macOS

Core Dump Analysis

Accelerated

Version 3.0

Dmitry Vostokov
Software Diagnostics Services

Published by OpenTask, Republic of Ireland

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: 978-1-912636-75-4 (Paperback)

Revision 3.00 (December 2022)

Contents

Exercise X1

Goal: Learn how to list stack traces, disassemble functions, follow function calls, check backtrace correctness, dump data, and get the environment.

Patterns: Manual Dump (Process); Stack Trace; Stack Trace Collection; Annotated Disassembly; Paratext; Not My Version; Environment Hint.

1. Load a core dump *App1-84080-20221124T015838Z* and *App1* executable:

```
% lldb -c ~/AMCDA-Dumps/App1-84080-20221124T015838Z -f ~/AMCDA-
Dumps/Apps/App1/Build/Products/Release/App1
((lldb) target create "/Users/training/AMCDA-Dumps/Apps/App1/Build/Products/Release/App1" --
core "/Users/training/AMCDA-Dumps/App1-84080-20221124T015838Z"
Core file '/Users/training/AMCDA-Dumps/App1-84080-20221124T015838Z' (arm64) was loaded.
(lldb)
```

2. List all threads:

```
(lldb) thread list
Process 0 stopped
* thread #1: tid = 0x0000, 0x000000018e9ce06c libsystem_kernel.dylib`__semwait_signal + 8
  thread #2: tid = 0x0001, 0x000000018e9ce06c libsystem_kernel.dylib`__semwait_signal + 8
  thread #3: tid = 0x0002, 0x000000018e9ce06c libsystem_kernel.dylib`__semwait_signal + 8
  thread #4: tid = 0x0003, 0x000000018e9ce06c libsystem_kernel.dylib`__semwait_signal + 8
  thread #5: tid = 0x0004, 0x000000018e9ce06c libsystem_kernel.dylib`__semwait_signal + 8
  thread #6: tid = 0x0005, 0x000000018e9ce06c libsystem_kernel.dylib`__semwait_signal + 8
```

Note: We see LLDB listed 6 threads with their TIDs numbered from 0.

3. Get all thread stack traces:

```
(lldb) thread backtrace all
* thread #1
  * frame #0: 0x000000018e9ce06c libsystem_kernel.dylib`__semwait_signal + 8
    frame #1: 0x000000018e8d6fc8 libsystem_c.dylib`nanosleep + 220
    frame #2: 0x000000018e8e1b78 libsystem_c.dylib`sleep + 52
    frame #3: 0x000000018e8e1b90 libsystem_c.dylib`sleep + 76
    frame #4: 0x0000000102603f90 App1`main + 164
    frame #5: 0x00000001028e108c dyld`start + 520
  thread #2
    frame #0: 0x000000018e9ce06c libsystem_kernel.dylib`__semwait_signal + 8
    frame #1: 0x000000018e8d6fc8 libsystem_c.dylib`nanosleep + 220
    frame #2: 0x000000018e8e1b78 libsystem_c.dylib`sleep + 52
    frame #3: 0x000000018e8e1b90 libsystem_c.dylib`sleep + 76
    frame #4: 0x0000000102603d6c App1`bar_one + 16
    frame #5: 0x0000000102603d80 App1`foo_one + 12
    frame #6: 0x0000000102603d9c App1`thread_one + 20
    frame #7: 0x000000018ea0826c libsystem_pthread.dylib`_pthread_start + 148
  thread #3
    frame #0: 0x000000018e9ce06c libsystem_kernel.dylib`__semwait_signal + 8
    frame #1: 0x000000018e8d6fc8 libsystem_c.dylib`nanosleep + 220
    frame #2: 0x000000018e8e1b78 libsystem_c.dylib`sleep + 52
    frame #3: 0x000000018e8e1b90 libsystem_c.dylib`sleep + 76
    frame #4: 0x0000000102603dbc App1`bar_two + 16
    frame #5: 0x0000000102603dd0 App1`foo_two + 12
```

```
    frame #6: 0x0000000102603dec App1`thread_two + 20
    frame #7: 0x000000018ea0826c libsystem_pthread.dylib`_pthread_start + 148
  thread #4
    frame #0: 0x000000018e9ce06c libsystem_kernel.dylib`__semwait_signal + 8
    frame #1: 0x000000018e8d6fc8 libsystem_c.dylib`nanosleep + 220
    frame #2: 0x000000018e8e1b78 libsystem_c.dylib`sleep + 52
    frame #3: 0x000000018e8e1b90 libsystem_c.dylib`sleep + 76
    frame #4: 0x0000000102603e0c App1`bar_three + 16
    frame #5: 0x0000000102603e20 App1`foo_three + 12
    frame #6: 0x0000000102603e3c App1`thread_three + 20
    frame #7: 0x000000018ea0826c libsystem_pthread.dylib`_pthread_start + 148
  thread #5
    frame #0: 0x000000018e9ce06c libsystem_kernel.dylib`__semwait_signal + 8
    frame #1: 0x000000018e8d6fc8 libsystem_c.dylib`nanosleep + 220
    frame #2: 0x000000018e8e1b78 libsystem_c.dylib`sleep + 52
    frame #3: 0x000000018e8e1b90 libsystem_c.dylib`sleep + 76
    frame #4: 0x0000000102603e5c App1`bar_four + 16
    frame #5: 0x0000000102603e70 App1`foo_four + 12
    frame #6: 0x0000000102603e8c App1`thread_four + 20
    frame #7: 0x000000018ea0826c libsystem_pthread.dylib`_pthread_start + 148
  thread #6
    frame #0: 0x000000018e9ce06c libsystem_kernel.dylib`__semwait_signal + 8
    frame #1: 0x000000018e8d6fc8 libsystem_c.dylib`nanosleep + 220
    frame #2: 0x000000018e8e1b78 libsystem_c.dylib`sleep + 52
    frame #3: 0x000000018e8e1b90 libsystem_c.dylib`sleep + 76
    frame #4: 0x0000000102603eac App1`bar_five + 16
    frame #5: 0x0000000102603ec0 App1`foo_five + 12
    frame #6: 0x0000000102603edc App1`thread_five + 20
    frame #7: 0x000000018ea0826c libsystem_pthread.dylib`_pthread_start + 148
```

4. Switch to thread #3 and get its stack trace:

```
(lldb) thread select 3
* thread #3
    frame #0: 0x000000018e9ce06c libsystem_kernel.dylib`__semwait_signal + 8
libsystem_kernel.dylib`:
->  0x18e9ce06c <+8>:  b.lo   0x18e9ce08c               ; <+40>
    0x18e9ce070 <+12>: pacibsp
    0x18e9ce074 <+16>: stp    x29, x30, [sp, #-0x10]!
    0x18e9ce078 <+20>: mov    x29, sp
```

Note: Also, we have code disassembly starting from the next instruction that was to be executed if the dump wasn't saved. The nice feature is annotated disassembly that shows symbolic names for branch and link destinations if we disassemble the function around the current address (**di** without an address can also be used, also, notice pointer authentication instructions for X30):

```
(lldb) di -a 0x18e9ce06c
libsystem_kernel.dylib`:
    0x18e9ce064 <+0>:  mov    x16, #0x14e
    0x18e9ce068 <+4>:  svc    #0x80
->  0x18e9ce06c <+8>:  b.lo   0x18e9ce08c               ; <+40>
    0x18e9ce070 <+12>: pacibsp
    0x18e9ce074 <+16>: stp    x29, x30, [sp, #-0x10]!
    0x18e9ce078 <+20>: mov    x29, sp
    0x18e9ce07c <+24>: bl     0x18e9cc328               ; cerror
    0x18e9ce080 <+28>: mov    sp, x29
    0x18e9ce084 <+32>: ldp    x29, x30, [sp], #0x10
    0x18e9ce088 <+36>: retab
    0x18e9ce08c <+40>: ret
```

