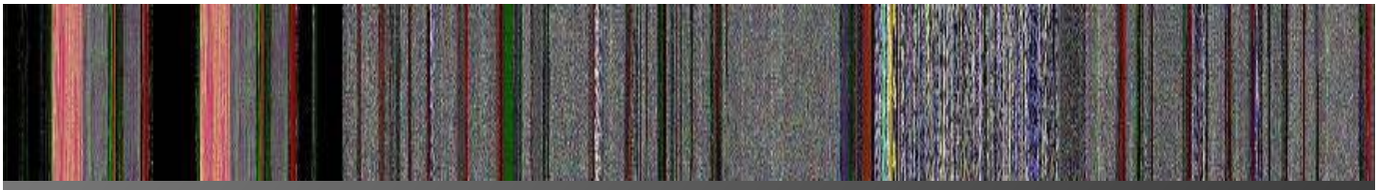

Defect
Detect

DUmps Binaries Logs INternals
School of Security

Linux
Core Dump Analysis
Accelerated

Fourth Edition

Dmitry Vostokov
Software Diagnostics Services

Accelerated Linux Core Dump Analysis: Training Course Transcript with GDB and WinDbg Practice Exercises, Fourth Edition

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: 978-1-912636-49-5 (Paperback)

Revision 4.00 (January 2025)

Contents

Exercise A1 (x64, GDB)

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

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

1. Load the core dump *App1.core.253* and *App1* executable from the x64/App1 directory:

```
~/ALCDA2/x64/App1$ gdb -c App1.core.253 -se App1
GNU gdb (Debian 8.2.1-2+b3) 8.2.1
Copyright (C) 2018 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".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from App1...done.
[New LWP 253]
[New LWP 254]
[New LWP 255]
[New LWP 256]
[New LWP 257]
[New LWP 258]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Core was generated by `./App1'.
#0  0x0000000000441a10 in nanosleep ()
[Current thread is 1 (Thread 0x21b3880 (LWP 253))]
```

2. Set logging to a file in case of lengthy output from some commands:

```
(gdb) set logging on App1.log
Copying output to App1.log.
```

3. List all threads:

```
(gdb) info threads
  Id   Target Id                      Frame
* 1    Thread 0x21b3880 (LWP 253)     0x0000000000441a10 in nanosleep ()
  2    Thread 0x7f0fc16fb700 (LWP 254) 0x0000000000441a10 in nanosleep ()
  3    Thread 0x7f0fc0efa700 (LWP 255) 0x0000000000441a10 in nanosleep ()
  4    Thread 0x7f0fc06f9700 (LWP 256) 0x0000000000441a10 in nanosleep ()
  5    Thread 0x7f0fbfef8700 (LWP 257) 0x0000000000441a10 in nanosleep ()
  6    Thread 0x7f0fbf6f7700 (LWP 258) 0x0000000000441a10 in nanosleep ()
```

4. Get the current thread stack trace:

```
(gdb) bt
#0  0x0000000000441a10 in nanosleep ()
#1  0x000000000044199a in sleep ()
#2  0x0000000000401d92 in main () at pthread_create.c:688
```

5. Get all thread stack traces:

```
(gdb) thread apply all bt

Thread 6 (Thread 0x7f0fbf6f7700 (LWP 258)):
#0  0x0000000000441a10 in nanosleep ()
#1  0x000000000044199a in sleep ()
#2  0x0000000000401cb7 in bar_five ()
#3  0x0000000000401cc8 in foo_five ()
#4  0x0000000000401ce1 in thread_five ()
#5  0x00000000004030d3 in start_thread (arg=<optimized out>) at pthread_create.c:486
#6  0x000000000044426f in clone ()

Thread 5 (Thread 0x7f0fbfef8700 (LWP 257)):
#0  0x0000000000441a10 in nanosleep ()
#1  0x000000000044199a in sleep ()
#2  0x0000000000401c78 in bar_four () at pthread_create.c:688
#3  0x0000000000401c89 in foo_four () at pthread_create.c:688
#4  0x0000000000401ca2 in thread_four () at pthread_create.c:688
#5  0x00000000004030d3 in start_thread (arg=<optimized out>) at pthread_create.c:486
#6  0x000000000044426f in clone ()

Thread 4 (Thread 0x7f0fc06f9700 (LWP 256)):
#0  0x0000000000441a10 in nanosleep ()
#1  0x000000000044199a in sleep ()
#2  0x0000000000401c39 in bar_three () at pthread_create.c:688
#3  0x0000000000401c4a in foo_three () at pthread_create.c:688
#4  0x0000000000401c63 in thread_three () at pthread_create.c:688
#5  0x00000000004030d3 in start_thread (arg=<optimized out>) at pthread_create.c:486
#6  0x000000000044426f in clone ()

Thread 3 (Thread 0x7f0fc0efa700 (LWP 255)):
#0  0x0000000000441a10 in nanosleep ()
#1  0x000000000044199a in sleep ()
#2  0x0000000000401bfa in bar_two () at pthread_create.c:688
#3  0x0000000000401c0b in foo_two () at pthread_create.c:688
#4  0x0000000000401c24 in thread_two () at pthread_create.c:688
#5  0x00000000004030d3 in start_thread (arg=<optimized out>) at pthread_create.c:486
#6  0x000000000044426f in clone ()

Thread 2 (Thread 0x7f0fc16fb700 (LWP 254)):
#0  0x0000000000441a10 in nanosleep ()
#1  0x000000000044199a in sleep ()
#2  0x0000000000401bbb in bar_one () at pthread_create.c:688
#3  0x0000000000401bcc in foo_one () at pthread_create.c:688
#4  0x0000000000401be5 in thread_one () at pthread_create.c:688
#5  0x00000000004030d3 in start_thread (arg=<optimized out>) at pthread_create.c:486
#6  0x000000000044426f in clone ()

Thread 1 (Thread 0x21b3880 (LWP 253)):
#0  0x0000000000441a10 in nanosleep ()
#1  0x000000000044199a in sleep ()
#2  0x0000000000401d92 in main () at pthread_create.c:688
```

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

```
(gdb) thread 2
[Switching to thread 2 (Thread 0x7f0fc16fb700 (LWP 254))]
#0  0x0000000000441a10 in nanosleep ()

(gdb) bt
#0  0x0000000000441a10 in nanosleep ()
#1  0x000000000044199a in sleep ()
#2  0x0000000000401bbb in bar_one () at pthread_create.c:688
#3  0x0000000000401bcc in foo_one () at pthread_create.c:688
#4  0x0000000000401be5 in thread_one () at pthread_create.c:688
#5  0x00000000004030d3 in start_thread (arg=<optimized out>) at pthread_create.c:486
#6  0x000000000044426f in clone ()

(gdb) info threads
  Id   Target Id                      Frame
  1    Thread 0x21b3880 (LWP 253)     0x0000000000441a10 in nanosleep ()
* 2    Thread 0x7f0fc16fb700 (LWP 254) 0x0000000000441a10 in nanosleep ()
  3    Thread 0x7f0fc0efa700 (LWP 255) 0x0000000000441a10 in nanosleep ()
  4    Thread 0x7f0fc06f9700 (LWP 256) 0x0000000000441a10 in nanosleep ()
  5    Thread 0x7f0fbfef8700 (LWP 257) 0x0000000000441a10 in nanosleep ()
  6    Thread 0x7f0fbf6f7700 (LWP 258) 0x0000000000441a10 in nanosleep ()
```

7. Check that *bar_one* called the *sleep* function by comparing the return address on the call stack from the disassembly output:

```
(gdb) disassemble bar_one
Dump of assembler code for function bar_one:
   0x0000000000401bad <+0>:     push   %rbp
   0x0000000000401bae <+1>:     mov    %rsp,%rbp
   0x0000000000401bb1 <+4>:     mov    $0xffffffff,%edi
   0x0000000000401bb6 <+9>:     callq  0x441960 <sleep>
   0x0000000000401bbb <+14>:    nop
   0x0000000000401bbc <+15>:    pop    %rbp
   0x0000000000401bbd <+16>:    retq
End of assembler dump.
```

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

8. Compare with Intel disassembly flavor:

```
(gdb) set disassembly-flavor intel

(gdb) disassemble bar_two
Dump of assembler code for function bar_one:
   0x0000000000401bad <+0>:     push   rbp
   0x0000000000401bae <+1>:     mov    rbp,rsp
   0x0000000000401bb1 <+4>:     mov    edi,0xffffffff
   0x0000000000401bb6 <+9>:     call   0x441960 <sleep>
   0x0000000000401bbb <+14>:    nop
   0x0000000000401bbc <+15>:    pop    rbp
   0x0000000000401bbd <+16>:    ret
End of assembler dump.

(gdb) set disassembly-flavor att
```

9. Get *App1* data section from the output of pmap (*App1.pmap.253*):

```
(gdb) ^Z
[2]+  Stopped                 gdb -c App1.core.253 -se App1

~/ALCDA2/x64/App1$ cat App1.pmap.253
253:   ./App1
0000000000400000      4K r---- App1
0000000000401000    588K r-x-- App1
0000000000494000    156K r---- App1
00000000004bc000     24K rw--- App1
00000000004c2000     24K rw---   [ anon ]
00000000021b3000    140K rw---   [ anon ]
00007f0fbeef7000      4K -----   [ anon ]
00007f0fbeef8000   8192K rw---   [ anon ]
00007f0fbf6f8000      4K -----   [ anon ]
00007f0fbf6f9000   8192K rw---   [ anon ]
00007f0fbfef9000      4K -----   [ anon ]
00007f0fbfefa000   8192K rw---   [ anon ]
00007f0fc06fa000      4K -----   [ anon ]
00007f0fc06fb000   8192K rw---   [ anon ]
00007f0fc0efb000      4K -----   [ anon ]
00007f0fc0efc000   8192K rw---   [ anon ]
00007ffdf4545000    132K rw---   [ stack ]
00007ffdf45c6000     16K r----   [ anon ]
00007ffdf45ca000      4K r-x--   [ anon ]
 total            42068K

~/ALCDA2/x64/App1$ fg
gdb -c App1.core.253 -se App1


(gdb)
```

