Linux
Core Dump Analysis
Accelerated

Dmitry Vostokov
Software Diagnostics Services

Published by OpenTask, Republic of Ireland

Copyright © 2015 by OpenTask

Copyright © 2015 by Software Diagnostics Services

Copyright © 2015 by Dmitry Vostokov

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, without the prior written permission of the publisher.

You must not circulate this book in any other binding or cover, and you must impose the same condition on any acquirer.

Product and company names mentioned in this book may be trademarks of their owners.

OpenTask books and magazines are available through booksellers and distributors worldwide. For further information or comments send requests to press@opentask.com.

A CIP catalog record for this book is available from the British Library.

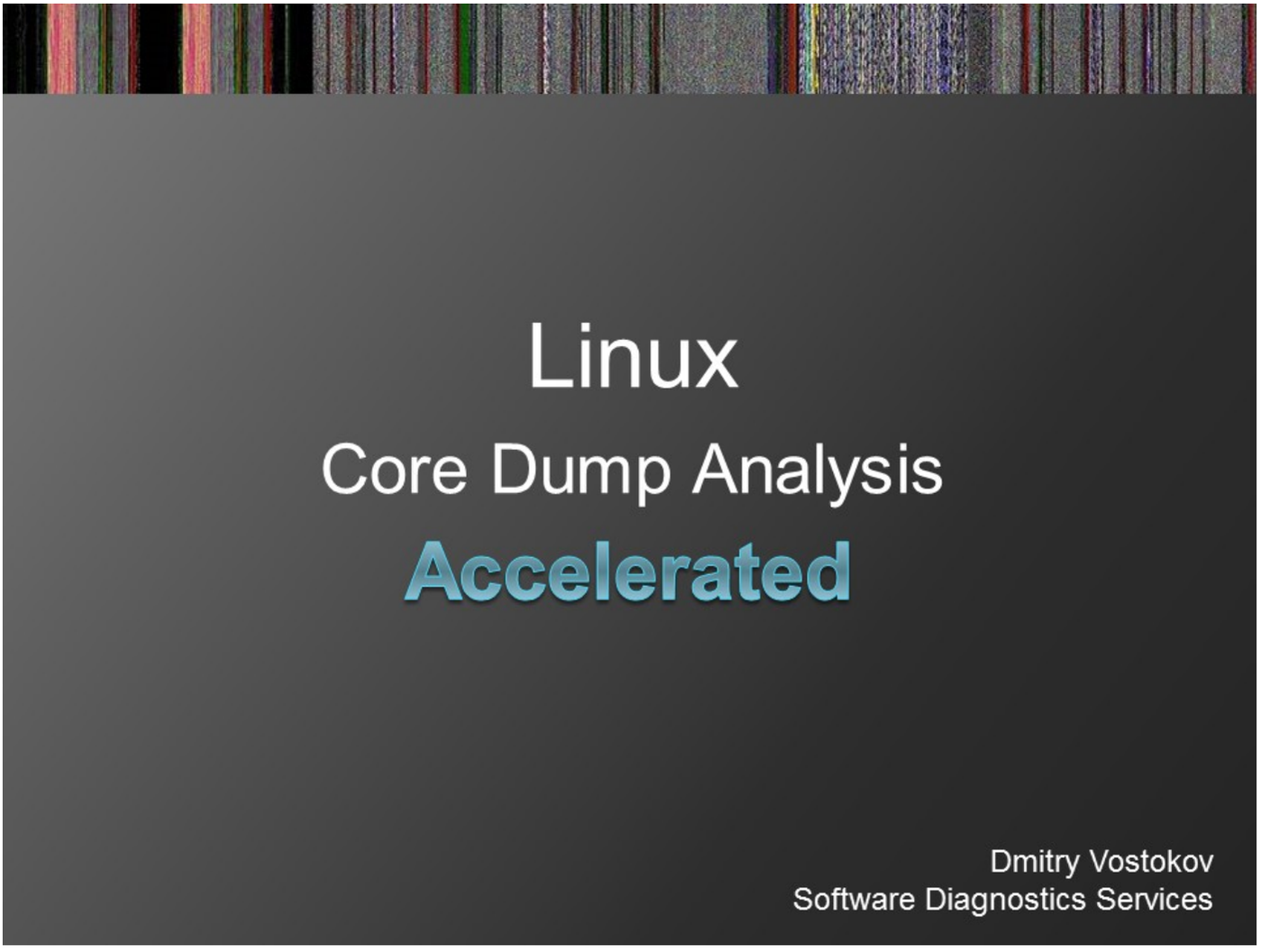
ISBN-l3: 978-1-908043-97-9 (Paperback)

1st printing, 2015

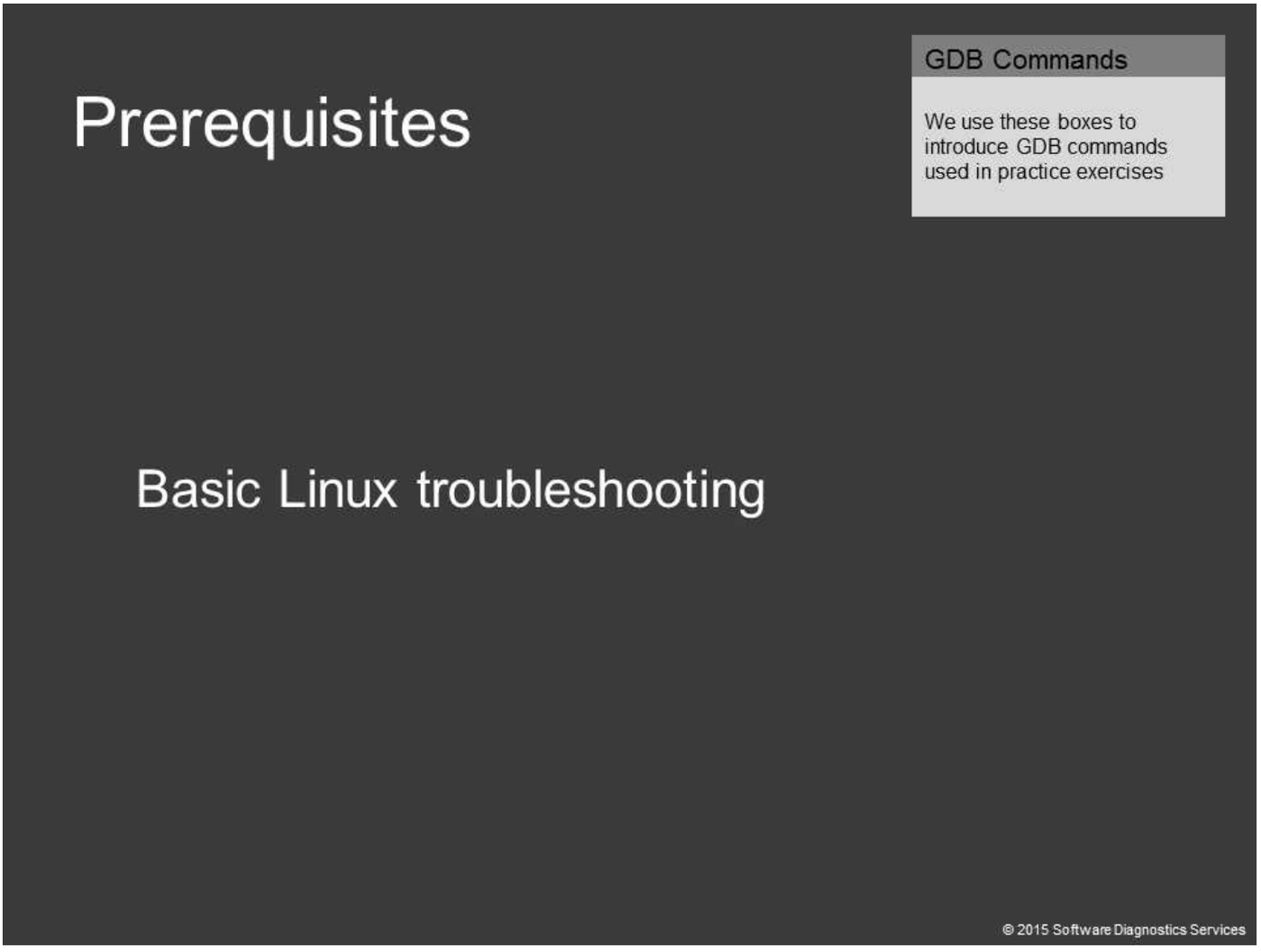
Revision 1.01 (February, 2016)

Contents

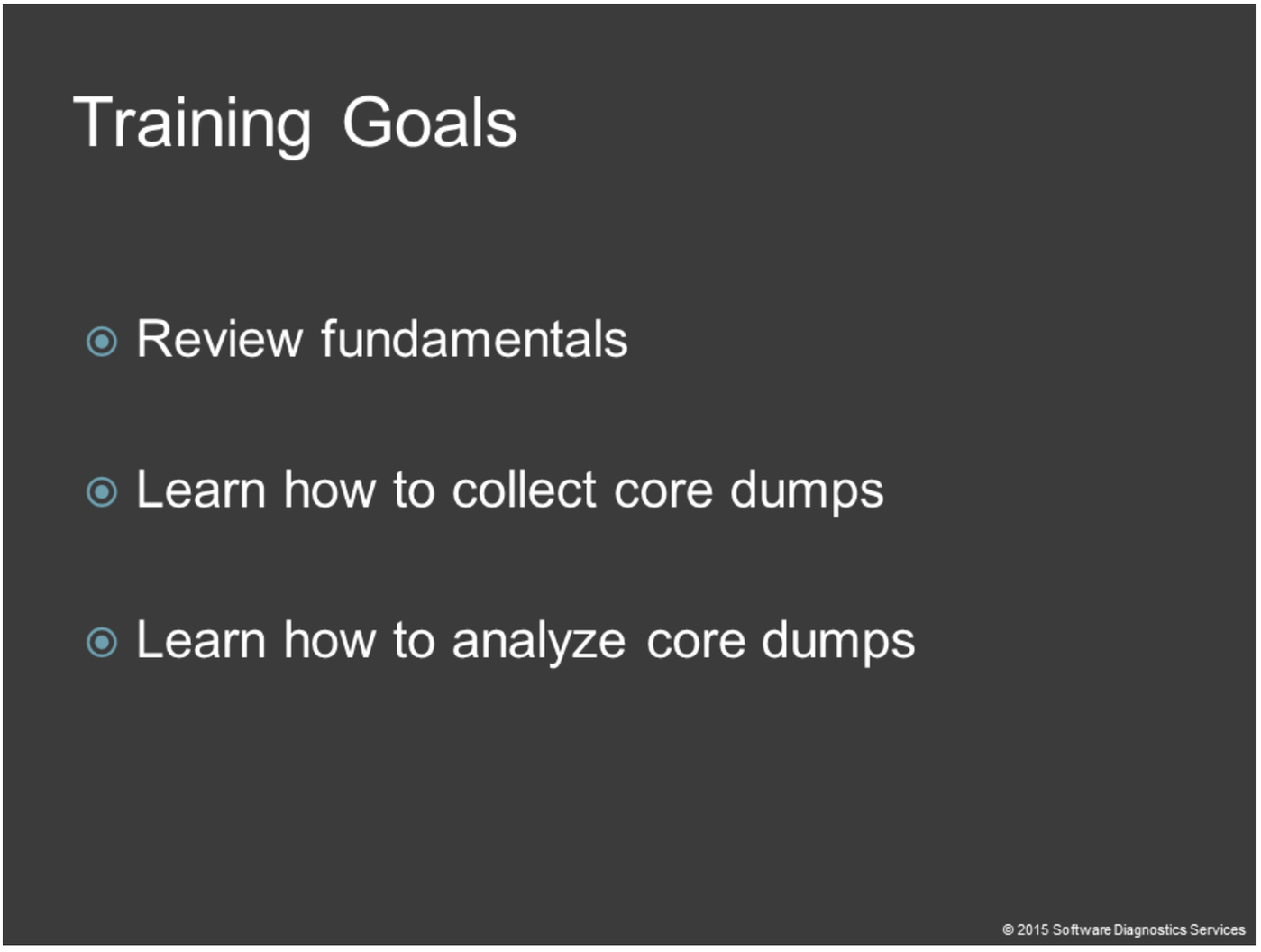
Presentation Slides and Transcript



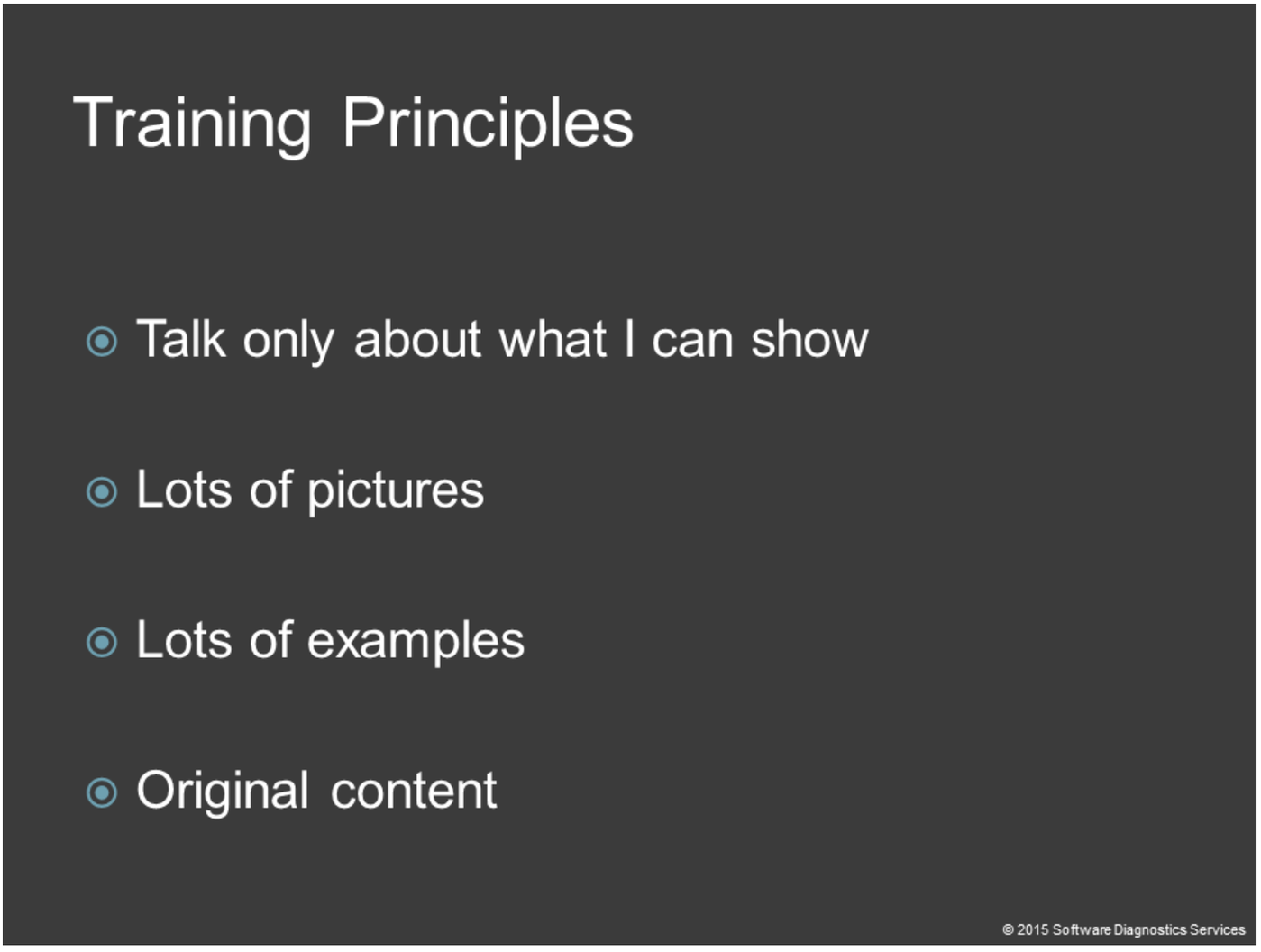
Hello, everyone, my name is Dmitry Vostokov, and I teach this training course.



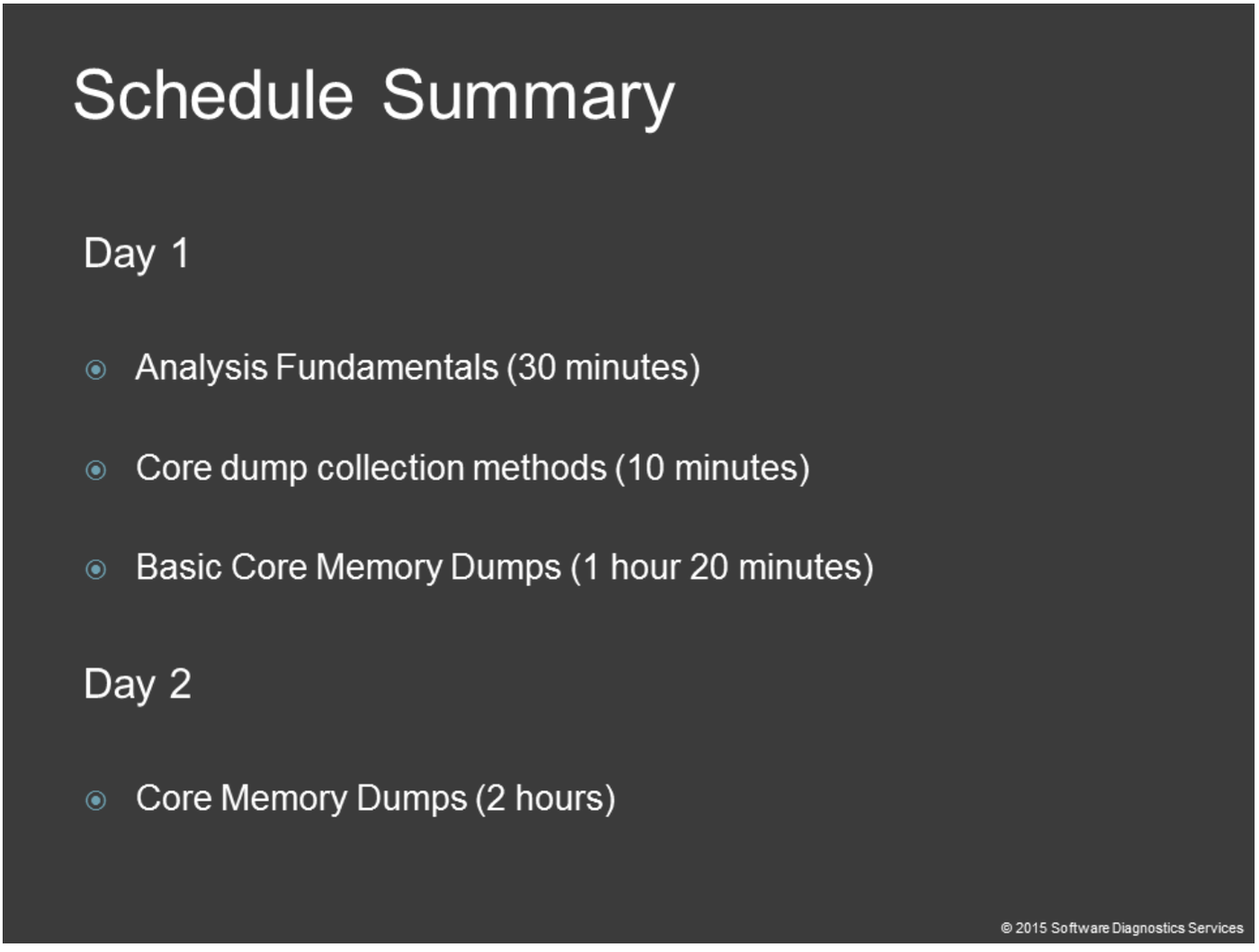
The prerequisites are hard to define. Some of you have software development experience and some not. However, one thing is certain that to get most of this training you are expected to have basic troubleshooting experience. Another thing I expect you to be familiar with is hexadecimal notation and that you have seen or can read programming source code in some language. The ability to read assembly language has some advantages but not necessary for this training. Windows memory dump analysis experience may really help here and ease the transition but not absolutely necessary. If you have read either **Accelerated Mac OS X Core Dump Analysis** or **Accelerated Windows Memory Dump Analysis** book or both, you may find the similar approach here.



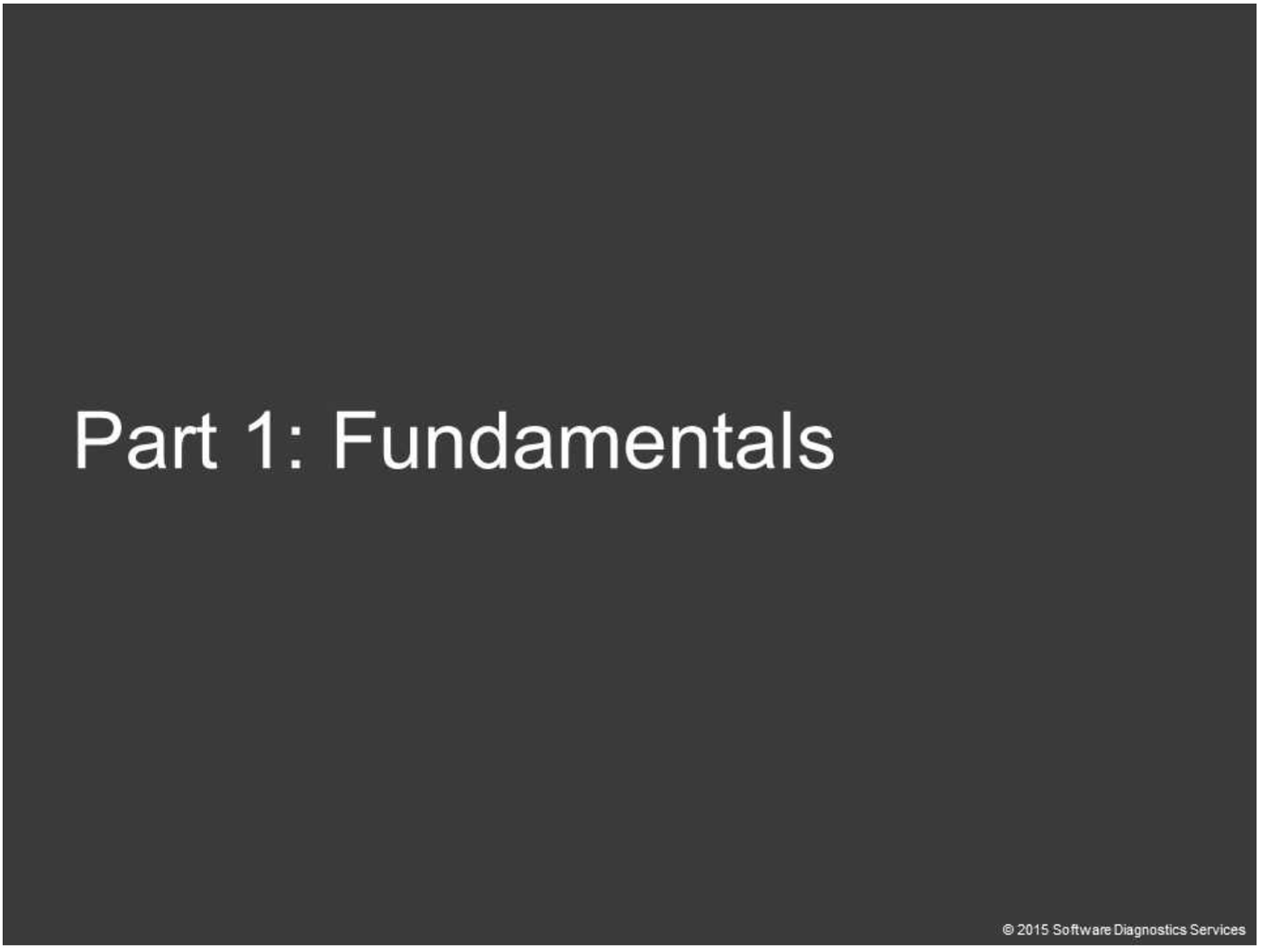
Our primary goal is to learn core dump analysis in an accelerated fashion. So first we review absolutely essential fundamentals necessary for core dump analysis. Also, this training is about user process core dump analysis and not about kernel core dump analysis. An additional goal is to leverage Windows or Mac OS X debugging, and memory dump analysis experience you may have.



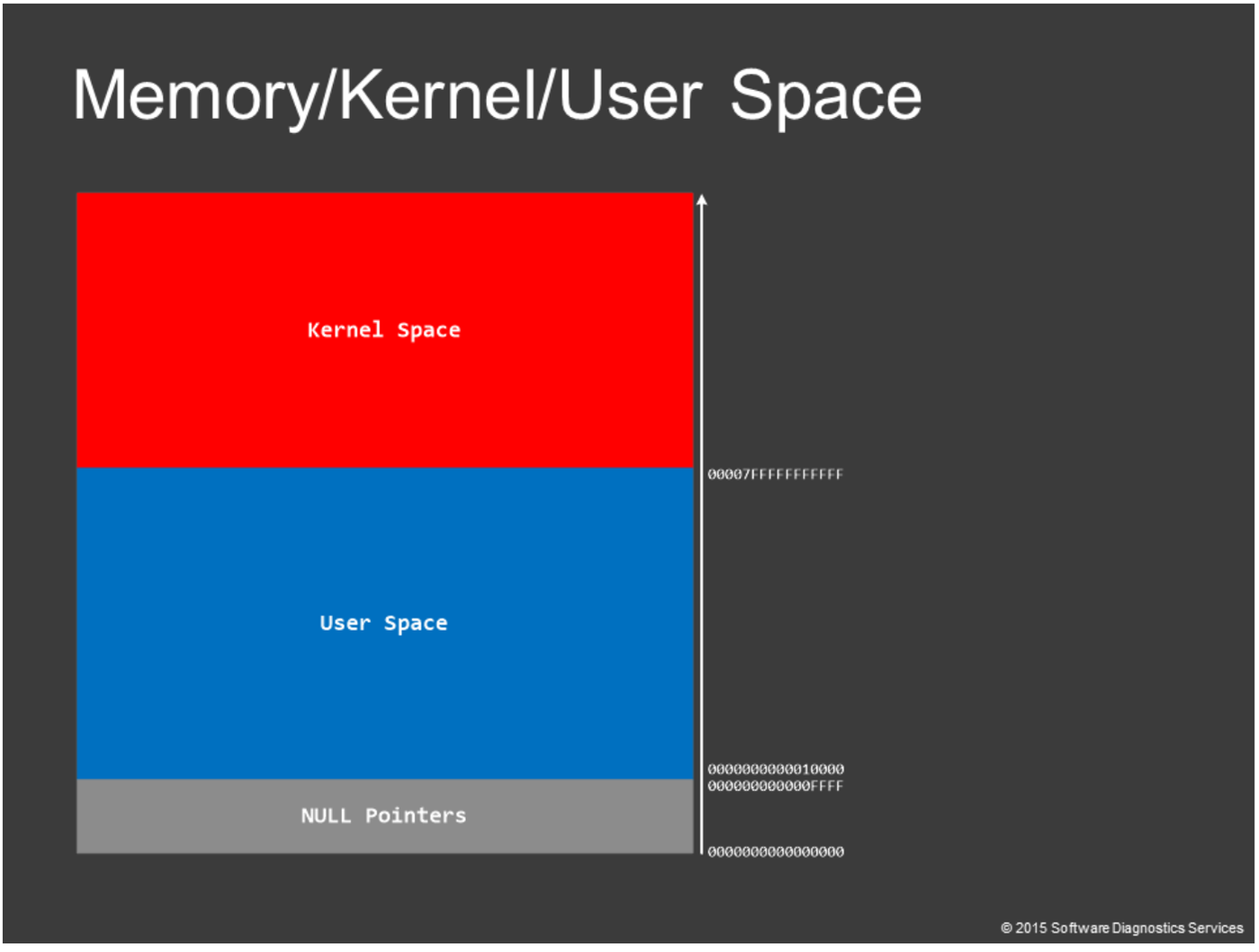
For me, there were many training formats to consider, and I decided that the best way is to concentrate on hands-on exercises. Specifically, for this training, I developed 13 of them, and they utilize the same pattern-driven approach I used in **Accelerated Windows Memory Dump Analysis** and **Accelerated Mac OS X Core Dump Analysis** training.



This is a roughly planned schedule.

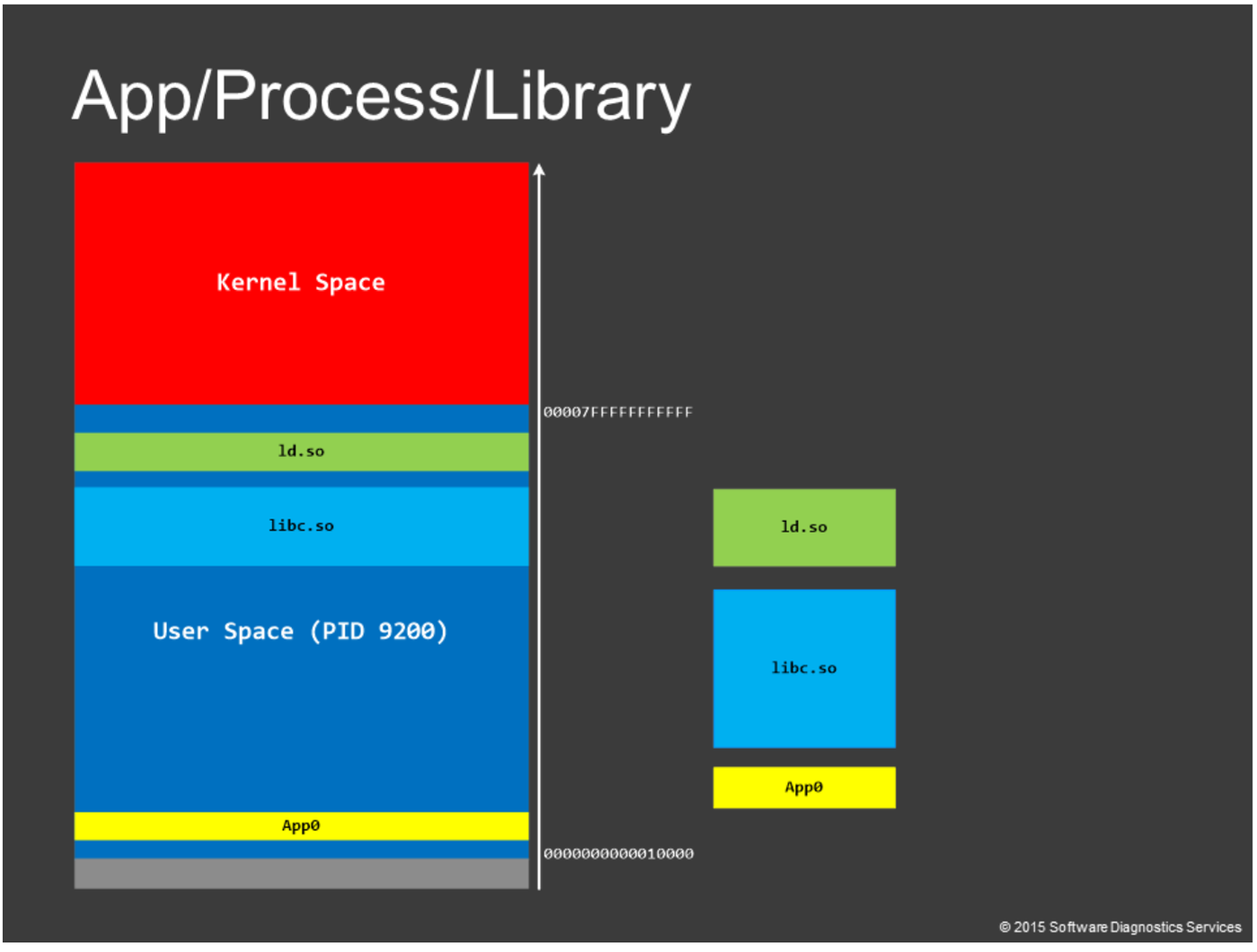


Now I present you some pictures. I use 64-bit examples. Most of the time fundamentals do not change when we move to 32-bit Linux and the analysis process most of the time is the same.

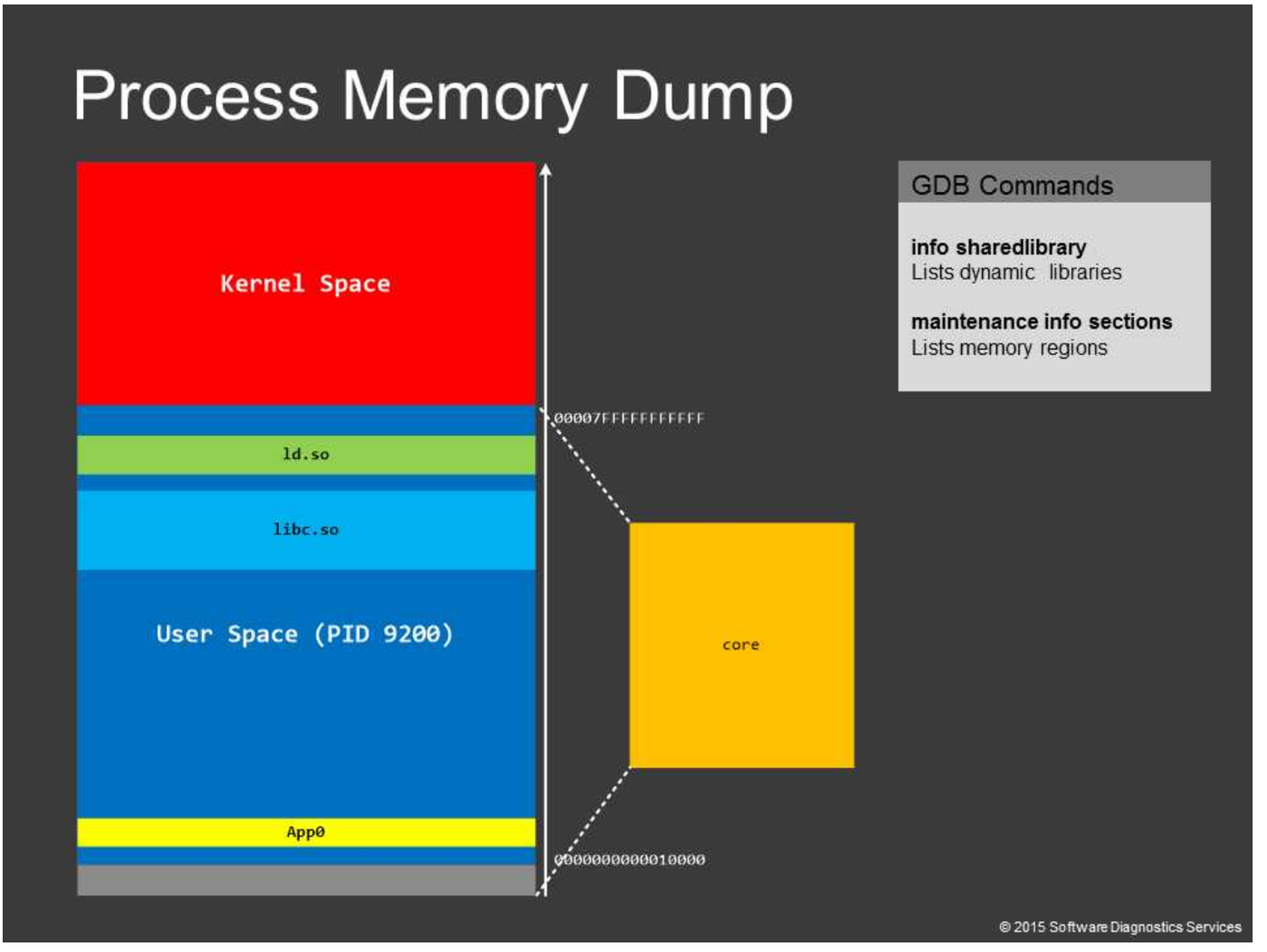


If you are coming from Windows or Mac OS X background, you find fundamentals almost the same. For every process Linux memory range is divided into kernel space part, user space part and an inaccessible part to catch null pointers[1]. This non-accessible region is different from Mac OS X where it is 1 GB. I follow the long tradition to use red color for kernel and blue color for user part. Please note that there is a difference between space and mode. The mode is execution privilege attribute, for example, code running in kernel space has higher execution privilege than code running in user space. However, kernel code can access user space and access data there. We say that such code is running in kernel mode. On the contrary, the application code from user space is running in user mode and because of its lower privilege, it cannot access kernel space. This prevents accidental kernel modifications. Otherwise, you could easily crash your system. I put addresses on the right. This uniform memory space is called process virtual space because it is an abstraction that allows us to analyze core dumps without thinking about how it is all organized in physical memory. When we look at process dumps, we are concerned with virtual space only. In this training, we will only see user space.

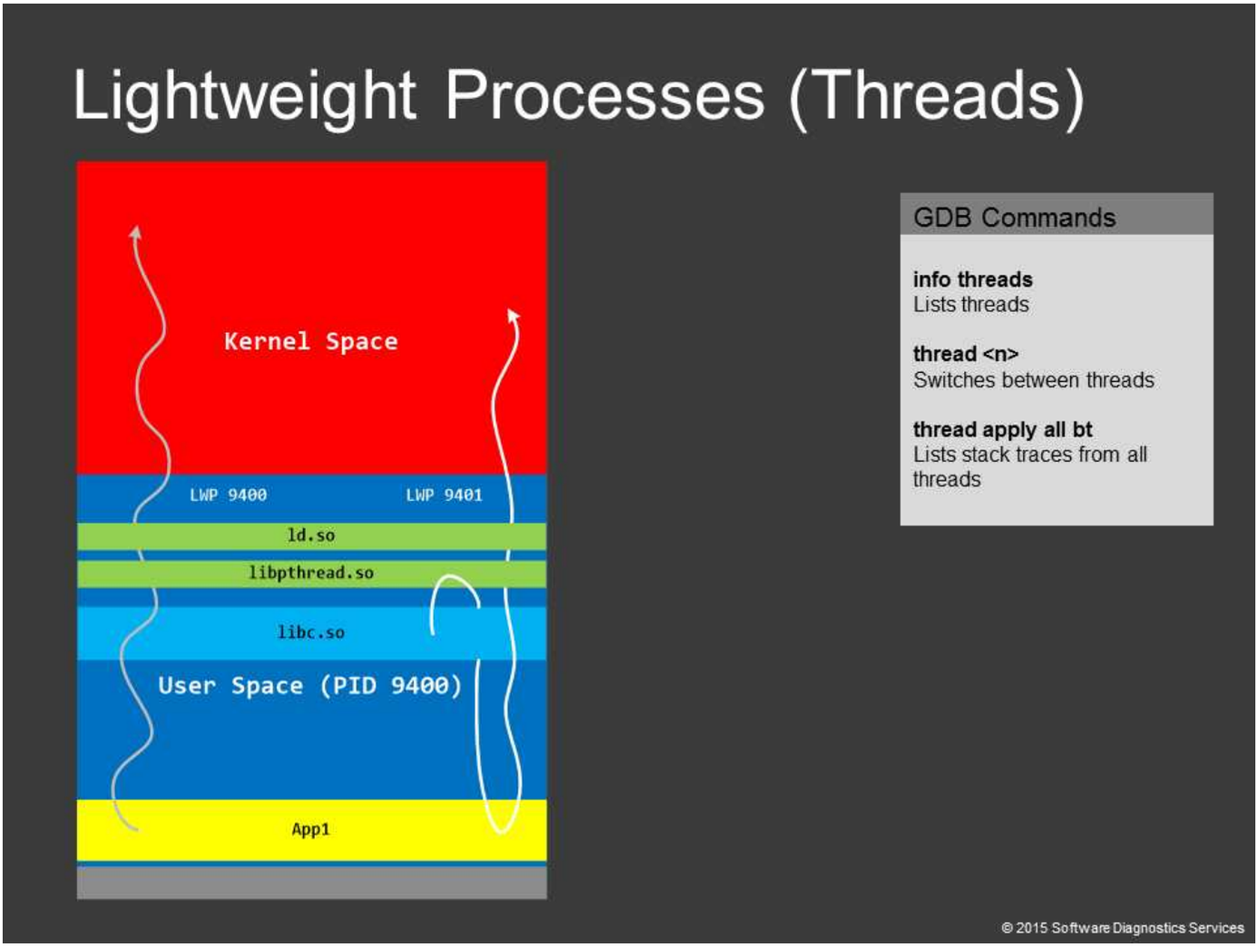
[1] On my Debian64 system it is 0xFFFF, as seen from /proc/sys/vm/mmap_min_addr value.



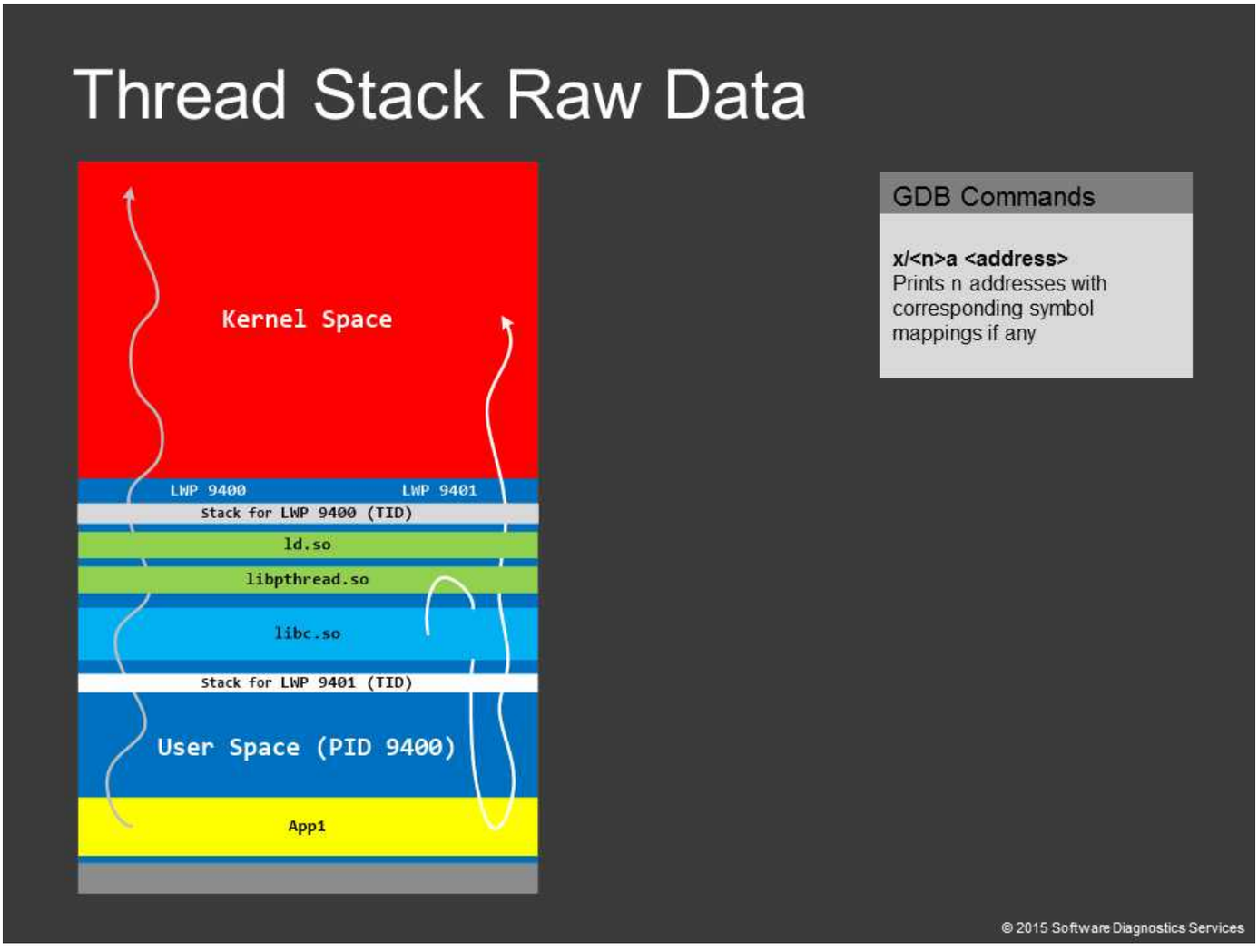
When an app is loaded, all its referenced dynamic libraries are mapped to virtual memory space. Different sections of the same file (like code and data) may be mapped into a different portion of memory. In contrast, modules in Windows are organized sequentially in virtual memory space. A process then is setup for running, and a process ID is assigned to it. If you run another such app, it will have the different virtual memory space.



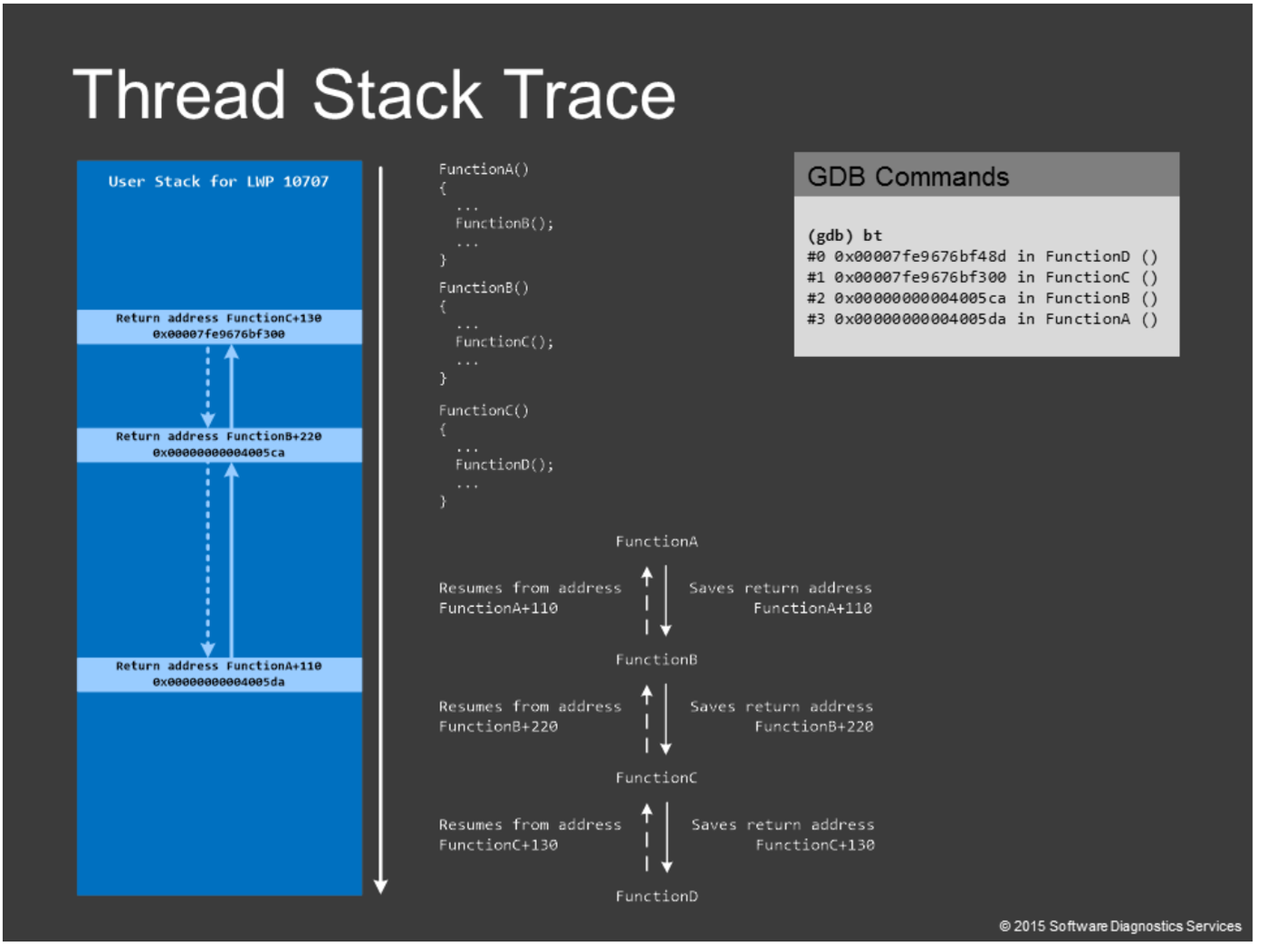
When we save a process core memory dump a user space portion of the process space is saved without any kernel space stuff. However, we never see such large core dumps unless we have memory leaks. This is because process space has gaps unfilled with code and data. These unallocated parts are not saved in a core dump. However, if some parts were paged out and reside in a page file, they are usually brought back before saving a core dump.



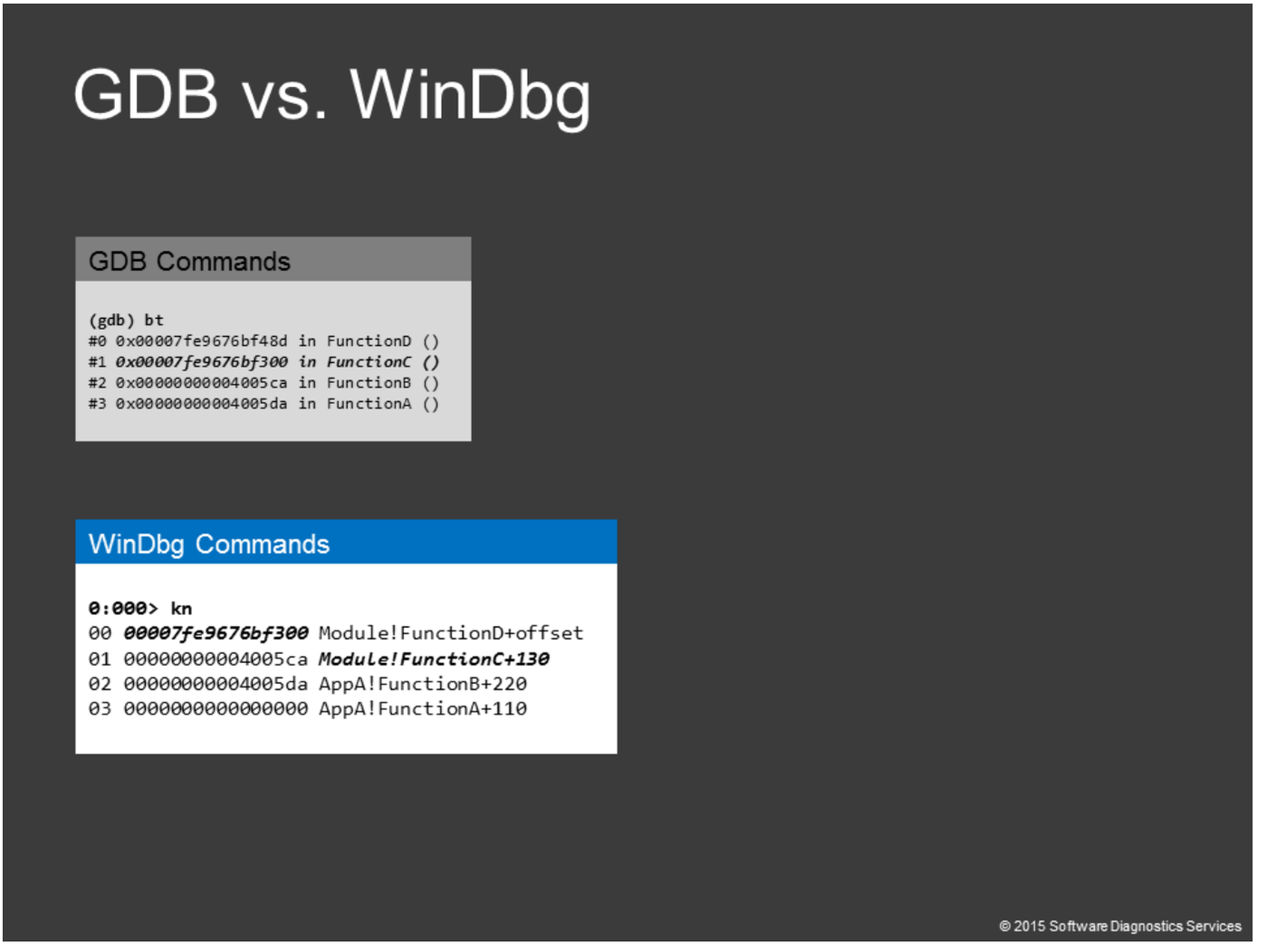
Now we come to another important fundamental concept in Linux core dump analysis: a thread or lightweight process (LWP). It is basically a unit of execution, and there can be many threads (LWPs) for a given process. Every thread just executes some code and performs various tasks. Every thread has its ID (LWP ID). In this training, we also learn how to navigate between process threads. Note that threads transition to kernel space via *libc* dynamic library similar to *ntdll* on Windows and *libsystem_kernel* in Mac OS X. Threads additional to the main thread (POSIX Threads) originate from *libc* and *libpthread* dynamic libraries similar to *libsystem_c* in Mac OS X.



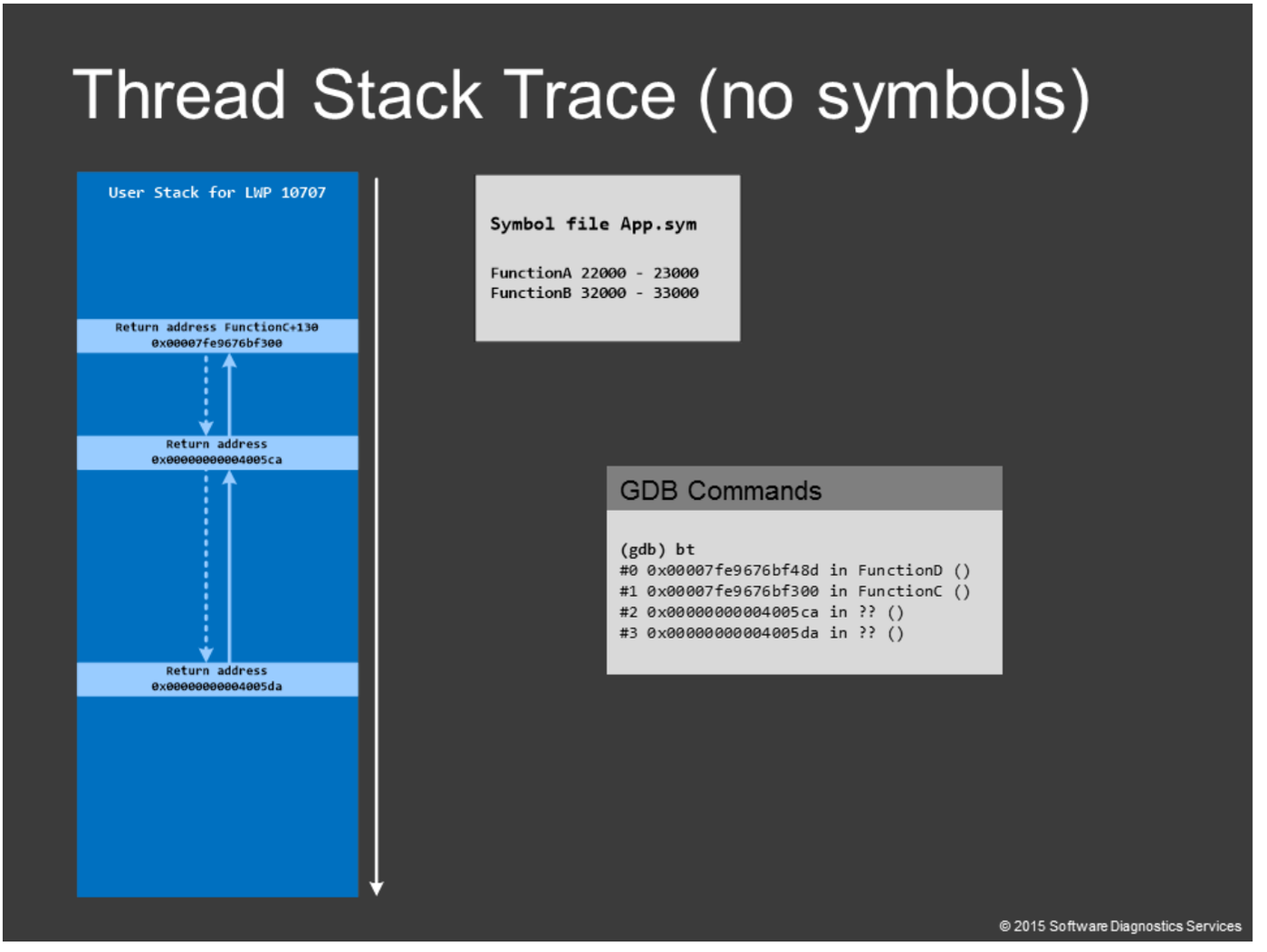
Every thread needs a temporary memory region to store its execution history and temporary data. This region is called a thread stack. Please note that the stack region is just any other memory region, and you can use any GDB data dumping commands there. We will also learn how to get thread stack region address range. Examining raw stack data can give some hints to the past app behavior: the so-called **Execution Residue** pattern.



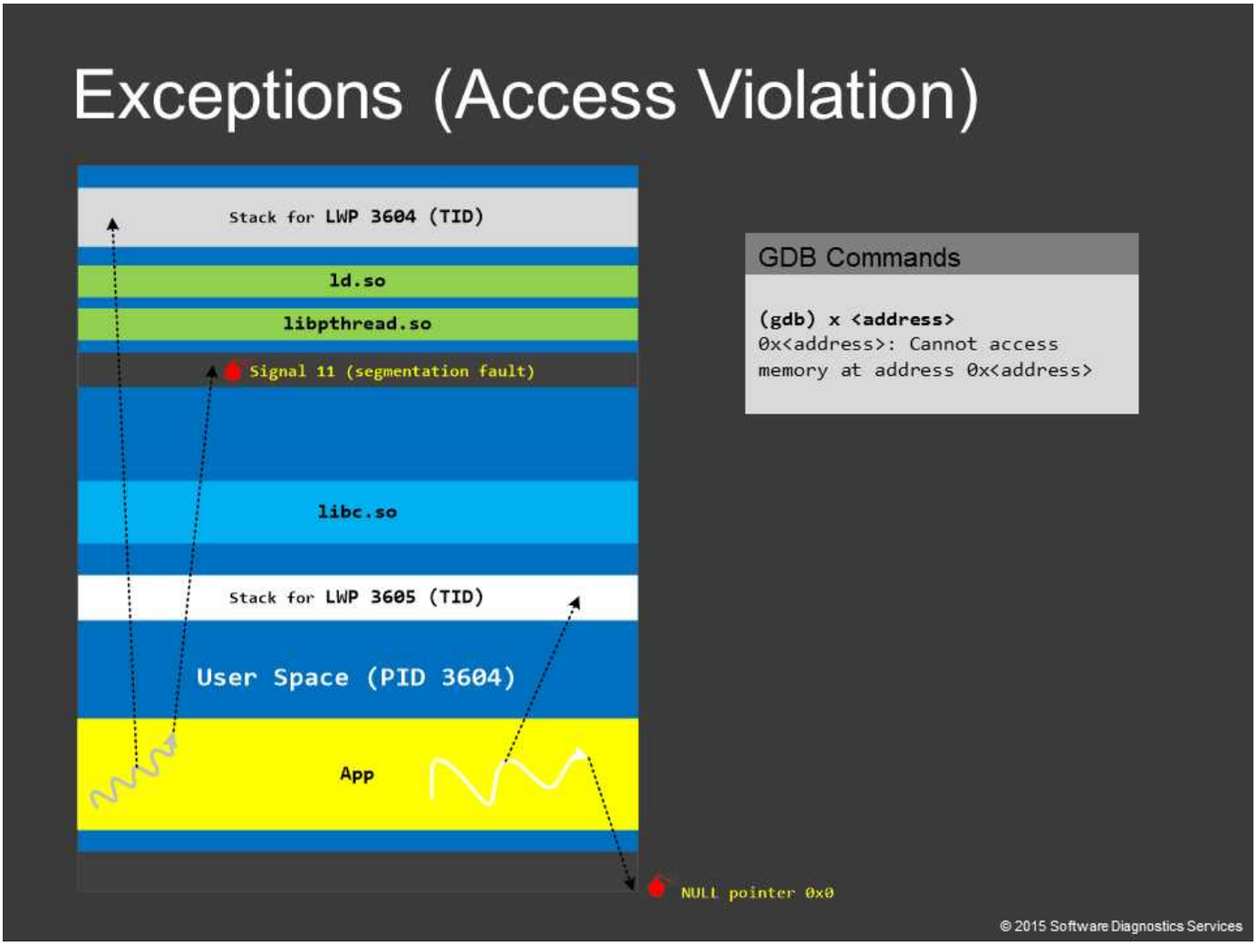
Now we explain thread stack traces. Suppose we have source code where FunctionA calls FunctionB at some point and FunctionB calls FunctionC and so on. This is a thread of execution. If FunctionA calls FunctionB, you expect the execution thread to return to the same place where it left, and to resume from there. This is achieved by saving a return address in the thread stack region. So every return address is saved and then restored during the course of a thread execution. Although the memory addresses grow from top to bottom on this picture return addresses are saved from bottom to top.This might seem counter-intuitive to all previous pictures, but this is how you see the output from GDB commands. What GDB does when you instruct it to dump a backtrace from a given thread is to analyze the thread raw stack data and figure out return addresses, map them to a symbolic form according to symbol files and show them from top to bottom. Note that FunctionD is not present in the raw stack data on the left because it is a currently executing function called from FunctionC. However, FunctionC called FunctionD, and the return address of FunctionC was saved. In the box on the right, we see the result of GDB **bt** command.