Note: We can also list any thread stack trace without switching to it:

```
(lldb) thread backtrace 4
  thread #4
    frame #0: 0x000000018e9ce06c libsystem_kernel.dylib`__semwait_signal + 8
    frame #1: 0x000000018e8d6fc8 libsystem_c.dylib`nanosleep + 220
    frame #2: 0x000000018e8e1b78 libsystem_c.dylib`sleep + 52
    frame #3: 0x000000018e8e1b90 libsystem_c.dylib`sleep + 76
    frame #4: 0x0000000102603e0c App1`bar_three + 16
    frame #5: 0x0000000102603e20 App1`foo_three + 12
    frame #6: 0x0000000102603e3c App1`thread_three + 20
    frame #7: 0x000000018ea0826c libsystem_pthread.dylib`_pthread_start + 148
```

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

```
(lldb) bt
* thread #3
  * frame #0: 0x000000018e9ce06c libsystem_kernel.dylib`__semwait_signal + 8
    frame #1: 0x000000018e8d6fc8 libsystem_c.dylib`nanosleep + 220
    frame #2: 0x000000018e8e1b78 libsystem_c.dylib`sleep + 52
    frame #3: 0x000000018e8e1b90 libsystem_c.dylib`sleep + 76
    frame #4: 0x0000000102603dbc App1`bar_two + 16
    frame #5: 0x0000000102603dd0 App1`foo_two + 12
    frame #6: 0x0000000102603dec App1`thread_two + 20
    frame #7: 0x000000018ea0826c libsystem_pthread.dylib`_pthread_start + 148

(lldb) di -n bar_two
App1`bar_two:
    0x102603dac <+0>:  stp    x29, x30, [sp, #-0x10]!
    0x102603db0 <+4>:  mov    x29, sp
    0x102603db4 <+8>:  mov    w0, #-0x1
    0x102603db8 <+12>: bl     0x102603fac               ; symbol stub for: sleep
    0x102603dbc <+16>: ldp    x29, x30, [sp], #0x10
    0x102603dc0 <+20>: ret
```

6. Follow the *bar_two* function to the *sleep* function code:

```
(lldb) di -n bar_two
App1`bar_two:
    0x102603dac <+0>:  stp    x29, x30, [sp, #-0x10]!
    0x102603db0 <+4>:  mov    x29, sp
    0x102603db4 <+8>:  mov    w0, #-0x1
    0x102603db8 <+12>: bl     0x102603fac               ; symbol stub for: sleep
    0x102603dbc <+16>: ldp    x29, x30, [sp], #0x10
    0x102603dc0 <+20>: ret

(lldb) di -a 0x102603fac
App1`sleep:
    0x102603fac <+0>: adrp   x16, 1
    0x102603fb0 <+4>: ldr    x16, [x16, #0x8]
    0x102603fb4 <+8>: br     x16
```

Note: This short code fragment is an indirect call to the *sleep* library function. We can calculate it as follows:

```
(lldb) x/a 0x102603000+0x1000*1+0x8
0x102604008: 0x000000018e8e1b44 libsystem_c.dylib`sleep
```

7. Disassemble the target address value:

```
(lldb) di -a 0x000000018e8e1b44
libsystem_c.dylib`sleep:
    0x18e8e1b44 <+0>:   pacibsp
    0x18e8e1b48 <+4>:   sub    sp, sp, #0x40
    0x18e8e1b4c <+8>:   stp    x20, x19, [sp, #0x20]
    0x18e8e1b50 <+12>:  stp    x29, x30, [sp, #0x30]
    0x18e8e1b54 <+16>:  add    x29, sp, #0x30
    0x18e8e1b58 <+20>:  mov    x19, x0
    0x18e8e1b5c <+24>:  stp    xzr, xzr, [sp]
    0x18e8e1b60 <+28>:  tbnz   w0, #0x1f, 0x18e8e1b88    ; <+68>
    0x18e8e1b64 <+32>:  mov    w8, w19
    0x18e8e1b68 <+36>:  stp    x8, xzr, [sp, #0x10]
    0x18e8e1b6c <+40>:  add    x0, sp, #0x10
    0x18e8e1b70 <+44>:  mov    x1, sp
    0x18e8e1b74 <+48>:  bl     0x18e8d6eec              ; nanosleep
    0x18e8e1b78 <+52>:  cmn    w0, #0x1
    0x18e8e1b7c <+56>:  b.eq   0x18e8e1ba0              ; <+92>
    0x18e8e1b80 <+60>:  mov    w19, #0x0
    0x18e8e1b84 <+64>:  b      0x18e8e1bc0              ; <+124>
    0x18e8e1b88 <+68>:  mov    w0, #0x7fffffff
    0x18e8e1b8c <+72>:  bl     0x18e8e1b44              ; <+0>
    0x18e8e1b90 <+76>:  mov    w8, #-0x7fffffff
    0x18e8e1b94 <+80>:  add    w9, w19, w0
    0x18e8e1b98 <+84>:  add    w19, w9, w8
    0x18e8e1b9c <+88>:  b      0x18e8e1bc0              ; <+124>
    0x18e8e1ba0 <+92>:  bl     0x18e942bb0              ; symbol stub for: __error
    0x18e8e1ba4 <+96>:  ldr    w8, [x0]
    0x18e8e1ba8 <+100>: cmp    w8, #0x4
    0x18e8e1bac <+104>: b.ne   0x18e8e1bc0              ; <+124>
    0x18e8e1bb0 <+108>: ldr    w8, [sp]
    0x18e8e1bb4 <+112>: ldr    x9, [sp, #0x8]
    0x18e8e1bb8 <+116>: cmp    x9, #0x0
    0x18e8e1bbc <+120>: cinc   w19, w8, ne
    0x18e8e1bc0 <+124>: mov    x0, x19
    0x18e8e1bc4 <+128>: ldp    x29, x30, [sp, #0x30]
    0x18e8e1bc8 <+132>: ldp    x20, x19, [sp, #0x20]
    0x18e8e1bcc <+136>: add    sp, sp, #0x40
    0x18e8e1bd0 <+140>: retab
```