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

```
(gdb) maintenance info sections
Exec file:
    `/home/coredump/ALCDA2/x64/App1/App1', file type elf64-x86-64.
 [0]     0x00400200->0x00400220 at 0x00000200: .note.ABI-tag ALLOC LOAD READONLY DATA HAS_CONTENTS
 [1]     0x00400220->0x00400244 at 0x00000220: .note.gnu.build-id ALLOC LOAD READONLY DATA HAS_CONTENTS
 [2]     0x00400248->0x004004d0 at 0x00000248: .rela.plt ALLOC LOAD READONLY DATA HAS_CONTENTS
 [3]     0x00401000->0x00401017 at 0x00001000: .init ALLOC LOAD READONLY CODE HAS_CONTENTS
 [4]     0x00401018->0x004010f0 at 0x00001018: .plt ALLOC LOAD READONLY CODE HAS_CONTENTS
 [5]     0x004010f0->0x004933d0 at 0x000010f0: .text ALLOC LOAD READONLY CODE HAS_CONTENTS
 [6]     0x004933d0->0x00493f77 at 0x000933d0: __libc_freeres_fn ALLOC LOAD READONLY CODE HAS_CONTENTS
 [7]     0x00493f78->0x00493f81 at 0x00093f78: .fini ALLOC LOAD READONLY CODE HAS_CONTENTS
 [8]     0x00494000->0x004ae73c at 0x00094000: .rodata ALLOC LOAD READONLY DATA HAS_CONTENTS
 [9]     0x004ae740->0x004bab50 at 0x000ae740: .eh_frame ALLOC LOAD READONLY DATA HAS_CONTENTS
 [10]     0x004bab50->0x004babfc at 0x000bab50: .gcc_except_table ALLOC LOAD READONLY DATA HAS_CONTENTS
 [11]     0x004bc0b0->0x004bc0d8 at 0x000bb0b0: .tdata ALLOC LOAD DATA HAS_CONTENTS
 [12]     0x004bc0d8->0x004bc120 at 0x000bb0d8: .tbss ALLOC
 [13]     0x004bc0d8->0x004bc0e0 at 0x000bb0d8: .preinit_array ALLOC LOAD DATA HAS_CONTENTS
 [14]     0x004bc0e0->0x004bc0f0 at 0x000bb0e0: .init_array ALLOC LOAD DATA HAS_CONTENTS
 [15]     0x004bc0f0->0x004bc100 at 0x000bb0f0: .fini_array ALLOC LOAD DATA HAS_CONTENTS
 [16]     0x004bc100->0x004beef4 at 0x000bb100: .data.rel.ro ALLOC LOAD DATA HAS_CONTENTS
 [17]     0x004beef8->0x004bf000 at 0x000bdef8: .got ALLOC LOAD DATA HAS_CONTENTS
 [18]     0x004bf000->0x004bf0f0 at 0x000be000: .got.plt ALLOC LOAD DATA HAS_CONTENTS
 [19]     0x004bf100->0x004c0c30 at 0x000be100: .data ALLOC LOAD DATA HAS_CONTENTS
 [20]     0x004c0c30->0x004c0c90 at 0x000bfc30: __libc_subfreeres ALLOC LOAD DATA HAS_CONTENTS
 [21]     0x004c0ca0->0x004c1408 at 0x000bfca0: __libc_IO_vtables ALLOC LOAD DATA HAS_CONTENTS
 [22]     0x004c1408->0x004c1410 at 0x000c0408: __libc_atexit ALLOC LOAD DATA HAS_CONTENTS
 [23]     0x004c1420->0x004c7528 at 0x000c0410: .bss ALLOC
```

```
 [24]     0x004c7528->0x004c7558 at 0x000c0410: __libc_freeres_ptrs ALLOC
 [25]     0x00000000->0x00000038 at 0x000c0410: .comment READONLY HAS_CONTENTS
 [26]     0x00000000->0x00000420 at 0x000c0450: .debug_aranges READONLY HAS_CONTENTS
 [27]     0x00000000->0x000372ad at 0x000c0870: .debug_info READONLY HAS_CONTENTS
 [28]     0x00000000->0x000057e8 at 0x000f7b1d: .debug_abbrev READONLY HAS_CONTENTS
 [29]     0x00000000->0x0000aa2b at 0x000fd305: .debug_line READONLY HAS_CONTENTS
 [30]     0x00000000->0x00004d08 at 0x00107d30: .debug_str READONLY HAS_CONTENTS
 [31]     0x00000000->0x0000d4b8 at 0x0010ca38: .debug_loc READONLY HAS_CONTENTS
 [32]     0x00000000->0x000024c0 at 0x00119ef0: .debug_ranges READONLY HAS_CONTENTS
Core file:
    `/home/coredump/ALCDA2/x64/App1/App1.core.253', file type elf64-x86-64.
 [0]     0x00000000->0x00002ec4 at 0x000003f8: note0 READONLY HAS_CONTENTS
 [1]     0x00000000->0x000000d8 at 0x00000518: .reg/253 HAS_CONTENTS
 [2]     0x00000000->0x000000d8 at 0x00000518: .reg HAS_CONTENTS
 [3]     0x00000000->0x00000200 at 0x0000060c: .reg2/253 HAS_CONTENTS
 [4]     0x00000000->0x00000200 at 0x0000060c: .reg2 HAS_CONTENTS
 [5]     0x00000000->0x00000340 at 0x00000820: .reg-xstate/253 HAS_CONTENTS
 [6]     0x00000000->0x00000340 at 0x00000820: .reg-xstate HAS_CONTENTS
 [7]     0x00000000->0x00000080 at 0x00000b74: .note.linuxcore.siginfo/253 HAS_CONTENTS
 [8]     0x00000000->0x00000080 at 0x00000b74: .note.linuxcore.siginfo HAS_CONTENTS
 [9]     0x00000000->0x000000d8 at 0x00000c78: .reg/254 HAS_CONTENTS
 [10]     0x00000000->0x00000200 at 0x00000d6c: .reg2/254 HAS_CONTENTS
 [11]     0x00000000->0x00000340 at 0x00000f80: .reg-xstate/254 HAS_CONTENTS
 [12]     0x00000000->0x00000080 at 0x000012d4: .note.linuxcore.siginfo/254 HAS_CONTENTS
 [13]     0x00000000->0x000000d8 at 0x000013d8: .reg/255 HAS_CONTENTS
 [14]     0x00000000->0x00000200 at 0x000014cc: .reg2/255 HAS_CONTENTS
 [15]     0x00000000->0x00000340 at 0x000016e0: .reg-xstate/255 HAS_CONTENTS
 [16]     0x00000000->0x00000080 at 0x00001a34: .note.linuxcore.siginfo/255 HAS_CONTENTS
 [17]     0x00000000->0x000000d8 at 0x00001b38: .reg/256 HAS_CONTENTS
 [18]     0x00000000->0x00000200 at 0x00001c2c: .reg2/256 HAS_CONTENTS
 [19]     0x00000000->0x00000340 at 0x00001e40: .reg-xstate/256 HAS_CONTENTS
--Type <RET> for more, q to quit, c to continue without paging--
 [20]     0x00000000->0x00000080 at 0x00002194: .note.linuxcore.siginfo/256 HAS_CONTENTS
 [21]     0x00000000->0x000000d8 at 0x00002298: .reg/257 HAS_CONTENTS
 [22]     0x00000000->0x00000200 at 0x0000238c: .reg2/257 HAS_CONTENTS
 [23]     0x00000000->0x00000340 at 0x000025a0: .reg-xstate/257 HAS_CONTENTS
 [24]     0x00000000->0x00000080 at 0x000028f4: .note.linuxcore.siginfo/257 HAS_CONTENTS
 [25]     0x00000000->0x000000d8 at 0x000029f8: .reg/258 HAS_CONTENTS
 [26]     0x00000000->0x00000200 at 0x00002aec: .reg2/258 HAS_CONTENTS
 [27]     0x00000000->0x00000340 at 0x00002d00: .reg-xstate/258 HAS_CONTENTS
 [28]     0x00000000->0x00000080 at 0x00003054: .note.linuxcore.siginfo/258 HAS_CONTENTS
 [29]     0x00000000->0x00000140 at 0x000030e8: .auxv HAS_CONTENTS
 [30]     0x00000000->0x0000007e at 0x0000323c: .note.linuxcore.file/258 HAS_CONTENTS
 [31]     0x00000000->0x0000007e at 0x0000323c: .note.linuxcore.file HAS_CONTENTS
 [32]     0x00401000->0x00494000 at 0x000032bc: load1 ALLOC LOAD READONLY CODE HAS_CONTENTS
 [33]     0x004bc000->0x004c2000 at 0x000962bc: load2 ALLOC LOAD HAS_CONTENTS
 [34]     0x004c2000->0x004c8000 at 0x0009c2bc: load3 ALLOC LOAD HAS_CONTENTS
 [35]     0x021b3000->0x021d6000 at 0x000a22bc: load4 ALLOC LOAD HAS_CONTENTS
 [36]     0x7f0fbeef7000->0x7f0fbeef8000 at 0x000c52bc: load5 ALLOC LOAD READONLY HAS_CONTENTS
 [37]     0x7f0fbeef8000->0x7f0fbf6f8000 at 0x000c62bc: load6 ALLOC LOAD HAS_CONTENTS
 [38]     0x7f0fbf6f8000->0x7f0fbf6f9000 at 0x008c62bc: load7 ALLOC LOAD READONLY HAS_CONTENTS
 [39]     0x7f0fbf6f9000->0x7f0fbfef9000 at 0x008c72bc: load8 ALLOC LOAD HAS_CONTENTS
 [40]     0x7f0fbfef9000->0x7f0fbfefa000 at 0x010c72bc: load9 ALLOC LOAD READONLY HAS_CONTENTS
 [41]     0x7f0fbfefa000->0x7f0fc06fa000 at 0x010c82bc: load10 ALLOC LOAD HAS_CONTENTS
 [42]     0x7f0fc06fa000->0x7f0fc06fb000 at 0x018c82bc: load11 ALLOC LOAD READONLY HAS_CONTENTS
 [43]     0x7f0fc06fb000->0x7f0fc0efb000 at 0x018c92bc: load12 ALLOC LOAD HAS_CONTENTS
 [44]     0x7f0fc0efb000->0x7f0fc0efc000 at 0x020c92bc: load13 ALLOC LOAD READONLY HAS_CONTENTS
 [45]     0x7f0fc0efc000->0x7f0fc16fc000 at 0x020ca2bc: load14 ALLOC LOAD HAS_CONTENTS
 [46]     0x7ffdf4545000->0x7ffdf4566000 at 0x028ca2bc: load15 ALLOC LOAD HAS_CONTENTS
 [47]     0x7ffdf45ca000->0x7ffdf45cb000 at 0x028eb2bc: load16 ALLOC LOAD READONLY CODE HAS_CONTENTS
```

