **Memory Analysis Patterns:** Unloaded Module; Memory Leak (Process Heap); Corrupt Structure; Abnormal Value (*from trace analysis patterns*)
- **Debugging Implementation Patterns:** Break-in; Code Breakpoint; Scope; Variable Value; Data Breakpoint
- \AWD4\Exercise-UD2.pdf

© 2022 Software Diagnostics Services

Exercise UD3

- **Goal:** Learn how to navigate parameters, static and local variables, and data structures
- **Elementary Diagnostics Patterns:** Crash
- **Memory Analysis Patterns:** Exception Stack Trace; Stack Overflow (User Mode); String Parameter; Module Variable
- **Debugging Implementation Patterns:** Break-in; Scope; Variable Value; Type Structure
- \AWD4\Exercise-UD3.pdf

© 2022 Software Diagnostics Services

Exercise UD4

- **Goal:** Learn how to use conditional breakpoints to log behavior
- **Elementary Diagnostics Patterns:** Use-case Deviation
- **Memory Analysis Patterns:** -
- **Debugging Implementation Patterns:** Break-in; Code Breakpoint; Breakpoint Action
- \AWD4\Exercise-UD4.pdf

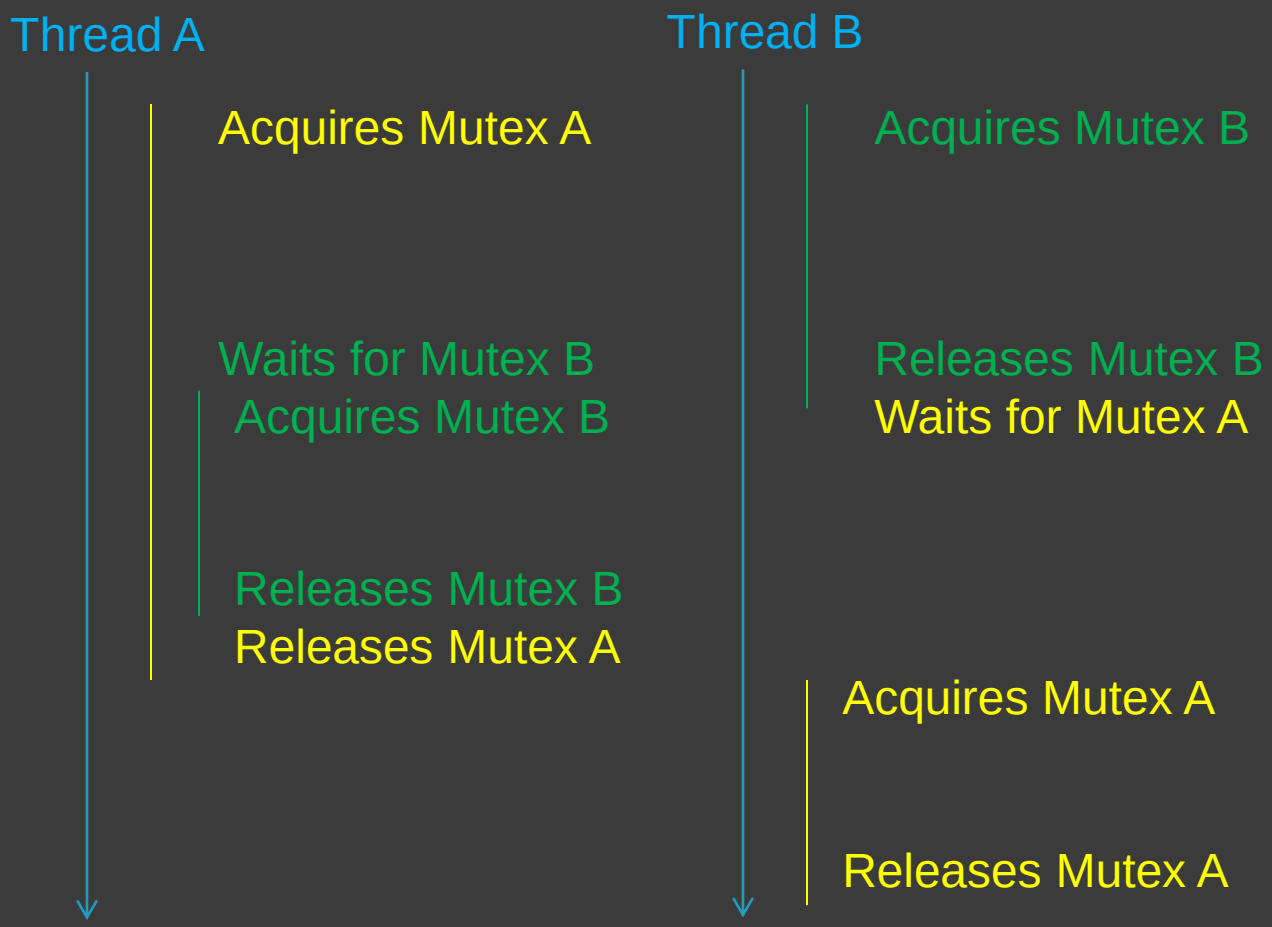
© 2022 Software Diagnostics Services

Exercise UD5

- **Goal:** Learn how to debug multiple processes and their deadlock
- **Elementary Diagnostics Patterns:** Crash; Hang
- **Memory Analysis Patterns:** Exception Stack Trace; Constant Subtrace; NULL Pointer (Data); Main Thread; Execution Residue (Unmanaged Space, User); Hidden Exception (User Space); Handled Exception (User Space); Wait Chain (Mutex Objects); Deadlock (Objects, User Space)
- **Debugging Implementation Patterns:** Break-in
- \AWD4\Exercise-UD5.pdf

