Public Preview Version

Mac OS X

Core Dump Analysis

Accelerated

Version 2.0

Dmitry Vostokov
Software Diagnostics Services

Prerequisites

Basic Mac OS X troubleshooting

GDB Commands

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

LLDB Commands

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

© 2014 Software Diagnostics Services

Training Goals

- Review fundamentals
- Learn how to collect core dumps
- Learn how to analyze core dumps

© 2014 Software Diagnostics Services

Training Principles

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

© 2014 Software Diagnostics Services

Schedule Summary

Day 1

- Analysis Fundamentals (30 minutes)
- Core dump collection methods (10 minutes)
- Basic Core Memory Dumps (1 hour 20 minutes)

Day 2

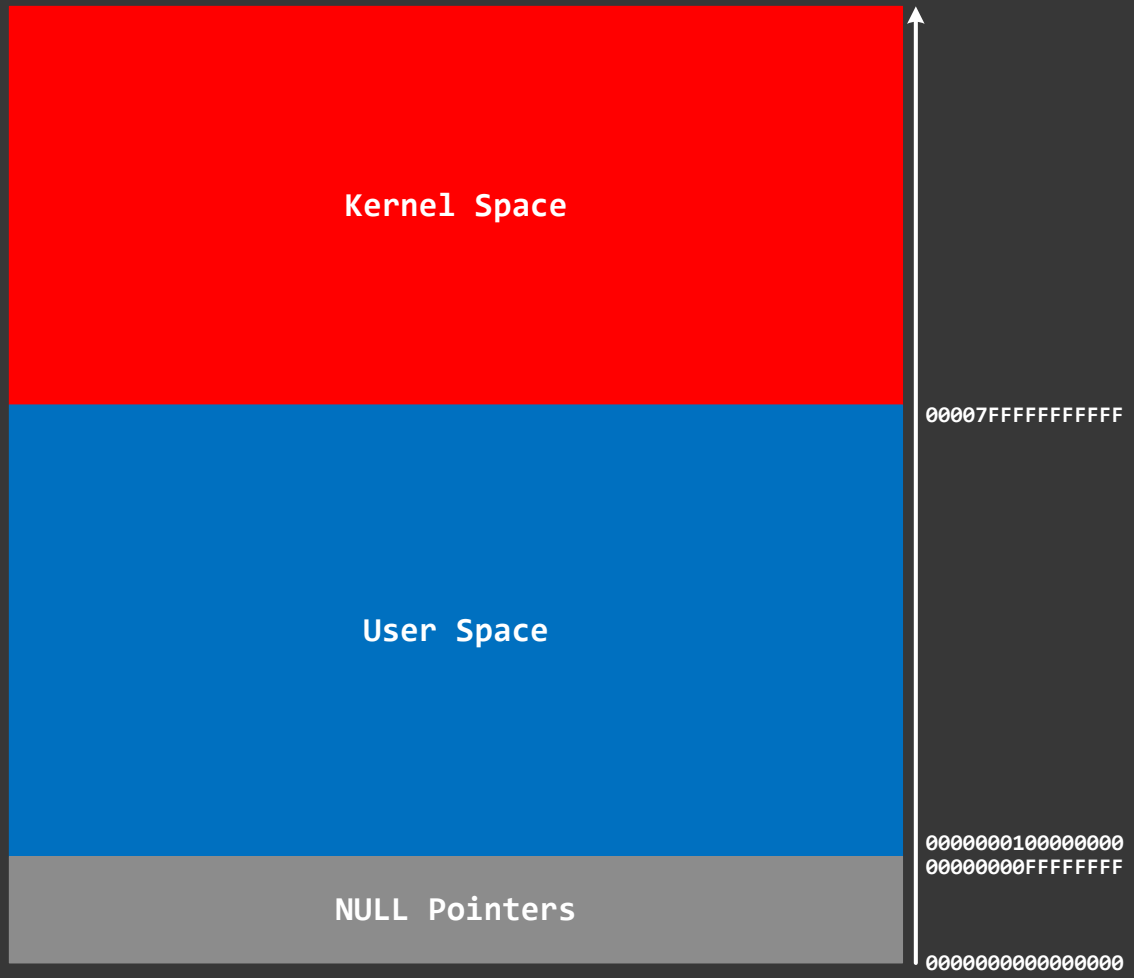
- Core Memory Dumps (2 hours)