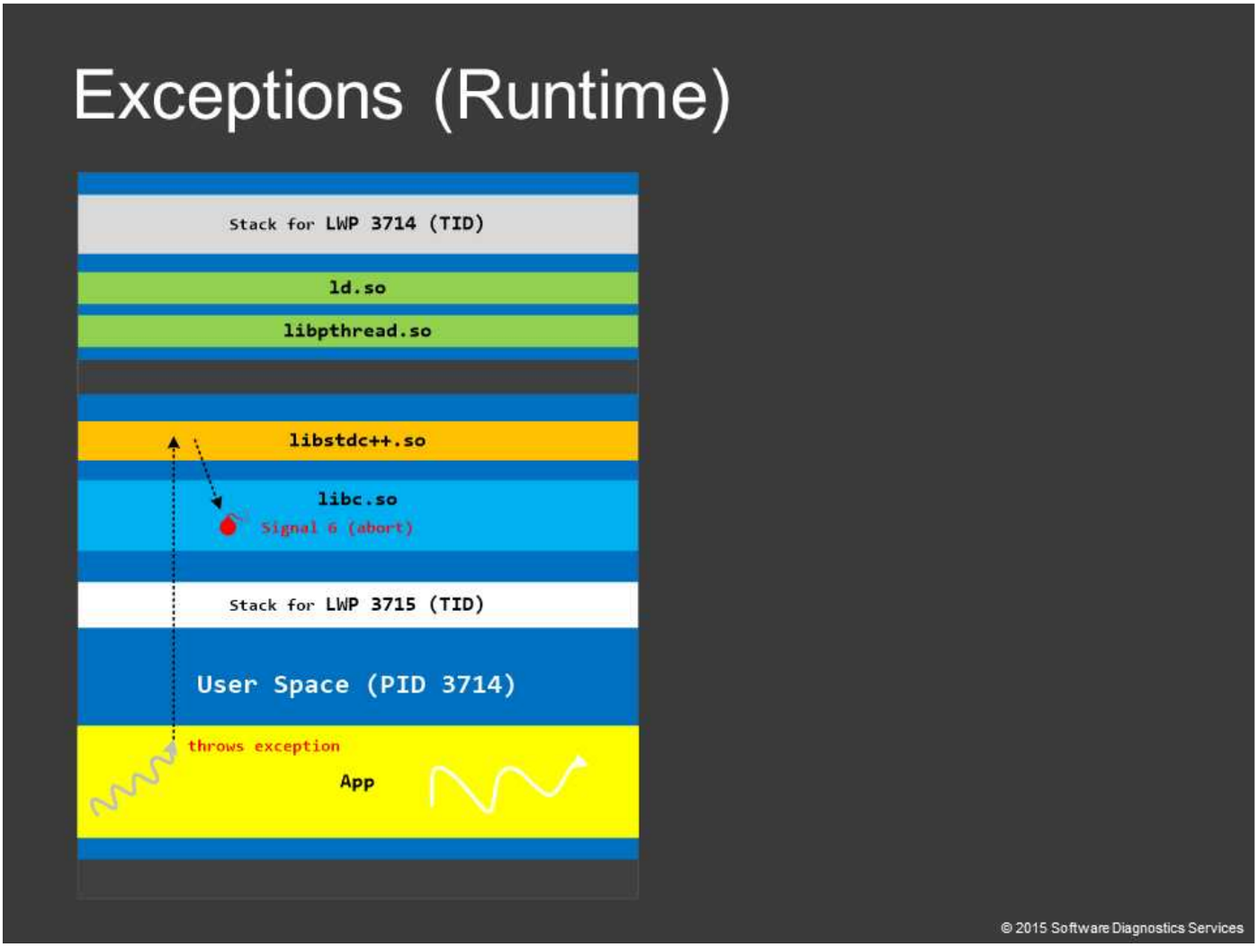
The difference from WinDbg (from Debugging Tools for Windows) here is that the return address is on the same line for the function to return (except for FunctionD, where the address is the next instruction to execute) whereas in WinDbg it is for the function on the next line.



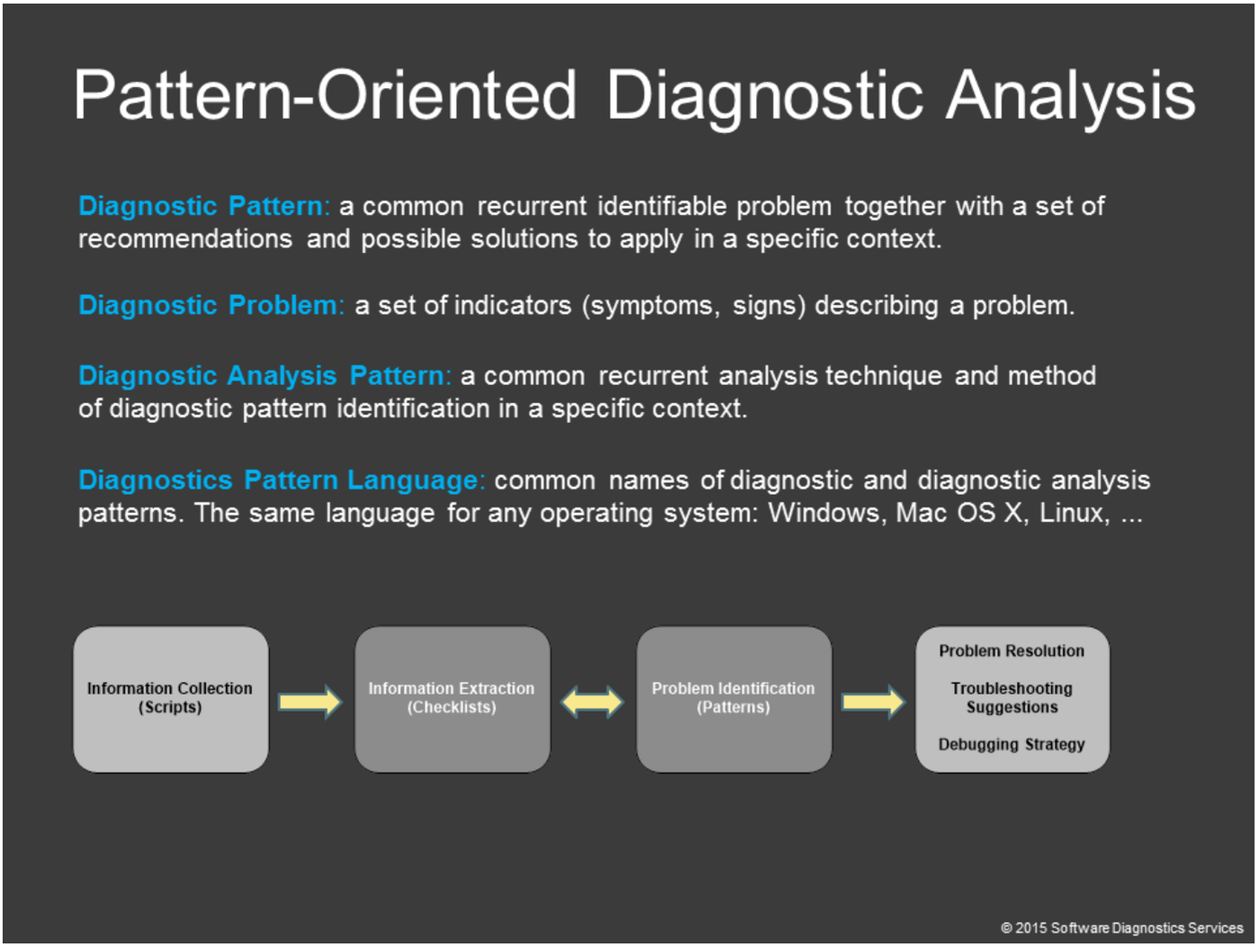
Here I'd like to show you why symbol files are important and what stack traces you get without them. Symbol files just provide mappings between memory address ranges and associated symbol names like the table of contents in a book. So in the absence of symbols, we are left with bare addresses that are saved in a dump. For example, without App symbols, we have the output shown in the box on the right.



Now we talk about access violation exceptions. During the thread execution, it accesses various memory addresses doing reads and writes. Sometimes memory is not present due to gaps in virtual address space or different protection levels like read-only or no-execute memory regions. If a thread tries to violate that, we get an exception that is also translated to a traditional UNIX signal. Certain regions are forbidden to read and write such as the first 64KB. If we have such an access violation there, then it is called a NULL pointer access. Note that any thread can have an exception (a victim thread in Mac OS X). It is also sometimes the case that code can catch these exceptions preventing a user from seeing error messages. Such exceptions can contribute to corruption, and we call then hidden.

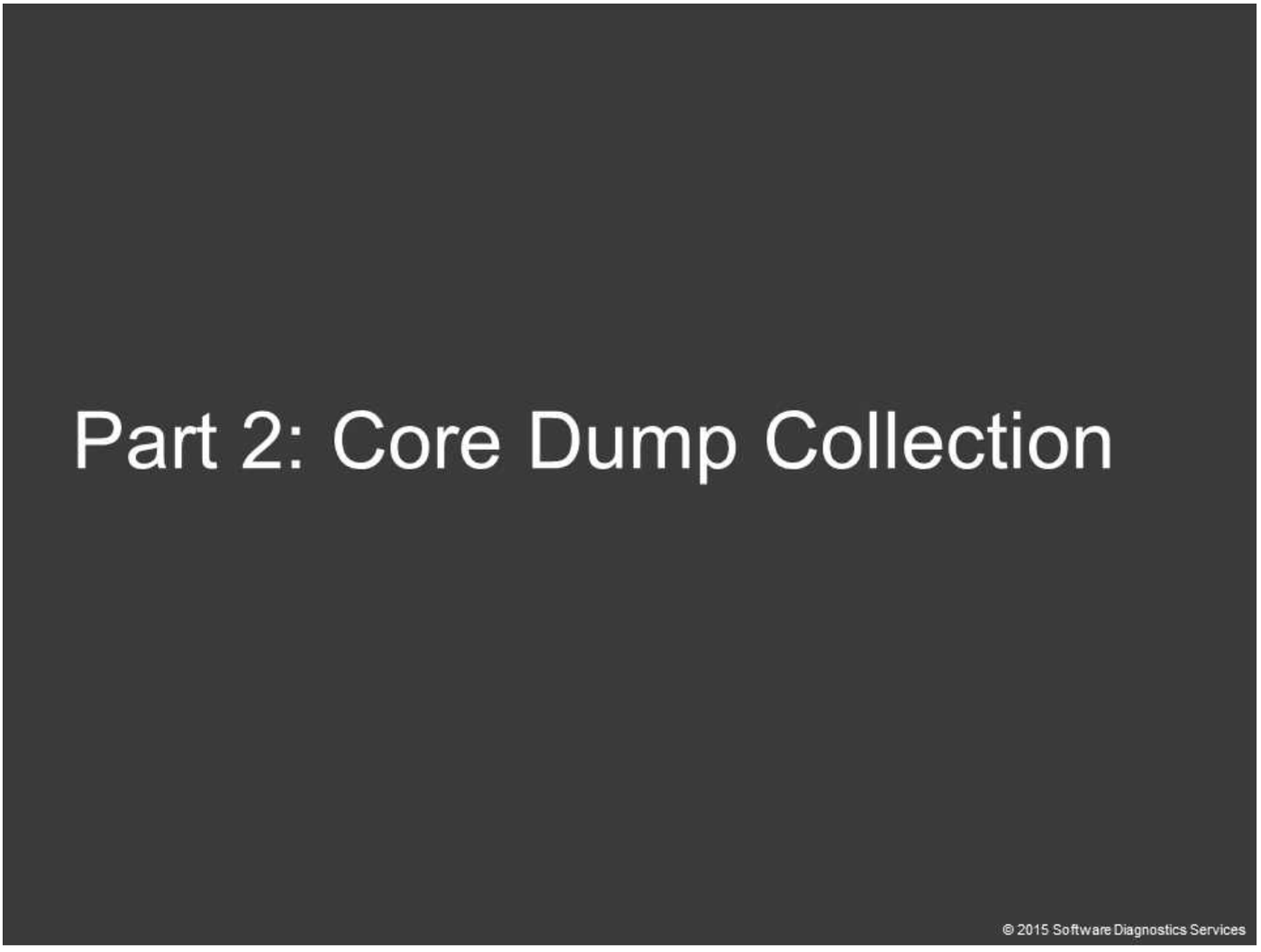


However, not all exceptions happen from invalid access. Many exceptions are generated by the code itself when it checks for some condition, and it is not satisfied, for example, when code checks a buffer or an array to verify whether it is full before trying to add more data. If it finds it is already full, the code throws an exception translated to SIGABRT. We would see that in one of our practice examples when C++ code throws a C++ exception. Such exceptions are usually called runtime exceptions.



A few words about logs, checklists, and patterns. Core memory dump analysis is usually an analysis of a text for the presence of diagnostic patterns. We run commands; they output text, and then we look at that textual output and when we find suspicious diagnostic indicators we execute more commands. Here pattern and command checklists can be very useful.

Core Dump Collection



Here I'd like to show you how to collect core dumps because by default this option is switched off on Linux.

Enabling Collection

- Temporary for the current user:

```
$ ulimit -c unlimited
```

- Permanent for every user except root:

Edit the file: `/etc/security/limits.conf`

Add or uncomment the line:

```
*      soft   core   unlimited
```

To limit root to 1GB add or uncomment this line:

```
root   hard   core   1000000
```

© 2015 Software Diagnostics Services

Usually, a process core dump is stored in the working directory of the process.

Generation Methods

- kill (requires ulimit):

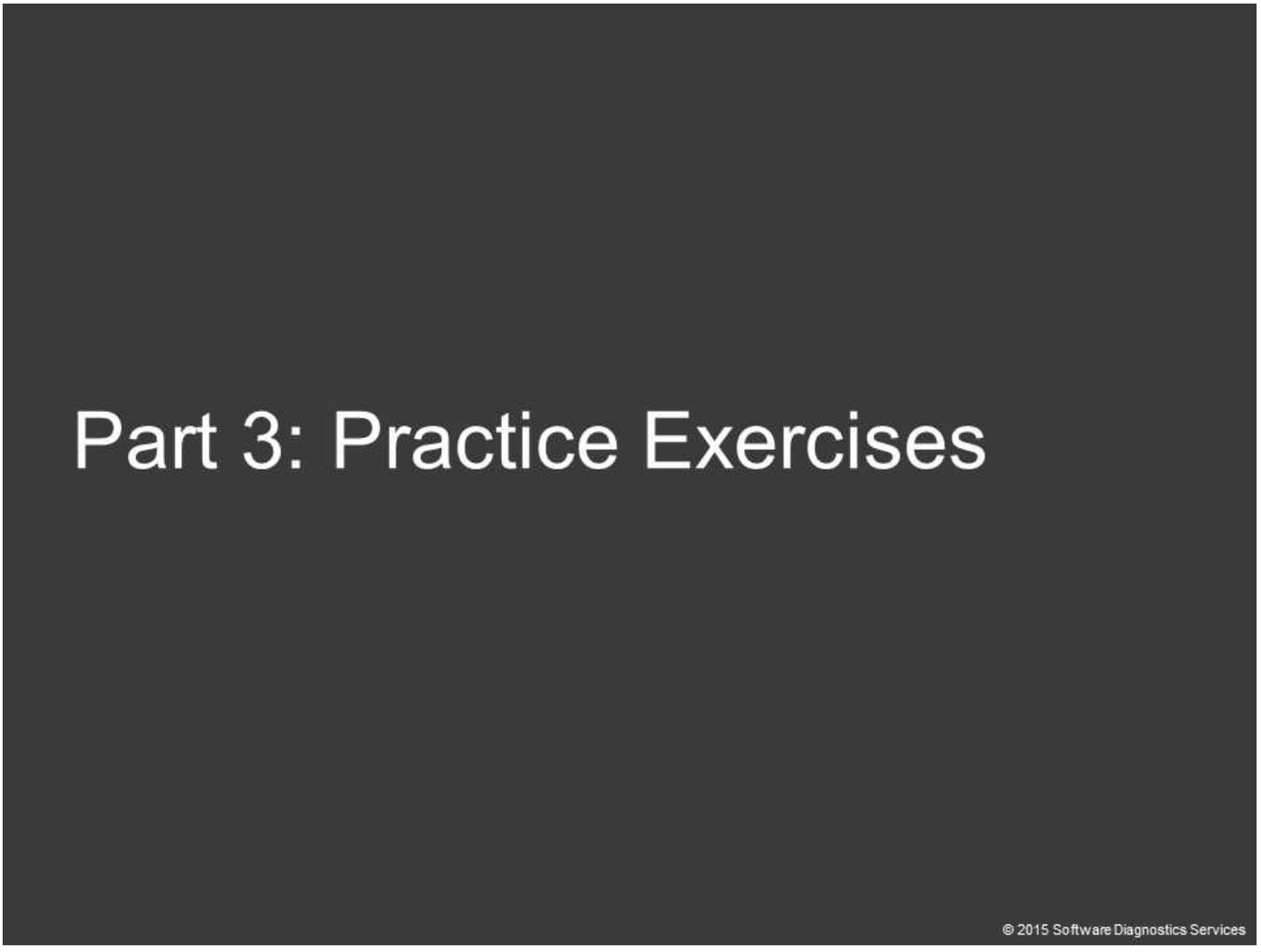
```
$ kill -s SIGQUIT PID
$ kill -s SIGABRT PID
```

- gcore:

```
$ gcore PID
```

© 2015 Software Diagnostics Services

Practice Exercises



Now we come to practice. The goal is to show you important commands and how their output helps in recognizing patterns of abnormal software behavior.

Links

- Memory Dumps:

Links are the below on this page

- Exercise Transcripts:

Included in this book

© 2015 Software Diagnostics Services

http://www.patterndiagnostics.com/Training/ALCDA/ALCDA-Dumps.tar.gz

Exercise 0

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

© 2015 Software Diagnostics Services

Exercise 0

Goal: Install GDB and check if GDB loads a core dump correctly.

Patterns: Incorrect Stack Trace.

1. Download and install the latest version of GDB. For Debian64 we used the following command in root mode:

```
root@debian64# apt-get install gdb
```

2. Verify that GDB is accessible and then exit it (**q** command):

```
training@debian64:~/ALCDA$ gdb

GNU gdb (GDB) 7.4.1-debian
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.

(gdb) q
training@debian64:~/ALCDA$
```

3. Load a core dump and *App0* executable:

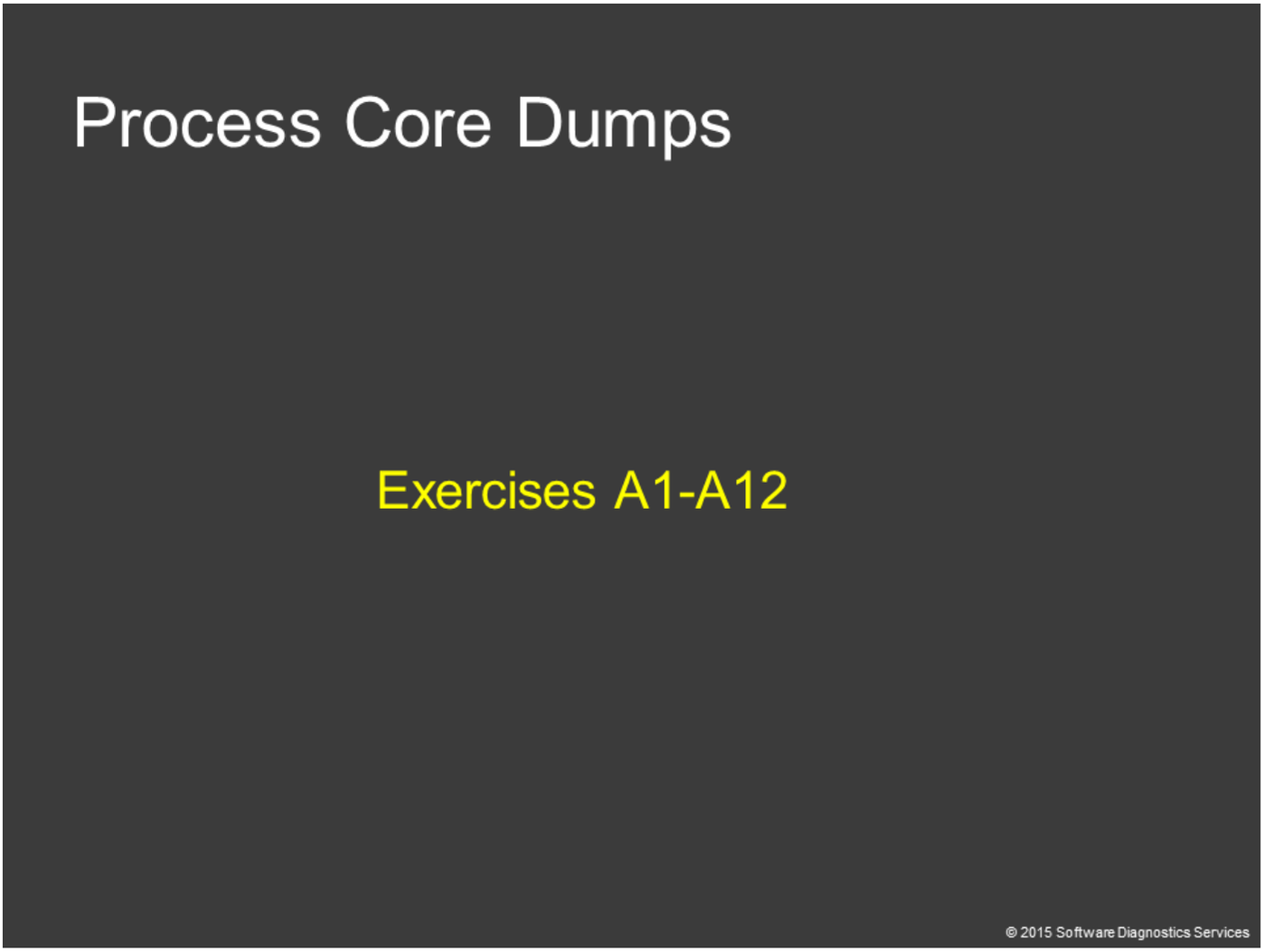
```
training@debian64:~/ALCDA$ gdb -c ./App0/core -se ./App0/App0

GNU gdb (GDB) 7.4.1-debian
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/training/ALCDA/App0/App0...(no debugging symbols found)...done.
[New LWP 3142]
Core was generated by `./App0'.
Program terminated with signal 6, Aborted.
#0  0x000000000041b145 in raise ()
```

4. Verify that the stack trace (back trace) is shown correctly with symbols:

```
(gdb) bt
#0  0x000000000041b145 in raise ()
#1  0x0000000000400eb0 in abort ()
#2  0x00000000004004d9 in bar ()
#3  0x00000000004004e7 in foo ()
#4  0x0000000000400502 in main ()
```

5. To avoid possible confusion and glitches we recommend to exit GDB after each exercise.

```
(gdb) q

training@debian64:~/ALCDA$
```



All exercises were modeled on real-life examples using specially constructed applications. We will learn how to recognize and use more than 30 analysis patterns.

Exercise A1

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

© 2015 Software Diagnostics Services

Exercise A1

Goal: Learn how to list stack traces, disassemble functions, check their correctness, dump data, get environment.

Patterns: Manual Dump, Stack Trace, Stack Trace Collection, Annotated Disassembly, Paratext, Not My Version, Environment Hint.

1. Load a core dump *core.3308* and *App1* executable:

```
training@debian64:~/ALCDA$ gdb -c ./App1/core.3308 -se ./App1/App1
GNU gdb (GDB) 7.4.1-debian
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/training/ALCDA/App1/App1...done.
[New LWP 3309]
[New LWP 3310]
[New LWP 3311]
[New LWP 3312]
[New LWP 3313]
[New LWP 3308]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Core was generated by `/home/training/ALCDA/App1/App1'.
#0  0x000000000042fdf1 in nanosleep ()
```

2. List all threads:

```
(gdb) info threads
  Id   Target Id         Frame
  6    LWP 3308          0x000000000042fdf1 in nanosleep ()
  5    LWP 3313          0x000000000042fdf1 in nanosleep ()
  4    LWP 3312          0x000000000042fdf1 in nanosleep ()
  3    LWP 3311          0x000000000042fdf1 in nanosleep ()
  2    LWP 3310          0x000000000042fdf1 in nanosleep ()
* 1    LWP 3309          0x000000000042fdf1 in nanosleep ()
```

3. Get all thread stack traces:

```
(gdb) thread apply all bt

Thread 6 (LWP 3308):
#0  0x000000000042fdf1 in nanosleep ()
#1  0x000000000042fcc0 in sleep ()
#2  0x00000000004006c1 in main ()
```

```
Thread 5 (LWP 3313):
#0  0x000000000042fdf1 in nanosleep ()
#1  0x000000000042fcc0 in sleep ()
#2  0x00000000004005f2 in bar_five ()
#3  0x0000000000400602 in foo_five ()
#4  0x000000000040061a in thread_five ()
#5  0x00000000004015f0 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#6  0x00000000004324a9 in clone ()
#7  0x0000000000000000 in ?? ()

Thread 4 (LWP 3312):
#0  0x000000000042fdf1 in nanosleep ()
#1  0x000000000042fcc0 in sleep ()
#2  0x00000000004005b5 in bar_four ()
#3  0x00000000004005c5 in foo_four ()
#4  0x00000000004005dd in thread_four ()
#5  0x00000000004015f0 in start_thread (arg=<optimized out>)
---Type <return> to continue, or q <return> to quit---
    at pthread_create.c:304
#6  0x00000000004324a9 in clone ()
#7  0x0000000000000000 in ?? ()

Thread 3 (LWP 3311):
#0  0x000000000042fdf1 in nanosleep ()
#1  0x000000000042fcc0 in sleep ()
#2  0x0000000000400578 in bar_three ()
#3  0x0000000000400588 in foo_three ()
#4  0x00000000004005a0 in thread_three ()
#5  0x00000000004015f0 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#6  0x00000000004324a9 in clone ()
#7  0x0000000000000000 in ?? ()

Thread 2 (LWP 3310):
#0  0x000000000042fdf1 in nanosleep ()
#1  0x000000000042fcc0 in sleep ()
#2  0x000000000040053b in bar_two ()
#3  0x000000000040054b in foo_two ()
#4  0x0000000000400563 in thread_two ()
#5  0x00000000004015f0 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#6  0x00000000004324a9 in clone ()
---Type <return> to continue, or q <return> to quit---
#7  0x0000000000000000 in ?? ()

Thread 1 (LWP 3309):
#0  0x000000000042fdf1 in nanosleep ()
#1  0x000000000042fcc0 in sleep ()
#2  0x00000000004004fe in bar_one ()
#3  0x000000000040050e in foo_one ()
#4  0x0000000000400526 in thread_one ()
#5  0x00000000004015f0 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#6  0x00000000004324a9 in clone ()
#7  0x0000000000000000 in ?? ()
```

4. Switch to the thread #2 and get its stack trace:

```
(gdb) thread 2
[Switching to thread 2 (LWP 3310)]
#0  0x000000000042fdf1 in nanosleep ()

(gdb) bt
#0  0x000000000042fdf1 in nanosleep ()
#1  0x000000000042fcc0 in sleep ()
#2  0x000000000040053b in bar_two ()
#3  0x000000000040054b in foo_two ()
#4  0x0000000000400563 in thread_two ()
#5  0x00000000004015f0 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#6  0x00000000004324a9 in clone ()
#7  0x0000000000000000 in ?? ()
```

5. Check that *bar_two* called *sleep* function:

```
(gdb) disassemble bar_two
Dump of assembler code for function bar_two:
   0x000000000040052d <+0>:     push   %rbp
   0x000000000040052e <+1>:     mov    %rsp,%rbp
   0x0000000000400531 <+4>:     mov    $0xffffffff,%edi
   0x0000000000400536 <+9>:     callq  0x42fbe0 <sleep>
   0x000000000040053b <+14>:    pop    %rbp
   0x000000000040053c <+15>:    retq
End of assembler dump.
```

We see that the address in the stack trace for *bar_two* function is the address to return to after calling *sleep* function.

6. Compare with Intel disassembly flavor:

```
(gdb) set disassembly-flavor intel

(gdb) disassemble bar_two
Dump of assembler code for function bar_two:
   0x000000000040052d <+0>:     push   rbp
   0x000000000040052e <+1>:     mov    rbp,rsp
   0x0000000000400531 <+4>:     mov    edi,0xffffffff
   0x0000000000400536 <+9>:     call   0x42fbe0 <sleep>
   0x000000000040053b <+14>:    pop    rbp
   0x000000000040053c <+15>:    ret
End of assembler dump.

(gdb) set disassembly-flavor att
```

7. Get App1 data section from the output of pmap (*pmap.3308*):

```
3308:   ./App1
0000000000400000    732K r-x--  /home/training/ALCDA/App1/App1
00000000006b6000      8K rw---  /home/training/ALCDA/App1/App1
00000000006b8000     28K rw---    [ anon ]
000000000227c000    140K rw---    [ anon ]
00007f2257e66000      4K -----    [ anon ]
00007f2257e67000   8192K rw---    [ anon ]
00007f2258667000      4K -----    [ anon ]
00007f2258668000   8192K rw---    [ anon ]
00007f2258e68000      4K -----    [ anon ]
00007f2258e69000   8192K rw---    [ anon ]
00007f2259669000      4K -----    [ anon ]
00007f225966a000   8192K rw---    [ anon ]
00007f2259e6a000      4K -----    [ anon ]
00007f2259e6b000   8192K rw---    [ anon ]
00007ffc7d24d000    132K rw---    [ stack ]
00007ffc7d299000      4K r-x--    [ anon ]
ffffffffff600000      4K r-x--    [ anon ]
 total            42028K
```

8. Compare with the section information in the core dump:

```
(gdb) maintenance info sections
Exec file:
    `/home/training/ALCDA/App1/App1', file type elf64-x86-64.
    0x00400158->0x00400178 at 0x00000158: .note.ABI-tag ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x00400178->0x0040019c at 0x00000178: .note.gnu.build-id ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004001a0->0x004002d8 at 0x000001a0: .rela.plt ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004002d8->0x004002e6 at 0x000002d8: .init ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x004002f0->0x004003c0 at 0x000002f0: .plt ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x004003c0->0x0048b1b8 at 0x000003c0: .text ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x0048b1c0->0x0048bd3e at 0x0008b1c0: __libc_freeres_fn ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x0048bd40->0x0048bda1 at 0x0008bd40: __libc_thread_freeres_fn ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x0048bda4->0x0048bdad at 0x0008bda4: .fini ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x0048bdc0->0x004a9d24 at 0x0008bdc0: .rodata ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004a9d28->0x004a9d88 at 0x000a9d28: __libc_subfreeres ALLOC LOAD READONLY DATA HAS_CONTENTS
---Type <return> to continue, or q <return> to quit---
    0x004a9d88->0x004a9d90 at 0x000a9d88: __libc_atexit ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004a9d90->0x004a9d98 at 0x000a9d90: __libc_thread_subfreeres ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004a9d98->0x004b686c at 0x000a9d98: .eh_frame ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004b686c->0x004b6986 at 0x000b686c: .gcc_except_table ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x006b6988->0x006b69b0 at 0x000b6988: .tdata ALLOC LOAD DATA HAS_CONTENTS
    0x006b69b0->0x006b69e0 at 0x000b69b0: .tbss ALLOC
    0x006b69b0->0x006b69c0 at 0x000b69b0: .init_array ALLOC LOAD DATA HAS_CONTENTS
    0x006b69c0->0x006b69d0 at 0x000b69c0: .fini_array ALLOC LOAD DATA HAS_CONTENTS
    0x006b69d0->0x006b69d8 at 0x000b69d0: .jcr ALLOC LOAD DATA HAS_CONTENTS
    0x006b69e0->0x006b6a50 at 0x000b69e0: .data.rel.ro ALLOC LOAD DATA HAS_CONTENTS
    0x006b6a50->0x006b6a60 at 0x000b6a50: .got ALLOC LOAD DATA HAS_CONTENTS
    0x006b6a60->0x006b6ae0 at 0x000b6a60: .got.plt ALLOC LOAD DATA HAS_CONTENTS
    0x006b6ae0->0x006b77f0 at 0x000b6ae0: .data ALLOC LOAD DATA HAS_CONTENTS
    0x006b7800->0x006beb68 at 0x000b77f0: .bss ALLOC
    0x006beb68->0x006beb98 at 0x000b77f0: __libc_freeres_ptrs ALLOC
    0x00000000->0x00000038 at 0x000b77f0: .comment READONLY HAS_CONTENTS
    0x00000000->0x00000390 at 0x000b7830: .debug_aranges READONLY HAS_CONTENTS
---Type <return> to continue, or q <return> to quit---
    0x00000000->0x00000ac3 at 0x000b7bc0: .debug_pubnames READONLY HAS_CONTENTS
    0x00000000->0x00011440 at 0x000b8683: .debug_info READONLY HAS_CONTENTS
    0x00000000->0x000021b1 at 0x000c9ac3: .debug_abbrev READONLY HAS_CONTENTS
    0x00000000->0x00002ebc at 0x000cbc74: .debug_line READONLY HAS_CONTENTS
    0x00000000->0x000038da at 0x000ceb30: .debug_str READONLY HAS_CONTENTS
    0x00000000->0x0000878e at 0x000d240a: .debug_loc READONLY HAS_CONTENTS
    0x00000000->0x00001280 at 0x000dab98: .debug_ranges READONLY HAS_CONTENTS
```

```
Core file:
    `/home/training/ALCDA/./App1/core.3308', file type elf64-x86-64.
    0x00000000->0x00002aa8 at 0x00000318: note0 READONLY HAS_CONTENTS
    0x00000000->0x000000d8 at 0x00000438: .reg/3309 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x00000438: .reg HAS_CONTENTS
    0x00000000->0x00000200 at 0x0000052c: .reg2/3309 HAS_CONTENTS
    0x00000000->0x00000200 at 0x0000052c: .reg2 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00000740: .reg-xstate/3309 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00000740: .reg-xstate HAS_CONTENTS
    0x00000000->0x000000d8 at 0x00000b04: .reg/3310 HAS_CONTENTS
    0x00000000->0x00000200 at 0x00000bf8: .reg2/3310 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00000e0c: .reg-xstate/3310 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x000011d0: .reg/3311 HAS_CONTENTS
    0x00000000->0x00000200 at 0x000012c4: .reg2/3311 HAS_CONTENTS
    0x00000000->0x00000340 at 0x000014d8: .reg-xstate/3311 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x0000189c: .reg/3312 HAS_CONTENTS
    0x00000000->0x00000200 at 0x00001990: .reg2/3312 HAS_CONTENTS
---Type <return> to continue, or q <return> to quit---
    0x00000000->0x00000340 at 0x00001ba4: .reg-xstate/3312 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x00001f68: .reg/3313 HAS_CONTENTS
    0x00000000->0x00000200 at 0x0000205c: .reg2/3313 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00002270: .reg-xstate/3313 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x00002634: .reg/3308 HAS_CONTENTS
    0x00000000->0x00000200 at 0x00002728: .reg2/3308 HAS_CONTENTS
    0x00000000->0x00000340 at 0x0000293c: .reg-xstate/3308 HAS_CONTENTS
    0x00000000->0x00000130 at 0x00002c90: .auxv HAS_CONTENTS
    0x00400000->0x00400000 at 0x00002dc0: load1 ALLOC READONLY CODE
    0x006b6000->0x006b8000 at 0x00002dc0: load2 ALLOC LOAD HAS_CONTENTS
    0x006b8000->0x006bf000 at 0x00004dc0: load3 ALLOC LOAD HAS_CONTENTS
    0x0227c000->0x0229f000 at 0x0000bdc0: load4 ALLOC LOAD HAS_CONTENTS
    0x7f2257e67000->0x7f2258667000 at 0x0002edc0: load5 ALLOC LOAD HAS_CONTENTS
    0x7f2258668000->0x7f2258e68000 at 0x0082edc0: load6 ALLOC LOAD HAS_CONTENTS
    0x7f2258e69000->0x7f2259669000 at 0x0102edc0: load7 ALLOC LOAD HAS_CONTENTS
    0x7f225966a000->0x7f2259e6a000 at 0x0182edc0: load8 ALLOC LOAD HAS_CONTENTS
    0x7f2259e6b000->0x7f225a66b000 at 0x0202edc0: load9 ALLOC LOAD HAS_CONTENTS
    0x7ffc7d24d000->0x7ffc7d26e000 at 0x0282edc0: load10 ALLOC LOAD HAS_CONTENTS
    0x7ffc7d299000->0x7ffc7d29a000 at 0x0284fdc0: load11 ALLOC LOAD READONLY CODE HAS_CONTENTS
    0xffffffffff600000->0xffffffffff601000 at 0x02850dc0: load12 ALLOC LOAD READONLY CODE HAS_CONTENTS
```

9. Dump data with possible symbolic information:

```
(gdb) x/512a 0x006b6000
0x6b6000:    0x0    0xc2740000001c
0x6b6010:    0x50fffd2880 0x80e0a7e100e4400
0x6b6020:    0x80e470b46  0xc29400000014
0x6b6030:    0x8fffd28b0  0x0
0x6b6040:    0xc2ac00000014      0x15fffd28a8
0x6b6050:    0x0    0xc2c400000014
0x6b6060:    0x8fffd28b0  0x0
0x6b6070:    0xc2dc00000014      0x8fffd28a8
0x6b6080:    0x0    0xc2f400000014
0x6b6090:    0x8fffd28a0  0x0
0x6b60a0:    0xc30c0000001c      0x24fffd2898
0x6b60b0:    0x80e0a5a300e4400   0xb42
0x6b60c0:    0xc32c00000014      0x8fffd28a8
0x6b60d0:    0x0    0xc34400000014
0x6b60e0:    0x8fffd28a0  0x0
0x6b60f0:    0xc35c0000002c      0x110fffd2898
0x6b6100:    0xe580283100e4100   0x44100e0ae4020580
0x6b6110:    0x44100e490b41080e  0x80e
0x6b6120:    0xc38c00000014      0x1fffd2978
0x6b6130:    0x0    0xc3a40000003c
0x6b6140:    0x166fffd29700xd430286100e4100
0x6b6150:    0x58d048e038f4a06   0x8150078347068c49
```

```
0x6b6160:    0x70c0a8702098008   0x20cc6a2020b4b08
0x6b6170:    0x8    0xc3e400000034
---Type <return> to continue, or q <return> to quit---
0x6b6180:    0xe6fffd2aa0 0xd430286100e4100
0x6b6190:    0x783088109805006   0x4e048e058d4f068c
0x6b61a0:    0x8070c0a5b02038f   0x8020cc655020b41
0x6b61b0:    0xc41c00000034      0xc1fffd2b58
0x6b61c0:    0xd430286100e4100   0x58d048e038f4a06
0x6b61d0:    0x8153078348068c45  0x20cc68f02098008
0x6b61e0:    0x8    0xc45400000034
0x6b61f0:    0xf1fffd2bf0 0xd430286100e4100
0x6b6200:    0x815e098007834806  0x8f048e058d068c08
0x6b6210:    0xb4508070c0a6103   0x8020cc69d02
0x6b6220:    0xc48c00000014      0x1afffd2cb8
0x6b6230:    0x0    0xc4a40000002c
0x6b6240:    0x99fffd2cc0 0xd430286100e4100
0x6b6250:    0x58d048e038f4606   0x730207834f068c4c
0x6b6260:    0x8070c      0xc4d400000014
0x6b6270:    0x46fffd2d30 0x0
0x6b6280:    0xc4ec00000014      0x1bfffd2d68
0x6b6290:    0x0    0xc5040000004c
0x6b62a0:    0xa3fffd2d70 0xe42028f100e4200
0x6b62b0:    0x48d200e45038e18   0x300e44058c280e45
0x6b62c0:    0x480783380e410686  0x41380e0a5202500e
0x6b62d0:    0x200e42280e41300e  0xe42100e42180e42
0x6b62e0:    0xb4908      0xc55400000044
0x6b62f0:    0xc8fffd2dd0 0xe46028f100e4200
---Type <return> to continue, or q <return> to quit---
0x6b6300:    0x48d200e42038e18   0x300e44058c280e45
0x6b6310:    0x470783380e410686  0xe41380ea202500e
0x6b6320:    0x42200e42280e4130  0x80e42100e42180e
0x6b6330:    0xc59c0000002c      0x67fffd2e58
0x6b6340:    0x80e0a7a100e4400   0xb47080e0a490b42
0x6b6350:    0xe460b47080e0a49   0x8
0x6b6360:    0xc5cc00000024      0x13cfffd2e98
0x6b6370:    0xe4b028c04834a00   0x80e0a7a02038640
0x6b6380:    0xb41  0xc5f400000034
0x6b6390:    0x109fffd2fb00xe480286100e4100
0x6b63a0:    0xa68300e44038318   0x80e41100e41180e
0x6b63b0:    0x41180e0a97020b49  0xb47080e41100e
0x6b63c0:    0xc62c00000024      0x6bfffd3088
0x6b63d0:    0x80e0a77100e4400   0xb49080e0a470b45
0x6b63e0:    0xb49080e0a470xc6540000004c
0x6b63f0:    0x178fffd30d00xe45028f100e4200
0x6b6400:    0x48d200e42038e18   0x300e41058c280e42
0x6b6410:    0x440783380e410686  0x380e0a015103700e
0x6b6420:    0xe42280e41300e41   0x42100e42180e4220
0x6b6430:    0xb41080e    0xc6a40000004c
0x6b6440:    0x157fffd32000xe49028f100e4200
0x6b6450:    0x48d200e42038e18   0x300e45058c280e48
0x6b6460:    0x4a0783380e410686  0x41380e012703700e
0x6b6470:    0x200e42280e41300e  0xe42100e42180e42
---Type <return> to continue, or q <return> to quit---
0x6b6480:    0x8    0xc6f400000024
0x6b6490:    0xb0fffd3310 0x8d4d058606834a00
0x6b64a0:    0x48c400e4c028e03   0x80e8c02
0x6b64b0:    0xc71c0000004c      0x194fffd3398
0x6b64c0:    0xe4a028f100e4200   0x48d200e45038e18
0x6b64d0:    0x300e41058c280e45  0x4a0783380e470686
0x6b64e0:    0x380e0a015403700e  0xe42280e41300e44
```

```
0x6b64f0:    0x42100e42180e4220  0xb47080e
0x6b6500:    0xc76c00000024      0x6bfffd34e8
0x6b6510:    0x80e0a77100e4400   0xb49080e0a470b45
0x6b6520:    0xb49080e0a470xc7940000004c
0x6b6530:    0x673fffd35300xe42028f100e4200
0x6b6540:    0x48d200e42038e18   0x300e41058c280e42
0x6b6550:    0x470783380e410686  0x380e0a7d0201900e
0x6b6560:    0xe42280e41300e44   0x42100e42180e4220
0x6b6570:    0xb45080e    0xc7e400000024
0x6b6580:    0xcffffd3b60 0x8c4d058606834a00
0x6b6590:    0x28e400e4c038d04   0x80eab02
0x6b65a0:    0xc80c0000004c      0x4b3fffd3c08
0x6b65b0:    0xe42028f100e4200   0x48d200e42038e18
0x6b65c0:    0x300e41058c280e42  0x470783380e410686
0x6b65d0:    0x380e0af20201a00e  0xe42280e41300e43
0x6b65e0:    0x42100e42180e4220  0xb41080e
0x6b65f0:    0xc85c00000014      0x8afffd4078
---Type <return> to continue, or q <return> to quit---
0x6b6600:    0x80e6c200e460200   0xc87400000014
0x6b6610:    0x9fffd40f0  0x0
0x6b6620:    0xc88c0000001c      0x26fffd40e8
0x6b6630:    0xa4a0283100e4100   0x80e510b45080e
0x6b6640:    0xc8ac0000001c      0x72fffd40f8
0x6b6650:    0xa7e0283100e5b00   0x80e4f0b45080e
0x6b6660:    0xc8cc00000014      0x9fffd4158
0x6b6670:    0x0    0xc8e40000001c
0x6b6680:    0x1afffd4150 0xe540283100e4100
0x6b6690:    0x8    0xc9040000003c
0x6b66a0:    0x113fffd41500xe44028c100e4200
0x6b66b0:    0x483200e44038618   0x100e41180e0ab902
0x6b66c0:    0xe0a560b4a080e42   0x47080e42100e4118
0x6b66d0:    0xb    0xc94400000014
0x6b66e0:    0x5fffd4230  0x0
0x6b66f0:    0xc95c00000014      0x25fffd4228
0x6b6700:    0x80e49100e5400     0xc97400000044
0x6b6710:    0x1f8fffd42400xe42028e100e4200
0x6b6720:    0x48c200e45038d18   0x300e440586280e41
0x6b6730:    0xacb02700e440683   0x200e41280e44300e
0x6b6740:    0xe42100e42180e42   0xb4108
0x6b6750:    0xc9bc0000002c      0x7cfffd43f8
0x6b6760:    0x80e0a76100e4400   0xb49080e0a570b46
0x6b6770:    0xe470b49080e0a47   0x8
---Type <return> to continue, or q <return> to quit---
0x6b6780:    0xc9ec00000024      0x13cfffd4448
0x6b6790:    0x5a020283100e4500  0xedb020b41080e0a
0x6b67a0:    0x8    0xca140000004c
0x6b67b0:    0x242fffd45600xe45028e100e6200
0x6b67c0:    0x48c200e45038d18   0x300e410586280e44
0x6b67d0:    0x7e0301800e440683  0x280ec341300e0a01
0x6b67e0:    0x180ecc42200ec641  0x80ece42100ecd42
0x6b67f0:    0xb45  0xca6400000034
0x6b6800:    0x1aafffd47600x43180e47100e4200
0x6b6810:    0x43200e42028f038e  0x300e41280e42048d
0x6b6820:    0x4501900e44380e41  0x58c06860783
0x6b6830:    0xca9c0000001c      0x87fffd48d8
0x6b6840:    0x8302864a600e4e00  0x3
0x6b6850:    0xcabc00000014      0x15fffd4948
0x6b6860:    0x0    0x901ffff00000000
0x6b6870:    0x601910070044c     0x5c01a41001ffff00
0x6b6880:    0x3c10502f30000     0x1ffff0000050481
```

```
0x6b6890:    0x1b10001b603670a   0x961201ffff000046
0x6b68a0:    0x309b6000004eb02   0x1b60a96000b82
0x6b68b0:    0x301b90c01ffff00   0x2ac02830003e5
0x6b68c0:    0x501c61101ffff00   0x8ae068b01fd0000
0x6b68d0:    0xffff00000508b400  0x9500018105660a01
0x6b68e0:    0x801ffff00000501   0x561004d053d
0x6b68f0:    0x1d301c11e01ffff   0xba20503f90000
---Type <return> to continue, or q <return> to quit---
0x6b6900:    0xa406cb0000050684  0x2a50990000b8a02
0x6b6910:    0x5720a01ffff0000   0x502950001d5
0x6b6920:    0x920301990b01ffff  0xff00000502ce0002
0x6b6930:    0x1f705600a01ff     0x1ffff00000502b3
0x6b6940:    0x850002c903028a0b  0xc01ffff00000503
0x6b6950:    0x970004db029601eb  0xa01ffff00000505
0x6b6960:    0x501ef0001b3056b   0x5650a01ffff0000
0x6b6970:    0x501e90001ad0x1f705600a01ffff
0x6b6980:    0x502b300    0x6bdec0 <_res>
0x6b6990:    0x6b7640 <_nl_global_locale>     0x6b7640 <_nl_global_locale>
0x6b69a0:    0x6b7660 <_nl_global_locale+32>  0x6b7648 <_nl_global_locale+8>
0x6b69b0 <__init_array_start>:   0x4004b0 <frame_dummy>     0x42f4c0 <init_cacheinfo>
0x6b69c0 <__fini_array_start>:   0x400480 <__do_global_dtors_aux> 0x46fcc0 <fini>
0x6b69d0 <__JCR_LIST__>:  0x0    0x0
0x6b69e0 <_dl_argv>:      0x6b72c0 <program_invocation_short_name>      0x7ffc7d26c7e8
0x6b69f0 <_dl_random>:    0x7ffc7d26c9b9      0x0
0x6b6a00 <__stack_prot>:  0x1000000    0x0
0x6b6a10 <env_path_list>: 0xffffffffffffffff  0x0
0x6b6a20 <capstr>:  0x6be130 <result.11783>   0x1
0x6b6a30 <max_capstrlen>: 0x0    0x0
0x6b6a40 <rtld_search_dirs>:     0x227d190    0x0
---Type <return> to continue, or q <return> to quit---
0x6b6a50:    0x403c00 <pthread_cancel> 0x0
0x6b6a60 <_GLOBAL_OFFSET_TABLE_>:      0x0    0x0
0x6b6a70 <_GLOBAL_OFFSET_TABLE_+16>:   0x0    0x41ea40 <__stpcpy_ssse3>
0x6b6a80 <_GLOBAL_OFFSET_TABLE_+32>:   0x41b040 <__strcpy_ssse3> 0x426950 <__memmove_ssse3>
0x6b6a90 <_GLOBAL_OFFSET_TABLE_+48>:   0x423f00 <__rawmemchr_sse42>     0x453760
<__strstr_sse42>
0x6b6aa0 <_GLOBAL_OFFSET_TABLE_+64>:   0x470340 <__strncpy_ssse3>0x425300 <__memcmp_sse4_1>
0x6b6ab0 <_GLOBAL_OFFSET_TABLE_+80>:   0x421820 <__strcasecmp_l_sse42>  0x41da30
<__memset_sse2>
0x6b6ac0 <_GLOBAL_OFFSET_TABLE_+96>:   0x41a080 <__strcmp_sse42> 0x47f710
<__strncasecmp_l_sse42>
0x6b6ad0 <_GLOBAL_OFFSET_TABLE_+112>:  0x421810 <__strcasecmp_sse42>    0x418b50
<__strchr_sse42>
0x6b6ae0 <data_start>:    0x0    0x0
0x6b6af0 <__nptl_nthreads>:      0x6    0x0
0x6b6b00 <stack_used>:    0x7f22586669c0      0x7f225a66a9c0
0x6b6b10 <stack_cache>:   0x6b6b10 <stack_cache>    0x6b6b10 <stack_cache>
0x6b6b20 <__sched_fifo_min_prio>:      0xffffffffffffffff  0x800000
0x6b6b30 <_dl_tls_static_size>:  0x1160 0x48c997 <_nl_default_default_domain>
0x6b6b40 <locale_alias_path.12333>:    0x48c9c9     0x6bc6e0 <initial>
0x6b6b50:    0x0    0x0
0x6b6b60 <_IO_2_1_stdin_>:0xfbad2088   0x0
---Type <return> to continue, or q <return> to quit---
0x6b6b70 <_IO_2_1_stdin_+16>:    0x0    0x0
0x6b6b80 <_IO_2_1_stdin_+32>:    0x0    0x0
0x6b6b90 <_IO_2_1_stdin_+48>:    0x0    0x0
0x6b6ba0 <_IO_2_1_stdin_+64>:    0x0    0x0
0x6b6bb0 <_IO_2_1_stdin_+80>:    0x0    0x0
0x6b6bc0 <_IO_2_1_stdin_+96>:    0x0    0x0
0x6b6bd0 <_IO_2_1_stdin_+112>:   0x0    0xffffffffffffffff
```

```
0x6b6be0 <_IO_2_1_stdin_+128>:   0x0    0x6bcb20 <_IO_stdfile_0_lock>
0x6b6bf0 <_IO_2_1_stdin_+144>:   0xffffffffffffffff  0x0
0x6b6c00 <_IO_2_1_stdin_+160>:   0x6b6e20 <_IO_wide_data_0>0x0
0x6b6c10 <_IO_2_1_stdin_+176>:   0x0    0x0
0x6b6c20 <_IO_2_1_stdin_+192>:   0x0    0x0
0x6b6c30 <_IO_2_1_stdin_+208>:   0x0    0x48d440 <_IO_file_jumps>
0x6b6c40 <_IO_2_1_stdout_>:      0xfbad2084   0x0
0x6b6c50 <_IO_2_1_stdout_+16>:   0x0    0x0
0x6b6c60 <_IO_2_1_stdout_+32>:   0x0    0x0
0x6b6c70 <_IO_2_1_stdout_+48>:   0x0    0x0
0x6b6c80 <_IO_2_1_stdout_+64>:   0x0    0x0
0x6b6c90 <_IO_2_1_stdout_+80>:   0x0    0x0
0x6b6ca0 <_IO_2_1_stdout_+96>:   0x0    0x6b6b60 <_IO_2_1_stdin_>
0x6b6cb0 <_IO_2_1_stdout_+112>:  0x1    0xffffffffffffffff
0x6b6cc0 <_IO_2_1_stdout_+128>:  0x0    0x6bcb30 <_IO_stdfile_1_lock>
0x6b6cd0 <_IO_2_1_stdout_+144>:  0xffffffffffffffff  0x0
0x6b6ce0 <_IO_2_1_stdout_+160>:  0x6b6f80 <_IO_wide_data_1>0x0
---Type <return> to continue, or q <return> to quit---
0x6b6cf0 <_IO_2_1_stdout_+176>:  0x0    0x0
0x6b6d00 <_IO_2_1_stdout_+192>:  0x0    0x0
0x6b6d10 <_IO_2_1_stdout_+208>:  0x0    0x48d440 <_IO_file_jumps>
0x6b6d20 <_IO_2_1_stderr_>:      0xfbad2086   0x0
0x6b6d30 <_IO_2_1_stderr_+16>:   0x0    0x0
0x6b6d40 <_IO_2_1_stderr_+32>:   0x0    0x0
0x6b6d50 <_IO_2_1_stderr_+48>:   0x0    0x0
0x6b6d60 <_IO_2_1_stderr_+64>:   0x0    0x0
0x6b6d70 <_IO_2_1_stderr_+80>:   0x0    0x0
0x6b6d80 <_IO_2_1_stderr_+96>:   0x0    0x6b6c40 <_IO_2_1_stdout_>
0x6b6d90 <_IO_2_1_stderr_+112>:  0x2    0xffffffffffffffff
0x6b6da0 <_IO_2_1_stderr_+128>:  0x0    0x6bcb40 <_IO_stdfile_2_lock>
0x6b6db0 <_IO_2_1_stderr_+144>:  0xffffffffffffffff  0x0
0x6b6dc0 <_IO_2_1_stderr_+160>:  0x6b70e0 <_IO_wide_data_2>0x0
0x6b6dd0 <_IO_2_1_stderr_+176>:  0x0    0x0
0x6b6de0 <_IO_2_1_stderr_+192>:  0x0    0x0
0x6b6df0 <_IO_2_1_stderr_+208>:  0x0    0x48d440 <_IO_file_jumps>
0x6b6e00 <_IO_list_all>:  0x6b6d20 <_IO_2_1_stderr_>0x0
0x6b6e10:    0x0    0x0
0x6b6e20 <_IO_wide_data_0>:      0x0    0x0
0x6b6e30 <_IO_wide_data_0+16>:   0x0    0x0
0x6b6e40 <_IO_wide_data_0+32>:   0x0    0x0
0x6b6e50 <_IO_wide_data_0+48>:   0x0    0x0
0x6b6e60 <_IO_wide_data_0+64>:   0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x6b6e70 <_IO_wide_data_0+80>:   0x0    0x0
0x6b6e80 <_IO_wide_data_0+96>:   0x0    0x0
0x6b6e90 <_IO_wide_data_0+112>:  0x0    0x0
0x6b6ea0 <_IO_wide_data_0+128>:  0x0    0x0
0x6b6eb0 <_IO_wide_data_0+144>:  0x0    0x0
0x6b6ec0 <_IO_wide_data_0+160>:  0x0    0x0
0x6b6ed0 <_IO_wide_data_0+176>:  0x0    0x0
0x6b6ee0 <_IO_wide_data_0+192>:  0x0    0x0
0x6b6ef0 <_IO_wide_data_0+208>:  0x0    0x0
0x6b6f00 <_IO_wide_data_0+224>:  0x0    0x0
0x6b6f10 <_IO_wide_data_0+240>:  0x0    0x0
0x6b6f20 <_IO_wide_data_0+256>:  0x0    0x0
0x6b6f30 <_IO_wide_data_0+272>:  0x0    0x0
0x6b6f40 <_IO_wide_data_0+288>:  0x0    0x0
0x6b6f50 <_IO_wide_data_0+304>:  0x0    0x0
0x6b6f60 <_IO_wide_data_0+320>:  0x48d1c0 <_IO_wfile_jumps>0x0
0x6b6f70:    0x0    0x0
```

```
0x6b6f80 <_IO_wide_data_1>:      0x0    0x0
0x6b6f90 <_IO_wide_data_1+16>:   0x0    0x0
0x6b6fa0 <_IO_wide_data_1+32>:   0x0    0x0
0x6b6fb0 <_IO_wide_data_1+48>:   0x0    0x0
0x6b6fc0 <_IO_wide_data_1+64>:   0x0    0x0
0x6b6fd0 <_IO_wide_data_1+80>:   0x0    0x0
0x6b6fe0 <_IO_wide_data_1+96>:   0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x6b6ff0 <_IO_wide_data_1+112>:  0x0    0x0
```

The output is in the following format:

```
address:  value1  value2
```

Because the size of each value is 8 bytes the next address is +16 bytes or $+10_{hex}$. The addresses can have associated symbolic names:

```
address <name>:  value1  value2
```

For example, from the output above:

```
0x6b6af0 <__nptl_nthreads>:      0x6    0x0
```

Each value may also have an associated symbolic value:

```
address <name>:  value1 <name1>    value2
```

For example, from the output above:

```
0x6b69e0 <_dl_argv>:      0x6b72c0 <program_invocation_short_name>      0x7ffc7d26c7e8
```

10. Explore the contents of memory pointed to by *__nptl_nthreads*, *_dl_argv*, *program_invocation_short_name* and 0x7ffc7d26c7e8 addresses:

```
(gdb) x/u 0x6b6af0
0x6b6af0 <__nptl_nthreads>:      6

(gdb) x/u &__nptl_nthreads
0x6b6af0 <__nptl_nthreads>:      6

(gdb) x/2a 0x6b69e0
0x6b69e0 <_dl_argv>:      0x6b72c0 <program_invocation_short_name>      0x7ffc7d26c7e8

(gdb) x/2a &_dl_argv
0x6b69e0 <_dl_argv>:      0x6b72c0 <program_invocation_short_name>      0x7ffc7d26c7e8

(gdb) x/a 0x6b72c0
0x6b72c0 <program_invocation_short_name>:     0x7ffc7d26d9a9

(gdb) x/a &program_invocation_short_name
0x6b72c0 <program_invocation_short_name>:     0x7ffc7d26d9a9

(gdb) x/a 0x7ffc7d26d9a9
0x7ffc7d26d9a9:      "App1"
```

```
(gdb) x/10a 0x7ffc7d26c7e8
0x7ffc7d26c7e8:     0x0    0x1
0x7ffc7d26c7f8:     0x7ffc7d26d9a7      0x0
0x7ffc7d26c808:     0x7ffc7d26d9ae      0x7ffc7d26d9be
0x7ffc7d26c818:     0x7ffc7d26d9c9      0x7ffc7d26d9d9
0x7ffc7d26c828:     0x7ffc7d26d9e7      0x7ffc7d26df08

(gdb) x/10c 0x7ffc7d26d9a7
0x7ffc7d26d9a7:     46 '.' 47 '/' 65 'A' 112 'p'      112 'p'      49 '1' 0 '\000'     83 'S'
0x7ffc7d26d9af:     72 'H' 69 'E'

(gdb) x/s 0x7ffc7d26d9a7
0x7ffc7d26d9a7:      "./App1"

(gdb) x/5s 0x7ffc7d26d9a7
0x7ffc7d26d9a7:      "./App1"
0x7ffc7d26d9ae:      "SHELL=/bin/bash"
0x7ffc7d26d9be:      "TERM=linux"
0x7ffc7d26d9c9:      "HUSHLOGIN=FALSE"
0x7ffc7d26d9d9:      "USER=training"
```

11. Explore the contents of memory pointed to by *environ* variable address:

```
(gdb) x/a &environ
0x6bd4c8 <environ>: 0x7ffc7d26c808

(gdb) x/10a 0x7ffc7d26c808
0x7ffc7d26c808:     0x7ffc7d26d9ae      0x7ffc7d26d9be
0x7ffc7d26c818:     0x7ffc7d26d9c9      0x7ffc7d26d9d9
0x7ffc7d26c828:     0x7ffc7d26d9e7      0x7ffc7d26df08
0x7ffc7d26c838:     0x7ffc7d26df20      0x7ffc7d26df5e
0x7ffc7d26c848:     0x7ffc7d26df7c      0x7ffc7d26df8d

(gdb) x/4s 0x7ffc7d26d9ae
0x7ffc7d26d9ae:      "SHELL=/bin/bash"
0x7ffc7d26d9be:      "TERM=linux"
0x7ffc7d26d9c9:      "HUSHLOGIN=FALSE"
0x7ffc7d26d9d9:      "USER=training"
```

12. Get the list of loaded modules:

```
(gdb) info sharedlibrary
No shared libraries loaded at this time.
```

We don’t see any shared libraries because they were statically linked. We also created the version of a dynamically linked *App1.shared* executable. If we load its core dump we see the list of shared libraries:

```
training@debian64:~/ALCDA$ gdb -c ./App1/core.5476 -se ./App1/App1.shared
GNU gdb (GDB) 7.4.1-debian
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/training/ALCDA/App1/App1.shared...(no debugging symbols
found)...done.
[New LWP 5477]
```

```
[New LWP 5478]
[New LWP 5479]
[New LWP 5480]
[New LWP 5481]
[New LWP 5476]

warning: Can't read pathname for load map: Input/output error.
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Core was generated by `/home/training/ALCDA/App1/App1.shared'.
#0  0x00007f25a013e48d in nanosleep () from /lib/x86_64-linux-gnu/libc.so.6

(gdb) info sharedlibrary
From                To                  Syms Read   Shared Object Library
0x00007f25a0423690  0x00007f25a042ece8  Yes (*)     /lib/x86_64-linux-gnu/libpthread.so.0
0x00007f25a00b1b80  0x00007f25a01c9c2c  Yes (*)     /lib/x86_64-linux-gnu/libc.so.6
0x00007f25a063aaf0  0x00007f25a0652c83  Yes (*)     /lib64/ld-linux-x86-64.so.2
(*): Shared library is missing debugging information.
```

13. Disassemble *bar_two* function and follow the indirect *sleep* function call:

```
(gdb) disassemble bar_two
Dump of assembler code for function bar_two:
   0x00000000004005f9 <+0>:     push   %rbp
   0x00000000004005fa <+1>:     mov    %rsp,%rbp
   0x00000000004005fd <+4>:     mov    $0xffffffff,%edi
   0x0000000000400602 <+9>:     callq  0x4004a0 <sleep@plt>
   0x0000000000400607 <+14>:    pop    %rbp
   0x0000000000400608 <+15>:    retq
End of assembler dump.
```

```
(gdb) disassemble 0x4004a0
Dump of assembler code for function sleep@plt:
   0x00000000004004a0 <+0>:     jmpq   *0x20090a(%rip)        # 0x600db0 <sleep@got.plt>
   0x00000000004004a6 <+6>:     pushq  $0x2
   0x00000000004004ab <+11>:    jmpq   0x400470
End of assembler dump.
```

14. Dump the annotated value as a memory address interpreting its contents as a symbol:

```
(gdb) x/a 0x600db0
0x600db0 <sleep@got.plt>: 0x7f25a013e220 <sleep>
```

Exercise A2D

- **Goal:** Learn how to identify exceptions, find problem threads and CPU instructions
- **Patterns:** NULL Pointer (data), Active Thread
- \ALCDA-Dumps\Exercise-A2D.pdf

© 2015 Software Diagnostics Services

Exercise A2D

Goal: Learn how to identify exceptions, find problem threads and CPU instructions.

