



# Windows Memory Dump Analysis

## Accelerated

**Version 5.5**

**Part 1: Process User Space**

Dmitry Vostokov  
Software Diagnostics Services

# Prerequisites

## WinDbg Commands

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

## Basic Windows troubleshooting

# Training Goals

- Part 1A: Review fundamentals
- Part 1B: Learn how to analyze process dumps
- Part 2A: Review fundamentals
- Part 2B: Learn how to analyze kernel dumps
- Part 2C: Learn how to analyze complete (physical memory) dumps
- Part 2D: Learn how to analyze minidumps

# Training Principles

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

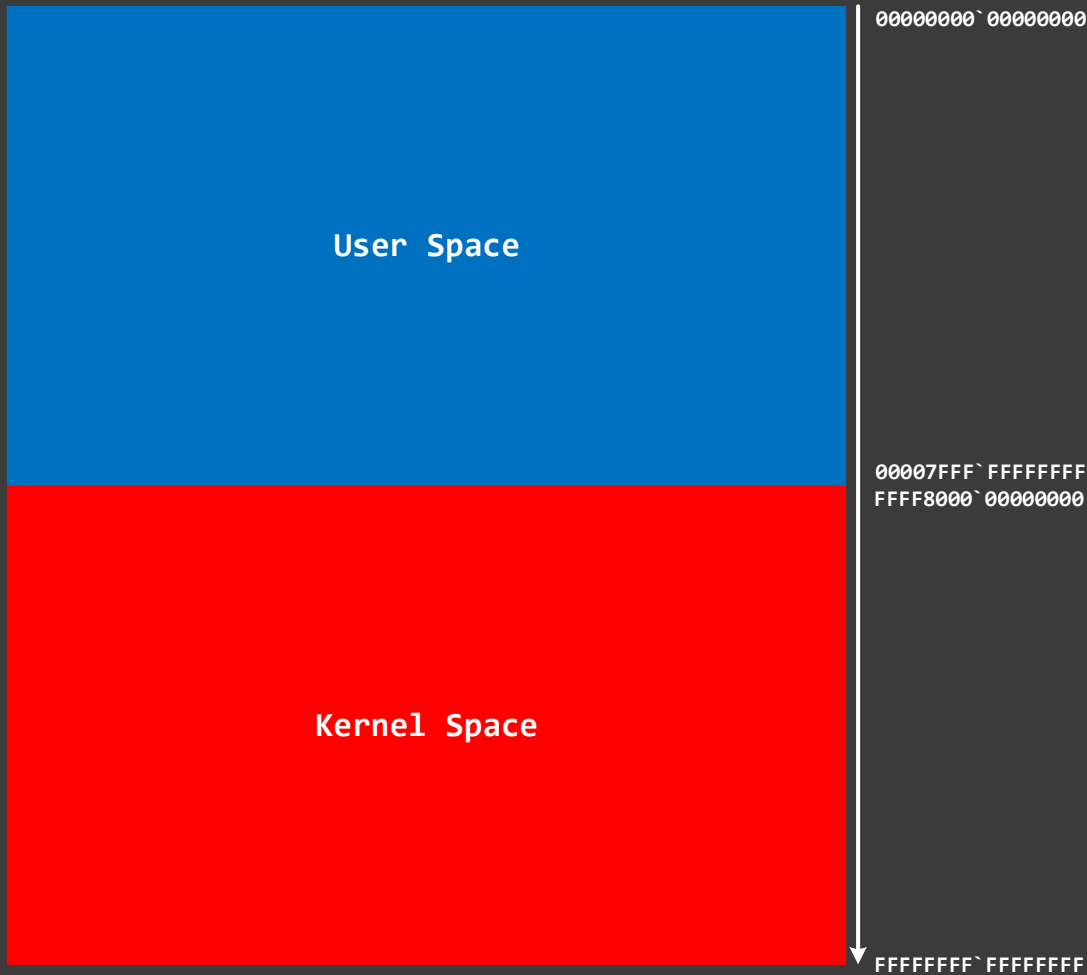
# Coverage (Part 1)

- ⦿ Windows 10 and 11 x64
- ⦿ Both x64 and x86 code, WOW64
- ⦿ Preliminary .NET Core analysis
- ⦿ Process memory dumps
- ⦿ Crashes, hangs, memory and handle leaks, CPU spikes

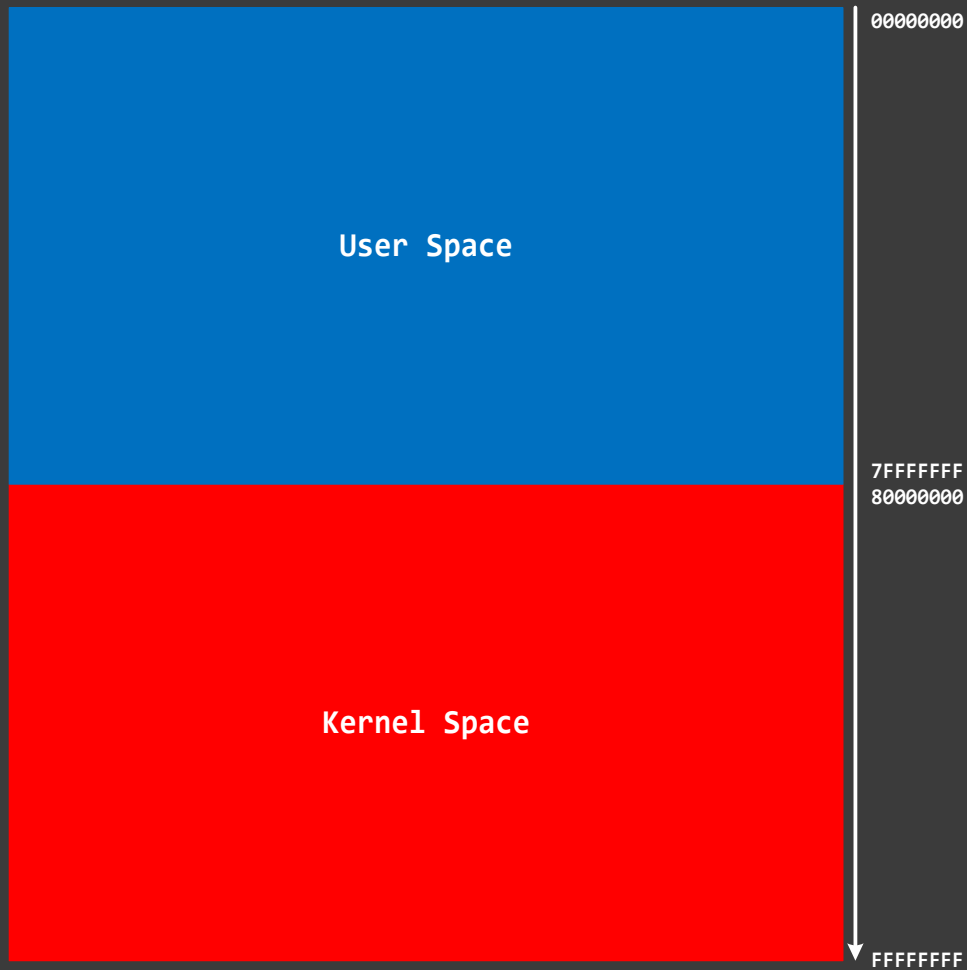
Most of the **exercises** are focused on **x64** code. For their x86 equivalents from older Windows versions, please refer to the previous edition of this course.