© 2014 Software Diagnostics Services

Part 1: Fundamentals

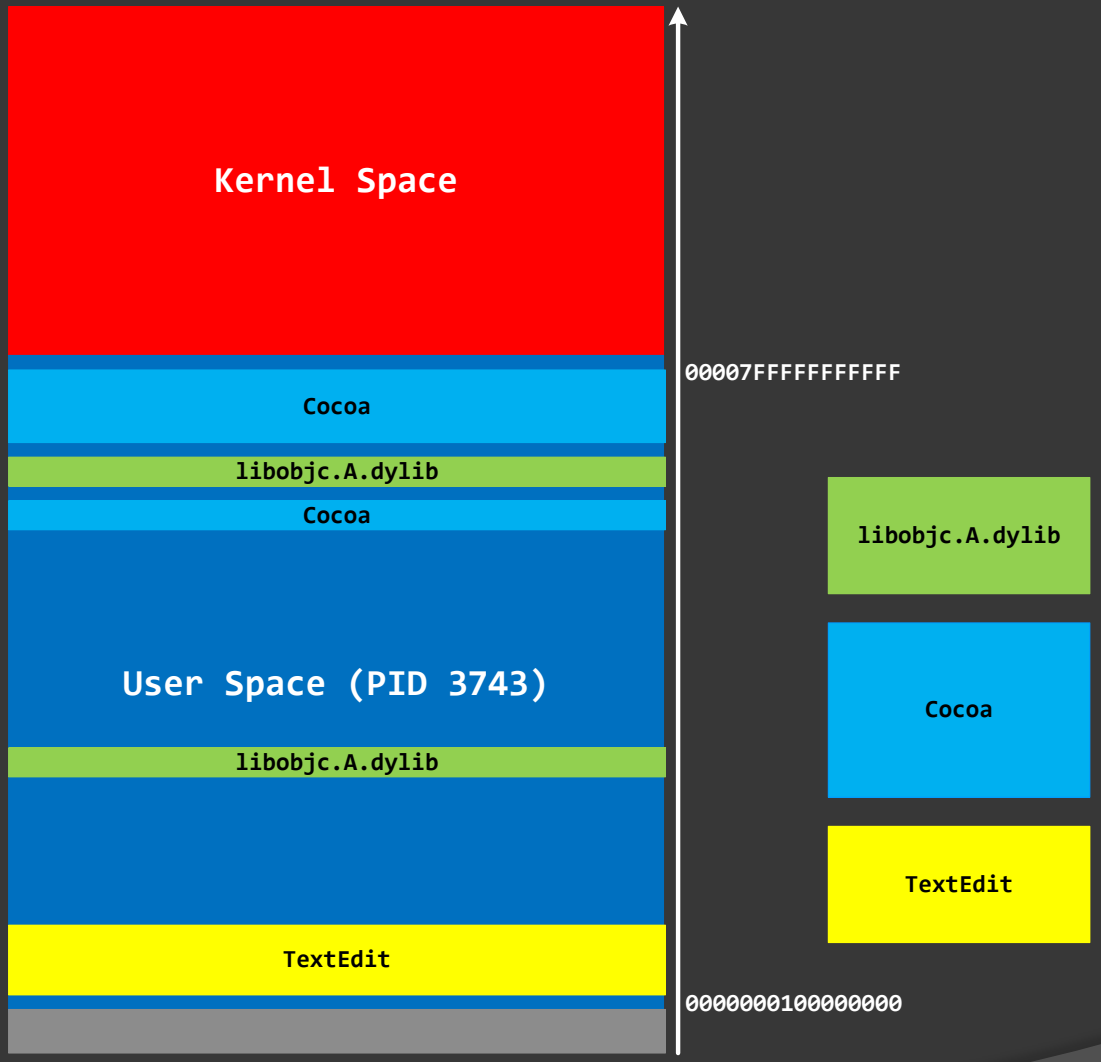
© 2014 Software Diagnostics Services

Memory/Kernel/User Space



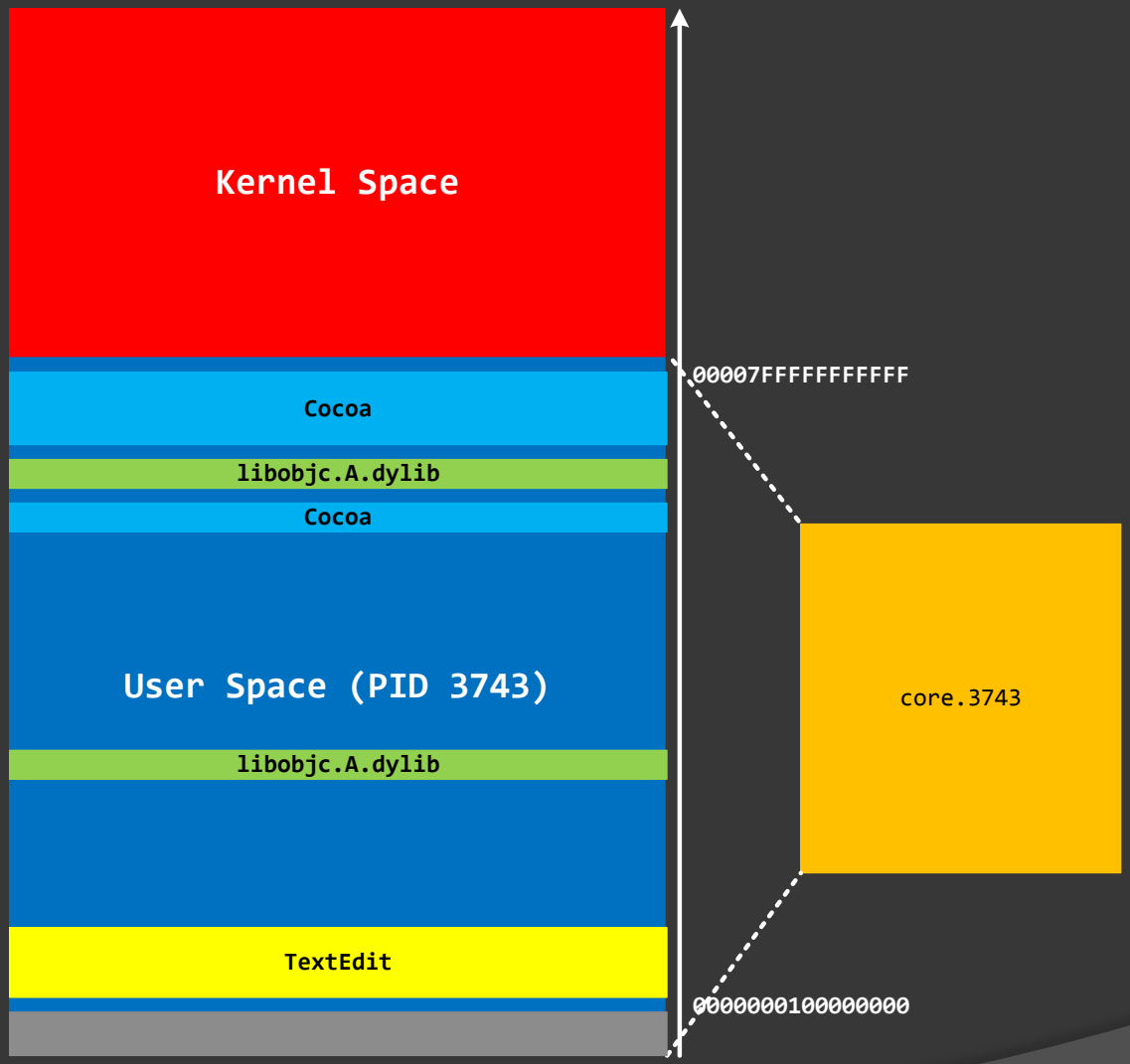
© 2014 Software Diagnostics Services

App/Process/Library



© 2014 Software Diagnostics Services

Process Memory Dump



GDB Commands

info sharedlibrary
Lists dynamic libraries

maintenance info sections
Lists memory regions

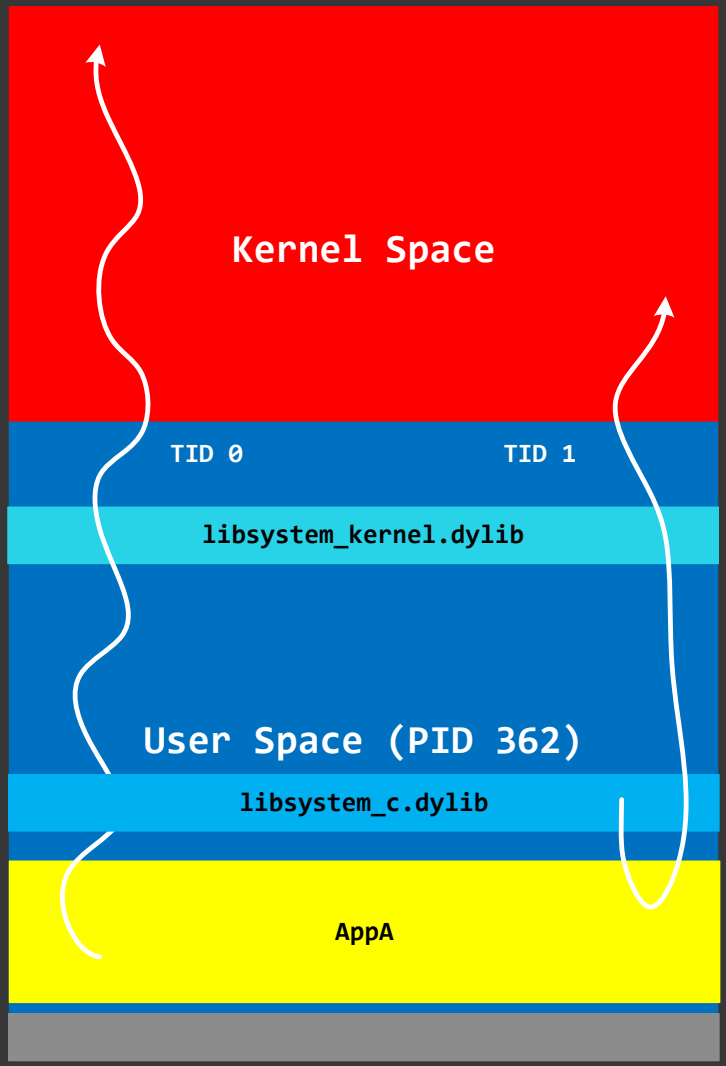
LLDB Commands

image list
Lists dynamic libraries

image dump sections
Lists memory regions

© 2014 Software Diagnostics Services

Process Threads



GDB Commands

info threads
Lists threads

thread <n>
Switches between threads

thread apply all bt