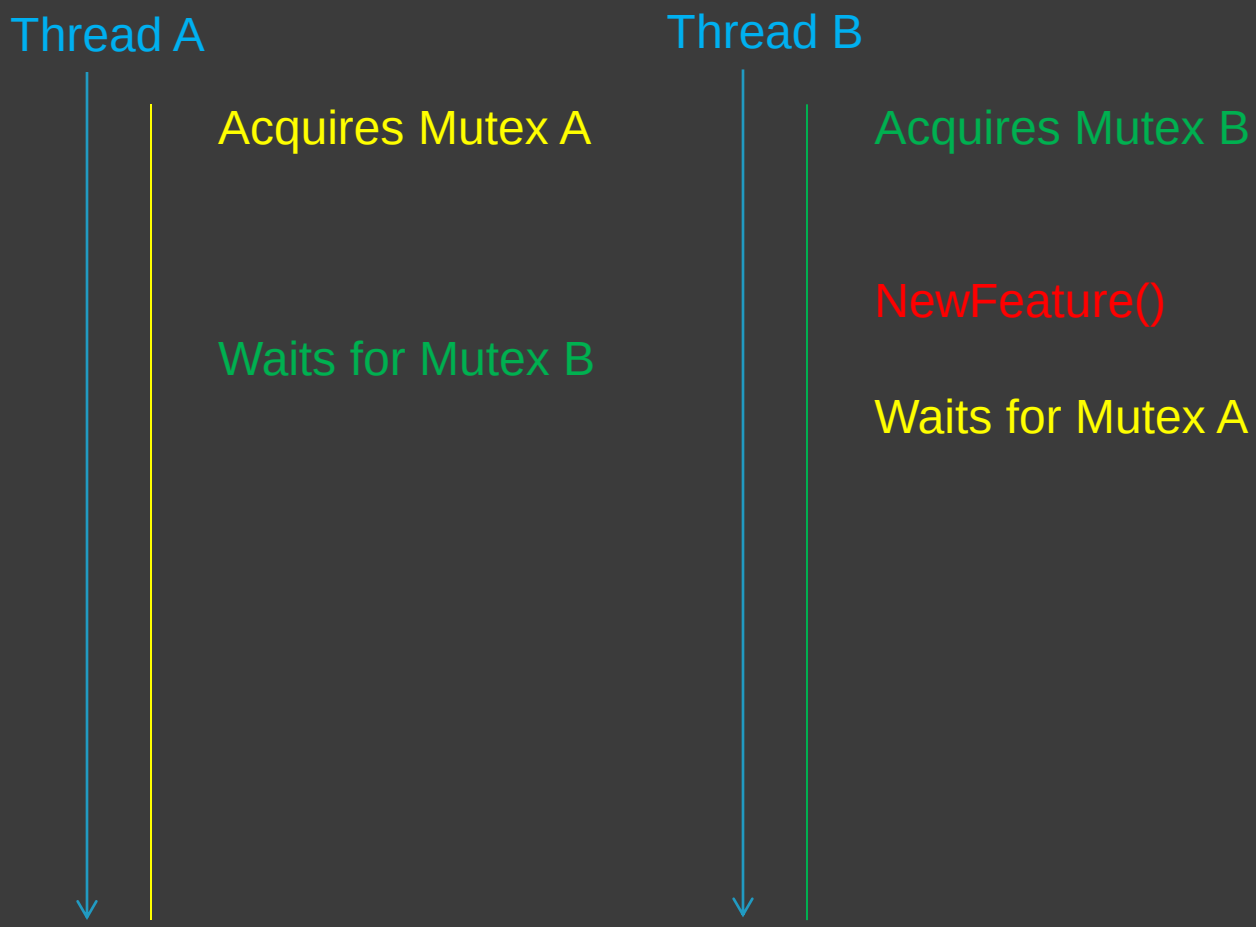
© 2022 Software Diagnostics Services

Expected Behavior



© 2022 Software Diagnostics Services

Deadlock



© 2022 Software Diagnostics Services

Exercise UD6

- **Goal:** Learn how to recognize when we need kernel level debugging
- **Elementary Diagnostics Patterns:** Hang; Counter Value
- **Memory Analysis Patterns:** Abnormal Value (*from trace analysis patterns*); Spiking Thread
- **Debugging Implementation Patterns:** Break-in; Code Breakpoint; Data Breakpoint; Code Trace
- \AWD4\Exercise-UD6.pdf

© 2022 Software Diagnostics Services

Exercise UD7

- **Goal:** Learn how to manipulate threads to debug race conditions
- **Elementary Diagnostics Patterns:** Crash
- **Memory Analysis Patterns:** Exception Stack Trace; NULL Pointer (Data)
- **Debugging Implementation Patterns:** Frozen Thread
- \AWD4\Exercise-UD7.pdf

© 2022 Software Diagnostics Services

Exercise UD8

- **Goal:** Learn how to inspect heap for signs of corruption
- **Elementary Diagnostics Patterns:** Crash
- **Memory Analysis Patterns:** Dynamic Memory Corruption (Process Heap); Module Variable; Exception Stack Trace
- **Debugging Implementation Patterns:** Break-in
- \AWD4\Exercise-UD8.pdf

© 2022 Software Diagnostics Services

Kernel Mode Debugging

Exercises KD6, KD9, KD10

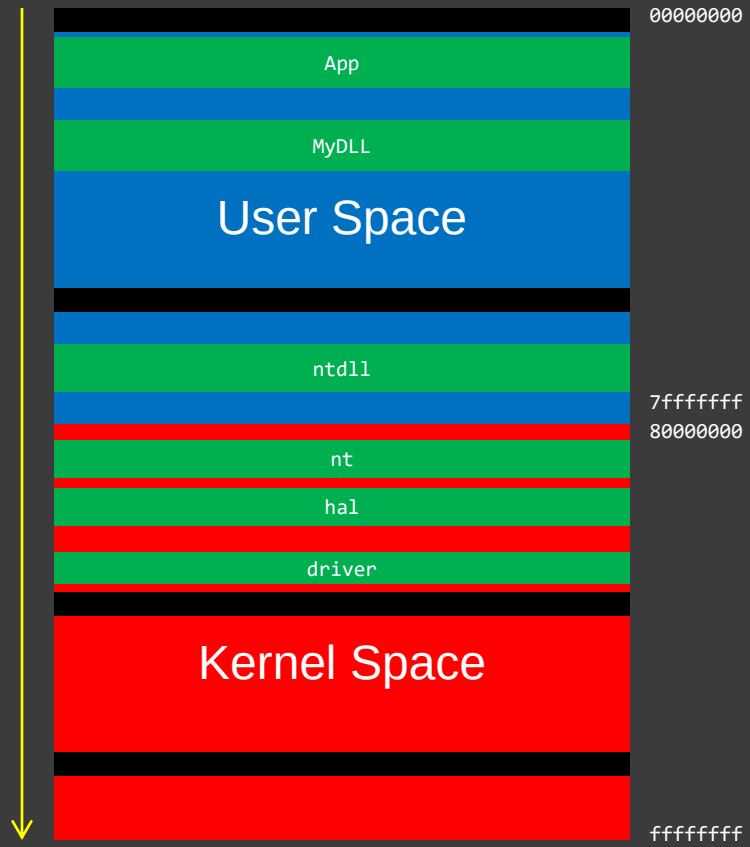
© 2022 Software Diagnostics Services

Exercise KD0

- **Goal:** Set up Hyper-V or VMware kernel debugging environment
- \AWD4\Exercise-KD0-Kernel-Debugging-Setup.pdf