11. Dump the **.data** section with possible symbolic information:

```
(gdb) x/256a 0x004bf100
0x4bf100:       0x0     0x0
0x4bf110 <__nptl_nthreads>:     0x6     0x0
0x4bf120 <stack_used>:  0x7f0fbf6f79c0  0x7f0fc16fb9c0
0x4bf130 <stack_cache>: 0x4bf130 <stack_cache>  0x4bf130 <stack_cache>
0x4bf140 <__sched_fifo_max_prio>:       0xffffffffffffffff      0x0
0x4bf150 <__elision_aconf>:     0x300000003     0x300000000
0x4bf160 <_dl_tls_static_size>: 0x1180  0x494a88 <_nl_default_default_domain>
0x4bf170 <__exit_funcs>:        0x4c5a80 <initial>      0x493040 <__gcc_personality_v0>
0x4bf180 <_IO_list_all>:        0x4bf1a0 <_IO_2_1_stderr_>      0x0
0x4bf190:       0x0     0x0
0x4bf1a0 <_IO_2_1_stderr_>:     0xfbad2086      0x0
0x4bf1b0 <_IO_2_1_stderr_+16>:  0x0     0x0
0x4bf1c0 <_IO_2_1_stderr_+32>:  0x0     0x0
0x4bf1d0 <_IO_2_1_stderr_+48>:  0x0     0x0
0x4bf1e0 <_IO_2_1_stderr_+64>:  0x0     0x0
0x4bf1f0 <_IO_2_1_stderr_+80>:  0x0     0x0
0x4bf200 <_IO_2_1_stderr_+96>:  0x0     0x4bf3c0 <_IO_2_1_stdout_>
0x4bf210 <_IO_2_1_stderr_+112>: 0x8000000002    0xffffffffffffffff
0x4bf220 <_IO_2_1_stderr_+128>: 0x0     0x4c5ec0 <_IO_stdfile_2_lock>
0x4bf230 <_IO_2_1_stderr_+144>: 0xffffffffffffffff      0x0
0x4bf240 <_IO_2_1_stderr_+160>: 0x4bf280 <_IO_wide_data_2>      0x0
0x4bf250 <_IO_2_1_stderr_+176>: 0x0     0x0
0x4bf260 <_IO_2_1_stderr_+192>: 0x0     0x0
0x4bf270 <_IO_2_1_stderr_+208>: 0x0     0x4c1060 <_IO_file_jumps>
0x4bf280 <_IO_wide_data_2>:     0x0     0x0
0x4bf290 <_IO_wide_data_2+16>:  0x0     0x0
0x4bf2a0 <_IO_wide_data_2+32>:  0x0     0x0
0x4bf2b0 <_IO_wide_data_2+48>:  0x0     0x0
0x4bf2c0 <_IO_wide_data_2+64>:  0x0     0x0
0x4bf2d0 <_IO_wide_data_2+80>:  0x0     0x0
0x4bf2e0 <_IO_wide_data_2+96>:  0x0     0x0
0x4bf2f0 <_IO_wide_data_2+112>: 0x0     0x0
0x4bf300 <_IO_wide_data_2+128>: 0x0     0x0
0x4bf310 <_IO_wide_data_2+144>: 0x0     0x0
0x4bf320 <_IO_wide_data_2+160>: 0x0     0x0
0x4bf330 <_IO_wide_data_2+176>: 0x0     0x0
0x4bf340 <_IO_wide_data_2+192>: 0x0     0x0
0x4bf350 <_IO_wide_data_2+208>: 0x0     0x0
0x4bf360 <_IO_wide_data_2+224>: 0x0     0x0
0x4bf370 <_IO_wide_data_2+240>: 0x0     0x0
0x4bf380 <_IO_wide_data_2+256>: 0x0     0x0
0x4bf390 <_IO_wide_data_2+272>: 0x0     0x0
0x4bf3a0 <_IO_wide_data_2+288>: 0x0     0x0
0x4bf3b0 <_IO_wide_data_2+304>: 0x4c0e20 <_IO_wfile_jumps>      0x0
0x4bf3c0 <_IO_2_1_stdout_>:     0xfbad2084      0x0
0x4bf3d0 <_IO_2_1_stdout_+16>:  0x0     0x0
0x4bf3e0 <_IO_2_1_stdout_+32>:  0x0     0x0
0x4bf3f0 <_IO_2_1_stdout_+48>:  0x0     0x0
0x4bf400 <_IO_2_1_stdout_+64>:  0x0     0x0
0x4bf410 <_IO_2_1_stdout_+80>:  0x0     0x0
0x4bf420 <_IO_2_1_stdout_+96>:  0x0     0x4bf5e0 <_IO_2_1_stdin_>
0x4bf430 <_IO_2_1_stdout_+112>: 0x8000000001    0xffffffffffffffff
0x4bf440 <_IO_2_1_stdout_+128>: 0x0     0x4c5ed0 <_IO_stdfile_1_lock>
0x4bf450 <_IO_2_1_stdout_+144>: 0xffffffffffffffff      0x0
0x4bf460 <_IO_2_1_stdout_+160>: 0x4bf4a0 <_IO_wide_data_1>      0x0
0x4bf470 <_IO_2_1_stdout_+176>: 0x0     0x0
0x4bf480 <_IO_2_1_stdout_+192>: 0x0     0x0
```

```
--Type <RET> for more, q to quit, c to continue without paging--
0x4bf490 <_IO_2_1_stdout_+208>: 0x0     0x4c1060 <_IO_file_jumps>
0x4bf4a0 <_IO_wide_data_1>:     0x0     0x0
0x4bf4b0 <_IO_wide_data_1+16>:  0x0     0x0
0x4bf4c0 <_IO_wide_data_1+32>:  0x0     0x0
0x4bf4d0 <_IO_wide_data_1+48>:  0x0     0x0
0x4bf4e0 <_IO_wide_data_1+64>:  0x0     0x0
0x4bf4f0 <_IO_wide_data_1+80>:  0x0     0x0
0x4bf500 <_IO_wide_data_1+96>:  0x0     0x0
0x4bf510 <_IO_wide_data_1+112>: 0x0     0x0
0x4bf520 <_IO_wide_data_1+128>: 0x0     0x0
0x4bf530 <_IO_wide_data_1+144>: 0x0     0x0
0x4bf540 <_IO_wide_data_1+160>: 0x0     0x0
0x4bf550 <_IO_wide_data_1+176>: 0x0     0x0
0x4bf560 <_IO_wide_data_1+192>: 0x0     0x0
0x4bf570 <_IO_wide_data_1+208>: 0x0     0x0
0x4bf580 <_IO_wide_data_1+224>: 0x0     0x0
0x4bf590 <_IO_wide_data_1+240>: 0x0     0x0
0x4bf5a0 <_IO_wide_data_1+256>: 0x0     0x0
0x4bf5b0 <_IO_wide_data_1+272>: 0x0     0x0
0x4bf5c0 <_IO_wide_data_1+288>: 0x0     0x0
0x4bf5d0 <_IO_wide_data_1+304>: 0x4c0e20 <_IO_wfile_jumps>      0x0
0x4bf5e0 <_IO_2_1_stdin_>:      0xfbad2088      0x0
0x4bf5f0 <_IO_2_1_stdin_+16>:   0x0     0x0
0x4bf600 <_IO_2_1_stdin_+32>:   0x0     0x0
0x4bf610 <_IO_2_1_stdin_+48>:   0x0     0x0
0x4bf620 <_IO_2_1_stdin_+64>:   0x0     0x0
0x4bf630 <_IO_2_1_stdin_+80>:   0x0     0x0
0x4bf640 <_IO_2_1_stdin_+96>:   0x0     0x0
0x4bf650 <_IO_2_1_stdin_+112>:  0x8000000000    0xffffffffffffffff
0x4bf660 <_IO_2_1_stdin_+128>:  0x0     0x4c5ee0 <_IO_stdfile_0_lock>
0x4bf670 <_IO_2_1_stdin_+144>:  0xffffffffffffffff      0x0
0x4bf680 <_IO_2_1_stdin_+160>:  0x4bf6c0 <_IO_wide_data_0>      0x0
0x4bf690 <_IO_2_1_stdin_+176>:  0x0     0x0
0x4bf6a0 <_IO_2_1_stdin_+192>:  0x0     0x0
0x4bf6b0 <_IO_2_1_stdin_+208>:  0x0     0x4c1060 <_IO_file_jumps>
0x4bf6c0 <_IO_wide_data_0>:     0x0     0x0
0x4bf6d0 <_IO_wide_data_0+16>:  0x0     0x0
0x4bf6e0 <_IO_wide_data_0+32>:  0x0     0x0
0x4bf6f0 <_IO_wide_data_0+48>:  0x0     0x0
0x4bf700 <_IO_wide_data_0+64>:  0x0     0x0
0x4bf710 <_IO_wide_data_0+80>:  0x0     0x0
0x4bf720 <_IO_wide_data_0+96>:  0x0     0x0
0x4bf730 <_IO_wide_data_0+112>: 0x0     0x0
0x4bf740 <_IO_wide_data_0+128>: 0x0     0x0
0x4bf750 <_IO_wide_data_0+144>: 0x0     0x0
0x4bf760 <_IO_wide_data_0+160>: 0x0     0x0
0x4bf770 <_IO_wide_data_0+176>: 0x0     0x0
0x4bf780 <_IO_wide_data_0+192>: 0x0     0x0
0x4bf790 <_IO_wide_data_0+208>: 0x0     0x0
0x4bf7a0 <_IO_wide_data_0+224>: 0x0     0x0
0x4bf7b0 <_IO_wide_data_0+240>: 0x0     0x0
0x4bf7c0 <_IO_wide_data_0+256>: 0x0     0x0
0x4bf7d0 <_IO_wide_data_0+272>: 0x0     0x0
0x4bf7e0 <_IO_wide_data_0+288>: 0x0     0x0
0x4bf7f0 <_IO_wide_data_0+304>: 0x4c0e20 <_IO_wfile_jumps>      0x4bf1a0 <_IO_2_1_stderr_>
0x4bf800 <stdout>:      0x4bf3c0 <_IO_2_1_stdout_>      0x4bf5e0 <_IO_2_1_stdin_>
0x4bf810:       0x0     0x0
--Type <RET> for more, q to quit, c to continue without paging--
0x4bf820 <may_shrink_heap.11591>:       0x1fffffff      0x1
```

```
0x4bf830:       0x0     0x0
0x4bf840 <mp_>: 0x20000 0x20000
0x4bf850 <mp_+16>:      0x20000 0x8
0x4bf860 <mp_+32>:      0x0     0x1000000000000
0x4bf870 <mp_+48>:      0x0     0x0
0x4bf880 <mp_+64>:      0x0     0x21b41c0
0x4bf890 <mp_+80>:      0x40    0x408
0x4bf8a0 <mp_+96>:      0x7     0x0
0x4bf8b0:       0x0     0x0
0x4bf8c0 <__memalign_hook>:     0x41aad0 <memalign_hook_ini>    0x41b0e0 <realloc_hook_ini>
0x4bf8d0 <__malloc_hook>:       0x0     0x0
0x4bf8e0 <main_arena>:  0x0     0x0
0x4bf8f0 <main_arena+16>:       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:

```
0x4bf110 <__nptl_nthreads>:     0x6     0x0
```

Each value may also have an associated symbolic value:

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

For example, from the output above:

```
0x4bf8c0 <__memalign_hook>:     0x41aad0 <memalign_hook_ini>    0x41b0e0 <realloc_hook_ini>
```

12. Explore the contents of memory pointed to by *__nptl_nthreads*, *_nl_default_default_domain*, and *__memalign_hook* addresses (**/x** is for hex, **/d** is for decimals, **/u** is for unsigned decimals, **/g** is for 64-bit values, **/w** is for 32-bit values, **/h** is for 16-bit values, **/b** is for byte values, **/a** is for addresses, **/c** and **/s** are for chars and strings):

```
(gdb) x/d 0x4bf110
0x4bf110 <__nptl_nthreads>:     6

(gdb) x/u &__nptl_nthreads
0x4bf110 <__nptl_nthreads>:     6

(gdb) x/wx 0x4bf110
0x4bf110 <__nptl_nthreads>:     0x00000006

(gdb) x/gx 0x4bf110
0x4bf110 <__nptl_nthreads>:     0x0000000000000006

(gdb) x/hx 0x4bf110
0x4bf110 <__nptl_nthreads>:     0x0006

(gdb) x/bx 0x4bf110
0x4bf110 <__nptl_nthreads>:     0x06
```

```
(gdb) x/2a 0x4bf160
0x4bf160 <_dl_tls_static_size>: 0x1180  0x494a88 <_nl_default_default_domain>
```

Note: Some symbols and addresses (for example, **0x494a88**) belong to read-only sections of executable image. If GDB refuses to read them you may need to run this command:

```
(gdb) set trust-readonly-sections on

(gdb) x/a &_nl_default_default_domain
0x494a88 <_nl_default_default_domain>:  0x736567617373656d

(gdb) x/a 0x494a88
0x494a88 <_nl_default_default_domain>:  0x736567617373656d

(gdb) x/s 0x494a88
0x494a88 <_nl_default_default_domain>:  "messages"

(gdb) x/10a 0x494a88
0x494a88 <_nl_default_default_domain>:  0x736567617373656d      0x6c006f6c00756c00
0x494a98:       0x786c00586c0069        0x7273752f00656372
0x494aa8:       0x6c2f65726168732f      0x656c61636f
0x494ab8 <aliasfile.10131>:     0x2e656c61636f6c2f      0x7361696c61
0x494ac8:       0x0     0x0

(gdb) x/8c 0x494a88
0x494a88 <_nl_default_default_domain>:  109 'm' 101 'e' 115 's' 115 's' 97 'a'  103 'g' 101 'e'
115 's'

(gdb) x/10s 0x494a88
0x494a88 <_nl_default_default_domain>:  "messages"
0x494a91:       "lu"
0x494a94:       "lo"
0x494a97:       "li"
0x494a9a:       "lX"
0x494a9d:       "lx"
0x494aa0:       "rce"
0x494aa4:       "/usr/share/locale"
0x494ab6:       ""
0x494ab7:       ""
```

Note: We see that a hook function is installed for *memalign* but not *malloc*. Please find the following documentation for hook functions here:

https://www.gnu.org/software/libc/manual/html_node/Hooks-for-Malloc.html

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

```
(gdb) x/a &environ
0x4c5f48 <environ>:     0x7ffdf45637f8

(gdb) x/10a 0x7ffdf45637f8
0x7ffdf45637f8: 0x7ffdf4565756  0x7ffdf4565766
0x7ffdf4563808: 0x7ffdf456577d  0x7ffdf4565794
0x7ffdf4563818: 0x7ffdf45657a9  0x7ffdf45657c7
0x7ffdf4563828: 0x7ffdf45657d8  0x7ffdf45657f3
0x7ffdf4563838: 0x7ffdf45657fe  0x7ffdf4565812
```

```
(gdb) x/10s 0x7ffdf4565756
0x7ffdf4565756: "SHELL=/bin/bash"
0x7ffdf4565766: "HISTCONTROL=ignoreboth"
0x7ffdf456577d: "WSL_DISTRO_NAME=Debian"
0x7ffdf4565794: "NAME=DESKTOP-IS6V2L0"
0x7ffdf45657a9: "PWD=/home/coredump/ALCDA/App1"
0x7ffdf45657c7: "LOGNAME=coredump"
0x7ffdf45657d8: "MC_TMPDIR=/tmp/mc-coredump"
0x7ffdf45657f3: "MC_SID=192"
0x7ffdf45657fe: "HOME=/home/coredump"
0x7ffdf4565812: "LANG=en_US.UTF-8"
```

Note: The last command interprets **0x7ffdf4565756** as an address of the sequence of 10 null-terminated strings.

14. Now, we look at how to perform a memory search. It is not possible to search in the entire virtual memory, only in the valid regions (WinDbg debugger is able).

```
(gdb) find /g 0x004bc000, 0x004d2000, 6
0x4bd5f8 <_nl_C_LC_NUMERIC+56>
0x4be880 <tunable_list+928>
0x4bea40 <dyn_temp.10655+32>
0x4bf110 <__nptl_nthreads>
warning: Unable to access 16000 bytes of target memory at 0x4c6e18, halting search.
4 patterns found.

(gdb) x/gd 0x4bf110
0x4bf110 <__nptl_nthreads>:     6

(gdb) x/s 0x7ffdf4565756
0x7ffdf4565756: "SHELL=/bin/bash"

(gdb) find 0x7ffdf4565756, +100, "bash"
0x7ffdf4565761
1 pattern found.
```

Note: "bash" is considered a null-terminated array of characters for the search. To search for a string sequence without a null terminator, use a sequence of characters:

```
(gdb) find 0x7ffdf4565756, +100, "bin"
Pattern not found.