Patterns: NULL Pointer (data), Active Thread.

1. Load a core dump and *App2D* executable:

```
training@debian64:~/ALCDA$ gdb -c ./App2D/core -se ./App2D/App2D
GNU gdb (GDB) 7.4.1-debian
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/training/ALCDA/App2D/App2D...done.
[New LWP 3484]
[New LWP 3482]
[New LWP 3487]
[New LWP 3486]
[New LWP 3485]
[New LWP 3483]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Core was generated by `./App2D'.
Program terminated with signal 11, Segmentation fault.
#0  0x0000000000400500 in procA ()
```

2. List all threads:

```
(gdb) info threads
  Id   Target Id         Frame
  6    Thread 0x7f560d467700 (LWP 3483) 0x00000000004324a9 in clone ()
  5    Thread 0x7f560c465700 (LWP 3485) 0x000000000042fe31 in nanosleep ()
  4    Thread 0x7f560bc64700 (LWP 3486) 0x000000000042fe31 in nanosleep ()
  3    Thread 0x7f560b463700 (LWP 3487) 0x000000000042fe31 in nanosleep ()
  2    Thread 0x18b9860 (LWP 3482) 0x000000000042fe31 in nanosleep ()
* 1    Thread 0x7f560cc66700 (LWP 3484) 0x0000000000400500 in procA ()
```

3. The problem thread seems to be the current thread:

```
(gdb) thread 1
[Switching to thread 1 (Thread 0x7f560cc66700 (LWP 3484))]
#0  0x0000000000400500 in procA ()
```

```
(gdb) bt
#0  0x0000000000400500 in procA ()
#1  0x000000000040057a in bar_two ()
#2  0x000000000040058a in foo_two ()
#3  0x00000000004005a2 in thread_two ()
#4  0x0000000000401630 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#5  0x00000000004324e9 in clone ()
#6  0x0000000000000000 in ?? ()
```

4. Disassemble the problem instruction and check CPU register(s) details (NULL data pointer):

```
(gdb) x/i 0x400500
=> 0x400500 <procA+16>:    movl   $0x1,(%rax)

(gdb) info r $rax
rax            0x0  0

(gdb) x $rax
   0x0: Cannot access memory at address 0x0
```

5. List all thread stack traces and identify other anomalies such as non-waiting active threads:

```
(gdb) thread apply all bt

Thread 6 (Thread 0x7f560d467700 (LWP 3483)):
#0  0x00000000004324a9 in clone ()
#1  0x0000000000401560 in ?? () at pthread_create.c:217
#2  0x00007f560d467700 in ?? ()
#3  0x0000000000000000 in ?? ()

Thread 5 (Thread 0x7f560c465700 (LWP 3485)):
#0  0x000000000042fe31 in nanosleep ()
#1  0x000000000042fd00 in sleep ()
#2  0x00000000004005b7 in bar_three ()
#3  0x00000000004005c7 in foo_three ()
#4  0x00000000004005df in thread_three ()
#5  0x0000000000401630 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#6  0x00000000004324e9 in clone ()
#7  0x0000000000000000 in ?? ()

Thread 4 (Thread 0x7f560bc64700 (LWP 3486)):
#0  0x000000000042fe31 in nanosleep ()
#1  0x000000000042fd00 in sleep ()
#2  0x000000000040051a in procB ()
#3  0x00000000004005f4 in bar_four ()
#4  0x0000000000400604 in foo_four ()
---Type <return> to continue, or q <return> to quit---
#5  0x000000000040061c in thread_four ()
#6  0x0000000000401630 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#7  0x00000000004324e9 in clone ()
#8  0x0000000000000000 in ?? ()
```

```
Thread 3 (Thread 0x7f560b463700 (LWP 3487)):
#0  0x000000000042fe31 in nanosleep ()
#1  0x000000000042fd00 in sleep ()
#2  0x0000000000400631 in bar_five ()
#3  0x0000000000400641 in foo_five ()
#4  0x0000000000400659 in thread_five ()
#5  0x0000000000401630 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#6  0x00000000004324e9 in clone ()
#7  0x0000000000000000 in ?? ()

Thread 2 (Thread 0x18b9860 (LWP 3482)):
#0  0x000000000042fe31 in nanosleep ()
#1  0x000000000042fd00 in sleep ()
#2  0x0000000000400700 in main ()

Thread 1 (Thread 0x7f560cc66700 (LWP 3484)):
#0  0x0000000000400500 in procA ()
---Type <return> to continue, or q <return> to quit---
#1  0x000000000040057a in bar_two ()
#2  0x000000000040058a in foo_two ()
#3  0x00000000004005a2 in thread_two ()
#4  0x0000000000401630 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#5  0x00000000004324e9 in clone ()
#6  0x0000000000000000 in ?? ()
```

6. Check the CPU instruction and the stack pointer of the thread #6 for any signs of stack overflow (unaccessible stack addresses below the current stack pointer):

```
(gdb) thread 6
[Switching to thread 6 (Thread 0x7f560d467700 (LWP 3483))]
#0  0x00000000004324a9 in clone ()

(gdb) bt
#0  0x00000000004324a9 in clone ()
#1  0x0000000000401560 in ?? () at pthread_create.c:217
#2  0x00007f560d467700 in ?? ()
#3  0x0000000000000000 in ?? ()

(gdb) x/i 0x4324a9
=> 0x4324a9 <clone+57>:   test   %rax,%rax

(gdb) x/xg $rsp
0x7f560d466e90:     0x0000000000401560

(gdb) x/xg $rsp-8
0x7f560d466e88:     0x0000000000000000

(gdb) x/xg $rsp-x10
0x7f560d466e80:     0x0000000000000000
```

7. Switch to the thread #2 and verify that main function was recently engaged in thread creation (this may correlate with the last thread #6 caught in being created):

```
(gdb) thread 2
[Switching to thread 2 (Thread 0x18b9860 (LWP 3482))]
#0  0x000000000042fe31 in nanosleep ()

(gdb) bt
#0  0x000000000042fe31 in nanosleep ()
#1  0x000000000042fd00 in sleep ()
#2  0x0000000000400700 in main ()

(gdb) disassemble main
Dump of assembler code for function main:
   0x0000000000400660 <+0>:     push   %rbp
   0x0000000000400661 <+1>:     mov    %rsp,%rbp
   0x0000000000400664 <+4>:     sub    $0x40,%rsp
   0x0000000000400668 <+8>:     mov    %edi,-0x34(%rbp)
   0x000000000040066b <+11>:    mov    %rsi,-0x40(%rbp)
   0x000000000040066f <+15>:    lea    -0x8(%rbp),%rax
   0x0000000000400673 <+19>:    mov    $0x0,%ecx
   0x0000000000400678 <+24>:    mov    $0x40054f,%edx
   0x000000000040067d <+29>:    mov    $0x0,%esi
   0x0000000000400682 <+34>:    mov    %rax,%rdi
   0x0000000000400685 <+37>:    callq  0x4019c0 <__pthread_create_2_1>
   0x000000000040068a <+42>:    lea    -0x10(%rbp),%rax
   0x000000000040068e <+46>:    mov    $0x0,%ecx
   0x0000000000400693 <+51>:    mov    $0x40058c,%edx
   0x0000000000400698 <+56>:    mov    $0x0,%esi
   0x000000000040069d <+61>:    mov    %rax,%rdi
   0x00000000004006a0 <+64>:    callq  0x4019c0 <__pthread_create_2_1>
   0x00000000004006a5 <+69>:    lea    -0x18(%rbp),%rax
   0x00000000004006a9 <+73>:    mov    $0x0,%ecx
   0x00000000004006ae <+78>:    mov    $0x4005c9,%edx
   0x00000000004006b3 <+83>:    mov    $0x0,%esi
   0x00000000004006b8 <+88>:    mov    %rax,%rdi
   0x00000000004006bb <+91>:    callq  0x4019c0 <__pthread_create_2_1>
---Type <return> to continue, or q <return> to quit---
   0x00000000004006c0 <+96>:    lea    -0x20(%rbp),%rax
   0x00000000004006c4 <+100>:   mov    $0x0,%ecx
   0x00000000004006c9 <+105>:   mov    $0x400606,%edx
   0x00000000004006ce <+110>:   mov    $0x0,%esi
   0x00000000004006d3 <+115>:   mov    %rax,%rdi
   0x00000000004006d6 <+118>:   callq  0x4019c0 <__pthread_create_2_1>
   0x00000000004006db <+123>:   lea    -0x28(%rbp),%rax
   0x00000000004006df <+127>:   mov    $0x0,%ecx
   0x00000000004006e4 <+132>:   mov    $0x400643,%edx
   0x00000000004006e9 <+137>:   mov    $0x0,%esi
   0x00000000004006ee <+142>:   mov    %rax,%rdi
   0x00000000004006f1 <+145>:   callq  0x4019c0 <__pthread_create_2_1>
   0x00000000004006f6 <+150>:   mov    $0x3,%edi
   0x00000000004006fb <+155>:   callq  0x42fc20 <sleep>
   0x0000000000400700 <+160>:   mov    $0x0,%eax
   0x0000000000400705 <+165>:   leaveq
   0x0000000000400706 <+166>:   retq
End of assembler dump.
```

Exercise A2C

- **Goal:** Learn how to identify exceptions, find problem threads and CPU instructions
- **Patterns:** NULL Pointer (code), Active Thread
- \ALCDA-Dumps\Exercise-A2C.pdf

© 2015 Software Diagnostics Services

Exercise A2C

Goal: Learn how to identify exceptions, find problem threads and CPU instructions.

Patterns: NULL Pointer (code), Active Thread.

1. Load a core dump and *App2C* executable:

```
training@debian64:~/ALCDA$ gdb -c ./App2C/core -se ./App2C/App2C
GNU gdb (GDB) 7.4.1-debian
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/training/ALCDA/App2C/App2C...done.
[New LWP 3423]
[New LWP 3419]
[New LWP 3424]
[New LWP 3422]
[New LWP 3421]
[New LWP 3420]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Core was generated by `./App2C'.
Program terminated with signal 11, Segmentation fault.
#0  0x0000000000000000 in ?? ()
```

2. List all threads:

```
(gdb) info threads
  Id   Target Id         Frame
  6    Thread 0x7f5ef3b7e700 (LWP 3420) 0x00000000004324a9 in clone ()
  5    Thread 0x7f5ef337d700 (LWP 3421) 0x00000000004324a9 in clone ()
  4    Thread 0x7f5ef2b7c700 (LWP 3422) 0x00000000004324a9 in clone ()
  3    Thread 0x7f5ef1b7a700 (LWP 3424) 0x000000000042fe31 in nanosleep ()
  2    Thread 0x145a860 (LWP 3419) 0x000000000042fe31 in nanosleep ()
* 1    Thread 0x7f5ef237b700 (LWP 3423) 0x0000000000000000 in ?? ()
```

3. The problem thread seems to be the current thread:

```
(gdb) bt
#0  0x0000000000000000 in ?? ()
#1  0x0000000000400531 in procB ()
#2  0x00000000004005f8 in bar_four ()
#3  0x0000000000400608 in foo_four ()
#4  0x0000000000400620 in thread_four ()
#5  0x0000000000401630 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#6  0x00000000004324e9 in clone ()
#7  0x0000000000000000 in ?? ()
```

4. Check the CPU instruction and a dereferenced pointer for any signs of a NULL pointer:

```
(gdb) disassemble procB
Dump of assembler code for function procB:
   0x0000000000400516 <+0>:     push   %rbp
   0x0000000000400517 <+1>:     mov    %rsp,%rbp
   0x000000000040051a <+4>:     sub    $0x10,%rsp
   0x000000000040051e <+8>:     movq   $0x0,-0x8(%rbp)
   0x0000000000400526 <+16>:    mov    -0x8(%rbp),%rdx
   0x000000000040052a <+20>:    mov    $0x0,%eax
   0x000000000040052f <+25>:    callq  *%rdx
   0x0000000000400531 <+27>:    leaveq
   0x0000000000400532 <+28>:    retq
End of assembler dump.

(gdb) info r rdx
rdx            0x0  0
```

5. List all thread stack traces and identify other anomalies such as non-waiting active threads:

```
(gdb) thread apply all bt

Thread 6 (Thread 0x7f5ef3b7e700 (LWP 3420)):
#0  0x00000000004324a9 in clone ()
#1  0x0000000000401560 in ?? () at pthread_create.c:217
#2  0x00007f5ef3b7e700 in ?? ()
#3  0x0000000000000000 in ?? ()

Thread 5 (Thread 0x7f5ef337d700 (LWP 3421)):
#0  0x00000000004324a9 in clone ()
#1  0x0000000000401560 in ?? () at pthread_create.c:217
#2  0x00007f5ef337d700 in ?? ()
#3  0x0000000000000000 in ?? ()

Thread 4 (Thread 0x7f5ef2b7c700 (LWP 3422)):
#0  0x00000000004324a9 in clone ()
#1  0x0000000000401560 in ?? () at pthread_create.c:217
#2  0x00007f5ef2b7c700 in ?? ()
#3  0x0000000000000000 in ?? ()

Thread 3 (Thread 0x7f5ef1b7a700 (LWP 3424)):
#0  0x000000000042fe31 in nanosleep ()
#1  0x000000000042fd00 in sleep ()
#2  0x0000000000400635 in bar_five ()
#3  0x0000000000400645 in foo_five ()
---Type <return> to continue, or q <return> to quit---
#4  0x000000000040065d in thread_five ()
#5  0x0000000000401630 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#6  0x00000000004324e9 in clone ()
#7  0x0000000000000000 in ?? ()

Thread 2 (Thread 0x145a860 (LWP 3419)):
#0  0x000000000042fe31 in nanosleep ()
#1  0x000000000042fd00 in sleep ()
#2  0x0000000000400704 in main ()
```

```
Thread 1 (Thread 0x7f5ef237b700 (LWP 3423)):
#0  0x0000000000000000 in ?? ()
#1  0x0000000000400531 in procB ()
#2  0x00000000004005f8 in bar_four ()
#3  0x0000000000400608 in foo_four ()
#4  0x0000000000400620 in thread_four ()
#5  0x0000000000401630 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#6  0x00000000004324e9 in clone ()
#7  0x0000000000000000 in ?? ()
```

Exercise A3

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

© 2015 Software Diagnostics Services

Exercise A3

Goal: Learn how to identify spiking threads.

Patterns: Spiking Thread.

1. The application **App3** was consuming CPU (from **top** output):

```
top - 22:32:00 up 23 min,  1 user,  load average: 0.95, 0.38, 0.17
Tasks:  64 total,   1 running,  63 sleeping,   0 stopped,   0 zombie
%Cpu(s):100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:    505464 total,   174140 used,   331324 free,    11872 buffers
KiB Swap:   223228 total,        0 used,   223228 free,   122696 cached

  PID USER      PR  NI  VIRT  RES  SHR S  %CPU %MEM    TIME+  COMMAND
 3712 training  20   0 42040    4    0 S  99.9  0.0   2:06.33 App3
    1 root      20   0 10648  836  704 S   0.0  0.2   0:01.34 init
    2 root      20   0     0    0    0 S   0.0  0.0   0:00.00 kthreadd
    3 root      20   0     0    0    0 S   0.0  0.0   0:00.03 ksoftirqd/0
    5 root      20   0     0    0    0 S   0.0  0.0   0:00.00 kworker/u:0
    6 root      rt   0     0    0    0 S   0.0  0.0   0:00.00 migration/0
    7 root      rt   0     0    0    0 S   0.0  0.0   0:00.01 watchdog/0
    8 root       0 -20     0    0    0 S   0.0  0.0   0:00.00 cpuset
    9 root       0 -20     0    0    0 S   0.0  0.0   0:00.00 khelper
   10 root      20   0     0    0    0 S   0.0  0.0   0:00.00 kdevtmpfs
   11 root       0 -20     0    0    0 S   0.0  0.0   0:00.00 netns
   12 root      20   0     0    0    0 S   0.0  0.0   0:00.01 sync_supers
   13 root      20   0     0    0    0 S   0.0  0.0   0:00.00 bdi-default
   14 root       0 -20     0    0    0 S   0.0  0.0   0:00.00 kintegrityd
   15 root       0 -20     0    0    0 S   0.0  0.0   0:00.00 kblockd
   16 root      20   0     0    0    0 S   0.0  0.0   0:00.00 khungtaskd
   17 root      20   0     0    0    0 S   0.0  0.0   0:00.00 kswapd0
   18 root      25   5     0    0    0 S   0.0  0.0   0:00.00 ksmd
```

Its core dump was saved using **gcore**:

```
    5 root      20   0     0    0    0 S   0.0  0.0   0:00.00 kworker/u:0
    6 root      rt   0     0    0    0 S   0.0  0.0   0:00.00 migration/0
    7 root      rt   0     0    0    0 S   0.0  0.0   0:00.01 watchdog/0
    8 root       0 -20     0    0    0 S   0.0  0.0   0:00.00 cpuset
    9 root       0 -20     0    0    0 S   0.0  0.0   0:00.00 khelper
   10 root      20   0     0    0    0 S   0.0  0.0   0:00.00 kdevtmpfs
   11 root       0 -20     0    0    0 S   0.0  0.0   0:00.00 netns
   12 root      20   0     0    0    0 S   0.0  0.0   0:00.01 sync_supers
   13 root      20   0     0    0    0 S   0.0  0.0   0:00.00 bdi-default
   14 root       0 -20     0    0    0 S   0.0  0.0   0:00.00 kintegrityd
   15 root       0 -20     0    0    0 S   0.0  0.0   0:00.00 kblockd
   16 root      20   0     0    0    0 S   0.0  0.0   0:00.00 khungtaskd
   17 root      20   0     0    0    0 S   0.0  0.0   0:00.00 kswapd0
   18 root      25   5     0    0    0 S   0.0  0.0   0:00.00 ksmd
training@debian64:~/ALCDA/App3$ gcore 3712
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
[New Thread 0x7f515137a700 (LWP 3717)]
[New Thread 0x7f5151b7b700 (LWP 3716)]
[New Thread 0x7f515237c700 (LWP 3715)]
[New Thread 0x7f5152b7d700 (LWP 3714)]
[New Thread 0x7f515337e700 (LWP 3713)]
0x00000000004329d1 in nanosleep ()
Saved corefile core.3712
training@debian64:~/ALCDA/App3$ _
```

2. Load a core dump *core.3712* and *App3* executable:

```
training@debian64:~/ALCDA$ gdb -c ./App3/core.3712 -se ./App3/App3
GNU gdb (GDB) 7.4.1-debian
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/training/ALCDA/App3/App3...done.
[New LWP 3713]
[New LWP 3714]
[New LWP 3715]
[New LWP 3716]
[New LWP 3717]
[New LWP 3712]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Core was generated by `/home/training/ALCDA/App3/App3'.
#0  0x00000000004329d1 in nanosleep ()
```

2. List all threads:

```
(gdb) info threads
  Id   Target Id         Frame
  6    LWP 3712          0x00000000004329d1 in nanosleep ()
  5    LWP 3717          0x00000000004007a3 in isnan ()
  4    LWP 3716          0x00000000004329d1 in nanosleep ()
  3    LWP 3715          0x00000000004329d1 in nanosleep ()
  2    LWP 3714          0x00000000004329d1 in nanosleep ()
* 1    LWP 3713          0x00000000004329d1 in nanosleep ()
```

3. Switch to the problem thread #5:

```
(gdb) thread 5
[Switching to thread 5 (LWP 3717)]
#0  0x00000000004007a3 in isnan ()

(gdb) bt
#0  0x00000000004007a3 in isnan ()
#1  0x0000000000400743 in sqrt ()
#2  0x0000000000400528 in procB ()
#3  0x0000000000400639 in bar_five ()
#4  0x0000000000400649 in foo_five ()
#5  0x0000000000400661 in thread_five ()
#6  0x0000000000403e30 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#7  0x0000000000435089 in clone ()
#8  0x0000000000000000 in ?? ()
```

4. Disassemble the problem instruction and check if it is normal:

```
(gdb) x/i 0x4007a3
=> 0x4007a3 <isnan+35>:   retq
```

5. Disassemble the return address for *procB* function to come back to see an infinite loop:

```
(gdb) disassemble 0x400528
Dump of assembler code for function procB:
   0x0000000000400500 <+0>:     push   %rbp
   0x0000000000400501 <+1>:     mov    %rsp,%rbp
   0x0000000000400504 <+4>:     sub    $0x20,%rsp
   0x0000000000400508 <+8>:     movabs $0x3fd5555555555555,%rax
   0x0000000000400512 <+18>:    mov    %rax,-0x8(%rbp)
   0x0000000000400516 <+22>:    mov    -0x8(%rbp),%rax
   0x000000000040051a <+26>:    mov    %rax,-0x18(%rbp)
   0x000000000040051e <+30>:    movsd  -0x18(%rbp),%xmm0
   0x0000000000400523 <+35>:    callq  0x400710 <sqrt>
   0x0000000000400528 <+40>:    movsd  %xmm0,-0x18(%rbp)
   0x000000000040052d <+45>:    mov    -0x18(%rbp),%rax
   0x0000000000400531 <+49>:    mov    %rax,-0x8(%rbp)
   0x0000000000400535 <+53>:    jmp    0x400516 <procB+22>
End of assembler dump.
```

Exercise A4

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

© 2015 Software Diagnostics Services

Exercise A4

Goal: Learn how to identify heap regions and heap corruption.

Patterns: Heap Corruption.

1. Load a core dump and *App4* executable:

```
training@debian64:~/ALCDA$ gdb -c ./App4/core -se ./App4/App4
GNU gdb (GDB) 7.4.1-debian
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/training/ALCDA/App4/App4...done.
[New LWP 11060]
[New LWP 11057]
[New LWP 11062]
[New LWP 11061]
[New LWP 11059]
[New LWP 11058]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Core was generated by `./App4'.
Program terminated with signal 11, Segmentation fault.
#0  0x000000000041482e in _int_malloc ()
```

2. List threads:

```
(gdb) info threads
  Id   Target Id         Frame
  6    Thread 0x7efd13f93700 (LWP 11058) 0x00000000004325c9 in clone ()
  5    Thread 0x7efd13792700 (LWP 11059) 0x00000000004325c9 in clone ()
  4    Thread 0x7efd12790700 (LWP 11061) 0x000000000042ff51 in nanosleep ()
  3    Thread 0x7efd11f8f700 (LWP 11062) 0x000000000042ff51 in nanosleep ()
  2    Thread 0x1ab2860 (LWP 11057) 0x000000000042ff51 in nanosleep ()
* 1    Thread 0x7efd12f91700 (LWP 11060) 0x000000000041482e in _int_malloc ()
```

3. The identifed problem thread #1 is the current thread. List its stack trace:

```
(gdb) bt
#0  0x000000000041482e in _int_malloc ()
#1  0x0000000000416d88 in malloc ()
#2  0x00000000004005dc in proc ()
#3  0x00000000004006ee in bar_three ()
#4  0x00000000004006fe in foo_three ()
#5  0x0000000000400716 in thread_three ()
#6  0x0000000000401760 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#7  0x0000000000432609 in clone ()
#8  0x0000000000000000 in ?? ()
```

4. We see that the segmentation fault happened internally in *malloc* function when *proc* was allocating heap memory. Diassemble *proc* function:

```
(gdb) disassemble proc
Dump of assembler code for function proc:
   0x00000000004004f0 <+0>:     push   %rbp
   0x00000000004004f1 <+1>:     mov    %rsp,%rbp
   0x00000000004004f4 <+4>:     sub    $0x40,%rsp
   0x00000000004004f8 <+8>:     mov    $0x400,%edi
   0x00000000004004fd <+13>:    callq  0x416d20 <malloc>
   0x0000000000400502 <+18>:    mov    %rax,-0x8(%rbp)
   0x0000000000400506 <+22>:    mov    $0x400,%edi
   0x000000000040050b <+27>:    callq  0x416d20 <malloc>
   0x0000000000400510 <+32>:    mov    %rax,-0x10(%rbp)
   0x0000000000400514 <+36>:    mov    $0x400,%edi
   0x0000000000400519 <+41>:    callq  0x416d20 <malloc>
   0x000000000040051e <+46>:    mov    %rax,-0x18(%rbp)
   0x0000000000400522 <+50>:    mov    $0x400,%edi
   0x0000000000400527 <+55>:    callq  0x416d20 <malloc>
   0x000000000040052c <+60>:    mov    %rax,-0x20(%rbp)
   0x0000000000400530 <+64>:    mov    $0x400,%edi
   0x0000000000400535 <+69>:    callq  0x416d20 <malloc>
   0x000000000040053a <+74>:    mov    %rax,-0x28(%rbp)
   0x000000000040053e <+78>:    mov    $0x400,%edi
   0x0000000000400543 <+83>:    callq  0x416d20 <malloc>
   0x0000000000400548 <+88>:    mov    %rax,-0x30(%rbp)
   0x000000000040054c <+92>:    mov    $0x400,%edi
   0x0000000000400551 <+97>:    callq  0x416d20 <malloc>
---Type <return> to continue, or q <return> to quit---
   0x0000000000400556 <+102>:   mov    %rax,-0x38(%rbp)
   0x000000000040055a <+106>:   mov    -0x30(%rbp),%rax
   0x000000000040055e <+110>:   mov    %rax,%rdi
   0x0000000000400561 <+113>:   callq  0x416c50 <free>
   0x0000000000400566 <+118>:   mov    -0x20(%rbp),%rax
   0x000000000040056a <+122>:   mov    %rax,%rdi
   0x000000000040056d <+125>:   callq  0x416c50 <free>
   0x0000000000400572 <+130>:   mov    -0x10(%rbp),%rax
   0x0000000000400576 <+134>:   mov    %rax,%rdi
   0x0000000000400579 <+137>:   callq  0x416c50 <free>
   0x000000000040057e <+142>:   mov    -0x10(%rbp),%rax
   0x0000000000400582 <+146>:   movl   $0x6c6c6548,(%rax)
   0x0000000000400588 <+152>:   movl   $0x7243206f,0x4(%rax)
   0x000000000040058f <+159>:   movl   $0x21687361,0x8(%rax)
   0x0000000000400596 <+166>:   movb   $0x0,0xc(%rax)
   0x000000000040059a <+170>:   mov    -0x20(%rbp),%rax
   0x000000000040059e <+174>:   movl   $0x6c6c6548,(%rax)
   0x00000000004005a4 <+180>:   movl   $0x7243206f,0x4(%rax)
   0x00000000004005ab <+187>:   movl   $0x21687361,0x8(%rax)
   0x00000000004005b2 <+194>:   movb   $0x0,0xc(%rax)
   0x00000000004005b6 <+198>:   mov    -0x30(%rbp),%rax
   0x00000000004005ba <+202>:   movl   $0x6c6c6548,(%rax)
   0x00000000004005c0 <+208>:   movl   $0x7243206f,0x4(%rax)
   0x00000000004005c7 <+215>:   movl   $0x21687361,0x8(%rax)
---Type <return> to continue, or q <return> to quit---
   0x00000000004005ce <+222>:   movb   $0x0,0xc(%rax)
   0x00000000004005d2 <+226>:   mov    $0x200,%edi
   0x00000000004005d7 <+231>:   callq  0x416d20 <malloc>
   0x00000000004005dc <+236>:   mov    %rax,-0x10(%rbp)
   0x00000000004005e0 <+240>:   mov    $0x400,%edi
   0x00000000004005e5 <+245>:   callq  0x416d20 <malloc>
```

```
   0x00000000004005ea <+250>:    mov    %rax,-0x20(%rbp)
   0x00000000004005ee <+254>:    mov    $0x200,%edi
   0x00000000004005f3 <+259>:    callq  0x416d20 <malloc>
   0x00000000004005f8 <+264>:    mov    %rax,-0x30(%rbp)
   0x00000000004005fc <+268>:    mov    $0x12c,%edi
   0x0000000000400601 <+273>:    callq  0x42fd40 <sleep>
   0x0000000000400606 <+278>:    mov    -0x38(%rbp),%rax
   0x000000000040060a <+282>:    mov    %rax,%rdi
   0x000000000040060d <+285>:    callq  0x416c50 <free>
   0x0000000000400612 <+290>:    mov    -0x30(%rbp),%rax
   0x0000000000400616 <+294>:    mov    %rax,%rdi
   0x0000000000400619 <+297>:    callq  0x416c50 <free>
   0x000000000040061e <+302>:    mov    -0x28(%rbp),%rax
   0x0000000000400622 <+306>:    mov    %rax,%rdi
   0x0000000000400625 <+309>:    callq  0x416c50 <free>
   0x000000000040062a <+314>:    mov    -0x20(%rbp),%rax
   0x000000000040062e <+318>:    mov    %rax,%rdi
   0x0000000000400631 <+321>:    callq  0x416c50 <free>
---Type <return> to continue, or q <return> to quit---
   0x0000000000400636 <+326>:    mov    -0x18(%rbp),%rax
   0x000000000040063a <+330>:    mov    %rax,%rdi
   0x000000000040063d <+333>:    callq  0x416c50 <free>
   0x0000000000400642 <+338>:    mov    -0x10(%rbp),%rax
   0x0000000000400646 <+342>:    mov    %rax,%rdi
   0x0000000000400649 <+345>:    callq  0x416c50 <free>
   0x000000000040064e <+350>:    mov    -0x8(%rbp),%rax
   0x0000000000400652 <+354>:    mov    %rax,%rdi
   0x0000000000400655 <+357>:    callq  0x416c50 <free>
   0x000000000040065a <+362>:    mov    $0xffffffff,%edi
   0x000000000040065f <+367>:    callq  0x42fd40 <sleep>
   0x0000000000400664 <+372>:    leaveq
   0x0000000000400665 <+373>:    retq
End of assembler dump.
```

We see that before the problem *malloc* call there were three buffer writes to memory addresses pointed to by values located at the following addresses: rbp-0x10, rbp-0x20, and rbp-0x30 (highlighted in red in disassembly). However, before buffer writes there were *free* function calls with values located at the same addresses: rbp-0x30, rbp-0x20, and rbp-0x10 (highlighted in blue in disassembly).

5. We have the standard function prolog (highlighted in green in disassembly). Switch to the stack frame #2 to check the addresses, their values, and memory contents they point to:

```
(gdb) frame 2
#2  0x00000000004005dc in proc ()

(gdb) x/xg $rbp-0x10
0x7efd12f90d30:     0x0000000001ab5020

(gdb) x/s 0x1ab5020
0x1ab5020:    "Hello Crash!"

(gdb) x/xg $rbp-0x20
0x7efd12f90d20:     0x0000000001ab5840

(gdb) x/s 0x1ab5840
0x1ab5840:    "Hello Crash!"
```

6. We know the addresses passed to heap management functions, for example, 0x1abxxxx. Find the heap region in the section list:

```
(gdb) maintenance info sections
Exec file:
    `/home/training/ALCDA/App4/App4', file type elf64-x86-64.
    0x00400158->0x00400178 at 0x00000158: .note.ABI-tag ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x00400178->0x0040019c at 0x00000178: .note.gnu.build-id ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004001a0->0x004002d8 at 0x000001a0: .rela.plt ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004002d8->0x004002e6 at 0x000002d8: .init ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x004002f0->0x004003c0 at 0x000002f0: .plt ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x004003c0->0x0048b318 at 0x000003c0: .text ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x0048b320->0x0048be9e at 0x0008b320: __libc_freeres_fn ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x0048bea0->0x0048bf01 at 0x0008bea0: __libc_thread_freeres_fn ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x0048bf04->0x0048bf0d at 0x0008bf04: .fini ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x0048bf20->0x004a9e84 at 0x0008bf20: .rodata ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004a9e88->0x004a9ee8 at 0x000a9e88: __libc_subfreeres ALLOC LOAD READONLY DATA HAS_CONTENTS
---Type <return> to continue, or q <return> to quit---
    0x004a9ee8->0x004a9ef0 at 0x000a9ee8: __libc_atexit ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004a9ef0->0x004a9ef8 at 0x000a9ef0: __libc_thread_subfreeres ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004a9ef8->0x004b69ec at 0x000a9ef8: .eh_frame ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004b69ec->0x004b6b06 at 0x000b69ec: .gcc_except_table ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x006b6b08->0x006b6b30 at 0x000b6b08: .tdata ALLOC LOAD DATA HAS_CONTENTS
    0x006b6b30->0x006b6b60 at 0x000b6b30: .tbss ALLOC
    0x006b6b30->0x006b6b40 at 0x000b6b30: .init_array ALLOC LOAD DATA HAS_CONTENTS
    0x006b6b40->0x006b6b50 at 0x000b6b40: .fini_array ALLOC LOAD DATA HAS_CONTENTS
    0x006b6b50->0x006b6b58 at 0x000b6b50: .jcr ALLOC LOAD DATA HAS_CONTENTS
    0x006b6b60->0x006b6bd0 at 0x000b6b60: .data.rel.ro ALLOC LOAD DATA HAS_CONTENTS
    0x006b6bd0->0x006b6be0 at 0x000b6bd0: .got ALLOC LOAD DATA HAS_CONTENTS
    0x006b6be0->0x006b6c60 at 0x000b6be0: .got.plt ALLOC LOAD DATA HAS_CONTENTS
    0x006b6c60->0x006b7970 at 0x000b6c60: .data ALLOC LOAD DATA HAS_CONTENTS
    0x006b7980->0x006bece8 at 0x000b7970: .bss ALLOC
    0x006bece8->0x006bed18 at 0x000b7970: __libc_freeres_ptrs ALLOC
    0x00000000->0x00000038 at 0x000b7970: .comment READONLY HAS_CONTENTS
    0x00000000->0x00000390 at 0x000b79b0: .debug_aranges READONLY HAS_CONTENTS
---Type <return> to continue, or q <return> to quit---
    0x00000000->0x00000ac3 at 0x000b7d40: .debug_pubnames READONLY HAS_CONTENTS
    0x00000000->0x00011440 at 0x000b8803: .debug_info READONLY HAS_CONTENTS
    0x00000000->0x000021b1 at 0x000c9c43: .debug_abbrev READONLY HAS_CONTENTS
    0x00000000->0x00002ebc at 0x000cbdf4: .debug_line READONLY HAS_CONTENTS
    0x00000000->0x000038da at 0x000cecb0: .debug_str READONLY HAS_CONTENTS
    0x00000000->0x0000878e at 0x000d258a: .debug_loc READONLY HAS_CONTENTS
    0x00000000->0x00001280 at 0x000dad18: .debug_ranges READONLY HAS_CONTENTS
Core file:
    `/home/training/ALCDA/./App4/core', file type elf64-x86-64.
    0x00000000->0x00002aa8 at 0x00000430: note0 READONLY HAS_CONTENTS
    0x00000000->0x000000d8 at 0x000004b4: .reg/11060 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x000004b4: .reg HAS_CONTENTS
    0x00000000->0x00000130 at 0x00000644: .auxv HAS_CONTENTS
    0x00000000->0x00000200 at 0x00000788: .reg2/11060 HAS_CONTENTS
    0x00000000->0x00000200 at 0x00000788: .reg2 HAS_CONTENTS
    0x00000000->0x00000340 at 0x0000099c: .reg-xstate/11060 HAS_CONTENTS
    0x00000000->0x00000340 at 0x0000099c: .reg-xstate HAS_CONTENTS
    0x00000000->0x000000d8 at 0x00000d60: .reg/11057 HAS_CONTENTS
    0x00000000->0x00000200 at 0x00000e54: .reg2/11057 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00001068: .reg-xstate/11057 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x0000142c: .reg/11062 HAS_CONTENTS
    0x00000000->0x00000200 at 0x00001520: .reg2/11062 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00001734: .reg-xstate/11062 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x00001af8: .reg/11061 HAS_CONTENTS
---Type <return> to continue, or q <return> to quit---
    0x00000000->0x00000200 at 0x00001bec: .reg2/11061 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00001e00: .reg-xstate/11061 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x000021c4: .reg/11059 HAS_CONTENTS
    0x00000000->0x00000200 at 0x000022b8: .reg2/11059 HAS_CONTENTS
    0x00000000->0x00000340 at 0x000024cc: .reg-xstate/11059 HAS_CONTENTS
```

```
    0x00000000->0x000000d8 at 0x00002890: .reg/11058 HAS_CONTENTS
    0x00000000->0x00000200 at 0x00002984: .reg2/11058 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00002b98: .reg-xstate/11058 HAS_CONTENTS
    0x00400000->0x00401000 at 0x00003000: load1a ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x00401000->0x00401000 at 0x00004000: load1b ALLOC READONLY CODE
    0x006b6000->0x006b8000 at 0x00004000: load2 ALLOC LOAD HAS_CONTENTS
    0x006b8000->0x006bf000 at 0x00006000: load3 ALLOC LOAD HAS_CONTENTS
    0x01ab2000->0x01ad5000 at 0x0000d000: load4 ALLOC LOAD HAS_CONTENTS
    0x7efd1178f000->0x7efd11790000 at 0x00030000: load5 ALLOC LOAD READONLY HAS_CONTENTS
    0x7efd11790000->0x7efd11f90000 at 0x00031000: load6 ALLOC LOAD HAS_CONTENTS
    0x7efd11f90000->0x7efd11f91000 at 0x00831000: load7 ALLOC LOAD READONLY HAS_CONTENTS
    0x7efd11f91000->0x7efd12791000 at 0x00832000: load8 ALLOC LOAD HAS_CONTENTS
    0x7efd12791000->0x7efd12792000 at 0x01032000: load9 ALLOC LOAD READONLY HAS_CONTENTS
    0x7efd12792000->0x7efd12f92000 at 0x01033000: load10 ALLOC LOAD HAS_CONTENTS
---Type <return> to continue, or q <return> to quit---
    0x7efd12f92000->0x7efd12f93000 at 0x01833000: load11 ALLOC LOAD READONLY HAS_CONTENTS
    0x7efd12f93000->0x7efd13793000 at 0x01834000: load12 ALLOC LOAD HAS_CONTENTS
    0x7efd13793000->0x7efd13794000 at 0x02034000: load13 ALLOC LOAD READONLY HAS_CONTENTS
    0x7efd13794000->0x7efd13f94000 at 0x02035000: load14 ALLOC LOAD HAS_CONTENTS
    0x7fff1b488000->0x7fff1b4aa000 at 0x02835000: load15 ALLOC LOAD HAS_CONTENTS
    0x7fff1b544000->0x7fff1b545000 at 0x02857000: load16 ALLOC LOAD READONLY CODE HAS_CONTENTS
    0xffffffffff600000->0xffffffffff600000 at 0x02858000: load17 ALLOC READONLY CODE
```

7. Check the faulting instruction and the problem memory address:

```
(gdb) bt
#0  0x000000000041482e in _int_malloc ()
#1  0x0000000000416d88 in malloc ()
#2  0x00000000004005dc in proc ()
#3  0x00000000004006ee in bar_three ()
#4  0x00000000004006fe in foo_three ()
#5  0x0000000000400716 in thread_three ()
#6  0x0000000000401760 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#7  0x0000000000432609 in clone ()
#8  0x0000000000000000 in ?? ()

(gdb) x/i $rip
=> 0x41482e <_int_malloc+622>:   mov    %rbx,0x10(%r12)

(gdb) x $r12+0x10
 0x21687371: Cannot access memory at address 0x21687371

(gdb) p (char[4])0x21687371
$1 = "qsh!"
```

We see that “sh!” fragment correlates with “Hello Crash!” buffer overwrite that we saw previously.

Exercise A5

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

© 2015 Software Diagnostics Services

Exercise A5

Goal: Learn how to identify stack corruption.

Patterns: Local Buffer Overflow, Execution Residue.

1. Load a core dump and *App5* executable:

```
training@debian64:~/ALCDA$ gdb -c ./App5/core -se ./App5/App5
GNU gdb (GDB) 7.4.1-debian
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/training/ALCDA/App5/App5...done.
[New LWP 12864]
[New LWP 12863]
[New LWP 12865]
[New LWP 12868]
[New LWP 12867]
[New LWP 12866]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Core was generated by `./App5'.
Program terminated with signal 11, Segmentation fault.
#0  0x0000000000000000 in ?? ()
```

2. List threads and show stack trace of the problem thread:

```
(gdb) info threads
  Id   Target Id         Frame
  6    Thread 0x7fc3dc9dd700 (LWP 12866) 0x000000000042fed1 in nanosleep ()
  5    Thread 0x7fc3dc1dc700 (LWP 12867) 0x000000000042fed1 in nanosleep ()
  4    Thread 0x7fc3db9db700 (LWP 12868) 0x000000000042fed1 in nanosleep ()
  3    Thread 0x7fc3dd1de700 (LWP 12865) 0x000000000042fed1 in nanosleep ()
  2    Thread 0x8ee860 (LWP 12863) 0x000000000042fed1 in nanosleep ()
* 1    Thread 0x7fc3dd9df700 (LWP 12864) 0x0000000000000000 in ?? ()

(gdb) bt
#0  0x0000000000000000 in ?? ()
#1  0x0000000000000000 in ?? ()
```

3. We don’t see expected stack trace frames as in a normal thread stack trace:

```
(gdb) thread apply 3 bt

Thread 3 (Thread 0x7fc3dd1de700 (LWP 12865)):
#0  0x000000000042fed1 in nanosleep ()
#1  0x000000000042fda0 in sleep ()
#2  0x0000000000400619 in bar_two ()
#3  0x0000000000400629 in foo_two ()
#4  0x0000000000400641 in thread_two ()
#5  0x00000000004016d0 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#6  0x0000000000432589 in clone ()
#7  0x0000000000000000 in ?? ()
```

4. Dump raw stack data around the current stack pointer and find an ASCII buffer around a return address:

```
(gdb) x/100a $rsp
0x7fc3dd9debc8:     0x0    0x0
0x7fc3dd9debd8:     0x0    0x0
0x7fc3dd9debe8:     0x0    0x0
0x7fc3dd9debf8:     0x0    0x0
0x7fc3dd9dec08:     0x0    0x0
0x7fc3dd9dec18:     0x0    0x0
0x7fc3dd9dec28:     0x0    0x0
0x7fc3dd9dec38:     0x0    0x0
0x7fc3dd9dec48:     0x0    0x0
0x7fc3dd9dec58:     0x0    0x0
0x7fc3dd9dec68:     0x0    0x0
0x7fc3dd9dec78:     0x0    0x0
0x7fc3dd9dec88:     0x0    0x0
0x7fc3dd9dec98:     0x0    0x0
0x7fc3dd9deca8:     0x0    0x7fc3dd9ded38
0x7fc3dd9decb8:     0x422077654e20794d  0x7542207265676769
0x7fc3dd9decc8:     0x72656666   0x0
0x7fc3dd9decd8:     0x0    0x0
0x7fc3dd9dece8:     0x0    0x0
0x7fc3dd9decf8:     0x0    0x0
0x7fc3dd9ded08:     0x0    0x0
0x7fc3dd9ded18:     0x0    0x0
0x7fc3dd9ded28:     0x7fc3dd9ded48      0x4005cc <procA+40>
0x7fc3dd9ded38:     0x422077654e20794d  0x7542207265676769
---Type <return> to continue, or q <return> to quit---
0x7fc3dd9ded48:     0x72656666   0x0
0x7fc3dd9ded58:     0x0    0x0
0x7fc3dd9ded68:     0x0    0x0
0x7fc3dd9ded78:     0x0    0x0
0x7fc3dd9ded88:     0x0    0x0
0x7fc3dd9ded98:     0x7fc300000000      0x0
0x7fc3dd9deda8:     0x0    0x0
0x7fc3dd9dedb8:     0x0    0x0
0x7fc3dd9dedc8:     0x0    0x0
0x7fc3dd9dedd8:     0x0    0x0
0x7fc3dd9dede8:     0x0    0x0
0x7fc3dd9dedf8:     0x0    0x0
0x7fc3dd9dee08:     0x0    0x0
0x7fc3dd9dee18:     0x0    0x7fc3dd9df700
0x7fc3dd9dee28:     0x722f707d72b64fb1  0x48c1a0 <default_attr>
0x7fc3dd9dee38:     0x7fc3dd9df9c0      0x0
0x7fc3dd9dee48:     0x3    0x8da8cb46a9964fb1
```

```
0x7fc3dd9dee58:     0x722f70fd5f9a4fb1  0x0
0x7fc3dd9dee68:     0x0    0x0
0x7fc3dd9dee78:     0x0    0x0
0x7fc3dd9dee88:     0x0    0x7fc3dd9df700
0x7fc3dd9dee98:     0x432589 <clone+121>      0x0
0x7fc3dd9deea8:     0x0    0x0
0x7fc3dd9deeb8:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7fc3dd9deec8:     0x0    0x0
0x7fc3dd9deed8:     0x0    0x0

(gdb) x/s 0x7fc3dd9ded38
0x7fc3dd9ded38:      "My New Bigger Buffer"
```

Exercise A6

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

© 2015 Software Diagnostics Services

Exercise A6

Goal: Learn how to identify stack overflow, stack boundaries, reconstruct stack trace.

Patterns: Stack Overflow, Execution Residue.

1. Load a core dump and *App6* executable:

```
training@debian64:~/ALCDA$ gdb -c ./App6/core -se ./App6/App6
GNU gdb (GDB) 7.4.1-debian
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/training/ALCDA/App6/App6...done.
[New LWP 13251]
[New LWP 13252]
[New LWP 13253]
[New LWP 13254]
[New LWP 13255]
[New LWP 13250]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Core was generated by `./App6'.
Program terminated with signal 11, Segmentation fault.
#0  0x00000000004004fb in procF ()
```

2. List threads:

```
(gdb) info threads
  Id   Target Id         Frame
  6    Thread 0x199f860 (LWP 13250) 0x000000000042fe91 in nanosleep ()
  5    Thread 0x7eff44905700 (LWP 13255) 0x000000000042fe91 in nanosleep ()
  4    Thread 0x7eff45106700 (LWP 13254) 0x000000000042fe91 in nanosleep ()
  3    Thread 0x7eff45907700 (LWP 13253) 0x000000000042fe91 in nanosleep ()
  2    Thread 0x7eff46108700 (LWP 13252) 0x000000000042fe91 in nanosleep ()
* 1    Thread 0x7eff46909700 (LWP 13251) 0x00000000004004fb in procF ()
```

3. If we try to print the problem stack trace we get the endless number of frames, so we quit:

```
(gdb) bt
#0  0x00000000004004fb in procF ()
#1  0x000000000040054b in procF ()
#2  0x000000000040054b in procF ()
#3  0x000000000040054b in procF ()
#4  0x000000000040054b in procF ()
#5  0x000000000040054b in procF ()
#6  0x000000000040054b in procF ()
#7  0x000000000040054b in procF ()
#8  0x000000000040054b in procF ()
#9  0x000000000040054b in procF ()
#10 0x000000000040054b in procF ()
```

```
#11 0x000000000040054b in procF ()
#12 0x000000000040054b in procF ()
#13 0x000000000040054b in procF ()
#14 0x000000000040054b in procF ()
#15 0x000000000040054b in procF ()
#16 0x000000000040054b in procF ()
#17 0x000000000040054b in procF ()
#18 0x000000000040054b in procF ()
#19 0x000000000040054b in procF ()
#20 0x000000000040054b in procF ()
#21 0x000000000040054b in procF ()
#22 0x000000000040054b in procF ()
#23 0x000000000040054b in procF ()
---Type <return> to continue, or q <return> to quit---
#24 0x000000000040054b in procF ()
#25 0x000000000040054b in procF ()
#26 0x000000000040054b in procF ()
#27 0x000000000040054b in procF ()
#28 0x000000000040054b in procF ()
#29 0x000000000040054b in procF ()
#30 0x000000000040054b in procF ()
#31 0x000000000040054b in procF ()
#32 0x000000000040054b in procF ()
#33 0x000000000040054b in procF ()
#34 0x000000000040054b in procF ()
#35 0x000000000040054b in procF ()
#36 0x000000000040054b in procF ()
#37 0x000000000040054b in procF ()
#38 0x000000000040054b in procF ()
#39 0x000000000040054b in procF ()
#40 0x000000000040054b in procF ()
#41 0x000000000040054b in procF ()
#42 0x000000000040054b in procF ()
#43 0x000000000040054b in procF ()
#44 0x000000000040054b in procF ()
#45 0x000000000040054b in procF ()
#46 0x000000000040054b in procF ()
#47 0x000000000040054b in procF ()
---Type <return> to continue, or q <return> to quit---
#48 0x000000000040054b in procF ()
#49 0x000000000040054b in procF ()
#50 0x000000000040054b in procF ()
#51 0x000000000040054b in procF ()
#52 0x000000000040054b in procF ()
#53 0x000000000040054b in procF ()
#54 0x000000000040054b in procF ()
#55 0x000000000040054b in procF ()
#56 0x000000000040054b in procF ()
#57 0x000000000040054b in procF ()
#58 0x000000000040054b in procF ()
#59 0x000000000040054b in procF ()
#60 0x000000000040054b in procF ()
#61 0x000000000040054b in procF ()
#62 0x000000000040054b in procF ()
#63 0x000000000040054b in procF ()
#64 0x000000000040054b in procF ()
#65 0x000000000040054b in procF ()
#66 0x000000000040054b in procF ()
#67 0x000000000040054b in procF ()
#68 0x000000000040054b in procF ()
```

```
#69 0x000000000040054b in procF ()
#70 0x000000000040054b in procF ()
#71 0x000000000040054b in procF ()
---Type <return> to continue, or q <return> to quit---
#72 0x000000000040054b in procF ()
#73 0x000000000040054b in procF ()
#74 0x000000000040054b in procF ()
#75 0x000000000040054b in procF ()
#76 0x000000000040054b in procF ()
#77 0x000000000040054b in procF ()
#78 0x000000000040054b in procF ()
#79 0x000000000040054b in procF ()
#80 0x000000000040054b in procF ()
#81 0x000000000040054b in procF ()
#82 0x000000000040054b in procF ()
#83 0x000000000040054b in procF ()
#84 0x000000000040054b in procF ()
#85 0x000000000040054b in procF ()
#86 0x000000000040054b in procF ()
#87 0x000000000040054b in procF ()
#88 0x000000000040054b in procF ()
#89 0x000000000040054b in procF ()
#90 0x000000000040054b in procF ()
#91 0x000000000040054b in procF ()
#92 0x000000000040054b in procF ()
#93 0x000000000040054b in procF ()
#94 0x000000000040054b in procF ()
#95 0x000000000040054b in procF ()
---Type <return> to continue, or q <return> to quit---
#96 0x000000000040054b in procF ()
#97 0x000000000040054b in procF ()
#98 0x000000000040054b in procF ()
#99 0x000000000040054b in procF ()
#100 0x000000000040054b in procF ()
#101 0x000000000040054b in procF ()
#102 0x000000000040054b in procF ()
#103 0x000000000040054b in procF ()
#104 0x000000000040054b in procF ()
#105 0x000000000040054b in procF ()
#106 0x000000000040054b in procF ()
#107 0x000000000040054b in procF ()
#108 0x000000000040054b in procF ()
#109 0x000000000040054b in procF ()
#110 0x000000000040054b in procF ()
#111 0x000000000040054b in procF ()
#112 0x000000000040054b in procF ()
#113 0x000000000040054b in procF ()
#114 0x000000000040054b in procF ()
#115 0x000000000040054b in procF ()
#116 0x000000000040054b in procF ()
#117 0x000000000040054b in procF ()
#118 0x000000000040054b in procF ()
#119 0x000000000040054b in procF ()
---Type <return> to continue, or q <return> to quit---q
Quit
```

It looks like a stack overflow.