8. Get the *App1* stack section from the output of *vmmap_84080.log*:

```
[...]

Virtual Memory Map of process 84080 (App1)
Output report format:  2.4  -- 64-bit process
VM page size:  16384 bytes

==== Non-writable regions for process 84080
REGION TYPE                    START - END         [ VSIZE  RSDNT  DIRTY   SWAP] PRT/MAX SHRMOD PURGE    REGION DETAIL
__TEXT                      102600000-102604000    [   16K    16K     0K     0K] r-x/r-x SM=COW          /Users/USER/*/App1

[...]

==== Writable regions for process 84080
REGION TYPE                    START - END         [ VSIZE  RSDNT  DIRTY   SWAP] PRT/MAX SHRMOD PURGE    REGION DETAIL
Kernel Alloc Once           102710000-102718000    [   32K     0K     0K    16K] rw-/rwx SM=COW
MALLOC metadata             102720000-102724000    [   16K    16K    16K     0K] rw-/rwx SM=COW
MALLOC metadata             102728000-102730000    [   32K    32K    32K     0K] rw-/rwx SM=COW
MALLOC metadata             102738000-102740000    [   32K    32K    32K     0K] rw-/rwx SM=PRV
MALLOC metadata             102748000-102750000    [   32K     0K     0K    32K] rw-/rwx SM=PRV
MALLOC metadata             10275c000-102760000    [   16K    16K    16K     0K] rw-/rwx SM=COW
__DATA                      102954000-102958000    [   16K    16K    16K     0K] rw-/rw- SM=COW          /usr/lib/dyld
MALLOC_TINY                 12e600000-12e700000    [ 1024K    16K    16K    16K] rw-/rwx SM=PRV          MallocHelperZone_0x10271c000
MALLOC_SMALL                12e800000-12f000000    [ 8192K    16K    16K    16K] rw-/rwx SM=PRV          MallocHelperZone_0x10271c000
Stack                       16d004000-16d800000    [ 8176K    16K    16K    16K] rw-/rwx SM=COW          thread 0
Stack                       16d804000-16d88c000    [  544K     0K     0K    16K] rw-/rwx SM=COW          thread 1
```

```
Stack                    16d890000-16d918000    [  544K     0K     0K    16K] rw-/rwx SM=COW          thread 2
Stack                    16d91c000-16d9a4000    [  544K     0K     0K    16K] rw-/rwx SM=COW          thread 3
Stack                    16d9a8000-16da30000    [  544K     0K     0K    16K] rw-/rwx SM=COW          thread 4
Stack                    16da34000-16dabc000    [  544K     0K     0K    16K] rw-/rwx SM=COW          thread 5

[...]
```

Note: Page size on the Mx platform is 16KB.

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

```
(lldb) image dump sections App1
Sections for '/Users/training/AMCDA-Dumps/Apps/App1/Build/Products/Release/App1' (arm64):
  SectID     Type             Load Address                             Perm File Off.  File Size  Flags      Section Name
  ---------- ---------------- ---------------------------------------  ---- ---------- ---------- ---------- ------------------------
  0x00000100 container        [0x0000000000000000-0x0000000100000000)* ---  0x00000000 0x00000000 0x00000000 App1.__PAGEZERO
  0x00000200 container        [0x0000000102600000-0x0000000102604000)  r-x  0x00000000 0x00004000 0x00000000 App1.__TEXT
  0x00000001 code             [0x0000000102603d5c-0x0000000102603fa0)  r-x  0x00003d5c 0x00000244 0x80000400 App1.__TEXT.__text
  0x00000002 code             [0x0000000102603fa0-0x0000000102603fb8)  r-x  0x00003fa0 0x00000018 0x80000408 App1.__TEXT.__stubs
  0x00000003 compact-unwind   [0x0000000102603fb8-0x0000000102604000)  r-x  0x00003fb8 0x00000048 0x00000000 App1.__TEXT.__unwind_info
  0x00000300 container        [0x0000000102604000-0x0000000102608000)  rw-  0x00004000 0x00004000 0x00000010 App1.__DATA_CONST
  0x00000004 data-ptrs        [0x0000000102604000-0x0000000102604010)  rw-  0x00004000 0x00000010 0x00000006 App1.__DATA_CONST.__got
  0x00000400 container        [0x0000000102608000-0x0000000102610000)  r--  0x00008000 0x000052c0 0x00000000 App1.__LINKEDIT
```

Note: We don't see the __DATA section because it is possibly combined with thread 0 stack region since the address of the *environ* variable can be found at the top of that region:

```
(lldb) p/x environ
(void *) $0 = 0x000000016d7ffa88
```