# Part 1A: Fundamentals

# Process Space (x64)

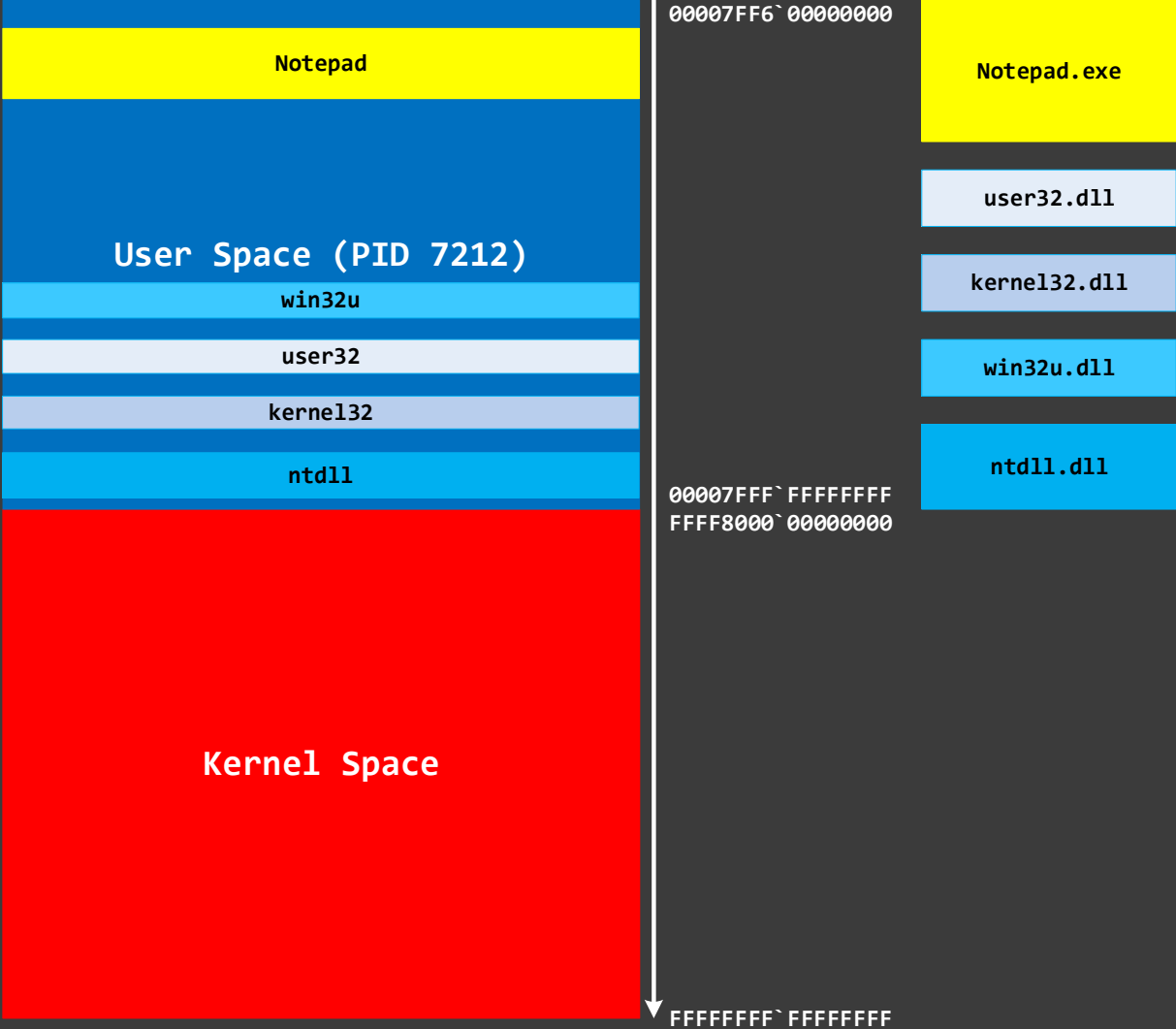


# Process Space (x86)

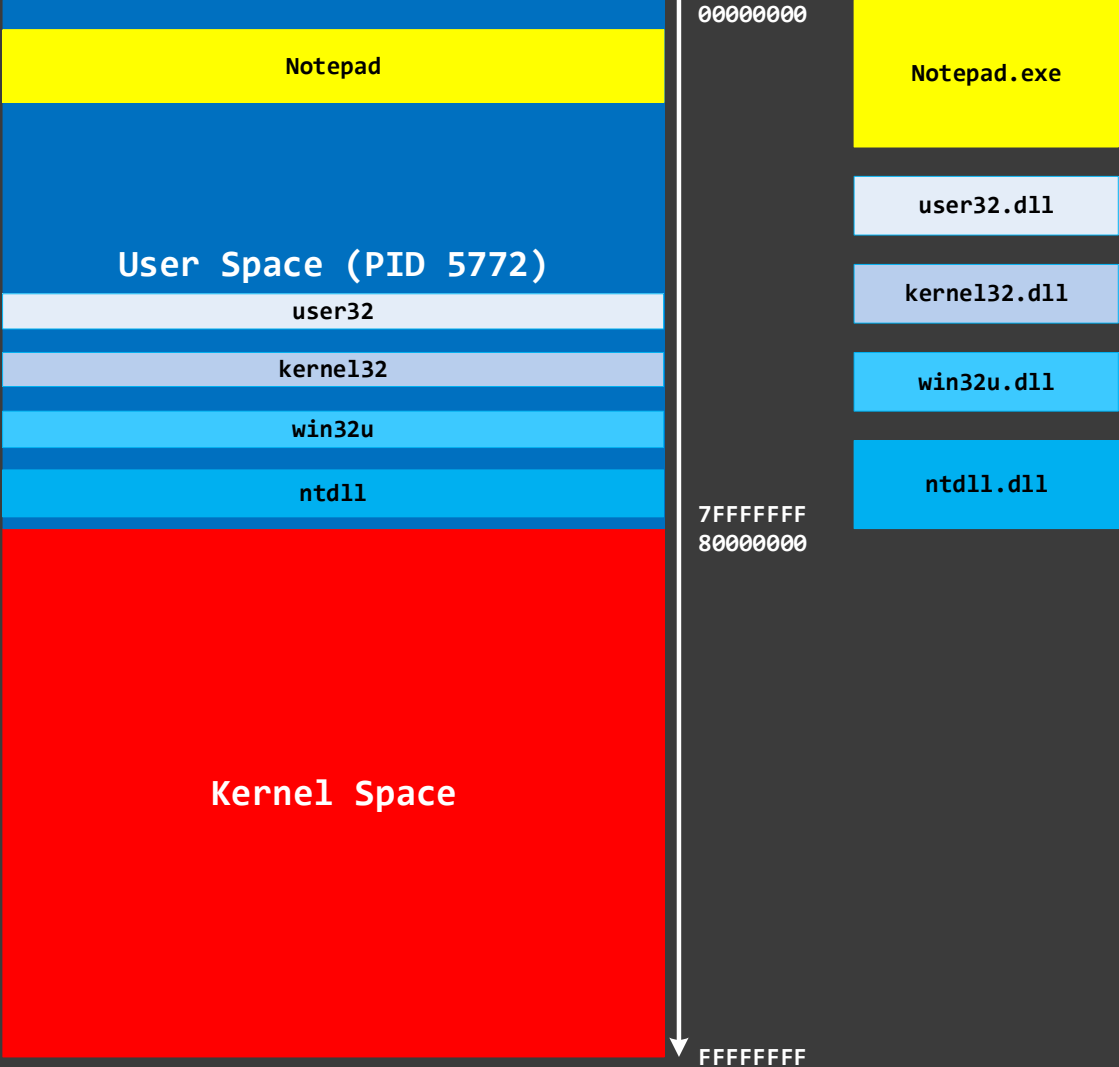




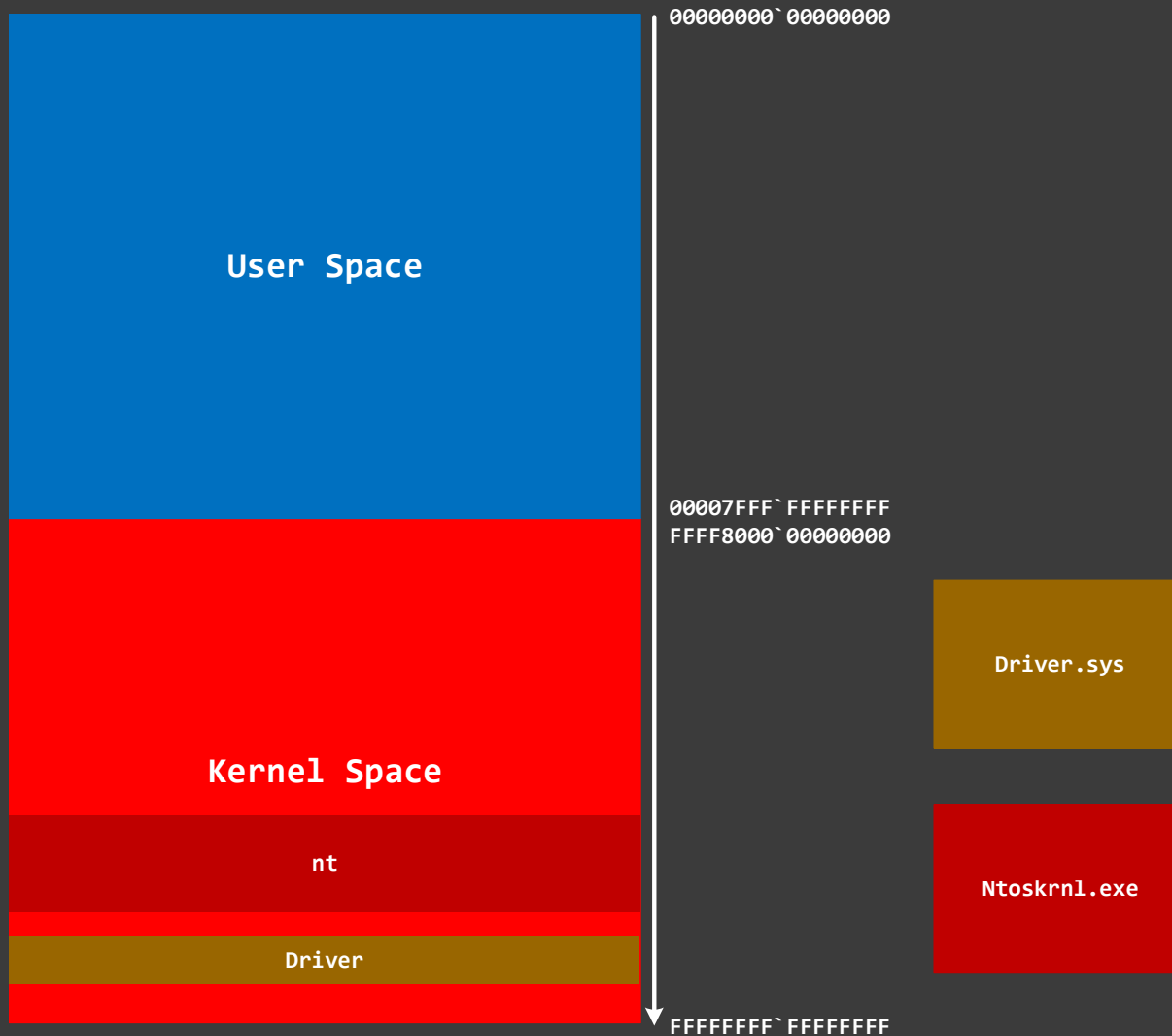
# Application/Process/Module (x64)