4. Check if this is a stack overflow indeed. Stack region can be identified from *pmap.13250* from the thread number. Since the problem thread has LWP 13251 it should be located just below the main stack region:

```
13250:   ./App6
0000000000400000    732K r-x--  /home/training/ALCDA/App6/App6
00000000006b6000      8K rw---  /home/training/ALCDA/App6/App6
00000000006b8000     28K rw---    [ anon ]
000000000199f000    140K rw---    [ anon ]
00007eff44105000      4K -----    [ anon ]
00007eff44106000   8192K rw---    [ anon ]
00007eff44906000      4K -----    [ anon ]
00007eff44907000   8192K rw---    [ anon ]
00007eff45107000      4K -----    [ anon ]
00007eff45108000   8192K rw---    [ anon ]
00007eff45908000      4K -----    [ anon ]
00007eff45909000   8192K rw---    [ anon ]
00007eff46109000      4K -----    [ anon ]
00007eff4610a000   8192K rw---    [ anon ]
00007ffde0705000    132K rw---    [ stack ]
00007ffde0788000      4K r-x--    [ anon ]
ffffffffff600000      4K r-x--    [ anon ]
 total            42028K
```

5. Check that manually based on the stack pointer value and section bundary addresses:

```
(gdb) x $rsp
0x7eff46109ec0:     0x0

(gdb) frame 1
#1  0x000000000040054b in procF ()

(gdb) x $rsp
0x7eff4610a0e0:     0x0

(gdb) frame 2
#2  0x000000000040054b in procF ()

(gdb) x $rsp
0x7eff4610a300:     0x0

(gdb) maintenance info sections
Exec file:
    `/home/training/ALCDA/App6/App6', file type elf64-x86-64.
    0x00400158->0x00400178 at 0x00000158: .note.ABI-tag ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x00400178->0x0040019c at 0x00000178: .note.gnu.build-id ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004001a0->0x004002d8 at 0x000001a0: .rela.plt ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004002d8->0x004002e6 at 0x000002d8: .init ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x004002f0->0x004003c0 at 0x000002f0: .plt ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x004003c0->0x0048b258 at 0x000003c0: .text ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x0048b260->0x0048bdde at 0x0008b260: __libc_freeres_fn ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x0048bde0->0x0048be41 at 0x0008bde0: __libc_thread_freeres_fn ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x0048be44->0x0048be4d at 0x0008be44: .fini ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x0048be60->0x004a9dc4 at 0x0008be60: .rodata ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004a9dc8->0x004a9e28 at 0x000a9dc8: __libc_subfreeres ALLOC LOAD READONLY DATA HAS_CONTENTS
---Type <return> to continue, or q <return> to quit---
    0x004a9e28->0x004a9e30 at 0x000a9e28: __libc_atexit ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004a9e30->0x004a9e38 at 0x000a9e30: __libc_thread_subfreeres ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004a9e38->0x004b694c at 0x000a9e38: .eh_frame ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004b694c->0x004b6a66 at 0x000b694c: .gcc_except_table ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x006b6a68->0x006b6a90 at 0x000b6a68: .tdata ALLOC LOAD DATA HAS_CONTENTS
    0x006b6a90->0x006b6ac0 at 0x000b6a90: .tbss ALLOC
```

```
    0x006b6a90->0x006b6aa0 at 0x000b6a90: .init_array ALLOC LOAD DATA HAS_CONTENTS
    0x006b6aa0->0x006b6ab0 at 0x000b6aa0: .fini_array ALLOC LOAD DATA HAS_CONTENTS
    0x006b6ab0->0x006b6ab8 at 0x000b6ab0: .jcr ALLOC LOAD DATA HAS_CONTENTS
    0x006b6ac0->0x006b6b30 at 0x000b6ac0: .data.rel.ro ALLOC LOAD DATA HAS_CONTENTS
    0x006b6b30->0x006b6b40 at 0x000b6b30: .got ALLOC LOAD DATA HAS_CONTENTS
    0x006b6b40->0x006b6bc0 at 0x000b6b40: .got.plt ALLOC LOAD DATA HAS_CONTENTS
    0x006b6bc0->0x006b78d0 at 0x000b6bc0: .data ALLOC LOAD DATA HAS_CONTENTS
    0x006b78e0->0x006bec48 at 0x000b78d0: .bss ALLOC
    0x006bec48->0x006bec78 at 0x000b78d0: __libc_freeres_ptrs ALLOC
    0x00000000->0x00000038 at 0x000b78d0: .comment READONLY HAS_CONTENTS
    0x00000000->0x00000390 at 0x000b7910: .debug_aranges READONLY HAS_CONTENTS
---Type <return> to continue, or q <return> to quit---
    0x00000000->0x00000ac3 at 0x000b7ca0: .debug_pubnames READONLY HAS_CONTENTS
    0x00000000->0x00011440 at 0x000b8763: .debug_info READONLY HAS_CONTENTS
    0x00000000->0x000021b1 at 0x000c9ba3: .debug_abbrev READONLY HAS_CONTENTS
    0x00000000->0x00002ebc at 0x000cbd54: .debug_line READONLY HAS_CONTENTS
    0x00000000->0x000038da at 0x000cec10: .debug_str READONLY HAS_CONTENTS
    0x00000000->0x0000878e at 0x000d24ea: .debug_loc READONLY HAS_CONTENTS
    0x00000000->0x00001280 at 0x000dac78: .debug_ranges READONLY HAS_CONTENTS
Core file:
    `/home/training/ALCDA/./App6/core', file type elf64-x86-64.
    0x00000000->0x00002aa8 at 0x00000430: note0 READONLY HAS_CONTENTS
    0x00000000->0x000000d8 at 0x000004b4: .reg/13251 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x000004b4: .reg HAS_CONTENTS
    0x00000000->0x00000130 at 0x00000644: .auxv HAS_CONTENTS
    0x00000000->0x00000200 at 0x00000788: .reg2/13251 HAS_CONTENTS
    0x00000000->0x00000200 at 0x00000788: .reg2 HAS_CONTENTS
    0x00000000->0x00000340 at 0x0000099c: .reg-xstate/13251 HAS_CONTENTS
    0x00000000->0x00000340 at 0x0000099c: .reg-xstate HAS_CONTENTS
    0x00000000->0x000000d8 at 0x00000d60: .reg/13252 HAS_CONTENTS
    0x00000000->0x00000200 at 0x00000e54: .reg2/13252 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00001068: .reg-xstate/13252 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x0000142c: .reg/13253 HAS_CONTENTS
    0x00000000->0x00000200 at 0x00001520: .reg2/13253 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00001734: .reg-xstate/13253 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x00001af8: .reg/13254 HAS_CONTENTS
---Type <return> to continue, or q <return> to quit---
    0x00000000->0x00000200 at 0x00001bec: .reg2/13254 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00001e00: .reg-xstate/13254 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x000021c4: .reg/13255 HAS_CONTENTS
    0x00000000->0x00000200 at 0x000022b8: .reg2/13255 HAS_CONTENTS
    0x00000000->0x00000340 at 0x000024cc: .reg-xstate/13255 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x00002890: .reg/13250 HAS_CONTENTS
    0x00000000->0x00000200 at 0x00002984: .reg2/13250 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00002b98: .reg-xstate/13250 HAS_CONTENTS
    0x00400000->0x00401000 at 0x00003000: load1a ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x00401000->0x00401000 at 0x00004000: load1b ALLOC READONLY CODE
    0x006b6000->0x006b8000 at 0x00004000: load2 ALLOC LOAD HAS_CONTENTS
    0x006b8000->0x006bf000 at 0x00006000: load3 ALLOC LOAD HAS_CONTENTS
    0x0199f000->0x019c2000 at 0x0000d000: load4 ALLOC LOAD HAS_CONTENTS
    0x7eff44105000->0x7eff44106000 at 0x00030000: load5 ALLOC LOAD READONLY HAS_CONTENTS
    0x7eff44106000->0x7eff44906000 at 0x00031000: load6 ALLOC LOAD HAS_CONTENTS
    0x7eff44906000->0x7eff44907000 at 0x00831000: load7 ALLOC LOAD READONLY HAS_CONTENTS
    0x7eff44907000->0x7eff45107000 at 0x00832000: load8 ALLOC LOAD HAS_CONTENTS
    0x7eff45107000->0x7eff45108000 at 0x01032000: load9 ALLOC LOAD READONLY HAS_CONTENTS
    0x7eff45108000->0x7eff45908000 at 0x01033000: load10 ALLOC LOAD HAS_CONTENTS
---Type <return> to continue, or q <return> to quit---
    0x7eff45908000->0x7eff45909000 at 0x01833000: load11 ALLOC LOAD READONLY HAS_CONTENTS
    0x7eff45909000->0x7eff46109000 at 0x01834000: load12 ALLOC LOAD HAS_CONTENTS
    0x7eff46109000->0x7eff4610a000 at 0x02034000: load13 ALLOC LOAD READONLY HAS_CONTENTS
    0x7eff4610a000->0x7eff4690a000 at 0x02035000: load14 ALLOC LOAD HAS_CONTENTS
    0x7ffde0704000->0x7ffde0726000 at 0x02835000: load15 ALLOC LOAD HAS_CONTENTS
    0x7ffde0788000->0x7ffde0789000 at 0x02857000: load16 ALLOC LOAD READONLY CODE HAS_CONTENTS
    0xffffffffff600000->0xffffffffff600000 at 0x02858000: load17 ALLOC READONLY CODE
```

6. Dump the bottom of the raw stack to see execution residue such as thread startup:

```
(gdb) x/1024a 0x7eff4690a000-0x2000
0x7eff46908000:     0x0    0x0
0x7eff46908010:     0x0    0x0
0x7eff46908020:     0x0    0x0
0x7eff46908030:     0x0    0x0
0x7eff46908040:     0x0    0x0
0x7eff46908050:     0x0    0x0
0x7eff46908060:     0x0    0x0
0x7eff46908070:     0x7eff46908290      0x40054b <procF+91>
0x7eff46908080:     0x0    0x600000000
0x7eff46908090:     0xffffffff   0x7
0x7eff469080a0:     0xffffffff   0x0
0x7eff469080b0:     0x0    0x0
0x7eff469080c0:     0x0    0x0
0x7eff469080d0:     0x0    0x0
0x7eff469080e0:     0x0    0x0
0x7eff469080f0:     0x0    0x0
0x7eff46908100:     0x0    0x0
0x7eff46908110:     0x0    0x0
0x7eff46908120:     0x0    0x0
0x7eff46908130:     0x0    0x0
0x7eff46908140:     0x0    0x0
0x7eff46908150:     0x0    0x0
0x7eff46908160:     0x0    0x0
0x7eff46908170:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7eff46908180:     0x0    0x0
0x7eff46908190:     0x0    0x0
0x7eff469081a0:     0x0    0x0
0x7eff469081b0:     0x0    0x0
0x7eff469081c0:     0x0    0x0
0x7eff469081d0:     0x0    0x0
0x7eff469081e0:     0x0    0x0
0x7eff469081f0:     0x0    0x0
0x7eff46908200:     0x0    0x0
0x7eff46908210:     0x0    0x0
0x7eff46908220:     0x0    0x0
0x7eff46908230:     0x0    0x0
0x7eff46908240:     0x0    0x0
0x7eff46908250:     0x0    0x0
0x7eff46908260:     0x0    0x0
0x7eff46908270:     0x0    0x0
0x7eff46908280:     0x0    0x0
0x7eff46908290:     0x7eff469084b0      0x40054b <procF+91>
0x7eff469082a0:     0x0    0x500000000
0x7eff469082b0:     0xffffffff   0x6
0x7eff469082c0:     0xffffffff   0x0
0x7eff469082d0:     0x0    0x0
0x7eff469082e0:     0x0    0x0
0x7eff469082f0:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7eff46908300:     0x0    0x0
0x7eff46908310:     0x0    0x0
0x7eff46908320:     0x0    0x0
0x7eff46908330:     0x0    0x0
0x7eff46908340:     0x0    0x0
0x7eff46908350:     0x0    0x0
0x7eff46908360:     0x0    0x0
```

```
0x7eff46908370:     0x0    0x0
0x7eff46908380:     0x0    0x0
0x7eff46908390:     0x0    0x0
0x7eff469083a0:     0x0    0x0
0x7eff469083b0:     0x0    0x0
0x7eff469083c0:     0x0    0x0
0x7eff469083d0:     0x0    0x0
0x7eff469083e0:     0x0    0x0
0x7eff469083f0:     0x0    0x0
0x7eff46908400:     0x0    0x0
0x7eff46908410:     0x0    0x0
0x7eff46908420:     0x0    0x0
0x7eff46908430:     0x0    0x0
0x7eff46908440:     0x0    0x0
0x7eff46908450:     0x0    0x0
0x7eff46908460:     0x0    0x0
0x7eff46908470:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7eff46908480:     0x0    0x0
0x7eff46908490:     0x0    0x0
0x7eff469084a0:     0x0    0x0
0x7eff469084b0:     0x7eff469086d0      0x40054b <procF+91>
0x7eff469084c0:     0x0    0x400000000
0x7eff469084d0:     0xffffffff   0x5
0x7eff469084e0:     0xffffffff   0x0
0x7eff469084f0:     0x0    0x0
0x7eff46908500:     0x0    0x0
0x7eff46908510:     0x0    0x0
0x7eff46908520:     0x0    0x0
0x7eff46908530:     0x0    0x0
0x7eff46908540:     0x0    0x0
0x7eff46908550:     0x0    0x0
0x7eff46908560:     0x0    0x0
0x7eff46908570:     0x0    0x0
0x7eff46908580:     0x0    0x0
0x7eff46908590:     0x0    0x0
0x7eff469085a0:     0x0    0x0
0x7eff469085b0:     0x0    0x0
0x7eff469085c0:     0x0    0x0
0x7eff469085d0:     0x0    0x0
0x7eff469085e0:     0x0    0x0
0x7eff469085f0:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7eff46908600:     0x0    0x0
0x7eff46908610:     0x0    0x0
0x7eff46908620:     0x0    0x0
0x7eff46908630:     0x0    0x0
0x7eff46908640:     0x0    0x0
0x7eff46908650:     0x0    0x0
0x7eff46908660:     0x0    0x0
0x7eff46908670:     0x0    0x0
0x7eff46908680:     0x0    0x0
0x7eff46908690:     0x0    0x0
0x7eff469086a0:     0x0    0x0
0x7eff469086b0:     0x0    0x0
0x7eff469086c0:     0x0    0x0
0x7eff469086d0:     0x7eff469088f0      0x40054b <procF+91>
0x7eff469086e0:     0x0    0x300000000
0x7eff469086f0:     0xffffffff   0x4
0x7eff46908700:     0xffffffff   0x0
```

```
0x7eff46908710:     0x0    0x0
0x7eff46908720:     0x0    0x0
0x7eff46908730:     0x0    0x0
0x7eff46908740:     0x0    0x0
0x7eff46908750:     0x0    0x0
0x7eff46908760:     0x0    0x0
0x7eff46908770:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7eff46908780:     0x0    0x0
0x7eff46908790:     0x0    0x0
0x7eff469087a0:     0x0    0x0
0x7eff469087b0:     0x0    0x0
0x7eff469087c0:     0x0    0x0
0x7eff469087d0:     0x0    0x0
0x7eff469087e0:     0x0    0x0
0x7eff469087f0:     0x0    0x0
0x7eff46908800:     0x0    0x0
0x7eff46908810:     0x0    0x0
0x7eff46908820:     0x0    0x0
0x7eff46908830:     0x0    0x0
0x7eff46908840:     0x0    0x0
0x7eff46908850:     0x0    0x0
0x7eff46908860:     0x0    0x0
0x7eff46908870:     0x0    0x0
0x7eff46908880:     0x0    0x0
0x7eff46908890:     0x0    0x0
0x7eff469088a0:     0x0    0x0
0x7eff469088b0:     0x0    0x0
0x7eff469088c0:     0x0    0x0
0x7eff469088d0:     0x0    0x0
0x7eff469088e0:     0x0    0x0
0x7eff469088f0:     0x7eff46908b10      0x40054b <procF+91>
---Type <return> to continue, or q <return> to quit---
0x7eff46908900:     0x0    0x200000000
0x7eff46908910:     0xffffffff   0x3
0x7eff46908920:     0xffffffff   0x0
0x7eff46908930:     0x0    0x0
0x7eff46908940:     0x0    0x0
0x7eff46908950:     0x0    0x0
0x7eff46908960:     0x0    0x0
0x7eff46908970:     0x0    0x0
0x7eff46908980:     0x0    0x0
0x7eff46908990:     0x0    0x0
0x7eff469089a0:     0x0    0x0
0x7eff469089b0:     0x0    0x0
0x7eff469089c0:     0x0    0x0
0x7eff469089d0:     0x0    0x0
0x7eff469089e0:     0x0    0x0
0x7eff469089f0:     0x0    0x0
0x7eff46908a00:     0x0    0x0
0x7eff46908a10:     0x0    0x0
0x7eff46908a20:     0x0    0x0
0x7eff46908a30:     0x0    0x0
0x7eff46908a40:     0x0    0x0
0x7eff46908a50:     0x0    0x0
0x7eff46908a60:     0x0    0x0
0x7eff46908a70:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7eff46908a80:     0x0    0x0
0x7eff46908a90:     0x0    0x0
```

```
0x7eff46908aa0:     0x0    0x0
0x7eff46908ab0:     0x0    0x0
0x7eff46908ac0:     0x0    0x0
0x7eff46908ad0:     0x0    0x0
0x7eff46908ae0:     0x0    0x0
0x7eff46908af0:     0x0    0x0
0x7eff46908b00:     0x0    0x0
0x7eff46908b10:     0x7eff46908d30      0x40054b <procF+91>
0x7eff46908b20:     0x0    0x100000000
0x7eff46908b30:     0xffffffff   0x2
0x7eff46908b40:     0xffffffff   0x0
0x7eff46908b50:     0x0    0x0
0x7eff46908b60:     0x0    0x0
0x7eff46908b70:     0x0    0x0
0x7eff46908b80:     0x0    0x0
0x7eff46908b90:     0x0    0x0
0x7eff46908ba0:     0x0    0x0
0x7eff46908bb0:     0x0    0x0
0x7eff46908bc0:     0x0    0x0
0x7eff46908bd0:     0x0    0x0
0x7eff46908be0:     0x0    0x0
0x7eff46908bf0:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7eff46908c00:     0x0    0x0
0x7eff46908c10:     0x0    0x0
0x7eff46908c20:     0x0    0x0
0x7eff46908c30:     0x0    0x0
0x7eff46908c40:     0x0    0x0
0x7eff46908c50:     0x0    0x0
0x7eff46908c60:     0x0    0x0
0x7eff46908c70:     0x0    0x0
0x7eff46908c80:     0x0    0x0
0x7eff46908c90:     0x0    0x0
0x7eff46908ca0:     0x0    0x0
0x7eff46908cb0:     0x0    0x0
0x7eff46908cc0:     0x0    0x0
0x7eff46908cd0:     0x0    0x0
0x7eff46908ce0:     0x0    0x0
0x7eff46908cf0:     0x0    0x0
0x7eff46908d00:     0x0    0x0
0x7eff46908d10:     0x0    0x0
0x7eff46908d20:     0x0    0x0
0x7eff46908d30:     0x7eff46908d40      0x40055b <procE+14>
0x7eff46908d40:     0x7eff46908d50      0x400575 <bar_one+24>
0x7eff46908d50:     0x7eff46908d60      0x400585 <foo_one+14>
0x7eff46908d60:     0x7eff46908d80      0x40059d <thread_one+22>
0x7eff46908d70:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7eff46908d80:     0x0    0x401690 <start_thread+208>
0x7eff46908d90:     0x0    0x7eff46909700
0x7eff46908da0:     0x0    0x0
0x7eff46908db0:     0x0    0x0
0x7eff46908dc0:     0x0    0x0
0x7eff46908dd0:     0x0    0x0
0x7eff46908de0:     0x0    0x0
0x7eff46908df0:     0x0    0x0
0x7eff46908e00:     0x0    0x0
0x7eff46908e10:     0x0    0x0
0x7eff46908e20:     0x7eff46909700      0xdf48debbb04adfea
0x7eff46908e30:     0x48c160 <default_attr>   0x7eff469099c0
```

```
0x7eff46908e40:     0x0    0x3
0x7eff46908e50:     0x22b6539aab6adfea  0xdf48de3b9ce6dfea
0x7eff46908e60:     0x0    0x0
0x7eff46908e70:     0x0    0x0
0x7eff46908e80:     0x0    0x0
0x7eff46908e90:     0x7eff46909700      0x432549 <clone+121>
0x7eff46908ea0:     0x0    0x0
0x7eff46908eb0:     0x0    0x0
0x7eff46908ec0:     0x0    0x0
0x7eff46908ed0:     0x0    0x0
0x7eff46908ee0:     0x0    0x0
0x7eff46908ef0:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7eff46908f00:     0x0    0x0
0x7eff46908f10:     0x0    0x0
0x7eff46908f20:     0x0    0x0
0x7eff46908f30:     0x0    0x0
0x7eff46908f40:     0x0    0x0
0x7eff46908f50:     0x0    0x0
0x7eff46908f60:     0x0    0x0
0x7eff46908f70:     0x0    0x0
0x7eff46908f80:     0x0    0x0
0x7eff46908f90:     0x0    0x0
0x7eff46908fa0:     0x0    0x0
0x7eff46908fb0:     0x0    0x0
0x7eff46908fc0:     0x0    0x0
0x7eff46908fd0:     0x0    0x0
0x7eff46908fe0:     0x0    0x0
0x7eff46908ff0:     0x0    0x0
0x7eff46909000:     0x0    0x0
0x7eff46909010:     0x0    0x0
0x7eff46909020:     0x0    0x0
0x7eff46909030:     0x0    0x0
0x7eff46909040:     0x0    0x0
0x7eff46909050:     0x0    0x0
0x7eff46909060:     0x0    0x0
0x7eff46909070:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7eff46909080:     0x0    0x0
0x7eff46909090:     0x0    0x0
0x7eff469090a0:     0x0    0x0
0x7eff469090b0:     0x0    0x0
0x7eff469090c0:     0x0    0x0
0x7eff469090d0:     0x0    0x0
0x7eff469090e0:     0x0    0x0
0x7eff469090f0:     0x0    0x0
0x7eff46909100:     0x0    0x0
0x7eff46909110:     0x0    0x0
0x7eff46909120:     0x0    0x0
0x7eff46909130:     0x0    0x0
0x7eff46909140:     0x0    0x0
0x7eff46909150:     0x0    0x0
0x7eff46909160:     0x0    0x0
0x7eff46909170:     0x0    0x0
0x7eff46909180:     0x0    0x0
0x7eff46909190:     0x0    0x0
0x7eff469091a0:     0x0    0x0
0x7eff469091b0:     0x0    0x0
0x7eff469091c0:     0x0    0x0
0x7eff469091d0:     0x0    0x0
```

```
0x7eff469091e0:     0x0    0x0
0x7eff469091f0:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7eff46909200:     0x0    0x0
0x7eff46909210:     0x0    0x0
0x7eff46909220:     0x0    0x0
0x7eff46909230:     0x0    0x0
0x7eff46909240:     0x0    0x0
0x7eff46909250:     0x0    0x0
0x7eff46909260:     0x0    0x0
0x7eff46909270:     0x0    0x0
0x7eff46909280:     0x0    0x0
0x7eff46909290:     0x0    0x0
0x7eff469092a0:     0x0    0x0
0x7eff469092b0:     0x0    0x0
0x7eff469092c0:     0x0    0x0
0x7eff469092d0:     0x0    0x0
0x7eff469092e0:     0x0    0x0
0x7eff469092f0:     0x0    0x0
0x7eff46909300:     0x0    0x0
0x7eff46909310:     0x0    0x0
0x7eff46909320:     0x0    0x0
0x7eff46909330:     0x0    0x0
0x7eff46909340:     0x0    0x0
0x7eff46909350:     0x0    0x0
0x7eff46909360:     0x0    0x0
0x7eff46909370:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7eff46909380:     0x0    0x0
0x7eff46909390:     0x0    0x0
0x7eff469093a0:     0x0    0x0
0x7eff469093b0:     0x0    0x0
0x7eff469093c0:     0x0    0x0
0x7eff469093d0:     0x0    0x0
0x7eff469093e0:     0x0    0x0
0x7eff469093f0:     0x0    0x0
0x7eff46909400:     0x0    0x0
0x7eff46909410:     0x0    0x0
0x7eff46909420:     0x0    0x0
0x7eff46909430:     0x0    0x0
0x7eff46909440:     0x0    0x0
0x7eff46909450:     0x0    0x0
0x7eff46909460:     0x0    0x0
0x7eff46909470:     0x0    0x0
0x7eff46909480:     0x0    0x0
0x7eff46909490:     0x0    0x0
0x7eff469094a0:     0x0    0x0
0x7eff469094b0:     0x0    0x0
0x7eff469094c0:     0x0    0x0
0x7eff469094d0:     0x0    0x0
0x7eff469094e0:     0x0    0x0
0x7eff469094f0:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7eff46909500:     0x0    0x0
0x7eff46909510:     0x0    0x0
0x7eff46909520:     0x0    0x0
0x7eff46909530:     0x0    0x0
0x7eff46909540:     0x0    0x0
0x7eff46909550:     0x0    0x0
0x7eff46909560:     0x0    0x0
```

```
0x7eff46909570:     0x0    0x0
0x7eff46909580:     0x0    0x0
0x7eff46909590:     0x0    0x0
0x7eff469095a0:     0x0    0x0
0x7eff469095b0:     0x0    0x0
0x7eff469095c0:     0x0    0x0
0x7eff469095d0:     0x0    0x0
0x7eff469095e0:     0x0    0x0
0x7eff469095f0:     0x0    0x0
0x7eff46909600:     0x0    0x0
0x7eff46909610:     0x0    0x0
0x7eff46909620:     0x0    0x0
0x7eff46909630:     0x0    0x0
0x7eff46909640:     0x0    0x0
0x7eff46909650:     0x0    0x0
0x7eff46909660:     0x0    0x0
0x7eff46909670:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7eff46909680:     0x0    0x0
0x7eff46909690:     0x0    0x0
0x7eff469096a0:     0x0    0x7eff46909db8
0x7eff469096b0:     0x6b7720 <_nl_global_locale>     0x6b7720 <_nl_global_locale>
0x7eff469096c0:     0x6b7740 <_nl_global_locale+32>  0x6b7728 <_nl_global_locale+8>
0x7eff469096d0:     0x0    0x0
0x7eff469096e0:     0x0    0x0
0x7eff469096f0:     0x0    0x0
0x7eff46909700:     0x7eff46909700      0x19a1680
0x7eff46909710:     0x7eff46909700      0x1
0x7eff46909720:     0x0    0xe63e8b268d639000
0x7eff46909730:     0x6ff56fa46f5dd825  0x0
0x7eff46909740:     0x0    0x0
0x7eff46909750:     0x0    0x0
0x7eff46909760:     0x0    0x0
0x7eff46909770:     0x0    0x0
0x7eff46909780:     0x0    0x0
0x7eff46909790:     0x0    0x0
0x7eff469097a0:     0x0    0x0
0x7eff469097b0:     0x0    0x0
0x7eff469097c0:     0x0    0x0
0x7eff469097d0:     0x0    0x0
0x7eff469097e0:     0x0    0x0
0x7eff469097f0:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7eff46909800:     0x0    0x0
0x7eff46909810:     0x0    0x0
0x7eff46909820:     0x0    0x0
0x7eff46909830:     0x0    0x0
0x7eff46909840:     0x0    0x0
0x7eff46909850:     0x0    0x0
0x7eff46909860:     0x0    0x0
0x7eff46909870:     0x0    0x0
0x7eff46909880:     0x0    0x0
0x7eff46909890:     0x0    0x0
0x7eff469098a0:     0x0    0x0
0x7eff469098b0:     0x0    0x0
0x7eff469098c0:     0x0    0x0
0x7eff469098d0:     0x0    0x0
0x7eff469098e0:     0x0    0x0
0x7eff469098f0:     0x0    0x0
0x7eff46909900:     0x0    0x0
```

```
0x7eff46909910:     0x0    0x0
0x7eff46909920:     0x0    0x0
0x7eff46909930:     0x0    0x0
0x7eff46909940:     0x0    0x0
0x7eff46909950:     0x0    0x0
0x7eff46909960:     0x0    0x0
0x7eff46909970:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7eff46909980:     0x0    0x0
0x7eff46909990:     0x0    0x0
0x7eff469099a0:     0x0    0x0
0x7eff469099b0:     0x0    0x0
0x7eff469099c0:     0x6b6be0 <stack_used>     0x7eff461089c0
0x7eff469099d0:     0x33c2000033c3      0x7eff469099e0
0x7eff469099e0:     0x7eff469099e0      0xfffffffffffffe0
0x7eff469099f0:     0x0    0x0
0x7eff46909a00:     0x7eff46908e20      0x0
0x7eff46909a10:     0x0    0x0
0x7eff46909a20:     0x0    0x0
0x7eff46909a30:     0x0    0x0
0x7eff46909a40:     0x0    0x0
0x7eff46909a50:     0x0    0x0
0x7eff46909a60:     0x0    0x0
0x7eff46909a70:     0x0    0x0
0x7eff46909a80:     0x0    0x0
0x7eff46909a90:     0x0    0x0
0x7eff46909aa0:     0x0    0x0
0x7eff46909ab0:     0x0    0x0
0x7eff46909ac0:     0x0    0x0
0x7eff46909ad0:     0x0    0x0
0x7eff46909ae0:     0x0    0x0
0x7eff46909af0:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7eff46909b00:     0x0    0x0
0x7eff46909b10:     0x0    0x0
0x7eff46909b20:     0x0    0x0
0x7eff46909b30:     0x0    0x0
0x7eff46909b40:     0x0    0x0
0x7eff46909b50:     0x0    0x0
0x7eff46909b60:     0x0    0x0
0x7eff46909b70:     0x0    0x0
0x7eff46909b80:     0x0    0x0
0x7eff46909b90:     0x0    0x0
0x7eff46909ba0:     0x0    0x0
0x7eff46909bb0:     0x0    0x0
0x7eff46909bc0:     0x0    0x0
0x7eff46909bd0:     0x0    0x0
0x7eff46909be0:     0x0    0x0
0x7eff46909bf0:     0x0    0x0
0x7eff46909c00:     0x0    0x0
0x7eff46909c10:     0x7eff46909a10      0x0
0x7eff46909c20:     0x0    0x0
0x7eff46909c30:     0x0    0x0
0x7eff46909c40:     0x0    0x0
0x7eff46909c50:     0x0    0x0
0x7eff46909c60:     0x0    0x0
0x7eff46909c70:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7eff46909c80:     0x0    0x0
0x7eff46909c90:     0x0    0x0
```

```
0x7eff46909ca0:     0x0    0x0
0x7eff46909cb0:     0x0    0x0
0x7eff46909cc0:     0x0    0x0
0x7eff46909cd0:     0x0    0x0
0x7eff46909ce0:     0x0    0x0
0x7eff46909cf0:     0x0    0x0
0x7eff46909d00:     0x0    0x0
0x7eff46909d10:     0x0    0x0
0x7eff46909d20:     0x35e98b19e80ac     0x0
0x7eff46909d30:     0x0    0x0
0x7eff46909d40:     0x400587 <thread_one>     0x0
0x7eff46909d50:     0x0    0x0
0x7eff46909d60:     0x0    0x0
0x7eff46909d70:     0x0    0x0
0x7eff46909d80:     0x0    0x0
0x7eff46909d90:     0x7eff46109000      0x801000
0x7eff46909da0:     0x1000 0x1000
0x7eff46909db0:     0x0    0x0
0x7eff46909dc0:     0x0    0x0
0x7eff46909dd0:     0x0    0x0
0x7eff46909de0:     0x0    0x0
0x7eff46909df0:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7eff46909e00:     0x0    0x0
0x7eff46909e10:     0x0    0x0
0x7eff46909e20:     0x0    0x0
0x7eff46909e30:     0x0    0x0
0x7eff46909e40:     0x0    0x0
0x7eff46909e50:     0x0    0x0
0x7eff46909e60:     0x0    0x0
0x7eff46909e70:     0x0    0x0
0x7eff46909e80:     0x0    0x0
0x7eff46909e90:     0x0    0x0
0x7eff46909ea0:     0x0    0x0
0x7eff46909eb0:     0x0    0x0
0x7eff46909ec0:     0x0    0x0
0x7eff46909ed0:     0x0    0x0
0x7eff46909ee0:     0x0    0x0
0x7eff46909ef0:     0x0    0x0
0x7eff46909f00:     0x0    0x0
0x7eff46909f10:     0x0    0x0
0x7eff46909f20:     0x0    0x0
0x7eff46909f30:     0x0    0x0
0x7eff46909f40:     0x0    0x0
0x7eff46909f50:     0x0    0x0
0x7eff46909f60:     0x0    0x0
0x7eff46909f70:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7eff46909f80:     0x0    0x0
0x7eff46909f90:     0x0    0x0
0x7eff46909fa0:     0x0    0x0
0x7eff46909fb0:     0x0    0x0
0x7eff46909fc0:     0x0    0x0
0x7eff46909fd0:     0x0    0x0
0x7eff46909fe0:     0x0    0x0
0x7eff46909ff0:     0x0    0x0
```

7. See that the reconstruction of the stack trace is possible because of standard function prologue and epilogue:

```
[...]
0x7eff46908070:     0x7eff46908290      0x40054b <procF+91>
0x7eff46908290:     0x7eff469084b0      0x40054b <procF+91>
0x7eff469084b0:     0x7eff469086d0      0x40054b <procF+91>
0x7eff469086d0:     0x7eff469088f0      0x40054b <procF+91>
0x7eff469088f0:     0x7eff46908b10      0x40054b <procF+91>
0x7eff46908b10:     0x7eff46908d30      0x40054b <procF+91>
0x7eff46908d30:     0x7eff46908d40      0x40055b <procE+14>
0x7eff46908d40:     0x7eff46908d50      0x400575 <bar_one+24>
0x7eff46908d50:     0x7eff46908d60      0x400585 <foo_one+14>
0x7eff46908d60:     0x7eff46908d80      0x40059d <thread_one+22>
0x7eff46908d80:     0x0    0x401690 <start_thread+208>

(gdb) disass procF
Dump of assembler code for function procF:
   0x00000000004004f0 <+0>:     push   %rbp
   0x00000000004004f1 <+1>:     mov    %rsp,%rbp
   0x00000000004004f4 <+4>:     sub    $0x210,%rsp
   0x00000000004004fb <+11>:    mov    %edi,-0x204(%rbp)
   0x0000000000400501 <+17>:    lea    -0x200(%rbp),%rsi
   0x0000000000400508 <+24>:    mov    $0x0,%eax
   0x000000000040050d <+29>:    mov    $0x40,%edx
   0x0000000000400512 <+34>:    mov    %rsi,%rdi
   0x0000000000400515 <+37>:    mov    %rdx,%rcx
   0x0000000000400518 <+40>:    rep stos %rax,%es:(%rdi)
   0x000000000040051b <+43>:    movl   $0xffffffff,-0x200(%rbp)
   0x0000000000400525 <+53>:    mov    -0x204(%rbp),%eax
   0x000000000040052b <+59>:    add    $0x1,%eax
   0x000000000040052e <+62>:    mov    %eax,-0x1f8(%rbp)
   0x0000000000400534 <+68>:    movl   $0xffffffff,-0x1f0(%rbp)
   0x000000000040053e <+78>:    mov    -0x1f8(%rbp),%eax
   0x0000000000400544 <+84>:    mov    %eax,%edi
   0x0000000000400546 <+86>:    callq  0x4004f0 <procF>
=> 0x000000000040054b <+91>:    leaveq
   0x000000000040054c <+92>:    retq
End of assembler dump.
```

8. Use back trace command variant to get to the bottom of the stack trace:

```
(gdb) bt -20
#15399 0x000000000040054b in procF ()
#15400 0x000000000040054b in procF ()
#15401 0x000000000040054b in procF ()
#15402 0x000000000040054b in procF ()
#15403 0x000000000040054b in procF ()
#15404 0x000000000040054b in procF ()
#15405 0x000000000040054b in procF ()
#15406 0x000000000040054b in procF ()
#15407 0x000000000040054b in procF ()
#15408 0x000000000040054b in procF ()
#15409 0x000000000040054b in procF ()
#15410 0x000000000040054b in procF ()
#15411 0x000000000040054b in procF ()
#15412 0x000000000040055b in procE ()
#15413 0x0000000000400575 in bar_one ()
#15414 0x0000000000400585 in foo_one ()
#15415 0x000000000040059d in thread_one ()
```

```
#15416 0x0000000000401690 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#15417 0x0000000000432549 in clone ()
#15418 0x0000000000000000 in ?? ()
```

Exercise A7

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

© 2015 Software Diagnostics Services

Exercise A7

Goal: Learn how to identify active threads.

Patterns: Divide by Zero, Active Thread.

1. Load a core dump and *App7* executable:

```
training@debian64:~/ALCDA$ gdb -c ./App7/core -se ./App7/App7
GNU gdb (GDB) 7.4.1-debian
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/training/ALCDA/App7/App7...done.
[New LWP 14843]
[New LWP 14844]
[New LWP 14842]
[New LWP 14841]
[New LWP 14840]
[New LWP 14845]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Core was generated by `./App7'.
Program terminated with signal 8, Arithmetic exception.
#0  0x000000000040056f in procD ()
```

2. List and identify the possible problem threads:

```
(gdb) info threads
  Id   Target Id         Frame
  6    Thread 0x7f0f6706a700 (LWP 14845) 0x00000000004004fb in procF ()
  5    Thread 0xe3f860 (LWP 14840) 0x000000000042ff91 in nanosleep ()
  4    Thread 0x7f0f6906e700 (LWP 14841) 0x000000000042ff91 in nanosleep ()
  3    Thread 0x7f0f6886d700 (LWP 14842) 0x000000000042ff91 in nanosleep ()
  2    Thread 0x7f0f6786b700 (LWP 14844) 0x000000000042ff91 in nanosleep ()
* 1    Thread 0x7f0f6806c700 (LWP 14843) 0x000000000040056f in procD ()
```

3. List stack trace for the current problem thread #1 and identify the problem instruction:

```
(gdb) bt
Thread 1 (Thread 0x7f0f6806c700 (LWP 14843)):
#0  0x000000000040056f in procD ()
#1  0x0000000000400587 in procC ()
#2  0x000000000040070d in bar_three ()
#3  0x000000000040071d in foo_three ()
#4  0x0000000000400735 in thread_three ()
#5  0x00000000004017a0 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#6  0x0000000000432649 in clone ()
#7  0x0000000000000000 in ?? ()
```

```
(gdb) x/i $rip
=> 0x40056f <procD+18>:   idivl  -0x8(%rbp)

(gdb) info r $rax
rax            0x1  1

(gdb) x/w $rbp-0x8
0x7f0f6806bd28:     0x00000000
```

4. Check the currently executing instruction of identified non-waiting thread #6 and compare the stack pointer with the stack region boundaries since we suspect stack overflow:

```
(gdb) thread 6
[Switching to thread 6 (Thread 0x7f0f6706a700 (LWP 14845))]
#0  0x00000000004004fb in procF ()

(gdb) bt
#0  0x00000000004004fb in procF ()
#1  0x000000000040054b in procF ()
#2  0x000000000040054b in procF ()
#3  0x000000000040054b in procF ()
#4  0x000000000040054b in procF ()
#5  0x000000000040054b in procF ()
#6  0x000000000040054b in procF ()
#7  0x000000000040054b in procF ()
#8  0x000000000040054b in procF ()
#9  0x000000000040054b in procF ()
#10 0x000000000040054b in procF ()
#11 0x000000000040054b in procF ()
#12 0x000000000040054b in procF ()
#13 0x000000000040054b in procF ()
#14 0x000000000040054b in procF ()
#15 0x000000000040054b in procF ()
#16 0x000000000040054b in procF ()
#17 0x000000000040054b in procF ()
#18 0x000000000040054b in procF ()
#19 0x000000000040054b in procF ()
#20 0x000000000040054b in procF ()
#21 0x000000000040054b in procF ()
#22 0x000000000040054b in procF ()
#23 0x000000000040054b in procF ()
---Type <return> to continue, or q <return> to quit---
#24 0x000000000040054b in procF ()
#25 0x000000000040054b in procF ()
#26 0x000000000040054b in procF ()
#27 0x000000000040054b in procF ()
#28 0x000000000040054b in procF ()
#29 0x000000000040054b in procF ()
#30 0x000000000040054b in procF ()
#31 0x000000000040054b in procF ()
#32 0x000000000040054b in procF ()
#33 0x000000000040054b in procF ()
#34 0x000000000040054b in procF ()
#35 0x000000000040054b in procF ()
#36 0x000000000040054b in procF ()
#37 0x000000000040054b in procF ()
#38 0x000000000040054b in procF ()
#39 0x000000000040054b in procF ()
#40 0x000000000040054b in procF ()
#41 0x000000000040054b in procF ()
```

```
#42 0x000000000040054b in procF ()
#43 0x000000000040054b in procF ()
#44 0x000000000040054b in procF ()
#45 0x000000000040054b in procF ()
#46 0x000000000040054b in procF ()
#47 0x000000000040054b in procF ()
---Type <return> to continue, or q <return> to quit---
#48 0x000000000040054b in procF ()
#49 0x000000000040054b in procF ()
#50 0x000000000040054b in procF ()
#51 0x000000000040054b in procF ()
#52 0x000000000040054b in procF ()
#53 0x000000000040054b in procF ()
#54 0x000000000040054b in procF ()
#55 0x000000000040054b in procF ()
#56 0x000000000040054b in procF ()
#57 0x000000000040054b in procF ()
#58 0x000000000040054b in procF ()
#59 0x000000000040054b in procF ()
#60 0x000000000040054b in procF ()
#61 0x000000000040054b in procF ()
#62 0x000000000040054b in procF ()
#63 0x000000000040054b in procF ()
#64 0x000000000040054b in procF ()
#65 0x000000000040054b in procF ()
#66 0x000000000040054b in procF ()
#67 0x000000000040054b in procF ()
#68 0x000000000040054b in procF ()
#69 0x000000000040054b in procF ()
#70 0x000000000040054b in procF ()
#71 0x000000000040054b in procF ()
---Type <return> to continue, or q <return> to quit---
#72 0x000000000040054b in procF ()
#73 0x000000000040054b in procF ()
#74 0x000000000040054b in procF ()
#75 0x000000000040054b in procF ()
#76 0x000000000040054b in procF ()
#77 0x000000000040054b in procF ()
#78 0x000000000040054b in procF ()
#79 0x000000000040054b in procF ()
#80 0x000000000040054b in procF ()
#81 0x000000000040054b in procF ()
#82 0x000000000040054b in procF ()
#83 0x000000000040054b in procF ()
#84 0x000000000040054b in procF ()
#85 0x000000000040054b in procF ()
#86 0x000000000040054b in procF ()
#87 0x000000000040054b in procF ()
#88 0x000000000040054b in procF ()
#89 0x000000000040054b in procF ()
#90 0x000000000040054b in procF ()
#91 0x000000000040054b in procF ()
#92 0x000000000040054b in procF ()
#93 0x000000000040054b in procF ()
#94 0x000000000040054b in procF ()
#95 0x000000000040054b in procF ()
---Type <return> to continue, or q <return> to quit---
#96 0x000000000040054b in procF ()
#97 0x000000000040054b in procF ()
#98 0x000000000040054b in procF ()
```

```
#99 0x000000000040054b in procF ()
#100 0x000000000040054b in procF ()
#101 0x000000000040054b in procF ()
#102 0x000000000040054b in procF ()
#103 0x000000000040054b in procF ()
#104 0x000000000040054b in procF ()
#105 0x000000000040054b in procF ()
#106 0x000000000040054b in procF ()
#107 0x000000000040054b in procF ()
#108 0x000000000040054b in procF ()
#109 0x000000000040054b in procF ()
#110 0x000000000040054b in procF ()
#111 0x000000000040054b in procF ()
#112 0x000000000040054b in procF ()
#113 0x000000000040054b in procF ()
#114 0x000000000040054b in procF ()
#115 0x000000000040054b in procF ()
#116 0x000000000040054b in procF ()
#117 0x000000000040054b in procF ()
#118 0x000000000040054b in procF ()
#119 0x000000000040054b in procF ()
---Type <return> to continue, or q <return> to quit---
#120 0x000000000040054b in procF ()
#121 0x000000000040054b in procF ()
#122 0x000000000040054b in procF ()
#123 0x000000000040054b in procF ()
#124 0x000000000040054b in procF ()
#125 0x000000000040054b in procF ()
#126 0x000000000040054b in procF ()
#127 0x000000000040054b in procF ()
#128 0x000000000040054b in procF ()
#129 0x000000000040054b in procF ()
#130 0x000000000040054b in procF ()
#131 0x000000000040054b in procF ()
#132 0x000000000040054b in procF ()
#133 0x000000000040054b in procF ()
#134 0x000000000040054b in procF ()
#135 0x000000000040054b in procF ()
#136 0x000000000040054b in procF ()
#137 0x000000000040054b in procF ()
#138 0x000000000040054b in procF ()
#139 0x000000000040054b in procF ()
#140 0x000000000040054b in procF ()
#141 0x000000000040054b in procF ()
#142 0x000000000040054b in procF ()
#143 0x000000000040054b in procF ()
---Type <return> to continue, or q <return> to quit---
#144 0x000000000040054b in procF ()
#145 0x000000000040054b in procF ()
#146 0x000000000040054b in procF ()
#147 0x000000000040054b in procF ()
#148 0x000000000040054b in procF ()
#149 0x000000000040054b in procF ()
#150 0x000000000040054b in procF ()
#151 0x000000000040054b in procF ()
#152 0x000000000040054b in procF ()
#153 0x000000000040054b in procF ()
#154 0x000000000040054b in procF ()
#155 0x000000000040054b in procF ()
#156 0x000000000040054b in procF ()
```

```
#157 0x000000000040054b in procF ()
#158 0x000000000040054b in procF ()
#159 0x000000000040054b in procF ()
#160 0x000000000040054b in procF ()
#161 0x000000000040054b in procF ()
#162 0x000000000040054b in procF ()
#163 0x000000000040054b in procF ()
#164 0x000000000040054b in procF ()
#165 0x000000000040054b in procF ()
#166 0x000000000040054b in procF ()
#167 0x000000000040054b in procF ()
---Type <return> to continue, or q <return> to quit---
#168 0x000000000040054b in procF ()
#169 0x000000000040054b in procF ()
#170 0x000000000040054b in procF ()
#171 0x000000000040054b in procF ()
#172 0x000000000040054b in procF ()
#173 0x000000000040054b in procF ()
#174 0x000000000040054b in procF ()
#175 0x000000000040054b in procF ()
#176 0x000000000040054b in procF ()
#177 0x000000000040054b in procF ()
#178 0x000000000040054b in procF ()
#179 0x000000000040054b in procF ()
#180 0x000000000040054b in procF ()
#181 0x000000000040054b in procF ()
#182 0x000000000040054b in procF ()
#183 0x000000000040054b in procF ()
#184 0x000000000040054b in procF ()
#185 0x000000000040054b in procF ()
#186 0x000000000040054b in procF ()
#187 0x000000000040054b in procF ()
#188 0x000000000040054b in procF ()
#189 0x000000000040054b in procF ()
#190 0x000000000040054b in procF ()
#191 0x000000000040054b in procF ()
---Type <return> to continue, or q <return> to quit---
#192 0x000000000040054b in procF ()
#193 0x000000000040054b in procF ()
#194 0x000000000040054b in procF ()
#195 0x000000000040054b in procF ()
#196 0x000000000040054b in procF ()
#197 0x000000000040054b in procF ()
#198 0x000000000040054b in procF ()
#199 0x000000000040054b in procF ()
#200 0x000000000040054b in procF ()
#201 0x000000000040054b in procF ()
#202 0x000000000040054b in procF ()
#203 0x000000000040054b in procF ()
#204 0x000000000040054b in procF ()
#205 0x000000000040054b in procF ()
#206 0x000000000040054b in procF ()
#207 0x000000000040054b in procF ()
#208 0x000000000040054b in procF ()
#209 0x000000000040054b in procF ()
#210 0x000000000040054b in procF ()
#211 0x000000000040054b in procF ()
#212 0x000000000040054b in procF ()
#213 0x000000000040054b in procF ()
#214 0x000000000040054b in procF ()
```

```
#215 0x000000000040054b in procF ()
---Type <return> to continue, or q <return> to quit---
#216 0x000000000040054b in procF ()
#217 0x000000000040054b in procF ()
#218 0x000000000040054b in procF ()
#219 0x000000000040054b in procF ()
#220 0x000000000040054b in procF ()
#221 0x000000000040054b in procF ()
#222 0x000000000040054b in procF ()
#223 0x000000000040054b in procF ()
#224 0x000000000040054b in procF ()
#225 0x000000000040054b in procF ()
#226 0x000000000040054b in procF ()
#227 0x000000000040055b in procE ()
#228 0x000000000040079b in bar_five ()
#229 0x00000000004007ab in foo_five ()
#230 0x00000000004007c3 in thread_five ()
#231 0x00000000004017a0 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#232 0x0000000000432649 in clone ()
#233 0x0000000000000000 in ?? ()

(gdb) x/i $rip
=> 0x4004fb <procF+11>:    mov    %edi,-0x1004(%rbp)

(gdb) disassemble $rip
Dump of assembler code for function procF:
   0x00000000004004f0 <+0>:      push   %rbp
   0x00000000004004f1 <+1>:      mov    %rsp,%rbp
   0x00000000004004f4 <+4>:      sub    $0x1010,%rsp
=> 0x00000000004004fb <+11>:     mov    %edi,-0x1004(%rbp)
   0x0000000000400501 <+17>:     lea    -0x1000(%rbp),%rsi
   0x0000000000400508 <+24>:     mov    $0x0,%eax
   0x000000000040050d <+29>:     mov    $0x200,%edx
   0x0000000000400512 <+34>:     mov    %rsi,%rdi
   0x0000000000400515 <+37>:     mov    %rdx,%rcx
   0x0000000000400518 <+40>:     rep stos %rax,%es:(%rdi)
   0x000000000040051b <+43>:     movl   $0xffffffff,-0x1000(%rbp)
   0x0000000000400525 <+53>:     mov    -0x1004(%rbp),%eax
   0x000000000040052b <+59>:     add    $0x1,%eax
   0x000000000040052e <+62>:     mov    %eax,-0xff8(%rbp)
   0x0000000000400534 <+68>:     movl   $0xffffffff,-0xff0(%rbp)
   0x000000000040053e <+78>:     mov    -0xff8(%rbp),%eax
   0x0000000000400544 <+84>:     mov    %eax,%edi
   0x0000000000400546 <+86>:     callq  0x4004f0 <procF>
   0x000000000040054b <+91>:     leaveq
   0x000000000040054c <+92>:     retq
End of assembler dump.

(gdb) info r rsp
rsp            0x7f0f66f850e0    0x7f0f66f850e0

(gdb) maintenance info sections
Exec file:
    `/home/training/ALCDA/App7/App7', file type elf64-x86-64.
    0x00400158->0x00400178 at 0x00000158: .note.ABI-tag ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x00400178->0x0040019c at 0x00000178: .note.gnu.build-id ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004001a0->0x004002d8 at 0x000001a0: .rela.plt ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004002d8->0x004002e6 at 0x000002d8: .init ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x004002f0->0x004003c0 at 0x000002f0: .plt ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x004003c0->0x0048b358 at 0x000003c0: .text ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x0048b360->0x0048bede at 0x0008b360: __libc_freeres_fn ALLOC LOAD READONLY CODE HAS_CONTENTS
```

```
    0x0048bee0->0x0048bf41 at 0x0008bee0: __libc_thread_freeres_fn ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x0048bf44->0x0048bf4d at 0x0008bf44: .fini ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x0048bf60->0x004a9ec4 at 0x0008bf60: .rodata ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004a9ec8->0x004a9f28 at 0x000a9ec8: __libc_subfreeres ALLOC LOAD READONLY DATA HAS_CONTENTS
---Type <return> to continue, or q <return> to quit---
    0x004a9f28->0x004a9f30 at 0x000a9f28: __libc_atexit ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004a9f30->0x004a9f38 at 0x000a9f30: __libc_thread_subfreeres ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004a9f38->0x004b6acc at 0x000a9f38: .eh_frame ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004b6acc->0x004b6be6 at 0x000b6acc: .gcc_except_table ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x006b6be8->0x006b6c10 at 0x000b6be8: .tdata ALLOC LOAD DATA HAS_CONTENTS
    0x006b6c10->0x006b6c40 at 0x000b6c10: .tbss ALLOC
    0x006b6c10->0x006b6c20 at 0x000b6c10: .init_array ALLOC LOAD DATA HAS_CONTENTS
    0x006b6c20->0x006b6c30 at 0x000b6c20: .fini_array ALLOC LOAD DATA HAS_CONTENTS
    0x006b6c30->0x006b6c38 at 0x000b6c30: .jcr ALLOC LOAD DATA HAS_CONTENTS
    0x006b6c40->0x006b6cb0 at 0x000b6c40: .data.rel.ro ALLOC LOAD DATA HAS_CONTENTS
    0x006b6cb0->0x006b6cc0 at 0x000b6cb0: .got ALLOC LOAD DATA HAS_CONTENTS
    0x006b6cc0->0x006b6d40 at 0x000b6cc0: .got.plt ALLOC LOAD DATA HAS_CONTENTS
    0x006b6d40->0x006b7a50 at 0x000b6d40: .data ALLOC LOAD DATA HAS_CONTENTS
    0x006b7a60->0x006bedc8 at 0x000b7a50: .bss ALLOC
    0x006bedc8->0x006bedf8 at 0x000b7a50: __libc_freeres_ptrs ALLOC
    0x00000000->0x00000038 at 0x000b7a50: .comment READONLY HAS_CONTENTS
    0x00000000->0x00000390 at 0x000b7a90: .debug_aranges READONLY HAS_CONTENTS
---Type <return> to continue, or q <return> to quit---
    0x00000000->0x00000ac3 at 0x000b7e20: .debug_pubnames READONLY HAS_CONTENTS
    0x00000000->0x00011440 at 0x000b88e3: .debug_info READONLY HAS_CONTENTS
    0x00000000->0x000021b1 at 0x000c9d23: .debug_abbrev READONLY HAS_CONTENTS
    0x00000000->0x00002ebc at 0x000cbed4: .debug_line READONLY HAS_CONTENTS
    0x00000000->0x000038da at 0x000ced90: .debug_str READONLY HAS_CONTENTS
    0x00000000->0x0000878e at 0x000d266a: .debug_loc READONLY HAS_CONTENTS
    0x00000000->0x00001280 at 0x000dadf8: .debug_ranges READONLY HAS_CONTENTS
Core file:
    `/home/training/ALCDA/./App7/core', file type elf64-x86-64.
    0x00000000->0x00002aa8 at 0x00000430: note0 READONLY HAS_CONTENTS
    0x00000000->0x000000d8 at 0x000004b4: .reg/14843 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x000004b4: .reg HAS_CONTENTS
    0x00000000->0x00000130 at 0x00000644: .auxv HAS_CONTENTS
    0x00000000->0x00000200 at 0x00000788: .reg2/14843 HAS_CONTENTS
    0x00000000->0x00000200 at 0x00000788: .reg2 HAS_CONTENTS
    0x00000000->0x00000340 at 0x0000099c: .reg-xstate/14843 HAS_CONTENTS
    0x00000000->0x00000340 at 0x0000099c: .reg-xstate HAS_CONTENTS
    0x00000000->0x000000d8 at 0x00000d60: .reg/14844 HAS_CONTENTS
    0x00000000->0x00000200 at 0x00000e54: .reg2/14844 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00001068: .reg-xstate/14844 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x0000142c: .reg/14842 HAS_CONTENTS
    0x00000000->0x00000200 at 0x00001520: .reg2/14842 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00001734: .reg-xstate/14842 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x00001af8: .reg/14841 HAS_CONTENTS
---Type <return> to continue, or q <return> to quit---
    0x00000000->0x00000200 at 0x00001bec: .reg2/14841 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00001e00: .reg-xstate/14841 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x000021c4: .reg/14840 HAS_CONTENTS
    0x00000000->0x00000200 at 0x000022b8: .reg2/14840 HAS_CONTENTS
    0x00000000->0x00000340 at 0x000024cc: .reg-xstate/14840 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x00002890: .reg/14845 HAS_CONTENTS
    0x00000000->0x00000200 at 0x00002984: .reg2/14845 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00002b98: .reg-xstate/14845 HAS_CONTENTS
    0x00400000->0x00401000 at 0x00003000: load1a ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x00401000->0x00401000 at 0x00004000: load1b ALLOC READONLY CODE
    0x006b6000->0x006b8000 at 0x00004000: load2 ALLOC LOAD HAS_CONTENTS
    0x006b8000->0x006bf000 at 0x00006000: load3 ALLOC LOAD HAS_CONTENTS
    0x00e3f000->0x00e62000 at 0x0000d000: load4 ALLOC LOAD HAS_CONTENTS
    0x7f0f6686a000->0x7f0f6686b000 at 0x00030000: load5 ALLOC LOAD READONLY HAS_CONTENTS
    0x7f0f6686b000->0x7f0f6706b000 at 0x00031000: load6 ALLOC LOAD HAS_CONTENTS
    0x7f0f6706b000->0x7f0f6706c000 at 0x00831000: load7 ALLOC LOAD READONLY HAS_CONTENTS
    0x7f0f6706c000->0x7f0f6786c000 at 0x00832000: load8 ALLOC LOAD HAS_CONTENTS
    0x7f0f6786c000->0x7f0f6786d000 at 0x01032000: load9 ALLOC LOAD READONLY HAS_CONTENTS
    0x7f0f6786d000->0x7f0f6806d000 at 0x01033000: load10 ALLOC LOAD HAS_CONTENTS
```

```
---Type <return> to continue, or q <return> to quit---
    0x7f0f6806d000->0x7f0f6806e000 at 0x01833000: load11 ALLOC LOAD READONLY HAS_CONTENTS
    0x7f0f6806e000->0x7f0f6886e000 at 0x01834000: load12 ALLOC LOAD HAS_CONTENTS
    0x7f0f6886e000->0x7f0f6886f000 at 0x02034000: load13 ALLOC LOAD READONLY HAS_CONTENTS
    0x7f0f6886f000->0x7f0f6906f000 at 0x02035000: load14 ALLOC LOAD HAS_CONTENTS
    0x7ffff76e9000->0x7ffff770b000 at 0x02835000: load15 ALLOC LOAD HAS_CONTENTS
    0x7ffff77c5000->0x7ffff77c6000 at 0x02857000: load16 ALLOC LOAD READONLY CODE HAS_CONTENTS
    0xffffffffff600000->0xffffffffff600000 at 0x02858000: load17 ALLOC READONLY CODE
```

We see that the stack pointer value 0x**7f0f66**f850e0 is inside the stack region address range 0x**7f0f66**86b000 - 0x**7f0f67**06b000.

Exercise A8

- **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
- \ALCDA-Dumps\Exercise-A8.pdf

© 2015 Software Diagnostics Services

Exercise A8

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.

1. Load a core dump and *App8* executable:

```
training@debian64:~/ALCDA$ gdb -c ./App8/core -se ./App8/App8
GNU gdb (GDB) 7.4.1-debian
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/training/ALCDA/App8/App8...done.
[New LWP 15203]
[New LWP 15204]
[New LWP 15205]
[New LWP 15206]
[New LWP 15207]
[New LWP 15202]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Core was generated by `./App8'.
Program terminated with signal 6, Aborted.
#0  0x00000000004524b5 in raise ()
```

2. List all thread stack traces:

```
(gdb) thread apply all bt

Thread 6 (Thread 0x1ae3880 (LWP 15202)):
#0  0x00000000004431f1 in nanosleep ()
#1  0x00000000004430c0 in sleep ()
#2  0x00000000004008f9 in main ()

Thread 5 (Thread 0x7f4cab440700 (LWP 15207)):
#0  0x00000000004431f1 in nanosleep ()
#1  0x00000000004430c0 in sleep ()
#2  0x0000000000400771 in procNE() ()
#3  0x0000000000400834 in bar_five() ()
#4  0x000000000040083f in foo_five() ()
#5  0x0000000000400852 in thread_five(void*) ()
#6  0x00000000004140f0 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#7  0x0000000000445879 in clone ()
#8  0x0000000000000000 in ?? ()
```

```
Thread 4 (Thread 0x7f4cabc41700 (LWP 15206)):
#0  0x00000000004431f1 in nanosleep ()
#1  0x00000000004430c0 in sleep ()
#2  0x0000000000400771 in procNE() ()
#3  0x0000000000400806 in bar_four() ()
#4  0x0000000000400811 in foo_four() ()
---Type <return> to continue, or q <return> to quit---
#5  0x0000000000400824 in thread_four(void*) ()
#6  0x00000000004140f0 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#7  0x0000000000445879 in clone ()
#8  0x0000000000000000 in ?? ()

Thread 3 (Thread 0x7f4cac442700 (LWP 15205)):
#0  0x00000000004431f1 in nanosleep ()
#1  0x00000000004430c0 in sleep ()
#2  0x000000000040073c in procH() ()
#3  0x00000000004007d8 in bar_three() ()
#4  0x00000000004007e3 in foo_three() ()
#5  0x00000000004007f6 in thread_three(void*) ()
#6  0x00000000004140f0 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#7  0x0000000000445879 in clone ()
#8  0x0000000000000000 in ?? ()

Thread 2 (Thread 0x7f4cacc43700 (LWP 15204)):
#0  0x00000000004431f1 in nanosleep ()
#1  0x00000000004430c0 in sleep ()
#2  0x0000000000400771 in procNE() ()
#3  0x00000000004007aa in bar_two() ()
#4  0x00000000004007b5 in foo_two() ()
---Type <return> to continue, or q <return> to quit---
#5  0x00000000004007c8 in thread_two(void*) ()
#6  0x00000000004140f0 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#7  0x0000000000445879 in clone ()
#8  0x0000000000000000 in ?? ()

Thread 1 (Thread 0x7f4cad444700 (LWP 15203)):
#0  0x00000000004524b5 in raise ()
#1  0x000000000041ca60 in abort ()
#2  0x000000000040562d in __gnu_cxx::__verbose_terminate_handler() ()
#3  0x0000000000405146 in __cxxabiv1::__terminate(void (*)()) ()
#4  0x0000000000405173 in std::terminate() ()
#5  0x000000000040165e in __cxa_throw ()
#6  0x00000000004006a3 in procB() ()
#7  0x00000000004006fa in procA() ()
#8  0x000000000040075c in procNH() ()
#9  0x000000000040077c in bar_one() ()
#10 0x0000000000400787 in foo_one() ()
#11 0x000000000040079a in thread_one(void*) ()
#12 0x00000000004140f0 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#13 0x0000000000445879 in clone ()
#14 0x0000000000000000 in ?? ()
```

We have C++ exception processing in thread #1.