(gdb) find 0x7ffdf4565756, +100, 'b', 'i', 'n'
0x7ffdf456575d
1 pattern found.
```

15. Get the list of loaded modules:

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

Note: 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 *App1.shared.core.275*, we see the list of shared libraries:

```
~/ALCDA2/x64/App1$ gdb -c App1.shared.core.275 -se App1.shared
GNU gdb (Debian 8.2.1-2+b3) 8.2.1
Copyright (C) 2018 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".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from App1.shared...(no debugging symbols found)...done.
[New LWP 275]
[New LWP 276]
[New LWP 277]
[New LWP 278]
[New LWP 279]
[New LWP 280]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Core was generated by `./App1.shared'.
#0  0x00007f1ae471e720 in __GI___nanosleep (requested_time=requested_time@entry=0x7ffc74957a90,
remaining=remaining@entry=0x7ffc74957a90) at ../sysdeps/unix/sysv/linux/nanosleep.c:28
28      ../sysdeps/unix/sysv/linux/nanosleep.c: No such file or directory.
[Current thread is 1 (Thread 0x7f1ae4655740 (LWP 275))]
```

```
(gdb) info sharedlibrary
From                To                  Syms Read   Shared Object Library
0x00007f1ae481f5b0  0x00007f1ae482d641  Yes         /lib/x86_64-linux-gnu/libpthread.so.0
0x00007f1ae467a320  0x00007f1ae47c039b  Yes         /lib/x86_64-linux-gnu/libc.so.6
0x00007f1ae4848090  0x00007f1ae4865b20  Yes         /lib64/ld-linux-x86-64.so.2
```

16. Disassemble the *bar_one* function and follow the indirect *sleep* function call:

```
(gdb) disassemble bar_one
Dump of assembler code for function bar_one:
   0x0000557e17348145 <+0>:     push   %rbp
   0x0000557e17348146 <+1>:     mov    %rsp,%rbp
   0x0000557e17348149 <+4>:     mov    $0xffffffff,%edi
   0x0000557e1734814e <+9>:     callq  0x557e17348040 <sleep@plt>
   0x0000557e17348153 <+14>:    nop
   0x0000557e17348154 <+15>:    pop    %rbp
   0x0000557e17348155 <+16>:    retq
End of assembler dump.
```

```
(gdb) disassemble 0x557e17348040
Dump of assembler code for function sleep@plt:
   0x0000557e17348040 <+0>:     jmpq   *0x2fda(%rip)        # 0x557e1734b020 <sleep@got.plt>
   0x0000557e17348046 <+6>:     pushq  $0x1
   0x0000557e1734804b <+11>:    jmpq   0x557e17348020
End of assembler dump.
```

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

```
(gdb) p/x 0x0000557e17348046+0x2fda
$1 = 0x557e1734b020

(gdb) x/a 0x557e1734b020
0x557e1734b020 <sleep@got.plt>: 0x7f1ae471e5f0 <__sleep>
```

Note: Since GDB gets shared library images from your analysis system that do not correspond to shared libraries from the crash system, most likely you get some random symbolic information (and also an invalid backtrace from the **bt** command):

```
(gdb) x/a 0x557e1734b020
0x557e1734b020 <sleep@got.plt>:  0x7f1ae471e5f0 <__getpwnam_r+288>

(gdb) bt
#0  __getpwnam_r (name=<optimized out>, resbuf=0x7ffc74957bf0, buffer=<optimized out>,
buflen=94000043556960, result=<optimized out>)
    at ../nss/getXXbyYY_r.c:416
#1  0x00007f1ae3652700 in ?? ()
#2  0x00007f1ae3e53700 in ?? ()
#3  0x00007f1ae4654700 in ?? ()
#4  0x0000557e17348340 in ?? ()
#5  0x00007f1ae467c09b in _IO_acquire_lock_fct (p=<optimized out>) at libioP.h:759
#6  _IO_getc (fp=0x557e1734832a <main+170>) at getc.c:39
#7  0x00007ffc74957bf8 in ?? ()
#8  0x0000000100040000 in ?? ()
#9  0x0000557e17348280 in thread_five ()
#10 0x0000000000000000 in ?? ()
```

Note: You need the original shared library images and debug symbol files from the problem system. To get the right results for this exercise, you can recreate the *App1.shared* core dump (see *main.c* for build instructions if necessary). There is also the ARM64 A1 exercise with shared libraries from the problem system.

18. *App1.shared.pmap.275* also shows library memory regions:

```
(gdb) q

~/ALCDA2/x64/App1$ cat App1.shared.pmap.275
275:   ./App1.shared
0000557e17347000      4K r---- App1.shared
0000557e17348000      4K r-x-- App1.shared
0000557e17349000      4K r---- App1.shared
0000557e1734a000      4K r---- App1.shared
0000557e1734b000      4K rw--- App1.shared
0000557e179ca000    132K rw---   [ anon ]
00007f1ae1e50000      4K -----   [ anon ]
00007f1ae1e51000   8192K rw---   [ anon ]
00007f1ae2651000      4K -----   [ anon ]
00007f1ae2652000   8192K rw---   [ anon ]
00007f1ae2e52000      4K -----   [ anon ]
00007f1ae2e53000   8192K rw---   [ anon ]
00007f1ae3653000      4K -----   [ anon ]
00007f1ae3654000   8192K rw---   [ anon ]
00007f1ae3e54000      4K -----   [ anon ]
00007f1ae3e55000   8204K rw---   [ anon ]
00007f1ae4658000    136K r---- libc-2.28.so
00007f1ae467a000   1312K r-x-- libc-2.28.so
00007f1ae47c2000    304K r---- libc-2.28.so
00007f1ae480e000      4K ----- libc-2.28.so
00007f1ae480f000     16K r---- libc-2.28.so
00007f1ae4813000      8K rw--- libc-2.28.so
00007f1ae4815000     16K rw---   [ anon ]
00007f1ae4819000     24K r---- libpthread-2.28.so
00007f1ae481f000     60K r-x-- libpthread-2.28.so
00007f1ae482e000     24K r---- libpthread-2.28.so
```

```
00007f1ae4834000      4K r---- libpthread-2.28.so
00007f1ae4835000      4K rw--- libpthread-2.28.so
00007f1ae4836000     24K rw---   [ anon ]
00007f1ae4847000      4K r---- ld-2.28.so
00007f1ae4848000    120K r-x-- ld-2.28.so
00007f1ae4866000     32K r---- ld-2.28.so
00007f1ae486e000      4K r---- ld-2.28.so
00007f1ae486f000      4K rw--- ld-2.28.so
00007f1ae4870000      4K rw---   [ anon ]
00007ffc74939000    132K rw---   [ stack ]
00007ffc749ac000     16K r----   [ anon ]
00007ffc749b0000      4K r-x--   [ anon ]
 total            43400K
```

Exercise A1 (A64, GDB)

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

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

1. Load the core dump *App1.core.21174* and *App1* executable from the A64/App1 directory. We use **gdb-multiarch** on x64 Debian:

```
~/ALCDA2/A64/App1$ gdb-multiarch -c App1.core.21174 -se App1
GNU gdb (Debian 8.2.1-2+b3) 8.2.1
Copyright (C) 2018 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".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from App1...(no debugging symbols found)...done.
[New LWP 21175]
[New LWP 21176]
[New LWP 21177]
[New LWP 21178]
[New LWP 21179]
[New LWP 21174]
Core was generated by `./App1'.
#0  0x000000000040c9b4 in nanosleep ()
[Current thread is 1 (LWP 21175)]
```

2. Set logging to a file in case of lengthy output from some commands:

```
(gdb) set logging on App1.log
Copying output to App1.log.
```

Note: If you use the newer GDB 12.x on an ARM64 Ubuntu instance instead, you need to use different logging commands, and you may also want to set color highlighting off:

```
(gdb) set logging file App1.log

(gdb) set logging enabled on
Copying output to App1.log.
Copying debug output to App1.log.

(gdb) set style enabled off
```

3. List all threads:

```
(gdb) info threads
  Id   Target Id         Frame
* 1    LWP 21175         0x000000000040c9b4 in nanosleep ()
  2    LWP 21176         0x000000000040c9b4 in nanosleep ()
  3    LWP 21177         0x000000000040c9b4 in nanosleep ()
  4    LWP 21178         0x000000000040c9b4 in nanosleep ()
  5    LWP 21179         0x000000000040c9b4 in nanosleep ()
  6    LWP 21174         0x000000000040c9b4 in nanosleep ()
```

4. Get the current thread stack trace:

```
(gdb) bt
#0  0x000000000040c9b4 in nanosleep ()
#1  0x0000000000424cb4 in sleep ()
#2  0x00000000004031f8 in bar_one ()
#3  0x000000000040320c in foo_one ()
#4  0x0000000000403224 in thread_one ()
#5  0x0000000000404c34 in start_thread ()
#6  0x0000000000429b60 in thread_start ()
```

5. Get all thread stack traces:

```
(gdb) thread apply all bt

Thread 6 (LWP 21174):
#0  0x000000000040c9b4 in nanosleep ()
#1  0x0000000000424cb4 in sleep ()
#2  0x00000000004033e0 in main ()

Thread 5 (LWP 21179):
#0  0x000000000040c9b4 in nanosleep ()
#1  0x0000000000424cb4 in sleep ()
#2  0x0000000000403318 in bar_five ()
#3  0x000000000040332c in foo_five ()
#4  0x0000000000403344 in thread_five ()
#5  0x0000000000404c34 in start_thread ()
#6  0x0000000000429b60 in thread_start ()

Thread 4 (LWP 21178):
#0  0x000000000040c9b4 in nanosleep ()
#1  0x0000000000424cb4 in sleep ()
#2  0x00000000004032d0 in bar_four ()
#3  0x00000000004032e4 in foo_four ()
#4  0x00000000004032fc in thread_four ()
#5  0x0000000000404c34 in start_thread ()
#6  0x0000000000429b60 in thread_start ()

Thread 3 (LWP 21177):
#0  0x000000000040c9b4 in nanosleep ()
#1  0x0000000000424cb4 in sleep ()
#2  0x0000000000403288 in bar_three ()
#3  0x000000000040329c in foo_three ()
#4  0x00000000004032b4 in thread_three ()
#5  0x0000000000404c34 in start_thread ()
#6  0x0000000000429b60 in thread_start ()

Thread 2 (LWP 21176):
#0  0x000000000040c9b4 in nanosleep ()
```

```
#1  0x0000000000424cb4 in sleep ()
#2  0x0000000000403240 in bar_two ()
#3  0x0000000000403254 in foo_two ()
#4  0x000000000040326c in thread_two ()
#5  0x0000000000404c34 in start_thread ()
#6  0x0000000000429b60 in thread_start ()

Thread 1 (LWP 21175):
#0  0x000000000040c9b4 in nanosleep ()
#1  0x0000000000424cb4 in sleep ()
#2  0x00000000004031f8 in bar_one ()
#3  0x000000000040320c in foo_one ()
#4  0x0000000000403224 in thread_one ()
#5  0x0000000000404c34 in start_thread ()
#6  0x0000000000429b60 in thread_start ()
```

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

```
(gdb) thread 2
[Switching to thread 2 (LWP 21176)]
#0  0x000000000040c9b4 in nanosleep ()

(gdb) bt
#0  0x000000000040c9b4 in nanosleep ()
#1  0x0000000000424cb4 in sleep ()
#2  0x0000000000403240 in bar_two ()
#3  0x0000000000403254 in foo_two ()
#4  0x000000000040326c in thread_two ()
#5  0x0000000000404c34 in start_thread ()
#6  0x0000000000429b60 in thread_start ()

(gdb) info threads
  Id   Target Id         Frame
  1    LWP 21175         0x000000000040c9b4 in nanosleep ()
* 2    LWP 21176         0x000000000040c9b4 in nanosleep ()
  3    LWP 21177         0x000000000040c9b4 in nanosleep ()
  4    LWP 21178         0x000000000040c9b4 in nanosleep ()
  5    LWP 21179         0x000000000040c9b4 in nanosleep ()
  6    LWP 21174         0x000000000040c9b4 in nanosleep ()
```

7. Check that *bar_two* called the *sleep* function by comparing the return address on the call stack from the disassembly output:

```
(gdb) disassemble bar_two
Dump of assembler code for function bar_two:
   0x0000000000403230 <+0>:     stp     x29, x30, [sp, #-16]!
   0x0000000000403234 <+4>:     mov     x29, sp
   0x0000000000403238 <+8>:     mov     w0, #0xffffffff                 // #-1
   0x000000000040323c <+12>:    bl      0x424ba4 <sleep>
   0x0000000000403240 <+16>:    ldp     x29, x30, [sp], #16
   0x0000000000403244 <+20>:    ret
End of assembler dump.
```

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

8. Get the *App1* data section from the output of pmap (*App1.pmap.21174*):

```
(gdb) ^Z
[1]+  Stopped                 gdb -c App1.core.21174 -se App1

~/ALCDA2/A64/App1$ cat App1.pmap.21174
21174:   ./App1
0000000000400000    768K r-x-- App1
00000000004c0000    128K rw--- App1
0000000001fa0000    256K rw---   [ anon ]
0000fffccab40000     64K -----   [ anon ]
0000fffccab50000   8192K rw---   [ anon ]
0000fffccb350000     64K -----   [ anon ]
0000fffccb360000   8192K rw---   [ anon ]
0000fffccbb60000     64K -----   [ anon ]
0000fffccbb70000   8192K rw---   [ anon ]
0000fffccc370000     64K -----   [ anon ]
0000fffccc380000   8192K rw---   [ anon ]
0000fffcccb80000     64K -----   [ anon ]
0000fffcccb90000   8192K rw---   [ anon ]
0000fffccd390000     64K r----   [ anon ]
0000fffccd3a0000     64K r-x--   [ anon ]
0000ffffd3090000    192K rw---   [ stack ]
 total            42752K

