Windows
Memory Dump Analysis
Advanced

with Data Structures

Version 5

Dmitry Vostokov
Software Diagnostics Services

Prerequisites

Basic and intermediate level

Windows memory dump analysis

© 2024 Software Diagnostics Services

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

© 2024 Software Diagnostics Services

Training Principles

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

© 2024 Software Diagnostics Services

Practice Exercises

© 2024 Software Diagnostics Services

Links

- Memory Dumps:

Included in Exercise 0

- Exercise Transcripts:

Included in this book

© 2024 Software Diagnostics Services

Exercise 0

- **Goal:** Install WinDbg 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

© 2024 Software Diagnostics Services

Complete Memory Dumps

Exercises C1 – C13

© 2024 Software Diagnostics Services

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); Active Thread; Passive Thread; Coupled Processes (Weak); Coupled Processes (Strong); Wait Chain (ALPC); Zombie Processes; Stack Trace Collection (Predicate); Stack Trace Collection (CPUs); Input Thread; Truncated Stack Trace; Memory Data Model

- \AdvWMDA-Dumps\Exercise-C1A-Stack-Trace-Collection.pdf

© 2024 Software Diagnostics Services

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

© 2024 Software Diagnostics Services

Exercise C2

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

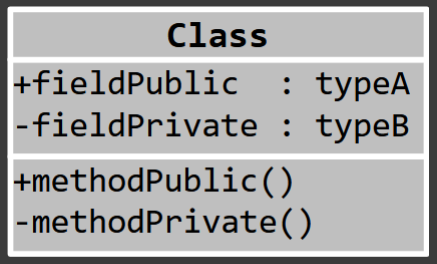
© 2024 Software Diagnostics Services

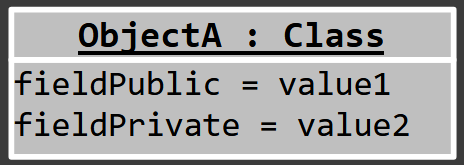
A Crash Dump Course in Unified Modeling Language