3. Go to the thread #2, identify execution residue of *work* functions and check their correctness:

```
(gdb) thread 2
[Switching to thread 2 (Thread 0x7f4cacc43700 (LWP 15204))]
#0  0x00000000004431f1 in nanosleep ()

(gdb) bt
#0  0x00000000004431f1 in nanosleep ()
#1  0x00000000004430c0 in sleep ()
#2  0x0000000000400771 in procNE() ()
#3  0x00000000004007aa in bar_two() ()
#4  0x00000000004007b5 in foo_two() ()
#5  0x00000000004007c8 in thread_two(void*) ()
#6  0x00000000004140f0 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#7  0x0000000000445879 in clone ()
#8  0x0000000000000000 in ?? ()

(gdb) x/512a $rsp-2000
0x7f4cacc42360:     0x0    0x0
0x7f4cacc42370:     0x0    0x0
0x7f4cacc42380:     0x0    0x0
0x7f4cacc42390:     0x0    0x0
0x7f4cacc423a0:     0x0    0x0
0x7f4cacc423b0:     0x0    0x0
0x7f4cacc423c0:     0x0    0x0
0x7f4cacc423d0:     0x0    0x0
0x7f4cacc423e0:     0x0    0x0
0x7f4cacc423f0:     0x0    0x0
0x7f4cacc42400:     0x0    0x0
0x7f4cacc42410:     0x0    0x0
0x7f4cacc42420:     0x0    0x0
0x7f4cacc42430:     0x0    0x0
0x7f4cacc42440:     0x0    0x0
0x7f4cacc42450:     0x0    0x0
0x7f4cacc42460:     0x0    0x0
0x7f4cacc42470:     0x0    0x0
0x7f4cacc42480:     0x0    0x0
0x7f4cacc42490:     0x0    0x0
0x7f4cacc424a0:     0x0    0x0
0x7f4cacc424b0:     0x0    0x0
0x7f4cacc424c0:     0x0    0x0
0x7f4cacc424d0:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7f4cacc424e0:     0x0    0x0
0x7f4cacc424f0:     0x0    0x0
0x7f4cacc42500:     0x0    0x0
0x7f4cacc42510:     0x0    0x0
0x7f4cacc42520:     0x0    0x0
0x7f4cacc42530:     0x0    0x0
0x7f4cacc42540:     0x0    0x0
0x7f4cacc42550:     0x0    0x0
0x7f4cacc42560:     0x0    0x0
0x7f4cacc42570:     0x0    0x0
0x7f4cacc42580:     0x0    0x0
0x7f4cacc42590:     0x0    0x0
0x7f4cacc425a0:     0x0    0x0
0x7f4cacc425b0:     0x0    0x0
0x7f4cacc425c0:     0x0    0x0
0x7f4cacc425d0:     0x0    0x0
```

```
0x7f4cacc425e0:     0x0    0x0
0x7f4cacc425f0:     0x0    0x0
0x7f4cacc42600:     0x0    0x0
0x7f4cacc42610:     0x0    0x0
0x7f4cacc42620:     0x0    0x0
0x7f4cacc42630:     0x0    0x0
0x7f4cacc42640:     0x0    0x0
0x7f4cacc42650:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7f4cacc42660:     0x0    0x0
0x7f4cacc42670:     0x0    0x0
0x7f4cacc42680:     0x0    0x0
0x7f4cacc42690:     0x0    0x0
0x7f4cacc426a0:     0x0    0x0
0x7f4cacc426b0:     0x0    0x0
0x7f4cacc426c0:     0x0    0x0
0x7f4cacc426d0:     0x0    0x0
0x7f4cacc426e0:     0x0    0x0
0x7f4cacc426f0:     0x0    0x0
0x7f4cacc42700:     0x0    0x0
0x7f4cacc42710:     0x0    0x0
0x7f4cacc42720:     0x0    0x0
0x7f4cacc42730:     0x0    0x0
0x7f4cacc42740:     0x0    0x0
0x7f4cacc42750:     0x0    0x0
0x7f4cacc42760:     0x0    0x0
0x7f4cacc42770:     0x0    0x0
0x7f4cacc42780:     0x0    0x0
0x7f4cacc42790:     0x0    0x0
0x7f4cacc427a0:     0x0    0x0
0x7f4cacc427b0:     0x0    0x0
0x7f4cacc427c0:     0x0    0x0
0x7f4cacc427d0:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7f4cacc427e0:     0x0    0x0
0x7f4cacc427f0:     0x0    0x0
0x7f4cacc42800:     0x0    0x0
0x7f4cacc42810:     0x0    0x0
0x7f4cacc42820:     0x0    0x0
0x7f4cacc42830:     0x0    0x0
0x7f4cacc42840:     0x0    0x0
0x7f4cacc42850:     0x0    0x0
0x7f4cacc42860:     0x7f4cacc42870      0x4005af <_Z6work_8v+9>
0x7f4cacc42870:     0x7f4cacc42880      0x4005ba <_Z6work_7v+9>
0x7f4cacc42880:     0x7f4cacc42890      0x4005c5 <_Z6work_6v+9>
0x7f4cacc42890:     0x7f4cacc428a0      0x4005d0 <_Z6work_5v+9>
0x7f4cacc428a0:     0x7f4cacc428b0      0x4005db <_Z6work_4v+9>
0x7f4cacc428b0:     0x7f4cacc428c0      0x4005e6 <_Z6work_3v+9>
0x7f4cacc428c0:     0x7f4cacc428d0      0x4005f1 <_Z6work_2v+9>
0x7f4cacc428d0:     0x7f4cacc428e0      0x4005fc <_Z6work_1v+9>
0x7f4cacc428e0:     0x7f4cacc42cf0      0x40060e <_Z4workv+16>
0x7f4cacc428f0:     0x0    0x0
0x7f4cacc42900:     0x0    0x0
0x7f4cacc42910:     0x0    0x0
0x7f4cacc42920:     0x0    0x0
0x7f4cacc42930:     0x0    0x0
0x7f4cacc42940:     0x0    0x0
0x7f4cacc42950:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7f4cacc42960:     0x0    0x0
```

```
0x7f4cacc42970:     0x0    0x0
0x7f4cacc42980:     0x0    0x0
0x7f4cacc42990:     0x0    0x0
0x7f4cacc429a0:     0x0    0x0
0x7f4cacc429b0:     0x0    0x0
0x7f4cacc429c0:     0x0    0x0
0x7f4cacc429d0:     0x0    0x0
0x7f4cacc429e0:     0x0    0x0
0x7f4cacc429f0:     0x0    0x0
0x7f4cacc42a00:     0x0    0x0
0x7f4cacc42a10:     0x0    0x0
0x7f4cacc42a20:     0x0    0x0
0x7f4cacc42a30:     0x0    0x0
0x7f4cacc42a40:     0x0    0x0
0x7f4cacc42a50:     0x0    0x0
0x7f4cacc42a60:     0x0    0x0
0x7f4cacc42a70:     0x0    0x0
0x7f4cacc42a80:     0x0    0x0
0x7f4cacc42a90:     0x0    0x0
0x7f4cacc42aa0:     0x0    0x0
0x7f4cacc42ab0:     0x0    0x0
0x7f4cacc42ac0:     0x0    0x0
0x7f4cacc42ad0:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7f4cacc42ae0:     0x0    0x0
0x7f4cacc42af0:     0x0    0x0
0x7f4cacc42b00:     0x0    0x0
0x7f4cacc42b10:     0x0    0x0
0x7f4cacc42b20:     0x0    0x4431e6 <nanosleep+38>
0x7f4cacc42b30:     0x0    0x4430c0 <sleep+224>
0x7f4cacc42b40:     0x0    0x0
0x7f4cacc42b50:     0x0    0x0
0x7f4cacc42b60:     0x0    0x0
0x7f4cacc42b70:     0x0    0x0
0x7f4cacc42b80:     0x0    0x0
0x7f4cacc42b90:     0x0    0x0
0x7f4cacc42ba0:     0x0    0x0
0x7f4cacc42bb0:     0x0    0x0
0x7f4cacc42bc0:     0x0    0x0
0x7f4cacc42bd0:     0x0    0x0
0x7f4cacc42be0:     0x0    0x0
0x7f4cacc42bf0:     0x0    0x0
0x7f4cacc42c00:     0x0    0x0
0x7f4cacc42c10:     0x0    0x0
0x7f4cacc42c20:     0x0    0x0
0x7f4cacc42c30:     0x0    0x0
0x7f4cacc42c40:     0x0    0x0
0x7f4cacc42c50:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7f4cacc42c60:     0x10000      0x0
0x7f4cacc42c70:     0x0    0x0
0x7f4cacc42c80:     0x0    0x0
0x7f4cacc42c90:     0x0    0x0
0x7f4cacc42ca0:     0x0    0x0
0x7f4cacc42cb0:     0x0    0x0
0x7f4cacc42cc0:     0x0    0x0
0x7f4cacc42cd0:     0x0    0x0
0x7f4cacc42ce0:     0xfffffed2   0x3ad3affa
0x7f4cacc42cf0:     0x7f4cacc42d00      0x0
0x7f4cacc42d00:     0x7f4cacc42d20      0x49c740 <default_attr>
```

```
0x7f4cacc42d10:     0x7f4cacc439c0      0x400771 <_Z6procNEv+19>
0x7f4cacc42d20:     0x7f4cacc42d30      0x4007aa <_Z7bar_twov+9>
0x7f4cacc42d30:     0x7f4cacc42d40      0x4007b5 <_Z7foo_twov+9>
0x7f4cacc42d40:     0x7f4cacc42d60      0x4007c8 <_Z10thread_twoPv+17>
0x7f4cacc42d50:     0x0    0x0
0x7f4cacc42d60:     0x0    0x4140f0 <start_thread+208>
0x7f4cacc42d70:     0x0    0x7f4cacc43700
0x7f4cacc42d80:     0x0    0x0
0x7f4cacc42d90:     0x0    0x0
0x7f4cacc42da0:     0x0    0x0
0x7f4cacc42db0:     0x0    0x0
0x7f4cacc42dc0:     0x0    0x0
0x7f4cacc42dd0:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7f4cacc42de0:     0x0    0x0
0x7f4cacc42df0:     0x0    0x0
0x7f4cacc42e00:     0x7f4cacc43700      0x4987c54266ee5578
0x7f4cacc42e10:     0x49c740 <default_attr>   0x7f4cacc439c0
0x7f4cacc42e20:     0x0    0x3
0x7f4cacc42e30:     0xb71e9cca3c0e5578  0x4987c5c0e7825578
0x7f4cacc42e40:     0x0    0x0
0x7f4cacc42e50:     0x0    0x0
0x7f4cacc42e60:     0x0    0x0
0x7f4cacc42e70:     0x7f4cacc43700      0x445879 <clone+121>
0x7f4cacc42e80:     0x0    0x0
0x7f4cacc42e90:     0x0    0x0
0x7f4cacc42ea0:     0x0    0x0
0x7f4cacc42eb0:     0x0    0x0
0x7f4cacc42ec0:     0x0    0x0
0x7f4cacc42ed0:     0x0    0x0
0x7f4cacc42ee0:     0x0    0x0
0x7f4cacc42ef0:     0x0    0x0
0x7f4cacc42f00:     0x0    0x0
0x7f4cacc42f10:     0x0    0x0
0x7f4cacc42f20:     0x0    0x0
0x7f4cacc42f30:     0x0    0x0
0x7f4cacc42f40:     0x0    0x0
0x7f4cacc42f50:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7f4cacc42f60:     0x0    0x0
0x7f4cacc42f70:     0x0    0x0
0x7f4cacc42f80:     0x0    0x0
0x7f4cacc42f90:     0x0    0x0
0x7f4cacc42fa0:     0x0    0x0
0x7f4cacc42fb0:     0x0    0x0
0x7f4cacc42fc0:     0x0    0x0
0x7f4cacc42fd0:     0x0    0x0
0x7f4cacc42fe0:     0x0    0x0
0x7f4cacc42ff0:     0x0    0x0
0x7f4cacc43000:     0x0    0x0
0x7f4cacc43010:     0x0    0x0
0x7f4cacc43020:     0x0    0x0
0x7f4cacc43030:     0x0    0x0
0x7f4cacc43040:     0x0    0x0
0x7f4cacc43050:     0x0    0x0
0x7f4cacc43060:     0x0    0x0
0x7f4cacc43070:     0x0    0x0
0x7f4cacc43080:     0x0    0x0
0x7f4cacc43090:     0x0    0x0
0x7f4cacc430a0:     0x0    0x0
```

```
0x7f4cacc430b0:     0x0    0x0
0x7f4cacc430c0:     0x0    0x0
0x7f4cacc430d0:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7f4cacc430e0:     0x0    0x0
0x7f4cacc430f0:     0x0    0x0
0x7f4cacc43100:     0x0    0x0
0x7f4cacc43110:     0x0    0x0
0x7f4cacc43120:     0x0    0x0
0x7f4cacc43130:     0x0    0x0
0x7f4cacc43140:     0x0    0x0
0x7f4cacc43150:     0x0    0x0
0x7f4cacc43160:     0x0    0x0
0x7f4cacc43170:     0x0    0x0
0x7f4cacc43180:     0x0    0x0
0x7f4cacc43190:     0x0    0x0
0x7f4cacc431a0:     0x0    0x0
0x7f4cacc431b0:     0x0    0x0
0x7f4cacc431c0:     0x0    0x0
0x7f4cacc431d0:     0x0    0x0
0x7f4cacc431e0:     0x0    0x0
0x7f4cacc431f0:     0x0    0x0
0x7f4cacc43200:     0x0    0x0
0x7f4cacc43210:     0x0    0x0
0x7f4cacc43220:     0x0    0x0
0x7f4cacc43230:     0x0    0x0
0x7f4cacc43240:     0x0    0x0
0x7f4cacc43250:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7f4cacc43260:     0x0    0x0
0x7f4cacc43270:     0x0    0x0
0x7f4cacc43280:     0x0    0x0
0x7f4cacc43290:     0x0    0x0
0x7f4cacc432a0:     0x0    0x0
0x7f4cacc432b0:     0x0    0x0
0x7f4cacc432c0:     0x0    0x0
0x7f4cacc432d0:     0x0    0x0
0x7f4cacc432e0:     0x0    0x0
0x7f4cacc432f0:     0x0    0x0
0x7f4cacc43300:     0x0    0x0
0x7f4cacc43310:     0x0    0x0
0x7f4cacc43320:     0x0    0x0
0x7f4cacc43330:     0x0    0x0
0x7f4cacc43340:     0x0    0x0
0x7f4cacc43350:     0x0    0x0

(gdb) disassemble 0x4005e6
Dump of assembler code for function _Z6work_3v:
   0x00000000004005dd <+0>:     push   %rbp
   0x00000000004005de <+1>:     mov    %rsp,%rbp
   0x00000000004005e1 <+4>:     callq  0x4005d2 <_Z6work_4v>
   0x00000000004005e6 <+9>:     pop    %rbp
   0x00000000004005e7 <+10>:    retq
End of assembler dump.
```

4. Go to the thread #3, identify handled exception processing code, and check its validity:

```
(gdb) thread 3
[Switching to thread 3 (Thread 0x7f4cac442700 (LWP 15205))]
#0  0x00000000004431f1 in nanosleep ()

(gdb) bt
#0  0x00000000004431f1 in nanosleep ()
#1  0x00000000004430c0 in sleep ()
#2  0x000000000040073c in procH() ()
#3  0x00000000004007d8 in bar_three() ()
#4  0x00000000004007e3 in foo_three() ()
#5  0x00000000004007f6 in thread_three(void*) ()
#6  0x00000000004140f0 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#7  0x0000000000445879 in clone ()
#8  0x0000000000000000 in ?? ()

(gdb) x/512a $rsp-2000
0x7f4cac441350:     0x0    0x0
0x7f4cac441360:     0x0    0x0
0x7f4cac441370:     0x0    0x0
0x7f4cac441380:     0x0    0x0
0x7f4cac441390:     0x0    0x0
0x7f4cac4413a0:     0x0    0x0
0x7f4cac4413b0:     0x0    0x0
0x7f4cac4413c0:     0x0    0x0
0x7f4cac4413d0:     0x0    0x0
0x7f4cac4413e0:     0x0    0x0
0x7f4cac4413f0:     0x0    0x0
0x7f4cac441400:     0x0    0x0
0x7f4cac441410:     0x0    0x0
0x7f4cac441420:     0x0    0x0
0x7f4cac441430:     0x0    0x0
0x7f4cac441440:     0x0    0x0
0x7f4cac441450:     0x0    0x0
0x7f4cac441460:     0x0    0x0
0x7f4cac441470:     0x0    0x0
0x7f4cac441480:     0x0    0x0
0x7f4cac441490:     0x0    0x0
0x7f4cac4414a0:     0x0    0x0
0x7f4cac4414b0:     0x0    0x0
0x7f4cac4414c0:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7f4cac4414d0:     0x0    0x0
0x7f4cac4414e0:     0x0    0x0
0x7f4cac4414f0:     0x6cdc20 <object.5602>     0x0
0x7f4cac441500:     0x1b   0x411e96 <fde_single_encoding_compare+118>
0x7f4cac441510:     0x4005a6 <_Z6work_8v>      0x4005a0 <_Z6work_9v>
0x7f4cac441520:     0x1ae91a0     0x1
0x7f4cac441530:     0x1ae91a0     0x2
0x7f4cac441540:     0x411e20 <fde_single_encoding_compare>  0x4116de <frame_downheap+78>
0x7f4cac441550:     0x0    0x6cdc20 <object.5602>
0x7f4cac441560:     0x4ca398 <__EH_FRAME_BEGIN__+66800>     0x1ae9190
0x7f4cac441570:     0x1ae91a0     0x6cdc20 <object.5602>
0x7f4cac441580:     0x411e20 <fde_single_encoding_compare>  0x2de
0x7f4cac441590:     0x6cdc20 <object.5602>     0x0
0x7f4cac4415a0:     0x1b   0x6e4e60 <main_arena>
0x7f4cac4415b0:     0x411e20 <fde_single_encoding_compare>  0x1
0x7f4cac4415c0:     0x2    0x1ae5d10
```

```
0x7f4cac4415d0:     0x1f0  0x7f4cac441610
0x7f4cac4415e0:     0x10   0x4ba0c8 <__EH_FRAME_BEGIN__+544>
0x7f4cac4415f0:     0x10   0x412599 <search_object+1209>
0x7f4cac441600:     0x7f4cac441630      0x400714 <_Z5procHv+13>
0x7f4cac441610:     0x7f4cac441638      0x7f4c0000001b
0x7f4cac441620:     0xb    0x7f4cac441cd8
0x7f4cac441630:     0x400707 <_Z5procHv>      0x42
---Type <return> to continue, or q <return> to quit---
0x7f4cac441640:     0x7f4cac441cf0      0x7f4cac441600
0x7f4cac441650:     0x0    0x6cdc20 <object.5602>
0x7f4cac441660:     0x7f4cac441a98      0x416700 <pthread_cancel>
0x7f4cac441670:     0x6cdc20 <object.5602>    0x7f4cac441a98
0x7f4cac441680:     0x1b   0x4130a0 <_Unwind_Find_FDE+208>
0x7f4cac441690:     0x7f4cac441cf8      0x401c2a <__gxx_personality_v0+202>
0x7f4cac4416a0:     0x4ca40c     0x4006fa <_Z5procAv+9>
0x7f4cac4416b0:     0x0    0x0
0x7f4cac4416c0:     0x0    0x4ba0dd <__EH_FRAME_BEGIN__+565>
0x7f4cac4416d0:     0x0    0x7f4cac4419f0
0x7f4cac4416e0:     0x4ba0dd <__EH_FRAME_BEGIN__+565>      0x3
0x7f4cac4416f0:     0x7f4cac441760      0x41022c <uw_frame_state_for+828>
0x7f4cac441700:     0x3    0x4ba033 <__EH_FRAME_BEGIN__+395>
0x7f4cac441710:     0xfffffffffffffff8  0x4ca40c
0x7f4cac441720:     0x7f4cac441900      0x7f4cac4419f0
0x7f4cac441730:     0x4    0x1ae5c90
0x7f4cac441740:     0x7f4cac441900      0x0
0x7f4cac441750:     0x3    0x410c3b <_Unwind_RaiseException_Phase2+59>
0x7f4cac441760:     0x0    0x0
0x7f4cac441770:     0x0    0x0
0x7f4cac441780:     0x0    0x0
0x7f4cac441790:     0xfffffffffffffe8  0x1
0x7f4cac4417a0:     0x0    0x0
0x7f4cac4417b0:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7f4cac4417c0:     0xfffffffffffffff0  0x1
0x7f4cac4417d0:     0x0    0x0
0x7f4cac4417e0:     0x0    0x0
0x7f4cac4417f0:     0x0    0x0
0x7f4cac441800:     0x0    0x0
0x7f4cac441810:     0x0    0x0
0x7f4cac441820:     0x0    0x0
0x7f4cac441830:     0x0    0x0
0x7f4cac441840:     0x0    0x0
0x7f4cac441850:     0x0    0x0
0x7f4cac441860:     0xfffffffffffffff8  0x1
0x7f4cac441870:     0x0    0x0
0x7f4cac441880:     0x0    0x10
0x7f4cac441890:     0x6    0x0
0x7f4cac4418a0:     0x1    0x400748 <_Z5procHv+65>
0x7f4cac4418b0:     0x401b60 <__gxx_personality_v0>  0xfffffffffffffff8
0x7f4cac4418c0:     0x1    0x4105ff <uw_install_context_1+191>
0x7f4cac4418d0:     0x7f4cac441d10      0x0
0x7f4cac4418e0:     0x7f4cac4419f0      0x7f4cac441ca0
0x7f4cac4418f0:     0x1ae5c90    0x411265 <_Unwind_RaiseException+309>
0x7f4cac441900:     0x7f4cac441c68      0x7f4cac441c70
0x7f4cac441910:     0x0    0x7f4cac441c78
0x7f4cac441920:     0x0    0x0
0x7f4cac441930:     0x7f4cac441ca0      0x0
---Type <return> to continue, or q <return> to quit---
0x7f4cac441940:     0x0    0x0
0x7f4cac441950:     0x0    0x0
```

```
0x7f4cac441960:     0x7f4cac441c80      0x7f4cac441c88
0x7f4cac441970:     0x7f4cac441c90      0x7f4cac441c98
0x7f4cac441980:     0x7f4cac441ca8      0x0
0x7f4cac441990:     0x7f4cac441cb0      0x401651 <__cxa_throw+81>
0x7f4cac4419a0:     0x0    0x0
0x7f4cac4419b0:     0x0    0x411130 <_Unwind_RaiseException>
0x7f4cac4419c0:     0x4000000000000000  0x0
0x7f4cac4419d0:     0x0    0x0
0x7f4cac4419e0:     0x0    0x0
0x7f4cac4419f0:     0x0    0x0
0x7f4cac441a00:     0x0    0x7f4cac441cd8
0x7f4cac441a10:     0x0    0x0
0x7f4cac441a20:     0x7f4cac441d00      0x7f4cac4418d0
0x7f4cac441a30:     0x0    0x0
0x7f4cac441a40:     0x0    0x0
0x7f4cac441a50:     0x7f4cac441ce0      0x7f4cac441ce8
0x7f4cac441a60:     0x7f4cac441c90      0x7f4cac441c98
0x7f4cac441a70:     0x7f4cac441d08      0x0
0x7f4cac441a80:     0x7f4cac441d10      0x40072a <_Z5procHv+35>
0x7f4cac441a90:     0x4ca40c     0x0
0x7f4cac441aa0:     0x0    0x400707 <_Z5procHv>
0x7f4cac441ab0:     0x4000000000000000  0x0
---Type <return> to continue, or q <return> to quit---
0x7f4cac441ac0:     0x0    0x0
0x7f4cac441ad0:     0x0    0x0
0x7f4cac441ae0:     0x0    0x0
0x7f4cac441af0:     0x0    0x0
0x7f4cac441b00:     0x0    0x0
0x7f4cac441b10:     0xfffffffffffffff8  0x4431e6 <nanosleep+38>
0x7f4cac441b20:     0x0    0x4430c0 <sleep+224>
0x7f4cac441b30:     0x0    0x0
0x7f4cac441b40:     0x0    0x7f4cac441cd8
0x7f4cac441b50:     0x0    0x0
0x7f4cac441b60:     0x7f4cac441d00      0x7f4cac4418d0
0x7f4cac441b70:     0x0    0x0
0x7f4cac441b80:     0x0    0x0
0x7f4cac441b90:     0x7f4cac441ce0      0x7f4cac441ce8
0x7f4cac441ba0:     0x7f4cac441c90      0x7f4cac441c98
0x7f4cac441bb0:     0x7f4cac441d08      0x0
0x7f4cac441bc0:     0x0    0x0
0x7f4cac441bd0:     0x0    0x0
0x7f4cac441be0:     0xffffffffffffffff  0x1
0x7f4cac441bf0:     0x0    0x0
0x7f4cac441c00:     0x0    0x10
0x7f4cac441c10:     0x6    0x0
0x7f4cac441c20:     0x1    0x400748 <_Z5procHv+65>
0x7f4cac441c30:     0x401b60 <__gxx_personality_v0>  0xfffffffffffffff8
---Type <return> to continue, or q <return> to quit---
0x7f4cac441c40:     0x1    0x10
0x7f4cac441c50:     0x10000      0x0
0x7f4cac441c60:     0x0    0x0
0x7f4cac441c70:     0x0    0x0
0x7f4cac441c80:     0x0    0x0
0x7f4cac441c90:     0x0    0x0
0x7f4cac441ca0:     0x0    0x0
0x7f4cac441cb0:     0x0    0x0
0x7f4cac441cc0:     0x0    0x0
0x7f4cac441cd0:     0xfffffed2   0x3ad34c4d
0x7f4cac441ce0:     0x49c740 <default_attr>   0x0
0x7f4cac441cf0:     0x7f4cac441d20      0x49c740 <default_attr>
```

```
0x7f4cac441d00:     0x7f4cac4429c0      0x40073c <_Z5procHv+53>
0x7f4cac441d10:     0x0    0x0
0x7f4cac441d20:     0x7f4cac441d30      0x4007d8 <_Z9bar_threev+9>
0x7f4cac441d30:     0x7f4cac441d40      0x4007e3 <_Z9foo_threev+9>
0x7f4cac441d40:     0x7f4cac441d60      0x4007f6 <_Z12thread_threePv+17>
0x7f4cac441d50:     0x0    0x0
0x7f4cac441d60:     0x0    0x4140f0 <start_thread+208>
0x7f4cac441d70:     0x0    0x7f4cac442700
0x7f4cac441d80:     0x0    0x0
0x7f4cac441d90:     0x0    0x0
0x7f4cac441da0:     0x0    0x0
0x7f4cac441db0:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7f4cac441dc0:     0x0    0x0
0x7f4cac441dd0:     0x0    0x0
0x7f4cac441de0:     0x0    0x0
0x7f4cac441df0:     0x0    0x0
0x7f4cac441e00:     0x7f4cac442700      0x4987c54266ee5578
0x7f4cac441e10:     0x49c740 <default_attr>   0x7f4cac4429c0
0x7f4cac441e20:     0x0    0x3
0x7f4cac441e30:     0xb71e9dca5c0e5578  0x4987c5c0e7825578
0x7f4cac441e40:     0x0    0x0
0x7f4cac441e50:     0x0    0x0
0x7f4cac441e60:     0x0    0x0
0x7f4cac441e70:     0x7f4cac442700      0x445879 <clone+121>
0x7f4cac441e80:     0x0    0x0
0x7f4cac441e90:     0x0    0x0
0x7f4cac441ea0:     0x0    0x0
0x7f4cac441eb0:     0x0    0x0
0x7f4cac441ec0:     0x0    0x0
0x7f4cac441ed0:     0x0    0x0
0x7f4cac441ee0:     0x0    0x0
0x7f4cac441ef0:     0x0    0x0
0x7f4cac441f00:     0x0    0x0
0x7f4cac441f10:     0x0    0x0
0x7f4cac441f20:     0x0    0x0
0x7f4cac441f30:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7f4cac441f40:     0x0    0x0
0x7f4cac441f50:     0x0    0x0
0x7f4cac441f60:     0x0    0x0
0x7f4cac441f70:     0x0    0x0
0x7f4cac441f80:     0x0    0x0
0x7f4cac441f90:     0x0    0x0
0x7f4cac441fa0:     0x0    0x0
0x7f4cac441fb0:     0x0    0x0
0x7f4cac441fc0:     0x0    0x0
0x7f4cac441fd0:     0x0    0x0
0x7f4cac441fe0:     0x0    0x0
0x7f4cac441ff0:     0x0    0x0
0x7f4cac442000:     0x0    0x0
0x7f4cac442010:     0x0    0x0
0x7f4cac442020:     0x0    0x0
0x7f4cac442030:     0x0    0x0
0x7f4cac442040:     0x0    0x0
0x7f4cac442050:     0x0    0x0
0x7f4cac442060:     0x0    0x0
0x7f4cac442070:     0x0    0x0
0x7f4cac442080:     0x0    0x0
0x7f4cac442090:     0x0    0x0
```

```
0x7f4cac4420a0:     0x0    0x0
0x7f4cac4420b0:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7f4cac4420c0:     0x0    0x0
0x7f4cac4420d0:     0x0    0x0
0x7f4cac4420e0:     0x0    0x0
0x7f4cac4420f0:     0x0    0x0
0x7f4cac442100:     0x0    0x0
0x7f4cac442110:     0x0    0x0
0x7f4cac442120:     0x0    0x0
0x7f4cac442130:     0x0    0x0
0x7f4cac442140:     0x0    0x0
0x7f4cac442150:     0x0    0x0
0x7f4cac442160:     0x0    0x0
0x7f4cac442170:     0x0    0x0
0x7f4cac442180:     0x0    0x0
0x7f4cac442190:     0x0    0x0
0x7f4cac4421a0:     0x0    0x0
0x7f4cac4421b0:     0x0    0x0
0x7f4cac4421c0:     0x0    0x0
0x7f4cac4421d0:     0x0    0x0
0x7f4cac4421e0:     0x0    0x0
0x7f4cac4421f0:     0x0    0x0
0x7f4cac442200:     0x0    0x0
0x7f4cac442210:     0x0    0x0
0x7f4cac442220:     0x0    0x0
0x7f4cac442230:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7f4cac442240:     0x0    0x0
0x7f4cac442250:     0x0    0x0
0x7f4cac442260:     0x0    0x0
0x7f4cac442270:     0x0    0x0
0x7f4cac442280:     0x0    0x0
0x7f4cac442290:     0x0    0x0
0x7f4cac4422a0:     0x0    0x0
0x7f4cac4422b0:     0x0    0x0
0x7f4cac4422c0:     0x0    0x0
0x7f4cac4422d0:     0x0    0x0
0x7f4cac4422e0:     0x0    0x0
0x7f4cac4422f0:     0x0    0x0
0x7f4cac442300:     0x0    0x0
0x7f4cac442310:     0x0    0x0
0x7f4cac442320:     0x0    0x0
0x7f4cac442330:     0x0    0x0
0x7f4cac442340:     0x0    0x0

(gdb) disassemble 0x411265
Dump of assembler code for function _Unwind_RaiseException:
   0x0000000000411130 <+0>:     push   %rbp
   0x0000000000411131 <+1>:     mov    %rsp,%rbp
   0x0000000000411134 <+4>:     push   %r15
   0x0000000000411136 <+6>:     lea    0x10(%rbp),%rsi
   0x000000000041113a <+10>:    push   %r14
   0x000000000041113c <+12>:    push   %r13
   0x000000000041113e <+14>:    lea    -0x3a0(%rbp),%r13
   0x0000000000411145 <+21>:    push   %r12
   0x0000000000411147 <+23>:    mov    %rdi,%r12
   0x000000000041114a <+26>:    mov    %r13,%rdi
   0x000000000041114d <+29>:    push   %rbx
   0x000000000041114e <+30>:    lea    -0x2b0(%rbp),%rbx
```

```
   0x0000000000411155 <+37>:    push   %rdx
   0x0000000000411156 <+38>:    push   %rax
   0x0000000000411157 <+39>:    sub    $0x368,%rsp
   0x000000000041115e <+46>:    mov    0x8(%rbp),%rdx
   0x0000000000411162 <+50>:    callq  0x410ca0 <uw_init_context_1>
   0x0000000000411167 <+55>:    mov    $0x1e,%ecx
   0x000000000041116c <+60>:    mov    %rbx,%rdi
   0x000000000041116f <+63>:    mov    %r13,%rsi
   0x0000000000411172 <+66>:    rep movsq %ds:(%rsi),%es:(%rdi)
   0x0000000000411175 <+69>:    jmp    0x4111bc <_Unwind_RaiseException+140>
   0x0000000000411177 <+71>:    nopw   0x0(%rax,%rax,1)
---Type <return> to continue, or q <return> to quit---
   0x0000000000411180 <+80>:    test   %eax,%eax
   0x0000000000411182 <+82>:    jne    0x4111f0 <_Unwind_RaiseException+192>
   0x0000000000411184 <+84>:    mov    -0x70(%rbp),%rax
   0x0000000000411188 <+88>:    test   %rax,%rax
   0x000000000041118b <+91>:    je     0x4111ad <_Unwind_RaiseException+125>
   0x000000000041118d <+93>:    mov    %rbx,%r8
   0x0000000000411190 <+96>:    mov    %r12,%rcx
   0x0000000000411193 <+99>:    mov    (%r12),%rdx
   0x0000000000411197 <+103>:   mov    $0x1,%esi
   0x000000000041119c <+108>:   mov    $0x1,%edi
   0x00000000004111a1 <+113>:   callq  *%rax
   0x00000000004111a3 <+115>:   cmp    $0x6,%eax
   0x00000000004111a6 <+118>:   je     0x411200 <_Unwind_RaiseException+208>
   0x00000000004111a8 <+120>:   cmp    $0x8,%eax
   0x00000000004111ab <+123>:   jne    0x4111f0 <_Unwind_RaiseException+192>
   0x00000000004111ad <+125>:   lea    -0x1c0(%rbp),%rsi
   0x00000000004111b4 <+132>:   mov    %rbx,%rdi
   0x00000000004111b7 <+135>:   callq  0x410a60 <uw_update_context>
   0x00000000004111bc <+140>:   lea    -0x1c0(%rbp),%rsi
   0x00000000004111c3 <+147>:   mov    %rbx,%rdi
   0x00000000004111c6 <+150>:   callq  0x40fef0 <uw_frame_state_for>
   0x00000000004111cb <+155>:   cmp    $0x5,%eax
   0x00000000004111ce <+158>:   jne    0x411180 <_Unwind_RaiseException+80>
   0x00000000004111d0 <+160>:   mov    -0x28(%rbp),%rbx
---Type <return> to continue, or q <return> to quit---
   0x00000000004111d4 <+164>:   mov    -0x20(%rbp),%r12
   0x00000000004111d8 <+168>:   mov    -0x18(%rbp),%r13
   0x00000000004111dc <+172>:   mov    -0x10(%rbp),%r14
   0x00000000004111e0 <+176>:   mov    -0x8(%rbp),%r15
   0x00000000004111e4 <+180>:   leaveq
   0x00000000004111e5 <+181>:   retq
   0x00000000004111e6 <+182>:   nopw   %cs:0x0(%rax,%rax,1)
   0x00000000004111f0 <+192>:   mov    $0x3,%eax
   0x00000000004111f5 <+197>:   jmp    0x4111d0 <_Unwind_RaiseException+160>
   0x00000000004111f7 <+199>:   nopw   0x0(%rax,%rax,1)
   0x0000000000411200 <+208>:   mov    -0x1f0(%rbp),%rdx
   0x0000000000411207 <+215>:   mov    -0x220(%rbp),%rax
   0x000000000041120e <+222>:   mov    $0x1e,%ecx
   0x0000000000411213 <+227>:   movq   $0x0,0x10(%r12)
   0x000000000041121c <+236>:   mov    %rbx,%rdi
   0x000000000041121f <+239>:   mov    %r13,%rsi
   0x0000000000411222 <+242>:   shr    $0x3f,%rdx
   0x0000000000411226 <+246>:   sub    %rdx,%rax
   0x0000000000411229 <+249>:   mov    %rax,0x18(%r12)
   0x000000000041122e <+254>:   rep movsq %ds:(%rsi),%es:(%rdi)
   0x0000000000411231 <+257>:   mov    %rbx,%rsi
   0x0000000000411234 <+260>:   mov    %r12,%rdi
   0x0000000000411237 <+263>:   callq  0x410c00 <_Unwind_RaiseException_Phase2>
```

```
   0x000000000041123c <+268>:    cmp    $0x7,%eax
---Type <return> to continue, or q <return> to quit---
   0x000000000041123f <+271>:    jne    0x4111d0 <_Unwind_RaiseException+160>
   0x0000000000411241 <+273>:    mov    %rbx,%rsi
   0x0000000000411244 <+276>:    mov    %r13,%rdi
   0x0000000000411247 <+279>:    callq  0x410540 <uw_install_context_1>
   0x000000000041124c <+284>:    mov    -0x218(%rbp),%r12
   0x0000000000411253 <+291>:    mov    -0x220(%rbp),%rdi
   0x000000000041125a <+298>:    mov    %rax,%rbx
   0x000000000041125d <+301>:    mov    %r12,%rsi
   0x0000000000411260 <+304>:    callq  0x411120 <_Unwind_DebugHook>
   0x0000000000411265 <+309>:    mov    %rbx,%rcx
   0x0000000000411268 <+312>:    mov    %r12,0x8(%rbp,%rbx,1)
   0x000000000041126d <+317>:    mov    -0x38(%rbp),%rax
   0x0000000000411271 <+321>:    lea    0x8(%rbp,%rcx,1),%rcx
   0x0000000000411276 <+326>:    mov    -0x30(%rbp),%rdx
   0x000000000041127a <+330>:    mov    -0x28(%rbp),%rbx
   0x000000000041127e <+334>:    mov    -0x20(%rbp),%r12
   0x0000000000411282 <+338>:    mov    -0x18(%rbp),%r13
   0x0000000000411286 <+342>:    mov    -0x10(%rbp),%r14
   0x000000000041128a <+346>:    mov    -0x8(%rbp),%r15
   0x000000000041128e <+350>:    mov    0x0(%rbp),%rbp
   0x0000000000411292 <+354>:    mov    %rcx,%rsp
   0x0000000000411295 <+357>:    retq
End of assembler dump.
```

Exercise A9

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

© 2015 Software Diagnostics Services

Exercise A9

Goal: Learn how to identify heap leaks.

Patterns: Heap Leak, Execution Residue, Module Hint.

1. The application App9 was found consuming more and more memory. Several core memory dumps were saved at different times with corresponding pmap logs. Load a core dump *core.16531.2* and *App9* executable:

```
training@debian64:~/ALCDA$ gdb -c ./App9/core.16531.2 -se ./App9/App9
GNU gdb (GDB) 7.4.1-debian
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/training/ALCDA/App9/App9...done.
[New LWP 16532]
[New LWP 16533]
[New LWP 16534]
[New LWP 16535]
[New LWP 16536]
[New LWP 16531]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Core was generated by `/home/training/ALCDA/App9/App9'.
#0  0x000000000042feb1 in nanosleep ()
```

2. Notice the size of the largest section and quit gdb:

```
(gdb) maintenance info sections
Exec file:
    `/home/training/ALCDA/App9/App9', file type elf64-x86-64.
    0x00400158->0x00400178 at 0x00000158: .note.ABI-tag ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x00400178->0x0040019c at 0x00000178: .note.gnu.build-id ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004001a0->0x004002d8 at 0x000001a0: .rela.plt ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004002d8->0x004002e6 at 0x000002d8: .init ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x004002f0->0x004003c0 at 0x000002f0: .plt ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x004003c0->0x0048b278 at 0x000003c0: .text ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x0048b280->0x0048bdfe at 0x0008b280: __libc_freeres_fn ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x0048be00->0x0048be61 at 0x0008be00: __libc_thread_freeres_fn ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x0048be64->0x0048be6d at 0x0008be64: .fini ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x0048be80->0x004a9de4 at 0x0008be80: .rodata ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004a9de8->0x004a9e48 at 0x000a9de8: __libc_subfreeres ALLOC LOAD READONLY DATA HAS_CONTENTS
---Type <return> to continue, or q <return> to quit---
    0x004a9e48->0x004a9e50 at 0x000a9e48: __libc_atexit ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004a9e50->0x004a9e58 at 0x000a9e50: __libc_thread_subfreeres ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004a9e58->0x004b69ac at 0x000a9e58: .eh_frame ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004b69ac->0x004b6ac6 at 0x000b69ac: .gcc_except_table ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x006b6ac8->0x006b6af0 at 0x000b6ac8: .tdata ALLOC LOAD DATA HAS_CONTENTS
    0x006b6af0->0x006b6b20 at 0x000b6af0: .tbss ALLOC
    0x006b6af0->0x006b6b00 at 0x000b6af0: .init_array ALLOC LOAD DATA HAS_CONTENTS
    0x006b6b00->0x006b6b10 at 0x000b6b00: .fini_array ALLOC LOAD DATA HAS_CONTENTS
    0x006b6b10->0x006b6b18 at 0x000b6b10: .jcr ALLOC LOAD DATA HAS_CONTENTS
    0x006b6b20->0x006b6b90 at 0x000b6b20: .data.rel.ro ALLOC LOAD DATA HAS_CONTENTS
```

```
    0x006b6b90->0x006b6ba0 at 0x000b6b90: .got ALLOC LOAD DATA HAS_CONTENTS
    0x006b6ba0->0x006b6c20 at 0x000b6ba0: .got.plt ALLOC LOAD DATA HAS_CONTENTS
    0x006b6c20->0x006b7930 at 0x000b6c20: .data ALLOC LOAD DATA HAS_CONTENTS
    0x006b7940->0x006beca8 at 0x000b7930: .bss ALLOC
    0x006beca8->0x006becd8 at 0x000b7930: __libc_freeres_ptrs ALLOC
    0x00000000->0x00000038 at 0x000b7930: .comment READONLY HAS_CONTENTS
    0x00000000->0x00000390 at 0x000b7970: .debug_aranges READONLY HAS_CONTENTS
---Type <return> to continue, or q <return> to quit---
    0x00000000->0x00000ac3 at 0x000b7d00: .debug_pubnames READONLY HAS_CONTENTS
    0x00000000->0x00011440 at 0x000b87c3: .debug_info READONLY HAS_CONTENTS
    0x00000000->0x000021b1 at 0x000c9c03: .debug_abbrev READONLY HAS_CONTENTS
    0x00000000->0x00002ebc at 0x000cbdb4: .debug_line READONLY HAS_CONTENTS
    0x00000000->0x000038da at 0x000cec70: .debug_str READONLY HAS_CONTENTS
    0x00000000->0x0000878e at 0x000d254a: .debug_loc READONLY HAS_CONTENTS
    0x00000000->0x00001280 at 0x000dacd8: .debug_ranges READONLY HAS_CONTENTS
Core file:
    `/home/training/ALCDA/./App9/core.16531.2', file type elf64-x86-64.
    0x00000000->0x00002aa8 at 0x00000318: note0 READONLY HAS_CONTENTS
    0x00000000->0x000000d8 at 0x00000438: .reg/16532 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x00000438: .reg HAS_CONTENTS
    0x00000000->0x00000200 at 0x0000052c: .reg2/16532 HAS_CONTENTS
    0x00000000->0x00000200 at 0x0000052c: .reg2 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00000740: .reg-xstate/16532 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00000740: .reg-xstate HAS_CONTENTS
    0x00000000->0x000000d8 at 0x00000b04: .reg/16533 HAS_CONTENTS
    0x00000000->0x00000200 at 0x00000bf8: .reg2/16533 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00000e0c: .reg-xstate/16533 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x000011d0: .reg/16534 HAS_CONTENTS
    0x00000000->0x00000200 at 0x000012c4: .reg2/16534 HAS_CONTENTS
    0x00000000->0x00000340 at 0x000014d8: .reg-xstate/16534 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x0000189c: .reg/16535 HAS_CONTENTS
    0x00000000->0x00000200 at 0x00001990: .reg2/16535 HAS_CONTENTS
---Type <return> to continue, or q <return> to quit---
    0x00000000->0x00000340 at 0x00001ba4: .reg-xstate/16535 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x00001f68: .reg/16536 HAS_CONTENTS
    0x00000000->0x00000200 at 0x0000205c: .reg2/16536 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00002270: .reg-xstate/16536 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x00002634: .reg/16531 HAS_CONTENTS
    0x00000000->0x00000200 at 0x00002728: .reg2/16531 HAS_CONTENTS
    0x00000000->0x00000340 at 0x0000293c: .reg-xstate/16531 HAS_CONTENTS
    0x00000000->0x00000130 at 0x00002c90: .auxv HAS_CONTENTS
    0x00400000->0x00400000 at 0x00002dc0: load1 ALLOC READONLY CODE
    0x006b6000->0x006b8000 at 0x00002dc0: load2 ALLOC LOAD HAS_CONTENTS
    0x006b8000->0x006bf000 at 0x00004dc0: load3 ALLOC LOAD HAS_CONTENTS
    0x00986000->0x04bca000 at 0x0000bdc0: load4 ALLOC LOAD HAS_CONTENTS
    0x7f5eca49b000->0x7f5ecac9b000 at 0x0424fdc0: load5 ALLOC LOAD HAS_CONTENTS
    0x7f5ecac9c000->0x7f5ecb49c000 at 0x04a4fdc0: load6 ALLOC LOAD HAS_CONTENTS
    0x7f5ecb49d000->0x7f5ecbc9d000 at 0x0524fdc0: load7 ALLOC LOAD HAS_CONTENTS
    0x7f5ecbc9e000->0x7f5ecc49e000 at 0x05a4fdc0: load8 ALLOC LOAD HAS_CONTENTS
    0x7f5ecc49f000->0x7f5eccc9f000 at 0x0624fdc0: load9 ALLOC LOAD HAS_CONTENTS
    0x7fffe9d7b000->0x7fffe9d9c000 at 0x06a4fdc0: load10 ALLOC LOAD HAS_CONTENTS
    0x7fffe9d9d000->0x7fffe9d9e000 at 0x06a70dc0: load11 ALLOC LOAD READONLY CODE HAS_CONTENTS
    0xffffffffff600000->0xffffffffff601000 at 0x06a71dc0: load12 ALLOC LOAD READONLY CODE HAS_CONTENTS

(gdb) q
```

3. Load a core dump *core.16531.3* and *App9* executable:

```
training@debian64:~/ALCDA$ gdb -c ./App9/core.16531.3 -se ./App9/App9
GNU gdb (GDB) 7.4.1-debian
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/training/ALCDA/App9/App9...done.
[New LWP 16532]
[New LWP 16533]
[New LWP 16534]
[New LWP 16535]
[New LWP 16536]
[New LWP 16531]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Core was generated by `/home/training/ALCDA/App9/App9'.
#0  0x0000000000042feb1 in nanosleep ()
```

4. Notice that the size of the largest section increased considerably after some time.

```
(gdb) maintenance info sections
Exec file:
    `/home/training/ALCDA/App9/App9', file type elf64-x86-64.
    0x00400158->0x00400178 at 0x00000158: .note.ABI-tag ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x00400178->0x0040019c at 0x00000178: .note.gnu.build-id ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004001a0->0x004002d8 at 0x000001a0: .rela.plt ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004002d8->0x004002e6 at 0x000002d8: .init ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x004002f0->0x004003c0 at 0x000002f0: .plt ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x004003c0->0x0048b278 at 0x000003c0: .text ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x0048b280->0x0048bdfe at 0x0008b280: __libc_freeres_fn ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x0048be00->0x0048be61 at 0x0008be00: __libc_thread_freeres_fn ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x0048be64->0x0048be6d at 0x0008be64: .fini ALLOC LOAD READONLY CODE HAS_CONTENTS
    0x0048be80->0x004a9de4 at 0x0008be80: .rodata ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004a9de8->0x004a9e48 at 0x000a9de8: __libc_subfreeres ALLOC LOAD READONLY DATA HAS_CONTENTS
---Type <return> to continue, or q <return> to quit---
    0x004a9e48->0x004a9e50 at 0x000a9e48: __libc_atexit ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004a9e50->0x004a9e58 at 0x000a9e50: __libc_thread_subfreeres ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004a9e58->0x004b69ac at 0x000a9e58: .eh_frame ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x004b69ac->0x004b6ac6 at 0x000b69ac: .gcc_except_table ALLOC LOAD READONLY DATA HAS_CONTENTS
    0x006b6ac8->0x006b6af0 at 0x000b6ac8: .tdata ALLOC LOAD DATA HAS_CONTENTS
    0x006b6af0->0x006b6b20 at 0x000b6af0: .tbss ALLOC
    0x006b6af0->0x006b6b00 at 0x000b6af0: .init_array ALLOC LOAD DATA HAS_CONTENTS
    0x006b6b00->0x006b6b10 at 0x000b6b00: .fini_array ALLOC LOAD DATA HAS_CONTENTS
    0x006b6b10->0x006b6b18 at 0x000b6b10: .jcr ALLOC LOAD DATA HAS_CONTENTS
    0x006b6b20->0x006b6b90 at 0x000b6b20: .data.rel.ro ALLOC LOAD DATA HAS_CONTENTS
    0x006b6b90->0x006b6ba0 at 0x000b6b90: .got ALLOC LOAD DATA HAS_CONTENTS
    0x006b6ba0->0x006b6c20 at 0x000b6ba0: .got.plt ALLOC LOAD DATA HAS_CONTENTS
    0x006b6c20->0x006b7930 at 0x000b6c20: .data ALLOC LOAD DATA HAS_CONTENTS
    0x006b7940->0x006beca8 at 0x000b7930: .bss ALLOC
    0x006beca8->0x006becd8 at 0x000b7930: __libc_freeres_ptrs ALLOC
    0x00000000->0x00000038 at 0x000b7930: .comment READONLY HAS_CONTENTS
    0x00000000->0x00000390 at 0x000b7970: .debug_aranges READONLY HAS_CONTENTS
---Type <return> to continue, or q <return> to quit---
    0x00000000->0x00000ac3 at 0x000b7d00: .debug_pubnames READONLY HAS_CONTENTS
    0x00000000->0x00011440 at 0x000b87c3: .debug_info READONLY HAS_CONTENTS
    0x00000000->0x000021b1 at 0x000c9c03: .debug_abbrev READONLY HAS_CONTENTS
    0x00000000->0x00002ebc at 0x000cbdb4: .debug_line READONLY HAS_CONTENTS
    0x00000000->0x000038da at 0x000cec70: .debug_str READONLY HAS_CONTENTS
```

```
    0x00000000->0x0000878e at 0x000d254a: .debug_loc READONLY HAS_CONTENTS
    0x00000000->0x00001280 at 0x000dacd8: .debug_ranges READONLY HAS_CONTENTS
Core file:
    `/home/training/ALCDA/./App9/core.16531.3', file type elf64-x86-64.
    0x00000000->0x00002aa8 at 0x00000318: note0 READONLY HAS_CONTENTS
    0x00000000->0x000000d8 at 0x00000438: .reg/16532 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x00000438: .reg HAS_CONTENTS
    0x00000000->0x00000200 at 0x0000052c: .reg2/16532 HAS_CONTENTS
    0x00000000->0x00000200 at 0x0000052c: .reg2 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00000740: .reg-xstate/16532 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00000740: .reg-xstate HAS_CONTENTS
    0x00000000->0x000000d8 at 0x00000b04: .reg/16533 HAS_CONTENTS
    0x00000000->0x00000200 at 0x00000bf8: .reg2/16533 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00000e0c: .reg-xstate/16533 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x000011d0: .reg/16534 HAS_CONTENTS
    0x00000000->0x00000200 at 0x000012c4: .reg2/16534 HAS_CONTENTS
    0x00000000->0x00000340 at 0x000014d8: .reg-xstate/16534 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x0000189c: .reg/16535 HAS_CONTENTS
    0x00000000->0x00000200 at 0x00001990: .reg2/16535 HAS_CONTENTS
---Type <return> to continue, or q <return> to quit---
    0x00000000->0x00000340 at 0x00001ba4: .reg-xstate/16535 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x00001f68: .reg/16536 HAS_CONTENTS
    0x00000000->0x00000200 at 0x0000205c: .reg2/16536 HAS_CONTENTS
    0x00000000->0x00000340 at 0x00002270: .reg-xstate/16536 HAS_CONTENTS
    0x00000000->0x000000d8 at 0x00002634: .reg/16531 HAS_CONTENTS
    0x00000000->0x00000200 at 0x00002728: .reg2/16531 HAS_CONTENTS
    0x00000000->0x00000340 at 0x0000293c: .reg-xstate/16531 HAS_CONTENTS
    0x00000000->0x00000130 at 0x00002c90: .auxv HAS_CONTENTS
    0x00400000->0x00400000 at 0x00002dc0: load1 ALLOC READONLY CODE
    0x006b6000->0x006b8000 at 0x00002dc0: load2 ALLOC LOAD HAS_CONTENTS
    0x006b8000->0x006bf000 at 0x00004dc0: load3 ALLOC LOAD HAS_CONTENTS
    0x00986000->0x08ca1000 at 0x0000bdc0: load4 ALLOC LOAD HAS_CONTENTS
    0x7f5eca49b000->0x7f5ecac9b000 at 0x08326dc0: load5 ALLOC LOAD HAS_CONTENTS
    0x7f5ecac9c000->0x7f5ecb49c000 at 0x08b26dc0: load6 ALLOC LOAD HAS_CONTENTS
    0x7f5ecb49d000->0x7f5ecbc9d000 at 0x09326dc0: load7 ALLOC LOAD HAS_CONTENTS
    0x7f5ecbc9e000->0x7f5ecc49e000 at 0x09b26dc0: load8 ALLOC LOAD HAS_CONTENTS
    0x7f5ecc49f000->0x7f5eccc9f000 at 0x0a326dc0: load9 ALLOC LOAD HAS_CONTENTS
    0x7fffe9d7b000->0x7fffe9d9c000 at 0x0ab26dc0: load10 ALLOC LOAD HAS_CONTENTS
    0x7fffe9d9d000->0x7fffe9d9e000 at 0x0ab47dc0: load11 ALLOC LOAD READONLY CODE HAS_CONTENTS
    0xffffffffff600000->0xffffffffff601000 at 0x0ab48dc0: load12 ALLOC LOAD READONLY CODE HAS_CONTENTS
```

5. Examine section contents for any execution residue and hints (we choose some smaller address range from the section address range):

```
(gdb) x/1000a 0x7ca1000
0x7ca1000:    0x0    0x0
0x7ca1010:    0x0    0x111
0x7ca1020:    0x657461636f6c6c61  0x79726f6d656d2064
0x7ca1030:    0x0    0x0
0x7ca1040:    0x4004f0 <procD>    0x0
0x7ca1050:    0x0    0x0
0x7ca1060:    0x0    0x0
0x7ca1070:    0x0    0x0
0x7ca1080:    0x0    0x0
0x7ca1090:    0x0    0x0
0x7ca10a0:    0x0    0x0
0x7ca10b0:    0x0    0x0
0x7ca10c0:    0x0    0x0
0x7ca10d0:    0x0    0x0
0x7ca10e0:    0x0    0x0
0x7ca10f0:    0x0    0x0
0x7ca1100:    0x0    0x0
0x7ca1110:    0x0    0x0
0x7ca1120:    0x0    0x111
```

```
0x7ca1130:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca1140:   0x0    0x0
0x7ca1150:   0x4004f0 <procD>    0x0
0x7ca1160:   0x0    0x0
0x7ca1170:   0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7ca1180:   0x0    0x0
0x7ca1190:   0x0    0x0
0x7ca11a0:   0x0    0x0
0x7ca11b0:   0x0    0x0
0x7ca11c0:   0x0    0x0
0x7ca11d0:   0x0    0x0
0x7ca11e0:   0x0    0x0
0x7ca11f0:   0x0    0x0
0x7ca1200:   0x0    0x0
0x7ca1210:   0x0    0x0
0x7ca1220:   0x0    0x0
0x7ca1230:   0x0    0x111
0x7ca1240:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca1250:   0x0    0x0
0x7ca1260:   0x4004f0 <procD>    0x0
0x7ca1270:   0x0    0x0
0x7ca1280:   0x0    0x0
0x7ca1290:   0x0    0x0
0x7ca12a0:   0x0    0x0
0x7ca12b0:   0x0    0x0
0x7ca12c0:   0x0    0x0
0x7ca12d0:   0x0    0x0
0x7ca12e0:   0x0    0x0
0x7ca12f0:   0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7ca1300:   0x0    0x0
0x7ca1310:   0x0    0x0
0x7ca1320:   0x0    0x0
0x7ca1330:   0x0    0x0
0x7ca1340:   0x0    0x111
0x7ca1350:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca1360:   0x0    0x0
0x7ca1370:   0x4004f0 <procD>    0x0
0x7ca1380:   0x0    0x0
0x7ca1390:   0x0    0x0
0x7ca13a0:   0x0    0x0
0x7ca13b0:   0x0    0x0
0x7ca13c0:   0x0    0x0
0x7ca13d0:   0x0    0x0
0x7ca13e0:   0x0    0x0
0x7ca13f0:   0x0    0x0
0x7ca1400:   0x0    0x0
0x7ca1410:   0x0    0x0
0x7ca1420:   0x0    0x0
0x7ca1430:   0x0    0x0
0x7ca1440:   0x0    0x0
0x7ca1450:   0x0    0x111
0x7ca1460:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca1470:   0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7ca1480:   0x4004f0 <procD>    0x0
0x7ca1490:   0x0    0x0
0x7ca14a0:   0x0    0x0
0x7ca14b0:   0x0    0x0
```

```
0x7ca14c0:   0x0    0x0
0x7ca14d0:   0x0    0x0
0x7ca14e0:   0x0    0x0
0x7ca14f0:   0x0    0x0
0x7ca1500:   0x0    0x0
0x7ca1510:   0x0    0x0
0x7ca1520:   0x0    0x0
0x7ca1530:   0x0    0x0
0x7ca1540:   0x0    0x0
0x7ca1550:   0x0    0x0
0x7ca1560:   0x0    0x111
0x7ca1570:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca1580:   0x0    0x0
0x7ca1590:   0x4004f0 <procD>    0x0
0x7ca15a0:   0x0    0x0
0x7ca15b0:   0x0    0x0
0x7ca15c0:   0x0    0x0
0x7ca15d0:   0x0    0x0
0x7ca15e0:   0x0    0x0
0x7ca15f0:   0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7ca1600:   0x0    0x0
0x7ca1610:   0x0    0x0
0x7ca1620:   0x0    0x0
0x7ca1630:   0x0    0x0
0x7ca1640:   0x0    0x0
0x7ca1650:   0x0    0x0
0x7ca1660:   0x0    0x0
0x7ca1670:   0x0    0x111
0x7ca1680:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca1690:   0x0    0x0
0x7ca16a0:   0x4004f0 <procD>    0x0
0x7ca16b0:   0x0    0x0
0x7ca16c0:   0x0    0x0
0x7ca16d0:   0x0    0x0
0x7ca16e0:   0x0    0x0
0x7ca16f0:   0x0    0x0
0x7ca1700:   0x0    0x0
0x7ca1710:   0x0    0x0
0x7ca1720:   0x0    0x0
0x7ca1730:   0x0    0x0
0x7ca1740:   0x0    0x0
0x7ca1750:   0x0    0x0
0x7ca1760:   0x0    0x0
0x7ca1770:   0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7ca1780:   0x0    0x111
0x7ca1790:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca17a0:   0x0    0x0
0x7ca17b0:   0x4004f0 <procD>    0x0
0x7ca17c0:   0x0    0x0
0x7ca17d0:   0x0    0x0
0x7ca17e0:   0x0    0x0
0x7ca17f0:   0x0    0x0
0x7ca1800:   0x0    0x0
0x7ca1810:   0x0    0x0
0x7ca1820:   0x0    0x0
0x7ca1830:   0x0    0x0
0x7ca1840:   0x0    0x0
0x7ca1850:   0x0    0x0
```

```
0x7ca1860:   0x0    0x0
0x7ca1870:   0x0    0x0
0x7ca1880:   0x0    0x0
0x7ca1890:   0x0    0x111
0x7ca18a0:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca18b0:   0x0    0x0
0x7ca18c0:   0x4004f0 <procD>    0x0
0x7ca18d0:   0x0    0x0
0x7ca18e0:   0x0    0x0
0x7ca18f0:   0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7ca1900:   0x0    0x0
0x7ca1910:   0x0    0x0
0x7ca1920:   0x0    0x0
0x7ca1930:   0x0    0x0
0x7ca1940:   0x0    0x0
0x7ca1950:   0x0    0x0
0x7ca1960:   0x0    0x0
0x7ca1970:   0x0    0x0
0x7ca1980:   0x0    0x0
0x7ca1990:   0x0    0x0
0x7ca19a0:   0x0    0x111
0x7ca19b0:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca19c0:   0x0    0x0
0x7ca19d0:   0x4004f0 <procD>    0x0
0x7ca19e0:   0x0    0x0
0x7ca19f0:   0x0    0x0
0x7ca1a00:   0x0    0x0
0x7ca1a10:   0x0    0x0
0x7ca1a20:   0x0    0x0
0x7ca1a30:   0x0    0x0
0x7ca1a40:   0x0    0x0
0x7ca1a50:   0x0    0x0
0x7ca1a60:   0x0    0x0
0x7ca1a70:   0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7ca1a80:   0x0    0x0
0x7ca1a90:   0x0    0x0
0x7ca1aa0:   0x0    0x0
0x7ca1ab0:   0x0    0x111
0x7ca1ac0:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca1ad0:   0x0    0x0
0x7ca1ae0:   0x4004f0 <procD>    0x0
0x7ca1af0:   0x0    0x0
0x7ca1b00:   0x0    0x0
0x7ca1b10:   0x0    0x0
0x7ca1b20:   0x0    0x0
0x7ca1b30:   0x0    0x0
0x7ca1b40:   0x0    0x0
0x7ca1b50:   0x0    0x0
0x7ca1b60:   0x0    0x0
0x7ca1b70:   0x0    0x0
0x7ca1b80:   0x0    0x0
0x7ca1b90:   0x0    0x0
0x7ca1ba0:   0x0    0x0
0x7ca1bb0:   0x0    0x0
0x7ca1bc0:   0x0    0x111
0x7ca1bd0:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca1be0:   0x0    0x0
0x7ca1bf0:   0x4004f0 <procD>    0x0
```

```
---Type <return> to continue, or q <return> to quit---
0x7ca1c00:   0x0    0x0
0x7ca1c10:   0x0    0x0
0x7ca1c20:   0x0    0x0
0x7ca1c30:   0x0    0x0
0x7ca1c40:   0x0    0x0
0x7ca1c50:   0x0    0x0
0x7ca1c60:   0x0    0x0
0x7ca1c70:   0x0    0x0
0x7ca1c80:   0x0    0x0
0x7ca1c90:   0x0    0x0
0x7ca1ca0:   0x0    0x0
0x7ca1cb0:   0x0    0x0
0x7ca1cc0:   0x0    0x0
0x7ca1cd0:   0x0    0x111
0x7ca1ce0:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca1cf0:   0x0    0x0
0x7ca1d00:   0x4004f0 <procD>    0x0
0x7ca1d10:   0x0    0x0
0x7ca1d20:   0x0    0x0
0x7ca1d30:   0x0    0x0
0x7ca1d40:   0x0    0x0
0x7ca1d50:   0x0    0x0
0x7ca1d60:   0x0    0x0
0x7ca1d70:   0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7ca1d80:   0x0    0x0
0x7ca1d90:   0x0    0x0
0x7ca1da0:   0x0    0x0
0x7ca1db0:   0x0    0x0
0x7ca1dc0:   0x0    0x0
0x7ca1dd0:   0x0    0x0
0x7ca1de0:   0x0    0x111
0x7ca1df0:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca1e00:   0x0    0x0
0x7ca1e10:   0x4004f0 <procD>    0x0
0x7ca1e20:   0x0    0x0
0x7ca1e30:   0x0    0x0
0x7ca1e40:   0x0    0x0
0x7ca1e50:   0x0    0x0
0x7ca1e60:   0x0    0x0
0x7ca1e70:   0x0    0x0
0x7ca1e80:   0x0    0x0
0x7ca1e90:   0x0    0x0
0x7ca1ea0:   0x0    0x0
0x7ca1eb0:   0x0    0x0
0x7ca1ec0:   0x0    0x0
0x7ca1ed0:   0x0    0x0
0x7ca1ee0:   0x0    0x0
0x7ca1ef0:   0x0    0x111
---Type <return> to continue, or q <return> to quit---
0x7ca1f00:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca1f10:   0x0    0x0
0x7ca1f20:   0x4004f0 <procD>    0x0
0x7ca1f30:   0x0    0x0
0x7ca1f40:   0x0    0x0
0x7ca1f50:   0x0    0x0
0x7ca1f60:   0x0    0x0
0x7ca1f70:   0x0    0x0
0x7ca1f80:   0x0    0x0
```

```
0x7ca1f90:   0x0    0x0
0x7ca1fa0:   0x0    0x0
0x7ca1fb0:   0x0    0x0
0x7ca1fc0:   0x0    0x0
0x7ca1fd0:   0x0    0x0
0x7ca1fe0:   0x0    0x0
0x7ca1ff0:   0x0    0x0
0x7ca2000:   0x0    0x111
0x7ca2010:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca2020:   0x0    0x0
0x7ca2030:   0x4004f0 <procD>    0x0
0x7ca2040:   0x0    0x0
0x7ca2050:   0x0    0x0
0x7ca2060:   0x0    0x0
0x7ca2070:   0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7ca2080:   0x0    0x0
0x7ca2090:   0x0    0x0
0x7ca20a0:   0x0    0x0
0x7ca20b0:   0x0    0x0
0x7ca20c0:   0x0    0x0
0x7ca20d0:   0x0    0x0
0x7ca20e0:   0x0    0x0
0x7ca20f0:   0x0    0x0
0x7ca2100:   0x0    0x0
0x7ca2110:   0x0    0x111
0x7ca2120:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca2130:   0x0    0x0
0x7ca2140:   0x4004f0 <procD>    0x0
0x7ca2150:   0x0    0x0
0x7ca2160:   0x0    0x0
0x7ca2170:   0x0    0x0
0x7ca2180:   0x0    0x0
0x7ca2190:   0x0    0x0
0x7ca21a0:   0x0    0x0
0x7ca21b0:   0x0    0x0
0x7ca21c0:   0x0    0x0
0x7ca21d0:   0x0    0x0
0x7ca21e0:   0x0    0x0
0x7ca21f0:   0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7ca2200:   0x0    0x0
0x7ca2210:   0x0    0x0
0x7ca2220:   0x0    0x111
0x7ca2230:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca2240:   0x0    0x0
0x7ca2250:   0x4004f0 <procD>    0x0
0x7ca2260:   0x0    0x0
0x7ca2270:   0x0    0x0
0x7ca2280:   0x0    0x0
0x7ca2290:   0x0    0x0
0x7ca22a0:   0x0    0x0
0x7ca22b0:   0x0    0x0
0x7ca22c0:   0x0    0x0
0x7ca22d0:   0x0    0x0
0x7ca22e0:   0x0    0x0
0x7ca22f0:   0x0    0x0
0x7ca2300:   0x0    0x0
0x7ca2310:   0x0    0x0
0x7ca2320:   0x0    0x0
```

```
0x7ca2330:   0x0    0x111
0x7ca2340:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca2350:   0x0    0x0
0x7ca2360:   0x4004f0 <procD>    0x0
0x7ca2370:   0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7ca2380:   0x0    0x0
0x7ca2390:   0x0    0x0
0x7ca23a0:   0x0    0x0
0x7ca23b0:   0x0    0x0
0x7ca23c0:   0x0    0x0
0x7ca23d0:   0x0    0x0
0x7ca23e0:   0x0    0x0
0x7ca23f0:   0x0    0x0
0x7ca2400:   0x0    0x0
0x7ca2410:   0x0    0x0
0x7ca2420:   0x0    0x0
0x7ca2430:   0x0    0x0
0x7ca2440:   0x0    0x111
0x7ca2450:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca2460:   0x0    0x0
0x7ca2470:   0x4004f0 <procD>    0x0
0x7ca2480:   0x0    0x0
0x7ca2490:   0x0    0x0
0x7ca24a0:   0x0    0x0
0x7ca24b0:   0x0    0x0
0x7ca24c0:   0x0    0x0
0x7ca24d0:   0x0    0x0
0x7ca24e0:   0x0    0x0
0x7ca24f0:   0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7ca2500:   0x0    0x0
0x7ca2510:   0x0    0x0
0x7ca2520:   0x0    0x0
0x7ca2530:   0x0    0x0
0x7ca2540:   0x0    0x0
0x7ca2550:   0x0    0x111
0x7ca2560:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca2570:   0x0    0x0
0x7ca2580:   0x4004f0 <procD>    0x0
0x7ca2590:   0x0    0x0
0x7ca25a0:   0x0    0x0
0x7ca25b0:   0x0    0x0
0x7ca25c0:   0x0    0x0
0x7ca25d0:   0x0    0x0
0x7ca25e0:   0x0    0x0
0x7ca25f0:   0x0    0x0
0x7ca2600:   0x0    0x0
0x7ca2610:   0x0    0x0
0x7ca2620:   0x0    0x0
0x7ca2630:   0x0    0x0
0x7ca2640:   0x0    0x0
0x7ca2650:   0x0    0x0
0x7ca2660:   0x0    0x111
0x7ca2670:   0x657461636f6c6c61  0x79726f6d656d2064
---Type <return> to continue, or q <return> to quit---
0x7ca2680:   0x0    0x0
0x7ca2690:   0x4004f0 <procD>    0x0
0x7ca26a0:   0x0    0x0
0x7ca26b0:   0x0    0x0
```

```
0x7ca26c0:   0x0    0x0
0x7ca26d0:   0x0    0x0
0x7ca26e0:   0x0    0x0
0x7ca26f0:   0x0    0x0
0x7ca2700:   0x0    0x0
0x7ca2710:   0x0    0x0
0x7ca2720:   0x0    0x0
0x7ca2730:   0x0    0x0
0x7ca2740:   0x0    0x0
0x7ca2750:   0x0    0x0
0x7ca2760:   0x0    0x0
0x7ca2770:   0x0    0x111
0x7ca2780:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca2790:   0x0    0x0
0x7ca27a0:   0x4004f0 <procD>    0x0
0x7ca27b0:   0x0    0x0
0x7ca27c0:   0x0    0x0
0x7ca27d0:   0x0    0x0
0x7ca27e0:   0x0    0x0
0x7ca27f0:   0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7ca2800:   0x0    0x0
0x7ca2810:   0x0    0x0
0x7ca2820:   0x0    0x0
0x7ca2830:   0x0    0x0
0x7ca2840:   0x0    0x0
0x7ca2850:   0x0    0x0
0x7ca2860:   0x0    0x0
0x7ca2870:   0x0    0x0
0x7ca2880:   0x0    0x111
0x7ca2890:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca28a0:   0x0    0x0
0x7ca28b0:   0x4004f0 <procD>    0x0
0x7ca28c0:   0x0    0x0
0x7ca28d0:   0x0    0x0
0x7ca28e0:   0x0    0x0
0x7ca28f0:   0x0    0x0
0x7ca2900:   0x0    0x0
0x7ca2910:   0x0    0x0
0x7ca2920:   0x0    0x0
0x7ca2930:   0x0    0x0
0x7ca2940:   0x0    0x0
0x7ca2950:   0x0    0x0
0x7ca2960:   0x0    0x0
0x7ca2970:   0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7ca2980:   0x0    0x0
0x7ca2990:   0x0    0x111
0x7ca29a0:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca29b0:   0x0    0x0
0x7ca29c0:   0x4004f0 <procD>    0x0
0x7ca29d0:   0x0    0x0
0x7ca29e0:   0x0    0x0
0x7ca29f0:   0x0    0x0
0x7ca2a00:   0x0    0x0
0x7ca2a10:   0x0    0x0
0x7ca2a20:   0x0    0x0
0x7ca2a30:   0x0    0x0
0x7ca2a40:   0x0    0x0
0x7ca2a50:   0x0    0x0
```

```
0x7ca2a60:   0x0    0x0
0x7ca2a70:   0x0    0x0
0x7ca2a80:   0x0    0x0
0x7ca2a90:   0x0    0x0
0x7ca2aa0:   0x0    0x111
0x7ca2ab0:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca2ac0:   0x0    0x0
0x7ca2ad0:   0x4004f0 <procD>    0x0
0x7ca2ae0:   0x0    0x0
0x7ca2af0:   0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7ca2b00:   0x0    0x0
0x7ca2b10:   0x0    0x0
0x7ca2b20:   0x0    0x0
0x7ca2b30:   0x0    0x0
0x7ca2b40:   0x0    0x0
0x7ca2b50:   0x0    0x0
0x7ca2b60:   0x0    0x0
0x7ca2b70:   0x0    0x0
0x7ca2b80:   0x0    0x0
0x7ca2b90:   0x0    0x0
0x7ca2ba0:   0x0    0x0
0x7ca2bb0:   0x0    0x111
0x7ca2bc0:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca2bd0:   0x0    0x0
0x7ca2be0:   0x4004f0 <procD>    0x0
0x7ca2bf0:   0x0    0x0
0x7ca2c00:   0x0    0x0
0x7ca2c10:   0x0    0x0
0x7ca2c20:   0x0    0x0
0x7ca2c30:   0x0    0x0
0x7ca2c40:   0x0    0x0
0x7ca2c50:   0x0    0x0
0x7ca2c60:   0x0    0x0
0x7ca2c70:   0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7ca2c80:   0x0    0x0
0x7ca2c90:   0x0    0x0
0x7ca2ca0:   0x0    0x0
0x7ca2cb0:   0x0    0x0
0x7ca2cc0:   0x0    0x111
0x7ca2cd0:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca2ce0:   0x0    0x0
0x7ca2cf0:   0x4004f0 <procD>    0x0
0x7ca2d00:   0x0    0x0
0x7ca2d10:   0x0    0x0
0x7ca2d20:   0x0    0x0
0x7ca2d30:   0x0    0x0
0x7ca2d40:   0x0    0x0
0x7ca2d50:   0x0    0x0
0x7ca2d60:   0x0    0x0
0x7ca2d70:   0x0    0x0
0x7ca2d80:   0x0    0x0
0x7ca2d90:   0x0    0x0
0x7ca2da0:   0x0    0x0
0x7ca2db0:   0x0    0x0
0x7ca2dc0:   0x0    0x0
0x7ca2dd0:   0x0    0x111
0x7ca2de0:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca2df0:   0x0    0x0
```

```
---Type <return> to continue, or q <return> to quit---
0x7ca2e00:   0x4004f0 <procD>    0x0
0x7ca2e10:   0x0    0x0
0x7ca2e20:   0x0    0x0
0x7ca2e30:   0x0    0x0
0x7ca2e40:   0x0    0x0
0x7ca2e50:   0x0    0x0
0x7ca2e60:   0x0    0x0
0x7ca2e70:   0x0    0x0
0x7ca2e80:   0x0    0x0
0x7ca2e90:   0x0    0x0
0x7ca2ea0:   0x0    0x0
0x7ca2eb0:   0x0    0x0
0x7ca2ec0:   0x0    0x0
0x7ca2ed0:   0x0    0x0
0x7ca2ee0:   0x0    0x111
0x7ca2ef0:   0x657461636f6c6c61  0x79726f6d656d2064
0x7ca2f00:   0x0    0x0
0x7ca2f10:   0x4004f0 <procD>    0x0
0x7ca2f20:   0x0    0x0
0x7ca2f30:   0x0    0x0

(gdb) x/s 0x7ca2ef0
0x7ca2ef0:    "allocated memory"
```