10. Dump the last 4KB of the stack region (512 8-byte values = 4096 or 0x1000 bytes) with possible symbolic information (we can find the *environ* variable address there too):

```
(lldb) x/512a 0x000000016d800000-0x1000
error: Normally, 'memory read' will not read over 1024 bytes of data.
error: Please use --force to override this restriction just once.
error: or set target.max-memory-read-size if you will often need a larger limit.

(lldb) x/512a 0x000000016d800000-0x1000 --force
0x16d7ff000: 0x0000000248e0ce34
0x16d7ff008: 0x0000000248e01bb4
0x16d7ff010: 0x0000000248e024fc
0x16d7ff018: 0x0000000248e020dc
0x16d7ff020: 0x0000000248e0cf0c
0x16d7ff028: 0x0000000248e0240c
0x16d7ff030: 0x000000016d7ff900
0x16d7ff038: 0x925b8001028e19c8 (0x00000001028e19c8) dyld`dyld4::prepare(dyld4::APIs&, dyld3::MachOAnalyzer const*) + 2124
0x16d7ff040: 0x0000000102610bf0
0x16d7ff048: 0x0000000248e0d5f4
0x16d7ff050: 0x0000000248e0d52c
0x16d7ff058: 0x0000000248e0cd04
0x16d7ff060: 0x0000000248e0d054
0x16d7ff068: 0x0000000248e0cf7c
0x16d7ff070: 0x0000000248e01d94
0x16d7ff078: 0x0000000248e01f94
0x16d7ff080: 0x0000000248e0276c
0x16d7ff088: 0x0000000248e0d38c
0x16d7ff090: 0x0000000248e0cb7c
0x16d7ff098: 0x0000000248e0b384
0x16d7ff0a0: 0x0000000248e0d2cc
0x16d7ff0a8: 0x0000000248e054ec
0x16d7ff0b0: 0x0000000248e01a6c
0x16d7ff0b8: 0x0000000248e022a4
0x16d7ff0c0: 0x0000000248e0d124
0x16d7ff0c8: 0x0000000248e09fd4
0x16d7ff0d0: 0x0000000248e08fb4
0x16d7ff0d8: 0x0000000248e0c364
0x16d7ff0e0: 0x0000000248e03f04
0x16d7ff0e8: 0x0000000248e0d44c
0x16d7ff0f0: 0x0000000248e021a4
0x16d7ff0f8: 0x0000000248e0295c
0x16d7ff100: 0x0000000248e0ca04
0x16d7ff108: 0x0000000248e01e94
0x16d7ff110: 0x0000000248e0521c
```

```
0x16d7ff118: 0x0000000248e044e4
0x16d7ff120: 0x0000000248e16574
0x16d7ff128: 0x0000000248e0a0bc
0x16d7ff130: 0x0000000248e0d1ec
0x16d7ff138: 0x0000000248e0258c
0x16d7ff140: 0x0000000248e0287c
0x16d7ff148: 0x0000000248e02664
0x16d7ff150: 0x0000000248e06d7c
0x16d7ff158: 0x0000000248e01c64
0x16d7ff160: 0x0000000248e0cde4
0x16d7ff168: 0x0000000248e01b64
0x16d7ff170: 0x0000000248e024ac
0x16d7ff178: 0x0000000248e0208c
0x16d7ff180: 0x000000016d7ff2d0
0x16d7ff188: 0x000000016d7ff1a0
0x16d7ff190: 0x000000016d7ff900
0x16d7ff198: 0x01568001028e1e94 (0x00000001028e1e94) dyld`dyld4::prepare(dyld4::APIs&, dyld3::MachOAnalyzer const*) + 3352
0x16d7ff1a0: 0x000000016d7ff2d0
0x16d7ff1a8: 0x0000000000000000
0x16d7ff1b0: 0x0000000000000000
0x16d7ff1b8: 0x000000016d7ff1a0
0x16d7ff1c0: 0x0000000000000000
0x16d7ff1c8: 0x0000000000000000
0x16d7ff1d0: 0x0000000000000000
0x16d7ff1d8: 0x0000000000000000
0x16d7ff1e0: 0x0000000000000000
0x16d7ff1e8: 0x0000000000000000
0x16d7ff1f0: 0x000000001f07000c
0x16d7ff1f8: 0x0000000000000000
0x16d7ff200: 0x0000000000000000
0x16d7ff208: 0x0000000000000000
0x16d7ff210: 0x0000000000000000
0x16d7ff218: 0x0000000000000000
0x16d7ff220: 0x0000000000000000
0x16d7ff228: 0x0000000000000000
0x16d7ff230: 0x0000000000000000
0x16d7ff238: 0x0000000000000000
0x16d7ff240: 0x0000000000000000
0x16d7ff248: 0x0000000000000000
0x16d7ff250: 0x0000000000000000
0x16d7ff258: 0x0000000000000000
0x16d7ff260: 0x000000016d7ff040
0x16d7ff268: 0x000000000000002a
0x16d7ff270: 0x0000000000000001
0x16d7ff278: 0x0000000000000000
0x16d7ff280: 0x0000000102954310 dyld`_NSConcreteStackBlock
0x16d7ff288: 0x0000000042000000
0x16d7ff290: 0x00000001028e2888 dyld`invocation function for block in dyld4::prepare(dyld4::APIs&, dyld3::MachOAnalyzer const*)
0x16d7ff298: 0x000000010293c250 dyld`__block_descriptor_tmp.24
0x16d7ff2a0: 0x000000016d7ff2a8
0x16d7ff2a8: 0x0000000000000000
0x16d7ff2b0: 0x000000016d7ff2a8
0x16d7ff2b8: 0x0000004002000000
0x16d7ff2c0: 0x00000001028e286c dyld`__Block_byref_object_copy_.18
0x16d7ff2c8: 0x00000001028e2880 dyld`__Block_byref_object_dispose_.19
0x16d7ff2d0: 0x0000000000000000
0x16d7ff2d8: 0x0000000000000000
0x16d7ff2e0: 0x0000000000000000
0x16d7ff2e8: 0x0000000102954310 dyld`_NSConcreteStackBlock
0x16d7ff2f0: 0x0000000042000000
0x16d7ff2f8: 0x00000001028e26c0 dyld`invocation function for block in dyld4::prepare(dyld4::APIs&, dyld3::MachOAnalyzer const*)
0x16d7ff300: 0x000000010293c220 dyld`__block_descriptor_tmp.17
0x16d7ff308: 0x000000016d7ff378
0x16d7ff310: 0x0000000102610060 -> 0x000000010293e400 dyld`vtable for dyld4::APIs + 16
0x16d7ff318: 0x0101000000000101
0x16d7ff320: 0x0000000101000000
0x16d7ff328: 0x000000016d7ff368
0x16d7ff330: 0x0000000000000000
0x16d7ff338: 0x0000000000000000
0x16d7ff340: 0x0100000000000101
0x16d7ff348: 0x0000000101000000
0x16d7ff350: 0x000000016d7ff368
0x16d7ff358: 0x0000000000000000
0x16d7ff360: 0x0000000000000000
0x16d7ff368: 0x0000000000000000
0x16d7ff370: 0x0000000102610bf0
0x16d7ff378: 0x0000000000000000
0x16d7ff380: 0x000000016d7ff378
0x16d7ff388: 0x0000005002000000
0x16d7ff390: 0x00000001028e25a0 dyld`__Block_byref_object_copy_.11
0x16d7ff398: 0x00000001028e25bc dyld`__Block_byref_object_dispose_.12
0x16d7ff3a0: 0x000000016d7ff3f0
0x16d7ff3a8: 0x0000000000000010
0x16d7ff3b0: 0x0000000000000001
0x16d7ff3b8: 0x0000000000000000
0x16d7ff3c0: 0x0000000000000000
```

```
0x16d7ff3c8: 0x0000000102610bf0
0x16d7ff3d0: 0x0000000000000000
0x16d7ff3d8: 0x0000000000000000
0x16d7ff3e0: 0x0000000000000000
0x16d7ff3e8: 0x000000010293c078 dyld`dyld4::sConfigBuffer + 8
0x16d7ff3f0: 0x0000000102610bf0
0x16d7ff3f8: 0x000000016d7ffa68 -> 0x0000000102600000 App1`_mh_execute_header
0x16d7ff400: 0x0000001a01000012
0x16d7ff408: 0x0000000102954300 dyld`dyld4::sSyscallDelegate
0x16d7ff410: 0x000000016d7ff850 -> 0x0000000102603eec App1`main
0x16d7ff418: 0x45748001028e375c (0x00000001028e375c)
dyld`dyld4::ProcessConfig::Process::pathFromFileHexStrings(dyld4::SyscallDelegate&, char const*) + 124
0x16d7ff420: 0x000000016d7fff64
0x16d7ff428: 0x62696c2f7273752f
0x16d7ff430: 0x000000016d7ff5c8
0x16d7ff438: 0x000000016d7ff5a0 -> 0x0000000102954310 dyld`_NSConcreteStackBlock
0x16d7ff440: 0x000000018e887cc0
0x16d7ff448: 0x000000016d7ff54f
0x16d7ff450: 0x000000016d7ff500
0x16d7ff458: 0x5a52800102911ed8 (0x0000000102911ed8) dyld`invocation function for block in dyld3::MachOFile::forEachSection(void
(dyld3::MachOFile::SectionInfo const&, bool, bool&) block_pointer) const + 528
0x16d7ff460: 0x000000002e5d8000
0x16d7ff468: 0x0000000024151f49
0x16d7ff470: 0x0000000213db8000
0x16d7ff478: 0x00000001028dc000
0x16d7ff480: 0x66306db6e99fee38
0x16d7ff488: 0xc644690ddf6d5c8c
0x16d7ff490: 0x0000000000003eec
0x16d7ff498: 0x0000000000000000
0x16d7ff4a0: 0x000000000000002a
0x16d7ff4a8: 0x0000000000002330
0x16d7ff4b0: 0x000000018e887df8
0x16d7ff4b8: 0x0000000100000000
0x16d7ff4c0: 0x0000000000000006
0x16d7ff4c8: 0x0000000000000000
0x16d7ff4d0: 0x000000018e888128
0x16d7ff4d8: 0x000000018e887020
0x16d7ff4e0: 0x0000000000000017
0x16d7ff4e8: 0x000000016d7ff5a0 -> 0x0000000102954310 dyld`_NSConcreteStackBlock