~/ALCDA2/A64/App1$ fg
gdb -c App1.core.21174 -se App1


(gdb)
```

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

```
(gdb) p/x 0x00000000004c0000+128*1024
$1 = 0x4e0000

(gdb) maintenance info sections
Exec file: `/home/ubuntu/ALCDA2/A64/App1/App1', file type elf64-littleaarch64.
 [0]     0x00400190->0x004001b0 at 0x00000190: .note.ABI-tag ALLOC LOAD READONLY DATA HAS_CONTENTS
 [1]     0x004001b0->0x004001d4 at 0x000001b0: .note.gnu.build-id ALLOC LOAD READONLY DATA HAS_CONTENTS
 [2]     0x004001d8->0x00400250 at 0x000001d8: .rela.plt ALLOC LOAD READONLY DATA HAS_CONTENTS
 [3]     0x00400250->0x00400264 at 0x00000250: .init ALLOC LOAD READONLY CODE HAS_CONTENTS
 [4]     0x00400270->0x004002c0 at 0x00000270: .plt ALLOC LOAD READONLY CODE HAS_CONTENTS
 [5]     0x004002c0->0x00487098 at 0x000002c0: .text ALLOC LOAD READONLY CODE HAS_CONTENTS
 [6]     0x00487098->0x00488d68 at 0x00087098: __libc_freeres_fn ALLOC LOAD READONLY CODE HAS_CONTENTS
 [7]     0x00488d68->0x004891b8 at 0x00088d68: __libc_thread_freeres_fn ALLOC LOAD READONLY CODE
HAS_CONTENTS
 [8]     0x004891b8->0x004891c8 at 0x000891b8: .fini ALLOC LOAD READONLY CODE HAS_CONTENTS
 [9]     0x004891d0->0x004a16ad at 0x000891d0: .rodata ALLOC LOAD READONLY DATA HAS_CONTENTS
 [10]    0x004a16ad->0x004a16ae at 0x000a16ad: .stapsdt.base ALLOC LOAD READONLY DATA HAS_CONTENTS
 [11]    0x004a16b0->0x004a1de8 at 0x000a16b0: __libc_IO_vtables ALLOC LOAD READONLY DATA HAS_CONTENTS
 [12]    0x004a1de8->0x004a1e50 at 0x000a1de8: __libc_subfreeres ALLOC LOAD READONLY DATA HAS_CONTENTS
 [13]    0x004a1e50->0x004a1e58 at 0x000a1e50: __libc_atexit ALLOC LOAD READONLY DATA HAS_CONTENTS
 [14]    0x004a1e58->0x004a1e68 at 0x000a1e58: __libc_thread_subfreeres ALLOC LOAD READONLY DATA
HAS_CONTENTS
 [15]    0x004a1e68->0x004b047c at 0x000a1e68: .eh_frame ALLOC LOAD READONLY DATA HAS_CONTENTS
 [16]    0x004b047c->0x004b0639 at 0x000b047c: .gcc_except_table ALLOC LOAD READONLY DATA HAS_CONTENTS
 [17]    0x004cfb20->0x004cfb48 at 0x000bfb20: .tdata ALLOC LOAD DATA HAS_CONTENTS
 [18]    0x004cfb48->0x004cfb98 at 0x000bfb48: .tbss ALLOC
 [19]    0x004cfb48->0x004cfb50 at 0x000bfb48: .init_array ALLOC LOAD DATA HAS_CONTENTS
 [20]    0x004cfb50->0x004cfb60 at 0x000bfb50: .fini_array ALLOC LOAD DATA HAS_CONTENTS
 [21]    0x004cfb60->0x004cfb68 at 0x000bfb60: .jcr ALLOC LOAD DATA HAS_CONTENTS
 [22]    0x004cfb68->0x004cff24 at 0x000bfb68: .data.rel.ro ALLOC LOAD DATA HAS_CONTENTS
```

```
 [23]     0x004cff28->0x004cffe8 at 0x000bff28: .got ALLOC LOAD DATA HAS_CONTENTS
 [24]     0x004cffe8->0x004d0028 at 0x000bffe8: .got.plt ALLOC LOAD DATA HAS_CONTENTS
 [25]     0x004d0030->0x004d1580 at 0x000c0030: .data ALLOC LOAD DATA HAS_CONTENTS
 [26]     0x004d1580->0x004d8050 at 0x000c1580: .bss ALLOC
 [27]     0x004d8050->0x004d8088 at 0x000c1580: __libc_freeres_ptrs ALLOC
 [28]     0x00000000->0x00000031 at 0x000c1580: .comment READONLY HAS_CONTENTS
 [29]     0x00000000->0x00001cb0 at 0x000c15b4: .note.stapsdt READONLY HAS_CONTENTS
Core file: `/home/ubuntu/ALCDA2/A64/App1/App1.core.21174', file type elf64-littleaarch64.
 [0]      0x00000000->0x00001c94 at 0x000003c0: note0 READONLY HAS_CONTENTS
 [1]      0x00000000->0x00000110 at 0x000004e0: .reg/21175 HAS_CONTENTS
 [2]      0x00000000->0x00000110 at 0x000004e0: .reg HAS_CONTENTS
 [3]      0x00000000->0x00000210 at 0x0000060c: .reg2/21175 HAS_CONTENTS
 [4]      0x00000000->0x00000210 at 0x0000060c: .reg2 HAS_CONTENTS
 [5]      0x00000000->0x00000080 at 0x00000830: .note.linuxcore.siginfo/21175 HAS_CONTENTS
 [6]      0x00000000->0x00000080 at 0x00000830: .note.linuxcore.siginfo HAS_CONTENTS
 [7]      0x00000000->0x00000110 at 0x00000934: .reg/21176 HAS_CONTENTS
 [8]      0x00000000->0x00000210 at 0x00000a60: .reg2/21176 HAS_CONTENTS
 [9]      0x00000000->0x00000080 at 0x00000c84: .note.linuxcore.siginfo/21176 HAS_CONTENTS
 [10]     0x00000000->0x00000110 at 0x00000d88: .reg/21177 HAS_CONTENTS
 [11]     0x00000000->0x00000210 at 0x00000eb4: .reg2/21177 HAS_CONTENTS
 [12]     0x00000000->0x00000080 at 0x000010d8: .note.linuxcore.siginfo/21177 HAS_CONTENTS
 [13]     0x00000000->0x00000110 at 0x000011dc: .reg/21178 HAS_CONTENTS
 [14]     0x00000000->0x00000210 at 0x00001308: .reg2/21178 HAS_CONTENTS
 [15]     0x00000000->0x00000080 at 0x0000152c: .note.linuxcore.siginfo/21178 HAS_CONTENTS
 [16]     0x00000000->0x00000110 at 0x00001630: .reg/21179 HAS_CONTENTS
 [17]     0x00000000->0x00000210 at 0x0000175c: .reg2/21179 HAS_CONTENTS
--Type <RET> for more, q to quit, c to continue without paging--
 [18]     0x00000000->0x00000080 at 0x00001980: .note.linuxcore.siginfo/21179 HAS_CONTENTS
 [19]     0x00000000->0x00000110 at 0x00001a84: .reg/21174 HAS_CONTENTS
 [20]     0x00000000->0x00000210 at 0x00001bb0: .reg2/21174 HAS_CONTENTS
 [21]     0x00000000->0x00000080 at 0x00001dd4: .note.linuxcore.siginfo/21174 HAS_CONTENTS
 [22]     0x00000000->0x00000160 at 0x00001e68: .auxv HAS_CONTENTS
 [23]     0x00000000->0x00000076 at 0x00001fdc: .note.linuxcore.file/21174 HAS_CONTENTS
 [24]     0x00000000->0x00000076 at 0x00001fdc: .note.linuxcore.file HAS_CONTENTS
 [25]     0x00400000->0x004c0000 at 0x00002054: load1 ALLOC LOAD READONLY CODE HAS_CONTENTS
 [26]     0x004c0000->0x004e0000 at 0x000c2054: load2 ALLOC LOAD HAS_CONTENTS
 [27]     0x01fa0000->0x01fe0000 at 0x000e2054: load3 ALLOC LOAD HAS_CONTENTS
 [28]     0xfffccab40000->0xfffccab50000 at 0x00122054: load4 ALLOC LOAD READONLY HAS_CONTENTS
 [29]     0xfffccab50000->0xfffccb350000 at 0x00132054: load5 ALLOC LOAD HAS_CONTENTS
 [30]     0xfffccb350000->0xfffccb360000 at 0x00932054: load6 ALLOC LOAD READONLY HAS_CONTENTS
 [31]     0xfffccb360000->0xfffccbb60000 at 0x00942054: load7 ALLOC LOAD HAS_CONTENTS
 [32]     0xfffccbb60000->0xfffccbb70000 at 0x01142054: load8 ALLOC LOAD READONLY HAS_CONTENTS
 [33]     0xfffccbb70000->0xfffccc370000 at 0x01152054: load9 ALLOC LOAD HAS_CONTENTS
 [34]     0xfffccc370000->0xfffccc380000 at 0x01952054: load10 ALLOC LOAD READONLY HAS_CONTENTS
 [35]     0xfffccc380000->0xfffcccb80000 at 0x01962054: load11 ALLOC LOAD HAS_CONTENTS
 [36]     0xfffcccb80000->0xfffcccb90000 at 0x02162054: load12 ALLOC LOAD READONLY HAS_CONTENTS
 [37]     0xfffcccb90000->0xfffccd390000 at 0x02172054: load13 ALLOC LOAD HAS_CONTENTS
 [38]     0xfffccd3a0000->0xfffccd3b0000 at 0x02972054: load14 ALLOC LOAD READONLY CODE HAS_CONTENTS
 [39]     0xffffd3090000->0xffffd30c0000 at 0x02982054: load15 ALLOC LOAD HAS_CONTENTS
```

10. Dump the first 600 addresses from the **.data** section with possible symbolic information:

```
(gdb) x/600a 0x004d0030
0x4d0030:       0x0     0x4d0038 <stack_cache>
0x4d0040 <stack_cache+8>:       0x4d0038 <stack_cache>  0x6
0x4d0050 <stack_used>:  0xfffccb34f140  0xfffccd38f140
0x4d0060 <__sched_fifo_max_prio>:       0xffffffffffffffff      0x890
0x4d0070 <__exit_funcs>:        0x4d5eb0 <initial>      0x486b88 <__gcc_personality_v0>
0x4d0080 <_IO_list_all>:        0x4d0088 <_IO_2_1_stderr_>      0xfbad2086
0x4d0090 <_IO_2_1_stderr_+8>:   0x0     0x0
0x4d00a0 <_IO_2_1_stderr_+24>:  0x0     0x0
0x4d00b0 <_IO_2_1_stderr_+40>:  0x0     0x0
0x4d00c0 <_IO_2_1_stderr_+56>:  0x0     0x0
0x4d00d0 <_IO_2_1_stderr_+72>:  0x0     0x0
0x4d00e0 <_IO_2_1_stderr_+88>:  0x0     0x0
```