6. Compare pmap logs *pmap.16531.1*, *pmap.16531.2*, and *pmap.16531.3* (the first one was saved before the leak started and the other two correspond to core dumps we looked at):

```
16531:   ./App9
0000000000400000    732K r-x--  /home/training/ALCDA/App9/App9
00000000006b6000      8K rw---  /home/training/ALCDA/App9/App9
00000000006b8000     28K rw---    [ anon ]
0000000000986000   1460K rw---    [ anon ]
00007f5eca49a000      4K -----    [ anon ]
00007f5eca49b000   8192K rw---    [ anon ]
00007f5ecac9b000      4K -----    [ anon ]
00007f5ecac9c000   8192K rw---    [ anon ]
00007f5ecb49c000      4K -----    [ anon ]
00007f5ecb49d000   8192K rw---    [ anon ]
00007f5ecbc9d000      4K -----    [ anon ]
00007f5ecbc9e000   8192K rw---    [ anon ]
00007f5ecc49e000      4K -----    [ anon ]
00007f5ecc49f000   8192K rw---    [ anon ]
00007fffe9d7b000    132K rw---    [ stack ]
00007fffe9d9d000      4K r-x--    [ anon ]
ffffffffff600000      4K r-x--    [ anon ]
 total            43348K
```

```
16531:   ./App9
0000000000400000    732K r-x--  /home/training/ALCDA/App9/App9
00000000006b6000      8K rw---  /home/training/ALCDA/App9/App9
00000000006b8000     28K rw---    [ anon ]
0000000000986000  67856K rw---    [ anon ]
00007f5eca49a000      4K -----    [ anon ]
00007f5eca49b000   8192K rw---    [ anon ]
00007f5ecac9b000      4K -----    [ anon ]
00007f5ecac9c000   8192K rw---    [ anon ]
00007f5ecb49c000      4K -----    [ anon ]
00007f5ecb49d000   8192K rw---    [ anon ]
00007f5ecbc9d000      4K -----    [ anon ]
00007f5ecbc9e000   8192K rw---    [ anon ]
00007f5ecc49e000      4K -----    [ anon ]
00007f5ecc49f000   8192K rw---    [ anon ]
00007fffe9d7b000    132K rw---    [ stack ]
00007fffe9d9d000      4K r-x--    [ anon ]
ffffffffff600000      4K r-x--    [ anon ]
 total           109744K


16531:   ./App9
0000000000400000    732K r-x--  /home/training/ALCDA/App9/App9
00000000006b6000      8K rw---  /home/training/ALCDA/App9/App9
00000000006b8000     28K rw---    [ anon ]
0000000000986000 134252K rw---    [ anon ]
00007f5eca49a000      4K -----    [ anon ]
00007f5eca49b000   8192K rw---    [ anon ]
00007f5ecac9b000      4K -----    [ anon ]
00007f5ecac9c000   8192K rw---    [ anon ]
00007f5ecb49c000      4K -----    [ anon ]
00007f5ecb49d000   8192K rw---    [ anon ]
00007f5ecbc9d000      4K -----    [ anon ]
00007f5ecbc9e000   8192K rw---    [ anon ]
00007f5ecc49e000      4K -----    [ anon ]
00007f5ecc49f000   8192K rw---    [ anon ]
00007fffe9d7b000    132K rw---    [ stack ]
00007fffe9d9d000      4K r-x--    [ anon ]
ffffffffff600000      4K r-x--    [ anon ]
 total           176140K
```

Exercise A10

- **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
- \ALCDA-Dumps\Exercise-A10.pdf

© 2015 Software Diagnostics Services

Exercise A10

Goal: Learn how to identify heap contention wait chains, synchronization issues, advanced disassembly, dump arrays.

Patterns: Heap Corruption, Heap Contention, Wait Chain, Critical Region, Self-Diagnostics.

1. When we launched App10 we got this console output and a core dump was saved:

```
training@debian64:~/ALCDA/App10$ ./App10
*** glibc detected *** ./App10: double free or corruption (!prev): 0x0000000001b681a0 ***
======= Backtrace: =========
[0x412042]
[0x416c27]
[0x400586]
[0x40067e]
[0x40068e]
[0x4006a6]
[0x4016c0]
[0x432589]
======= Memory map: ========
00400000-004b8000 r-xp 00000000 08:01 28961                              /home/training/ALCDA/App10/App10
006b8000-006b9000 rw-p 000b8000 08:01 28961                              /home/training/ALCDA/App10/App10
006b9000-006d4000 rw-p 00000000 00:00 0
01ab1000-029cc000 rw-p 00000000 00:00 0                                  [heap]
7f2654000000-7f2654021000 rw-p 00000000 00:00 0
7f2654021000-7f2658000000 ---p 00000000 00:00 0
7f265c000000-7f265cf3f000 rw-p 00000000 00:00 0
7f265cf3f000-7f2660000000 ---p 00000000 00:00 0
7f2663374000-7f2663375000 ---p 00000000 00:00 0
7f2663375000-7f2663b75000 rw-p 00000000 00:00 0
7f2663b75000-7f2663b76000 ---p 00000000 00:00 0
7f2663b76000-7f2664376000 rw-p 00000000 00:00 0
7f2664376000-7f2664377000 ---p 00000000 00:00 0
7f2664377000-7f2664b77000 rw-p 00000000 00:00 0
7f2664b77000-7f2664b78000 ---p 00000000 00:00 0
7f2664b78000-7f2665378000 rw-p 00000000 00:00 0
7f2665378000-7f2665379000 ---p 00000000 00:00 0
7f2665379000-7f2665b79000 rw-p 00000000 00:00 0
7ffd51920000-7ffd51941000 rw-p 00000000 00:00 0                          [stack]
7ffd519ce000-7ffd519cf000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
Aborted (core dumped)
```

2. Load a core dump and *App10* executable:

```
training@debian64:~/ALCDA$ gdb -c ./App10/core -se ./App10/App10
GNU gdb (GDB) 7.4.1-debian
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/training/ALCDA/App10/App10...done.
[New LWP 17002]
[New LWP 17003]
[New LWP 16998]
[New LWP 16999]
[New LWP 17001]
[New LWP 17000]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Core was generated by `./App10'.
Program terminated with signal 6, Aborted.
#0  0x000000000043ef65 in raise ()
```

3. Check all threads and identify problem top frames:

```
(gdb) info threads
  Id   Target Id         Frame
  6    Thread 0x7f2665377700 (LWP 17000) 0x00000000004151a1 in _int_malloc ()
  5    Thread 0x7f2664b76700 (LWP 17001) __lll_unlock_wake_private ()
    at ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S:343
  4    Thread 0x7f2665b78700 (LWP 16999) __lll_lock_wait_private ()
    at ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S:97
  3    Thread 0x1ab1860 (LWP 16998) 0x000000000042fed1 in nanosleep ()
  2    Thread 0x7f2663b74700 (LWP 17003) __lll_lock_wait_private ()
    at ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S:97
* 1    Thread 0x7f2664375700 (LWP 17002) 0x000000000043ef65 in raise ()
```

4. Check thread 2 and find where it was being executed:

```
(gdb) thread 2
[Switching to thread 2 (Thread 0x7f2663b74700 (LWP 17003))]
#0  __lll_lock_wait_private ()
    at ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S:97
97     ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S: No such file or directory.

(gdb) bt
#0  __lll_lock_wait_private ()
    at ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S:97
#1  0x0000000000418836 in _L_lock_9558 ()
#2  0x000000000041 6c1c in free ()
#3  0x0000000000400586 in proc ()
#4  0x00000000004006bb in bar_five ()
#5  0x00000000004006cb in foo_five ()
#6  0x00000000004006e3 in thread_five ()
#7  0x00000000004016c0 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#8  0x0000000000432589 in clone ()
#9  0x0000000000000000 in ?? ()
```

```
(gdb) disassemble proc
Dump of assembler code for function proc:
   0x00000000004004f0 <+0>:     push   %rbp
   0x00000000004004f1 <+1>:     mov    %rsp,%rbp
   0x00000000004004f4 <+4>:     push   %rbx
   0x00000000004004f5 <+5>:     sub    $0x18,%rsp
   0x00000000004004f9 <+9>:     callq  0x40ac70 <rand>
   0x00000000004004fe <+14>:    mov    %eax,%ecx
   0x0000000000400500 <+16>:    mov    $0x68db8bad,%edx
   0x0000000000400505 <+21>:    mov    %ecx,%eax
   0x0000000000400507 <+23>:    imul   %edx
   0x0000000000400509 <+25>:    sar    $0xc,%edx
   0x000000000040050c <+28>:    mov    %ecx,%eax
   0x000000000040050e <+30>:    sar    $0x1f,%eax
   0x0000000000400511 <+33>:    mov    %edx,%ebx
   0x0000000000400513 <+35>:    sub    %eax,%ebx
   0x0000000000400515 <+37>:    mov    %ebx,%eax
   0x0000000000400517 <+39>:    mov    %eax,-0x14(%rbp)
   0x000000000040051a <+42>:    mov    -0x14(%rbp),%eax
   0x000000000040051d <+45>:    imul   $0x2710,%eax,%eax
   0x0000000000400523 <+51>:    mov    %ecx,%edx
   0x0000000000400525 <+53>:    sub    %eax,%edx
   0x0000000000400527 <+55>:    mov    %edx,%eax
   0x0000000000400529 <+57>:    mov    %eax,-0x14(%rbp)
   0x000000000040052c <+60>:    callq  0x40ac70 <rand>
---Type <return> to continue, or q <return> to quit---
   0x0000000000400531 <+65>:    mov    %eax,%ecx
   0x0000000000400533 <+67>:    mov    $0x68db8bad,%edx
   0x0000000000400538 <+72>:    mov    %ecx,%eax
   0x000000000040053a <+74>:    imul   %edx
   0x000000000040053c <+76>:    sar    $0xc,%edx
   0x000000000040053f <+79>:    mov    %ecx,%eax
   0x0000000000400541 <+81>:    sar    $0x1f,%eax
   0x0000000000400544 <+84>:    mov    %edx,%ebx
   0x0000000000400546 <+86>:    sub    %eax,%ebx
   0x0000000000400548 <+88>:    mov    %ebx,%eax
   0x000000000040054a <+90>:    mov    %eax,-0x18(%rbp)
   0x000000000040054d <+93>:    mov    -0x18(%rbp),%eax
   0x0000000000400550 <+96>:    imul   $0x2710,%eax,%eax
   0x0000000000400556 <+102>:   mov    %ecx,%edx
   0x0000000000400558 <+104>:   sub    %eax,%edx
   0x000000000040055a <+106>:   mov    %edx,%eax
   0x000000000040055c <+108>:   mov    %eax,-0x18(%rbp)
   0x000000000040055f <+111>:   mov    -0x14(%rbp),%eax
   0x0000000000400562 <+114>:   cltq
   0x0000000000400564 <+116>:   mov    0x6b8fc0(,%rax,8),%rax
   0x000000000040056c <+124>:   test   %rax,%rax
   0x000000000040056f <+127>:   je     0x400597 <proc+167>
   0x0000000000400571 <+129>:   mov    -0x14(%rbp),%eax
   0x0000000000400574 <+132>:   cltq
---Type <return> to continue, or q <return> to quit---
   0x0000000000400576 <+134>:   mov    0x6b8fc0(,%rax,8),%rax
   0x000000000040057e <+142>:   mov    %rax,%rdi
   0x0000000000400581 <+145>:   callq  0x416bc0 <free>
   0x0000000000400586 <+150>:   mov    -0x14(%rbp),%eax
   0x0000000000400589 <+153>:   cltq
   0x000000000040058b <+155>:   movq   $0x0,0x6b8fc0(,%rax,8)
   0x0000000000400597 <+167>:   mov    -0x18(%rbp),%eax
   0x000000000040059a <+170>:   cltq
```

```
   0x000000000040059c <+172>:    mov    %rax,%rdi
   0x000000000040059f <+175>:    callq  0x416c90 <malloc>
   0x00000000004005a4 <+180>:    mov    %rax,%rdx
   0x00000000004005a7 <+183>:    mov    -0x14(%rbp),%eax
   0x00000000004005aa <+186>:    cltq
   0x00000000004005ac <+188>:    mov    %rdx,0x6b8fc0(,%rax,8)
   0x00000000004005b4 <+196>:    jmpq   0x4004f9 <proc+9>
End of assembler dump.
```

5. Check the thread #4 and find where it was being executed:

```
(gdb) thread 4
[Switching to thread 4 (Thread 0x7f2665b78700 (LWP 16999))]
#0  __lll_lock_wait_private ()
    at ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S:97
97    in ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S

(gdb) bt
#0  __lll_lock_wait_private ()
    at ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S:97
#1  0x0000000000418836 in _L_lock_9558 ()
#2  0x0000000000416c1c in free ()
#3  0x0000000000400586 in proc ()
#4  0x00000000004005c7 in bar_one ()
#5  0x00000000004005d7 in foo_one ()
#6  0x00000000004005ef in thread_one ()
#7  0x00000000004016c0 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#8  0x0000000000432589 in clone ()
#9  0x0000000000000000 in ?? ()

(gdb) disassemble proc
Dump of assembler code for function proc:
   0x00000000004004f0 <+0>:      push   %rbp
   0x00000000004004f1 <+1>:      mov    %rsp,%rbp
   0x00000000004004f4 <+4>:      push   %rbx
   0x00000000004004f5 <+5>:      sub    $0x18,%rsp
   0x00000000004004f9 <+9>:      callq  0x40ac70 <rand>
   0x00000000004004fe <+14>:     mov    %eax,%ecx
   0x0000000000400500 <+16>:     mov    $0x68db8bad,%edx
   0x0000000000400505 <+21>:     mov    %ecx,%eax
   0x0000000000400507 <+23>:     imul   %edx
   0x0000000000400509 <+25>:     sar    $0xc,%edx
   0x000000000040050c <+28>:     mov    %ecx,%eax
   0x000000000040050e <+30>:     sar    $0x1f,%eax
   0x0000000000400511 <+33>:     mov    %edx,%ebx
   0x0000000000400513 <+35>:     sub    %eax,%ebx
   0x0000000000400515 <+37>:     mov    %ebx,%eax
   0x0000000000400517 <+39>:     mov    %eax,-0x14(%rbp)
   0x000000000040051a <+42>:     mov    -0x14(%rbp),%eax
   0x000000000040051d <+45>:     imul   $0x2710,%eax,%eax
   0x0000000000400523 <+51>:     mov    %ecx,%edx
   0x0000000000400525 <+53>:     sub    %eax,%edx
   0x0000000000400527 <+55>:     mov    %edx,%eax
   0x0000000000400529 <+57>:     mov    %eax,-0x14(%rbp)
   0x000000000040052c <+60>:     callq  0x40ac70 <rand>
---Type <return> to continue, or q <return> to quit---
   0x0000000000400531 <+65>:     mov    %eax,%ecx
   0x0000000000400533 <+67>:     mov    $0x68db8bad,%edx
   0x0000000000400538 <+72>:     mov    %ecx,%eax
```

```
   0x000000000040053a <+74>:    imul   %edx
   0x000000000040053c <+76>:    sar    $0xc,%edx
   0x000000000040053f <+79>:    mov    %ecx,%eax
   0x0000000000400541 <+81>:    sar    $0x1f,%eax
   0x0000000000400544 <+84>:    mov    %edx,%ebx
   0x0000000000400546 <+86>:    sub    %eax,%ebx
   0x0000000000400548 <+88>:    mov    %ebx,%eax
   0x000000000040054a <+90>:    mov    %eax,-0x18(%rbp)
   0x000000000040054d <+93>:    mov    -0x18(%rbp),%eax
   0x0000000000400550 <+96>:    imul   $0x2710,%eax,%eax
   0x0000000000400556 <+102>:   mov    %ecx,%edx
   0x0000000000400558 <+104>:   sub    %eax,%edx
   0x000000000040055a <+106>:   mov    %edx,%eax
   0x000000000040055c <+108>:   mov    %eax,-0x18(%rbp)
   0x000000000040055f <+111>:   mov    -0x14(%rbp),%eax
   0x0000000000400562 <+114>:   cltq
   0x0000000000400564 <+116>:   mov    0x6b8fc0(,%rax,8),%rax
   0x000000000040056c <+124>:   test   %rax,%rax
   0x000000000040056f <+127>:   je     0x400597 <proc+167>
   0x0000000000400571 <+129>:   mov    -0x14(%rbp),%eax
   0x0000000000400574 <+132>:   cltq
---Type <return> to continue, or q <return> to quit---
   0x0000000000400576 <+134>:   mov    0x6b8fc0(,%rax,8),%rax
   0x000000000040057e <+142>:   mov    %rax,%rdi
   0x0000000000400581 <+145>:   callq  0x416bc0 <free>
   0x0000000000400586 <+150>:   mov    -0x14(%rbp),%eax
   0x0000000000400589 <+153>:   cltq
   0x000000000040058b <+155>:   movq   $0x0,0x6b8fc0(,%rax,8)
   0x0000000000400597 <+167>:   mov    -0x18(%rbp),%eax
   0x000000000040059a <+170>:   cltq
   0x000000000040059c <+172>:   mov    %rax,%rdi
   0x000000000040059f <+175>:   callq  0x416c90 <malloc>
   0x00000000004005a4 <+180>:   mov    %rax,%rdx
   0x00000000004005a7 <+183>:   mov    -0x14(%rbp),%eax
   0x00000000004005aa <+186>:   cltq
   0x00000000004005ac <+188>:   mov    %rdx,0x6b8fc0(,%rax,8)
   0x00000000004005b4 <+196>:   jmpq   0x4004f9 <proc+9>
End of assembler dump.
```

6. Check the thread #5 and find where it was being executed:

```
(gdb) thread 5
[Switching to thread 5 (Thread 0x7f2664b76700 (LWP 17001))]
#0  __lll_unlock_wake_private ()
    at ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S:343
343    in ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S

(gdb) bt
#0  __lll_unlock_wake_private ()
    at ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S:343
#1  0x000000000041886d in _L_unlock_9670 ()
#2  0x0000000000416d22 in malloc ()
#3  0x00000000004005a4 in proc ()
#4  0x0000000000400641 in bar_three ()
#5  0x0000000000400651 in foo_three ()
#6  0x0000000000400669 in thread_three ()
#7  0x00000000004016c0 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#8  0x0000000000432589 in clone ()
#9  0x0000000000000000 in ?? ()
```

```
(gdb) disassemble proc
Dump of assembler code for function proc:
   0x00000000004004f0 <+0>:     push   %rbp
   0x00000000004004f1 <+1>:     mov    %rsp,%rbp
   0x00000000004004f4 <+4>:     push   %rbx
   0x00000000004004f5 <+5>:     sub    $0x18,%rsp
   0x00000000004004f9 <+9>:     callq  0x40ac70 <rand>
   0x00000000004004fe <+14>:    mov    %eax,%ecx
   0x0000000000400500 <+16>:    mov    $0x68db8bad,%edx
   0x0000000000400505 <+21>:    mov    %ecx,%eax
   0x0000000000400507 <+23>:    imul   %edx
   0x0000000000400509 <+25>:    sar    $0xc,%edx
   0x000000000040050c <+28>:    mov    %ecx,%eax
   0x000000000040050e <+30>:    sar    $0x1f,%eax
   0x0000000000400511 <+33>:    mov    %edx,%ebx
   0x0000000000400513 <+35>:    sub    %eax,%ebx
   0x0000000000400515 <+37>:    mov    %ebx,%eax
   0x0000000000400517 <+39>:    mov    %eax,-0x14(%rbp)
   0x000000000040051a <+42>:    mov    -0x14(%rbp),%eax
   0x000000000040051d <+45>:    imul   $0x2710,%eax,%eax
   0x0000000000400523 <+51>:    mov    %ecx,%edx
   0x0000000000400525 <+53>:    sub    %eax,%edx
   0x0000000000400527 <+55>:    mov    %edx,%eax
   0x0000000000400529 <+57>:    mov    %eax,-0x14(%rbp)
   0x000000000040052c <+60>:    callq  0x40ac70 <rand>
---Type <return> to continue, or q <return> to quit---
   0x0000000000400531 <+65>:    mov    %eax,%ecx
   0x0000000000400533 <+67>:    mov    $0x68db8bad,%edx
   0x0000000000400538 <+72>:    mov    %ecx,%eax
   0x000000000040053a <+74>:    imul   %edx
   0x000000000040053c <+76>:    sar    $0xc,%edx
   0x000000000040053f <+79>:    mov    %ecx,%eax
   0x0000000000400541 <+81>:    sar    $0x1f,%eax
   0x0000000000400544 <+84>:    mov    %edx,%ebx
   0x0000000000400546 <+86>:    sub    %eax,%ebx
   0x0000000000400548 <+88>:    mov    %ebx,%eax
   0x000000000040054a <+90>:    mov    %eax,-0x18(%rbp)
   0x000000000040054d <+93>:    mov    -0x18(%rbp),%eax
   0x0000000000400550 <+96>:    imul   $0x2710,%eax,%eax
   0x0000000000400556 <+102>:   mov    %ecx,%edx
   0x0000000000400558 <+104>:   sub    %eax,%edx
   0x000000000040055a <+106>:   mov    %edx,%eax
   0x000000000040055c <+108>:   mov    %eax,-0x18(%rbp)
   0x000000000040055f <+111>:   mov    -0x14(%rbp),%eax
   0x0000000000400562 <+114>:   cltq
   0x0000000000400564 <+116>:   mov    0x6b8fc0(,%rax,8),%rax
   0x000000000040056c <+124>:   test   %rax,%rax
   0x000000000040056f <+127>:   je     0x400597 <proc+167>
   0x0000000000400571 <+129>:   mov    -0x14(%rbp),%eax
   0x0000000000400574 <+132>:   cltq
---Type <return> to continue, or q <return> to quit---
   0x0000000000400576 <+134>:   mov    0x6b8fc0(,%rax,8),%rax
   0x000000000040057e <+142>:   mov    %rax,%rdi
   0x0000000000400581 <+145>:   callq  0x416bc0 <free>
   0x0000000000400586 <+150>:   mov    -0x14(%rbp),%eax
   0x0000000000400589 <+153>:   cltq
   0x000000000040058b <+155>:   movq   $0x0,0x6b8fc0(,%rax,8)
   0x0000000000400597 <+167>:   mov    -0x18(%rbp),%eax
   0x000000000040059a <+170>:   cltq
```

```
   0x000000000040059c <+172>:    mov    %rax,%rdi
   0x000000000040059f <+175>:    callq  0x416c90 <malloc>
   0x00000000004005a4 <+180>:    mov    %rax,%rdx
   0x00000000004005a7 <+183>:    mov    -0x14(%rbp),%eax
   0x00000000004005aa <+186>:    cltq
   0x00000000004005ac <+188>:    mov    %rdx,0x6b8fc0(,%rax,8)
   0x00000000004005b4 <+196>:    jmpq   0x4004f9 <proc+9>
End of assembler dump.
```

We see some buffer 0x6b8fc0 “sandwiched” between *free* and *malloc* calls that internally call “lock” and “unlock” functions.

7. Check the thread #6 and find where it was being executed:

```
(gdb) thread 6
[Switching to thread 6 (Thread 0x7f2665377700 (LWP 17000))]
#0  0x00000000004151a1 in _int_malloc ()

(gdb) bt
#0  0x00000000004151a1 in _int_malloc ()
#1  0x0000000000416cf8 in malloc ()
#2  0x00000000004005a4 in proc ()
#3  0x0000000000400604 in bar_two ()
#4  0x0000000000400614 in foo_two ()
#5  0x000000000040062c in thread_two ()
#6  0x00000000004016c0 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#7  0x0000000000432589 in clone ()
#8  0x0000000000000000 in ?? ()

(gdb) disassemble proc
Dump of assembler code for function proc:
   0x00000000004004f0 <+0>:      push   %rbp
   0x00000000004004f1 <+1>:      mov    %rsp,%rbp
   0x00000000004004f4 <+4>:      push   %rbx
   0x00000000004004f5 <+5>:      sub    $0x18,%rsp
   0x00000000004004f9 <+9>:      callq  0x40ac70 <rand>
   0x00000000004004fe <+14>:     mov    %eax,%ecx
   0x0000000000400500 <+16>:     mov    $0x68db8bad,%edx
   0x0000000000400505 <+21>:     mov    %ecx,%eax
   0x0000000000400507 <+23>:     imul   %edx
   0x0000000000400509 <+25>:     sar    $0xc,%edx
   0x000000000040050c <+28>:     mov    %ecx,%eax
   0x000000000040050e <+30>:     sar    $0x1f,%eax
   0x0000000000400511 <+33>:     mov    %edx,%ebx
   0x0000000000400513 <+35>:     sub    %eax,%ebx
   0x0000000000400515 <+37>:     mov    %ebx,%eax
   0x0000000000400517 <+39>:     mov    %eax,-0x14(%rbp)
   0x000000000040051a <+42>:     mov    -0x14(%rbp),%eax
   0x000000000040051d <+45>:     imul   $0x2710,%eax,%eax
   0x0000000000400523 <+51>:     mov    %ecx,%edx
   0x0000000000400525 <+53>:     sub    %eax,%edx
   0x0000000000400527 <+55>:     mov    %edx,%eax
   0x0000000000400529 <+57>:     mov    %eax,-0x14(%rbp)
   0x000000000040052c <+60>:     callq  0x40ac70 <rand>
---Type <return> to continue, or q <return> to quit---
   0x0000000000400531 <+65>:     mov    %eax,%ecx
   0x0000000000400533 <+67>:     mov    $0x68db8bad,%edx
   0x0000000000400538 <+72>:     mov    %ecx,%eax
   0x000000000040053a <+74>:     imul   %edx
```

```
   0x000000000040053c <+76>:    sar    $0xc,%edx
   0x000000000040053f <+79>:    mov    %ecx,%eax
   0x0000000000400541 <+81>:    sar    $0x1f,%eax
   0x0000000000400544 <+84>:    mov    %edx,%ebx
   0x0000000000400546 <+86>:    sub    %eax,%ebx
   0x0000000000400548 <+88>:    mov    %ebx,%eax
   0x000000000040054a <+90>:    mov    %eax,-0x18(%rbp)
   0x000000000040054d <+93>:    mov    -0x18(%rbp),%eax
   0x0000000000400550 <+96>:    imul   $0x2710,%eax,%eax
   0x0000000000400556 <+102>:   mov    %ecx,%edx
   0x0000000000400558 <+104>:   sub    %eax,%edx
   0x000000000040055a <+106>:   mov    %edx,%eax
   0x000000000040055c <+108>:   mov    %eax,-0x18(%rbp)
   0x000000000040055f <+111>:   mov    -0x14(%rbp),%eax
   0x0000000000400562 <+114>:   cltq
   0x0000000000400564 <+116>:   mov    0x6b8fc0(,%rax,8),%rax
   0x000000000040056c <+124>:   test   %rax,%rax
   0x000000000040056f <+127>:   je     0x400597 <proc+167>
   0x0000000000400571 <+129>:   mov    -0x14(%rbp),%eax
   0x0000000000400574 <+132>:   cltq
---Type <return> to continue, or q <return> to quit---
   0x0000000000400576 <+134>:   mov    0x6b8fc0(,%rax,8),%rax
   0x000000000040057e <+142>:   mov    %rax,%rdi
   0x0000000000400581 <+145>:   callq  0x416bc0 <free>
   0x0000000000400586 <+150>:   mov    -0x14(%rbp),%eax
   0x0000000000400589 <+153>:   cltq
   0x000000000040058b <+155>:   movq   $0x0,0x6b8fc0(,%rax,8)
   0x0000000000400597 <+167>:   mov    -0x18(%rbp),%eax
   0x000000000040059a <+170>:   cltq
   0x000000000040059c <+172>:   mov    %rax,%rdi
   0x000000000040059f <+175>:   callq  0x416c90 <malloc>
   0x00000000004005a4 <+180>:   mov    %rax,%rdx
   0x00000000004005a7 <+183>:   mov    -0x14(%rbp),%eax
   0x00000000004005aa <+186>:   cltq
   0x00000000004005ac <+188>:   mov    %rdx,0x6b8fc0(,%rax,8)
   0x00000000004005b4 <+196>:   jmpq   0x4004f9 <proc+9>
End of assembler dump.
```

8. Check the thread #1 and identify a diagnostic message:

```
(gdb) thread 1
[Switching to thread 1 (Thread 0x7f2664375700 (LWP 17002))]
#0  0x000000000043ef65 in raise ()

(gdb) bt
#0  0x000000000043ef65 in raise ()
#1  0x0000000000409fc0 in abort ()
#2  0x000000000040bf5b in __libc_message ()
#3  0x0000000000412042 in malloc_printerr ()
#4  0x0000000000416c27 in free ()
#5  0x0000000000400586 in proc ()
#6  0x000000000040067e in bar_four ()
#7  0x000000000040068e in foo_four ()
#8  0x00000000004006a6 in thread_four ()
#9  0x00000000004016c0 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#10 0x0000000000432589 in clone ()
#11 0x0000000000000000 in ?? ()
```

```
(gdb) disassemble __libc_message
Dump of assembler code for function __libc_message:
   0x0000000000040bc00 <+0>:     push   %rbp
   0x0000000000040bc01 <+1>:     movzbl %al,%eax
   0x0000000000040bc04 <+4>:     mov    %rsp,%rbp
   0x0000000000040bc07 <+7>:     push   %r15
   0x0000000000040bc09 <+9>:     push   %r14
   0x0000000000040bc0b <+11>:    push   %r13
   0x0000000000040bc0d <+13>:    push   %r12
   0x0000000000040bc0f <+15>:    push   %rbx
   0x0000000000040bc10 <+16>:    sub    $0x718,%rsp
   0x0000000000040bc17 <+23>:    mov    %rdx,-0xd0(%rbp)
   0x0000000000040bc1e <+30>:    lea    0x0(,%rax,4),%rdx
   0x0000000000040bc26 <+38>:    mov    $0x40bc74,%eax
   0x0000000000040bc2b <+43>:    mov    %edi,-0x72c(%rbp)
   0x0000000000040bc31 <+49>:    mov    %rcx,-0xc8(%rbp)
   0x0000000000040bc38 <+56>:    sub    %rdx,%rax
   0x0000000000040bc3b <+59>:    lea    -0x31(%rbp),%rdx
   0x0000000000040bc3f <+63>:    mov    %r8,-0xc0(%rbp)
   0x0000000000040bc46 <+70>:    mov    %r9,-0xb8(%rbp)
   0x0000000000040bc4d <+77>:    mov    $0x48d89b,%edi
   0x0000000000040bc52 <+82>:    jmpq   *%rax
   0x0000000000040bc54 <+84>:    movaps %xmm7,-0xf(%rdx)
   0x0000000000040bc58 <+88>:    movaps %xmm6,-0x1f(%rdx)
   0x0000000000040bc5c <+92>:    movaps %xmm5,-0x2f(%rdx)
---Type <return> to continue, or q <return> to quit---
   0x0000000000040bc60 <+96>:    movaps %xmm4,-0x3f(%rdx)
   0x0000000000040bc64 <+100>:   movaps %xmm3,-0x4f(%rdx)
   0x0000000000040bc68 <+104>:   movaps %xmm2,-0x5f(%rdx)
   0x0000000000040bc6c <+108>:   movaps %xmm1,-0x6f(%rdx)
   0x0000000000040bc70 <+112>:   movaps %xmm0,-0x7f(%rdx)
   0x0000000000040bc74 <+116>:   lea    0x10(%rbp),%rax
   0x0000000000040bc78 <+120>:   movl   $0x10,-0x100(%rbp)
   0x0000000000040bc82 <+130>:   movl   $0x30,-0xfc(%rbp)
   0x0000000000040bc8c <+140>:   mov    %rsi,-0x728(%rbp)
   0x0000000000040bc93 <+147>:   mov    %rax,-0xf8(%rbp)
   0x0000000000040bc9a <+154>:   lea    -0xe0(%rbp),%rax
   0x0000000000040bca1 <+161>:   mov    %rax,-0xf0(%rbp)
   0x0000000000040bca8 <+168>:   mov    -0x100(%rbp),%rax
   0x0000000000040bcaf <+175>:   mov    %rax,-0x120(%rbp)
   0x0000000000040bcb6 <+182>:   lea    0x10(%rbp),%rax
   0x0000000000040bcba <+186>:   mov    %rax,-0x118(%rbp)
   0x0000000000040bcc1 <+193>:   lea    -0xe0(%rbp),%rax
   0x0000000000040bcc8 <+200>:   mov    %rax,-0x110(%rbp)
   0x0000000000040bccf <+207>:   callq  0x43fbc0 <__secure_getenv>
   0x0000000000040bcd4 <+212>:   test   %rax,%rax
   0x0000000000040bcd7 <+215>:   je     0x40bce2 <__libc_message+226>
   0x0000000000040bcd9 <+217>:   cmpb   $0x0,(%rax)
   0x0000000000040bcdc <+220>:   jne    0x40beb0 <__libc_message+688>
   0x0000000000040bce2 <+226>:   xor    %eax,%eax
---Type <return> to continue, or q <return> to quit---
   0x0000000000040bce4 <+228>:   mov    $0x902,%esi
   0x0000000000040bce9 <+233>:   mov    $0x48d8ae,%edi
   0x0000000000040bcee <+238>:   callq  0x430639 <__open_nocancel>
   0x0000000000040bcf3 <+243>:   cmp    $0xffffffff,%eax
   0x0000000000040bcf6 <+246>:   mov    %eax,-0x730(%rbp)
   0x0000000000040bcfc <+252>:   je     0x40beb0 <__libc_message+688>
   0x0000000000040bd02 <+258>:   mov    -0x728(%rbp),%rax
   0x0000000000040bd09 <+265>:   movzbl (%rax),%r15d
   0x0000000000040bd0d <+269>:   test   %r15b,%r15b
```

```
   0x000000000040bd10 <+272>:    je     0x40bed6 <__libc_message+726>
   0x000000000040bd16 <+278>:    mov    -0x728(%rbp),%r12
   0x000000000040bd1d <+285>:    xor    %r14d,%r14d
   0x000000000040bd20 <+288>:    xor    %r13d,%r13d
   0x000000000040bd23 <+291>:    mov    %r12,%rax
   0x000000000040bd26 <+294>:    nopw   %cs:0x0(%rax,%rax,1)
   0x000000000040bd30 <+304>:    movzbl (%rax),%edx
   0x000000000040bd33 <+307>:    jmp    0x40bd49 <__libc_message+329>
   0x000000000040bd35 <+309>:    nopl   (%rax)
   0x000000000040bd38 <+312>:    mov    $0x25,%esi
   0x000000000040bd3d <+317>:    callq  0x424040 <strchrnul>
   0x000000000040bd42 <+322>:    movzbl (%rax),%edx
   0x000000000040bd45 <+325>:    test   %dl,%dl
   0x000000000040bd47 <+327>:    je     0x40bd60 <__libc_message+352>
   0x000000000040bd49 <+329>:    cmp    $0x25,%dl
---Type <return> to continue, or q <return> to quit---
   0x000000000040bd4c <+332>:    lea    0x1(%rax),%rdi
   0x000000000040bd50 <+336>:    jne    0x40bd38 <__libc_message+312>
   0x000000000040bd52 <+338>:    cmpb   $0x73,0x1(%rax)
   0x000000000040bd56 <+342>:    lea    0x1(%rax),%rdi
   0x000000000040bd5a <+346>:    jne    0x40bd38 <__libc_message+312>
   0x000000000040bd5c <+348>:    nopl   0x0(%rax)
   0x000000000040bd60 <+352>:    cmp    $0x25,%r15b
   0x000000000040bd64 <+356>:    je     0x40bda8 <__libc_message+424>
   0x000000000040bd66 <+358>:    mov    %rax,%rcx
   0x000000000040bd69 <+361>:    mov    %r12,%rbx
   0x000000000040bd6c <+364>:    sub    %r12,%rcx
   0x000000000040bd6f <+367>:    mov    %rax,%r12
   0x000000000040bd72 <+370>:    sub    $0x30,%rsp
   0x000000000040bd76 <+374>:    mov    %r12,%rax
   0x000000000040bd79 <+377>:    lea    0xf(%rsp),%rdx
   0x000000000040bd7e <+382>:    and    $0xfffffffffffffff0,%rdx
   0x000000000040bd82 <+386>:    mov    %rbx,(%rdx)
   0x000000000040bd85 <+389>:    mov    %r13,0x10(%rdx)
   0x000000000040bd89 <+393>:    lea    0x1(%r14),%ebx
   0x000000000040bd8d <+397>:    mov    %rcx,0x8(%rdx)
   0x000000000040bd91 <+401>:    movzbl (%r12),%r15d
   0x000000000040bd96 <+406>:    mov    %rdx,%r13
   0x000000000040bd99 <+409>:    test   %r15b,%r15b
   0x000000000040bd9c <+412>:    je     0x40bde8 <__libc_message+488>
---Type <return> to continue, or q <return> to quit---
   0x000000000040bd9e <+414>:    mov    %ebx,%r14d
   0x000000000040bda1 <+417>:    jmp    0x40bd30 <__libc_message+304>
   0x000000000040bda3 <+419>:    nopl   0x0(%rax,%rax,1)
   0x000000000040bda8 <+424>:    cmpb   $0x73,0x1(%r12)
   0x000000000040bdae <+430>:    jne    0x40bd66 <__libc_message+358>
   0x000000000040bdb0 <+432>:    mov    -0x100(%rbp),%edx
   0x000000000040bdb6 <+438>:    cmp    $0x2f,%edx
   0x000000000040bdb9 <+441>:    ja     0x40bebf <__libc_message+703>
   0x000000000040bdbf <+447>:    mov    %edx,%eax
   0x000000000040bdc1 <+449>:    add    -0xf0(%rbp),%rax
   0x000000000040bdc8 <+456>:    add    $0x8,%edx
   0x000000000040bdcb <+459>:    mov    %edx,-0x100(%rbp)
   0x000000000040bdd1 <+465>:    mov    (%rax),%rbx
   0x000000000040bdd4 <+468>:    add    $0x2,%r12
   0x000000000040bdd8 <+472>:    mov    %rbx,%rdi
   0x000000000040bddb <+475>:    callq  0x41bb90 <strlen>
   0x000000000040bde0 <+480>:    mov    %rax,%rcx
   0x000000000040bde3 <+483>:    jmp    0x40bd72 <__libc_message+370>
   0x000000000040bde5 <+485>:    nopl   (%rax)
```

```
   0x0000000000040bde8 <+488>:    movslq %ebx,%r9
   0x0000000000040bdeb <+491>:    xor    %ecx,%ecx
   0x0000000000040bded <+493>:    mov    %r9,%rax
   0x0000000000040bdf0 <+496>:    shl    $0x4,%rax
   0x0000000000040bdf4 <+500>:    add    $0x10,%rax
---Type <return> to continue, or q <return> to quit---
   0x0000000000040bdf8 <+504>:    sub    %rax,%rsp
   0x0000000000040bdfb <+507>:    movslq %r14d,%rax
   0x0000000000040bdfe <+510>:    mov    %r14d,%r14d
   0x0000000000040be01 <+513>:    lea    0xf(%rsp),%r8
   0x0000000000040be06 <+518>:    shl    $0x4,%rax
   0x0000000000040be0a <+522>:    shl    $0x4,%r14
   0x0000000000040be0e <+526>:    and    $0xfffffffffffffff0,%r8
   0x0000000000040be12 <+530>:    lea    (%r8,%rax,1),%rdx
   0x0000000000040be16 <+534>:    sub    $0x10,%rax
   0x0000000000040be1a <+538>:    mov    %r8,%r12
   0x0000000000040be1d <+541>:    sub    %r14,%rax
   0x0000000000040be20 <+544>:    lea    (%rax,%r8,1),%rdi
   0x0000000000040be24 <+548>:    mov    %r13,%rax
   0x0000000000040be27 <+551>:    nopw   0x0(%rax,%rax,1)
   0x0000000000040be30 <+560>:    mov    (%rax),%rsi
   0x0000000000040be33 <+563>:    mov    %rcx,%r13
   0x0000000000040be36 <+566>:    mov    %rsi,(%rdx)
   0x0000000000040be39 <+569>:    mov    0x8(%rax),%rsi
   0x0000000000040be3d <+573>:    mov    %rsi,0x8(%rdx)
   0x0000000000040be41 <+577>:    add    0x8(%rax),%r13
   0x0000000000040be45 <+581>:    sub    $0x10,%rdx
   0x0000000000040be49 <+585>:    cmp    %rdi,%rdx
   0x0000000000040be4c <+588>:    mov    0x10(%rax),%rax
   0x0000000000040be50 <+592>:    mov    %r13,%rcx
---Type <return> to continue, or q <return> to quit---
   0x0000000000040be53 <+595>:    jne    0x40be30 <__libc_message+560>
   0x0000000000040be55 <+597>:    movslq -0x730(%rbp),%r10
   0x0000000000040be5c <+604>:    mov    $0x14,%r15d
   0x0000000000040be62 <+610>:    nopw   0x0(%rax,%rax,1)
   0x0000000000040be68 <+616>:    mov    %r9,%rdx
   0x0000000000040be6b <+619>:    mov    %r8,%rsi
   0x0000000000040be6e <+622>:    mov    %r10,%rdi
   0x0000000000040be71 <+625>:    mov    %r15d,%eax
   0x0000000000040be74 <+628>:    syscall
   0x0000000000040be76 <+630>:    cmp    $0xfffffffffffffffc,%rax
   0x0000000000040be7a <+634>:    mov    %rax,%r14
   0x0000000000040be7d <+637>:    je     0x40be68 <__libc_message+616>
   0x0000000000040be7f <+639>:    mov    -0x72c(%rbp),%ecx
   0x0000000000040be85 <+645>:    test   %ecx,%ecx
   0x0000000000040be87 <+647>:    jne    0x40bef2 <__libc_message+754>
   0x0000000000040be89 <+649>:    cmp    %r14,%r13
   0x0000000000040be8c <+652>:    jne    0x40bed6 <__libc_message+726>
   0x0000000000040be8e <+654>:    mov    $0x1,%eax
   0x0000000000040be93 <+659>:    mov    -0x72c(%rbp),%edx
   0x0000000000040be99 <+665>:    test   %edx,%edx
   0x0000000000040be9b <+667>:    jne    0x40bf49 <__libc_message+841>
   0x0000000000040bea1 <+673>:    lea    -0x28(%rbp),%rsp
   0x0000000000040bea5 <+677>:    pop    %rbx
   0x0000000000040bea6 <+678>:    pop    %r12
---Type <return> to continue, or q <return> to quit---
   0x0000000000040bea8 <+680>:    pop    %r13
   0x0000000000040beaa <+682>:    pop    %r14
   0x0000000000040beac <+684>:    pop    %r15
   0x0000000000040beae <+686>:    leaveq
```

```
   0x000000000040beaf <+687>:    retq
   0x000000000040beb0 <+688>:    movl   $0x2,-0x730(%rbp)
   0x000000000040beba <+698>:    jmpq   0x40bd02 <__libc_message+258>
   0x000000000040bebf <+703>:    mov    -0xf8(%rbp),%rax
   0x000000000040bec6 <+710>:    lea    0x8(%rax),%rdx
   0x000000000040beca <+714>:    mov    %rdx,-0xf8(%rbp)
   0x000000000040bed1 <+721>:    jmpq   0x40bdd1 <__libc_message+465>
   0x000000000040bed6 <+726>:    mov    -0x728(%rbp),%rsi
   0x000000000040bedd <+733>:    lea    -0x120(%rbp),%rdx
   0x000000000040bee4 <+740>:    mov    $0x3,%edi
   0x000000000040bee9 <+745>:    callq  0x431520 <vsyslog>
   0x000000000040beee <+750>:    xor    %eax,%eax
   0x000000000040bef0 <+752>:    jmp    0x40be93 <__libc_message+659>
   0x000000000040bef2 <+754>:    lea    0x1(%r13),%rdi
   0x000000000040bef6 <+758>:    callq  0x416c90 <malloc>
   0x000000000040befb <+763>:    test   %rax,%rax
   0x000000000040befe <+766>:    mov    %rax,-0x738(%rbp)
   0x000000000040bf05 <+773>:    je     0x40be89 <__libc_message+649>
   0x000000000040bf07 <+775>:    xor    %r15d,%r15d
   0x000000000040bf0a <+778>:    nopw   0x0(%rax,%rax,1)
---Type <return> to continue, or q <return> to quit---
   0x000000000040bf10 <+784>:    mov    0x8(%r12),%rdx
   0x000000000040bf15 <+789>:    mov    (%r12),%rsi
   0x000000000040bf19 <+793>:    mov    %rax,%rdi
   0x000000000040bf1c <+796>:    add    $0x1,%r15d
   0x000000000040bf20 <+800>:    add    $0x10,%r12
   0x000000000040bf24 <+804>:    callq  0x41e5b0 <mempcpy>
   0x000000000040bf29 <+809>:    cmp    %ebx,%r15d
   0x000000000040bf2c <+812>:    jl     0x40bf10 <__libc_message+784>
   0x000000000040bf2e <+814>:    movb   $0x0,(%rax)
   0x000000000040bf31 <+817>:    mov    -0x738(%rbp),%rdi
   0x000000000040bf38 <+824>:    xchg   %rdi,0x2c5721(%rip)        # 0x6d1660 <__abort_msg>
   0x000000000040bf3f <+831>:    callq  0x416bc0 <free>
   0x000000000040bf44 <+836>:    jmpq   0x40be89 <__libc_message+649>
   0x000000000040bf49 <+841>:    cmpl   $0x1,-0x72c(%rbp)
   0x000000000040bf50 <+848>:    jle    0x40bf56 <__libc_message+854>
   0x000000000040bf52 <+850>:    test   %al,%al
   0x000000000040bf54 <+852>:    jne    0x40bf5b <__libc_message+859>
   0x000000000040bf56 <+854>:    callq  0x409e40 <abort>
   0x000000000040bf5b <+859>:    lea    -0x320(%rbp),%rbx
   0x000000000040bf62 <+866>:    mov    $0x40,%esi
   0x000000000040bf67 <+871>:    mov    %rbx,%rdi
   0x000000000040bf6a <+874>:    callq  0x432e70 <backtrace>
   0x000000000040bf6f <+879>:    cmp    $0x2,%eax
---Type <return> to continue, or q <return> to quit---
   0x000000000040bf72 <+882>:    mov    %eax,%r12d
   0x000000000040bf75 <+885>:    jle    0x40bf56 <__libc_message+854>
   0x000000000040bf77 <+887>:    mov    -0x730(%rbp),%edi
   0x000000000040bf7d <+893>:    mov    $0x1d,%edx
   0x000000000040bf82 <+898>:    mov    $0x48d8b7,%esi
   0x000000000040bf87 <+903>:    lea    -0x720(%rbp),%r13
   0x000000000040bf8e <+910>:    callq  0x430759 <__write_nocancel>
   0x000000000040bf93 <+915>:    mov    -0x730(%rbp),%edx
   0x000000000040bf99 <+921>:    lea    -0x1(%r12),%esi
   0x000000000040bf9e <+926>:    lea    0x8(%rbx),%rdi
   0x000000000040bfa2 <+930>:    callq  0x432f50 <backtrace_symbols_fd>
   0x000000000040bfa7 <+935>:    mov    -0x730(%rbp),%edi
   0x000000000040bfad <+941>:    mov    $0x1d,%edx
   0x000000000040bfb2 <+946>:    mov    $0x48d8d5,%esi
   0x000000000040bfb7 <+951>:    callq  0x430759 <__write_nocancel>
```

```
   0x000000000040bfbc <+956>:    xor    %esi,%esi
   0x000000000040bfbe <+958>:    mov    $0x48d8f3,%edi
   0x000000000040bfc3 <+963>:    xor    %eax,%eax
   0x000000000040bfc5 <+965>:    callq  0x430639 <__open_nocancel>
   0x000000000040bfca <+970>:    mov    %eax,%r12d
   0x000000000040bfcd <+973>:    mov    $0x400,%edx
   0x000000000040bfd2 <+978>:    mov    %r13,%rsi
   0x000000000040bfd5 <+981>:    mov    %r12d,%edi
   0x000000000040bfd8 <+984>:    callq  0x4306f9 <__read_nocancel>
---Type <return> to continue, or q <return> to quit---
   0x000000000040bfdd <+989>:    movslq %eax,%rbx
   0x000000000040bfe0 <+992>:    test   %rbx,%rbx
   0x000000000040bfe3 <+995>:    jle    0x40bffd <__libc_message+1021>
   0x000000000040bfe5 <+997>:    mov    -0x730(%rbp),%edi
   0x000000000040bfeb <+1003>:   mov    %rbx,%rdx
   0x000000000040bfee <+1006>:   mov    %r13,%rsi
   0x000000000040bff1 <+1009>:   callq  0x430759 <__write_nocancel>
   0x000000000040bff6 <+1014>:   cltq
   0x000000000040bff8 <+1016>:   cmp    %rbx,%rax
   0x000000000040bffb <+1019>:   je     0x40bfcd <__libc_message+973>
   0x000000000040bffd <+1021>:   movslq %r12d,%rdi
   0x000000000040c000 <+1024>:   mov    $0x3,%eax
   0x000000000040c005 <+1029>:   syscall
   0x000000000040c007 <+1031>:   jmpq   0x40bf56 <__libc_message+854>
End of assembler dump.

(gdb) x 0x6d1660
0x6d1660 <__abort_msg>:   0x00007f26540008b0

(gdb) x/s 0x7f26540008b0
0x7f26540008b0:     "*** glibc detected *** ./App10: double free or corruption (!prev):
0x0000000001b681a0 ***\n"
```