0x16d7ff4f0: 0x000000018e887000
0x16d7ff4f8: 0x000000016d7ff608
0x16d7ff500: 0x000000016d7ff590
0x16d7ff508: 0x96238001028ddf98 (0x00000001028ddf98) dyld`dyld3::MachOFile::forEachLoadCommand(Diagnostics&, void (load_command const*,
bool&) block_pointer) const + 168
0x16d7ff510: 0x0000000000000000
0x16d7ff518: 0x000000016d7ff5d8
0x16d7ff520: 0x000000018e887000
0x16d7ff528: 0x000000016d7ff628
0x16d7ff530: 0x000000016d7ff5c0
0x16d7ff538: 0x54428001028ddf98 (0x00000001028ddf98) dyld`dyld3::MachOFile::forEachLoadCommand(Diagnostics&, void (load_command const*,
bool&) block_pointer) const + 168
0x16d7ff540: 0x0000000000000000
0x16d7ff548: 0x0000000000000000
0x16d7ff550: 0x0000000000000000
0x16d7ff558: 0x0000000000000000
0x16d7ff560: 0x000000016d7ff738
0x16d7ff568: 0x000000010293c0fc dyld`dyld4::sConfigBuffer + 140
0x16d7ff570: 0x000000010293c078 dyld`dyld4::sConfigBuffer + 8
0x16d7ff578: 0x000000010293c0f0 dyld`dyld4::sConfigBuffer + 128
0x16d7ff580: 0x000000018e887000
0x16d7ff588: 0x000000016d7ff658 -> 0x0000000102954310 dyld`_NSConcreteStackBlock
0x16d7ff590: 0x000000016d7ff640
0x16d7ff598: 0x6148800102911c9c (0x0000000102911c9c) dyld`dyld3::MachOFile::forEachSection(void (dyld3::MachOFile::SectionInfo const&,
bool, bool&) block_pointer) const + 220
0x16d7ff5a0: 0x0000000102954310 dyld`_NSConcreteStackBlock
0x16d7ff5a8: 0x0000000042000000
0x16d7ff5b0: 0x0000000102911cc8 dyld`invocation function for block in dyld3::MachOFile::forEachSection(void
(dyld3::MachOFile::SectionInfo const&, bool, bool&) block_pointer) const
0x16d7ff5b8: 0x000000010293efd8 dyld`__block_descriptor_tmp.87
0x16d7ff5c0: 0x000000016d7ff658 -> 0x0000000102954310 dyld`_NSConcreteStackBlock
0x16d7ff5c8: 0x000000016d7ff5e8
0x16d7ff5d0: 0x000000018e887000
0x16d7ff5d8: 0x000000016d7ff614
0x16d7ff5e0: 0x0000000042000000
0x16d7ff5e8: 0x0000000000000000
0x16d7ff5f0: 0x000000016d7ff5e8
0x16d7ff5f8: 0x0000002000000000
0x16d7ff600: 0x000000010000000a
0x16d7ff608: 0x0000000000000000
0x16d7ff610: 0x626f5f5f6d7ff608
0x16d7ff618: 0x6567616d695f636a
0x16d7ff620: 0x000000006f666e69
0x16d7ff628: 0x37f570e601fc0072
0x16d7ff630: 0x000000016d7ff688
0x16d7ff638: 0x000000018e887000
0x16d7ff640: 0x000000016d7ff6c0
```

```
0x16d7ff648: 0xdd5e00010290fd90 (0x000000010290fd90) dyld`DyldSharedCache::objcOpt() const + 216
0x16d7ff650: 0x000000016d7ff680
0x16d7ff658: 0x0000000102954310 dyld`_NSConcreteStackBlock
0x16d7ff660: 0x0000000042000000
0x16d7ff668: 0x000000010290fddc dyld`invocation function for block in DyldSharedCache::objcOpt() const
0x16d7ff670: 0x000000010293ebf8 dyld`__block_descriptor_tmp.41
0x16d7ff678: 0x000000016d7ff688
0x16d7ff680: 0x000000000e6dc000
0x16d7ff688: 0x0000000000000000
0x16d7ff690: 0x000000016d7ff688
0x16d7ff698: 0x0000002000000000
0x16d7ff6a0: 0x000000018e8babe0 libobjc.A.dylib`_objc_opt_data
0x16d7ff6a8: 0x000000060293c0f0
0x16d7ff6b0: 0x0000000102954300 dyld`dyld4::sSyscallDelegate
0x16d7ff6b8: 0x000000010293c110 dyld`dyld4::sConfigBuffer + 160
0x16d7ff6c0: 0x000000016d7ff8c0
0x16d7ff6c8: 0x5c1d0001028e4024 (0x00000001028e4024) dyld`dyld4::ProcessConfig::DyldCache::DyldCache(dyld4::ProcessConfig::Process&,
dyld4::ProcessConfig::Security const&, dyld4::ProcessConfig::Logging const&, dyld4::SyscallDelegate&) + 548
0x16d7ff6d0: 0x0000000102600000 App1`_mh_execute_header
0x16d7ff6d8: 0x000000016d7ff7d8
0x16d7ff6e0: 0x0000000000000000
0x16d7ff6e8: 0x0000000100000000
0x16d7ff6f0: 0x0000000000000001
0x16d7ff6f8: 0x0000000000000000
0x16d7ff700: 0x0000000000000000
0x16d7ff708: 0x0000000000000000
0x16d7ff710: 0x000000010293c0f0 dyld`dyld4::sConfigBuffer + 128
0x16d7ff718: 0x000000010293c078 dyld`dyld4::sConfigBuffer + 8
0x16d7ff720: 0x0000000102600000 App1`_mh_execute_header
0x16d7ff728: 0x000000016d7ff7f8
0x16d7ff730: 0x0000000000000000
0x16d7ff738: 0x0000000000000000
0x16d7ff740: 0x0000000000000000
0x16d7ff748: 0x0000000000000000
0x16d7ff750: 0x0000000000000000
0x16d7ff758: 0x0000000000000000
0x16d7ff760: 0x0000000000000000
0x16d7ff768: 0x000000010293c070 dyld`dyld4::sConfigBuffer
0x16d7ff770: 0x000000016d7ff840
0x16d7ff778: 0x000000016d7ff800
0x16d7ff780: 0x000000016d7ff7e0
0x16d7ff788: 0x791080018e9d02f8 (0x000000018e9d02f8) libsystem_kernel.dylib`clock_get_time + 100
0x16d7ff790: 0x0000000000007c000
0x16d7ff798: 0x0000002c00001200
0x16d7ff7a0: 0x0000050700000000
0x16d7ff7a8: 0x0000044c00000000
0x16d7ff7b0: 0x0000000100000000
0x16d7ff7b8: 0x000190f500000000
0x16d7ff7c0: 0x000000001ad957d8
0x16d7ff7c8: 0x0000020300000008
0x16d7ff7d0: 0x000000016d7ff840
0x16d7ff7d8: 0x000000016d7ff830
0x16d7ff7e0: 0x000000016d7ff820
0x16d7ff7e8: 0x173400018e8d6f60 (0x000000018e8d6f60) libsystem_c.dylib`nanosleep + 116
0x16d7ff7f0: 0x000000018ea0cde8 libsystem_pthread.dylib`_pthread_attr_default
0x16d7ff7f8: 0x0000000000000000
0x16d7ff800: 0x1ad957d8000190f5
0x16d7ff808: 0x0000000000000000
0x16d7ff810: 0x0000000102603eec App1`main
0x16d7ff818: 0x000000007fffffff
0x16d7ff820: 0x000000016d7ff860
0x16d7ff828: 0x706400018e8e1b78 (0x000000018e8e1b78) libsystem_c.dylib`sleep + 52
0x16d7ff830: 0x0000000000000000
0x16d7ff838: 0x0000000000000000
0x16d7ff840: 0x000000007fffffff
0x16d7ff848: 0x0000000000000000
0x16d7ff850: 0x0000000102603eec App1`main
0x16d7ff858: 0x00000000ffffffff
0x16d7ff860: 0x000000016d7ff8a0
0x16d7ff868: 0x772700018e8e1b90 (0x000000018e8e1b90) libsystem_c.dylib`sleep + 76
0x16d7ff870: 0x0000000000000000
0x16d7ff878: 0x0000000000000000
0x16d7ff880: 0x0000000000000000
0x16d7ff888: 0x000000010293c070 dyld`dyld4::sConfigBuffer
0x16d7ff890: 0x0000000102603eec App1`main
0x16d7ff898: 0x0000000102610060 -> 0x000000010293e400 dyld`vtable for dyld4::APIs + 16
0x16d7ff8a0: 0x000000016d7ff900
0x16d7ff8a8: 0x1131800102603f90 (0x0000000102603f90) App1`main + 164
0x16d7ff8b0: 0x0000000000000000
0x16d7ff8b8: 0x0000000000000000
0x16d7ff8c0: 0x0000000000000000
0x16d7ff8c8: 0x000000016dab7000
0x16d7ff8d0: 0x000000016da2b000
0x16d7ff8d8: 0x000000016d99f000
0x16d7ff8e0: 0x000000016d913000
0x16d7ff8e8: 0x000000016d887000
```

```
0x16d7ff8f0: 0x000000016d7ffa78
0x16d7ff8f8: 0x0000000000000001
0x16d7ff900: 0x000000016d7ffa50
0x16d7ff908: 0x0000001028e108c dyld`start + 520
0x16d7ff910: 0x0000000000000000
0x16d7ff918: 0x0000000000000000
0x16d7ff920: 0x0000000000000000
0x16d7ff928: 0x0000000000000000
0x16d7ff930: 0x0000000000000000
0x16d7ff938: 0x0000000000000000
0x16d7ff940: 0x0000000102940138 dyld`_os_lock_type_unfair
0x16d7ff948: 0x0000000000000000
0x16d7ff950: 0x000000004d55545a
0x16d7ff958: 0x000020a000000000
0x16d7ff960: 0x4d55545a00000000
0x16d7ff968: 0x0000000000000000
0x16d7ff970: 0x0000000000000000
0x16d7ff978: 0xffffffffffffffff
0x16d7ff980: 0xffffffffe928006af
0x16d7ff988: 0x4d55545a4d55545a
0x16d7ff990: 0x0000000000000000
0x16d7ff998: 0x0000000000000000