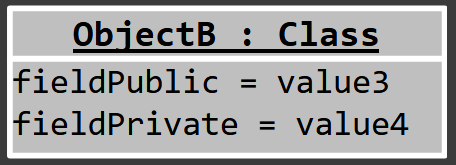
Part I

© 2024 Software Diagnostics Services

Classes and Objects

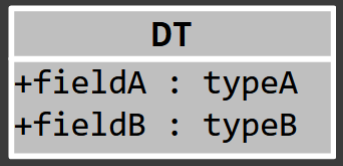


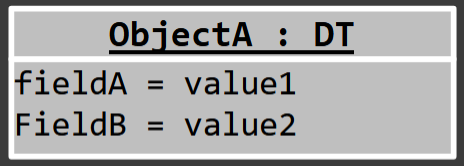


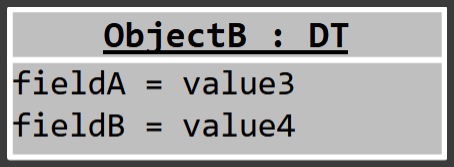


© 2024 Software Diagnostics Services

Data Types as Classes







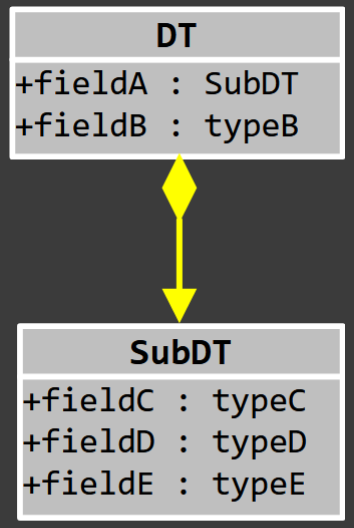
© 2024 Software Diagnostics Services

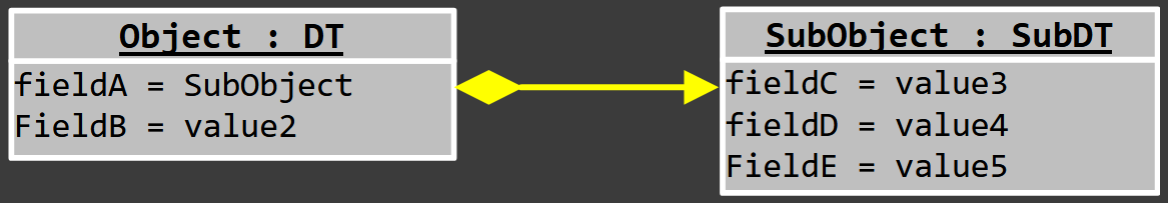
Memory as Objects

```
0: kd> dt nt!_DISPATCHER_HEADER
   +0x000 Lock             : Int4B
   +0x000 LockNV           : Int4B
   +0x000 Type             : UChar
   +0x001 Signalling       : UChar
   +0x002 Size             : UChar
   +0x003 Reserved1        : UChar
   +0x000 TimerType        : Uchar
...

0: kd> dt nt!_DISPATCHER_HEADER ffffbe0c89120080
   +0x000 Lock             : 0n3
   +0x000 LockNV           : 0n3
   +0x000 Type             : 0x3 ''
   +0x001 Signalling       : 0 ''
   +0x002 Size             : 0 ''
   +0x003 Reserved1        : 0 ''
   +0x000 TimerType        : 0x3 ‘’
...
```

© 2024 Software Diagnostics Services

Aggregation





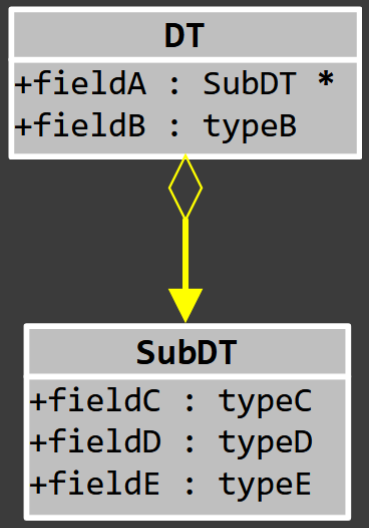
© 2024 Software Diagnostics Services

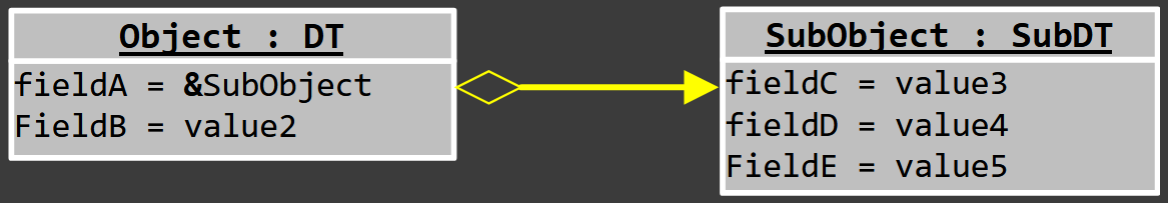
Aggregation in Memory

```
0: kd> dt nt!_EPROCESS
   +0x000 Pcb              : _KPROCESS
   +0x438 ProcessLock      : _EX_PUSH_LOCK
   +0x440 UniqueProcessId  : Ptr64 Void
   +0x448 ActiveProcessLinks : _LIST_ENTRY
   +0x458 RundownProtect   : _EX_RUNDOWN_REF
   +0x460 Flags2           : Uint4B
...

0: kd> dt nt!_KPROCESS
   +0x000 Header           : _DISPATCHER_HEADER
   +0x018 ProfileListHead  : _LIST_ENTRY
   +0x028 DirectoryTableBase : Uint8B
   +0x030 ThreadListHead   : _LIST_ENTRY
   +0x040 ProcessLock      : Uint4B
   +0x044 ProcessTimerDelay : Uint4B
   +0x048 DeepFreezeStartTime : Uint8B
   +0x050 Affinity         : _KAFFINITY_EX
...
```