© 2022 Software Diagnostics Services

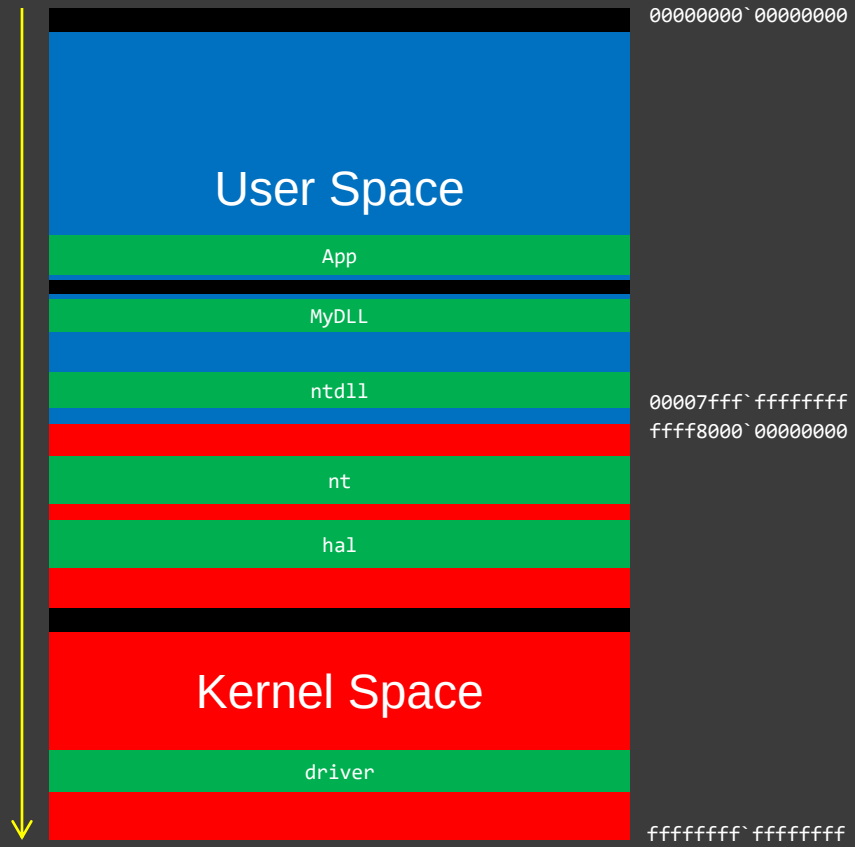
Space Review (x86)




```
0: kd> lmk
start    end        module name
80200000 8020a000   BATTC
8020a000 8020c900   compbatt
8020d000 80215000   msisadrv
80215000 8021e000   WMILIB
8021e000 8022b000   WDFLDR
8022b000 80266000   CLFS
80266000 8026e000   BOOTVID
[...]
81800000 81ba1000   nt
81ba1000 81bd5000   hal
[...]
87eb3000 87ed6000   ndiswan
87ed6000 87ee1000   ndistapi
87ee1000 87ef8000   rasl2tp
87ef8000 87f03000   TDI
[...]
937b4000 93800000   srv
9446d000 94480000   dump_LSI_SCSI
96ca1000 96cc9000   fastfat
```

© 2022 Software Diagnostics Services

Space Review (x64)

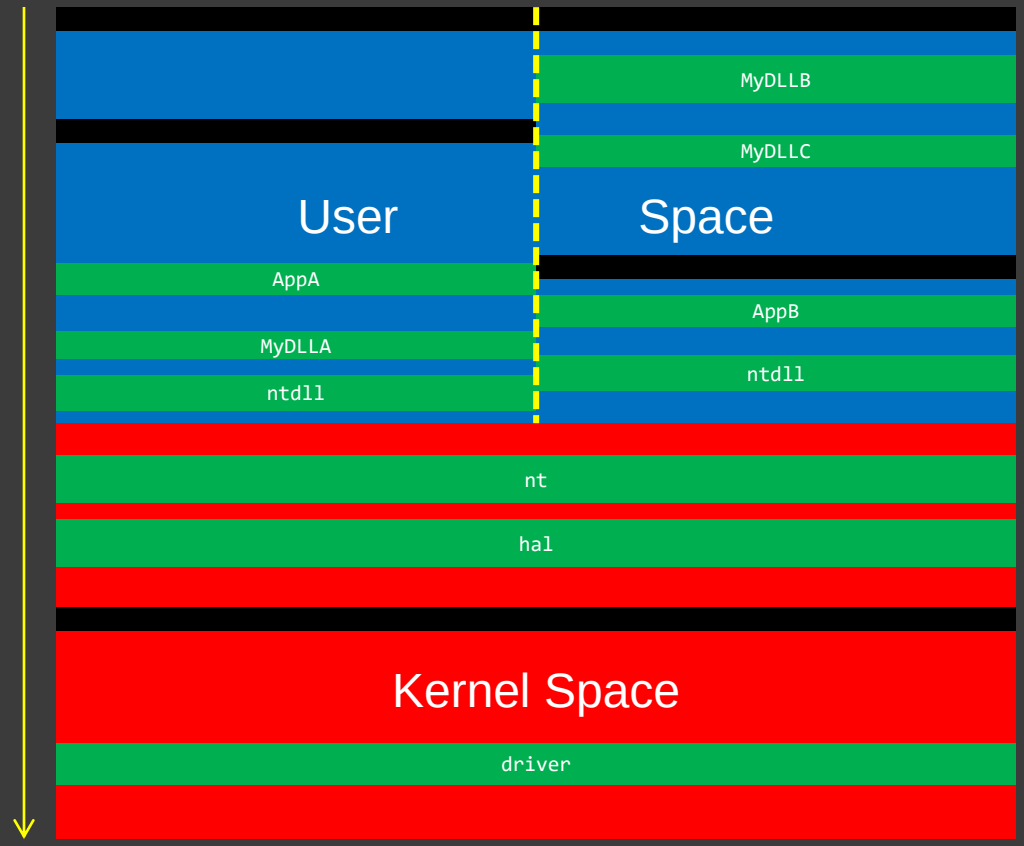



```
0: kd> lmk
start             end                 module name
ffffd414`7bc00000 ffffd414`7bf90000   win32kfull
ffffd414`7bf90000 ffffd414`7c1c4000   win32kbase
ffffd414`7c1d0000 ffffd414`7c20f000   cdd
ffffd414`7c730000 ffffd414`7c7ac000   win32k
fffff802`1cc05000 fffff802`1d566000   nt
fffff802`1d566000 fffff802`1d5f2000   hal
fffff802`1d800000 fffff802`1d80b000   kd
fffff803`9f200000 fffff803`9f264000   CLFS
fffff803`9f270000 fffff803`9f294000   tm
fffff803`9f2a0000 fffff803`9f2b7000   PSHED
fffff803`9f2c0000 fffff803`9f2cb000   BOOTVID
[...]
fffff803`a0290000 fffff803`a0534000   tcpip
fffff803`a0540000 fffff803`a05b6000   fwpkclnt
fffff803`a05c0000 fffff803`a05ed000   wfplwfs
fffff803`a05f0000 fffff803`a06ac000   fvevol
fffff803`a06b0000 fffff803`a06bb000   volume
fffff803`a06c0000 fffff803`a0727000   volsnap
fffff803`a0730000 fffff803`a077c000   rdyboost
fffff803`a0780000 fffff803`a07a4000   mup
fffff803`a07b0000 fffff803`a07c1000   iorate
fffff803`a07e0000 fffff803`a07fc000   disk
[...]
fffff803`a3020000 fffff803`a3028000   driver
fffff803`a3040000 fffff803`a304f000   dump_diskdump
fffff803`a3070000 fffff803`a308f000   dump_LSI_SAS
fffff803`a30b0000 fffff803`a30cd000   dump_dumpfve
fffff803`a3280000 fffff803`a3346000   dxgmms2
fffff803`a3350000 fffff803`a3398000   WUDFRd
fffff803`a33a0000 fffff803`a33c7000   luafv
fffff803`a33d0000 fffff803`a33f8000   wcifs
```

