



Windows Memory Dump Analysis Advanced

with Data Structures

Version 4.1

Dmitry Vostokov
Software Diagnostics Services

Prerequisites

Basic and intermediate level

Windows memory dump analysis

Training Goals

- Use UML for communication
- Learn fundamentals of device drivers
- Learn specialized analysis techniques and commands in the context of x64 complete memory dumps
- Learn how to navigate data structures such as linked lists and arrays

Training Principles

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

Practice Exercises

Links

- Memory Dumps:

Included in Exercise 0

- Exercise Transcripts:

Included in this book

Exercise 0

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

Complete Memory Dumps

Exercises C1 – C12

Exercise C1A

- ◎ **Goal:** Learn how to get stack traces related to sessions, processes and threads; diagnose different process relationships and thread types
- ◎ **Patterns:** Stack Trace Collection (Unmanaged Space); Passive Thread; Coupled Processes (Weak); Coupled Processes (Strong); Wait Chain (ALPC); Stack Trace Collection (Predicate); Stack Trace Collection (CPUs); Input Thread; Truncated Stack Trace; Memory Data Model
- ◎ [AdvWMDA-Dumps\Exercise-C1A-Stack-Trace-Collection.pdf](#)

Exercise C1B

- ◎ **Goal:** Get stack traces from WOW64 processes and reconstruct them manually.
- ◎ **Patterns:** Virtualized Process; Main Thread; Execution Residue; Past Stack Trace; Glued Stack Trace
- ◎ [\AdvWMDA-Dumps\Exercise-C1B-WOW64.pdf](#)