Lists stack traces from all threads

LLDB Commands

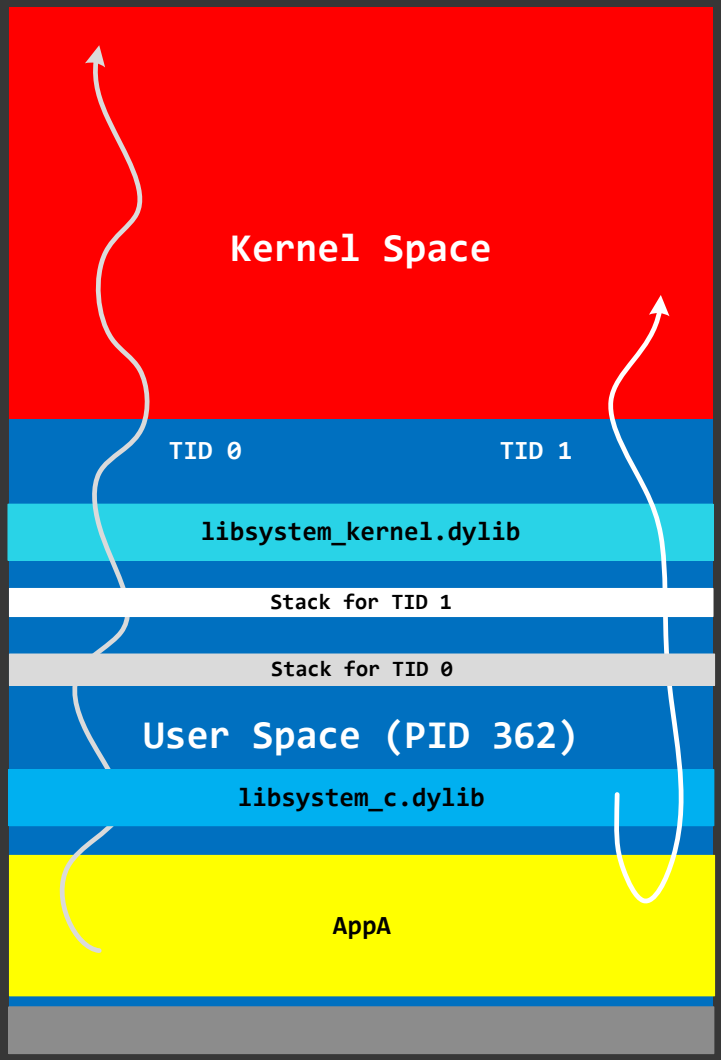
thread list
Lists threads

thread select <n>
Switches between threads

thread backtrace all
Lists stack traces from all threads

© 2014 Software Diagnostics Services

Thread Stack Raw Data



GDB Commands

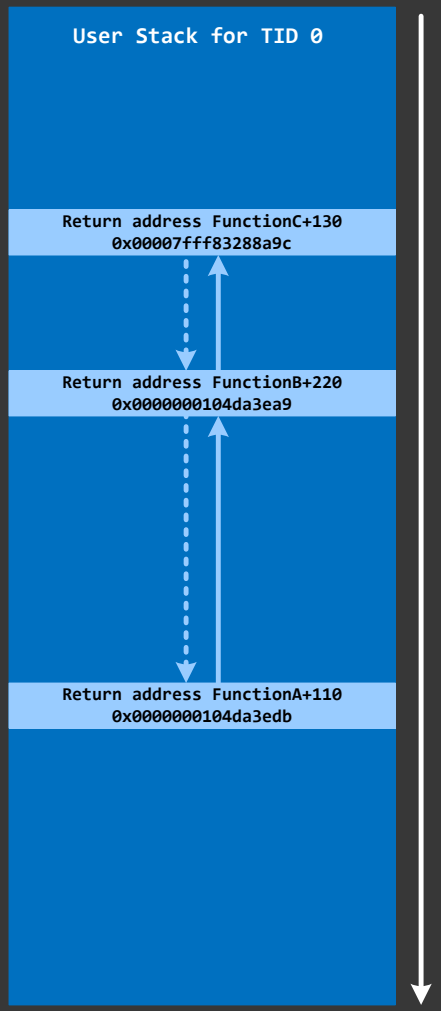
x/<n>a <address>
Prints n addresses with corresponding symbol mappings if any

LLDB Commands

x/<n>a <address>
Prints n addresses with corresponding symbol mappings if any

© 2014 Software Diagnostics Services

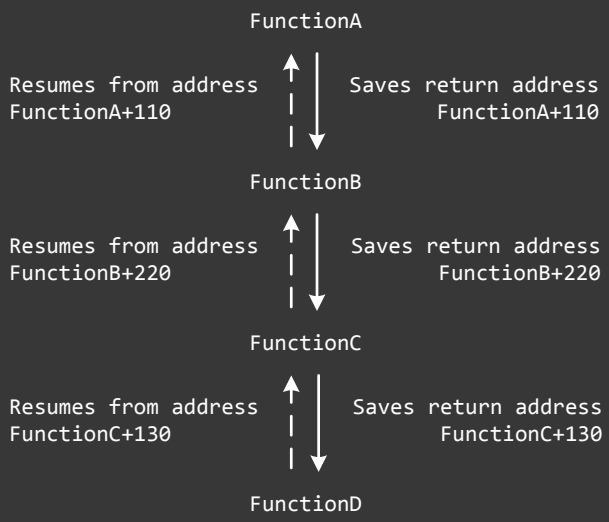
Thread Stack Trace




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

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

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



LLDB Commands

```
(lldb) bt
frame #0: 0x00007fff885e982a Module`FunctionD + offset
frame #1: 0x00007fff83288a9c Module`FunctionC + 130
frame #2: 0x0000000104da3ea9 AppA`FunctionB + 220
frame #3: 0x0000000104da3edb AppA`FunctionA + 110
```

GDB Commands

```
(gdb) bt
#0 0x00007fff885e982a in FunctionD ()
#1 0x00007fff83288a9c in FunctionC ()
#2 0x0000000104da3ea9 in FunctionB ()
#3 0x0000000104da3edb in FunctionA ()
```

© 2014 Software Diagnostics Services

GDB and LLDB vs. WinDbg

GDB Commands

```
(gdb) bt
#0 0x00007fff885e982a in FunctionD ()
#1 0x00007fff83288a9c in FunctionC ()
#2 0x0000000104da3ea9 in FunctionB ()
#3 0x0000000104da3edb in FunctionA ()
```