© 2022 Software Diagnostics Services

Space Review

```
0: kd> !process <address> 3f

0: kd> .process /r /p <address>

0: kd> !thread <address> 3f

0: kd> .thread /r /p <address>

0: kd> .thread /w <address>
```

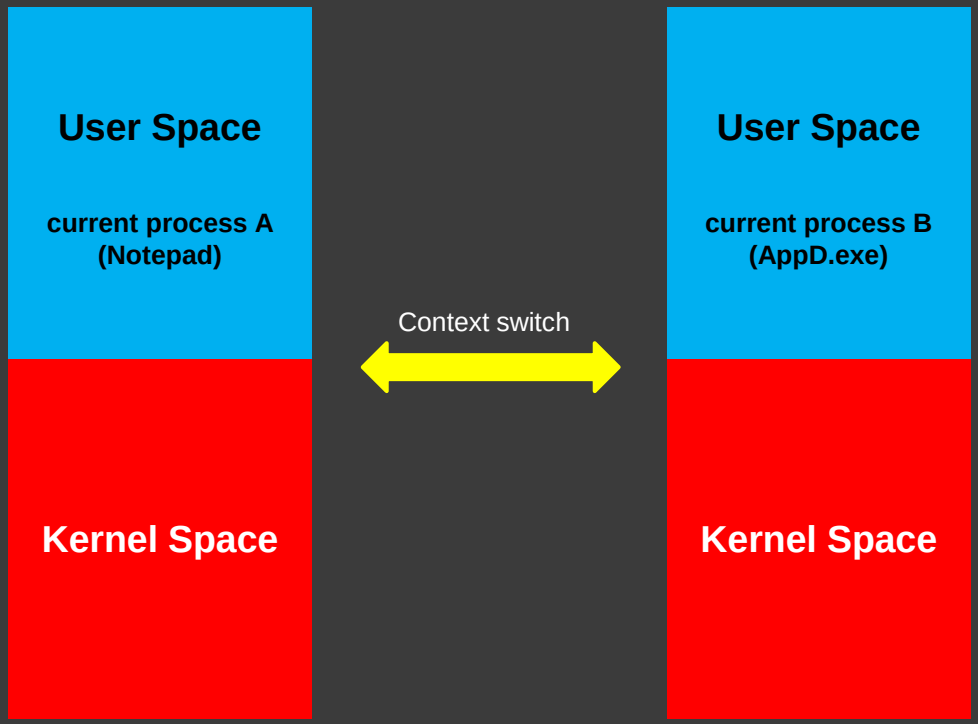


Complete stack traces (x64 + x86)

© 2022 Software Diagnostics Services

Context Switch

We always see the current process space



© 2022 Software Diagnostics Services

Common Commands

- **.logopen <file>**
  Opens a log file to save all subsequent output
- View commands
  Dump everything or selected processes and threads (context changes automatically)
- Switch commands
  Switch to a specific process or thread

© 2022 Software Diagnostics Services

View Commands

- **!process 0 0**

  Lists all processes

- **!process <address> 3f**

  Lists process information including CPU times, environment, modules and its thread stack traces

- **!thread <address> 1f**

  Shows thread information and stack trace

© 2022 Software Diagnostics Services

Switch Commands

- **.process /r /p <address>**

  Switches to a specified process. Its context becomes current. Reloads symbol files for user space. Now we can use commands like !cs

```
0: kd> .process /r /p fffffa80044d8b30
Implicit process is now fffffa80`044d8b30
Loading User Symbols
...............................
```

- **.thread <address>**

  Switches to a specified thread. Assumes the current process context.

  Now we can use commands like k*

- **.thread /r /p <address>**

  The same as the previous command but makes the thread process context current and reloads symbol files for user space:

```
0: kd> .thread /r /p fffffa80051b7060
Implicit thread is now fffffa80`051b7060
Implicit process is now fffffa80`044d8b30
Loading User Symbols
...............................
```

© 2022 Software Diagnostics Services

Exercise KD6

- **Goal:** Learn how to use kernel level debugging to catch corruption caused by a driver or other process
- **Elementary Diagnostics Patterns:** Hang; Counter Value
- **Memory Analysis Patterns:** Abnormal Value (*from trace analysis patterns*); Spiking Thread
- **Debugging Implementation Patterns:** Code Breakpoint; Data Breakpoint; Code Trace
- \AWD4\Exercise-KD6.pdf

© 2022 Software Diagnostics Services

Exercise KD9

- **Goal:** Learn how to debug a 32-bit process under x64 Windows
- **Elementary Diagnostics Patterns:** Hang
- **Memory Analysis Patterns:** Virtualized Process; Debugger Bug; Execution Residue (Unmanaged Space, User); Rough Stack Trace (Unmanaged Space); Message Box; String Parameter; Near Exception
- **Debugging Implementation Patterns:** Break-in
- \AWD4\Exercise-KD9.pdf