```
0x4d00f0 <_IO_2_1_stderr_+104>: 0x4d02b0 <_IO_2_1_stdout_>      0x2
0x4d0100 <_IO_2_1_stderr_+120>: 0xffffffffffffffff      0x0
0x4d0110 <_IO_2_1_stderr_+136>: 0x4d6428 <_IO_stdfile_2_lock>   0xffffffffffffffff
0x4d0120 <_IO_2_1_stderr_+152>: 0x0     0x4d0168 <_IO_wide_data_2>
0x4d0130 <_IO_2_1_stderr_+168>: 0x0     0x0
0x4d0140 <_IO_2_1_stderr_+184>: 0x0     0x0
0x4d0150 <_IO_2_1_stderr_+200>: 0x0     0x0
0x4d0160 <_IO_2_1_stderr_+216>: 0x4a1950 <_IO_file_jumps>       0x0
0x4d0170 <_IO_wide_data_2+8>:   0x0     0x0
0x4d0180 <_IO_wide_data_2+24>:  0x0     0x0
0x4d0190 <_IO_wide_data_2+40>:  0x0     0x0
0x4d01a0 <_IO_wide_data_2+56>:  0x0     0x0
0x4d01b0 <_IO_wide_data_2+72>:  0x0     0x0
0x4d01c0 <_IO_wide_data_2+88>:  0x0     0x0
0x4d01d0 <_IO_wide_data_2+104>: 0x0     0x0
0x4d01e0 <_IO_wide_data_2+120>: 0x0     0x0
0x4d01f0 <_IO_wide_data_2+136>: 0x0     0x0
0x4d0200 <_IO_wide_data_2+152>: 0x0     0x0
0x4d0210 <_IO_wide_data_2+168>: 0x0     0x0
0x4d0220 <_IO_wide_data_2+184>: 0x0     0x0
0x4d0230 <_IO_wide_data_2+200>: 0x0     0x0
0x4d0240 <_IO_wide_data_2+216>: 0x0     0x0
0x4d0250 <_IO_wide_data_2+232>: 0x0     0x0
0x4d0260 <_IO_wide_data_2+248>: 0x0     0x0
0x4d0270 <_IO_wide_data_2+264>: 0x0     0x0
0x4d0280 <_IO_wide_data_2+280>: 0x0     0x0
0x4d0290 <_IO_wide_data_2+296>: 0x0     0x0
0x4d02a0 <_IO_wide_data_2+312>: 0x0     0x4a1800 <_IO_wfile_jumps>
0x4d02b0 <_IO_2_1_stdout_>:     0xfbad2084      0x0
0x4d02c0 <_IO_2_1_stdout_+16>:  0x0     0x0
0x4d02d0 <_IO_2_1_stdout_+32>:  0x0     0x0
0x4d02e0 <_IO_2_1_stdout_+48>:  0x0     0x0
0x4d02f0 <_IO_2_1_stdout_+64>:  0x0     0x0
0x4d0300 <_IO_2_1_stdout_+80>:  0x0     0x0
0x4d0310 <_IO_2_1_stdout_+96>:  0x0     0x4d04d8 <_IO_2_1_stdin_>
0x4d0320 <_IO_2_1_stdout_+112>: 0x1     0xffffffffffffffff
0x4d0330 <_IO_2_1_stdout_+128>: 0x0     0x4d6438 <_IO_stdfile_1_lock>
0x4d0340 <_IO_2_1_stdout_+144>: 0xffffffffffffffff      0x0
--Type <RET> for more, q to quit, c to continue without paging--
0x4d0350 <_IO_2_1_stdout_+160>: 0x4d0390 <_IO_wide_data_1>      0x0
0x4d0360 <_IO_2_1_stdout_+176>: 0x0     0x0
0x4d0370 <_IO_2_1_stdout_+192>: 0x0     0x0
0x4d0380 <_IO_2_1_stdout_+208>: 0x0     0x4a1950 <_IO_file_jumps>
0x4d0390 <_IO_wide_data_1>:     0x0     0x0
0x4d03a0 <_IO_wide_data_1+16>:  0x0     0x0
0x4d03b0 <_IO_wide_data_1+32>:  0x0     0x0
0x4d03c0 <_IO_wide_data_1+48>:  0x0     0x0
0x4d03d0 <_IO_wide_data_1+64>:  0x0     0x0
0x4d03e0 <_IO_wide_data_1+80>:  0x0     0x0
0x4d03f0 <_IO_wide_data_1+96>:  0x0     0x0
0x4d0400 <_IO_wide_data_1+112>: 0x0     0x0
0x4d0410 <_IO_wide_data_1+128>: 0x0     0x0
0x4d0420 <_IO_wide_data_1+144>: 0x0     0x0
0x4d0430 <_IO_wide_data_1+160>: 0x0     0x0
0x4d0440 <_IO_wide_data_1+176>: 0x0     0x0
0x4d0450 <_IO_wide_data_1+192>: 0x0     0x0
0x4d0460 <_IO_wide_data_1+208>: 0x0     0x0
0x4d0470 <_IO_wide_data_1+224>: 0x0     0x0
0x4d0480 <_IO_wide_data_1+240>: 0x0     0x0
0x4d0490 <_IO_wide_data_1+256>: 0x0     0x0
```

```
0x4d04a0 <_IO_wide_data_1+272>: 0x0     0x0
0x4d04b0 <_IO_wide_data_1+288>: 0x0     0x0
0x4d04c0 <_IO_wide_data_1+304>: 0x0     0x0
0x4d04d0 <_IO_wide_data_1+320>: 0x4a1800 <_IO_wfile_jumps>      0xfbad2088
0x4d04e0 <_IO_2_1_stdin_+8>:    0x0     0x0
0x4d04f0 <_IO_2_1_stdin_+24>:   0x0     0x0
0x4d0500 <_IO_2_1_stdin_+40>:   0x0     0x0
0x4d0510 <_IO_2_1_stdin_+56>:   0x0     0x0
0x4d0520 <_IO_2_1_stdin_+72>:   0x0     0x0
0x4d0530 <_IO_2_1_stdin_+88>:   0x0     0x0
0x4d0540 <_IO_2_1_stdin_+104>:  0x0     0x0
0x4d0550 <_IO_2_1_stdin_+120>:  0xffffffffffffffff      0x0
0x4d0560 <_IO_2_1_stdin_+136>:  0x4d6448 <_IO_stdfile_0_lock>   0xffffffffffffffff
0x4d0570 <_IO_2_1_stdin_+152>:  0x0     0x4d05b8 <_IO_wide_data_0>
0x4d0580 <_IO_2_1_stdin_+168>:  0x0     0x0
0x4d0590 <_IO_2_1_stdin_+184>:  0x0     0x0
0x4d05a0 <_IO_2_1_stdin_+200>:  0x0     0x0
0x4d05b0 <_IO_2_1_stdin_+216>:  0x4a1950 <_IO_file_jumps>       0x0
0x4d05c0 <_IO_wide_data_0+8>:   0x0     0x0
0x4d05d0 <_IO_wide_data_0+24>:  0x0     0x0
0x4d05e0 <_IO_wide_data_0+40>:  0x0     0x0
0x4d05f0 <_IO_wide_data_0+56>:  0x0     0x0
0x4d0600 <_IO_wide_data_0+72>:  0x0     0x0
0x4d0610 <_IO_wide_data_0+88>:  0x0     0x0
0x4d0620 <_IO_wide_data_0+104>: 0x0     0x0
0x4d0630 <_IO_wide_data_0+120>: 0x0     0x0
0x4d0640 <_IO_wide_data_0+136>: 0x0     0x0
0x4d0650 <_IO_wide_data_0+152>: 0x0     0x0
0x4d0660 <_IO_wide_data_0+168>: 0x0     0x0
--Type <RET> for more, q to quit, c to continue without paging--
0x4d0670 <_IO_wide_data_0+184>: 0x0     0x0
0x4d0680 <_IO_wide_data_0+200>: 0x0     0x0
0x4d0690 <_IO_wide_data_0+216>: 0x0     0x0
0x4d06a0 <_IO_wide_data_0+232>: 0x0     0x0
0x4d06b0 <_IO_wide_data_0+248>: 0x0     0x0
0x4d06c0 <_IO_wide_data_0+264>: 0x0     0x0
0x4d06d0 <_IO_wide_data_0+280>: 0x0     0x0
0x4d06e0 <_IO_wide_data_0+296>: 0x0     0x0
0x4d06f0 <_IO_wide_data_0+312>: 0x0     0x4a1800 <_IO_wfile_jumps>
0x4d0700 <stderr>:      0x4d0088 <_IO_2_1_stderr_>      0x4d02b0 <_IO_2_1_stdout_>
0x4d0710 <stdin>:       0x4d04d8 <_IO_2_1_stdin_>       0x20000
0x4d0720 <mp_+8>:       0x20000 0x20000
0x4d0730 <mp_+24>:      0x8     0x0
0x4d0740 <mp_+40>:      0x1000000000000 0x0
0x4d0750 <mp_+56>:      0x0     0x0
0x4d0760 <mp_+72>:      0x1fa0f88       0x40
0x4d0770 <mp_+88>:      0x408   0x7
0x4d0780 <mp_+104>:     0x0     0x0
0x4d0790 <main_arena+8>:        0x0     0x0
0x4d07a0 <main_arena+24>:       0x0     0x0
0x4d07b0 <main_arena+40>:       0x0     0x0
0x4d07c0 <main_arena+56>:       0x0     0x0
0x4d07d0 <main_arena+72>:       0x0     0x0
0x4d07e0 <main_arena+88>:       0x0     0x1fa28a0
0x4d07f0 <main_arena+104>:      0x0     0x4d07e8 <main_arena+96>
0x4d0800 <main_arena+120>:      0x4d07e8 <main_arena+96>        0x4d07f8 <main_arena+112>
0x4d0810 <main_arena+136>:      0x4d07f8 <main_arena+112>       0x4d0808 <main_arena+128>
0x4d0820 <main_arena+152>:      0x4d0808 <main_arena+128>       0x4d0818 <main_arena+144>
0x4d0830 <main_arena+168>:      0x4d0818 <main_arena+144>       0x4d0828 <main_arena+160>
0x4d0840 <main_arena+184>:      0x4d0828 <main_arena+160>       0x4d0838 <main_arena+176>
```

```
0x4d0850 <main_arena+200>:      0x4d0838 <main_arena+176>       0x4d0848 <main_arena+192>
0x4d0860 <main_arena+216>:      0x4d0848 <main_arena+192>       0x4d0858 <main_arena+208>
0x4d0870 <main_arena+232>:      0x4d0858 <main_arena+208>       0x4d0868 <main_arena+224>
0x4d0880 <main_arena+248>:      0x4d0868 <main_arena+224>       0x4d0878 <main_arena+240>
0x4d0890 <main_arena+264>:      0x4d0878 <main_arena+240>       0x4d0888 <main_arena+256>
0x4d08a0 <main_arena+280>:      0x4d0888 <main_arena+256>       0x4d0898 <main_arena+272>
0x4d08b0 <main_arena+296>:      0x4d0898 <main_arena+272>       0x4d08a8 <main_arena+288>
0x4d08c0 <main_arena+312>:      0x4d08a8 <main_arena+288>       0x4d08b8 <main_arena+304>
0x4d08d0 <main_arena+328>:      0x4d08b8 <main_arena+304>       0x4d08c8 <main_arena+320>
0x4d08e0 <main_arena+344>:      0x4d08c8 <main_arena+320>       0x4d08d8 <main_arena+336>
0x4d08f0 <main_arena+360>:      0x4d08d8 <main_arena+336>       0x4d08e8 <main_arena+352>
0x4d0900 <main_arena+376>:      0x4d08e8 <main_arena+352>       0x4d08f8 <main_arena+368>
0x4d0910 <main_arena+392>:      0x4d08f8 <main_arena+368>       0x4d0908 <main_arena+384>
0x4d0920 <main_arena+408>:      0x4d0908 <main_arena+384>       0x4d0918 <main_arena+400>
0x4d0930 <main_arena+424>:      0x4d0918 <main_arena+400>       0x4d0928 <main_arena+416>
0x4d0940 <main_arena+440>:      0x4d0928 <main_arena+416>       0x4d0938 <main_arena+432>
0x4d0950 <main_arena+456>:      0x4d0938 <main_arena+432>       0x4d0948 <main_arena+448>
0x4d0960 <main_arena+472>:      0x4d0948 <main_arena+448>       0x4d0958 <main_arena+464>
0x4d0970 <main_arena+488>:      0x4d0958 <main_arena+464>       0x4d0968 <main_arena+480>
0x4d0980 <main_arena+504>:      0x4d0968 <main_arena+480>       0x4d0978 <main_arena+496>
--Type <RET> for more, q to quit, c to continue without paging--
0x4d0990 <main_arena+520>:      0x4d0978 <main_arena+496>       0x4d0988 <main_arena+512>
0x4d09a0 <main_arena+536>:      0x4d0988 <main_arena+512>       0x4d0998 <main_arena+528>
0x4d09b0 <main_arena+552>:      0x4d0998 <main_arena+528>       0x4d09a8 <main_arena+544>
0x4d09c0 <main_arena+568>:      0x4d09a8 <main_arena+544>       0x4d09b8 <main_arena+560>
0x4d09d0 <main_arena+584>:      0x4d09b8 <main_arena+560>       0x4d09c8 <main_arena+576>
0x4d09e0 <main_arena+600>:      0x4d09c8 <main_arena+576>       0x4d09d8 <main_arena+592>
0x4d09f0 <main_arena+616>:      0x4d09d8 <main_arena+592>       0x4d09e8 <main_arena+608>
0x4d0a00 <main_arena+632>:      0x4d09e8 <main_arena+608>       0x4d09f8 <main_arena+624>
0x4d0a10 <main_arena+648>:      0x4d09f8 <main_arena+624>       0x4d0a08 <main_arena+640>
0x4d0a20 <main_arena+664>:      0x4d0a08 <main_arena+640>       0x4d0a18 <main_arena+656>
0x4d0a30 <main_arena+680>:      0x4d0a18 <main_arena+656>       0x4d0a28 <main_arena+672>
0x4d0a40 <main_arena+696>:      0x4d0a28 <main_arena+672>       0x4d0a38 <main_arena+688>
0x4d0a50 <main_arena+712>:      0x4d0a38 <main_arena+688>       0x4d0a48 <main_arena+704>
0x4d0a60 <main_arena+728>:      0x4d0a48 <main_arena+704>       0x4d0a58 <main_arena+720>
0x4d0a70 <main_arena+744>:      0x4d0a58 <main_arena+720>       0x4d0a68 <main_arena+736>
0x4d0a80 <main_arena+760>:      0x4d0a68 <main_arena+736>       0x4d0a78 <main_arena+752>
0x4d0a90 <main_arena+776>:      0x4d0a78 <main_arena+752>       0x4d0a88 <main_arena+768>
0x4d0aa0 <main_arena+792>:      0x4d0a88 <main_arena+768>       0x4d0a98 <main_arena+784>
0x4d0ab0 <main_arena+808>:      0x4d0a98 <main_arena+784>       0x4d0aa8 <main_arena+800>
0x4d0ac0 <main_arena+824>:      0x4d0aa8 <main_arena+800>       0x4d0ab8 <main_arena+816>
0x4d0ad0 <main_arena+840>:      0x4d0ab8 <main_arena+816>       0x4d0ac8 <main_arena+832>
0x4d0ae0 <main_arena+856>:      0x4d0ac8 <main_arena+832>       0x4d0ad8 <main_arena+848>
0x4d0af0 <main_arena+872>:      0x4d0ad8 <main_arena+848>       0x4d0ae8 <main_arena+864>
0x4d0b00 <main_arena+888>:      0x4d0ae8 <main_arena+864>       0x4d0af8 <main_arena+880>
0x4d0b10 <main_arena+904>:      0x4d0af8 <main_arena+880>       0x4d0b08 <main_arena+896>
0x4d0b20 <main_arena+920>:      0x4d0b08 <main_arena+896>       0x4d0b18 <main_arena+912>
0x4d0b30 <main_arena+936>:      0x4d0b18 <main_arena+912>       0x4d0b28 <main_arena+928>
0x4d0b40 <main_arena+952>:      0x4d0b28 <main_arena+928>       0x4d0b38 <main_arena+944>
0x4d0b50 <main_arena+968>:      0x4d0b38 <main_arena+944>       0x4d0b48 <main_arena+960>
0x4d0b60 <main_arena+984>:      0x4d0b48 <main_arena+960>       0x4d0b58 <main_arena+976>
0x4d0b70 <main_arena+1000>:     0x4d0b58 <main_arena+976>       0x4d0b68 <main_arena+992>
0x4d0b80 <main_arena+1016>:     0x4d0b68 <main_arena+992>       0x4d0b78 <main_arena+1008>
0x4d0b90 <main_arena+1032>:     0x4d0b78 <main_arena+1008>      0x4d0b88 <main_arena+1024>
0x4d0ba0 <main_arena+1048>:     0x4d0b88 <main_arena+1024>      0x4d0b98 <main_arena+1040>
0x4d0bb0 <main_arena+1064>:     0x4d0b98 <main_arena+1040>      0x4d0ba8 <main_arena+1056>
0x4d0bc0 <main_arena+1080>:     0x4d0ba8 <main_arena+1056>      0x4d0bb8 <main_arena+1072>
0x4d0bd0 <main_arena+1096>:     0x4d0bb8 <main_arena+1072>      0x4d0bc8 <main_arena+1088>
0x4d0be0 <main_arena+1112>:     0x4d0bc8 <main_arena+1088>      0x4d0bd8 <main_arena+1104>
0x4d0bf0 <main_arena+1128>:     0x4d0bd8 <main_arena+1104>      0x4d0be8 <main_arena+1120>
```