LLDB Commands

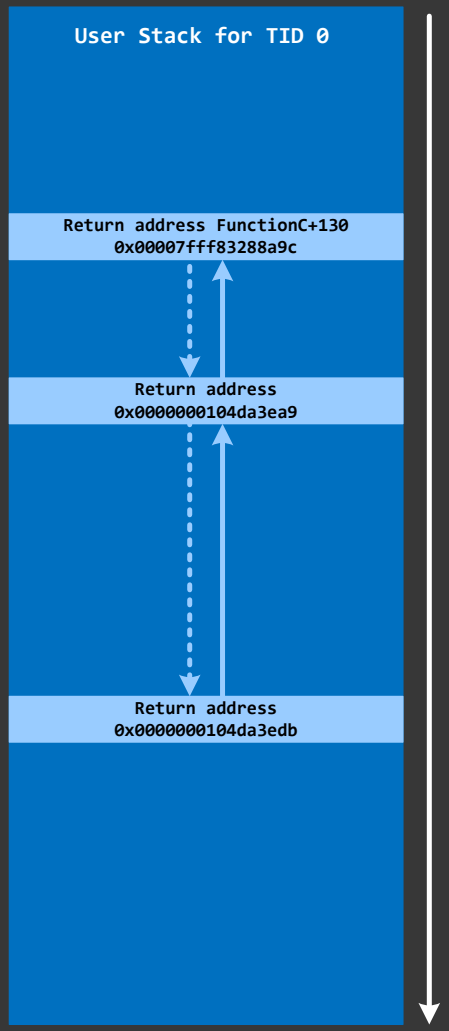
```
(lldb) bt
frame #0: 0x00007fff885e982a Module`FunctionD + offset
frame #1: 0x00007fff83288a9c Module`FunctionC + 130
frame #2: 0x0000000104da3ea9 AppA`FunctionB + 220
frame #3: 0x0000000104da3edb AppA`FunctionA + 110
```

WinDbg Commands

```
0:000> kn
00 00007fff83288a9c Module!FunctionD+offset
01 0000000104da3ea9 Module!FunctionC+130
02 0000000104da3edb AppA!FunctionB+220
03 0000000000000000 AppA!FunctionA+110
```

© 2014 Software Diagnostics Services

Thread Stack Trace (no dSYM)



Symbol file AppA.dSYM

FunctionA 22000 - 23000
FunctionB 32000 - 33000

GDB Commands

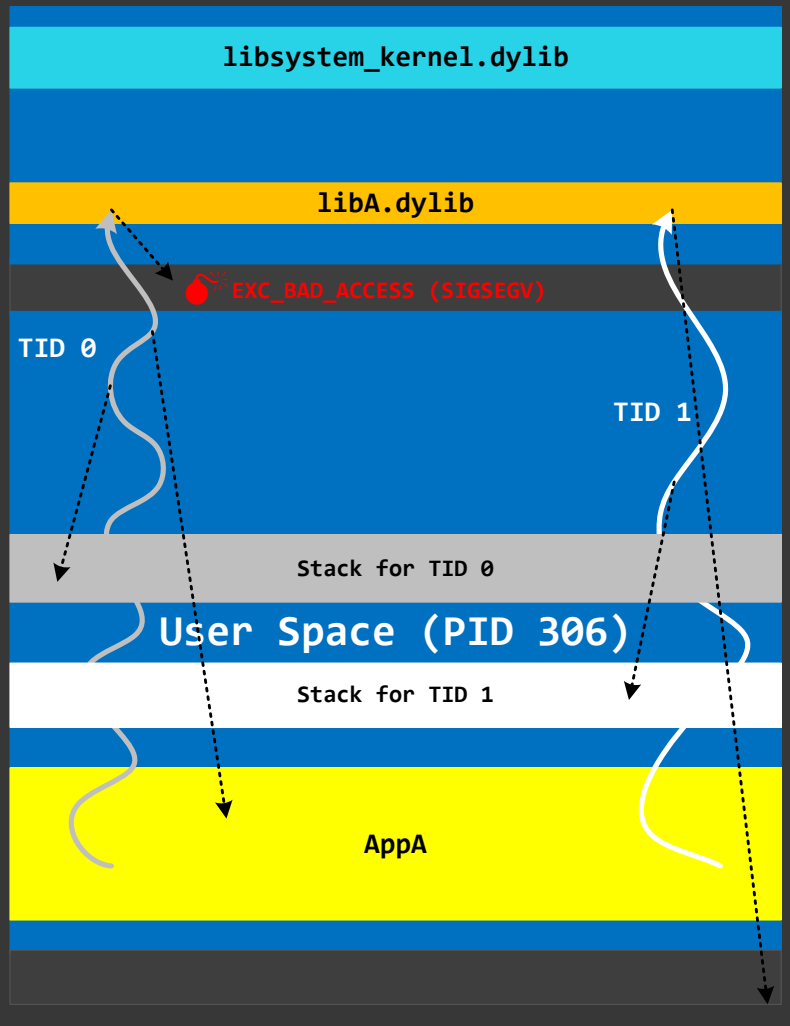
```
(gdb) bt
#0 0x00007fff885e982a in FunctionD ()
#1 0x00007fff83288a9c in FunctionC ()
#2 0x0000000104da3ea9 in ?? ()
#3 0x0000000104da3edb in ?? ()
```

LLDB Commands

```
(lldb) bt
frame #0: 0x00007fff885e982a Module`FunctionD + offset
frame #1: 0x00007fff83288a9c Module`FunctionC + 130
frame #2: 0x0000000104da3ea9 AppA`___lldb_unnamed_function1$$AppA + 220
frame #3: 0x0000000104da3edb AppA`___lldb_unnamed_function2$$AppA + 110
```