© 2022 Software Diagnostics Services

Exercise KD10

- **Goal:** Learn how to debug handle leaks in user and kernel mode debugging
- **Elementary Diagnostics Patterns:** Counter Value
- **Memory Analysis Patterns:** Handle Leak; Historical Information
- **Debugging Implementation Patterns:** Break-in; Usage Trace
- \AWD4\Exercise-KD10.pdf

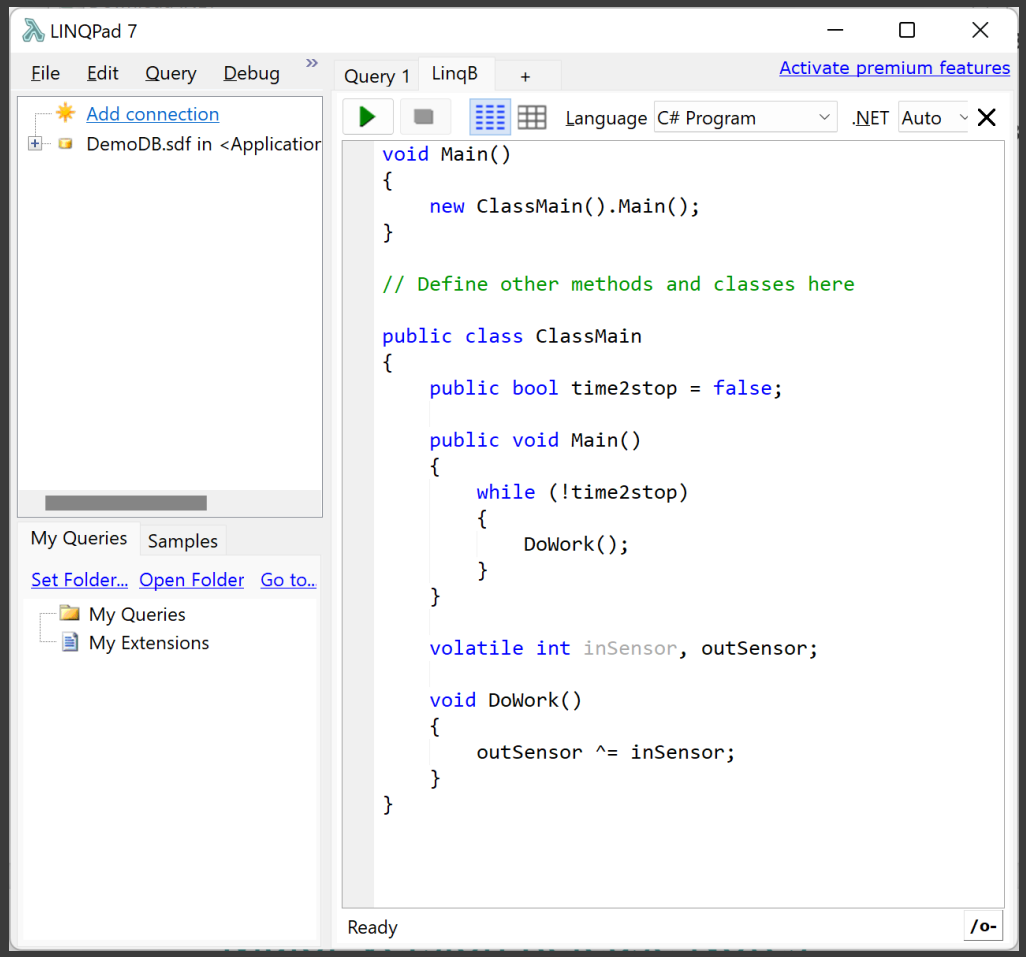
© 2022 Software Diagnostics Services

Managed Debugging

Exercise MD11

© 2022 Software Diagnostics Services

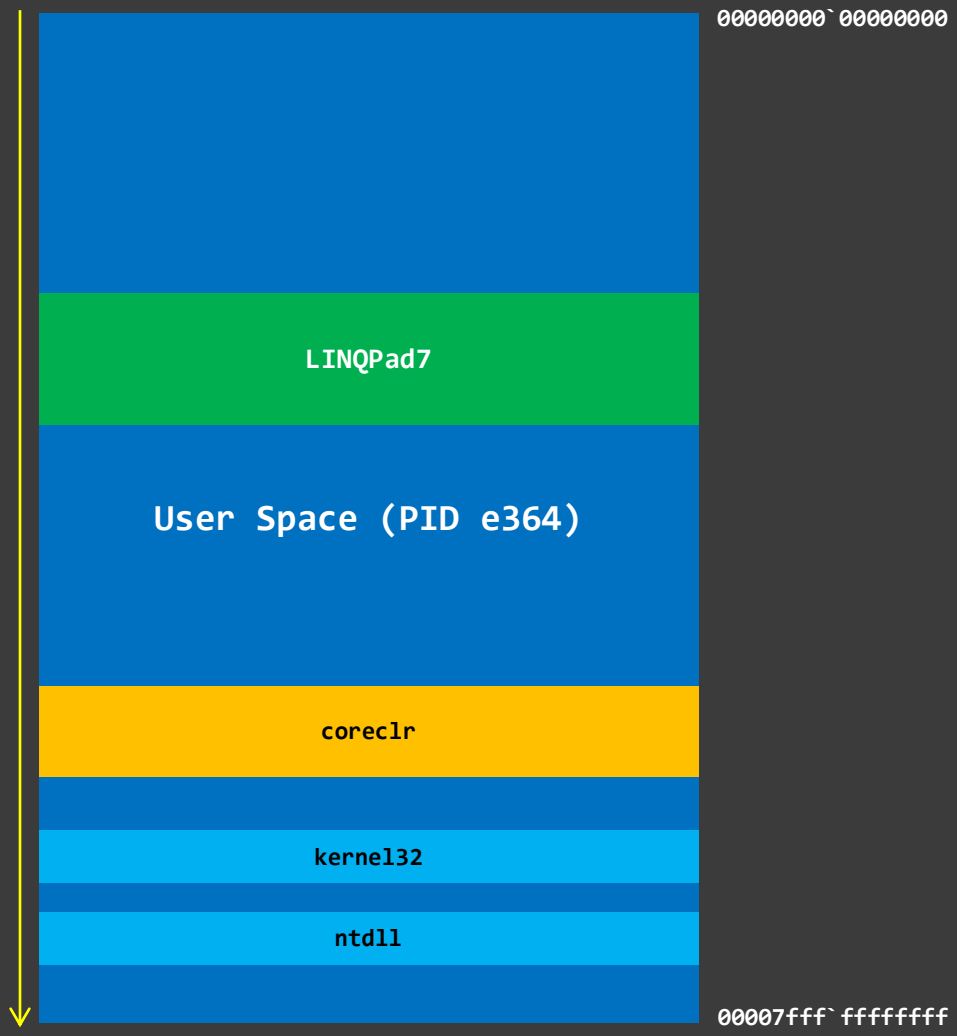
Modeling with LINQPad



http://www.linqpad.net/

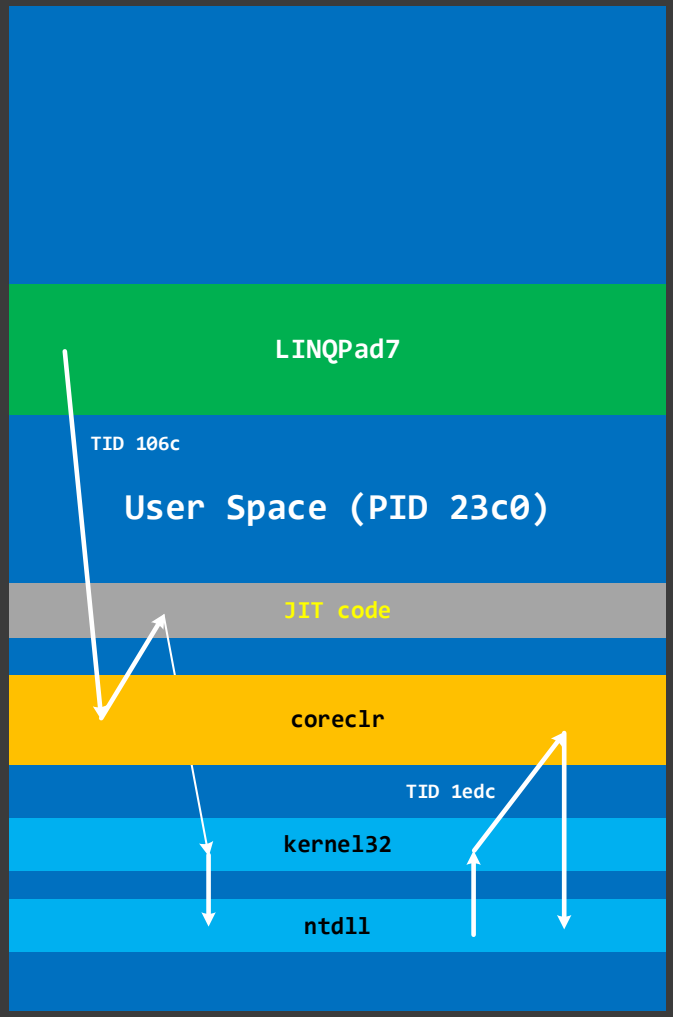
© 2022 Software Diagnostics Services

User / Managed Space



© 2022 Software Diagnostics Services

Process Threads



~<n>s
Switches between threads

k
Shows unmanaged stack trace

!Threads
Shows managed threads

!CLRStack
Shows managed stack trace

© 2022 Software Diagnostics Services

Stack Trace Example

```
0:000> kL
 # Child-SP          RetAddr           Call Site
00 00000029`6e3add78 00007ffa`7eb90458 win32u!NtUserWaitMessage+0x14
01 00000029`6e3add80 00007ffa`d7244d61 0x00007ffa`7eb90458
02 00000029`6e3ade40 00007ffa`d7247486 System_Windows_Forms!...+0x581
03 00000029`6e3adf50 00007ffa`d7247026 System_Windows_Forms!...+0x416
04 00000029`6e3ae000 00007ffa`d6ef212b System_Windows_Forms!...+0x46
05 00000029`6e3ae060 00007ffa`7cb021e4 System_Windows_Forms!...+0x3b
06 00000029`6e3ae0a0 00007ffa`7cae2534 0x00007ffa`7cb021e4
07 00000029`6e3ae350 00007ffa`7c70a16d 0x00007ffa`7cae2534
08 00000029`6e3ae680 00007ffa`7c7015c2 0x00007ffa`7c70a16d
09 00000029`6e3ae6e0 00007ffa`dc226ce3 0x00007ffa`7c7015c2