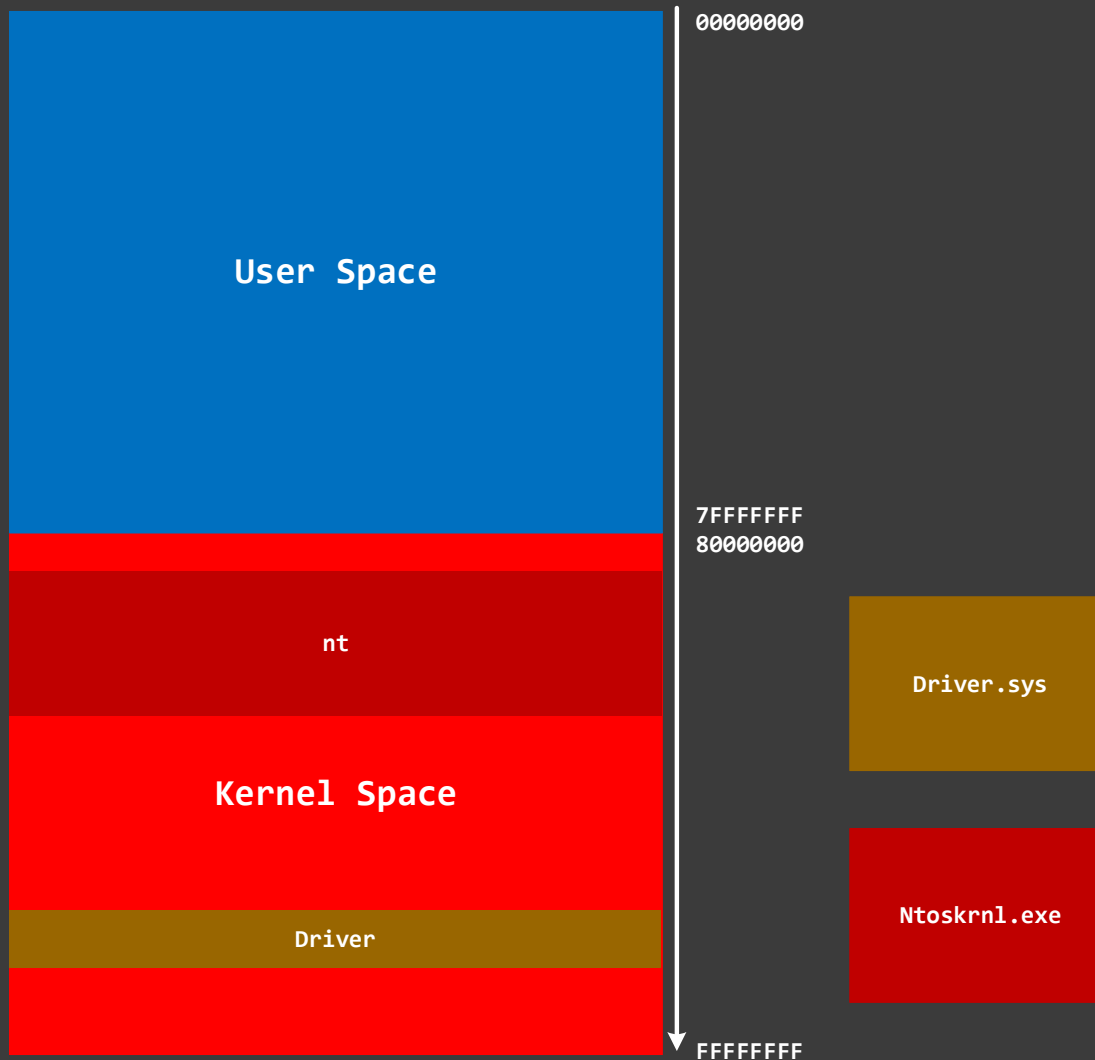
# Application/Process/Module (x86)



# OS Kernel/Driver/Module (x64)



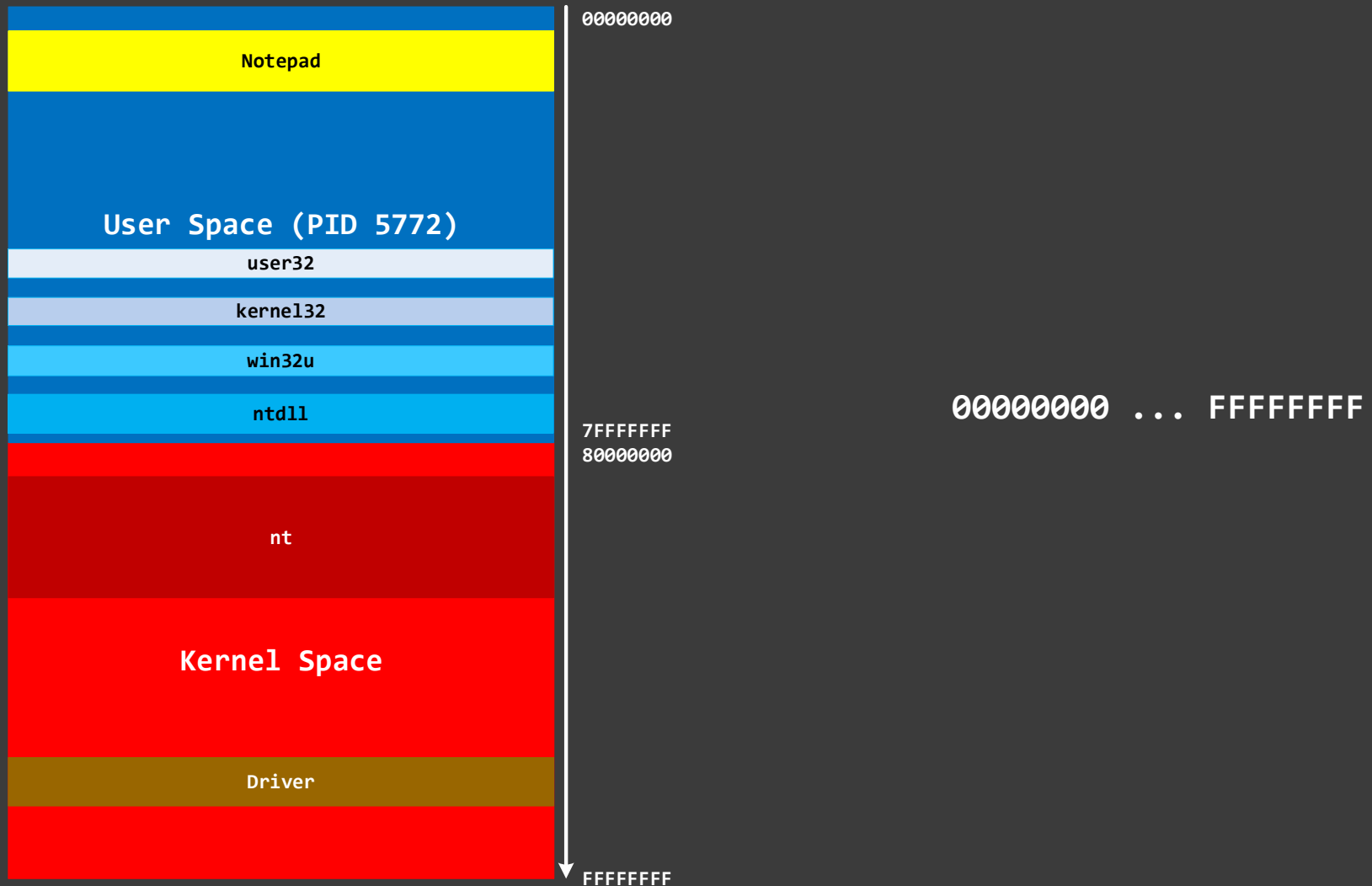
# OS Kernel/Driver/Module (x86)