0x16d7ff9a0: 0x0000000102954310 dyld`_NSConcreteStackBlock
0x16d7ff9a8: 0x0000000040000000
0x16d7ff9b0: 0x0000001028e1154 dyld`__start_block_invoke.3
0x16d7ff9b8: 0x000000010293c200 dyld`__block_descriptor_tmp.5
0x16d7ff9c0: 0x0000001028dc000
0x16d7ff9c8: 0x0000000102954310 dyld`_NSConcreteStackBlock
0x16d7ff9d0: 0x0000000042000000
0x16d7ff9d8: 0x0000001028e110c dyld`__start_block_invoke
0x16d7ff9e0: 0x000000010293c1d0 dyld`__block_descriptor_tmp
0x16d7ff9e8: 0x000000016d7ffa00
0x16d7ff9f0: 0x0000001028dc000
0x16d7ff9f8: 0x0000001028dc000
0x16d7ffa00: 0x0000000000000000
0x16d7ffa08: 0x000000016d7ffa00
0x16d7ffa10: 0x0000003002000000
0x16d7ffa18: 0x0000001028e10f8 dyld`__Block_byref_object_copy_
0x16d7ffa20: 0x0000001028e1104 dyld`__Block_byref_object_dispose_
0x16d7ffa28: 0x0000000000000000
0x16d7ffa30: 0x0000000000000000
0x16d7ffa38: 0x0000000000000000
0x16d7ffa40: 0x0000000000000000
0x16d7ffa48: 0x0000000000000000
0x16d7ffa50: 0x0000000000000000
0x16d7ffa58: 0x6527800000000000
0x16d7ffa60: 0x0000000000000000
0x16d7ffa68: 0x0000000102600000 App1`_mh_execute_header
0x16d7ffa70: 0x0000000000000001
0x16d7ffa78: 0x000000016d7ffba8
0x16d7ffa80: 0x0000000000000000
0x16d7ffa88: 0x000000016d7ffbaf
0x16d7ffa90: 0x000000016d7ffbcb
0x16d7ffa98: 0x000000016d7ffbda
0x16d7ffaa0: 0x000000016d7ffbee
0x16d7ffaa8: 0x000000016d7ffc27
0x16d7ffab0: 0x000000016d7ffc40
0x16d7ffab8: 0x000000016d7ffc75
0x16d7ffac0: 0x000000016d7ffc83
0x16d7ffac8: 0x000000016d7ffcc5
0x16d7ffad0: 0x000000016d7ffd0e
0x16d7ffad8: 0x000000016d7ffd36
0x16d7ffae0: 0x000000016d7ffd77
0x16d7ffae8: 0x000000016d7ffd85
0x16d7ffaf0: 0x000000016d7ffd98
0x16d7ffaf8: 0x000000016d7ffda0
0x16d7ffb00: 0x000000016d7ffdb5
0x16d7ffb08: 0x000000016d7ffdc6
0x16d7ffb10: 0x000000016d7ffe02
0x16d7ffb18: 0x000000016d7ffe13
0x16d7ffb20: 0x0000000000000000
0x16d7ffb28: 0x000000016d7ffb90
0x16d7ffb30: 0x000000016d7ffe60
0x16d7ffb38: 0x000000016d7ffe70
0x16d7ffb40: 0x000000016d7ffe8f
0x16d7ffb48: 0x000000016d7ffec4
0x16d7ffb50: 0x000000016d7ffee0
0x16d7ffb58: 0x000000016d7fff16
0x16d7ffb60: 0x000000016d7fff3c
0x16d7ffb68: 0x000000016d7fff65
0x16d7ffb70: 0x000000016d7fffa0
0x16d7ffb78: 0x000000016d7fffdd
0x16d7ffb80: 0x000000016d7ffeb
0x16d7ffb88: 0x0000000000000000
0x16d7ffb90: 0x6261747563657865
0x16d7ffb98: 0x3d687461705f656c
```

```
0x16d7ffba0: 0x0000317070412f2e
0x16d7ffba8: 0x5400317070412f2e
0x16d7ffbb0: 0x474f52505f4d5245
0x16d7ffbb8: 0x6c7070413d4d4152
0x16d7ffbc0: 0x6e696d7265545f65
0x16d7ffbc8: 0x4c4c454853006c61
0x16d7ffbd0: 0x737a2f6e69622f3d
0x16d7ffbd8: 0x783d4d5245540068
0x16d7ffbe0: 0x3635322d6d726574
0x16d7ffbe8: 0x4d5400726f6c6f63
0x16d7ffbf0: 0x61762f3d52494450
0x16d7ffbf8: 0x7265646c6f662f72
0x16d7ffc00: 0x7672382f6c6e2f73
0x16d7ffc08: 0x7836306435746b72
0x16d7ffc10: 0x5f35626e6d387231
0x16d7ffc18: 0x3030306d5f32636c
0x16d7ffc20: 0x54002f542f6e6730
0x16d7ffc28: 0x474f52505f4d5245
0x16d7ffc30: 0x535245565f4d4152
0x16d7ffc38: 0x003534343d4e4f49
0x16d7ffc40: 0x5345535f4d524554
0x16d7ffc48: 0x3d44495f4e4f4953
0x16d7ffc50: 0x3534393231354330
0x16d7ffc58: 0x39342d413842442d
0x16d7ffc60: 0x2d453337422d3945
0x16d7ffc68: 0x3831383030353634
0x16d7ffc70: 0x4553550042354644
0x16d7ffc78: 0x696e696172743d52
0x16d7ffc80: 0x415f48535300676e
0x16d7ffc88: 0x4b434f535f485455
0x16d7ffc90: 0x7461766972702f3d
0x16d7ffc98: 0x6f632f706d742f65
0x16d7ffca0: 0x2e656c7070612e6d
0x16d7ffca8: 0x2e6468636e75616c
0x16d7ffcb0: 0x4a4b4b695a385a4d
0x16d7ffcb8: 0x657473694c2f6579
0x16d7ffcc0: 0x544150007372656e
0x16d7ffcc8: 0x6c2f7273752f3d48
0x16d7ffcd0: 0x6e69622f6c61636f
0x16d7ffcd8: 0x69622f7273752f3a
0x16d7ffce0: 0x2f3a6e69622f3a6e
0x16d7ffce8: 0x6e6962732f727375
0x16d7ffcf0: 0x2f3a6e6962732f3a
0x16d7ffcf8: 0x2f7972617262694c
0x16d7ffd00: 0x73752f656c707041
0x16d7ffd08: 0x5f5f006e69622f72
0x16d7ffd10: 0x656c646e75424643
0x16d7ffd18: 0x696669746e656449
0x16d7ffd20: 0x612e6d6f633d7265
0x16d7ffd28: 0x7265542e656c7070
0x16d7ffd30: 0x5750006c616e696d
0x16d7ffd38: 0x73726573552f3d44
0x16d7ffd40: 0x6e696e696172742f
0x16d7ffd48: 0x2d4144434d412f67
0x16d7ffd50: 0x70412f73706d7544
0x16d7ffd58: 0x2f317070412f7370
0x16d7ffd60: 0x72502f646c697542
0x16d7ffd68: 0x522f73746375646f
0x16d7ffd70: 0x5800657361656c65
0x16d7ffd78: 0x5347414c465f4350
0x16d7ffd80: 0x435058003078303d
0x16d7ffd88: 0x4543495652455f5f
0x16d7ffd90: 0x00303d454d414e5f
0x16d7ffd98: 0x00313d4c564c4853
0x16d7ffda0: 0x73552f3d454d4f48
0x16d7ffda8: 0x696172742f737265
0x16d7ffdb0: 0x474f4c00676e696e
0x16d7ffdb8: 0x6172743d454d414e
0x16d7ffdc0: 0x4c4f00676e696e69
0x16d7ffdc8: 0x73552f3d44575044
0x16d7ffdd0: 0x696172742f737265
0x16d7ffdd8: 0x434d412f676e696e
0x16d7ffde0: 0x73706d75442d4144
0x16d7ffde8: 0x70412f737070412f
0x16d7ffdf0: 0x646c6975422f3170
0x16d7ffdf8: 0x746375646f72502f
0x16d7ffe00: 0x653d474e414c0073
0x16d7ffe08: 0x4654552e45495f6e
0x16d7ffe10: 0x73552f3d5f00382d
0x16d7ffe18: 0x696172742f737265
0x16d7ffe20: 0x434d412f676e696e
0x16d7ffe28: 0x73706d75442d4144
0x16d7ffe30: 0x70412f737070412f
0x16d7ffe38: 0x646c6975422f3170
0x16d7ffe40: 0x746375646f72502f
0x16d7ffe48: 0x7361656c65522f73
```

```
0x16d7ffe50: 0x317070412f2e2f65
0x16d7ffe58: 0x0000000000000000
0x16d7ffe60: 0x0000000000000000
0x16d7ffe68: 0x0000000000000000
0x16d7ffe70: 0x0000000000000000
0x16d7ffe78: 0x0000000000000000
0x16d7ffe80: 0x0000000000000000
0x16d7ffe88: 0x0000000000000000
0x16d7ffe90: 0x0000000000000000
0x16d7ffe98: 0x0000000000000000
0x16d7ffea0: 0x0000000000000000
0x16d7ffea8: 0x0000000000000000
0x16d7ffeb0: 0x0000000000000000
0x16d7ffeb8: 0x0000000000000000
0x16d7ffec0: 0x5f72747000000000
0x16d7ffec8: 0x00003d65676e756d
0x16d7ffed0: 0x0000000000000000
0x16d7ffed8: 0x0000000000000000
0x16d7ffee0: 0x6174735f6e69616d
0x16d7ffee8: 0x00000000003d6b63
0x16d7ffef0: 0x0000000000000000
0x16d7ffef8: 0x0000000000000000
0x16d7fff00: 0x0000000000000000
0x16d7fff08: 0x0000000000000000
0x16d7fff10: 0x7865000000000000
0x16d7fff18: 0x656c626174756365
0x16d7fff20: 0x78303d656c69665f
0x16d7fff28: 0x3030303031306131
0x16d7fff30: 0x64623178302c3231
0x16d7fff38: 0x646c796400313864
0x16d7fff40: 0x78303d656c69665f
0x16d7fff48: 0x3030303031306131
0x16d7fff50: 0x66666678302c3231
0x16d7fff58: 0x6430303066666666
0x16d7fff60: 0x6578650034393863
0x16d7fff68: 0x5f656c6261747563
0x16d7fff70: 0x373d687361686463
0x16d7fff78: 0x3063396235643761
0x16d7fff80: 0x3236386664643335
0x16d7fff88: 0x6461393065343935
0x16d7fff90: 0x6666333639636130
0x16d7fff98: 0x0030373063656532
0x16d7fffa0: 0x6261747563657865
0x16d7fffa8: 0x68746f6f625f656c
0x16d7fffb0: 0x373730643d687361
0x16d7fffb8: 0x3039646339313036
0x16d7fffc0: 0x3262323863313764
0x16d7fffc8: 0x3432656438346438
0x16d7fffd0: 0x6132663936306536
0x16d7fffd8: 0x6d72610064633832
0x16d7fffe0: 0x3d6962615f653436
0x16d7fffe8: 0x6f705f687400736f
0x16d7ffff0: 0x00000000003d7472
0x16d7ffff8: 0x0000000000000000
```