0a 00000029`6e3ae7b0 00007ffa`dc1bcc42 coreclr!CallDescrWorkerInternal+0x83
0b (Inline Function) --------`-------- coreclr!CallDescrWorkerWithHandler+0x57
0c 00000029`6e3ae7f0 00007ffa`dc1c3c09 coreclr!MethodDescCallSite::...+0x196
0d (Inline Function) --------`-------- coreclr!MethodDescCallSite::Call+0xb
0e 00000029`6e3ae930 00007ffa`dc1c4097 coreclr!RunMain+0x1f5
0f 00000029`6e3aeb10 00007ffa`dc1c4841 coreclr!Assembly::ExecuteMainMethod+0x1cb
10 00000029`6e3aeea0 00007ffa`dc0e21c1 coreclr!CorHost2::ExecuteAssembly+0x221
11 00000029`6e3af030 00007ffb`2ef54e2d coreclr!coreclr_execute_assembly+0x101
12 00000029`6e3af0d0 00007ffb`2ef62e27 hostpolicy!coreclr_t::execute_assembly+0x2d
13 00000029`6e3af120 00007ffb`2ef62a36 hostpolicy!run_app_for_context+0x387
14 00000029`6e3af280 00007ffb`2ef64262 hostpolicy!run_app+0x46
15 00000029`6e3af2d0 00007ffb`36ff3a7e hostpolicy!corehost_main+0x132
16 00000029`6e3af480 00007ffb`36ff72d8 hostfxr!execute_app+0x1de
17 (Inline Function) --------`-------- hostfxr!?A0xd51c85dd::read_config...+0x10a
18 00000029`6e3af570 00007ffb`36ff5b5b hostfxr!fx_muxer_t::handle_exec_...+0x214
19 00000029`6e3af660 00007ffb`36ff2109 hostfxr!fx_muxer_t::execute+0x39b
1a 00000029`6e3af7a0 00007ff6`ea712361 hostfxr!hostfxr_main_startupinfo+0x89
1b 00000029`6e3af8a0 00007ff6`ea712758 LINQPad6!exe_start+0x651
1c 00000029`6e3afad0 00007ff6`ea714608 LINQPad6!wmain+0x88
1d (Inline Function) --------`-------- LINQPad6!invoke_main+0x22
1e 00000029`6e3afb00 00007ffb`58387034 LINQPad6!__scrt_common_main_seh+0x10c
1f 00000029`6e3afb40 00007ffb`5a002651 kernel32!BaseThreadInitThunk+0x14
20 00000029`6e3afb70 00000000`00000000 ntdll!RtlUserThreadStart+0x21
```