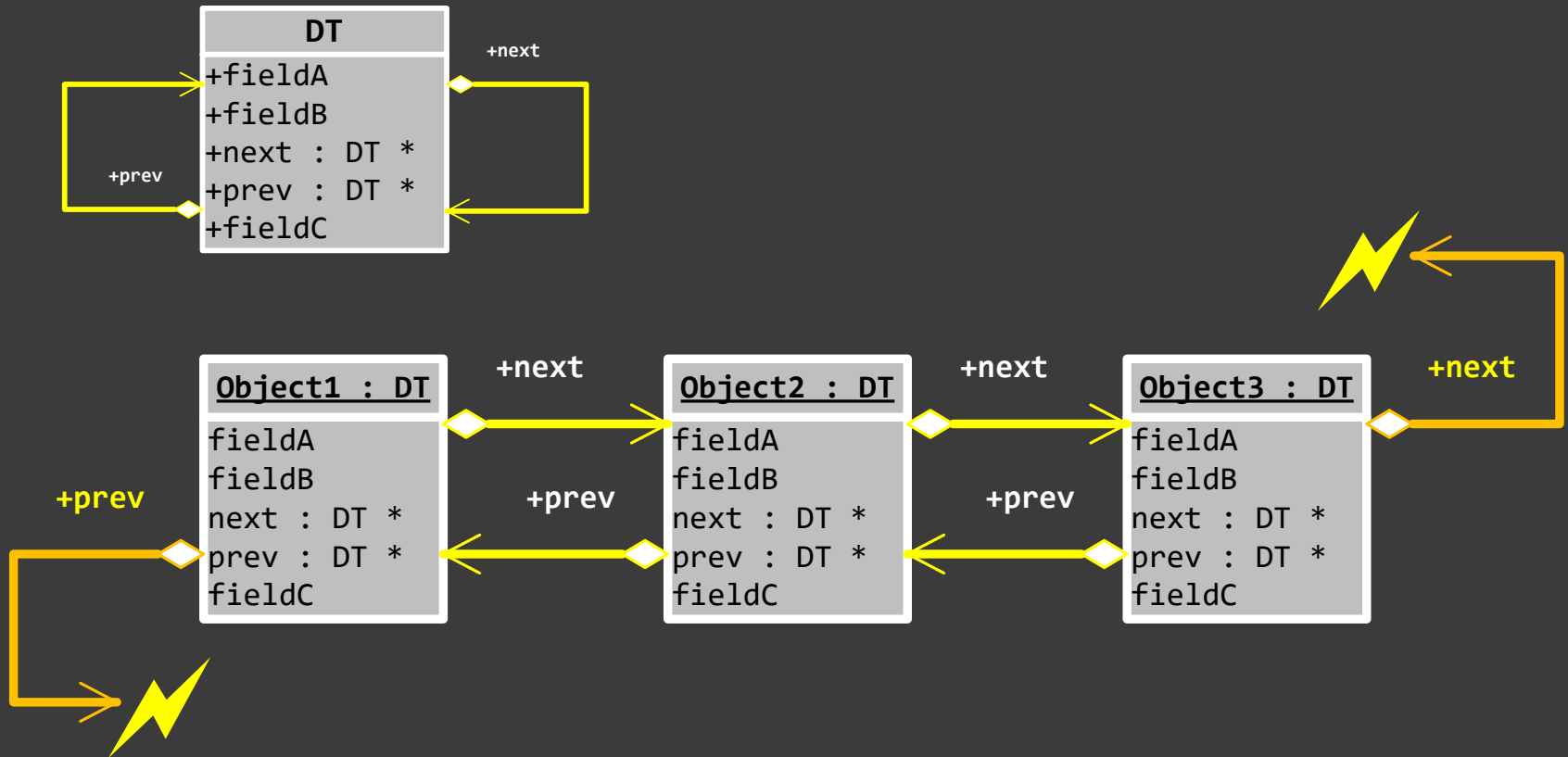
Exercise C2

- ◎ **Goal:** Learn how to assemble code and evaluate expressions; recognize byte ordering conventions; search memory for specific values
- ◎ **Patterns:** Value References
- ◎ [\AdvWMDA-Dumps\Exercise-C2-Memory-Search.pdf](#)

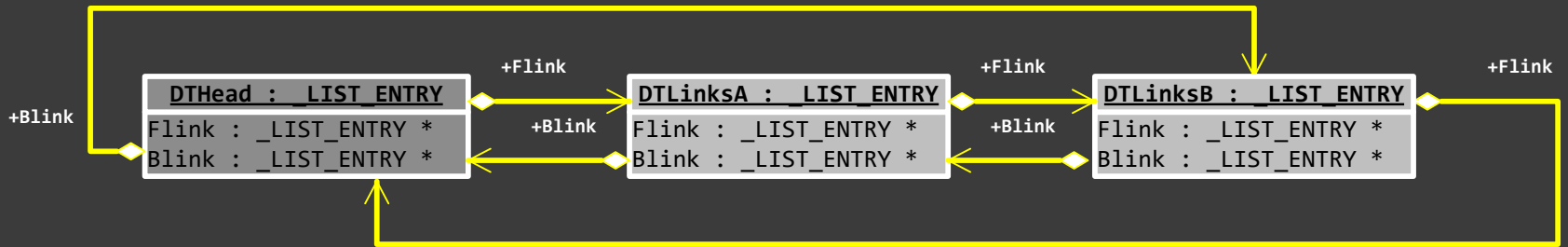
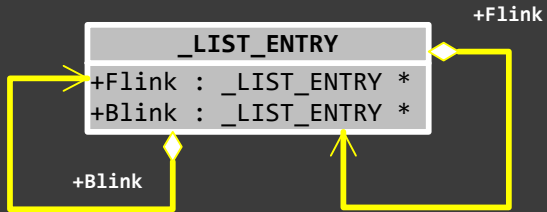
Exercise C3A

- ◎ **Goal:** Learn how to navigate linked lists
- ◎ **Patterns:** Structure Field Collection
- ◎ [\AdvWMDA-Dumps\Exercise-C3A-Linked-Lists.pdf](#)