11. Dump the contents of memory pointed to by *environ* variable in null-terminated string format:

```
(lldb) x/20s 0x000000016d7ffbaf
0x16d7ffbaf: "TERM_PROGRAM=Apple_Terminal"
0x16d7ffbcb: "SHELL=/bin/zsh"
0x16d7ffbda: "TERM=xterm-256color"
0x16d7ffbee: "TMPDIR=/var/folders/nl/8rvrkt5d06x1r8mnb5_lc2_m0000gn/T/"
0x16d7ffc27: "TERM_PROGRAM_VERSION=445"
0x16d7ffc40: "TERM_SESSION_ID=0C512945-DB8A-49E9-B73E-46500818DF5B"
0x16d7ffc75: "USER=training"
0x16d7ffc83: "SSH_AUTH_SOCK=/private/tmp/com.apple.launchd.MZ8ZiKKJye/Listeners"
0x16d7ffcc5: "PATH=/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/Library/Apple/usr/bin"
0x16d7ffd0e: "__CFBundleIdentifier=com.apple.Terminal"
0x16d7ffd36: "PWD=/Users/training/AMCDA-Dumps/Apps/App1/Build/Products/Release"
0x16d7ffd77: "XPC_FLAGS=0x0"
0x16d7ffd85: "XPC_SERVICE_NAME=0"
0x16d7ffd98: "SHLVL=1"
0x16d7ffda0: "HOME=/Users/training"
0x16d7ffdb5: "LOGNAME=training"
0x16d7ffdc6: "OLDPWD=/Users/training/AMCDA-Dumps/Apps/App1/Build/Products"
0x16d7ffe02: "LANG=en_IE.UTF-8"
```

```
0x16d7ffe13: "_=/Users/training/AMCDA-Dumps/Apps/App1/Build/Products/Release/./App1"
0x16d7ffe59: ""
```