© 2022 Software Diagnostics Services

Examining JIT Code

```
0:000> !IP2MD 0x00007ffa`7cae2534
MethodDesc:   00007ffa7c7f5aa0
Method Name:  LINQPad.UIProgram.Go(System.String[])
Class:        00007ffa7c803a28
MethodTable:  00007ffa7c7f7078
mdToken:      0000000060001A4
Module:       00007ffa7c79f798
IsJitted:     yes
Current CodeAddr:     00007ffa7cae1b90
Version History:
  ILCodeVersion:      0000000000000000
  ReJIT ID:           0
  IL Addr:            0000000000000000
     CodeAddr:           00007ffa7cae1b90  (MinOptJitted)
     NativeCodeVersion:  0000000000000000

0:000> !DumpModule 00007ffa7c79f798
Name: C:\Program Files\LINQPad6\LINQPad.GUI.dll
Attributes:              PEFile SupportsUpdateableMethods
Assembly:                00000177c5cdece0
BaseAddress:             00000177DFDE0000
PEFile:                  00000177C5CDE0E0
ModuleId:                00007FFA7C7D1860
ModuleIndex:             0000000000000001
LoaderHeap:              0000000000000000
TypeDefToMethodTableMap: 00007FFA7C7A0020
TypeRefToMethodTableMap: 00007FFA7C7A1860
MethodDefToDescMap:      00007FFA7C7A4138
FieldDefToDescMap:       00007FFA7C7AF300
MemberRefToDescMap:      0000000000000000
FileReferencesMap:       00007FFA7C7B7F78
AssemblyReferencesMap:   00007FFA7C7B7F80
MetaData start address:  00000177DFE887DC (719936 bytes)
```

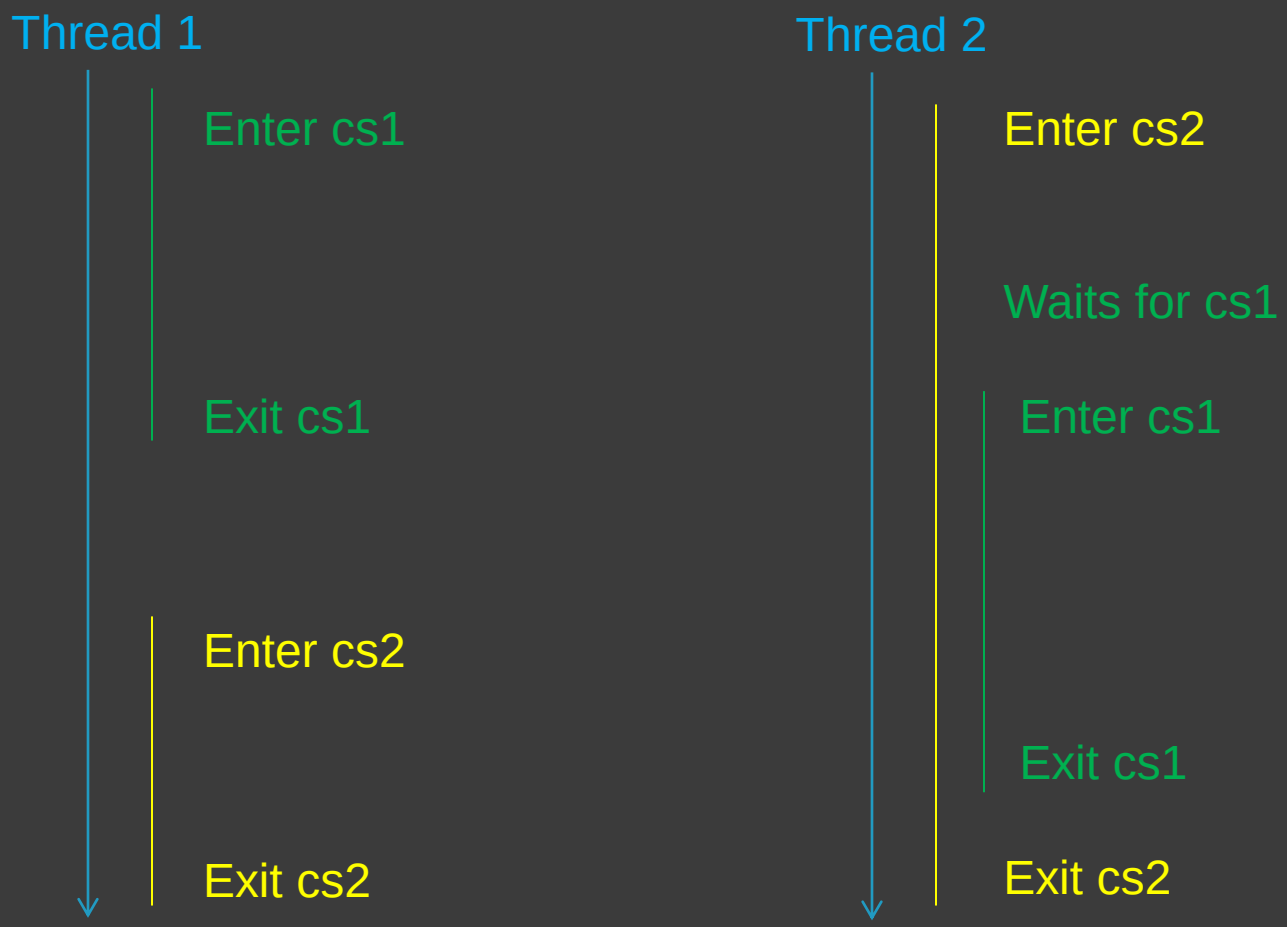
© 2022 Software Diagnostics Services

Exercise MD11

- **Goal:** Learn how to find problem modules, classes and methods, disassemble code, dump object references, recognize and analyze deadlocks
- **Elementary Diagnostics Patterns:** Crash; Hang
- **Memory Analysis Patterns:** Exception Stack Trace; NULL Pointer (Data); CLR Thread; JIT Code; Deadlock (Managed Space); Execution Residue (Managed Space); Hidden Exception (Managed Space)
- **Debugging Implementation Patterns:** Break-in
- \AWD4\Exercise-MD11.pdf