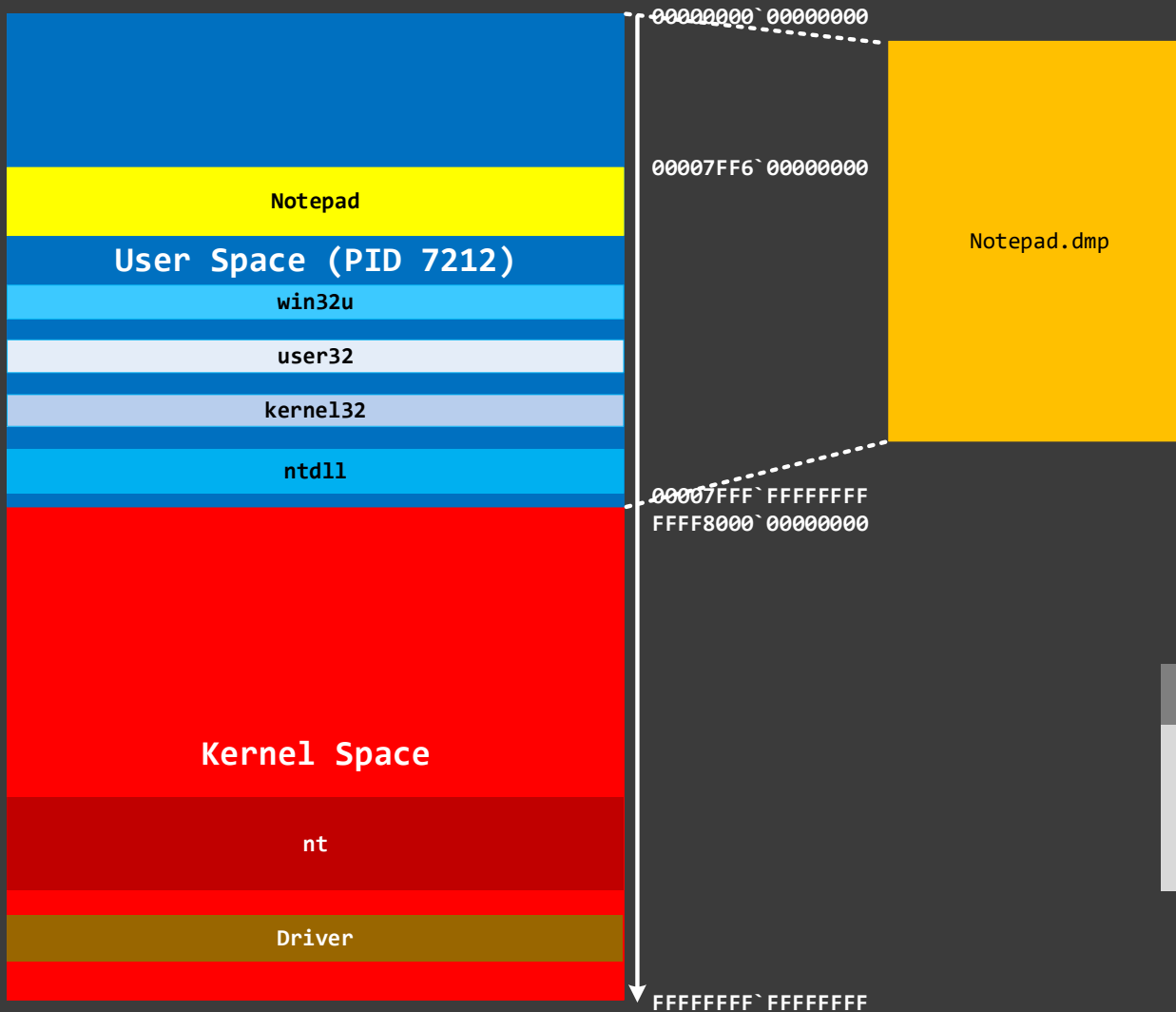
# Process Virtual Space (x64)



# Process Virtual Space (x86)



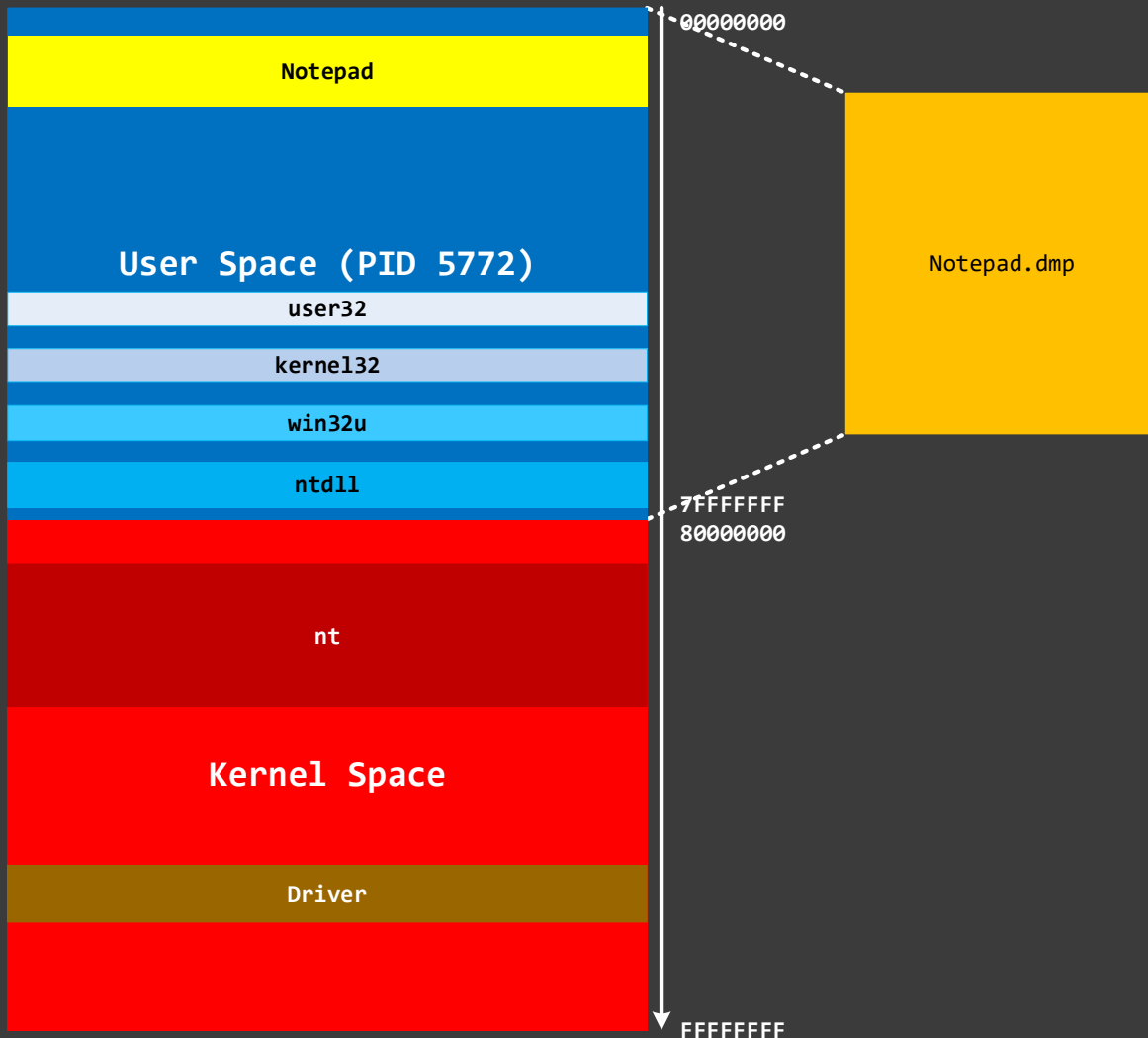
# Process Memory Dump (x64)