12. Get the list of loaded modules:

```
(lldb) image list
[  0] 87587B20-46B7-331A-ABC0-F7ECB36E3DEB 0x0000000102600000 /Users/training/AMCDA-Dumps/Apps/App1/Build/Products/Release/App1
[  1] 38EE9FE9-B66D-3066-8C5C-6DDF0D6944C6 0x00000001028dc000 /usr/lib/dyld
[  2] 9232C168-6ECA-3B7D-B081-E7C46B379836 0x000000019956d000 /usr/lib/libSystem.B.dylib
[  3] 7E9E684F-57B6-3196-8AEC-908B46DEEBD4 0x0000000199567000 /usr/lib/system/libcache.dylib
[  4] FB7DF5AC-35DB-3B80-B2F6-BC69375390AE 0x0000000199525000 /usr/lib/system/libcommonCrypto.dylib
[  5] 68788078-BF1D-3CD1-91A7-4C59FD78FB75 0x000000019954e000 /usr/lib/system/libcompiler_rt.dylib
[  6] 654D0DA0-8277-361D-88DC-1430504B5436 0x0000000199545000 /usr/lib/system/libcopyfile.dylib
[  7] 2D00FEEC-7984-342B-9516-5D49C5D98204 0x000000018e78b000 /usr/lib/system/libcorecrypto.dylib
[  8] B3C7A004-1069-3171-B630-2C386A8B399C 0x000000018e840000 /usr/lib/system/libdispatch.dylib
[  9] F298A03D-5BC7-3BCA-8880-B956E52EAD01 0x000000018ea0e000 /usr/lib/system/libdyld.dylib
[ 10] 49D72074-0C58-317C-9B8B-762C13C0C084 0x000000019955d000 /usr/lib/system/libkeymgr.dylib
[ 11] ED4EE8AE-EA60-33B7-9676-E6119B7449E3 0x0000000199500000 /usr/lib/system/libmacho.dylib
[ 12] B887350E-B1C9-386C-B5EB-26F08C7C0152 0x0000000198bf0000 /usr/lib/system/libquarantine.dylib
[ 13] 157C8E50-D4A5-3DFC-8E0B-756E03E2082B 0x000000019955a000 /usr/lib/system/libremovefile.dylib
[ 14] EC04DA81-C3B5-3AC5-9042-7F07DF48B42A 0x0000000193b86000 /usr/lib/system/libsystem_asl.dylib
[ 15] 96462BD5-6BB4-3B69-89C9-2C70FA8852E7 0x000000018e72d000 /usr/lib/system/libsystem_blocks.dylib
[ 16] B25D2080-BB9E-38D6-8236-9CEF4B2F11A3 0x000000018e8c8000 /usr/lib/system/libsystem_c.dylib
[ 17] 4928F3C4-D438-354F-BA1C-0BD79F6475F3 0x0000000199552000 /usr/lib/system/libsystem_collections.dylib
[ 18] 3977B29D-624D-3DEE-94EF-95D29FB25252 0x0000000198085000 /usr/lib/system/libsystem_configuration.dylib
[ 19] D38210EF-8F23-380B-8B43-BB06A7305F67 0x00000001972de000 /usr/lib/system/libsystem_containermanager.dylib
[ 20] D5F19732-3AA0-3B93-9F25-318A27DE5AC5 0x000000019925d000 /usr/lib/system/libsystem_coreservices.dylib
[ 21] 5D456083-E21E-319D-9BA0-57702B3FB09B 0x000000019113f000 /usr/lib/system/libsystem_darwin.dylib
[ 22] 10A4374A-D15A-31C8-AC6F-2DCC10D06444 0x000000019955e000 /usr/lib/system/libsystem_dnssd.dylib
[ 23] 5B14B45B-A15B-31AD-93FB-BAC43C001A23 0x000000018e8c5000 /usr/lib/system/libsystem_featureflags.dylib
[ 24] 413C2A97-5D32-317D-8E32-4258B8E728CE 0x000000018ea23000 /usr/lib/system/libsystem_info.dylib
[ 25] 31A9DAE0-FB1F-3CB8-8AB6-CA5A1192DFD8 0x00000001994c8000 /usr/lib/system/libsystem_m.dylib
[ 26] 427675C6-C4BF-390A-AF93-B28DAC36876A 0x000000018e815000 /usr/lib/system/libsystem_malloc.dylib
[ 27] 4C9F32FA-D88C-3966-A2F0-7030841C8093 0x0000000193b14000 /usr/lib/system/libsystem_networkextension.dylib
[ 28] 12A2A8B6-80B4-36CA-8245-830EBEDEF1C4 0x0000000191598000 /usr/lib/system/libsystem_notify.dylib
[ 29] E49E2F05-0E01-352E-8CB7-276F8EF8E6D6 0x000000019f8c1000 /usr/lib/system/libsystem_product_info_filter.dylib
[ 30] 2A2EB0A4-9822-36D1-999B-181D1BB964B5 0x000000019808a000 /usr/lib/system/libsystem_sandbox.dylib
[ 31] 18F251D3-8C66-3B8B-817A-C124498478F4 0x0000000199557000 /usr/lib/system/libsystem_secinit.dylib
[ 32] A9D87740-9C1D-3468-BF60-720A8D713CBA 0x000000018e9c9000 /usr/lib/system/libsystem_kernel.dylib
[ 33] A57FE7FB-9FF8-30CE-97A2-625D6DA20D00 0x000000018ea1b000 /usr/lib/system/libsystem_platform.dylib
[ 34] 63C4EEF9-69A5-38B1-996E-8D31B66A051D 0x000000018ea01000 /usr/lib/system/libsystem_pthread.dylib
[ 35] 2906E453-3254-32EA-880E-14AEEF5D7ECD 0x00000001952ea000 /usr/lib/system/libsystem_symptoms.dylib
[ 36] B5524014-1A7F-3D07-8855-5E75A55E4A11 0x000000018e771000 /usr/lib/system/libsystem_trace.dylib
[ 37] D9CA1CE3-6B1A-3E2B-BBAD-9D9B1DB00F92 0x0000000199532000 /usr/lib/system/libunwind.dylib
[ 38] 21D05A8B-D782-3FA7-9A9D-55A45E6E6621 0x000000018e72f000 /usr/lib/system/libxpc.dylib
[ 39] EC96F0FA-6341-3E1D-BE54-49B544E17F7D 0x000000018e887000 /usr/lib/libobjc.A.dylib
[ 40] 4E8D8A11-4217-3D56-9D41-5426F7CF307C 0x000000018e9b1000 /usr/lib/libc++abi.dylib
[ 41] 7E53021F-FDCE-3EC9-8B4C-97AD3B21D02E 0x0000000019953d000 /usr/lib/liboah.dylib
[ 42] 3D1E6031-901D-3DF1-9E9A-F85FF1C2E803 0x000000018e94a000 /usr/lib/libc++.1.dylib
```