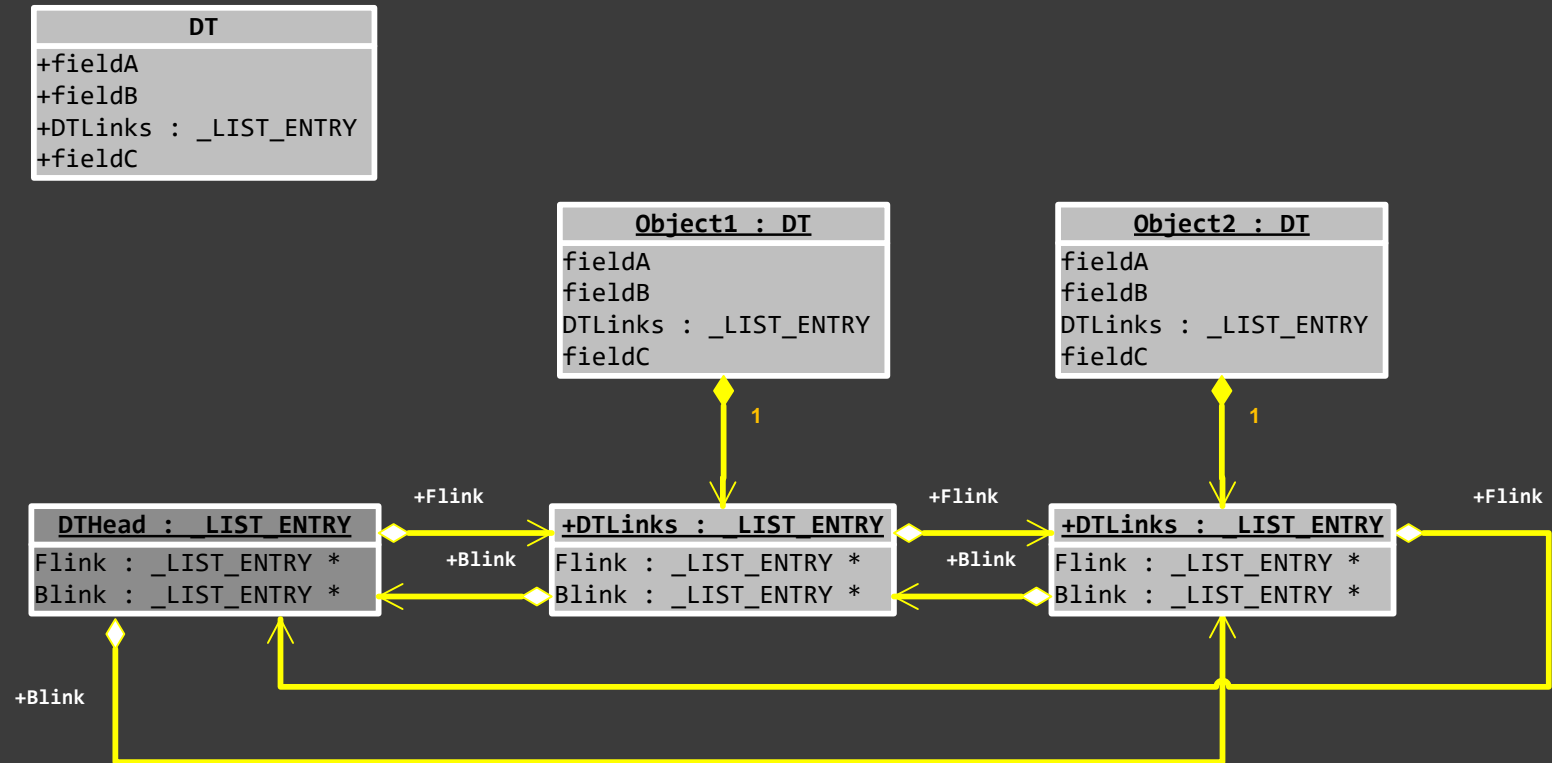
Linked List



_LIST_ENTRY



Linked Data Structures



Exercise C3B

- ◎ **Goal:** Learn how to navigate session processes using linked lists
- ◎ **Patterns:** Debugger Bug
- ◎ [\AdvWMDA-Dumps\Exercise-C3B-Session-Processes.pdf](#)