```
0x4d0c00 <main_arena+1144>:     0x4d0be8 <main_arena+1120>      0x4d0bf8 <main_arena+1136>
0x4d0c10 <main_arena+1160>:     0x4d0bf8 <main_arena+1136>      0x4d0c08 <main_arena+1152>
0x4d0c20 <main_arena+1176>:     0x4d0c08 <main_arena+1152>      0x4d0c18 <main_arena+1168>
0x4d0c30 <main_arena+1192>:     0x4d0c18 <main_arena+1168>      0x4d0c28 <main_arena+1184>
0x4d0c40 <main_arena+1208>:     0x4d0c28 <main_arena+1184>      0x4d0c38 <main_arena+1200>
0x4d0c50 <main_arena+1224>:     0x4d0c38 <main_arena+1200>      0x4d0c48 <main_arena+1216>
0x4d0c60 <main_arena+1240>:     0x4d0c48 <main_arena+1216>      0x4d0c58 <main_arena+1232>
0x4d0c70 <main_arena+1256>:     0x4d0c58 <main_arena+1232>      0x4d0c68 <main_arena+1248>
0x4d0c80 <main_arena+1272>:     0x4d0c68 <main_arena+1248>      0x4d0c78 <main_arena+1264>
0x4d0c90 <main_arena+1288>:     0x4d0c78 <main_arena+1264>      0x4d0c88 <main_arena+1280>
0x4d0ca0 <main_arena+1304>:     0x4d0c88 <main_arena+1280>      0x4d0c98 <main_arena+1296>
--Type <RET> for more, q to quit, c to continue without paging--
0x4d0cb0 <main_arena+1320>:     0x4d0c98 <main_arena+1296>      0x4d0ca8 <main_arena+1312>
0x4d0cc0 <main_arena+1336>:     0x4d0ca8 <main_arena+1312>      0x4d0cb8 <main_arena+1328>
0x4d0cd0 <main_arena+1352>:     0x4d0cb8 <main_arena+1328>      0x4d0cc8 <main_arena+1344>
0x4d0ce0 <main_arena+1368>:     0x4d0cc8 <main_arena+1344>      0x4d0cd8 <main_arena+1360>
0x4d0cf0 <main_arena+1384>:     0x4d0cd8 <main_arena+1360>      0x4d0ce8 <main_arena+1376>
0x4d0d00 <main_arena+1400>:     0x4d0ce8 <main_arena+1376>      0x4d0cf8 <main_arena+1392>
0x4d0d10 <main_arena+1416>:     0x4d0cf8 <main_arena+1392>      0x4d0d08 <main_arena+1408>
0x4d0d20 <main_arena+1432>:     0x4d0d08 <main_arena+1408>      0x4d0d18 <main_arena+1424>
0x4d0d30 <main_arena+1448>:     0x4d0d18 <main_arena+1424>      0x4d0d28 <main_arena+1440>
0x4d0d40 <main_arena+1464>:     0x4d0d28 <main_arena+1440>      0x4d0d38 <main_arena+1456>
0x4d0d50 <main_arena+1480>:     0x4d0d38 <main_arena+1456>      0x4d0d48 <main_arena+1472>
0x4d0d60 <main_arena+1496>:     0x4d0d48 <main_arena+1472>      0x4d0d58 <main_arena+1488>
0x4d0d70 <main_arena+1512>:     0x4d0d58 <main_arena+1488>      0x4d0d68 <main_arena+1504>
0x4d0d80 <main_arena+1528>:     0x4d0d68 <main_arena+1504>      0x4d0d78 <main_arena+1520>
0x4d0d90 <main_arena+1544>:     0x4d0d78 <main_arena+1520>      0x4d0d88 <main_arena+1536>
0x4d0da0 <main_arena+1560>:     0x4d0d88 <main_arena+1536>      0x4d0d98 <main_arena+1552>
0x4d0db0 <main_arena+1576>:     0x4d0d98 <main_arena+1552>      0x4d0da8 <main_arena+1568>
0x4d0dc0 <main_arena+1592>:     0x4d0da8 <main_arena+1568>      0x4d0db8 <main_arena+1584>
0x4d0dd0 <main_arena+1608>:     0x4d0db8 <main_arena+1584>      0x4d0dc8 <main_arena+1600>
0x4d0de0 <main_arena+1624>:     0x4d0dc8 <main_arena+1600>      0x4d0dd8 <main_arena+1616>
0x4d0df0 <main_arena+1640>:     0x4d0dd8 <main_arena+1616>      0x4d0de8 <main_arena+1632>
0x4d0e00 <main_arena+1656>:     0x4d0de8 <main_arena+1632>      0x4d0df8 <main_arena+1648>
0x4d0e10 <main_arena+1672>:     0x4d0df8 <main_arena+1648>      0x4d0e08 <main_arena+1664>
0x4d0e20 <main_arena+1688>:     0x4d0e08 <main_arena+1664>      0x4d0e18 <main_arena+1680>
0x4d0e30 <main_arena+1704>:     0x4d0e18 <main_arena+1680>      0x4d0e28 <main_arena+1696>
0x4d0e40 <main_arena+1720>:     0x4d0e28 <main_arena+1696>      0x4d0e38 <main_arena+1712>
0x4d0e50 <main_arena+1736>:     0x4d0e38 <main_arena+1712>      0x4d0e48 <main_arena+1728>
0x4d0e60 <main_arena+1752>:     0x4d0e48 <main_arena+1728>      0x4d0e58 <main_arena+1744>
0x4d0e70 <main_arena+1768>:     0x4d0e58 <main_arena+1744>      0x4d0e68 <main_arena+1760>
0x4d0e80 <main_arena+1784>:     0x4d0e68 <main_arena+1760>      0x4d0e78 <main_arena+1776>
0x4d0e90 <main_arena+1800>:     0x4d0e78 <main_arena+1776>      0x4d0e88 <main_arena+1792>
0x4d0ea0 <main_arena+1816>:     0x4d0e88 <main_arena+1792>      0x4d0e98 <main_arena+1808>
0x4d0eb0 <main_arena+1832>:     0x4d0e98 <main_arena+1808>      0x4d0ea8 <main_arena+1824>
0x4d0ec0 <main_arena+1848>:     0x4d0ea8 <main_arena+1824>      0x4d0eb8 <main_arena+1840>
0x4d0ed0 <main_arena+1864>:     0x4d0eb8 <main_arena+1840>      0x4d0ec8 <main_arena+1856>
0x4d0ee0 <main_arena+1880>:     0x4d0ec8 <main_arena+1856>      0x4d0ed8 <main_arena+1872>
0x4d0ef0 <main_arena+1896>:     0x4d0ed8 <main_arena+1872>      0x4d0ee8 <main_arena+1888>
0x4d0f00 <main_arena+1912>:     0x4d0ee8 <main_arena+1888>      0x4d0ef8 <main_arena+1904>
0x4d0f10 <main_arena+1928>:     0x4d0ef8 <main_arena+1904>      0x4d0f08 <main_arena+1920>
0x4d0f20 <main_arena+1944>:     0x4d0f08 <main_arena+1920>      0x4d0f18 <main_arena+1936>
0x4d0f30 <main_arena+1960>:     0x4d0f18 <main_arena+1936>      0x4d0f28 <main_arena+1952>
0x4d0f40 <main_arena+1976>:     0x4d0f28 <main_arena+1952>      0x4d0f38 <main_arena+1968>
0x4d0f50 <main_arena+1992>:     0x4d0f38 <main_arena+1968>      0x4d0f48 <main_arena+1984>
0x4d0f60 <main_arena+2008>:     0x4d0f48 <main_arena+1984>      0x4d0f58 <main_arena+2000>
0x4d0f70 <main_arena+2024>:     0x4d0f58 <main_arena+2000>      0x4d0f68 <main_arena+2016>
0x4d0f80 <main_arena+2040>:     0x4d0f68 <main_arena+2016>      0x4d0f78 <main_arena+2032>
0x4d0f90 <main_arena+2056>:     0x4d0f78 <main_arena+2032>      0x4d0f88 <main_arena+2048>
0x4d0fa0 <main_arena+2072>:     0x4d0f88 <main_arena+2048>      0x4d0f98 <main_arena+2064>
```