© 2024 Software Diagnostics Services

Composition





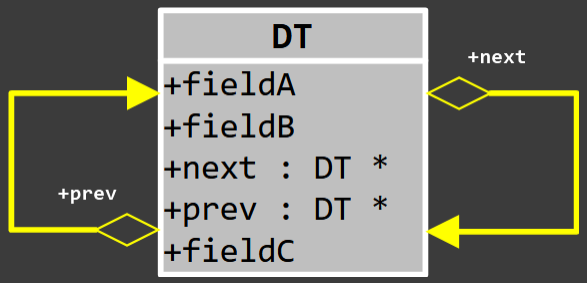
© 2024 Software Diagnostics Services

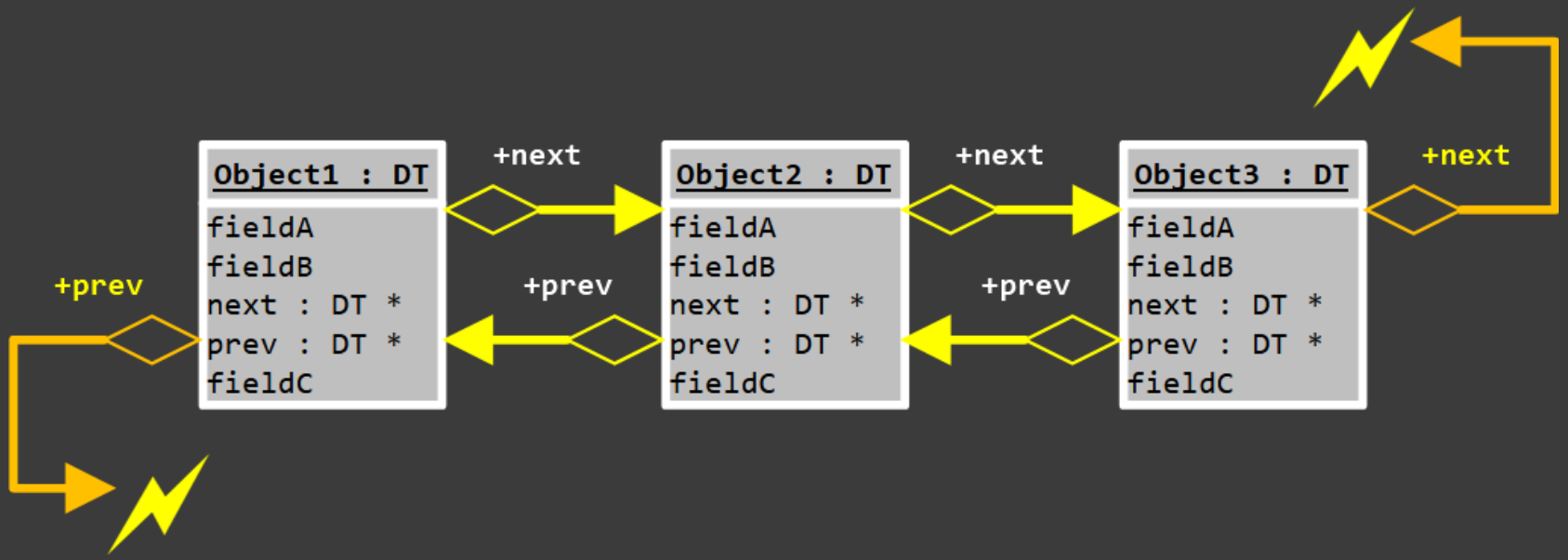
Composition in Memory

```
0: kd> dt nt!_EPROCESS
...
   +0x548 OwnerProcessId   : Uint8B
   +0x550 Peb              : Ptr64 _PEB
   +0x558 Session          : Ptr64 _MM_SESSION_SPACE
   +0x560 Spare1           : Ptr64 Void
...

0: kd> dt nt!_EPROCESS ffffbe0c89120080
   +0x548 OwnerProcessId   : 0x2b4
   +0x550 Peb              : 0x0000005b`47b81000 _PEB
   +0x558 Session          : 0xffffbe0c`87efbaa0 _MM_SESSION_SPACE
   +0x560 Spare1           : (null)
...

0: kd> dt nt!_PEB 0x0000005b`47b81000
   +0x000 InheritedAddressSpace : 0 ''
   +0x001 ReadImageFileExecOptions : 0 ''
   +0x002 BeingDebugged    : 0 ‘’
...
```