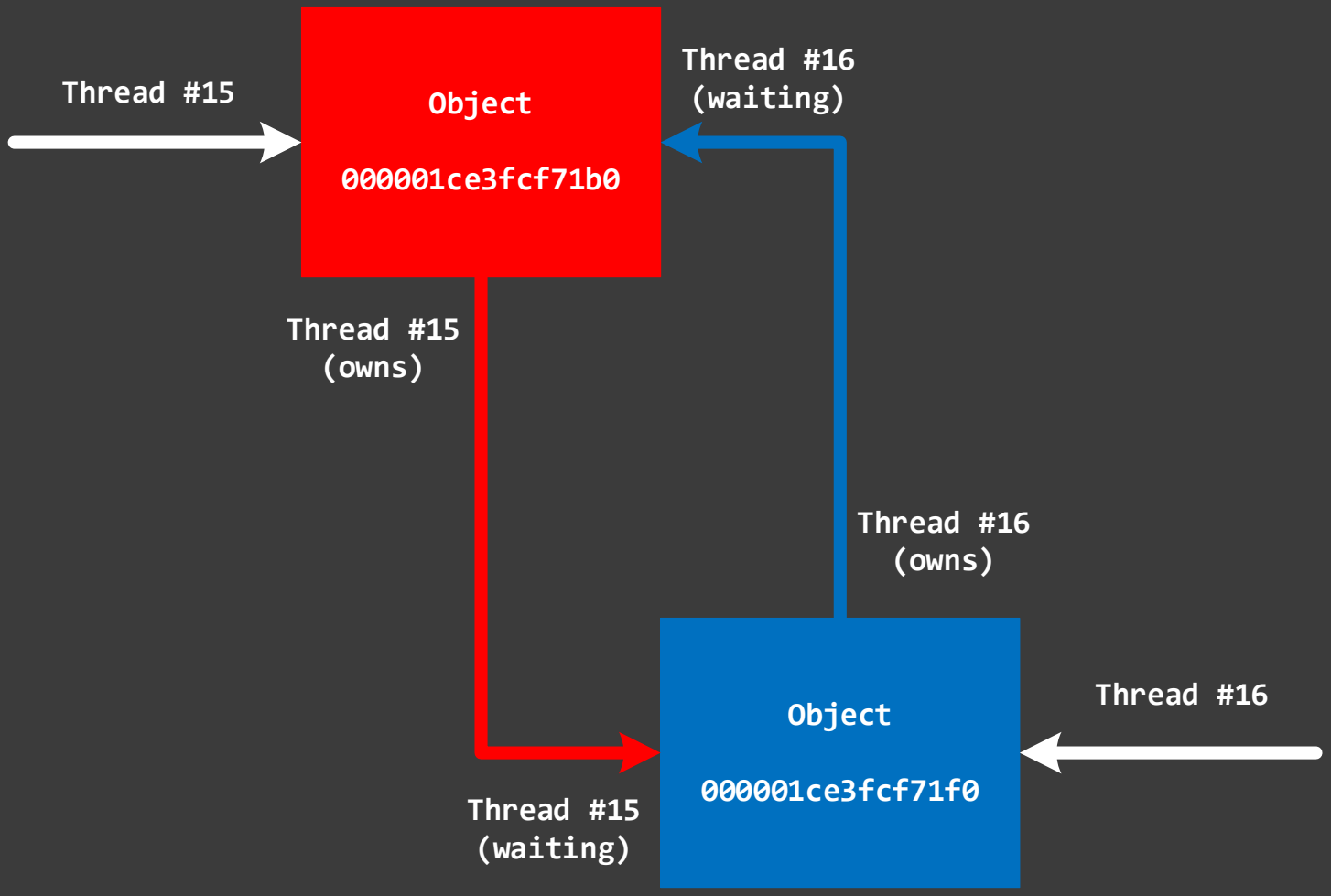
© 2022 Software Diagnostics Services

Expected Behavior



© 2022 Software Diagnostics Services

Deadlock



© 2022 Software Diagnostics Services

Exercise TD5

- **Goal:** Learn how to find hidden exceptions using Time Travel Debugging
- **Elementary Diagnostics Patterns:** Hang
- **Memory Analysis Patterns:** Hidden Exception (User Space)
- **Debugging Implementation Patterns:** Instruction Trace
- \AWD4\Exercise-TD5.pdf

© 2022 Software Diagnostics Services

Postmortem Debugging

- Memory dump collection
  - **.dump** / **.dumpcab** WinDbg commands
  - External methods
- Training
  - Accelerated Windows Memory Dump Analysis
  - Advanced Windows Memory Dump Analysis
  - Accelerated .NET Core Memory Dump Analysis

© 2022 Software Diagnostics Services

Analysis Pattern Links

Spiking Thread
Debugger Bug
Hidden Exception (User Space)
Execution Residue (Managed Space)
Managed Stack Trace
NULL Pointer (Code)
Stack Overflow (User Mode)
Deadlock (Objects, User Space)
Message Box
String Parameter
Handle Leak
Abnormal Value
Module Variable
JIT Code
Unloaded Module
Main Thread
Near Exception
CLR Thread
Exception Stack Trace
Handled Exception (User Space)
Managed Code Exception
NULL Pointer (Data)
Dynamic Memory Corruption (Process Heap)
Rough Stack Trace (Unmanaged Space)
Virtualized Process (WOW64)
Memory Leak (Process Heap)
Wait Chain (Mutex Objects)
Historical Information
Counter Value
Deadlock (Managed Space)
Hidden Exception (Managed Space)
Corrupt Structure
Execution Residue (Unmanaged Space, User)

© 2022 Software Diagnostics Services

Resources

- WinDbg Help / WinDbg.org (quick links)
- DumpAnalysis.org / SoftwareDiagnostics.Institute / PatternDiagnostics.com
- Debugging.TV / YouTube.com/DebuggingTV / YouTube.com/PatternDiagnostics
- Practical Foundations of Windows Debugging, Disassembling, Reversing, Second Edition
- Windows Debugging Notebook: Essential User Space WinDbg Commands
- Software Diagnostics Library
- Pattern-Driven Software Problem Solving
- Memory Dump Analysis Anthology (Diagnomicon)

© 2022 Software Diagnostics Services

Q&A

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

© 2022 Software Diagnostics Services

Thank you for attendance!

© 2022 Software Diagnostics Services