## WinDbg Commands

**!mv** command lists modules and their description

# Process Memory Dump (x86)

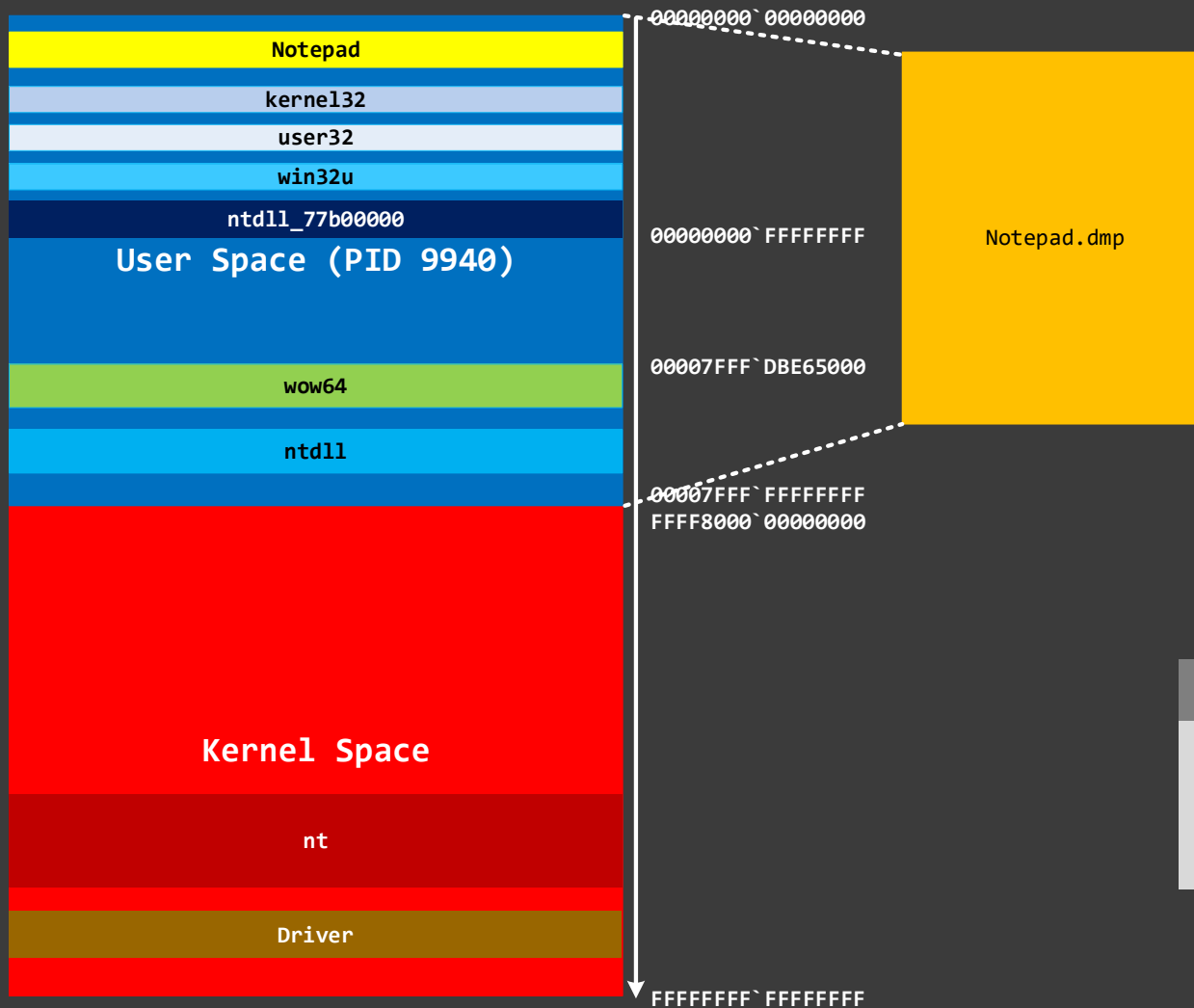


## WinDbg Commands

**!mv** command lists modules and their description



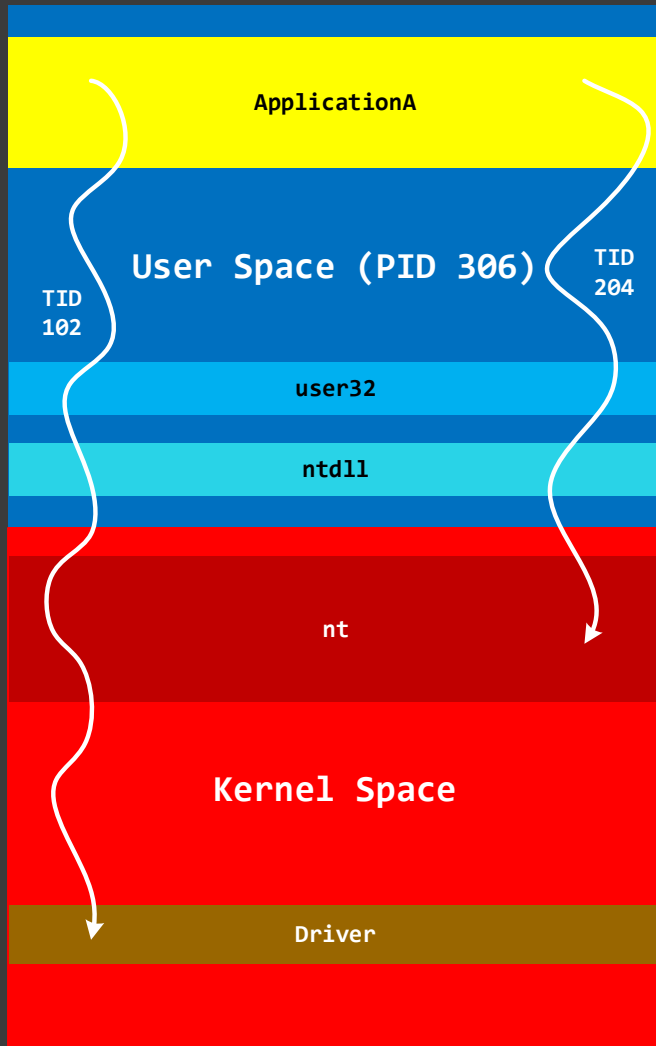
# Process Memory Dump (WOW64)



## WinDbg Commands

**!mv** command lists modules and their description

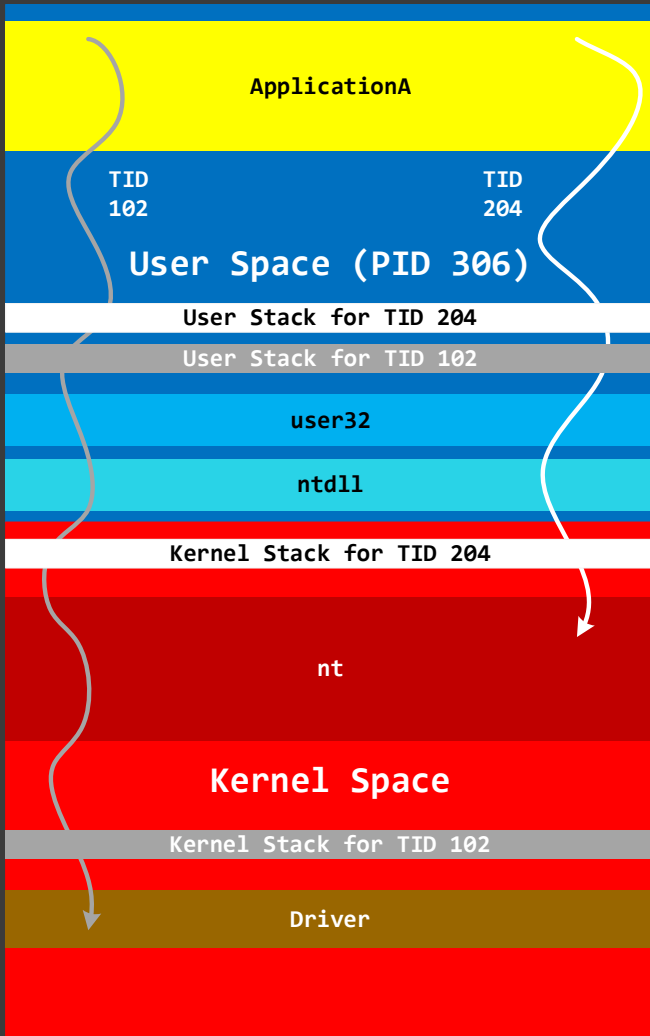
# Process Threads



## WinDbg Commands

Process dumps:  
~<n>s switches between threads

# Thread Stack Raw Data



## WinDbg Commands

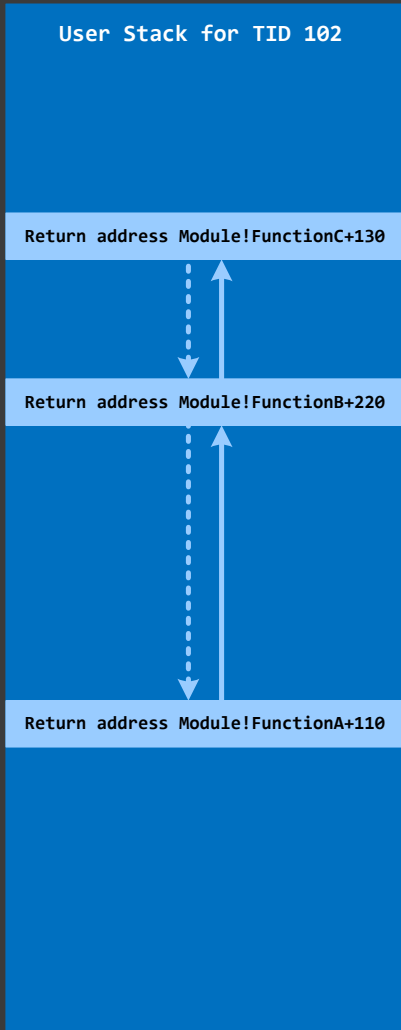
Process dumps:

**!teb**

Data:

**dc / dps / dpp / dpa / dpu**

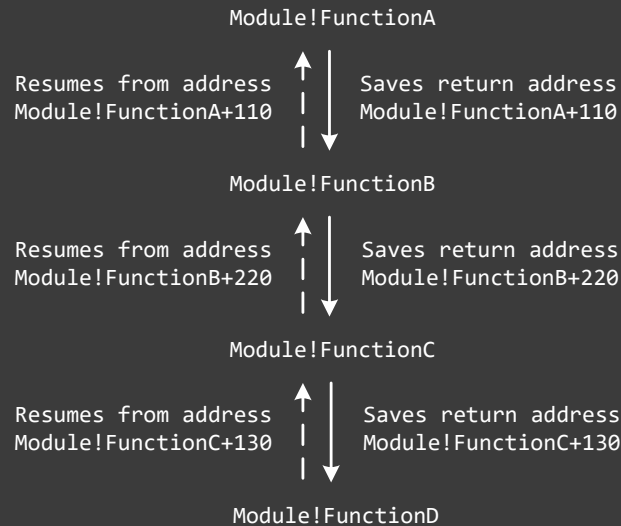
# Thread Stack Trace



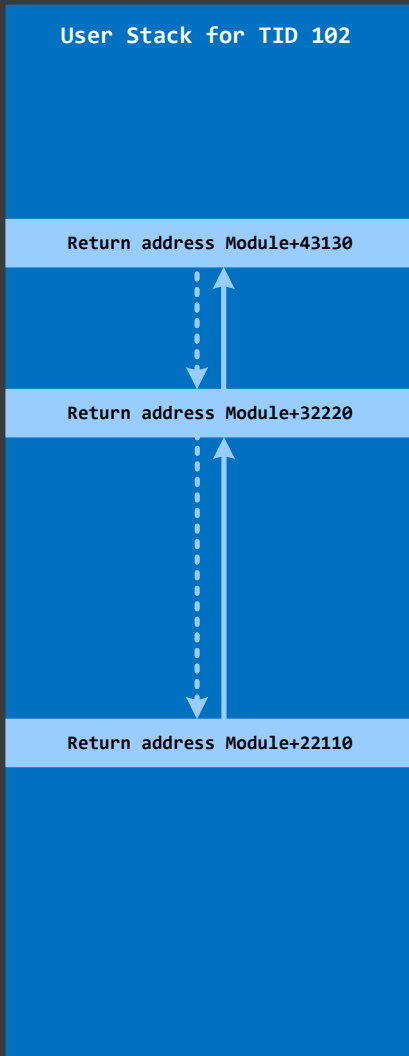
```
FunctionA()  
{  
  ...  
  FunctionB();  
  ...  
}  
FunctionB()  
{  
  ...  
  FunctionC();  
  ...  
}  
FunctionC()  
{  
  ...  
  FunctionD();  
  ...  
}
```

### WinDbg Commands

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



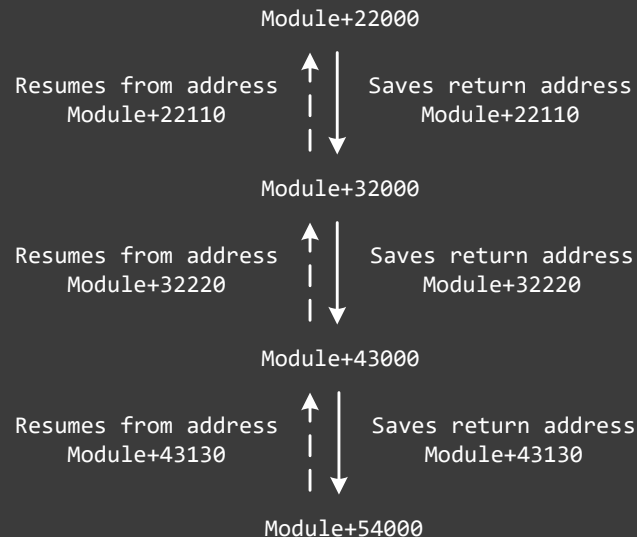
# Thread Stack Trace (no PDB)



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

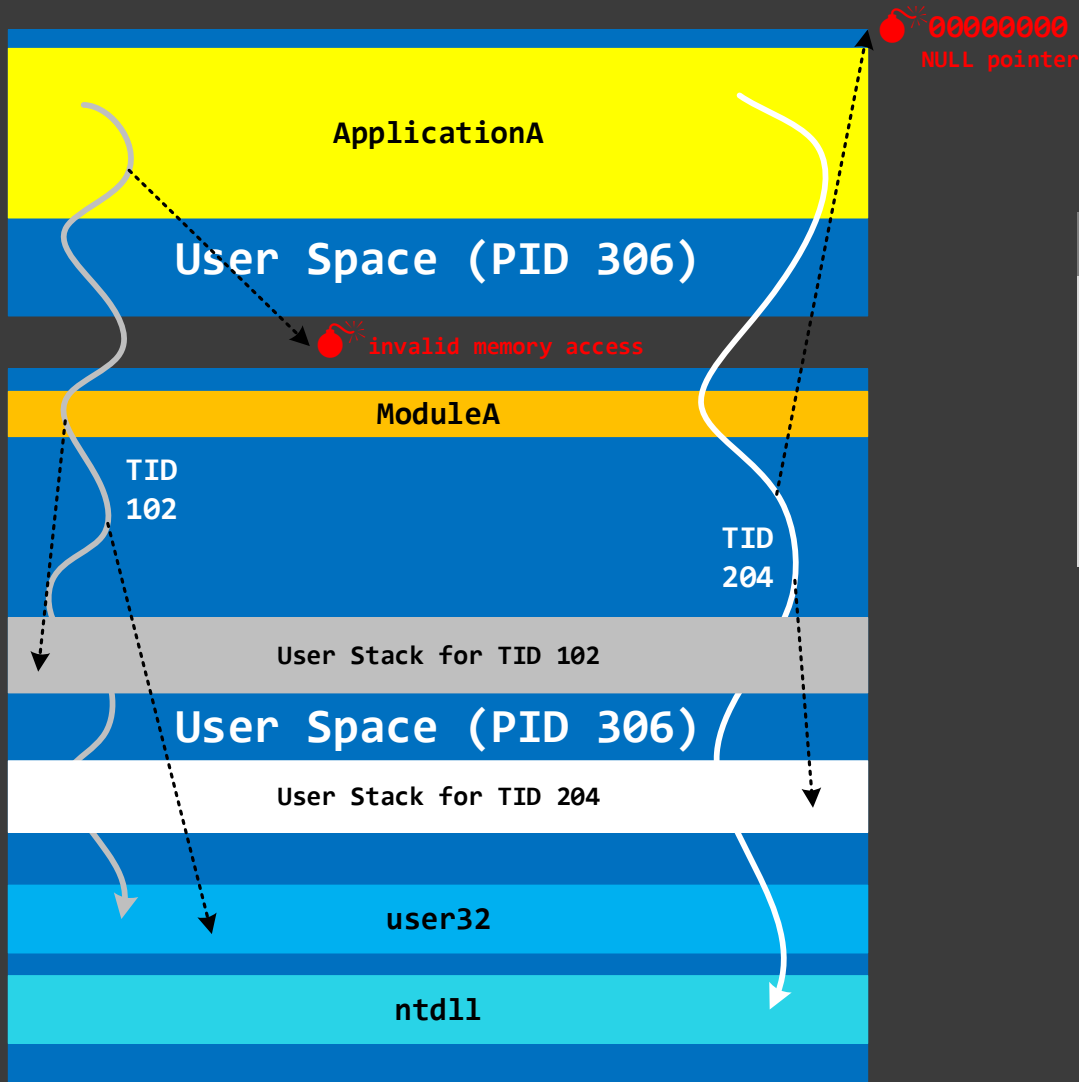
```
Symbol file Module.pdb  
  
FunctionA 22000 - 23000  
FunctionB 32000 - 33000  
FunctionC 43000 - 44000  
FunctionD 54000 - 55000
```

No symbols for Module



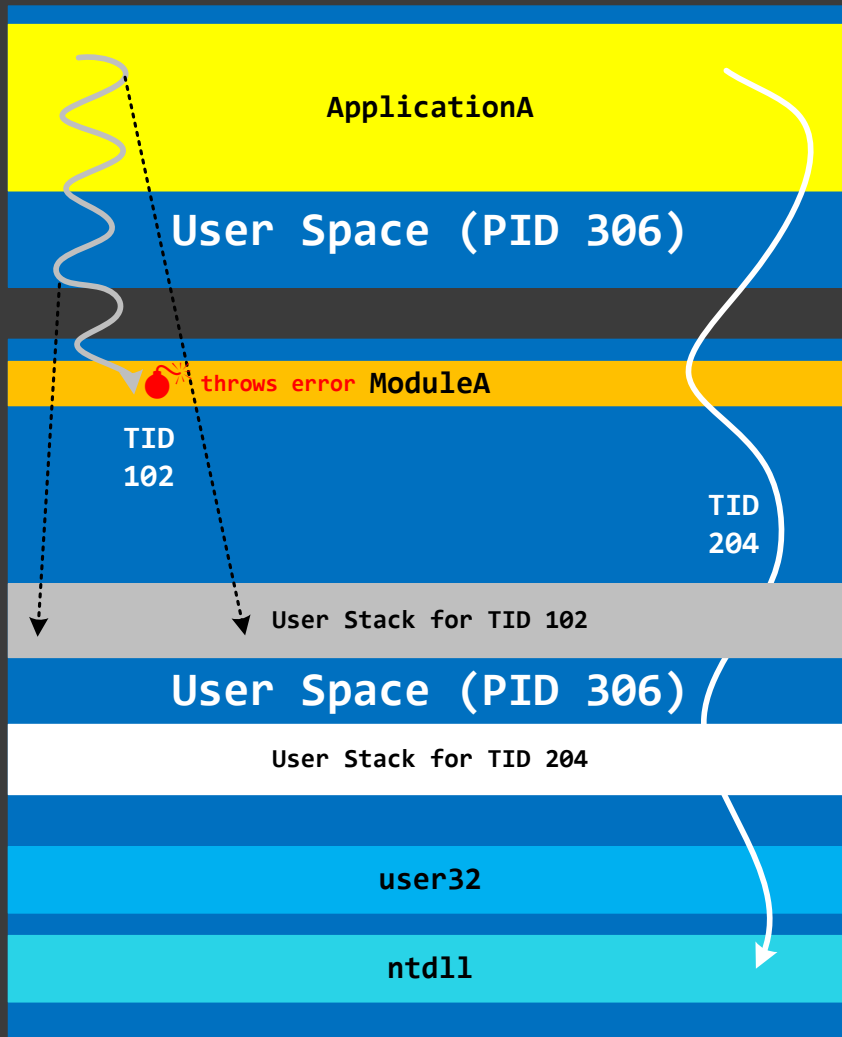
```
WinDbg Commands  
  
0:000> k  
Module+0  
Module+43130  
Module+32220  
Module+22110
```