© 2024 Software Diagnostics Services

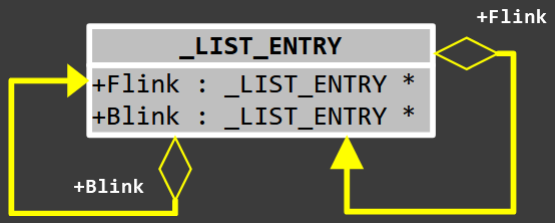
Linked List

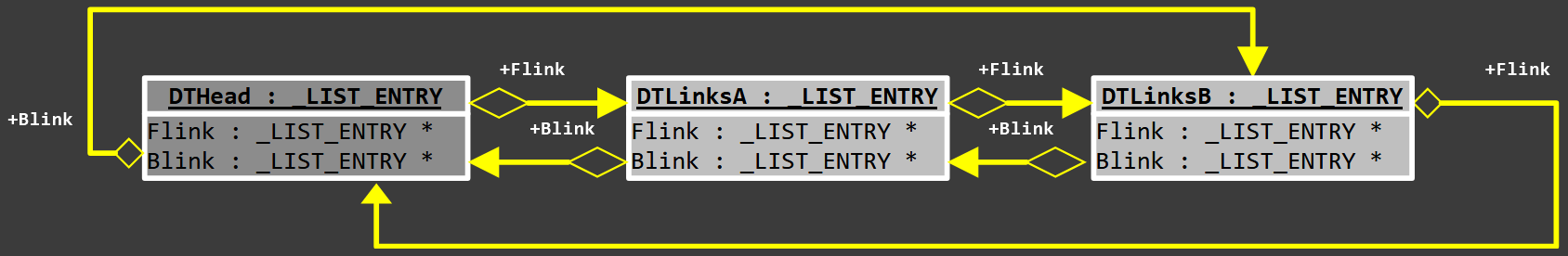




© 2024 Software Diagnostics Services

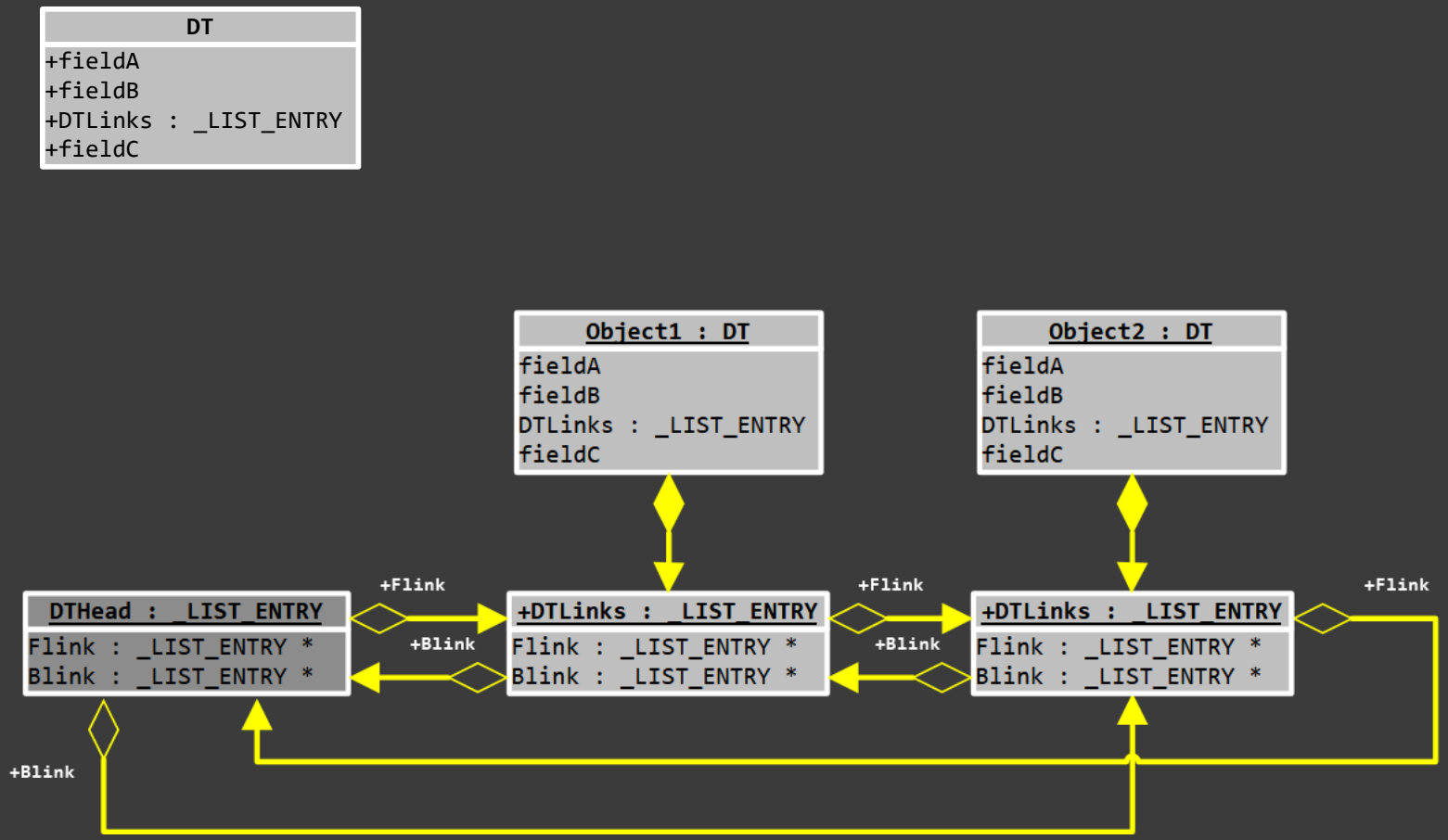
_LIST_ENTRY





© 2024 Software Diagnostics Services

Linked Data Structures



© 2024 Software Diagnostics Services

Exercise C3A

- **Goal:** Learn how to navigate linked lists
- **Patterns:** Structure Field Collection
- \AdvWMDA-Dumps\Exercise-C3A-Linked-Lists.pdf