9. Check the address that was being freed:

```
(gdb) bt
#0  0x000000000043ef65 in raise ()
#1  0x0000000000409fc0 in abort ()
#2  0x000000000040bf5b in __libc_message ()
#3  0x0000000000412042 in malloc_printerr ()
#4  0x0000000000416c27 in free ()
#5  0x0000000000400586 in proc ()
#6  0x000000000040067e in bar_four ()
#7  0x000000000040068e in foo_four ()
#8  0x00000000004006a6 in thread_four ()
#9  0x00000000004016c0 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#10 0x0000000000432589 in clone ()
#11 0x0000000000000000 in ?? ()

(gdb) frame 5
#5  0x0000000000400586 in proc ()
```

```
(gdb) disassemble proc
Dump of assembler code for function proc:
   0x00000000004004f0 <+0>:     push   %rbp
   0x00000000004004f1 <+1>:     mov    %rsp,%rbp
   0x00000000004004f4 <+4>:     push   %rbx
   0x00000000004004f5 <+5>:     sub    $0x18,%rsp
   0x00000000004004f9 <+9>:     callq  0x40ac70 <rand>
   0x00000000004004fe <+14>:    mov    %eax,%ecx
   0x0000000000400500 <+16>:    mov    $0x68db8bad,%edx
   0x0000000000400505 <+21>:    mov    %ecx,%eax
   0x0000000000400507 <+23>:    imul   %edx
   0x0000000000400509 <+25>:    sar    $0xc,%edx
   0x000000000040050c <+28>:    mov    %ecx,%eax
   0x000000000040050e <+30>:    sar    $0x1f,%eax
   0x0000000000400511 <+33>:    mov    %edx,%ebx
   0x0000000000400513 <+35>:    sub    %eax,%ebx
   0x0000000000400515 <+37>:    mov    %ebx,%eax
   0x0000000000400517 <+39>:    mov    %eax,-0x14(%rbp)
   0x000000000040051a <+42>:    mov    -0x14(%rbp),%eax
   0x000000000040051d <+45>:    imul   $0x2710,%eax,%eax
   0x0000000000400523 <+51>:    mov    %ecx,%edx
   0x0000000000400525 <+53>:    sub    %eax,%edx
   0x0000000000400527 <+55>:    mov    %edx,%eax
   0x0000000000400529 <+57>:    mov    %eax,-0x14(%rbp)
   0x000000000040052c <+60>:    callq  0x40ac70 <rand>
---Type <return> to continue, or q <return> to quit---
   0x0000000000400531 <+65>:    mov    %eax,%ecx
   0x0000000000400533 <+67>:    mov    $0x68db8bad,%edx
   0x0000000000400538 <+72>:    mov    %ecx,%eax
   0x000000000040053a <+74>:    imul   %edx
   0x000000000040053c <+76>:    sar    $0xc,%edx
   0x000000000040053f <+79>:    mov    %ecx,%eax
   0x0000000000400541 <+81>:    sar    $0x1f,%eax
   0x0000000000400544 <+84>:    mov    %edx,%ebx
   0x0000000000400546 <+86>:    sub    %eax,%ebx
   0x0000000000400548 <+88>:    mov    %ebx,%eax
   0x000000000040054a <+90>:    mov    %eax,-0x18(%rbp)
   0x000000000040054d <+93>:    mov    -0x18(%rbp),%eax
   0x0000000000400550 <+96>:    imul   $0x2710,%eax,%eax
   0x0000000000400556 <+102>:   mov    %ecx,%edx
   0x0000000000400558 <+104>:   sub    %eax,%edx
   0x000000000040055a <+106>:   mov    %edx,%eax
   0x000000000040055c <+108>:   mov    %eax,-0x18(%rbp)
   0x000000000040055f <+111>:   mov    -0x14(%rbp),%eax
   0x0000000000400562 <+114>:   cltq
   0x0000000000400564 <+116>:   mov    0x6b8fc0(,%rax,8),%rax
   0x000000000040056c <+124>:   test   %rax,%rax
   0x000000000040056f <+127>:   je     0x400597 <proc+167>
   0x0000000000400571 <+129>:   mov    -0x14(%rbp),%eax
   0x0000000000400574 <+132>:   cltq
---Type <return> to continue, or q <return> to quit---
   0x0000000000400576 <+134>:   mov    0x6b8fc0(,%rax,8),%rax
   0x000000000040057e <+142>:   mov    %rax,%rdi
   0x0000000000400581 <+145>:   callq  0x416bc0 <free>
=> 0x0000000000400586 <+150>:   mov    -0x14(%rbp),%eax
   0x0000000000400589 <+153>:   cltq
   0x000000000040058b <+155>:   movq   $0x0,0x6b8fc0(,%rax,8)
   0x0000000000400597 <+167>:   mov    -0x18(%rbp),%eax
   0x000000000040059a <+170>:   cltq
   0x000000000040059c <+172>:   mov    %rax,%rdi
```

```
   0x000000000040059f <+175>:    callq  0x416c90 <malloc>
   0x00000000004005a4 <+180>:    mov    %rax,%rdx
   0x00000000004005a7 <+183>:    mov    -0x14(%rbp),%eax
   0x00000000004005aa <+186>:    cltq
   0x00000000004005ac <+188>:    mov    %rdx,0x6b8fc0(,%rax,8)
   0x00000000004005b4 <+196>:    jmpq   0x4004f9 <proc+9>
End of assembler dump.

(gdb) x/dw $rbp-0x14
0x7f2664374d2c:     5084

(gdb) x/xg 0x6b8fc0+5084*8
0x6c2ea0 <pAllocBuf+40672>:      0x00007f265c6fc360
```

10. Dump the first 1000 elements of array *pAllocBuf* (0x6b8fc0) found in *proc* function disassembly:

```
(gdb) print/x *0x6b8fc0@1000
$0 = {0x1cbd6e0, 0x0, 0x5c3be260, 0x7f26, 0x0, 0x0, 0x0, 0x0, 0x1f1bad0, 0x0,
  0x0, 0x0, 0x25da380, 0x0, 0x0, 0x0, 0x1cf9240, 0x0, 0x0, 0x0, 0x0, 0x0,
  0x20a1d00, 0x0, 0x5ca57a90, 0x7f26, 0x27254c0, 0x0, 0x0, 0x0, 0x0, 0x0,
  0x1c578c0, 0x0, 0x1f12f70, 0x0, 0x5c646e10, 0x7f26, 0x5ce6d7b0, 0x7f26,
  0x5c6fafd0, 0x7f26, 0x1fcdbe0, 0x0, 0x0, 0x0, 0x5c4d6680, 0x7f26,
  0x5c9cea20, 0x7f26, 0x0, 0x0, 0x5ce85c40, 0x7f26, 0x5ccfd170, 0x7f26,
  0x1bcd0c0, 0x0, 0x1f9f660, 0x0, 0x0, 0x0, 0x29329a0, 0x0, 0x5c070600,
  0x7f26, 0x0, 0x0, 0x5cc061f0, 0x7f26, 0x5caca2e0, 0x7f26, 0x5caeda00,
  0x7f26, 0x5cb43450, 0x7f26, 0x5cde4670, 0x7f26, 0x5c2b7a40, 0x7f26, 0x0,
  0x0, 0x255ea90, 0x0, 0x1b03850, 0x0, 0x0, 0x0, 0x5cb9d570, 0x7f26, 0x0, 0x0,
  0x5c5e6910, 0x7f26, 0x20f8740, 0x0, 0x0, 0x0, 0x1e4b790, 0x0, 0x5c28f3b0,
  0x7f26, 0x0, 0x0, 0x1f6d790, 0x0, 0x5c63d3e0, 0x7f26, 0x5c84bc10, 0x7f26,
  0x27495e0, 0x0, 0x5ccb2950, 0x7f26, 0x5c1fa130, 0x7f26, 0x5c363ea0, 0x7f26,
  0x1c6d620, 0x0, 0x5c6e41b0, 0x7f26, 0x5ccb67b0, 0x7f26, 0x0, 0x0, 0x2477a20,
  0x0, 0x0, 0x0, 0x0, 0x0, 0x5c323d80, 0x7f26, 0x5c6b9ee0, 0x7f26, 0x1f1abe0,
  0x0, 0x5cdb8bb0, 0x7f26, 0x221fd00, 0x0, 0x0, 0x0, 0x2691b60, 0x0, 0x0, 0x0,
  0x5c5d4890, 0x7f26, 0x5cab7920, 0x7f26, 0x5c779f40, 0x7f26, 0x5c46dc10,
  0x7f26, 0x0, 0x0, 0x5cb74cd0, 0x7f26, 0x0, 0x0, 0x5c7b08c0, 0x7f26,
  0x20df200, 0x0, 0x1e163e0, 0x0, 0x5c0e5f30, 0x7f26, 0x5c183aa0, 0x7f26,
  0x24f5d80, 0x0, 0x0, 0x0, 0x23e48a0, 0x0, 0x0, 0x0, 0x5c3c04a0, 0x7f26,
  0x24d9690, 0x0, 0x0, 0x0, 0x5c86dba0, 0x7f26, 0x5c6b1310, 0x7f26, 0x0, 0x0,
  0x5c142a20, 0x7f26, 0x1aee3a0, 0x0, 0x293bd80, 0x0, 0x0, 0x0...}

(gdb) x/1000xg 0x6b8fc0
0x6b8fc0 <pAllocBuf>:     0x0000000001cbd6e0  0x00007f265c3be260
0x6b8fd0 <pAllocBuf+16>:  0x0000000000000000  0x0000000000000000
0x6b8fe0 <pAllocBuf+32>:  0x0000000001f1bad0  0x0000000000000000
0x6b8ff0 <pAllocBuf+48>:  0x00000000025da380  0x0000000000000000
0x6b9000 <pAllocBuf+64>:  0x0000000001cf9240  0x0000000000000000
0x6b9010 <pAllocBuf+80>:  0x0000000000000000  0x00000000020a1d00
0x6b9020 <pAllocBuf+96>:  0x00007f265ca57a90  0x00000000027254c0
0x6b9030 <pAllocBuf+112>: 0x0000000000000000  0x0000000000000000
0x6b9040 <pAllocBuf+128>: 0x0000000001c578c0  0x0000000001f12f70
0x6b9050 <pAllocBuf+144>: 0x00007f265c646e10  0x00007f265ce6d7b0
0x6b9060 <pAllocBuf+160>: 0x00007f265c6fafd0  0x0000000001fcdbe0
0x6b9070 <pAllocBuf+176>: 0x0000000000000000  0x00007f265c4d6680
0x6b9080 <pAllocBuf+192>: 0x00007f265c9cea20  0x0000000000000000
0x6b9090 <pAllocBuf+208>: 0x00007f265ce85c40  0x00007f265ccfd170
0x6b90a0 <pAllocBuf+224>: 0x0000000001bcd0c0  0x0000000001f9f660
0x6b90b0 <pAllocBuf+240>: 0x0000000000000000  0x00000000029329a0
0x6b90c0 <pAllocBuf+256>: 0x00007f265c070600  0x0000000000000000
0x6b90d0 <pAllocBuf+272>: 0x00007f265cc061f0  0x00007f265caca2e0
0x6b90e0 <pAllocBuf+288>: 0x00007f265caeda00  0x00007f265cb43450
```

```
0x6b90f0 <pAllocBuf+304>:  0x00007f265cde4670  0x00007f265c2b7a40
0x6b9100 <pAllocBuf+320>:  0x0000000000000000  0x000000000255ea90
0x6b9110 <pAllocBuf+336>:  0x0000000001b03850  0x0000000000000000
0x6b9120 <pAllocBuf+352>:  0x00007f265cb9d570  0x0000000000000000
0x6b9130 <pAllocBuf+368>:  0x00007f265c5e6910  0x00000000020f8740
---Type <return> to continue, or q <return> to quit---
0x6b9140 <pAllocBuf+384>:  0x0000000000000000  0x0000000001e4b790
0x6b9150 <pAllocBuf+400>:  0x00007f265c28f3b0  0x0000000000000000
0x6b9160 <pAllocBuf+416>:  0x0000000001f6d790  0x00007f265c63d3e0
0x6b9170 <pAllocBuf+432>:  0x00007f265c84bc10  0x00000000027495e0
0x6b9180 <pAllocBuf+448>:  0x00007f265ccb2950  0x00007f265c1fa130
0x6b9190 <pAllocBuf+464>:  0x00007f265c363ea0  0x0000000001c6d620
0x6b91a0 <pAllocBuf+480>:  0x00007f265c6e41b0  0x00007f265ccb67b0
0x6b91b0 <pAllocBuf+496>:  0x0000000000000000  0x0000000002477a20
0x6b91c0 <pAllocBuf+512>:  0x0000000000000000  0x0000000000000000
0x6b91d0 <pAllocBuf+528>:  0x00007f265c323d80  0x00007f265c6b9ee0
0x6b91e0 <pAllocBuf+544>:  0x0000000001f1abe0  0x00007f265cdb8bb0
0x6b91f0 <pAllocBuf+560>:  0x000000000221fd00  0x0000000000000000
0x6b9200 <pAllocBuf+576>:  0x0000000002691b60  0x0000000000000000
0x6b9210 <pAllocBuf+592>:  0x00007f265c5d4890  0x00007f265cab7920
0x6b9220 <pAllocBuf+608>:  0x00007f265c779f40  0x00007f265c46dc10
0x6b9230 <pAllocBuf+624>:  0x0000000000000000  0x00007f265cb74cd0
0x6b9240 <pAllocBuf+640>:  0x0000000000000000  0x00007f265c7b08c0
0x6b9250 <pAllocBuf+656>:  0x00000000020df200  0x0000000001e163e0
0x6b9260 <pAllocBuf+672>:  0x00007f265c0e5f30  0x00007f265c183aa0
0x6b9270 <pAllocBuf+688>:  0x00000000024f5d80  0x0000000000000000
0x6b9280 <pAllocBuf+704>:  0x00000000023e48a0  0x0000000000000000
0x6b9290 <pAllocBuf+720>:  0x00007f265c3c04a0  0x00000000024d9690
0x6b92a0 <pAllocBuf+736>:  0x0000000000000000  0x00007f265c86dba0
0x6b92b0 <pAllocBuf+752>:  0x00007f265c6b1310  0x0000000000000000
---Type <return> to continue, or q <return> to quit---q
Quit
```

Exercise A11

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

© 2015 Software Diagnostics Services

Exercise A11

Goal: Learn how to identify synchronization wait chains, deadlocks, hidden and handled exceptions.

Patterns: Wait Chains, Deadlock, Execution Residue, Handled Exception.

1. Load a core dump *core.18781* and *App11* executable:

```
training@debian64:~/ALCDA$ gdb -c ./App11/core.18781 -se ./App11/App11
GNU gdb (GDB) 7.4.1-debian
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/training/ALCDA/App11/App11...done.
[New LWP 18782]
[New LWP 18783]
[New LWP 18784]
[New LWP 18785]
[New LWP 18786]
[New LWP 18781]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Core was generated by `/home/training/ALCDA/App11/App11'.
#0  0x000000000043e4f1 in nanosleep ()
```

2. List all thread stack traces and identify possible wait chain and deadlock:

```
(gdb) thread apply all bt

Thread 6 (LWP 18781):
#0  0x000000000043e4f1 in nanosleep ()
#1  0x000000000043e3c0 in sleep ()
#2  0x0000000000400789 in main ()

Thread 5 (LWP 18786):
#0  0x000000000043e4f1 in nanosleep ()
#1  0x000000000043e3c0 in sleep ()
#2  0x000000000040069c in bar_five() ()
#3  0x00000000004006a7 in foo_five() ()
#4  0x00000000004006ba in thread_five(void*) ()
#5  0x000000000040f560 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#6  0x0000000000440b49 in clone ()
#7  0x0000000000000000 in ?? ()

Thread 4 (LWP 18785):
#0  __lll_lock_wait ()
    at ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S:136
#1  0x0000000000410fa3 in _L_lock_926 ()
#2  0x0000000000410ddb in __pthread_mutex_lock (mutex=0x6c5900)
    at pthread_mutex_lock.c:61
```

```
#3  0x00000000004005ac in procB() ()
---Type <return> to continue, or q <return> to quit---
#4  0x0000000000400669 in bar_four() ()
#5  0x0000000000400674 in foo_four() ()
#6  0x0000000000400687 in thread_four(void*) ()
#7  0x000000000040f560 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#8  0x0000000000440b49 in clone ()
#9  0x0000000000000000 in ?? ()

Thread 3 (LWP 18784):
#0  0x000000000043e4f1 in nanosleep ()
#1  0x000000000043e3c0 in sleep ()
#2  0x00000000004006 3b in bar_three() ()
#3  0x0000000000400646 in foo_three() ()
#4  0x0000000000400659 in thread_three(void*) ()
#5  0x000000000040f560 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#6  0x0000000000440b49 in clone ()
#7  0x0000000000000000 in ?? ()

Thread 2 (LWP 18783):
#0  __lll_lock_wait ()
    at ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S:136
#1  0x0000000000410fa3 in _L_lock_926 ()
#2  0x0000000000410ddb in __pthread_mutex_lock (mutex=0x6c5940)
---Type <return> to continue, or q <return> to quit---
    at pthread_mutex_lock.c:61
#3  0x0000000000400577 in procA() ()
#4  0x0000000000400608 in bar_two() ()
#5  0x0000000000400613 in foo_two() ()
#6  0x0000000000400626 in thread_two(void*) ()
#7  0x000000000040f560 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#8  0x0000000000440b49 in clone ()
#9  0x0000000000000000 in ?? ()

Thread 1 (LWP 18782):
#0  0x000000000043e4f1 in nanosleep ()
#1  0x000000000043e3c0 in sleep ()
#2  0x00000000004005da in bar_one() ()
#3  0x00000000004005e5 in foo_one() ()
#4  0x00000000004005f8 in thread_one(void*) ()
#5  0x000000000040f560 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#6  0x0000000000440b49 in clone ()
#7  0x0000000000000000 in ?? ()
```

3. Check the thread #4 and its waiting code:

```
(gdb) thread 4
[Switching to thread 4 (LWP 18785)]
#0  __lll_lock_wait ()
    at ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S:136
136    ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S: No such file or directory.
```

```
(gdb) bt
#0  __lll_lock_wait ()
    at ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S:136
#1  0x0000000000410fa3 in _L_lock_926 ()
#2  0x0000000000410ddb in __pthread_mutex_lock (mutex=0x6c5900)
    at pthread_mutex_lock.c:61
#3  0x00000000004005ac in procB() ()
#4  0x0000000000400669 in bar_four() ()
#5  0x0000000000400674 in foo_four() ()
#6  0x0000000000400687 in thread_four(void*) ()
#7  0x000000000040f560 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#8  0x0000000000440b49 in clone ()
#9  0x0000000000000000 in ?? ()

(gdb) disassemble procB
Dump of assembler code for function _Z5procBv:
   0x0000000000400594 <+0>:     push   %rbp
   0x0000000000400595 <+1>:     mov    %rsp,%rbp
   0x0000000000400598 <+4>:     mov    $0x6c5940,%edi
   0x000000000040059d <+9>:     callq  0x410da0 <__pthread_mutex_lock>
   0x00000000004005a2 <+14>:    mov    $0x6c5900,%edi
   0x00000000004005a7 <+19>:    callq  0x410da0 <__pthread_mutex_lock>
   0x00000000004005ac <+24>:    mov    $0x1e,%edi
   0x00000000004005b1 <+29>:    callq  0x43e2e0 <sleep>
   0x00000000004005b6 <+34>:    mov    $0x6c5900,%edi
   0x00000000004005bb <+39>:    callq  0x410da0 <__pthread_mutex_lock>
   0x00000000004005c0 <+44>:    mov    $0x6c5940,%edi
   0x00000000004005c5 <+49>:    callq  0x410da0 <__pthread_mutex_lock>
   0x00000000004005ca <+54>:    pop    %rbp
   0x00000000004005cb <+55>:    retq
End of assembler dump.
```

We see the thread #4 owns mutex **0x6c5940** but is waiting for mutex **0x6c5900**.

4. Check the thread #2 and its waiting code:

```
(gdb) thread 2
[Switching to thread 2 (LWP 18783)]
#0  __lll_lock_wait ()
    at ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S:136
136    in ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S

(gdb) bt
#0  __lll_lock_wait ()
    at ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S:136
#1  0x0000000000410fa3 in _L_lock_926 ()
#2  0x0000000000410ddb in __pthread_mutex_lock (mutex=0x6c5940)
    at pthread_mutex_lock.c:61
#3  0x0000000000400577 in procA() ()
#4  0x0000000000400608 in bar_two() ()
#5  0x0000000000400613 in foo_two() ()
#6  0x0000000000400626 in thread_two(void*) ()
#7  0x000000000040f560 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#8  0x0000000000440b49 in clone ()
#9  0x0000000000000000 in ?? ()
```

```
(gdb) disassemble procA
Dump of assembler code for function _Z5procAv:
   0x0000000000400546 <+0>:     push   %rbp
   0x0000000000400547 <+1>:     mov    %rsp,%rbp
   0x000000000040054a <+4>:     mov    $0x6c5900,%edi
   0x000000000040054f <+9>:     callq  0x410da0 <__pthread_mutex_lock>
   0x0000000000400554 <+14>:    callq  0x400520 <_Z5procCv>
   0x0000000000400559 <+19>:    mov    $0x6c5900,%edi
   0x000000000040055e <+24>:    callq  0x411a10 <__pthread_mutex_unlock>
   0x0000000000400563 <+29>:    mov    $0x14,%edi
   0x0000000000400568 <+34>:    callq  0x43e2e0 <sleep>
   0x000000000040056d <+39>:    mov    $0x6c5940,%edi
   0x0000000000400572 <+44>:    callq  0x410da0 <__pthread_mutex_lock>
   0x0000000000400577 <+49>:    mov    $0x6c5940,%edi
   0x000000000040057c <+54>:    callq  0x411a10 <__pthread_mutex_unlock>
   0x0000000000400581 <+59>:    jmp    0x400592 <_Z5procAv+76>
   0x0000000000400583 <+61>:    mov    %rax,%rdi
   0x0000000000400586 <+64>:    callq  0x4019a0 <__cxa_begin_catch>
   0x000000000040058b <+69>:    callq  0x401a10 <__cxa_end_catch>
   0x0000000000400590 <+74>:    jmp    0x400563 <_Z5procAv+29>
   0x0000000000400592 <+76>:    pop    %rbp
   0x0000000000400593 <+77>:    retq
End of assembler dump.
```

We see that the thread 2 is waiting for **0x6c5940** mutex but shouldn't own **0x6c5900** mutex because it should have unlocked it unless something happened in **procC**. We also notice catch exception processing which transfers execution for the block of code wiating for mutex **0x6c5940.**

5. Disassemble *procC* code:

```
(gdb) disassemble procC
Dump of assembler code for function _Z5procCv:
   0x0000000000400520 <+0>:     push   %rbp
   0x0000000000400521 <+1>:     mov    %rsp,%rbp
   0x0000000000400524 <+4>:     mov    $0x4,%edi
   0x0000000000400529 <+9>:     callq  0x400960 <__cxa_allocate_exception>
   0x000000000040052e <+14>:    movl   $0x0,(%rax)
   0x0000000000400534 <+20>:    mov    $0x0,%edx
   0x0000000000400539 <+25>:    mov    $0x6c4180,%esi
   0x000000000040053e <+30>:    mov    %rax,%rdi
   0x0000000000400541 <+33>:    callq  0x400ec0 <__cxa_throw>
End of assembler dump.
```

We see that code throws an exception so perhaps it was caught in the caller *procA* and mutex unlock wasn't called thus causing a deadlock.

6. Check if there was any exception processing:

```
(gdb) x/300a $rsp-2400
0x7f001b4a7318:     0x0    0x0
0x7f001b4a7328:     0x0    0x0
0x7f001b4a7338:     0x0    0x0
0x7f001b4a7348:     0x0    0x0
0x7f001b4a7358:     0x0    0x0
0x7f001b4a7368:     0x0    0x0
0x7f001b4a7378:     0x0    0x0
0x7f001b4a7388:     0x0    0x0
0x7f001b4a7398:     0x0    0x0
0x7f001b4a73a8:     0x0    0x0
```

```
0x7f001b4a73b8:     0x0    0x0
0x7f001b4a73c8:     0x0    0x0
0x7f001b4a73d8:     0x0    0x0
0x7f001b4a73e8:     0x0    0x0
0x7f001b4a73f8:     0x0    0x0
0x7f001b4a7408:     0x0    0x0
0x7f001b4a7418:     0x0    0x0
0x7f001b4a7428:     0x0    0x0
0x7f001b4a7438:     0x0    0x0
0x7f001b4a7448:     0x0    0x0
0x7f001b4a7458:     0x0    0x0
0x7f001b4a7468:     0x0    0x0
0x7f001b4a7478:     0x0    0x0
0x7f001b4a7488:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7f001b4a7498:     0x0    0x0
0x7f001b4a74a8:     0x0    0x0
0x7f001b4a74b8:     0x0    0x0
0x7f001b4a74c8:     0x0    0x0
0x7f001b4a74d8:     0x0    0x0
0x7f001b4a74e8:     0x0    0x0
0x7f001b4a74f8:     0x0    0x0
0x7f001b4a7508:     0x0    0x0
0x7f001b4a7518:     0x0    0x0
0x7f001b4a7528:     0x0    0x0
0x7f001b4a7538:     0x0    0x6c58c0 <object.5602>
0x7f001b4a7548:     0x0    0x1b
0x7f001b4a7558:     0x40d306 <fde_single_encoding_compare+118>    0x400520 <_Z5procCv>
0x7f001b4a7568:     0x400546 <_Z5procAv>      0x121b380
0x7f001b4a7578:     0x1    0x121b380
0x7f001b4a7588:     0x2    0x40d290 <fde_single_encoding_compare>
0x7f001b4a7598:     0x40cb4e <frame_downheap+78>     0x0
0x7f001b4a75a8:     0x6c58c0 <object.5602>    0x4c2bf8 <__EH_FRAME_BEGIN__+56848>
0x7f001b4a75b8:     0x121b370    0x121b380
0x7f001b4a75c8:     0x6c58c0 <object.5602>    0x40d290 <fde_single_encoding_compare>
0x7f001b4a75d8:     0x135  0x6c58c0 <object.5602>
0x7f001b4a75e8:     0x0    0x1b
0x7f001b4a75f8:     0x6dcb40 <main_arena>     0x40d290 <fde_single_encoding_compare>
---Type <return> to continue, or q <return> to quit---
0x7f001b4a7608:     0x0    0x1
0x7f001b4a7618:     0x1218960    0xa3
0x7f001b4a7628:     0x7f001b4a7660      0x0
0x7f001b4a7638:     0x4b4e40 <__EH_FRAME_BEGIN__+88> 0x2
0x7f001b4a7648:     0x40da09 <search_object+1209>    0x7f001b4a7680
0x7f001b4a7658:     0x400558 <_Z5procAv+18>   0x7f001b4a7688
0x7f001b4a7668:     0x7f000000001b      0xb
0x7f001b4a7678:     0x7f001b4a7d00      0x400546 <_Z5procAv>
0x7f001b4a7688:     0x4e   0x7f001b4a7cf0
0x7f001b4a7698:     0x7f001b4a7650      0x0
0x7f001b4a76a8:     0x6c58c0 <object.5602>    0x7f001b4a7ae8
0x7f001b4a76b8:     0x411c70 <pthread_cancel> 0x6c58c0 <object.5602>
0x7f001b4a76c8:     0x7f001b4a7ae8      0x1b
0x7f001b4a76d8:     0x40e510 <_Unwind_Find_FDE+208>  0x7f001b4a7d08
0x7f001b4a76e8:     0x4014ea <__gxx_personality_v0+202>     0x4c2c4c
0x7f001b4a76f8:     0x400546 <_Z5procAv>      0x0
0x7f001b4a7708:     0x0    0x0
0x7f001b4a7718:     0x4b4e55 <__EH_FRAME_BEGIN__+109>      0x0
0x7f001b4a7728:     0x7f001b4a7a40      0x4b4e55 <__EH_FRAME_BEGIN__+109>
0x7f001b4a7738:     0x3    0x7f001b4a77b0
0x7f001b4a7748:     0x40b69c <uw_frame_state_for+828>      0x3
```

```
0x7f001b4a7758:     0x4b4e33 <__EH_FRAME_BEGIN__+75> 0xfffffffffffffff8
0x7f001b4a7768:     0x4c2c4c      0x7f001b4a7950
0x7f001b4a7778:     0x7f001b4a7a40      0x4
---Type <return> to continue, or q <return> to quit---
0x7f001b4a7788:     0x1218930    0x7f001b4a7950
0x7f001b4a7798:     0x0    0x3
0x7f001b4a77a8:     0x40c0ab <_Unwind_RaiseException_Phase2+59>   0x0
0x7f001b4a77b8:     0x0    0x0
0x7f001b4a77c8:     0x0    0x0
0x7f001b4a77d8:     0x0    0x0
0x7f001b4a77e8:     0x0    0x0
0x7f001b4a77f8:     0x0    0x0
0x7f001b4a7808:     0x0    0xfffffffffffffff0
0x7f001b4a7818:     0x1    0x0
0x7f001b4a7828:     0x0    0x0
0x7f001b4a7838:     0x0    0x0
0x7f001b4a7848:     0x0    0x0
0x7f001b4a7858:     0x0    0x0
0x7f001b4a7868:     0x0    0x0
0x7f001b4a7878:     0x0    0x0
0x7f001b4a7888:     0x0    0x0
0x7f001b4a7898:     0x0    0x0
0x7f001b4a78a8:     0x0    0xfffffffffffffff8
0x7f001b4a78b8:     0x1    0x0
0x7f001b4a78c8:     0x0    0x0
0x7f001b4a78d8:     0x10   0x6
0x7f001b4a78e8:     0x0    0x1
0x7f001b4a78f8:     0x400593 <_Z5procAv+77>   0x401420 <__gxx_personality_v0>
---Type <return> to continue, or q <return> to quit---
0x7f001b4a7908:     0xfffffffffffffff8  0x1
0x7f001b4a7918:     0x40ba6f <uw_install_context_1+191>    0x7f001b4a7d20
0x7f001b4a7928:     0x0    0x7f001b4a7a40
0x7f001b4a7938:     0x7f001b4a7cf0      0x1218930
0x7f001b4a7948:     0x40c6d5 <_Unwind_RaiseException+309>  0x7f001b4a7cb8
0x7f001b4a7958:     0x7f001b4a7cc0      0x0
0x7f001b4a7968:     0x7f001b4a7cc8      0x0
0x7f001b4a7978:     0x0    0x7f001b4a7cf0
0x7f001b4a7988:     0x0    0x0
0x7f001b4a7998:     0x0    0x0
0x7f001b4a79a8:     0x0    0x7f001b4a7cd0
0x7f001b4a79b8:     0x7f001b4a7cd8      0x7f001b4a7ce0
0x7f001b4a79c8:     0x7f001b4a7ce8      0x7f001b4a7cf8
0x7f001b4a79d8:     0x0    0x7f001b4a7d00
0x7f001b4a79e8:     0x400f11 <__cxa_throw+81> 0x0
0x7f001b4a79f8:     0x0    0x0
0x7f001b4a7a08:     0x0    0x4000000000000000
0x7f001b4a7a18:     0x0    0x0
0x7f001b4a7a28:     0x0    0x0
0x7f001b4a7a38:     0x0    0x7f001b4a7cb8
0x7f001b4a7a48:     0x7f001b4a7cc0      0x0
0x7f001b4a7a58:     0x7f001b4a7d00      0x0
0x7f001b4a7a68:     0x0    0x7f001b4a7d10
0x7f001b4a7a78:     0x7f001b4a7920      0x0
---Type <return> to continue, or q <return> to quit---
0x7f001b4a7a88:     0x0    0x0
0x7f001b4a7a98:     0x0    0x7f001b4a7cd0
0x7f001b4a7aa8:     0x7f001b4a7cd8      0x7f001b4a7ce0
0x7f001b4a7ab8:     0x7f001b4a7ce8      0x7f001b4a7d18
0x7f001b4a7ac8:     0x0    0x7f001b4a7d20
0x7f001b4a7ad8:     0x400583 <_Z5procAv+61>   0x4c2c4c
```

```
0x7f001b4a7ae8:     0x0    0x0
0x7f001b4a7af8:     0x400546 <_Z5procAv>       0x4000000000000000
0x7f001b4a7b08:     0x0    0x0
0x7f001b4a7b18:     0x0    0x0
0x7f001b4a7b28:     0x43e4fd <nanosleep+61>    0x0
0x7f001b4a7b38:     0x43e3c0 <sleep+224>       0x0
0x7f001b4a7b48:     0x0    0x4000000000000000
0x7f001b4a7b58:     0x0    0x0
0x7f001b4a7b68:     0x0    0x0
0x7f001b4a7b78:     0x0    0x7f001b4a7cb8
0x7f001b4a7b88:     0x7f001b4a7cc0      0x0
0x7f001b4a7b98:     0x7f001b4a7d00      0x0
0x7f001b4a7ba8:     0x0    0x7f001b4a7d10
0x7f001b4a7bb8:     0x7f001b4a7920      0x0
0x7f001b4a7bc8:     0x0    0x0
0x7f001b4a7bd8:     0x0    0x0
0x7f001b4a7be8:     0x0    0x0
0x7f001b4a7bf8:     0x0    0x0
---Type <return> to continue, or q <return> to quit---
0x7f001b4a7c08:     0x0    0x0
0x7f001b4a7c18:     0x0    0x0
0x7f001b4a7c28:     0x0    0xfffffffffffffff8
0x7f001b4a7c38:     0x1    0x0
0x7f001b4a7c48:     0x0    0x0
0x7f001b4a7c58:     0x10   0x10000
0x7f001b4a7c68:     0x0    0x0
```

We see a reference **0x400583 <_Z5procAv+61>** from exception processing block in *procA* and also **0x400f11 <__cxa_throw+81>.** We check whether the symbolic information we found is not coincidental:

```
(gdb) disaassemble __cxa_throw
Dump of assembler code for function __cxa_throw:
   0x0000000000400ec0 <+0>:     mov    0x2c3bf1(%rip),%rax        # 0x6c4ab8
   0x0000000000400ec7 <+7>:     push   %rbx
   0x0000000000400ec8 <+8>:     lea    -0x20(%rdi),%rbx
   0x0000000000400ecc <+12>:    mov    %rsi,-0x70(%rdi)
   0x0000000000400ed0 <+16>:    mov    %rdx,-0x68(%rdi)
   0x0000000000400ed4 <+20>:    movl   $0x1,-0x80(%rdi)
   0x0000000000400edb <+27>:    mov    (%rax),%rax
   0x0000000000400ede <+30>:    mov    %rax,-0x60(%rdi)
   0x0000000000400ee2 <+34>:    mov    0x2c3baf(%rip),%rax        # 0x6c4a98
   0x0000000000400ee9 <+41>:    mov    (%rax),%rax
   0x0000000000400eec <+44>:    mov    %rax,-0x58(%rdi)
   0x0000000000400ef0 <+48>:    movabs $0x474e5543432b2b00,%rax
   0x0000000000400efa <+58>:    mov    %rax,-0x20(%rdi)
   0x0000000000400efe <+62>:    lea    -0x95(%rip),%rax        # 0x400e70
<_ZL23__gxx_exception_cleanup19_Unwind_Reason_CodeP17_Unwind_Exception>
   0x0000000000400f05 <+69>:    mov    %rax,-0x18(%rdi)
   0x0000000000400f09 <+73>:    mov    %rbx,%rdi
   0x0000000000400f0c <+76>:    callq  0x40c5a0 <_Unwind_RaiseException>
   0x0000000000400f11 <+81>:    mov    %rbx,%rdi
   0x0000000000400f14 <+84>:    callq  0x4019a0 <__cxa_begin_catch>
   0x0000000000400f19 <+89>:    callq  0x401ae0 <_ZSt9terminatev>
End of assembler dump.
```

Exercise A12

- **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
- \ALCDA-Dumps\Exercise-A12.pdf

© 2015 Software Diagnostics Services

Exercise A12

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.

1. Load a core dump *core.19138* and *App12* executable :

```
training@debian64:~/ALCDA$ gdb -c ./App12/core.19138 -se ./App12/App12
GNU gdb (GDB) 7.4.1-debian
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/training/ALCDA/App12/App12...(no debugging symbols found)...done.
[New LWP 19139]
[New LWP 19140]
[New LWP 19142]
[New LWP 19143]
[New LWP 19144]
[New LWP 19138]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Core was generated by `/home/training/ALCDA/App12/App12'.
#0  0x000000000043e4f1 in nanosleep ()
```

2. List all thread stack traces:

```
(gdb) thread apply all bt

Thread 6 (LWP 19138):
#0  0x000000000043e4f1 in nanosleep ()
#1  0x000000000043e3c0 in sleep ()
#2  0x0000000000400789 in main ()

Thread 5 (LWP 19144):
#0  0x000000000043e4f1 in nanosleep ()
#1  0x000000000043e3c0 in sleep ()
#2  0x000000000040069c in bar_five() ()
#3  0x00000000004006a7 in foo_five() ()
#4  0x00000000004006ba in thread_five(void*) ()
#5  0x000000000040f560 in start_thread ()
#6  0x0000000000440b49 in clone ()
#7  0x0000000000000000 in ?? ()

Thread 4 (LWP 19143):
#0  0x0000000000411eac in __lll_lock_wait ()
#1  0x0000000000410fa3 in _L_lock_926 ()
#2  0x0000000000410ddb in pthread_mutex_lock ()
#3  0x00000000004005ac in procB() ()
#4  0x0000000000400669 in bar_four() ()
```

```
#5  0x0000000000400674 in foo_four() ()
#6  0x0000000000400687 in thread_four(void*) ()
---Type <return> to continue, or q <return> to quit---
#7  0x000000000040f560 in start_thread ()
#8  0x0000000000440b49 in clone ()
#9  0x0000000000000000 in ?? ()

Thread 3 (LWP 19142):
#0  0x000000000043e4f1 in nanosleep ()
#1  0x000000000043e3c0 in sleep ()
#2  0x000000000040063b in bar_three() ()
#3  0x0000000000400646 in foo_three() ()
#4  0x0000000000400659 in thread_three(void*) ()
#5  0x000000000040f560 in start_thread ()
#6  0x0000000000440b49 in clone ()
#7  0x0000000000000000 in ?? ()

Thread 2 (LWP 19140):
#0  0x0000000000411eac in __lll_lock_wait ()
#1  0x0000000000410fa3 in _L_lock_926 ()
#2  0x0000000000410ddb in pthread_mutex_lock ()
#3  0x0000000000400577 in procA() ()
#4  0x0000000000400608 in bar_two() ()
#5  0x0000000000400613 in foo_two() ()
#6  0x0000000000400626 in thread_two(void*) ()
#7  0x000000000040f560 in start_thread ()
#8  0x0000000000440b49 in clone ()
---Type <return> to continue, or q <return> to quit---
#9  0x0000000000000000 in ?? ()

Thread 1 (LWP 19139):
#0  0x000000000043e4f1 in nanosleep ()
#1  0x000000000043e3c0 in sleep ()
#2  0x00000000004005da in bar_one() ()
#3  0x00000000004005e5 in foo_one() ()
#4  0x00000000004005f8 in thread_one(void*) ()
#5  0x000000000040f560 in start_thread ()
#6  0x0000000000440b49 in clone ()
#7  0x0000000000000000 in ?? ()
```

3. *App12* is an executable with stripped off debug symbols. Change the symbol file to *App12.debug* which is the same executable as *App12* but with debug symbols included:

```
(gdb) symbol-file ./App12/App12.debug
Reading symbols from /home/training/ALCDA/App12/App12.debug...done.
```

4. List all thread stack traces:

```
(gdb) thread apply all bt

Thread 6 (LWP 19138):
#0  0x000000000043e4f1 in nanosleep ()
#1  0x000000000043e3c0 in sleep ()
#2  0x0000000000400789 in main (argc=1, argv=0x7fff5d1572d8) at main.cpp:85

Thread 5 (LWP 19144):
#0  0x000000000043e4f1 in nanosleep ()
#1  0x000000000043e3c0 in sleep ()
#2  0x000000000040069c in bar_five () at main.cpp:69
#3  0x00000000004006a7 in foo_five () at main.cpp:69
```

```
#4  0x00000000004006ba in thread_five (arg=0x0) at main.cpp:69
#5  0x000000000040f560 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#6  0x0000000000440b49 in clone ()
#7  0x0000000000000000 in ?? ()

Thread 4 (LWP 19143):
#0  __lll_lock_wait ()
    at ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S:136
#1  0x0000000000410fa3 in _L_lock_926 ()
#2  0x0000000000410ddb in __pthread_mutex_lock (mutex=0x6c5900)
    at pthread_mutex_lock.c:61
#3  0x00000000004005ac in procB () at main.cpp:42
---Type <return> to continue, or q <return> to quit---
#4  0x0000000000400669 in bar_four () at main.cpp:68
#5  0x0000000000400674 in foo_four () at main.cpp:68
#6  0x0000000000400687 in thread_four (arg=0x0) at main.cpp:68
#7  0x000000000040f560 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#8  0x0000000000440b49 in clone ()
#9  0x0000000000000000 in ?? ()

Thread 3 (LWP 19142):
#0  0x000000000043e4f1 in nanosleep ()
#1  0x000000000043e3c0 in sleep ()
#2  0x000000000040063b in bar_three () at main.cpp:67
#3  0x0000000000400646 in foo_three () at main.cpp:67
#4  0x0000000000400659 in thread_three (arg=0x0) at main.cpp:67
#5  0x000000000040f560 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#6  0x0000000000440b49 in clone ()
#7  0x0000000000000000 in ?? ()

Thread 2 (LWP 19140):
#0  __lll_lock_wait ()
    at ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S:136
#1  0x0000000000410fa3 in _L_lock_926 ()
#2  0x0000000000410ddb in __pthread_mutex_lock (mutex=0x6c5940)
---Type <return> to continue, or q <return> to quit---
    at pthread_mutex_lock.c:61
#3  0x0000000000400577 in procA () at main.cpp:35
#4  0x0000000000400608 in bar_two () at main.cpp:66
#5  0x0000000000400613 in foo_two () at main.cpp:66
#6  0x0000000000400626 in thread_two (arg=0x0) at main.cpp:66
#7  0x000000000040f560 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#8  0x0000000000440b49 in clone ()
#9  0x0000000000000000 in ?? ()

Thread 1 (LWP 19139):
#0  0x000000000043e4f1 in nanosleep ()
#1  0x000000000043e3c0 in sleep ()
#2  0x00000000004005da in bar_one () at main.cpp:65
#3  0x00000000004005e5 in foo_one () at main.cpp:65
#4  0x00000000004005f8 in thread_one (arg=0x0) at main.cpp:65
#5  0x000000000040f560 in start_thread (arg=<optimized out>)
    at pthread_create.c:304
#6  0x0000000000440b49 in clone ()
#7  0x0000000000000000 in ?? ()
```

5. Switch to the thread #6 and its frame #2, and list arguments and locals:

```
(gdb) thread 6
[Switching to thread 6 (LWP 19138)]
#0  0x000000000043e4f1 in nanosleep ()

(gdb) bt
#0  0x000000000043e4f1 in nanosleep ()
#1  0x000000000043e3c0 in sleep ()
#2  0x0000000000400789 in main (argc=1, argv=0x7fff5d1572d8) at main.cpp:85

(gdb) frame 2
#2  0x0000000000400789 in main (argc=1, argv=0x7fff5d1572d8) at main.cpp:85
85         sleep(-1);

(gdb) info args
argc = 1
argv = 0x7fff5d1572d8

(gdb) info locals
No locals.
```

6. Examine *argv* array:

```
(gdb) print argv[0]
$1 = 0x7fff5d1579a3 "./App12"

(gdb) print *argv@10
$2 = {0x7fff5d1579a3 "./App12", 0x0, 0x7fff5d1579ab "SHELL=/bin/bash",
  0x7fff5d1579bb "TERM=linux", 0x7fff5d1579c6 "HUSHLOGIN=FALSE",
  0x7fff5d1579d6 "USER=training",
  0x7fff5d1579e4
"LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=4
0;31;01:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:
*.arj=01;31"...,
  0x7fff5d157f05 "MAIL=/var/mail/training",
  0x7fff5d157f1d "PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games",
0x7fff5d157f5b "PWD=/home/training/ALCDA/App12"}
```

7. Dump the region 0x8b8000 - 0x8db000 to a binary file:

```
(gdb) dump memory ./App12/mem.raw 0x8b8000 0x8db000
(gdb) q
```

8. List all functions:

```
(gdb) info functions
All defined functions:

File pthread_mutex_init.c:
int __pthread_mutex_init(pthread_mutex_t *, const pthread_mutexattr_t *);

File pthread_mutex_trylock.c:
int __pthread_mutex_trylock(pthread_mutex_t *);

File pthread_mutex_unlock.c:
int __pthread_mutex_unlock(pthread_mutex_t *);
int __pthread_mutex_unlock_usercnt(pthread_mutex_t *, int);
static int __pthread_mutex_unlock_full(pthread_mutex_t *, int);
```

```
File pthread_key_create.c:
int __pthread_key_create(pthread_key_t *, void (*)(void *));

File pthread_key_delete.c:
int pthread_key_delete(pthread_key_t);

File pthread_getspecific.c:
void *__pthread_getspecific(pthread_key_t);

File pthread_setspecific.c:
int __pthread_setspecific(pthread_key_t, const void *);
---Type <return> to continue, or q <return> to quit---

File pthread_cancel.c:
int pthread_cancel(pthread_t);

File tpp.c:
void __init_sched_fifo_prio(void);
int __pthread_current_priority(void);
int __pthread_tpp_change_priority(int, int);

File nptl-init.c:
void __pthread_initialize_minimal_internal(void);
static void sigcancel_handler(int, siginfo_t *, void *);
static void sighandler_setxid(int, siginfo_t *, void *);

File events.c:
void __nptl_create_event(void);
void __nptl_death_event(void);

File unwind.c:
void __pthread_unwind(__pthread_unwind_buf_t *);
void __pthread_unwind_next(__pthread_unwind_buf_t *);
static void unwind_cleanup(_Unwind_Reason_Code, struct _Unwind_Exception *);
static _Unwind_Reason_Code unwind_stop(int, _Unwind_Action,
    _Unwind_Exception_Class, struct _Unwind_Exception *, struct _Unwind_Context
---Type <return> to continue, or q <return> to quit---
     *, void *);

File ../sysdeps/unix/sysv/linux/x86_64/sigaction.c:
int __libc_sigaction(int, const struct sigaction *, struct sigaction *);

File ../nptl/sigaction.c:
int __sigaction(int, const struct sigaction *, struct sigaction *);

File pthread_mutex_lock.c:
int __pthread_mutex_lock(pthread_mutex_t *);
static int __pthread_mutex_lock_full(pthread_mutex_t *);

File allocatestack.c:
void __deallocate_stack(struct pthread *);

File pthread_create.c:
struct pthread *__find_in_stack_list(struct pthread *);

File allocatestack.c:
struct pthread *__find_thread_by_id(pid_t);
void __free_stacks(size_t);

File pthread_create.c:
```

```
void __free_tcb(struct pthread *);
---Type <return> to continue, or q <return> to quit---

File allocatestack.c:
int __make_stacks_executable(void **);

File pthread_create.c:
void __nptl_deallocate_tsd(void);

File allocatestack.c:
int __nptl_setxid(struct xid_command *);

File pthread_create.c:
int __pthread_create_2_1(pthread_t *, const pthread_attr_t *, void *(*)(
    void *), void *);

File allocatestack.c:
void __pthread_init_static_tls(struct link_map *);
void __reclaim_stacks(void);
void __wait_lookup_done(void);

File ../nptl/sysdeps/pthread/createthread.c:
static int do_clone(struct pthread *, const struct pthread_attr *, void *,
    int, int (*)(void *), int);

File allocatestack.c:
---Type <return> to continue, or q <return> to quit---
static void setxid_mark_thread(struct xid_command *, struct pthread *);

File pthread_create.c:
static int start_thread(void *);

File main.cpp:
void bar_five();
void bar_four();
void bar_one();
void bar_three();
void bar_two();
void foo_five();
void foo_four();
void foo_one();
void foo_three();
void foo_two();
int main(int, char const**);
void procA();
void procB();
void procC();
void *thread_five(void*);
void *thread_four(void*);
void *thread_one(void*);
void *thread_three(void*);
---Type <return> to continue, or q <return> to quit---
void *thread_two(void*);

Non-debugging symbols:
0x00007fff5d19d970  __vdso_clock_gettime
0x00007fff5d19d970  clock_gettime
0x00007fff5d19d9f0  __vdso_gettimeofday
0x00007fff5d19d9f0  gettimeofday
0x00007fff5d19da80  __vdso_time
```

```
0x00007fff5d19da80  time
0x00007fff5d19daa0  __vdso_getcpu
0x00007fff5d19daa0  getcpu
0x00000000004002d8  _init
0x00000000004003c0  _GLOBAL__sub_I_eh_alloc.cc
0x00000000004003f8  _start
0x0000000000400424  call_gmon_start
0x0000000000400440  deregister_tm_clones
0x0000000000400470  register_tm_clones
0x00000000004004b0  __do_global_dtors_aux
0x00000000004004e0  frame_dummy
0x0000000000400790  __cxxabiv1::__fundamental_type_info::~__fundamental_type_info()
0x0000000000400790  __cxxabiv1::__fundamental_type_info::~__fundamental_type_info()
0x00000000004007b0  __cxxabiv1::__fundamental_type_info::~__fundamental_type_inf---Type
<return> to continue, or q <return> to quit---
o()
0x00000000004007d0  std::type_info::~type_info()
0x00000000004007d0  std::type_info::~type_info()
0x00000000004007e0  std::type_info::__is_pointer_p() const
0x00000000004007f0  std::type_info::__is_function_p() const
0x0000000000400800  std::type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void**)
const
0x0000000000400810  std::type_info::~type_info()
0x0000000000400830  std::type_info::__do_catch(std::type_info const*, void**, unsigned int)
const
0x0000000000400880  __gnu_cxx::__concurrence_lock_error::what() const
0x0000000000400890  __gnu_cxx::__concurrence_unlock_error::what() const
0x00000000004008a0  __gnu_cxx::__concurrence_lock_error::~__concurrence_lock_error()
0x00000000004008a0  __gnu_cxx::__concurrence_lock_error::~__concurrence_lock_error()
0x00000000004008c0  __gnu_cxx::__concurrence_unlock_error::~__concurrence_unlock_error()
0x00000000004008c0  __gnu_cxx::__concurrence_unlock_error::~__concurrence_unlock_error()
0x00000000004008e0  __gnu_cxx::__concurrence_lock_error::~__concurrence_lock_error()
0x0000000000400900  __gnu_cxx::__concurrence_unlock_error::~__concurrence_unlock_error()
---Type <return> to continue, or q <return> to quit---q
Quit
```

9. List all variables:

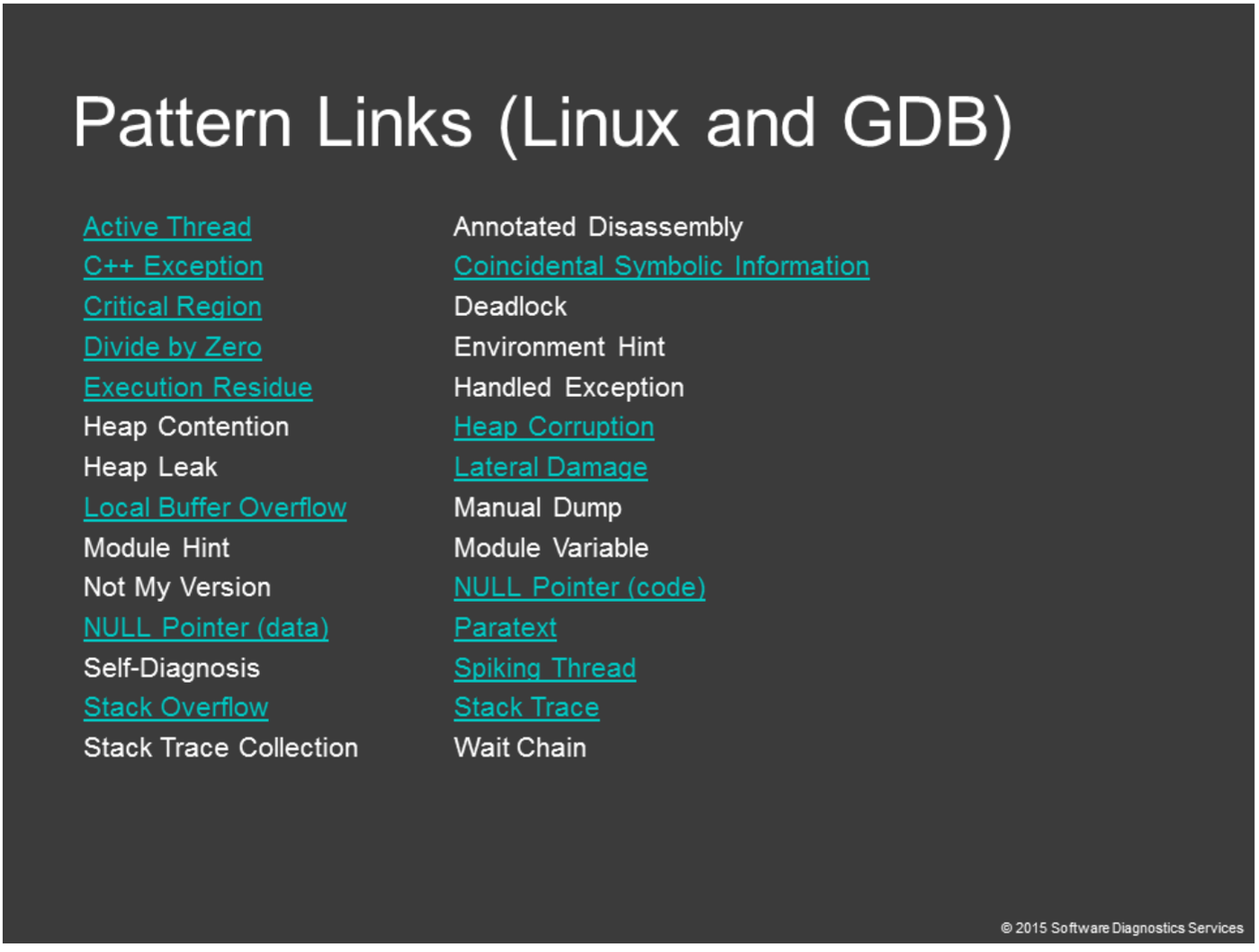
```
(gdb) info variables
All defined variables:

File vars.c:
size_t __default_stacksize;
int __is_smp;
struct pthread_key_struct __pthread_keys[1024];
int __pthread_multiple_threads;

File pthread_mutex_init.c:
static const struct pthread_mutexattr default_attr;

File tpp.c:
int __sched_fifo_max_prio;
int __sched_fifo_min_prio;

File nptl-init.c:
int __have_futex_clock_realtime;
size_t __static_tls_align_m1;
size_t __static_tls_size;
struct xid_command *__xidcmd;
static _Bool __nptl_initial_report_events;
```

```
static const char nptl_version[5];

File ../nptl/sysdeps/pthread/createthread.c:
---Type <return> to continue, or q <return> to quit---
int *__libc_multiple_threads_ptr;

File pthread_create.c:
unsigned int __nptl_nthreads;
int __pthread_debug;

File allocatestack.c:
list_t __stack_user;

File ../nptl_db/structs.def:
const uint32_t _thread_db___nptl_initial_report_events[3];
const uint32_t _thread_db___nptl_last_event[3];
const uint32_t _thread_db___nptl_nthreads[3];
const uint32_t _thread_db___pthread_keys[3];

File ../nptl_db/db_info.c:
const uint32_t _thread_db_const_thread_area;

File ../nptl_db/structs.def:
const uint32_t _thread_db_dtv_dtv[3];
const uint32_t _thread_db_dtv_t_pointer_val[3];
const uint32_t _thread_db_link_map_l_tls_modid[3];
const uint32_t _thread_db_list_t_next[3];
const uint32_t _thread_db_list_t_prev[3];
---Type <return> to continue, or q <return> to quit---
const uint32_t _thread_db_pthread_cancelhandling[3];
const uint32_t _thread_db_pthread_dtvp[3];
const uint32_t _thread_db_pthread_eventbuf[3];
const uint32_t _thread_db_pthread_eventbuf_eventmask[3];
const uint32_t _thread_db_pthread_eventbuf_eventmask_event_bits[3];
const uint32_t _thread_db_pthread_key_data_data[3];
const uint32_t _thread_db_pthread_key_data_level2_data[3];
const uint32_t _thread_db_pthread_key_data_seq[3];
const uint32_t _thread_db_pthread_key_struct_destr[3];
const uint32_t _thread_db_pthread_key_struct_seq[3];
const uint32_t _thread_db_pthread_list[3];
const uint32_t _thread_db_pthread_nextevent[3];
const uint32_t _thread_db_pthread_pid[3];
const uint32_t _thread_db_pthread_report_events[3];
const uint32_t _thread_db_pthread_schedparam_sched_priority[3];
const uint32_t _thread_db_pthread_schedpolicy[3];
const uint32_t _thread_db_pthread_specific[3];
const uint32_t _thread_db_pthread_start_routine[3];
const uint32_t _thread_db_pthread_tid[3];
const uint32_t _thread_db_sizeof_list_t;
const uint32_t _thread_db_sizeof_pthread;
const uint32_t _thread_db_sizeof_pthread_key_data;
const uint32_t _thread_db_sizeof_pthread_key_data_level2;
const uint32_t _thread_db_sizeof_pthread_key_struct;
---Type <return> to continue, or q <return> to quit---
const uint32_t _thread_db_sizeof_td_eventbuf_t;
const uint32_t _thread_db_sizeof_td_thr_events_t;
const uint32_t _thread_db_td_eventbuf_t_eventdata[3];
const uint32_t _thread_db_td_eventbuf_t_eventnum[3];
const uint32_t _thread_db_td_thr_events_t_event_bits[3];
```

```
File pthread_create.c:
static struct pthread *__nptl_last_event;
static td_thr_events_t __nptl_threads_events;
static const struct pthread_attr default_attr;

File allocatestack.c:
static uintptr_t in_flight_stack;
static list_t stack_cache;
static size_t stack_cache_actsize;
static int stack_cache_lock;
static const size_t stack_cache_maxsize;
static list_t stack_used;

File main.cpp:
pthread_mutex_t mutexA;
pthread_mutex_t mutexB;

Non-debugging symbols:
---Type <return> to continue, or q <return> to quit---
0x0000000000000000  __libc_resp
0x0000000000000008  __libc_tsd_LOCALE
0x0000000000000010  _nl_current_LC_CTYPE
0x0000000000000018  _nl_current_LC_MONETARY
0x0000000000000020  _nl_current_LC_NUMERIC
0x0000000000000028  (anonymous namespace)::get_global()::global
0x0000000000000038  __libc_errno
0x0000000000000040  __libc_tsd_MALLOC
0x0000000000000048  __libc_tsd_CTYPE_B
0x0000000000000050  __libc_tsd_CTYPE_TOUPPER
0x0000000000000058  __libc_tsd_CTYPE_TOLOWER
0x0000000000000060  data.11299
0x00000000004001a0  __rela_iplt_start
0x00000000004002d8  __rela_iplt_end
0x0000000000495c20  _IO_stdin_used
0x0000000000495c40  typeinfo name for __cxxabiv1::__fundamental_type_info
0x0000000000495c68  typeinfo name for void
0x0000000000495c6a  typeinfo name for void*
0x0000000000495c6d  typeinfo name for void const*
0x0000000000495c71  typeinfo name for bool
0x0000000000495c73  typeinfo name for bool*
0x0000000000495c76  typeinfo name for bool const*
0x0000000000495c7a  typeinfo name for wchar_t
0x0000000000495c7c  typeinfo name for wchar_t*
---Type <return> to continue, or q <return> to quit---q
Quit
```

10. List segment info:

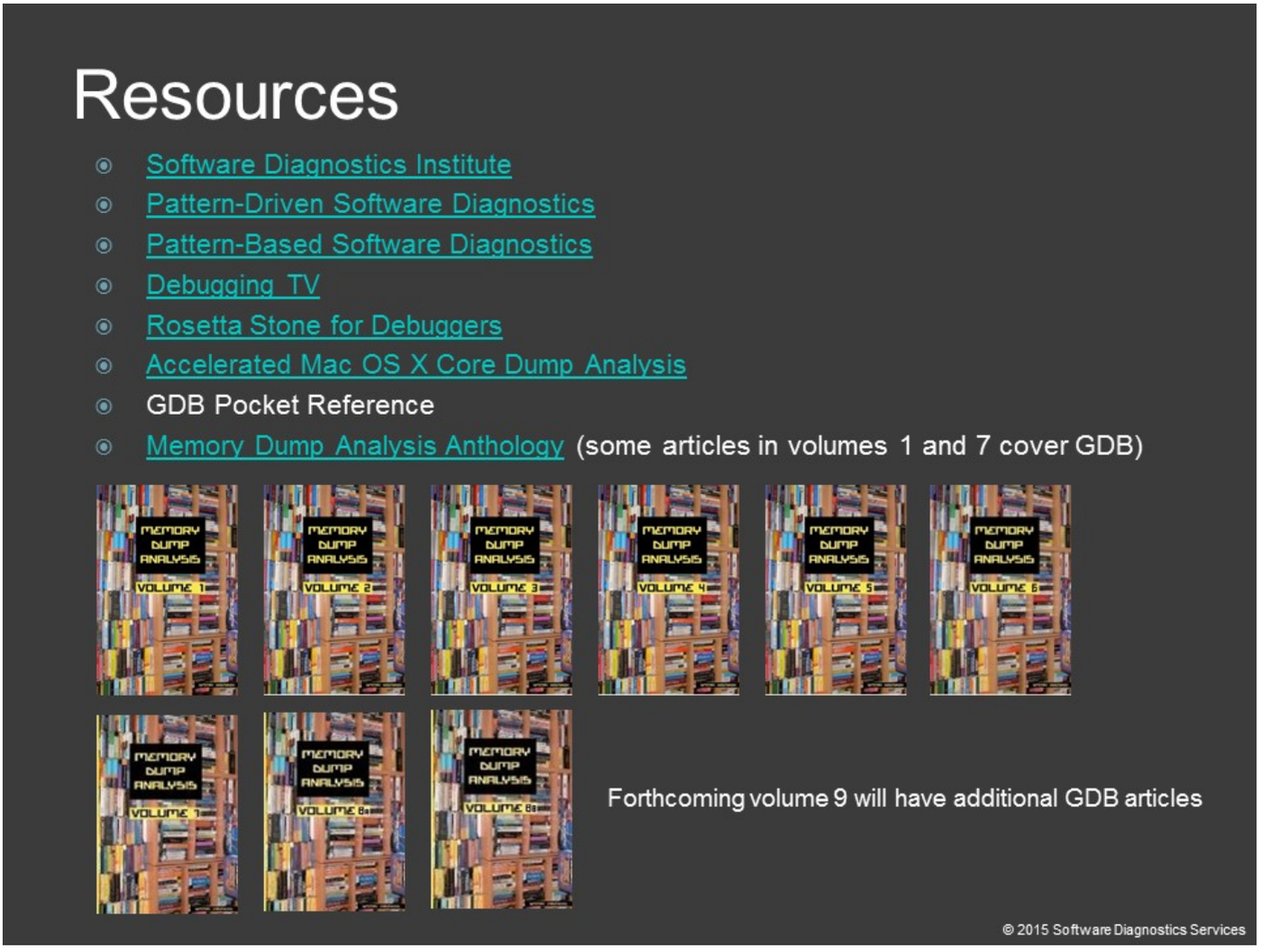
```
(gdb) info target
Symbols from "/home/training/ALCDA/App12/App12.debug".
Local core dump file:
      `/home/training/ALCDA/./App12/core.19138', file type elf64-x86-64.
      0x0000000000400000 - 0x0000000000400000 is load1
      0x00000000006c3000 - 0x00000000006c6000 is load2
      0x00000000006c6000 - 0x00000000006df000 is load3
      0x00000000008b8000 - 0x00000000008db000 is load4
      0x00007f1bed271000 - 0x00007f1beda71000 is load5
      0x00007f1beda72000 - 0x00007f1bee272000 is load6
      0x00007f1bee273000 - 0x00007f1beea73000 is load7
      0x00007f1beea74000 - 0x00007f1bef274000 is load8
      0x00007f1bef275000 - 0x00007f1befa75000 is load9
      0x00007fff5d137000 - 0x00007fff5d158000 is load10
      0x00007fff5d19d000 - 0x00007fff5d19e000 is load11
      0xffffffffff600000 - 0xffffffffff601000 is load12
Local exec file:
      `/home/training/ALCDA/App12/App12', file type elf64-x86-64.
      Entry point: 0x4003f8
      0x0000000000400158 - 0x0000000000400178 is .note.ABI-tag
      0x0000000000400178 - 0x000000000040019c is .note.gnu.build-id
      0x00000000004001a0 - 0x00000000004002d8 is .rela.plt
      0x00000000004002d8 - 0x00000000004002e6 is .init
      0x00000000004002f0 - 0x00000000004003c0 is .plt
      0x00000000004003c0 - 0x0000000000495014 is .text
      0x0000000000495020 - 0x0000000000495b9e is __libc_freeres_fn
      0x0000000000495ba0 - 0x0000000000495c01 is __libc_thread_freeres_fn
      0x0000000000495c04 - 0x0000000000495c0d is .fini
---Type <return> to continue, or q <return> to quit---
      0x0000000000495c20 - 0x00000000004b4d74 is .rodata
      0x00000000004b4d78 - 0x00000000004b4dd8 is __libc_subfreeres
      0x00000000004b4dd8 - 0x00000000004b4de0 is __libc_atexit
      0x00000000004b4de0 - 0x00000000004b4de8 is __libc_thread_subfreeres
      0x00000000004b4de8 - 0x00000000004c2c4c is .eh_frame
      0x00000000004c2c4c - 0x00000000004c2e7d is .gcc_except_table
      0x00000000006c3000 - 0x00000000006c3028 is .tdata
      0x00000000006c3028 - 0x00000000006c3068 is .tbss
      0x00000000006c3028 - 0x00000000006c3040 is .init_array
      0x00000000006c3040 - 0x00000000006c3050 is .fini_array
      0x00000000006c3050 - 0x00000000006c3058 is .jcr
      0x00000000006c3060 - 0x00000000006c4a00 is .data.rel.ro
      0x00000000006c4a00 - 0x00000000006c4ad0 is .got
      0x00000000006c4ad0 - 0x00000000006c4b50 is .got.plt
      0x00000000006c4b60 - 0x00000000006c5888 is .data
      0x00000000006c58a0 - 0x00000000006de908 is .bss
      0x00000000006de908 - 0x00000000006de938 is __libc_freeres_ptrs
```



Here is the link to pattern descriptions and additional GDB examples:

http://www.dumpanalysis.org/blog/index.php/category/core-dump-analysis/

Selected pattern descriptions are provided at the end of this book.