Exercise C4A

- ◎ **Goal:** Learn how to create scripts to extend WinDbg functionality (via built-in scripting)
- ◎ **Patterns:** Spiking Thread; Thread Waiting Time
- ◎ [\AdvWMDA-Dumps\Exercise-C4A-Scripting.pdf](#)

Exercise C4B

- ◎ **Goal:** Learn how to create scripts to extend WinDbg functionality (via JavaScript scripting)
- ◎ [\AdvWMDA-Dumps\Exercise-C4B-Scripting.pdf](#)

Exercise C5

- ◎ **Goal:** Learn how to inspect registry
- ◎ **Patterns:** Self-Diagnosis
- ◎ [\AdvWMDA-Dumps\Exercise-C5-Registry.pdf](#)

Exercise C6

- ◎ **Goal:** Learn how to inspect module (including system/kernel) variables and check them with extension command output; dump arrays
- ◎ **Patterns:** Module Variable; Regular Data
- ◎ [\AdvWMDA-Dumps\Exercise-C6-ModuleVariables.pdf](#)

Exercise C7

- ◎ **Goal:** Learn how to inspect various system (kernel) objects
- ◎ **Patterns:** System Object
- ◎ [\AdvWMDA-Dumps\Exercise-C7-SystemObjects.pdf](#)

Exercise C8

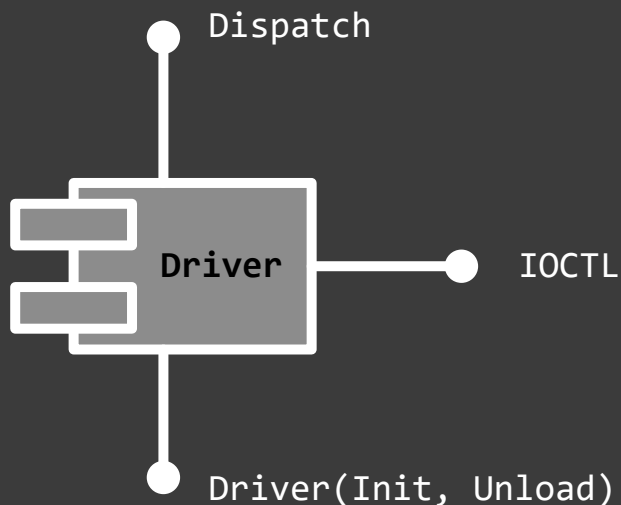
- ◎ **Goal:** Learn how to inspect network protocols and adapters
- ◎ **Patterns:** Disconnected Network Adapter
- ◎ [\AdvWMDA-Dumps\Exercise-C8-Network.pdf](#)

Exercise C9

- ◎ **Goal:** Learn how to inspect IRP, file, device, and driver objects
- ◎ **Patterns:** Stack Trace (I/O Requests); Stack Trace (I/O Devices)
- ◎ [\AdvWMDA-Dumps\Exercise-C9-Device-Drivers.pdf](#)

Device Driver

- A pluggable component for a device or several devices
- Creates device objects and symbolic links to them
- Provides entry points for I/O operations including IOCTL interface (I/O Control - used for any purpose)
- Implemented as a C structure with data and pointers to functions



`\Driver\<Name>`

`\FileSystem\<Name>`

Device Driver Example

```
3: kd> !drvobj \Driver\Beep 3
Driver object (ffffe000ea9309c0) is for:
  \Driver\Beep
Driver Extension List: (id , addr)
```

```
Device Object list:
ffffe000eac2c990
```