© 2024 Software Diagnostics Services

Exercise C3B

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

© 2024 Software Diagnostics Services

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

© 2024 Software Diagnostics Services

Exercise C4B

- **Goal:** Learn how to create scripts to extend WinDbg functionality (via JavaScript scripting)

- \AdvWMDA-Dumps\Exercise-C4B-Scripting.pdf

© 2024 Software Diagnostics Services

Exercise C5

- **Goal:** Learn how to inspect the registry
- \AdvWMDA-Dumps\Exercise-C5-Registry.pdf

© 2024 Software Diagnostics Services

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

© 2024 Software Diagnostics Services

Exercise C7

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

© 2024 Software Diagnostics Services

Exercise C8

- **Goal:** Learn how to inspect network protocols and adapters
- **Patterns:** Disconnected Network Adapter
- \AdvWMDA-Dumps\Exercise-C8-Network.pdf

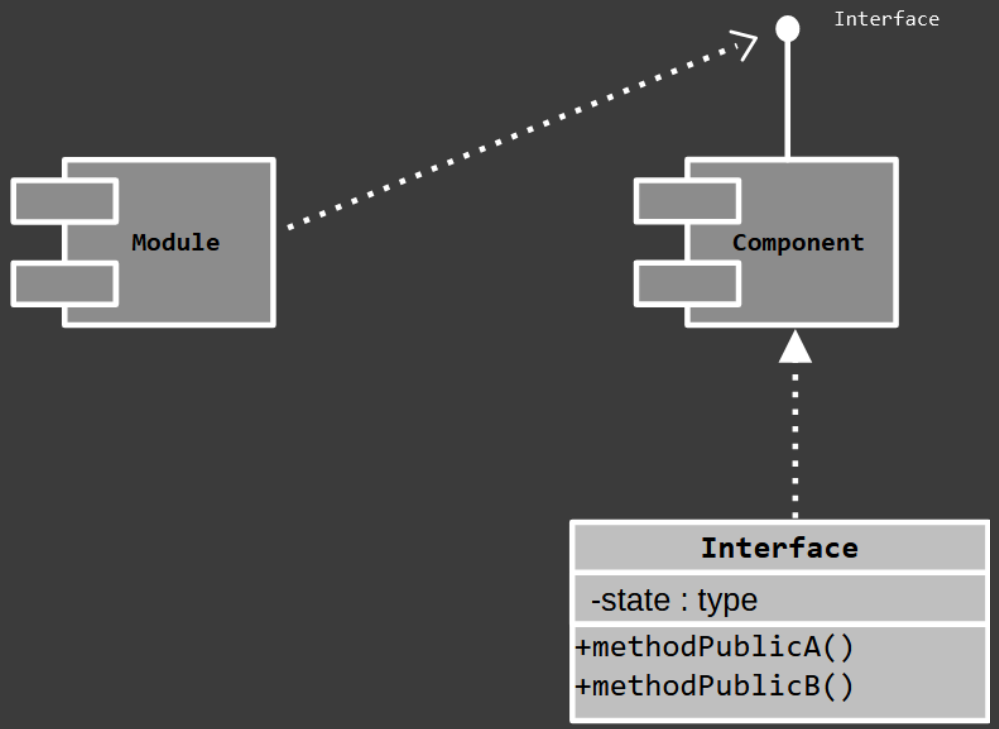
© 2024 Software Diagnostics Services

A Crash Dump Course in Unified Modeling Language

Part II

© 2024 Software Diagnostics Services

Components and Interfaces



© 2024 Software Diagnostics Services

Classes and Objects (C++)

```
class Interface // it can also be a struct Interface
{
public:
    void method();
private:
    int state;
};

Interface ObjectA, ObjectB;

void Interface::method(/* Interface *this */)
{
    state = 0; // this->state = 0;
}

void foo()
{
    ObjectA.method();
}
```

© 2024 Software Diagnostics Services

Objects and Methods (C)

```
struct Interface
{
    int state;
};

void method(struct Interface *obj);

struct Interface ObjectA, ObjectB;

void method(struct Interface *obj)
{
    obj->state = 0;
}

void foo()
{
    method(&ObjectA);
}
```

© 2024 Software Diagnostics Services

Classes and Objects Analogy (C)

```
struct Object
{
    int state;
};

struct Interface
{
    void (*method)(struct Object *obj);
};

void method(struct Object *obj)
{
    obj->state = 0;
}

struct Interface ClassA = {&method}; struct Object ObjectA, ObjectB;

void foo()
{
    ClassA.method(&ObjectA);
}
```