# Exceptions (Access Violation)



```
WinDbg Commands  
  
address=?????????  
  
Set exception context  
(process dump):  
.cxr
```

# Exceptions (Runtime)



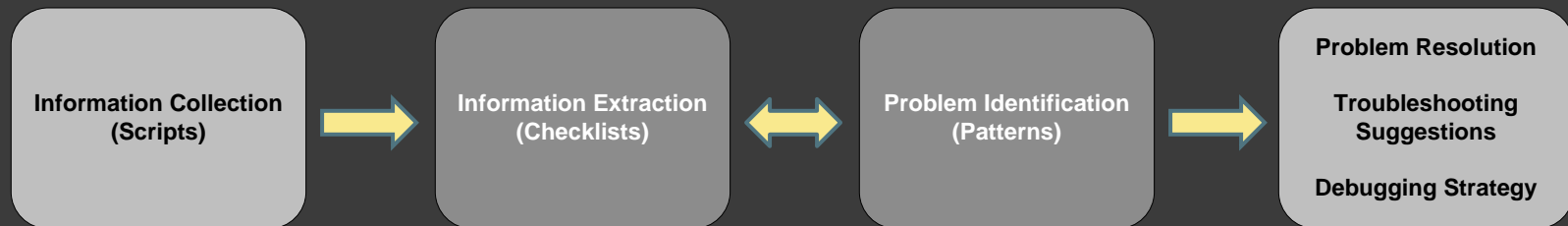
# Pattern-Oriented Diagnostic Analysis

**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>

**Patterns:** <http://www.dumpanalysis.org/blog/index.php/crash-dump-analysis-patterns/>



# Part 1B: Practice Exercises

# Links

- Memory Dumps:

Links are below on this page

- Exercise Transcripts:

Included in this book

# Exercise 0

- ⦿ **Goal:** Install Debugging Tools for Windows or WinDbg Preview, or pull Docker image, and check that symbols are set up correctly
- ⦿ **Patterns:** Incorrect Stack Trace
- ⦿ [\AWMDA-Dumps\Exercise-0-Download-Setup-WinDbg.pdf](#)

# Process Memory Dumps

Exercises P1 – P19

# Exercise P1

- ◎ **Goal:** Learn how to see dump file type and version, get a stack trace, check its correctness, perform default analysis, list threads and modules, check module version information, dump module data, check process environment
- ◎ **Patterns:** Manual Dump; Stack Trace; Not My Version; Environment Hint; Unknown Component
- ◎ [\AWMDA-Dumps\Exercise-P1-Analysis-normal-process-dump-notepad-64.pdf](#)

# Exercise P2

- ◎ **Goal:** Repeat exercise P1 using 32-bit notepad process memory dump
- ◎ [\AWMDA-Dumps\Exercise-P2-Analysis-normal-process-dump-notepad-32.pdf](#)

# Exercise P3

- ◎ **Goal:** Learn how to list stack traces, check their correctness, perform default analysis, list modules, check their version information, check thread age and CPU consumption
- ◎ **Patterns:** Stack Trace Collection
- ◎ [\AWMDA-Dumps\Exercise-P3-Analysis-normal-process-dump-MicrosoftEdge-64.pdf](#)

# Exercise P4

- ◎ **Goal:** Learn to recognize exceptions in process memory dumps and get their context
- ◎ **Patterns:** Exception Stack Trace; Multiple Exceptions; NULL Pointer (Data)
- ◎ [\AWMDA-Dumps\Exercise-P4-Analysis-process-dump-AppK-64-no-symbols.pdf](#)



# Exercise P5

- ◎ **Goal:** Learn how to load application symbols
- ◎ [\AWMDA-Dumps\Exercise-P5-Analysis-process-dump-AppK-64-with-symbols.pdf](#)

# Exercise P6

- ◎ **Goal:** Learn how to recognize heap corruption, dump contents of memory, follow critical section wait chains, check error and status codes
- ◎ **Patterns:** Dynamic Memory Corruption (Process Heap); Wait Chain (Critical Sections)
- ◎ [\AWMDA-Dumps\Exercise-P6-Analysis-process-dump-AppL-64.pdf](#)

# Exercise P7

- ◎ **Goal:** Learn how to debug heap corruption using page heap
- ◎ **Patterns:** Invalid Pointer; Instrumentation Information
- ◎ [\AWMDA-Dumps\Exercise-P7-Analysis-process-dump-AppL2-64.pdf](#)

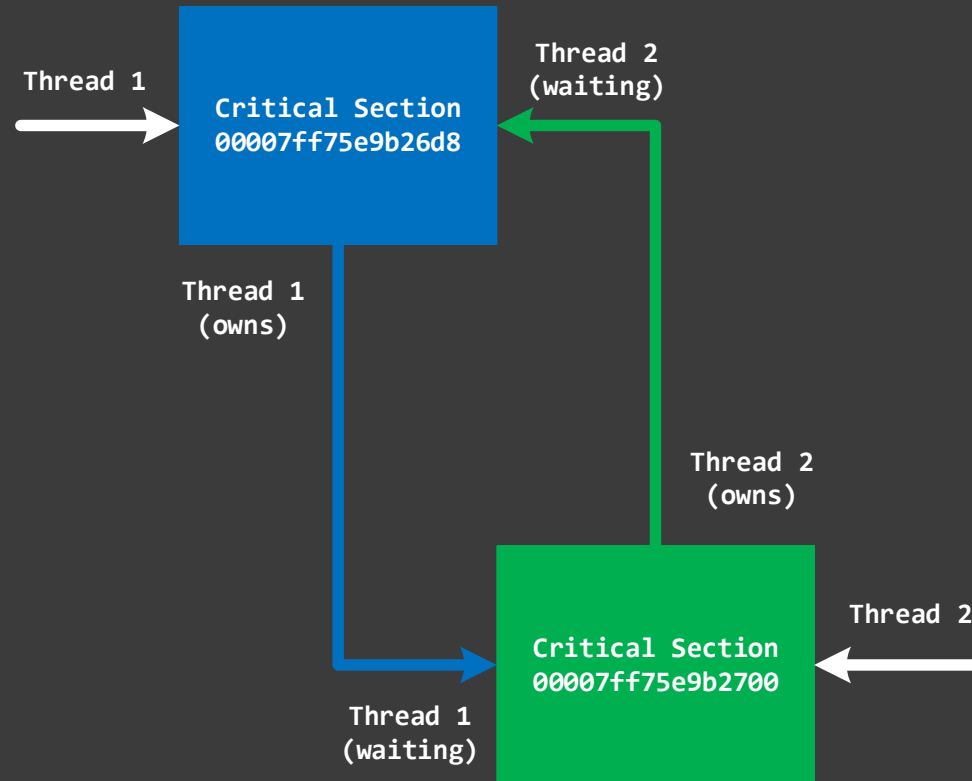
# Exercise P8

- ◎ **Goal:** Learn how to recognize CPU spikes, invalid pointers, disassemble code, and reconstruct stack trace
- ◎ **Patterns:** Wild Code; Active Thread; Spiking Thread; NULL Pointer (Code); Truncated Stack Trace; Stored Exception
- ◎ [\AWMDA-Dumps\Exercise-P8-Analysis-process-dump-AppM-64.pdf](#)

# Exercise P9

- ◎ **Goal:** Learn how to recognize critical section waits and deadlocks, dump raw stack data, and see hidden exceptions
- ◎ **Patterns:** Deadlock (Critical Sections); Hidden Exception
- ◎ [\AWMDA-Dumps\Exercise-P9-Analysis-process-dump-AppN-64.pdf](#)