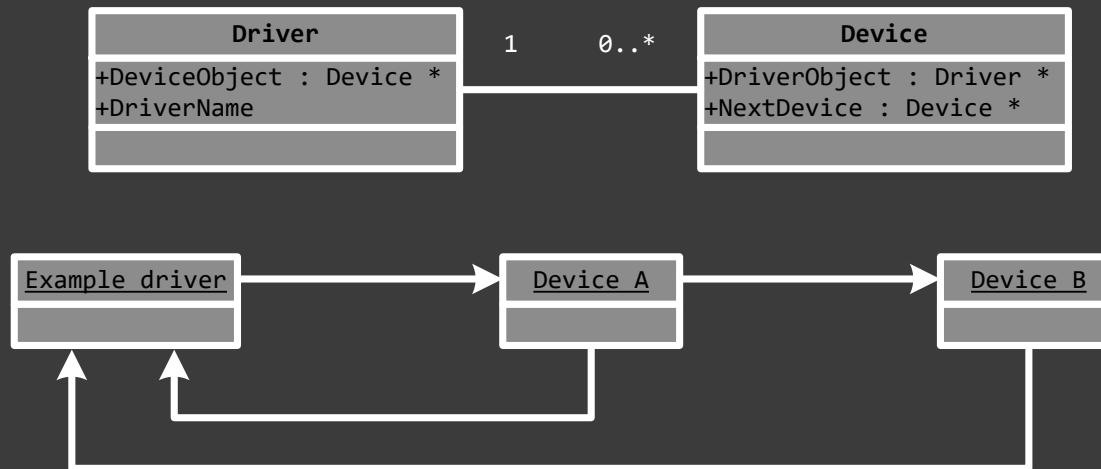
```
DriverEntry: fffff801c02e6000      Beep!GsDriverEntry
DriverStartIo: fffff801c02e16c0    Beep!BeepStartIo
DriverUnload: fffff801c02e1760    Beep!BeepUnload
AddDevice: 00000000
```

Dispatch routines:

[00] IRP_MJ_CREATE	fffff801c02e1430	Beep!BeepOpen
[01] IRP_MJ_CREATE_NAMED_PIPE	fffff8014875ad94	nt!IopInvalidDeviceRequest
[02] IRP_MJ_CLOSE	fffff801c02e1150	Beep!BeepClose
[03] IRP_MJ_READ	fffff8014875ad94	nt!IopInvalidDeviceRequest
[04] IRP_MJ_WRITE	fffff8014875ad94	nt!IopInvalidDeviceRequest
[05] IRP_MJ_QUERY_INFORMATION	fffff8014875ad94	nt!IopInvalidDeviceRequest
[...]		
[0c] IRP_MJ_DIRECTORY_CONTROL	fffff8014875ad94	nt!IopInvalidDeviceRequest
[0d] IRP_MJ_FILE_SYSTEM_CONTROL	fffff8014875ad94	nt!IopInvalidDeviceRequest
[0e] IRP_MJ_DEVICE_CONTROL	fffff801c02e1200	Beep!BeepDeviceControl
[0f] IRP_MJ_INTERNAL_DEVICE_CONTROL	fffff8014875ad94	nt!IopInvalidDeviceRequest
[10] IRP_MJ_SHUTDOWN	fffff8014875ad94	nt!IopInvalidDeviceRequest
[11] IRP_MJ_LOCK_CONTROL	fffff8014875ad94	nt!IopInvalidDeviceRequest
[...]		
[1b] IRP_MJ_PNP	fffff8014875ad94	nt!IopInvalidDeviceRequest