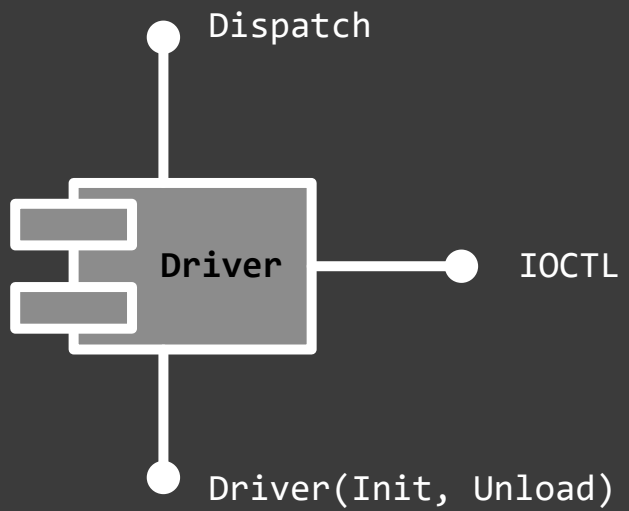
© 2024 Software Diagnostics Services

A Crash Dump Course in Windows Internals

© 2024 Software Diagnostics Services

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
- Class analogy



\Driver\<Name>

\FileSystem\<Name>

© 2024 Software Diagnostics Services

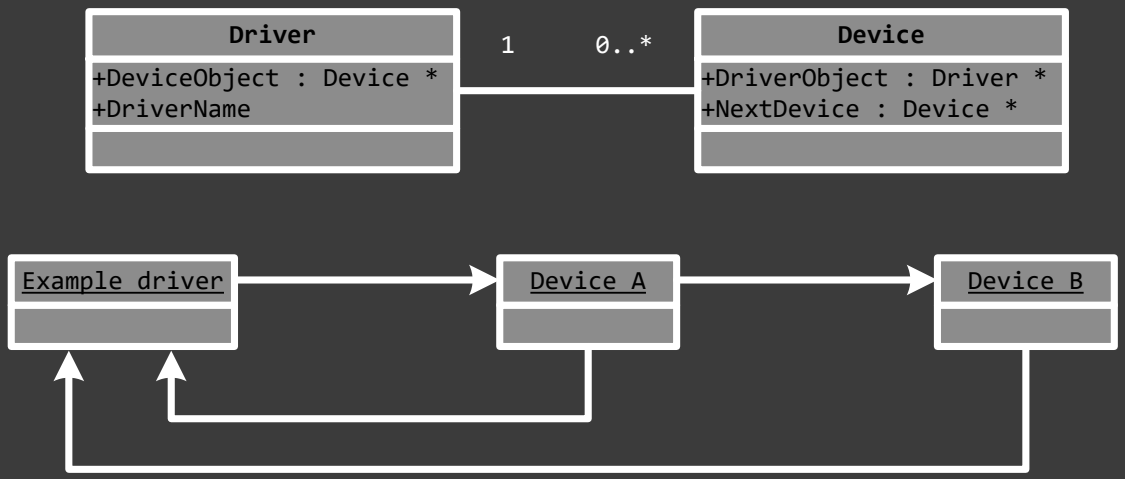
Device Driver Example

```
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
```