© 2014 Software Diagnostics Services

Exceptions (Access Violation)



GDB Commands

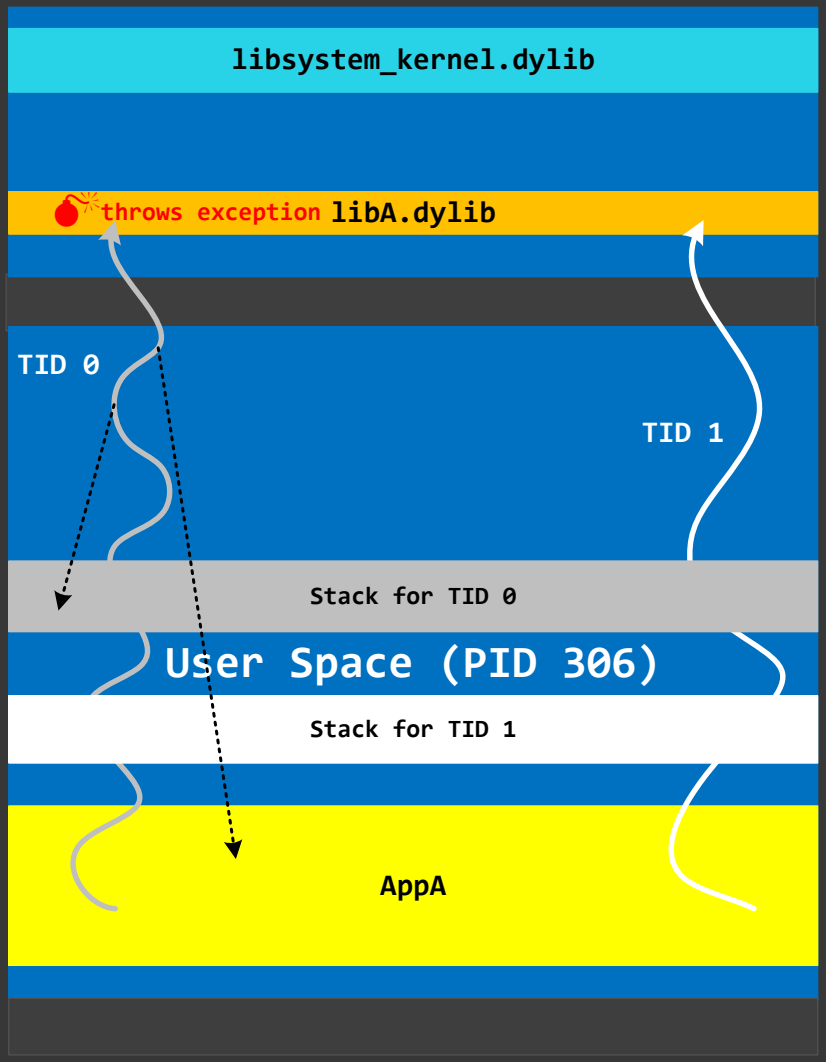
```
(gdb) x <address>
0x<address>: Cannot access
memory at address 0x<address>
```

LLDB Commands

```
(lldb) x <address>
error: core file does not contain 0x<address>
```

© 2014 Software Diagnostics Services

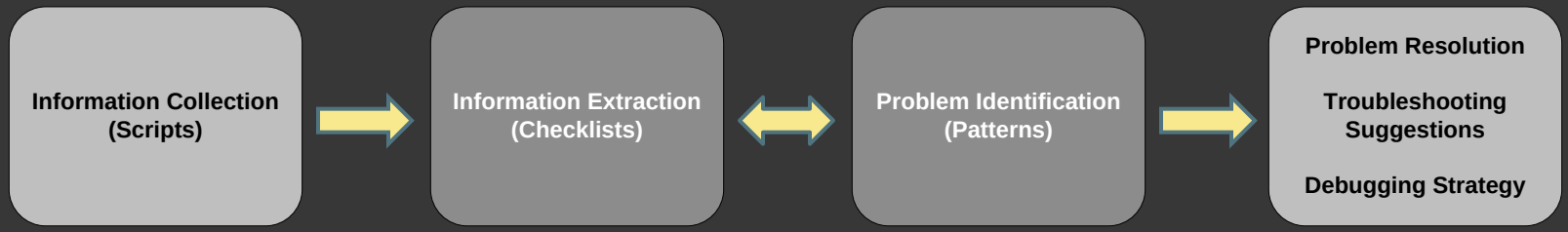
Exceptions (Runtime)



© 2014 Software Diagnostics Services

Pattern-Driven Analysis

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



© 2014 Software Diagnostics Services

Part 2: Core Dump Collection

© 2014 Software Diagnostics Services

Enabling Collection

- Temporary for the current terminal session:

```
$ ulimit -c unlimited
```

- Permanent for every user:

```
$ sudo vi /etc/launchd.conf
```

Add the line: `limit core unlimited`

© 2014 Software Diagnostics Services

Generation Methods

- Command line:

```
$ kill –s SIGQUIT PID
$ kill –s SIGABRT PID
```

- GUI:

Utilities \ Activity Monitor
View \ Send Signal to Process

© 2014 Software Diagnostics Services

Part 3: Practice Exercises

© 2014 Software Diagnostics Services

Links

- Memory Dumps:

NOT IN THE PUBLIC PREVIEW VERSION

- Exercise Transcripts:

NOT IN THE PUBLIC PREVIEW VERSION

© 2014 Software Diagnostics Services

Exercise 0 (GDB)

- **Goal:** Install Xcode and check if GDB loads a core dump correctly
- **Patterns:** Incorrect Stack Trace
- \AMCDA-Dumps\Exercise-A0-GDB.pdf

© 2014 Software Diagnostics Services

Exercise 0 (LLDB)

- **Goal:** Install Xcode and check if LLDB loads a core dump correctly
- **Patterns:** Incorrect Stack Trace
- \AMCDA-Dumps\Exercise-A0-LLDB.pdf

© 2014 Software Diagnostics Services

Process Core Dumps

Exercises A1-A12

© 2014 Software Diagnostics Services

Exercise A1 (GDB)

- **Goal:** Learn how to list stack traces, disassemble functions, check their correctness, dump data, compare core dumps with diagnostic reports, get environment
- **Patterns:** Manual Dump, Stack Trace, Stack Trace Collection, Annotated Disassembly, Paratext, Not My Version, Environment Hint
- \AMCDA-Dumps\Exercise-A1-GDB.pdf

© 2014 Software Diagnostics Services

Exercise A1 (LLDB)

- **Goal:** Learn how to list stack traces, disassemble functions, check their correctness, dump data, compare core dumps with diagnostic reports, get environment
- **Patterns:** Manual Dump, Stack Trace, Stack Trace Collection, Annotated Disassembly, Paratext, Not My Version, Environment Hint
- \AMCDA-Dumps\Exercise-A1-LLDB.pdf

© 2014 Software Diagnostics Services

Exercise A2 (GDB)

- **Goal:** Learn how to identify multiple exceptions, find problem CPU instructions
- **Patterns:** Multiple Exceptions, NULL Pointer (data), NULL Pointer (code)
- \AMCDA-Dumps\Exercise-A2-GDB.pdf

© 2014 Software Diagnostics Services

Exercise A2 (LLDB)

- **Goal:** Learn how to identify multiple exceptions, find problem CPU instructions
- **Patterns:** Multiple Exceptions, NULL Pointer (data), NULL Pointer (code)
- \AMCDA-Dumps\Exercise-A2-LLDB.pdf

© 2014 Software Diagnostics Services

Exercise A3 (GDB)

- **Goal:** Learn how to identify spiking threads
- **Patterns:** Spiking Thread
- \AMCDA-Dumps\Exercise-A3-GDB.pdf

© 2014 Software Diagnostics Services

Exercise A3 (LLDB)

- **Goal:** Learn how to identify spiking threads
- **Patterns:** Spiking Thread
- \AMCDA-Dumps\Exercise-A3-LLDB.pdf

© 2014 Software Diagnostics Services

Exercise A4 (GDB)

- **Goal:** Learn how to identify heap regions and heap corruption
- **Patterns:** Heap Corruption
- \AMCDA-Dumps\Exercise-A4-GDB.pdf

© 2014 Software Diagnostics Services

Exercise A4 (LLDB)

- **Goal:** Learn how to identify heap regions and heap corruption
- **Patterns:** Heap Corruption
- \AMCDA-Dumps\Exercise-A4-LLDB.pdf

© 2014 Software Diagnostics Services

Exercise A5 (GDB)

- **Goal:** Learn how to identify stack corruption
- **Patterns:** Local Buffer Overflow, Execution Residue
- \AMCDA-Dumps\Exercise-A5-GDB.pdf

© 2014 Software Diagnostics Services

Exercise A5 (LLDB)

- **Goal:** Learn how to identify stack corruption
- **Patterns:** Local Buffer Overflow, Execution Residue
- \AMCDA-Dumps\Exercise-A5-LLDB.pdf

© 2014 Software Diagnostics Services

Exercise A6 (GDB)

- **Goal:** Learn how to identify stack overflow, stack boundaries, reconstruct stack trace
- **Patterns:** Stack Overflow, Execution Residue
- \AMCDA-Dumps\Exercise-A6-GDB.pdf

© 2014 Software Diagnostics Services

Exercise A6 (LLDB)

- **Goal:** Learn how to identify stack overflow, stack boundaries, reconstruct stack trace
- **Patterns:** Stack Overflow, Execution Residue
- \AMCDA-Dumps\Exercise-A6-LLDB.pdf

© 2014 Software Diagnostics Services

Exercise A7 (GDB)

- **Goal:** Learn how to identify active threads
- **Patterns:** Divide by Zero, Active Thread
- \AMCDA-Dumps\Exercise-A7-GDB.pdf

© 2014 Software Diagnostics Services

Exercise A7 (LLDB)

- **Goal:** Learn how to identify active threads
- **Patterns:** Divide by Zero, Active Thread
- \AMCDA-Dumps\Exercise-A7-LLDB.pdf

© 2014 Software Diagnostics Services

Exercise A8 (GDB)

- **Goal:** Learn how to identify runtime exceptions, past execution residue and stack traces, identify handled exceptions
- **Patterns:** C++ Exception, Execution Residue, Coincidental Symbolic Information, Handled Exception
- \AMCDA-Dumps\Exercise-A8-GDB.pdf

© 2014 Software Diagnostics Services

Exercise A8 (LLDB)

- **Goal:** Learn how to identify runtime exceptions, past execution residue and stack traces, identify handled exceptions
- **Patterns:** C++ Exception, Execution Residue, Coincidental Symbolic Information, Handled Exception
- \AMCDA-Dumps\Exercise-A8-LLDB.pdf