Devices

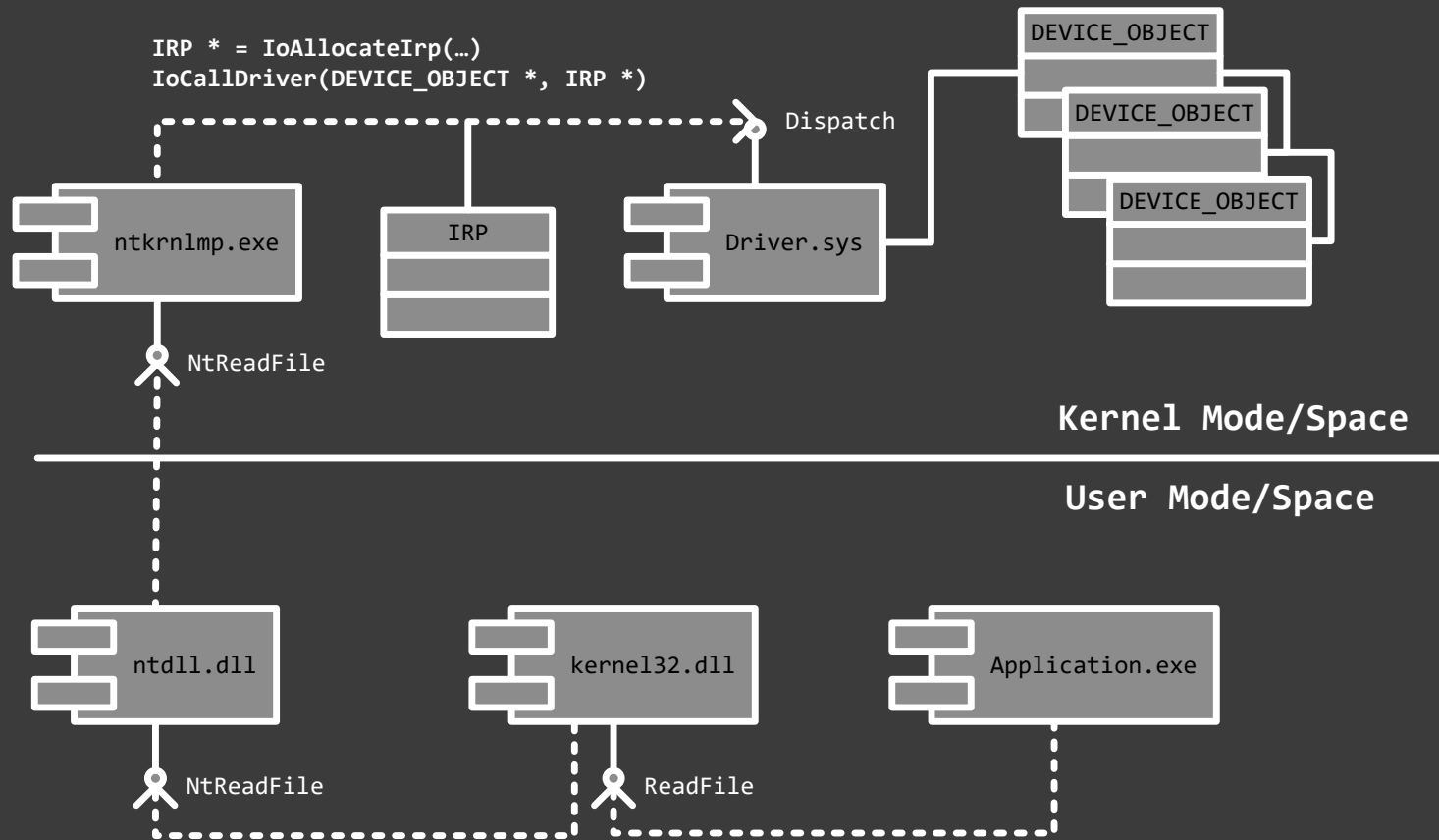
- Represents physical or logical device (**\Device\MousePad**)
- Target of an I/O operation
- Name: **\Device\<Name>** or **\FileSystem\<Name>**
- Implemented as a C structure