© 2024 Software Diagnostics Services

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
- Object analogy



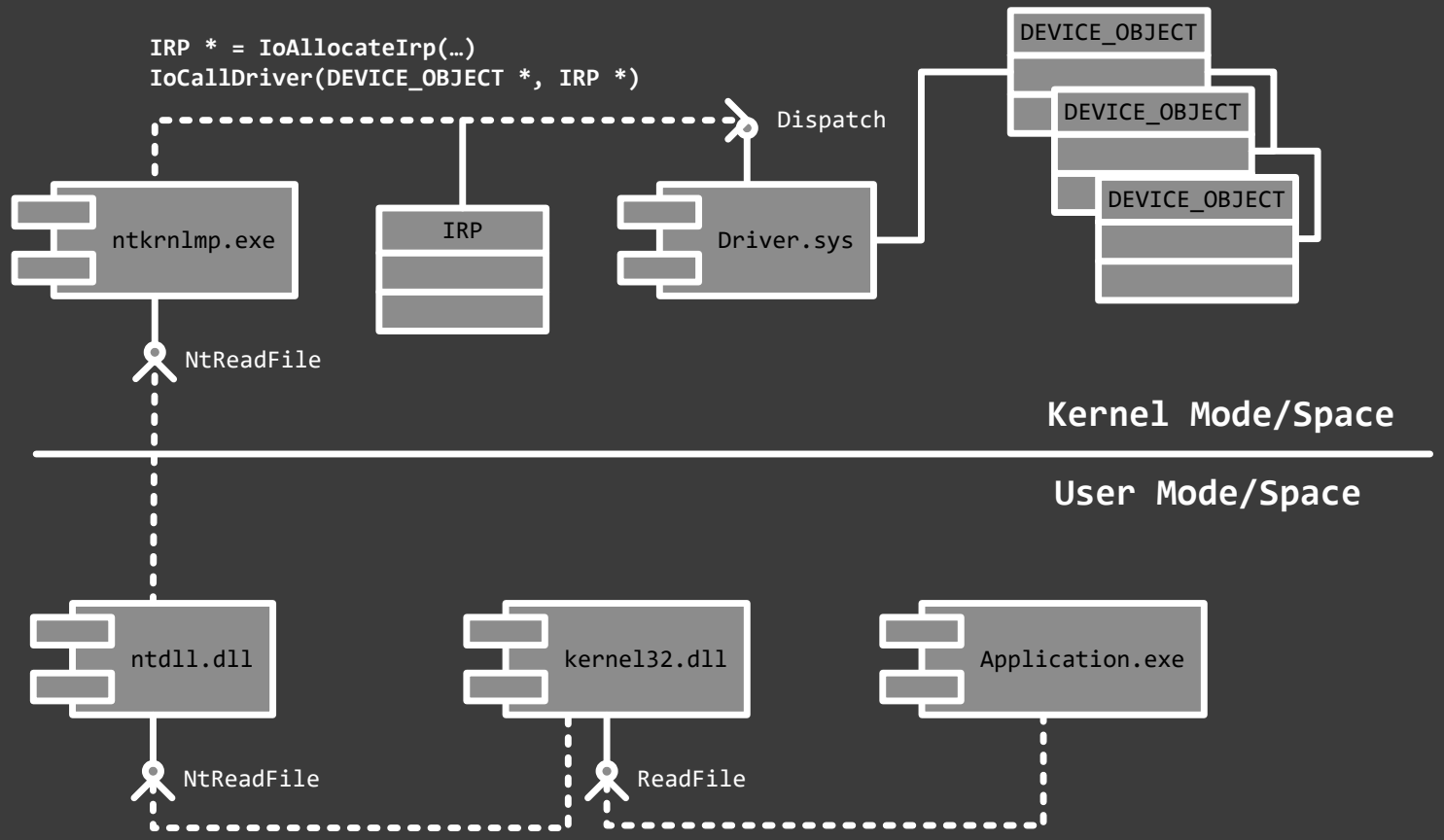
© 2024 Software Diagnostics Services

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

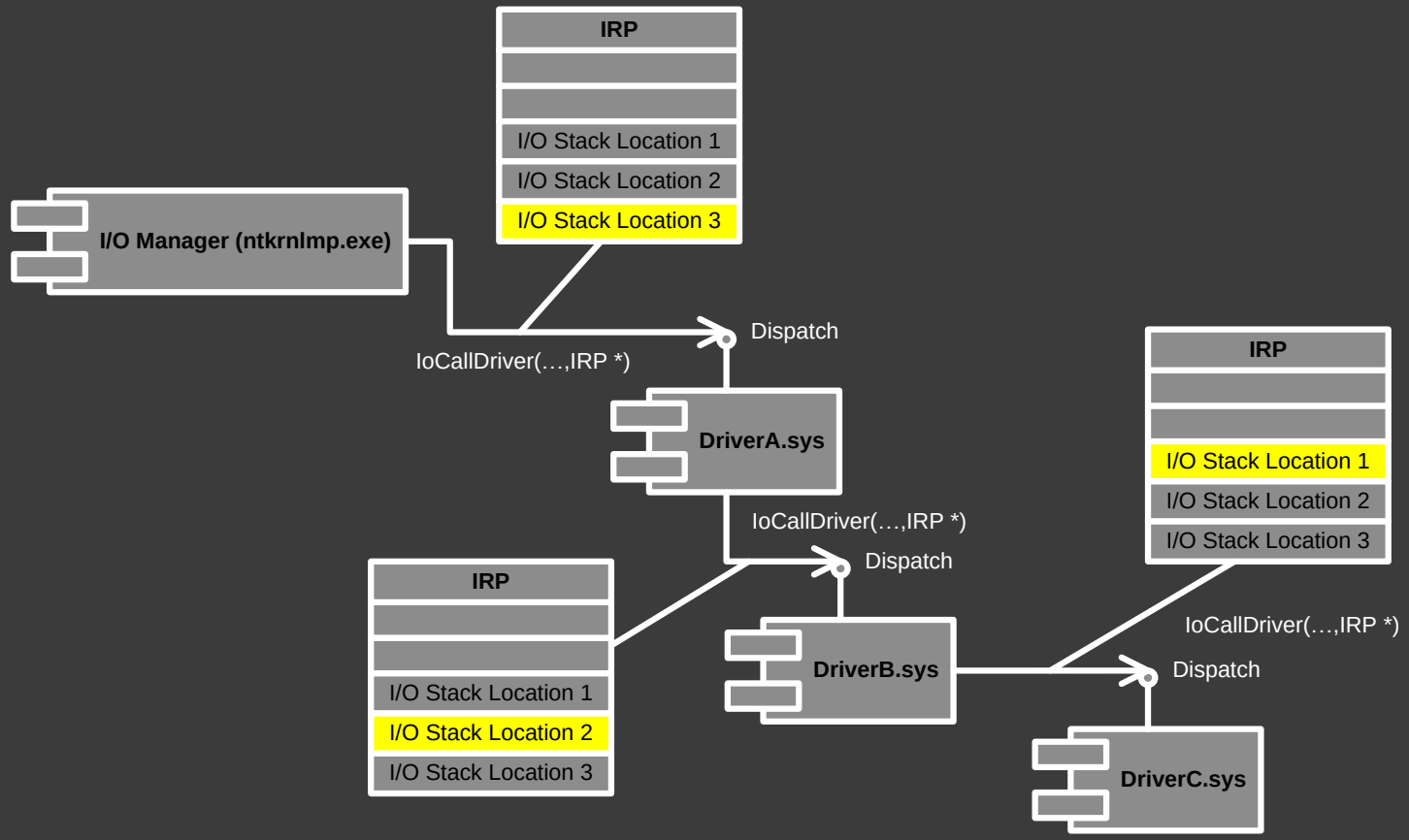
© 2024 Software Diagnostics Services

Big Picture



© 2024 Software Diagnostics Services

IRP Communication



© 2024 Software Diagnostics Services

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

© 2024 Software Diagnostics Services

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

© 2024 Software Diagnostics Services

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

© 2024 Software Diagnostics Services

Exercise C12

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

© 2024 Software Diagnostics Services

Warning

Because of the evolving nature of Generative AI LLMs and their differences, the following exercise output may differ from what you get when reproducing the results.

© 2024 Software Diagnostics Services

Exercise C13

- **Goal:** Learn how to use a Generative AI LLM as a memory dump analysis assistant
- **Patterns:** Annotated Stack Trace; Disassembly Summary; Region Summary; Coincidental Symbolic Information; Analysis Summary
- \AdvWMDA-Dumps\Exercise-C13-Generative-AI-LLM.pdf

© 2024 Software Diagnostics Services

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)
Zombie Processes
Unloaded Module
Annotated Stack Trace
Analysis Summary
Coincidental Symbolic Information