Software Diagnostics Institute:
http://www.dumpanalysis.org/

Pattern-Driven Software Diagnostics:
http://www.patterndiagnostics.com/Introduction-Software-Diagnostics-materials

Pattern-Based Software Diagnostics:
http://www.patterndiagnostics.com/pattern-based-diagnostics-materials

Debugging TV:
http://www.debugging.tv/

Rosetta Stone for Debuggers:
http://www.dumpanalysis.org/rosetta-stone-debuggers

Accelerated Mac OS X Core Dump Analysis:
http://www.patterndiagnostics.com/accelerated-macosx-core-dump-analysis-book

Memory Dump Analysis Anthology:
http://www.patterndiagnostics.com/ultimate-memory-analysis-reference

App Source Code

App0

```
//
//  main.c
//  App0 - Exercise 0 - Testing Linux GDB
//
//  Copyright (c) 2015 Software Diagnostics Services. All rights reserved.
//

#include <stdlib.h>

void bar()
{
    abort();
}

void foo()
{
    bar();
}

int main(int argc, const char * argv[])
{
    foo();
    return 0;
}
```

App1

```
//
//  main.c
//  App1 - Normal application with multiple threads
//
//  Copyright (c) 2015 Software Diagnostics Services. All rights reserved.
//

#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>

#define THREAD_DECLARE(num) void bar_##num()\
{\
    sleep(-1);\
}\
\
void foo_##num()\
{\
    bar_##num();\
}\
\
void * thread_##num (void *arg)\
{\
    foo_##num();\
\
    return 0;\
}

THREAD_DECLARE(one)
THREAD_DECLARE(two)
THREAD_DECLARE(three)
THREAD_DECLARE(four)
THREAD_DECLARE(five)

#define THREAD_CREATE(num) {pthread_t threadID_##num; pthread_create (&threadID_##num, NULL,
thread_##num, NULL);}

int main(int argc, const char * argv[])
{
    THREAD_CREATE(one)
    THREAD_CREATE(two)
    THREAD_CREATE(three)
    THREAD_CREATE(four)
    THREAD_CREATE(five)

    sleep(-1);
    return 0;
}
```

App2D

```
//
//  main.c
//  App2D - Shows NULL data pointer exception
//
//  Copyright (c) 2015 Software Diagnostics Services. All rights reserved.
//

#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>

void procA()
{
    int *p = NULL;

       *p = 1;
}

void procB()
{
       sleep(1);

       void (*pf)() = NULL;

    pf();
}

#define THREAD_DECLARE(num,func) void bar_##num()\
{\
func;\
}\
\
void foo_##num()\
{\
bar_##num();\
}\
\
void * thread_##num (void *arg)\
{\
foo_##num();\
\
return 0;\
}

THREAD_DECLARE(one,sleep(-1))
THREAD_DECLARE(two,procA())
THREAD_DECLARE(three,sleep(-1))
THREAD_DECLARE(four,procB())
THREAD_DECLARE(five,sleep(-1))

#define THREAD_CREATE(num) {pthread_t threadID_##num; pthread_create (&threadID_##num, NULL,
thread_##num, NULL);}
```

```
int main(int argc, const char * argv[])
{
    THREAD_CREATE(one)
    THREAD_CREATE(two)
    THREAD_CREATE(three)
    THREAD_CREATE(four)
    THREAD_CREATE(five)

    sleep(3);
    return 0;
}
```

App2C

```
//
//  main.c
//  App2C - Shows NULL code pointer exception
//
//  Copyright (c) 2015 Software Diagnostics Services. All rights reserved.
//

#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>

void procA()
{
    sleep(1);

       int *p = NULL;

       *p = 1;
}

void procB()
{
       void (*pf)() = NULL;

    pf();
}

#define THREAD_DECLARE(num,func) void bar_##num()\
{\
func;\
}\
\
void foo_##num()\
{\
bar_##num();\
}\
\
void * thread_##num (void *arg)\
{\
foo_##num();\
\
return 0;\
}

THREAD_DECLARE(one,sleep(-1))
THREAD_DECLARE(two,procA())
THREAD_DECLARE(three,sleep(-1))
THREAD_DECLARE(four,procB())
THREAD_DECLARE(five,sleep(-1))

#define THREAD_CREATE(num) {pthread_t threadID_##num; pthread_create (&threadID_##num, NULL,
thread_##num, NULL);}
```

```
int main(int argc, const char * argv[])
{
    THREAD_CREATE(one)
    THREAD_CREATE(two)
    THREAD_CREATE(three)
    THREAD_CREATE(four)
    THREAD_CREATE(five)

    sleep(3);
    return 0;
}
```

App3

```
//
//  main.c
//  App3 - Spiking Thread pattern
//
//  Copyright (c) 2015 Software Diagnostics Services. All rights reserved.
//

#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>

void procA()
{
    while (1)
    {
        sleep(1);
    }
}

void procB()
{
    double d = 1.0/3.0;
    while (1)
    {
        d = sqrt(d);
    }
}

#define THREAD_DECLARE(num,func) void bar_##num()\
{\
func;\
}\
\
void foo_##num()\
{\
bar_##num();\
}\
\
void * thread_##num (void *arg)\
{\
foo_##num();\
\
return 0;\
}

THREAD_DECLARE(one,sleep(-1))
THREAD_DECLARE(two,sleep(-1))
THREAD_DECLARE(three,procA())
THREAD_DECLARE(four,sleep(-1))
THREAD_DECLARE(five,procB())

#define THREAD_CREATE(num) {pthread_t threadID_##num; pthread_create (&threadID_##num, NULL,
thread_##num, NULL);}
```

```c
int main(int argc, const char * argv[])
{
    THREAD_CREATE(one)
    THREAD_CREATE(two)
    THREAD_CREATE(three)
    THREAD_CREATE(four)
    THREAD_CREATE(five)

    sleep(-1);
    return 0;
}
```

App4

```
//
//  main.c
//  App4 - Heap Corruption pattern
//
//  Copyright (c) 2015 Software Diagnostics Services. All rights reserved.
//

#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>

void proc()
{
    char *p1 = (char *) malloc (1024);
    char *p2 = (char *) malloc (1024);
    char *p3 = (char *) malloc (1024);
    char *p4 = (char *) malloc (1024);
    char *p5 = (char *) malloc (1024);
    char *p6 = (char *) malloc (1024);
    char *p7 = (char *) malloc (1024);

    free(p6);
    free(p4);
    free(p2);

    strcpy(p2, "Hello Crash!");
    strcpy(p4, "Hello Crash!");
    strcpy(p6, "Hello Crash!");

    p2 = (char *) malloc (512);
    p4 = (char *) malloc (1024);
    p6 = (char *) malloc (512);

    sleep(300);

    free (p7);
    free (p6);
    free (p5);
    free (p4);
    free (p3);
    free (p2);
    free (p1);

    sleep(-1);
}
```

```
#define THREAD_DECLARE(num,func) void bar_##num()\
{\
func;\
}\
\
void foo_##num()\
{\
bar_##num();\
}\
\
void * thread_##num (void *arg)\
{\
foo_##num();\
\
return 0;\
}

THREAD_DECLARE(one,sleep(-1))
THREAD_DECLARE(two,sleep(-1))
THREAD_DECLARE(three,proc())
THREAD_DECLARE(four,sleep(-1))
THREAD_DECLARE(five,sleep(-1))

#define THREAD_CREATE(num) {pthread_t threadID_##num; pthread_create (&threadID_##num, NULL,
thread_##num, NULL);}

int main(int argc, const char * argv[])
{
    THREAD_CREATE(one)
    THREAD_CREATE(two)
    THREAD_CREATE(three)
    THREAD_CREATE(four)
    THREAD_CREATE(five)

    sleep(-1);
    return 0;
}
```

App5

```
//
//  main.c
//  App5 - Local Buffer Overflow
//
//  Copyright (c) 2015 Software Diagnostics Services. All rights reserved.
//

#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>

void procB(char *buffer)
{
    char data[100] = "My New Bigger Buffer";
    memcpy (buffer, data, sizeof(data));
}

void procA()
{
    char data[10] = "My Buffer";
    procB(data);
}

#define THREAD_DECLARE(num,func) void bar_##num()\
{\
func;\
}\
\
void foo_##num()\
{\
bar_##num();\
}\
\
void * thread_##num (void *arg)\
{\
foo_##num();\
\
return 0;\
}

THREAD_DECLARE(one,procA())
THREAD_DECLARE(two,sleep(-1))
THREAD_DECLARE(three,sleep(-1))
THREAD_DECLARE(four,sleep(-1))
THREAD_DECLARE(five,sleep(-1))

#define THREAD_CREATE(num) {pthread_t threadID_##num; pthread_create (&threadID_##num, NULL,
thread_##num, NULL);}
```

```
int main(int argc, const char * argv[])
{
    THREAD_CREATE(one)
    THREAD_CREATE(two)
    THREAD_CREATE(three)
    THREAD_CREATE(four)
    THREAD_CREATE(five)

    sleep(-1);
    return 0;
}
```

App6

```
//
//  main.c
//  App6 - Stack Overflow
//
//  Copyright (c) 2015 Software Diagnostics Services. All rights reserved.
//

#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>

void procF(int i)
{
    int buffer[128] = {-1, 0, i+1, 0, -1};

    procF(buffer[2]);
}

void procE()
{
    procF(1);
}

#define THREAD_DECLARE(num,func) void bar_##num()\
{\
sleep(300);\
func;\
}\
\
void foo_##num()\
{\
bar_##num();\
}\
\
void * thread_##num (void *arg)\
{\
foo_##num();\
\
return 0;\
}

THREAD_DECLARE(one,procE())
THREAD_DECLARE(two,sleep(-1))
THREAD_DECLARE(three,sleep(-1))
THREAD_DECLARE(four,sleep(-1))
THREAD_DECLARE(five,sleep(-1))

#define THREAD_CREATE(num) {pthread_t threadID_##num; pthread_create (&threadID_##num, NULL,
thread_##num, NULL);}
```

```c
int main(int argc, const char * argv[])
{
    THREAD_CREATE(one)
    THREAD_CREATE(two)
    THREAD_CREATE(three)
    THREAD_CREATE(four)
    THREAD_CREATE(five)

    sleep(-1);
    return 0;
}
```

App7

```
//
//  main.c
//  App7 - Divide by Zero and Active Threads
//
//  Copyright (c) 2015 Software Diagnostics Services. All rights reserved.
//

#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>

void procF(int i)
{
    int buffer[1024] = {-1, 0, i+1, 0, -1};

    procF(buffer[2]);
}

void procE()
{
    procF(1);
}

int procD(int a, int b)
{
    return a/b;
}

int procC()
{
    return procD(1,0);
}

void procB(char *buffer)
{
    char data[100] = "My New Bigger Buffer";
    memcpy (buffer, data, sizeof(data));
}

void procA()
{
    char data[10] = "My Buffer";
    procB(data);
}
```

```
#define THREAD_DECLARE(num,func) void bar_##num()\
{\
sleep(300);\
func;\
}\
\
void foo_##num()\
{\
bar_##num();\
}\
\
void * thread_##num (void *arg)\
{\
foo_##num();\
\
return 0;\
}

THREAD_DECLARE(one,procA())
THREAD_DECLARE(two,sleep(-1))
THREAD_DECLARE(three,procC())
THREAD_DECLARE(four,sleep(-1))
THREAD_DECLARE(five,procE())

#define THREAD_CREATE(num) {pthread_t threadID_##num; pthread_create (&threadID_##num, NULL,
thread_##num, NULL);}

int main(int argc, const char * argv[])
{
    THREAD_CREATE(one)
    THREAD_CREATE(two)
    THREAD_CREATE(three)
    THREAD_CREATE(four)
    THREAD_CREATE(five)

    sleep(-1);
    return 0;
}
```

App8

```
//
//  main.cpp
//  App8 - C++ Exception, Execution Residue, Handled Exception
//
//  Copyright (c) 2015 Software Diagnostics Services. All rights reserved.
//

#include <string>

#define def_call(name,x,y) void name##_##x() { name##_##y(); }
#define def_final(name,x) void name##_##x() { }
#define def_init(name,y,size) void name() { int arr[size]; name##_##y(); *arr=0; }

def_final(work,9)
def_call(work,8,9)
def_call(work,7,8)
def_call(work,6,7)
def_call(work,5,6)
def_call(work,4,5)
def_call(work,3,4)
def_call(work,2,3)
def_call(work,1,2)
def_init(work,1,256)

class Exception
{
    int code;
    std::string description;

public:
    Exception(int _code, std::string _desc) : code(_code), description(_desc) {}
};

void procB()
{
    throw new Exception(5, "Access Denied");
}

void procNB()
{
    work();
}

void procA()
{
    procB();
}

void procNA()
{
    procNB();
}
```

```
void procH()
{
    try {
        procA();
    } catch (...) {
        sleep(-1);
    }
}

void procNH()
{
    sleep(300);
    procA();
}

void procNE()
{
    try {
        procNA();
    }
    catch (...)
    {
    }
    sleep(-1);
}

#define THREAD_DECLARE(num,func) void bar_##num()\
{\
func;\
}\
\
void foo_##num()\
{\
bar_##num();\
}\
\
void * thread_##num (void *arg)\
{\
foo_##num();\
\
return 0;\
}

THREAD_DECLARE(one,procNH())
THREAD_DECLARE(two,procNE())
THREAD_DECLARE(three,procH())
THREAD_DECLARE(four,procNE())
THREAD_DECLARE(five,procNE())

#define THREAD_CREATE(num) {pthread_t threadID_##num; pthread_create (&threadID_##num, NULL,
thread_##num, NULL);}

int main(int argc, const char * argv[])
{
    THREAD_CREATE(one)
    THREAD_CREATE(two)
    THREAD_CREATE(three)
    THREAD_CREATE(four)
    THREAD_CREATE(five)

    sleep(-1);
    return 0;
}
```

App9

```
//
//  main.c
//  App9 - Heap Leak pattern
//
//  Copyright (c) 2015 Software Diagnostics Services. All rights reserved.
//

#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>

void procD()
{
}

typedef void (**PFUNC)();

void procC(int iter)
{
    for (int i = 0; i < iter; ++i)
    {
        char *p = malloc(256);
        strcpy(p, "allocated memory");

        *(PFUNC)(p + 32) = &procD;
    }
}

void procB()
{
    procC(250000);
    sleep(300);
    procC(250000);
    sleep(-1);
}

void procA()
{
    procC(5000);
    sleep(300);
    procB();
}
```

```
#define THREAD_DECLARE(num,func) void bar_##num()\
{\
func;\
}\
\
void foo_##num()\
{\
bar_##num();\
}\
\
void * thread_##num (void *arg)\
{\
foo_##num();\
\
return 0;\
}

THREAD_DECLARE(one,sleep(-1))
THREAD_DECLARE(two,procA())
THREAD_DECLARE(three,sleep(-1))
THREAD_DECLARE(four,sleep(-1))
THREAD_DECLARE(five,sleep(-1))

#define THREAD_CREATE(num) {pthread_t threadID_##num; pthread_create (&threadID_##num, NULL,
thread_##num, NULL);}

int main(int argc, const char * argv[])
{
    THREAD_CREATE(one)
    THREAD_CREATE(two)
    THREAD_CREATE(three)
    THREAD_CREATE(four)
    THREAD_CREATE(five)

    sleep(-1);
    return 0;
}
```

App10

```
//
//  main.c
//  App10 - Heap Corruption, Heap Contention, Critical Region, Wait Chains, Self-Diagnostics
patterns
//
//  Copyright (c) 2015 Software Diagnostics Services. All rights reserved.
//

#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>

#define ARR_SIZE 10000

char *pAllocBuf [ARR_SIZE] = {0};

void proc()
{
    while (1)
    {
        int idx = rand()%ARR_SIZE;
        int malloc_size = rand()%ARR_SIZE;

        if (pAllocBuf[idx])
        {
            free(pAllocBuf[idx]);
            pAllocBuf[idx] = 0;
        }

        pAllocBuf[idx] = malloc(malloc_size);
    }
}

#define THREAD_DECLARE(num,func) void bar_##num()\
{\
func;\
}\
\
void foo_##num()\
{\
bar_##num();\
}\
\
void * thread_##num (void *arg)\
{\
foo_##num();\
\
return 0;\
}

THREAD_DECLARE(one,proc())
THREAD_DECLARE(two,proc())
THREAD_DECLARE(three,proc())
THREAD_DECLARE(four,proc())
THREAD_DECLARE(five,proc())

#define THREAD_CREATE(num) {pthread_t threadID_##num; pthread_create (&threadID_##num, NULL,
thread_##num, NULL);}
```

```
int main(int argc, const char * argv[])
{
    THREAD_CREATE(one)
    THREAD_CREATE(two)
    THREAD_CREATE(three)
    THREAD_CREATE(four)
    THREAD_CREATE(five)

    sleep(-1);
    return 0;
}
```

App11 / App12

```
//
//  main.c
//  App11/App12 - Wait Chains, Deadlock, Handled Exception patterns
//
//  Copyright (c) 2015 Software Diagnostics Services. All rights reserved.
//

#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>

pthread_mutex_t mutexA, mutexB;

void procC()
{
    throw 0;
}

void procA()
{
    try
    {
        pthread_mutex_lock(&mutexA);
        procC();
        pthread_mutex_unlock(&mutexA);
    }
    catch(...)
    {

    }

    sleep(20);
    pthread_mutex_lock(&mutexB);
    pthread_mutex_unlock(&mutexB);
}

void procB()
{
    pthread_mutex_lock(&mutexB);
    pthread_mutex_lock(&mutexA);
    sleep(30);
    pthread_mutex_lock(&mutexA);
    pthread_mutex_lock(&mutexB);
}
```

```
#define THREAD_DECLARE(num,func) void bar_##num()\
{\
func;\
}\
\
void foo_##num()\
{\
bar_##num();\
}\
\
void * thread_##num (void *arg)\
{\
foo_##num();\
\
return 0;\
}

THREAD_DECLARE(one,sleep(-1))
THREAD_DECLARE(two,procA())
THREAD_DECLARE(three,sleep(-1))
THREAD_DECLARE(four,procB())
THREAD_DECLARE(five,sleep(-1))

#define THREAD_CREATE(num) {pthread_t threadID_##num; pthread_create (&threadID_##num, NULL,
thread_##num, NULL);}

int main(int argc, const char * argv[])
{
    pthread_mutex_init(&mutexA, NULL);
    pthread_mutex_init(&mutexB, NULL);

    THREAD_CREATE(one)
    THREAD_CREATE(two)
    sleep(10);
    THREAD_CREATE(three)
    THREAD_CREATE(four)
    THREAD_CREATE(five)

    sleep(-1);
    return 0;
}
```

Selected Patterns

(edited articles from Software Diagnostics Institute, www.DumpAnalysis.org)

NULL Pointer (data)

This is a Linux variant of **NULL Pointer** (data) pattern previously described for Mac OS X[2] and Windows[3] platforms:

```
(gdb) bt
#0  0x0000000000400500 in procA ()
#1  0x000000000040057a in bar_two ()
#2  0x000000000040058a in foo_two ()
#3  0x00000000004005a2 in thread_two ()
#4  0x0000000000401630 in start_thread (arg=<optimized out>)
at pthread_create.c:304
#5  0x00000000004324e9 in clone ()
#6  0x0000000000000000 in ?? ()

(gdb) x/i 0x400500
=> 0x400500 <procA+16>: movl   $0x1,(%rax)

(gdb) info r $rax
rax            0x0 0

(gdb) x $rax
0x0: Cannot access memory at address 0x0
```

[2] http://www.dumpanalysis.org/blog/index.php/2012/03/25/crash-dump-analysis-patterns-part-6b-mac-os-x/
[3] http://www.dumpanalysis.org/blog/index.php/2009/04/14/crash-dump-analysis-patterns-part-6b/

Incomplete Stack Trace

Users of WinDbg debugger accustomed to full thread stack traces will wonder whether a thread starts from *main*:

```
(gdb) bt
#0  0x000000000042fed1 in nanosleep ()
#1  0x000000000042fda0 in sleep ()
#2  0x000000000040078a in main ()
```

Of course, not and by default, a stack trace is shown starting from *main* function. You can change this behavior by using the following command:

```
(gdb) set backtrace past-main
```

Now we see an additional frame:

```
(gdb) bt
#0  0x000000000042fed1 in nanosleep ()
#1  0x000000000042fda0 in sleep ()
#2  0x000000000040078a in main ()
#3  0x0000000000405283 in __libc_start_main ()
#4  0x00000000004003e9 in _start ()
```

Stack Trace

This is a Linux variant of **Stack Trace** pattern previously described for Mac OS X[4] and Windows[5] platforms. Here we show a stack trace when debug symbols are not available (stripped executable) and also how to apply debug symbols from the executable where they were preserved:

```
(gdb) bt
#0 0x000000000043e4f1 in nanosleep ()
#1 0x000000000043e3c0 in sleep ()
#2 0x0000000000400789 in main ()

(gdb) symbol-file ./App/App.debug
Reading symbols from /home/Apps/App/App.debug...done.

(gdb) bt
#0 0x000000000043e4f1 in nanosleep ()
#1 0x000000000043e3c0 in sleep ()
#2 0x0000000000400789 in main (argc=1, argv=0x7fff5d1572d8) at main.cpp:85
```

[4] http://www.dumpanalysis.org/blog/index.php/2012/03/25/crash-dump-analysis-patterns-part-25-mac-os-x/
[5] http://www.dumpanalysis.org/blog/index.php/2007/09/10/crash-dump-analysis-patterns-part-25/

NULL Pointer (code)

This is a Linux variant of **NULL Pointer** (code) pattern previously described for Mac OS X[6] and Windows[7] platforms:

```
(gdb) bt
#0  0x0000000000000000 in ?? ()
#1  0x0000000000400531 in procB ()
#2  0x00000000004005f8 in bar_four ()
#3  0x0000000000400608 in foo_four ()
#4  0x0000000000400620 in thread_four ()
#5  0x0000000000401630 in start_thread (arg=<optimized out>)
at pthread_create.c:304
#6  0x00000000004324e9 in clone ()
#7  0x0000000000000000 in ?? ()

(gdb) disassemble procB
Dump of assembler code for function procB:
0x0000000000400516 <+0>: push   %rbp
0x0000000000400517 <+1>: mov    %rsp,%rbp
0x000000000040051a <+4>: sub    $0x10,%rsp
0x000000000040051e <+8>: movq   $0x0,-0x8(%rbp)
0x0000000000400526 <+16>: mov    -0x8(%rbp),%rdx
0x000000000040052a <+20>: mov    $0x0,%eax
0x000000000040052f <+25>: callq  *%rdx
0x0000000000400531 <+27>: leaveq
0x0000000000400532 <+28>: retq
End of assembler dump.

(gdb) info r rdx
rdx            0x0 0
```

[6] http://www.dumpanalysis.org/blog/index.php/2012/05/03/crash-dump-analysis-patterns-part-6a-mac-os-x/
[7] http://www.dumpanalysis.org/blog/index.php/2008/04/28/crash-dump-analysis-patterns-part-6a/

Spiking Thread

This is a variant of **Spiking Thread** pattern previously described for Mac OS X[8] and Windows[9] platforms:

```
(gdb) info threads
Id Target Id Frame
6 LWP 3712 0x00000000004329d1 in nanosleep ()
5 LWP 3717 0x00000000004007a3 in isnan ()
4 LWP 3716 0x00000000004329d1 in nanosleep ()
3 LWP 3715 0x00000000004329d1 in nanosleep ()
2 LWP 3714 0x00000000004329d1 in nanosleep ()
* 1 LWP 3713 0x00000000004329d1 in nanosleep ()
```

We notice a non-waiting thread and switch to it:

```
(gdb) thread 5
[Switching to thread 5 (LWP 3717)]
#0 0x00000000004007a3 in isnan ()

(gdb) bt
#0 0x00000000004007a3 in isnan ()
#1 0x0000000000400743 in sqrt ()
#2 0x0000000000400528 in procB ()
#3 0x0000000000400639 in bar_five ()
#4 0x0000000000400649 in foo_five ()
#5 0x0000000000400661 in thread_five ()
#6 0x0000000000403e30 in start_thread ()
#7 0x0000000000435089 in clone ()
#8 0x0000000000000000 in ?? ()
```

If we disassemble the return address for procB function to come back from *sqrt* call we see an infinite loop:

```
(gdb) disassemble 0x400528
Dump of assembler code for function procB:
0x0000000000400500 <+0>: push   %rbp
0x0000000000400501 <+1>: mov    %rsp,%rbp
0x0000000000400504 <+4>: sub    $0x20,%rsp
0x0000000000400508 <+8>: movabs $0x3fd5555555555555,%rax
0x0000000000400512 <+18>: mov    %rax,-0x8(%rbp)
0x0000000000400516 <+22>: mov    -0x8(%rbp),%rax
0x000000000040051a <+26>: mov    %rax,-0x18(%rbp)
0x000000000040051e <+30>: movsd  -0x18(%rbp),%xmm0
0x0000000000400523 <+35>: callq  0x400710 <sqrt>
0x0000000000400528 <+40>: movsd  %xmm0,-0x18(%rbp)
0x000000000040052d <+45>: mov    -0x18(%rbp),%rax
0x0000000000400531 <+49>: mov    %rax,-0x8(%rbp)
0x0000000000400535 <+53>: jmp    0x400516 <procB+22>
End of assembler dump.
```

[8] http://www.dumpanalysis.org/blog/index.php/2012/05/09/crash-dump-analysis-patterns-part-14-mac-os-x/
[9] http://www.dumpanalysis.org/blog/index.php/2007/05/11/crash-dump-analysis-patterns-part-14/

Dynamic Memory Corruption (process heap)

This is a Linux variant of **Dynamic Memory Corruption** (process heap) pattern previously described for Mac OS X[10] and Windows[11] platforms.

The corruption may be internal to heap structures with subsequent memory access violation:

```
(gdb) bt
#0  0x000000000041482e in _int_malloc ()
#1  0x0000000000416d88 in malloc ()
#2  0x00000000004005dc in proc ()
#3  0x00000000004006ee in bar_three ()
#4  0x00000000004006fe in foo_three ()
#5  0x0000000000400716 in thread_three ()
#6  0x0000000000401760 in start_thread (arg=<optimized out>)
at pthread_create.c:304
#7  0x0000000000432609 in clone ()
#8  0x0000000000000000 in ?? ()

(gdb) x/i $rip
=> 0x41482e <_int_malloc+622>: mov    %rbx,0x10(%r12)

(gdb) x $r12+0x10
0x21687371: Cannot access memory at address 0x21687371

(gdb) p (char[4])0x21687371
$1 = "qsh!"
```

Or it may be detected with a diagnostic message (similar to double free):

```
(gdb) bt
#0  0x000000000043ef65 in raise ()
#1  0x0000000000409fc0 in abort ()
#2  0x000000000040bf5b in __libc_message ()
#3  0x0000000000412042 in malloc_printerr ()
#4  0x0000000000416c27 in free ()
#5  0x0000000000400586 in proc ()
#6  0x000000000040067e in bar_four ()
#7  0x000000000040068e in foo_four ()
#8  0x00000000004006a6 in thread_four ()
#9  0x00000000004016c0 in start_thread (arg=<optimized out>)
at pthread_create.c:304
#10 0x0000000000432589 in clone ()
#11 0x0000000000000000 in ?? ()
```

[10] http://www.dumpanalysis.org/blog/index.php/2012/05/27/crash-dump-analysis-patterns-part-2-mac-os-x/
[11] http://www.dumpanalysis.org/blog/index.php/2006/10/31/crash-dump-analysis-patterns-part-2/

Execution Residue

This is a Linux variant of **Execution Residue** pattern previously described for Mac OS X[12] and Windows[13] platforms. This is symbolic information left in a stack region including ASCII and UNICODE fragments or pointers to them, for example, return addresses from past function calls:

```
(gdb) bt
#0  0x00000000004431f1 in nanosleep ()
#1  0x00000000004430c0 in sleep ()
#2  0x0000000000400771 in procNE() ()
#3  0x00000000004007aa in bar_two() ()
#4  0x00000000004007b5 in foo_two() ()
#5  0x00000000004007c8 in thread_two(void*) ()
#6  0x00000000004140f0 in start_thread (arg=<optimized out>)
at pthread_create.c:304
#7  0x0000000000445879 in clone ()
#8  0x0000000000000000 in ?? ()

(gdb) x/512a $rsp-2000
0x7f4cacc42360: 0x0 0x0
0x7f4cacc42370: 0x0 0x0
0x7f4cacc42380: 0x0 0x0
0x7f4cacc42390: 0x0 0x0
[...]
0x7f4cacc42830: 0x0 0x0
0x7f4cacc42840: 0x0 0x0
0x7f4cacc42850: 0x0 0x0
0x7f4cacc42860: 0x7f4cacc42870 0x4005af <_Z6work_8v+9>
0x7f4cacc42870: 0x7f4cacc42880 0x4005ba <_Z6work_7v+9>
0x7f4cacc42880: 0x7f4cacc42890 0x4005c5 <_Z6work_6v+9>
0x7f4cacc42890: 0x7f4cacc428a0 0x4005d0 <_Z6work_5v+9>
0x7f4cacc428a0: 0x7f4cacc428b0 0x4005db <_Z6work_4v+9>
0x7f4cacc428b0: 0x7f4cacc428c0 0x4005e6 <_Z6work_3v+9>
0x7f4cacc428c0: 0x7f4cacc428d0 0x4005f1 <_Z6work_2v+9>
0x7f4cacc428d0: 0x7f4cacc428e0 0x4005fc <_Z6work_1v+9>
0x7f4cacc428e0: 0x7f4cacc42cf0 0x40060e <_Z4workv+16>
0x7f4cacc428f0: 0x0 0x0
0x7f4cacc42900: 0x0 0x0
0x7f4cacc42910: 0x0 0x0
[...]
0x7f4cacc42af0: 0x0 0x0
0x7f4cacc42b00: 0x0 0x0
0x7f4cacc42b10: 0x0 0x0
0x7f4cacc42b20: 0x0 0x4431e6 <nanosleep+38>
0x7f4cacc42b30: 0x0 0x4430c0 <sleep+224>
0x7f4cacc42b40: 0x0 0x0
0x7f4cacc42b50: 0x0 0x0
0x7f4cacc42b60: 0x0 0x0
0x7f4cacc42b70: 0x0 0x0
[...]
0x7f4cacc42cb0: 0x0 0x0
0x7f4cacc42cc0: 0x0 0x0
0x7f4cacc42cd0: 0x0 0x0
0x7f4cacc42ce0: 0xfffffed2 0x3ad3affa
0x7f4cacc42cf0: 0x7f4cacc42d00 0x0
0x7f4cacc42d00: 0x7f4cacc42d20 0x49c740 <default_attr>
```

[12] http://www.dumpanalysis.org/blog/index.php/2012/06/05/crash-dump-analysis-patterns-part-60-mac-os-x/
[13] http://www.dumpanalysis.org/blog/index.php/2008/04/29/crash-dump-analysis-patterns-part-60/

```
0x7f4cacc42d10: 0x7f4cacc439c0 0x400771 <_Z6procNEv+19>
0x7f4cacc42d20: 0x7f4cacc42d30 0x4007aa <_Z7bar_twov+9>
0x7f4cacc42d30: 0x7f4cacc42d40 0x4007b5 <_Z7foo_twov+9>
0x7f4cacc42d40: 0x7f4cacc42d60 0x4007c8 <_Z10thread_twoPv+17>
0x7f4cacc42d50: 0x0 0x0
0x7f4cacc42d60: 0x0 0x4140f0 <start_thread+208>
0x7f4cacc42d70: 0x0 0x7f4cacc43700
0x7f4cacc42d80: 0x0 0x0
0x7f4cacc42d90: 0x0 0x0
[...]
```

However, supposed return addresses need to be checked for **Coincidental Symbolic Information** pattern.

Coincidental Symbolic Information

This is a Linux variant of **Coincidental Symbolic Information** pattern previously described for Mac OS X[14] and Windows[15] platforms. The idea is the same: to disassemble the address to see if the preceding instruction is a call. If it is indeed then most likely the symbolic address is a return address from past **Execution Residue**:

```
(gdb) x/i 0x4005e6
0x4005e6 <_Z6work_3v+9>: pop    %rbp

(gdb) disassemble 0x4005e6
Dump of assembler code for function _Z6work_3v:
0x00000000004005dd <+0>: push   %rbp
0x00000000004005de <+1>: mov    %rsp,%rbp
0x00000000004005e1 <+4>: callq  0x4005d2 <_Z6work_4v>
0x00000000004005e6 <+9>: pop    %rbp
0x00000000004005e7 <+10>: retq
End of assembler dump.

(gdb) x/4i 0x49c740-4
0x49c73c: add    %al,(%rax)
0x49c73e: add    %al,(%rax)
0x49c740 <default_attr>: add    %al,(%rax)
0x49c742 <default_attr+2>: add    %al,(%rax)
```

[14] http://www.dumpanalysis.org/blog/index.php/2012/06/09/crash-dump-analysis-patterns-part-24-mac-os-x/
[15] http://www.dumpanalysis.org/blog/index.php/2007/08/30/crash-dump-analysis-patterns-part-24/

Stack Overflow (user mode)

This is a Linux variant of **Stack Overflow** (user mode) pattern previously described for Mac OS X[16] and Windows[17] platforms:

```
(gdb) bt
#0  0x00000000004004fb in procF ()
#1  0x000000000040054b in procF ()
#2  0x000000000040054b in procF ()
#3  0x000000000040054b in procF ()
#4  0x000000000040054b in procF ()
#5  0x000000000040054b in procF ()
#6  0x000000000040054b in procF ()
#7  0x000000000040054b in procF ()
#8  0x000000000040054b in procF ()
#9  0x000000000040054b in procF ()
#10 0x000000000040054b in procF ()
#11 0x000000000040054b in procF ()
#12 0x000000000040054b in procF ()
[...]

(gdb) bt -10
#15409 0x000000000040054b in procF ()
#15410 0x000000000040054b in procF ()
#15411 0x000000000040054b in procF ()
#15412 0x000000000040055b in procE ()
#15413 0x0000000000400575 in bar_one ()
#15414 0x0000000000400585 in foo_one ()
#15415 0x000000000040059d in thread_one ()
#15416 0x0000000000401690 in start_thread (arg=<optimized out>)
at pthread_create.c:304
#15417 0x0000000000432549 in clone ()
#15418 0x0000000000000000 in ?? ()
```

In case of a stack overflow, the stack pointer is decremented beyond the stack region boundary into a non-accessible region, so any stack memory access triggers an access violation:

```
(gdb) x $rsp
0x7eff46109ec0: 0x0

(gdb) frame 1
#1  0x000000000040054b in procF ()

(gdb) x $rsp
0x7eff4610a0e0: 0x0

(gdb) maintenance info sections
[...]
Core file:
[...]
0x7eff46109000->0x7eff4610a000 at 0x02034000: load13 ALLOC LOAD READONLY HAS_CONTENTS
0x7eff4610a000->0x7eff4690a000 at 0x02035000: load14 ALLOC LOAD HAS_CONTENTS
[...]
```

[16] http://www.dumpanalysis.org/blog/index.php/2012/07/17/crash-dump-analysis-patterns-part-16b-mac-os-x/
[17] http://www.dumpanalysis.org/blog/index.php/2008/06/10/crash-dump-analysis-patterns-part-16b/

Divide by Zero (user mode)

This is a Linux variant of **Divide by Zero** (user mode) pattern previously described for Mac OS X[18] and Windows[19] platforms:

```
GNU gdb (GDB)
[...]
Program terminated with signal 8, Arithmetic exception.
#0 0×000000000040056f in procD ()

(gdb) x/i $rip
=> 0x40056f <procD+18>: idivl -0×8(%rbp)

(gdb) info r $rax
rax 0x1 1

(gdb) x/w $rbp-0x8
0x7f0f6806bd28: 0×00000000
```

[18] http://www.dumpanalysis.org/blog/index.php/2012/07/18/crash-dump-analysis-patterns-part-78a-mac-os-x/
[19] http://www.dumpanalysis.org/blog/index.php/2008/12/01/crash-dump-analysis-patterns-part-78a/

Local Buffer Overflow

This is a Linux variant of **Local Buffer Overflow** pattern previously described for Mac OS X[20] and Windows[21] platforms. Most of the time simple mistakes in using memory and string manipulation functions are easily detected by the runtime. The more sophisticated example which overwrites stack trace without being detected involves overwriting indirectly via a pointer to a local buffer passed to the called function. In such cases, we might see incorrect and truncated stack traces:

```
(gdb) bt
#0  0×0000000000000000 in ?? ()
#1  0×0000000000000000 in ?? ()

(gdb) x/100a $rsp
[...]
0x7fc3dd9dece8: 0x0 0x0
0x7fc3dd9decf8: 0x0 0x0
0x7fc3dd9ded08: 0x0 0x0
0x7fc3dd9ded18: 0x0 0x0
0x7fc3dd9ded28: 0×7fc3dd9ded48 0×4005cc <procA+40>
0x7fc3dd9ded38: 0×422077654e20794d 0×7542207265676769
0×7fc3dd9ded48: 0×72656666 0×0
0×7fc3dd9ded58: 0×0 0×0
0×7fc3dd9ded68: 0×0 0×0
0×7fc3dd9ded78: 0×0 0×0
[...]
```

[20] http://www.dumpanalysis.org/blog/index.php/2012/07/19/crash-dump-analysis-patterns-part-36-mac-os-x/
[21] http://www.dumpanalysis.org/blog/index.php/2007/11/14/crash-dump-analysis-patterns-part-36/

C++ Exception

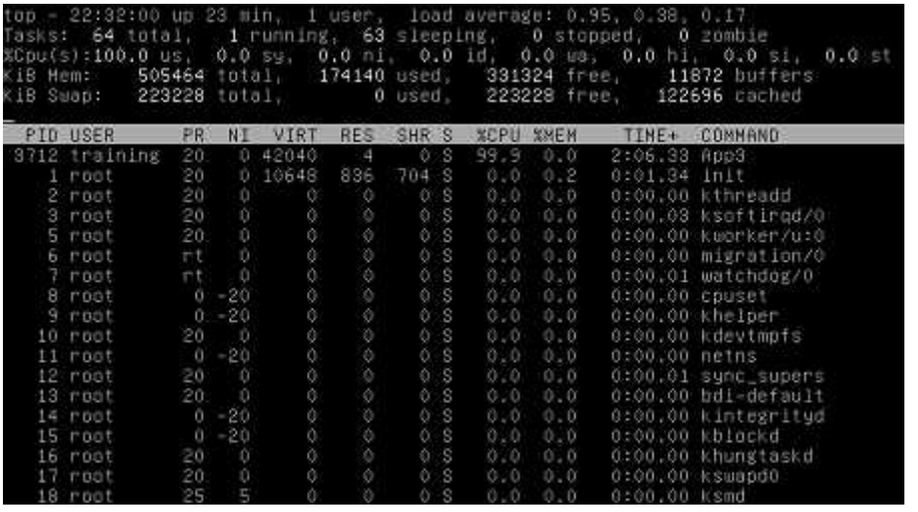
This is a Linux variant of **C++ Exception** pattern previously described for Mac OS X[22] and Windows[23] platforms:

```
(gdb) bt
#0 0x00007f0a1d0e5165 in *__GI_raise ()
at ../nptl/sysdeps/unix/sysv/linux/raise.c:64
#1 0x00007f0a1d0e83e0 in *__GI_abort () at abort.c:92
#2 0x00007f0a1db5789d in __gnu_cxx::__verbose_terminate_handler() ()
from /usr/lib/x86_64-linux-gnu/libstdc++.so.6
#3 0x00007f0a1db55996 in ?? () from /usr/lib/x86_64-linux-gnu/libstdc++.so.6
#4 0x00007f0a1db559c3 in std::terminate() ()
from /usr/lib/x86_64-linux-gnu/libstdc++.so.6
#5 0x00007f0a1db55bee in __cxa_throw ()
from /usr/lib/x86_64-linux-gnu/libstdc++.so.6
#6 0x0000000000400dcf in procB() ()
#7 0x0000000000400e26 in procA() ()
#8 0x0000000000400e88 in procNH() ()
#9 0x0000000000400ea8 in bar_one() ()
#10 0x0000000000400eb3 in foo_one() ()
#11 0x0000000000400ec6 in thread_one(void*) ()
#12 0x00007f0a1d444b50 in start_thread ()
#13 0x00007f0a1d18e95d in clone ()
at ../sysdeps/unix/sysv/linux/x86_64/clone.S:112
#14 0x0000000000000000 in ?? ()
```

[22] http://www.dumpanalysis.org/blog/index.php/2012/07/20/crash-dump-analysis-patterns-part-77-mac-os-x/
[23] http://www.dumpanalysis.org/blog/index.php/2008/10/21/crash-dump-analysis-patterns-part-77/

Paratext

This is Linux variant of **Paratext** pattern for Mac OS X[24]. Because of debugger tool limitations additional software logs and the output of other tools may help in memory dump analysis. Typical examples of such pattern usage can be the list of modules with version and path info, application crash specific information from instrumentation tools such as Valgrind, memory region names with attribution and boundaries, and CPU usage information. For example, top and pmap commands output:




```
14039: ./App1.shared
0000000000400000 4K r-x-- /home/training/ALCDA/App1/App1.shared
0000000000600000 4K rw--- /home/training/ALCDA/App1/App1.shared
0000000000611000 132K rw--- [ anon ]
00007fe8999a6000 4K ----- [ anon ]
00007fe8999a7000 8192K rw--- [ anon ]
00007fe89a1a7000 4K ----- [ anon ]
00007fe89a1a8000 8192K rw--- [ anon ]
00007fe89a9a8000 4K ----- [ anon ]
00007fe89a9a9000 8192K rw--- [ anon ]
00007fe89b1a9000 4K ----- [ anon ]
00007fe89b1aa000 8192K rw--- [ anon ]
00007fe89b9aa000 4K ----- [ anon ]
00007fe89b9ab000 8192K rw--- [ anon ]
00007fe89c1ab000 1540K r-x-- /lib/x86_64-linux-gnu/libc-2.13.so
00007fe89c32c000 2048K ----- /lib/x86_64-linux-gnu/libc-2.13.so
00007fe89c52c000 16K r---- /lib/x86_64-linux-gnu/libc-2.13.so
00007fe89c530000 4K rw--- /lib/x86_64-linux-gnu/libc-2.13.so
00007fe89c531000 20K rw--- [ anon ]
00007fe89c536000 92K r-x-- /lib/x86_64-linux-gnu/libpthread-2.13.so
00007fe89c54d000 2044K ----- /lib/x86_64-linux-gnu/libpthread-2.13.so
00007fe89c74c000 4K r---- /lib/x86_64-linux-gnu/libpthread-2.13.so
00007fe89c74d000 4K rw--- /lib/x86_64-linux-gnu/libpthread-2.13.so
00007fe89c74e000 16K rw--- [ anon ]
00007fe89c752000 128K r-x-- /lib/x86_64-linux-gnu/ld-2.13.so
00007fe89c966000 12K rw--- [ anon ]
00007fe89c96f000 8K rw--- [ anon ]
00007fe89c971000 4K r---- /lib/x86_64-linux-gnu/ld-2.13.so
00007fe89c972000 4K rw--- /lib/x86_64-linux-gnu/ld-2.13.so
00007fe89c973000 4K rw--- [ anon ]
00007ffd458c1000 132K rw--- [ stack ]
00007ffd459e9000 4K r-x-- [ anon ]
ffffffffff600000 4K r-x-- [ anon ]
total 47208K
```

[24] http://www.dumpanalysis.org/blog/index.php/2012/07/28/crash-dump-analysis-patterns-part-180-mac-os-x/

Active Thread

Here we publish a Linux variant of **Active Thread** pattern that was previously introduced for Mac OS X[25] and Windows[26]. Basically, it is a thread that is not waiting, sleeping, or suspended (most threads are). However, from a memory dump, it is not possible to find out whether it was **Spiking Thread** at the dump generation time (unless we have a set of memory snapshots and in each one we have the same or similar backtrace) and we don't have any **Paratext** with CPU consumption stats for threads. For example, in one core dump we have this thread:

```
(gdb) info threads
Id Target Id Frame
6 Thread 0×7f560d467700 (LWP 3483) 0×00000000004324a9 in clone ()
5 Thread 0×7f560c465700 (LWP 3485) 0×000000000042fe31 in nanosleep ()
4 Thread 0×7f560bc64700 (LWP 3486) 0×000000000042fe31 in nanosleep ()
3 Thread 0×7f560b463700 (LWP 3487) 0×000000000042fe31 in nanosleep ()
2 Thread 0×18b9860 (LWP 3482) 0×000000000042fe31 in nanosleep ()
1 Thread 0×7f560cc66700 (LWP 3484) 0×000000000042fe31 in nanosleep ()
```

Thread #6 is not waiting so we inspect its back trace:

```
(gdb) thread 6
[Switching to thread 6 (Thread 0x7f560d467700 (LWP 3483))]
#0 0x00000000004324a9 in clone ()

(gdb) bt
#0 0×00000000004324a9 in clone ()
#1 0×0000000000401560 in ?? () at pthread_create.c:217
#2 0×00007f560d467700 in ?? ()
#3 0×0000000000000000 in ?? ()

(gdb) x/i 0x4324a9
=> 0x4324a9 : test %rax,%rax
```

Perhaps the core dump was saved at the thread creation time.

[25] http://www.dumpanalysis.org/blog/index.php/2012/11/17/crash-dump-analysis-patterns-part-187-mac-os-x/
[26] http://www.dumpanalysis.org/blog/index.php/2015/10/31/crash-dump-analysis-patterns-part-232/

Lateral Damage

This is a Linux variant of **Lateral Damage** pattern previously described for Windows[27] platform. It also covers memory dumps where some usual commands may not work, and we have to find a workaround to simulate their output, for example, by using other commands:

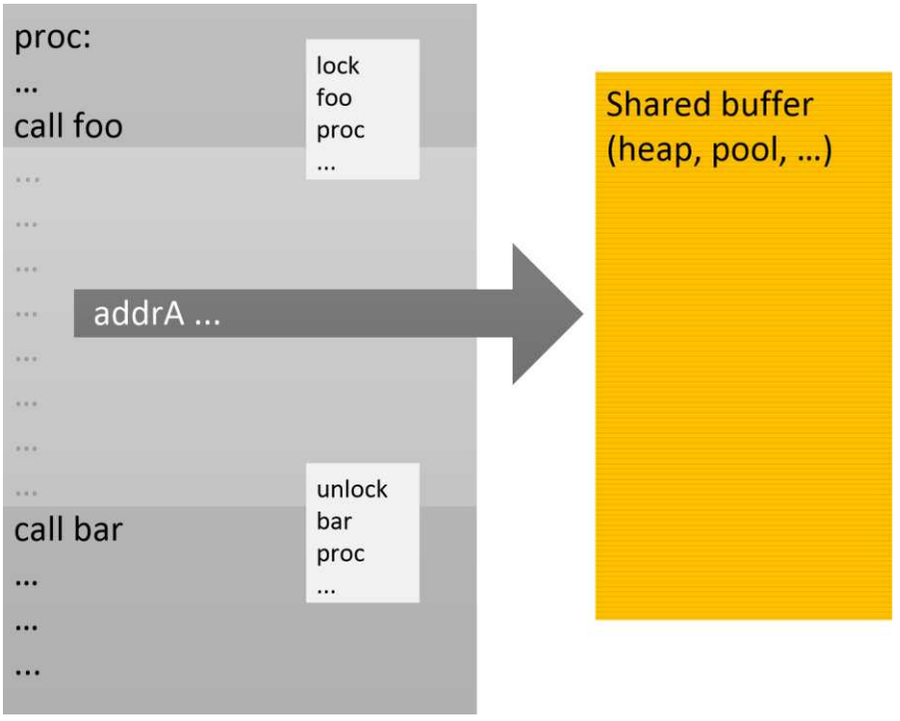
```
(gdb) info threads
Cannot find new threads: generic error

(gdb) thread apply all bt
Cannot find new threads: generic error

(gdb) thread 2
[Switching to thread 2 (LWP 12567)]
#0 0x000000000042ff51 in nanosleep ()

(gdb) thread 3
[Switching to thread 3 (LWP 12566)]
#0 0x000000000041482e in _int_malloc ()
```

[27] http://www.dumpanalysis.org/blog/index.php/2006/11/03/crash-dump-analysis-patterns-part-4/

Critical Region

We first introduced **Critical Region** pattern in Accelerated Mac OS X Core Dump Analysis[28] training but didn't submit the pattern itself to the catalog at that time.

A critical region is usually a region of code protected by synchronization objects such as critical sections and mutexes. However, **Critical Region** analysis pattern is about identifying code regions "sandwiched" between contending function calls (which may or may not involve synchronization objects and corresponding synchronization calls such as identified in Contention[29] patterns), and then identifying any possibly shared data referenced by such code regions:




```
(gdb) thread apply all bt

Thread 6 (Thread 0x7f2665377700 (LWP 17000)):
#0  0x00000000004151a1 in _int_malloc ()
#1  0x0000000000416cf8 in malloc ()
#2  0x00000000004005a4 in proc ()
#3  0x0000000000400604 in bar_two ()
#4  0x0000000000400614 in foo_two ()
#5  0x000000000040062c in thread_two ()
#6  0x00000000004016c0 in start_thread (arg=<optimized out>)
at pthread_create.c:304
#7  0x0000000000432589 in clone ()
#8  0x0000000000000000 in ?? ()
```

[28] http://www.patterndiagnostics.com/accelerated-macosx-core-dump-analysis-book
[29] http://www.dumpanalysis.org/blog/index.php/2010/09/21/contention-patterns/

```
Thread 5 (Thread 0x7f2664b76700 (LWP 17001)):
#0  __lll_unlock_wake_private ()
at ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S:343
#1  0x000000000041886d in _L_unlock_9670 ()
#2  0x0000000000416d22 in malloc ()
#3  0x00000000004005a4 in proc ()
#4  0x0000000000400641 in bar_three ()
#5  0x0000000000400651 in foo_three ()
#6  0x0000000000400669 in thread_three ()
#7  0x00000000004016c0 in start_thread (arg=<optimized out>)
at pthread_create.c:304
#8  0x0000000000432589 in clone ()
#9  0x0000000000000000 in ?? ()

Thread 4 (Thread 0x7f2665b78700 (LWP 16999)):
#0  __lll_lock_wait_private ()
at ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S:97
#1  0x0000000000418836 in _L_lock_9558 ()
#2  0x0000000000416c1c in free ()
#3  0x0000000000400586 in proc ()
#4  0x00000000004005c7 in bar_one ()
#5  0x00000000004005d7 in foo_one ()
#6  0x00000000004005ef in thread_one ()
#7  0x00000000004016c0 in start_thread (arg=<optimized out>)
at pthread_create.c:304
#8  0x0000000000432589 in clone ()
#9  0x0000000000000000 in ?? ()

Thread 3 (Thread 0x1ab1860 (LWP 16998)):
#0  0x000000000042fed1 in nanosleep ()
#1  0x000000000042fda0 in sleep ()
#2  0x000000000040078a in main ()

Thread 2 (Thread 0x7f2663b74700 (LWP 17003)):
#0  __lll_lock_wait_private ()
at ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S:97
#1  0x0000000000418836 in _L_lock_9558 ()
#2  0x0000000000416c1c in free ()
#3  0x0000000000400586 in proc ()
#4  0x00000000004006bb in bar_five ()
#5  0x00000000004006cb in foo_five ()
#6  0x00000000004006e3 in thread_five ()
#7  0x00000000004016c0 in start_thread (arg=<optimized out>)
at pthread_create.c:304
#8  0x0000000000432589 in clone ()
#9  0x0000000000000000 in ?? ()

Thread 1 (Thread 0x7f2664375700 (LWP 17002)):
#0  0x000000000043ef65 in raise ()
#1  0x0000000000409fc0 in abort ()
#2  0x000000000040bf5b in __libc_message ()
#3  0x0000000000412042 in malloc_printerr ()
#4  0x0000000000416c27 in free ()
#5  0x0000000000400586 in proc ()
#6  0x000000000040067e in bar_four ()
#7  0x000000000040068e in foo_four ()
#8  0x00000000004006a6 in thread_four ()
#9  0x00000000004016c0 in start_thread (arg=<optimized out>)
at pthread_create.c:304
```

```
#10 0x0000000000432589 in clone ()
#11 0x0000000000000000 in ?? ()
```

From threads #4 and #5 we can identify one such a region with a shared buffer 0×6b8fc0 which may further point to heap entries.

```
(gdb) disassemble proc
Dump of assembler code for function proc:
0x00000000004004f0 <+0>: push   %rbp
0x00000000004004f1 <+1>: mov    %rsp,%rbp
0x00000000004004f4 <+4>: push   %rbx
0x00000000004004f5 <+5>: sub    $0x18,%rsp
0x00000000004004f9 <+9>: callq  0x40ac70 <rand>
0x00000000004004fe <+14>: mov    %eax,%ecx
0x0000000000400500 <+16>: mov    $0x68db8bad,%edx
0x0000000000400505 <+21>: mov    %ecx,%eax
0x0000000000400507 <+23>: imul   %edx
0x0000000000400509 <+25>: sar    $0xc,%edx
0x000000000040050c <+28>: mov    %ecx,%eax
0x000000000040050e <+30>: sar    $0x1f,%eax
0x0000000000400511 <+33>: mov    %edx,%ebx
0x0000000000400513 <+35>: sub    %eax,%ebx
0x0000000000400515 <+37>: mov    %ebx,%eax
0x0000000000400517 <+39>: mov    %eax,-0x14(%rbp)
0x000000000040051a <+42>: mov    -0x14(%rbp),%eax
0x000000000040051d <+45>: imul   $0x2710,%eax,%eax
0x0000000000400523 <+51>: mov    %ecx,%edx
0x0000000000400525 <+53>: sub    %eax,%edx
0x0000000000400527 <+55>: mov    %edx,%eax
0x0000000000400529 <+57>: mov    %eax,-0x14(%rbp)
0x000000000040052c <+60>: callq  0x40ac70 <rand>
0x0000000000400531 <+65>: mov    %eax,%ecx
0x0000000000400533 <+67>: mov    $0x68db8bad,%edx
0x0000000000400538 <+72>: mov    %ecx,%eax
0x000000000040053a <+74>: imul   %edx
0x000000000040053c <+76>: sar    $0xc,%edx
0x000000000040053f <+79>: mov    %ecx,%eax
0x0000000000400541 <+81>: sar    $0x1f,%eax
0x0000000000400544 <+84>: mov    %edx,%ebx
0x0000000000400546 <+86>: sub    %eax,%ebx
0x0000000000400548 <+88>: mov    %ebx,%eax
0x000000000040054a <+90>: mov    %eax,-0x18(%rbp)
0x000000000040054d <+93>: mov    -0x18(%rbp),%eax
0x0000000000400550 <+96>: imul   $0x2710,%eax,%eax
0x0000000000400556 <+102>: mov    %ecx,%edx
0x0000000000400558 <+104>: sub    %eax,%edx
0x000000000040055a <+106>: mov    %edx,%eax
0x000000000040055c <+108>: mov    %eax,-0x18(%rbp)
0x000000000040055f <+111>: mov    -0x14(%rbp),%eax
0x0000000000400562 <+114>: cltq
0x0000000000400564 <+116>: mov    0x6b8fc0(,%rax,8),%rax
0x000000000040056c <+124>: test   %rax,%rax
0x000000000040056f <+127>: je     0x400597 <proc+167>
0x0000000000400571 <+129>: mov    -0x14(%rbp),%eax
0x0000000000400574 <+132>: cltq
0x0000000000400576 <+134>: mov    0x6b8fc0(,%rax,8),%rax
0x000000000040057e <+142>: mov    %rax,%rdi
```

```
0x0000000000400581 <+145>: callq  0x416bc0 <free>
0x0000000000400586 <+150>: mov    -0x14(%rbp),%eax
0x0000000000400589 <+153>: cltq
0x000000000040058b <+155>: movq   $0x0,0x6b8fc0(,%rax,8)
0x0000000000400597 <+167>: mov    -0x18(%rbp),%eax
0x000000000040059a <+170>: cltq
0x000000000040059c <+172>: mov    %rax,%rdi
0x000000000040059f <+175>: callq  0x416c90 <malloc>
0x00000000004005a4 <+180>: mov    %rax,%rdx
0x00000000004005a7 <+183>: mov    -0x14(%rbp),%eax
0x00000000004005aa <+186>: cltq
0x00000000004005ac <+188>: mov    %rdx,0x6b8fc0(,%rax,8)
0x00000000004005b4 <+196>: jmpq   0x4004f9 <proc+9>
End of assembler dump.
```