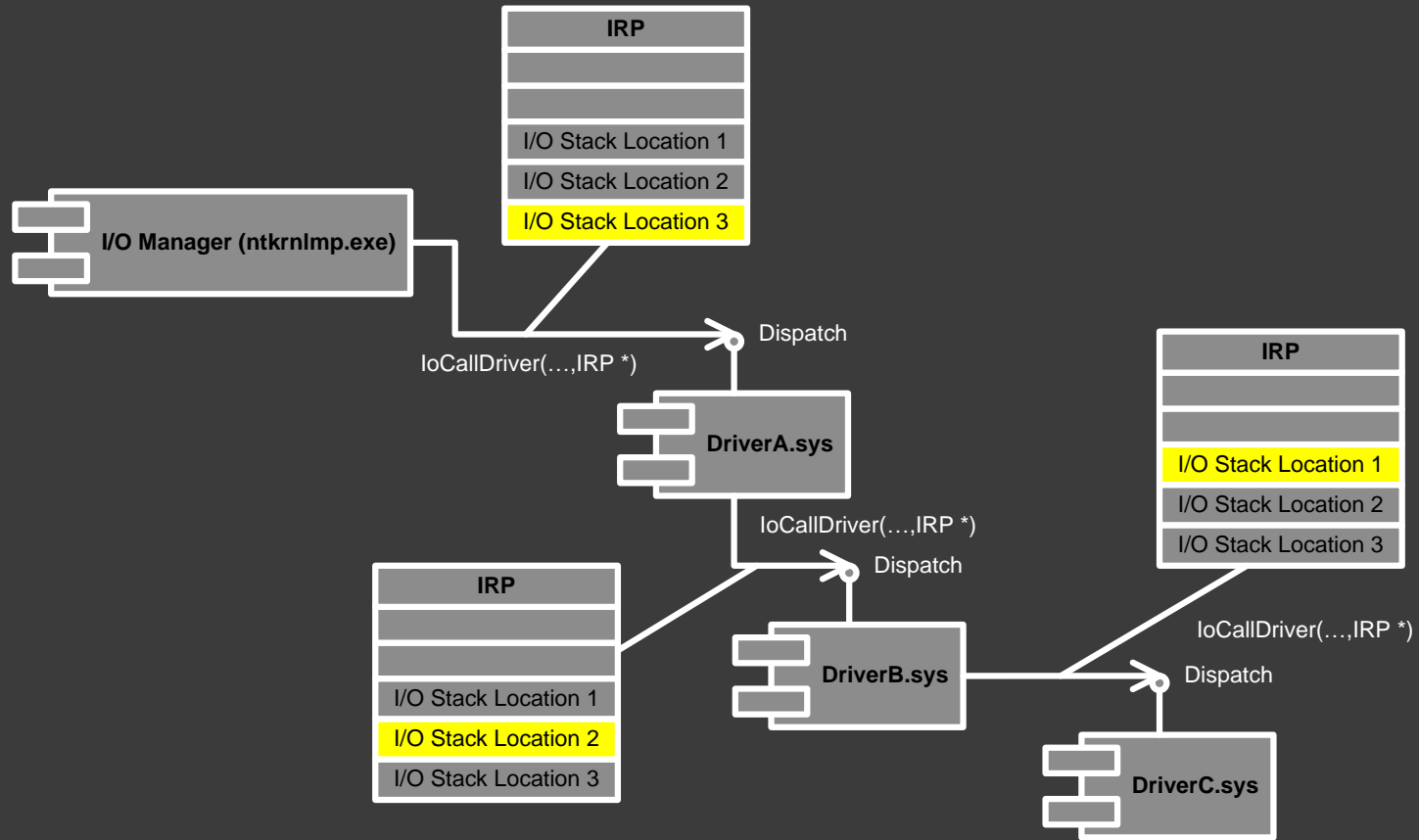
I/O Manager

- Provides an interface between drivers and OS
- Defines a detailed framework and specification for device drivers
- Provides support functions to drivers
- Packet-driven architecture: each I/O operation is described by IRP (I/O Request Packet) structure