© 2014 Software Diagnostics Services

Exercise A9 (GDB)

- **Goal:** Learn how to identify heap leaks
- **Patterns:** Heap Leak, Execution Residue, Module Hint
- \AMCDA-Dumps\Exercise-A9-GDB.pdf

© 2014 Software Diagnostics Services

Exercise A9 (LLDB)

- **Goal:** Learn how to identify heap leaks
- **Patterns:** Heap Leak, Execution Residue, Module Hint
- \AMCDA-Dumps\Exercise-A9-LLDB.pdf

© 2014 Software Diagnostics Services

Exercise A10 (GDB)

- **Goal:** Learn how to identify heap contention wait chains, synchronization issues, advanced disassembly, dump arrays
- **Patterns:** Double Free, Heap Contention, Wait Chain, Critical Region, Self-Diagnosis
- \AMCDA-Dumps\Exercise-A10-GDB.pdf

© 2014 Software Diagnostics Services

Exercise A10 (LLDB)

- **Goal:** Learn how to identify heap contention wait chains, synchronization issues, advanced disassembly, dump arrays
- **Patterns:** Double Free, Heap Contention, Wait Chain, Critical Region, Self-Diagnosis
- \AMCDA-Dumps\Exercise-A10-LLDB.pdf

© 2014 Software Diagnostics Services

Exercise A11 (GDB)

- **Goal:** Learn how to identify synchronization wait chains, deadlocks, hidden and handled exceptions
- **Patterns:** Wait Chains, Deadlock, Execution Residue, Handled Exception
- \AMCDA-Dumps\Exercise-A11-GDB.pdf

© 2014 Software Diagnostics Services

Exercise A11 (LLDB)

- **Goal:** Learn how to identify synchronization wait chains, deadlocks, hidden and handled exceptions
- **Patterns:** Wait Chains, Deadlock, Execution Residue, Handled Exception
- \AMCDA-Dumps\Exercise-A11-LLDB.pdf

© 2014 Software Diagnostics Services

Exercise A12 (GDB)

- **Goal:** Learn how to dump memory for post-processing, get the list of functions and module variables, load symbols, inspect arguments and local variables
- **Patterns:** Module Variable
- \AMCDA-Dumps\Exercise-A12-GDB.pdf

© 2014 Software Diagnostics Services

Exercise A12 (LLDB)

- **Goal:** Learn how to dump memory for post-processing, get the list of functions and module variables, load symbols, inspect arguments and local variables
- **Patterns:** Module Variable
- \AMCDA-Dumps\Exercise-A12-LLDB.pdf

© 2014 Software Diagnostics Services

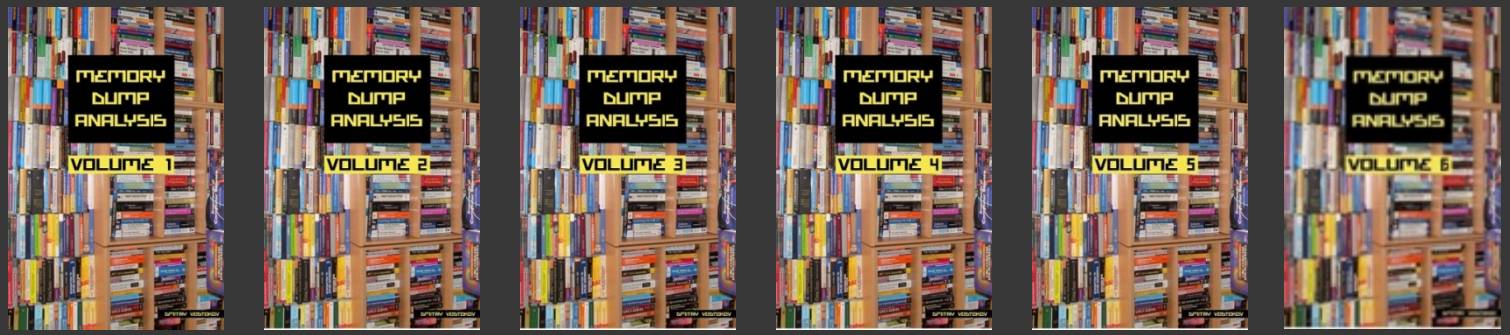
Pattern Links

Active Thread
C++ Exception
Critical Region
Divide by Zero
Environment Hint
Incorrect Stack Trace
Heap Contention
Heap Leak
Manual Dump
Module Variable
Not My Version
NULL Pointer (code)
Self-Diagnosis
Stack Overflow
Stack Trace Collection
Annotated Disassembly
Coincidental Symbolic Information
Deadlock
Double Free
Execution Residue
Handled Exception
Heap Corruption
Local Buffer Overflow
Module Hint
Multiple Exceptions
NULL Pointer (data)
Paratext
Spiking Thread
Stack Trace
Wait Chain

© 2014 Software Diagnostics Services

Resources

- Software Diagnostics Institute
- Pattern-Driven Software Diagnostics
- Pattern-Based Software Diagnostics
- Debugging TV
- Rosetta Stone for Debuggers
- GDB Pocket Reference
- GDB -> LLDB Map
- Memory Dump Analysis Anthology (volume 7 covers Mac OS X)



© 2014 Software Diagnostics Services

Q&A

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

© 2014 Software Diagnostics Services

Thank you for attendance!

© 2014 Software Diagnostics Services