# Deadlock



# Exercise P10

- ◎ **Goal:** Learn how to recognize application heap problems, buffer and stack overflow patterns, analyze raw stack data
- ◎ **Patterns:** Double Free; Local Buffer Overflow (User Space); Stack Overflow
- ◎ [\AWMDA-Dumps\Exercise-P10-Analysis-process-dump-AppO-64.pdf](#)

# Exercise P11

- ◎ **Goal:** Learn how to analyze exception patterns, raw stacks, and execution residue
- ◎ **Patterns:** Divide by Zero; C++ Exception; Execution Residue
- ◎ [\AWMDA-Dumps\Exercise-P11-Analysis-process-dump-AppP-64.pdf](#)



# Exercise P12

- ◎ **Goal:** Learn how to analyze managed space
- ◎ **Patterns:** Platform-Specific Debugger; CLR Thread; JIT Code (.NET); Managed Code Exception; Managed Stack Trace
- ◎ [\AWMDA-Dumps\Exercise-P12-Analysis-process-dump-AppR2-64.pdf](#)

# Exercise P13

- ◎ **Goal:** Learn how to analyze 32-process saved as a 64-bit process memory dump
- ◎ **Patterns:** Virtualized Process; Message Box
- ◎ [\AWMDA-Dumps\Exercise-P13-Analysis-process-dump-AppA-WOW64.pdf](#)

# Exercise P14

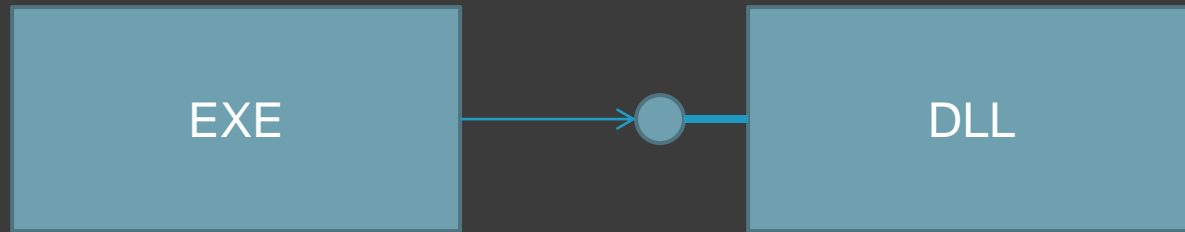
- ◎ **Goal:** Learn how to analyze process memory leaks
- ◎ **Patterns:** Thread Age; Memory Leak (Process Heap)
- ◎ [\AWMDA-Dumps\Exercise-P14-Analysis-process-dump-AppS-64.pdf](#)

# Parameters and Locals

[Debugging TV Frames episode 0x18](#)

# Symbol Types

- Exported and imported names



- Function and variable names
- Data types

# Exercise P15

- ◎ **Goal:** Learn how to navigate function parameters in cases of reduced symbolic information in 32-bit process memory dumps
- ◎ **Patterns:** Reduced Symbolic Information
- ◎ [\AWMDA-Dumps\Exercise-P15-Analysis-process-dump-notepad-32.pdf](#)

# Exercise P16

- ◎ **Goal:** Learn how to navigate function parameters in x64 process memory dumps
- ◎ **Patterns:** False Function Parameters; Injected Symbols
- ◎ [\AWMDA-Dumps\Exercise-P16-Analysis-process-dump-notepad-64.pdf](#)

# Exercise P17

- ◎ **Goal:** Learn how to navigate object wait chains in 32-bit memory dumps saved with ProcDump
- ◎ **Patterns:** Embedded Comments; Wait Chain (General); No Data Types; Deadlock (Mixed Objects, User Space)
- ◎ [\AWMDA-Dumps\Exercise-P17-Analysis-process-dump-AppQ-32.pdf](#)



# Exercise P18

- ◎ **Goal:** Learn how to navigate object wait chains in 64-bit memory dumps saved with ProcDump
- ◎ **Patterns:** Not My Thread; Blocked Thread (Software); Main Thread; Passive Thread (User Space); Coincidental Symbolic Information
- ◎ [\AWMDA-Dumps\Exercise-P18-Analysis-process-dump-AppQ-64.pdf](#)

# Exercise P19

- ◎ **Goal:** Learn how to analyze process handle leaks
- ◎ **Patterns:** Active Space; Handle Leak
- ◎ [\AWMDA-Dumps\Exercise-P19-Analysis-process-dump-AppT-64.pdf](#)

# Pattern Links

[Spiking Thread](#)

[C++ Exception](#)

[Divide by Zero](#)

[Dynamic Memory Corruption \(Process Heap\)](#)

[Execution Residue](#)

[Invalid Pointer](#)

[Manual Dump](#)

[Managed Stack Trace](#)

[Not My Version](#)

[NULL Pointer \(Code\)](#)

[Stack Trace Collection](#)

[Environment Hint](#)

[Unknown Component](#)

[Virtualized Process](#)

[False Function Parameters](#)

[Reduced Symbolic Information](#)

[Stored Exception](#)

[Instrumentation Information](#)

[JIT Code \(.NET\)](#)

[Embedded Comment](#)

[Deadlock \(Mixed Object, User Space\)](#)

[Blocked Thread \(Software\)](#)

[Passive Thread \(User Space\)](#)

[Active Space](#)

[Memory Leak \(Process Heap\)](#)

[CLR Thread](#)

[Deadlock \(Critical Sections\)](#)

[Double Free](#)

[Exception Stack Trace](#)

[Hidden Exception](#)

[Local Buffer Overflow \(User Space\)](#)

[Managed Code Exception](#)

[Multiple Exceptions](#)

[NULL Pointer \(Data\)](#)

[Stack Trace](#)

[Stack Overflow](#)

[Wild Code](#)

[Wait Chain \(Critical Sections\)](#)

[Message Box](#)

[Injected Symbols](#)

[Truncated Stack Trace](#)

[Incorrect Stack Trace](#)

[Active Thread](#)

[Thread Age](#)

[Wait Chain \(General\)](#)

[Not My Thread](#)

[Main Thread](#)

[Coincidental Symbolic Information](#)

[Handle Leak](#)

[Platform-Specific Debugger](#)

# Pattern Classification

[Space/Mode](#)

[Hooksware](#)

[DLL Link Patterns](#)

[Contention Patterns](#)

[Stack Trace Patterns](#)

[Exception Patterns](#)

[Module Patterns](#)

[Thread Patterns](#)

[Dynamic Memory Corruption Patterns](#)

[.NET / CLR / Managed Space Patterns](#)

[Falsity and Coincidence Patterns](#)

[Hidden Artifact Patterns](#)

[Frame Patterns](#)

[Memory dump type](#)

[Wait Chain Patterns](#)

[Insufficient Memory Patterns](#)

[Stack Overflow Patterns](#)

[Symbol Patterns](#)

[Meta-Memory Dump Patterns](#)

[Optimization Patterns](#)

[Process Patterns](#)

[Deadlock and Livelock Patterns](#)

[Executive Resource Patterns](#)

[RPC, LPC and ALPC Patterns](#)

[Pointer Patterns](#)

[CPU Consumption Patterns](#)

# Pattern Case Studies

More than 70 multiple pattern case studies:

<http://www.dumpanalysis.org/blog/index.php/pattern-cooperation/>

**Pattern Interaction** chapters in  
Memory Dump Analysis Anthology

# Additional Resources

- WinDbg Help / [WinDbg.org](https://winDBG.org) (quick links)
- [DumpAnalysis.org](https://DumpAnalysis.org) / [SoftwareDiagnostics.Institute](https://SoftwareDiagnostics.Institute) / [PatternDiagnostics.com](https://PatternDiagnostics.com)
- [Debugging.TV](https://Debugging.TV) / [YouTube.com/DebuggingTV](https://YouTube.com/DebuggingTV) / [YouTube.com/PatternDiagnostics](https://YouTube.com/PatternDiagnostics)
- Windows Internals, 6<sup>th</sup> ed. (Chapter 14. Crash Dump Analysis), 7<sup>th</sup> ed.
- Advanced Windows Debugging
- Inside Windows Debugging
- [Principles of Memory Dump Analysis](#)
- [Windows Debugging Notebook: Essential User Space WinDbg Commands](#)
- [Encyclopedia of Crash Dump Analysis Patterns, 3<sup>rd</sup> edition](#)
- [Memory Dump Analysis Anthology](#)



# Further Training Courses

- [Accelerated Windows Memory Dump Analysis, 5<sup>th</sup> Edition, Part 2](#)
- [Practical Foundations of Windows Debugging, Disassembling, Reversing](#)
- [Advanced Windows Memory Dump Analysis with Data Structures, 3<sup>rd</sup> Edition](#)
- [Accelerated .NET Memory Dump Analysis, 4<sup>th</sup> Edition](#)
- [Accelerated Windows Malware Analysis with Memory Dumps, 2<sup>nd</sup> Edition](#)
- [Accelerated Disassembly, Reconstruction and Reversing, Revised Edition](#)
- [Accelerated Windows Debugging<sup>3</sup>, 2<sup>nd</sup> Edition](#)

# Q&A

Please send your feedback using the contact form on [PatternDiagnostics.com](https://PatternDiagnostics.com)



Thank you for attendance!