Big Picture



IRP Communication



Exercise C10

- ◎ **Goal:** Learn how to inspect storage device queues and file system filter stack traces
- ◎ **Patterns:** Disk Packet Buildup; Stack Trace (File System Filters)
- ◎ [\AdvWMDA-Dumps\Exercise-C10-Storage-File-System-Filters.pdf](#)

Exercise C11

- ⦿ **Goal:** Learn how to analyze raw stack to mine for missing information manually
- ⦿ **Patterns:** Wait Chain (Window Messaging); Hidden Parameter; Data Correlation
- ⦿ [\AdvWMDA-Dumps\Exercise-C11-Window-Messaging.pdf](#)

Exercise C12

- ◎ **Goal:** Learn how to look for signs of past behavior
- ◎ **Patterns:** Black Box; Historical Information; Rough Stack Trace (Unmanaged Space)
- ◎ [\AdvWMDA-Dumps\Exercise-C12-Past-Behavior.pdf](#)

Pattern Links

[Stack Trace Collection \(Unmanaged Space\)](#)

[Coupled Processes \(Weak\)](#)

[Value References](#)

[Spiking Thread](#)

[Stack Trace Collection \(Predicate\)](#)

[Invalid Pointer](#)

[Disconnected Network Adapter](#)

[Stack Trace Collection \(I/O Requests\)](#)

[Wait Chain \(Window Messaging\)](#)

[Hidden Parameter](#)

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

[Incorrect Stack Trace](#)

[Stack Trace \(I/O Request\)](#)

[Stack Trace \(File System Filters\)](#)

[Input Thread](#)

[Black Box](#)

[Rough Stack Trace \(Unmanaged Space\)](#)

[Passive Thread](#)

[Virtualized Process](#)

[Module Variable](#)

[Thread Waiting Time](#)

[Self-Diagnosis](#)

[System Object](#)

[Historical Information](#)

[Main Thread](#)

[Execution Residue](#)

[Data Correlation](#)

[Coupled Processes \(Strong\)](#)

[Truncated Stack Trace](#)

[Stack Trace \(I/O Devices\)](#)

[Disk Packet Buildup](#)

[Debugger Bug](#)

[Historical Information](#)

[Memory Data Model](#)

Pattern Case Studies

70 multiple pattern case studies:

[http://www.dumpanalysis.org/blog/index.php/
pattern-cooperation/](http://www.dumpanalysis.org/blog/index.php/pattern-cooperation/)

Pattern Interaction chapters in
Memory Dump Analysis Anthology

[Hunting for a Driver](#)

Resources

- WinDbg Help / WinDbg.org (quick links)
- DumpAnalysis.org / SoftwareDiagnostics.Institute / PatternDiagnostics.com
- Debugging.TV / YouTube.com/DebuggingTV / YouTube.com/PatternDiagnostics
- WinDbg images <https://hub.docker.com/r/patterndiagnostics/windbg>
- UML Distilled
- Windows Kernel Programming
- Windows Internals, 7th ed.
- [Accelerated Windows Memory Dump Analysis, 5th Edition, Revision 3](#)
- [Encyclopedia of Crash Dump Analysis Patterns, 3rd Edition](#)
- [Memory Dump Analysis Anthology \(Diagnomicon\)](#)



Going Further

More basic/foundational:

- ◉ [Practical Foundations of Windows Debugging, Disassembling, Reversing](#)
- ◉ [Accelerated Windows Memory Dump Analysis](#)

Special topics:

- ◉ [Accelerated Windows Malware Analysis with Memory Dumps](#)
- ◉ [Accelerated Disassembly, Reconstruction and Reversing](#)
- ◉ [Accelerated .NET Core Memory Dump Analysis](#)
- ◉ [Accelerated Windows Debugging⁴](#)
- ◉ [Extended Windows Memory Dump Analysis](#)

Q&A

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

Thank you for attendance!