Passive Thread
Virtualized Process
Module Variable
Thread Waiting Time
Foreign Stack
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
Past Process
Memory Data Model
Active Thread
Structure Field Collection
Disassembly Summary
Region Summary

© 2024 Software Diagnostics Services

Pattern Case Studies

70 multiple pattern case studies:

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

Pattern Interaction chapters in Memory Dump Analysis Anthology

Hunting for a Driver

© 2024 Software Diagnostics Services

Resources

- WinDbg Help / WinDbg.org (quick links)
- DumpAnalysis.org / SoftwareDiagnostics.Institute / PatternDiagnostics.com
- Debugging.TV / YouTube.com/DebuggingTV / YouTube.com/PatternDiagnostics
- UML Distilled, 3rd ed.
- Schaum's Outline of UML, 2nd ed.
- Windows Kernel Programming, 2nd ed.
- Windows Internals, 7th ed.
- Accelerated Windows Memory Dump Analysis, 6th Edition
- Encyclopedia of Crash Dump Analysis Patterns, 3rd Edition
- Memory Dump Analysis Anthology (Diagnomicon)

© 2024 Software Diagnostics Services

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 Debugging[4]
- Extended Windows Memory Dump Analysis (Python scripting)
- Accelerated Windows API for Software Diagnostics
- Accelerated C & C++ for Windows Diagnostics (Memory thinking)

© 2024 Software Diagnostics Services

Q&A

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

© 2024 Software Diagnostics Services

Thank you for attendance!

© 2024 Software Diagnostics Services