```
0x4d0fb0 <main_arena+2088>:     0x4d0f98 <main_arena+2064>      0x4d0fa8 <main_arena+2080>
0x4d0fc0 <main_arena+2104>:     0x4d0fa8 <main_arena+2080>      0x4d0fb8 <main_arena+2096>
--Type <RET> for more, q to quit, c to continue without paging--
0x4d0fd0 <main_arena+2120>:     0x4d0fb8 <main_arena+2096>      0x4d0fc8 <main_arena+2112>
0x4d0fe0 <main_arena+2136>:     0x4d0fc8 <main_arena+2112>      0x0
0x4d0ff0 <main_arena+2152>:     0x0     0x4d0788 <main_arena>
0x4d1000 <main_arena+2168>:     0x0     0x1
0x4d1010 <main_arena+2184>:     0x3f078 0x3f078
0x4d1020 <__morecore>:  0x421c08 <__default_morecore>   0x1
0x4d1030 <__libc_malloc_initialized>:   0xffffffff00000001      0x41cc00 <memalign_hook_ini>
0x4d1040 <__realloc_hook>:      0x41d688 <realloc_hook_ini>     0x0
0x4d1050 <LogFacility>: 0xffffffff00000008      0xff00000002
0x4d1060 <cached_result.10628>: 0xffffffff      0xffffd30bf6dd
0x4d1070 <program_invocation_name>:     0xffffd30bf6db  0x10000
0x4d1080 <_dl_stack_flags>:     0x6     0x0
0x4d1090 <_dl_load_write_lock+8>:       0x0     0x1
0x4d10a0 <_dl_load_write_lock+24>:      0x0     0x0
0x4d10b0 <_dl_load_write_lock+40>:      0x0     0x0
0x4d10c0 <_dl_load_lock+8>:     0x0     0x1
0x4d10d0 <_dl_load_lock+24>:    0x0     0x0
0x4d10e0 <_dl_load_lock+40>:    0x0     0x42c6a0 <_dl_make_stack_executable>
0x4d10f0 <_dl_correct_cache_id>:        0x200000a03     0x4045a8 <__pthread_init_static_tls>
0x4d1100 <_dl_starting_up>:     0x1     0xfffffffffffffffe
0x4d1110 <_dl_argv>:    0x4d1068 <program_invocation_short_name>        0x0
0x4d1120 <builtin_modules>:     0x48ad20        0x48ac30
0x4d1130 <builtin_modules+16>:  0x7fffffff00000001      0x48ac40
0x4d1140 <builtin_modules+32>:  0x0     0x0
0x4d1150 <builtin_modules+48>:  0x0     0x48ac30
0x4d1160 <builtin_modules+64>:  0x48ad20        0x7fffffff00000001
0x4d1170 <builtin_modules+80>:  0x48ac50        0x0
0x4d1180 <builtin_modules+96>:  0x0     0x0
0x4d1190 <builtin_modules+112>: 0x48ad20        0x48ac60
0x4d11a0 <builtin_modules+128>: 0x7fffffff00000001      0x48ac70
0x4d11b0 <builtin_modules+144>: 0x0     0x0
0x4d11c0 <builtin_modules+160>: 0x0     0x48ac60
0x4d11d0 <builtin_modules+176>: 0x48ad20        0x7fffffff00000001
0x4d11e0 <builtin_modules+192>: 0x48ac88        0x0
0x4d11f0 <builtin_modules+208>: 0x0     0x0
0x4d1200 <builtin_modules+224>: 0x48ad20        0x48aca0
0x4d1210 <builtin_modules+240>: 0x7fffffff00000001      0x48acb0
0x4d1220 <builtin_modules+256>: 0x0     0x0
0x4d1230 <builtin_modules+272>: 0x0     0x48aca0
0x4d1240 <builtin_modules+288>: 0x48ad20        0x7fffffff00000001
0x4d1250 <builtin_modules+304>: 0x48acc0        0x0
0x4d1260 <builtin_modules+320>: 0x0     0x0
0x4d1270 <builtin_modules+336>: 0x48acd0        0x48ad20
0x4d1280 <builtin_modules+352>: 0x7fffffff00000001      0x48ace0
0x4d1290 <builtin_modules+368>: 0x0     0x0
0x4d12a0 <builtin_modules+384>: 0x0     0x48ad20
0x4d12b0 <builtin_modules+400>: 0x48acd0        0x7fffffff00000001
0x4d12c0 <builtin_modules+416>: 0x48acf0        0x0
0x4d12d0 <builtin_modules+432>: 0x0     0x0
0x4d12e0 <builtin_modules+448>: 0x48ad00        0x48ad20
```

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
```

Each value may also have an associated symbolic value:

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

For example, from the output above:

```
0x4d1110 <_dl_argv>:    0x4d1068 <program_invocation_short_name>        0x0
```

11. Explore the contents of memory pointed to by *__nptl_nthreads*, *program_invocation_short_name*, and *__realloc_hook* addresses (**/x** is for hex, **/d** is for decimals, **/u** is for unsigned decimals, **/g** is for 64-bit values, **/w** is for 32-bit values, **/h** is for 16-bit values, **/b** is for byte values, **/a** is for addresses, **/c** and **/s** are for chars and strings):

```
(gdb) x/d &__nptl_nthreads
0x4d0048 <__nptl_nthreads>:     6

(gdb) x/u 0x4d0048
0x4d0048 <__nptl_nthreads>:     6

(gdb) x/wx 0x4d0048
0x4d0048 <__nptl_nthreads>:     0x00000006

(gdb) x/gx 0x4d0048
0x4d0048 <__nptl_nthreads>:     0x0000000000000006

(gdb) x/hx 0x4d0048
0x4d0048 <__nptl_nthreads>:     0x0006

(gdb) x/bx 0x4d0048
0x4d0048 <__nptl_nthreads>:     0x06

(gdb) x/a 0x4d1068
0x4d1068 <program_invocation_short_name>:       0xffffd30bf6dd

(gdb) x/a 0xffffd30bf6dd
0xffffd30bf6dd: 0x4744580031707041

(gdb) x/s 0xffffd30bf6dd
0xffffd30bf6dd: "App1"

(gdb) x/8c 0xffffd30bf6dd
0xffffd30bf6dd: 65 'A'  112 'p' 112 'p' 49 '1'  0 '\000'        88 'X'  68 'D'  71 'G'

(gdb) x/10s 0xffffd30bf6dd
0xffffd30bf6dd: "App1"
0xffffd30bf6e2: "XDG_SESSION_ID=6850"
0xffffd30bf6f6: "HOSTNAME=instance-20211109-2004"
0xffffd30bf716: "SELINUX_ROLE_REQUESTED="
0xffffd30bf72e: "TERM=xterm-256color"
0xffffd30bf742: "SHELL=/bin/bash"
0xffffd30bf752: "HISTSIZE=1000"
0xffffd30bf760: "SSH_CLIENT=37.228.238.120 61099 22"
0xffffd30bf783: "SELINUX_USE_CURRENT_RANGE="
0xffffd30bf79e: "SSH_TTY=/dev/pts/1"

(gdb) x/a &__realloc_hook
0x4d1040 <__realloc_hook>:      0x41d688 <realloc_hook_ini>
```

```
(gdb) x/10i 0x41d688
   0x41d688 <realloc_hook_ini>: stp     x29, x30, [sp, #-112]!
   0x41d68c <realloc_hook_ini+4>:       mov     x29, sp
   0x41d690 <realloc_hook_ini+8>:       stp     x25, x26, [sp, #64]
   0x41d694 <realloc_hook_ini+12>:      adrp    x25, 0x4d0000
   0x41d698 <realloc_hook_ini+16>:      add     x2, x25, #0x718
   0x41d69c <realloc_hook_ini+20>:      stp     x21, x22, [sp, #32]
   0x41d6a0 <realloc_hook_ini+24>:      ldr     w3, [x2, #2328]
   0x41d6a4 <realloc_hook_ini+28>:      ldr     x21, 0x41da48
   0x41d6a8 <realloc_hook_ini+32>:      ldr     x2, 0x41da40
   0x41d6ac <realloc_hook_ini+36>:      stp     x19, x20, [sp, #16]
```

Note: We see that a hook function is installed for *realloc*. Please find the following documentation for hook functions here:

https://www.gnu.org/software/libc/manual/html_node/Hooks-for-Malloc.html

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

```
(gdb) x/a &environ
0x4d64c8 <environ>:     0xffffd30b8888

(gdb) x/10a 0xffffd30b8888
0xffffd30b8888: 0xffffd30bf6e2  0xffffd30bf6f6
0xffffd30b8898: 0xffffd30bf716  0xffffd30bf72e
0xffffd30b88a8: 0xffffd30bf742  0xffffd30bf752
0xffffd30b88b8: 0xffffd30bf760  0xffffd30bf783
0xffffd30b88c8: 0xffffd30bf79e  0xffffd30bf7b1

(gdb) x/10s 0xffffd30bf6e2
0xffffd30bf6e2: "XDG_SESSION_ID=6850"
0xffffd30bf6f6: "HOSTNAME=instance-20211109-2004"
0xffffd30bf716: "SELINUX_ROLE_REQUESTED="
0xffffd30bf72e: "TERM=xterm-256color"
0xffffd30bf742: "SHELL=/bin/bash"
0xffffd30bf752: "HISTSIZE=1000"
0xffffd30bf760: "SSH_CLIENT=37.228.238.120 61099 22"
0xffffd30bf783: "SELINUX_USE_CURRENT_RANGE="
0xffffd30bf79e: "SSH_TTY=/dev/pts/1"
0xffffd30bf7b1: "USER=opc"
```

13. Now, we look at how to perform a memory search. It is not possible to search in the entire virtual memory, only in the valid regions.

```
(gdb) find /g 0x004d0030, 0x005d0030, 6
0x4d0048 <__nptl_nthreads>
0x4d1080 <_dl_stack_flags>
0x4d7e00 <_dl_phnum>
warning: Unable to access 16000 bytes of target memory at 0x4dfb08, halting search.
3 patterns found.

(gdb) x/gd 0x4d0048
0x4d0048 <__nptl_nthreads>:     6

(gdb) find 0xffffd30bf6e2, +1000, "bash"
0xffffd30bf74d
1 pattern found.
```

```
(gdb) x/s 0xffffd30bf74d-11
0xffffd30bf742: "SHELL=/bin/bash"
```

Note: "bash" is considered a null-terminated array of characters for the search. To search for a string sequence without a null terminator, use a sequence of characters:

```
(gdb) find 0xffffd30bf6e2, +1000, "bin"
Pattern not found.

(gdb) find 0xffffd30bf6e2, +1000, 'b', 'i', 'n'
0xffffd30bf749
1 pattern found.
```

14. Get the list of loaded modules:

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

Note: 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 *App1.shared.core.184724* from the App1S directory, we see the list of shared libraries:

```
~/ALCDA2/A64/App1S$ gdb-multiarch -c App1.shared.core.184724 -se App1.shared
GNU gdb (Debian 8.2.1-2+b3) 8.2.1
Copyright (C) 2018 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".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from App1.shared...(no debugging symbols found)...done.
[New LWP 184724]
[New LWP 184725]
[New LWP 184726]
[New LWP 184727]
[New LWP 184728]
[New LWP 184729]

warning: Could not load shared library symbols for 2 libraries, e.g. /lib/aarch64-linux-
gnu/libc.so.6.
Use the "info sharedlibrary" command to see the complete listing.
Do you need "set solib-search-path" or "set sysroot"?
Core was generated by `./App1.shared'.
#0  0x0000ffff81451924 in ?? ()
[Current thread is 1 (LWP 184724)]
```

Note: Since GDB needs shared libraries from the crash system and we don't have them, most likely you get some random symbolic information (and, also, an invalid backtrace from the **bt** command):

```
(gdb) bt
#0  0x0000ffff81451924 in ?? ()
#1  0x0000ffff81586640 in ?? ()
Backtrace stopped: previous frame inner to this frame (corrupt stack?)

(gdb) info sharedlibrary
From                To                  Syms Read   Shared Object Library
                                        No          /lib/aarch64-linux-gnu/libc.so.6
                                        No          /lib/ld-linux-aarch64.so.1
```

15. Since we have saved shared libraries, we set the search path for them:

```
(gdb) set solib-search-path .
Reading symbols from /home/coredump/ALCDA2/A64/App1S/libc.so.6...(no debugging symbols
found)...done.
Reading symbols from /home/coredump/ALCDA2/A64/App1S/ld-linux-aarch64.so.1...(no debugging
symbols found)...done.

(gdb) bt
#0  0x0000ffff81451924 in clock_nanosleep () from /home/coredump/ALCDA2/A64/App1S/libc.so.6
#1  0x0000ffff81456aec in nanosleep () from /home/coredump/ALCDA2/A64/App1S/libc.so.6
#2  0x0000ffff814569b8 in sleep () from /home/coredump/ALCDA2/A64/App1S/libc.so.6
#3  0x0000aaaad0be0ab4 in main ()
```

16. Disassemble the *bar_one* function and follow the indirect *sleep* function call:

```
(gdb) disassemble bar_one
Dump of assembler code for function bar_one:
   0x0000aaaad0be0894 <+0>:     stp     x29, x30, [sp, #-16]!
   0x0000aaaad0be0898 <+4>:     mov     x29, sp
   0x0000aaaad0be089c <+8>:     mov     w0, #0xffffffff                 // #-1
   0x0000aaaad0be08a0 <+12>:    bl      0xaaaad0be0710 <sleep@plt>
   0x0000aaaad0be08a4 <+16>:    ldp     x29, x30, [sp], #16
   0x0000aaaad0be08a8 <+20>:    ret
End of assembler dump.

(gdb) disassemble 0xaaaad0be0710
Dump of assembler code for function sleep@plt:
   0x0000aaaad0be0710 <+0>:     adrp    x16, 0xaaaad0bf1000
   0x0000aaaad0be0714 <+4>:     ldr     x17, [x16, #4000]
   0x0000aaaad0be0718 <+8>:     add     x16, x16, #0xfa0
   0x0000aaaad0be071c <+12>:    br      x17
End of assembler dump.

(gdb) x/a 0xaaaad0bf1000+4000
0xaaaad0bf1fa0 <sleep@got.plt>: 0xffff81456970 <__sleep>
```

17. *App1.shared.pmap.184724* also shows library memory regions:

```
(gdb) q

~/ALCDA2/A64/App1S$ cat App1.shared.pmap.184724
184724:   ./App1.shared
0000aaaad0be0000      4K r-x-- App1.shared
0000aaaad0bf1000      4K r---- App1.shared
0000aaaad0bf2000      4K rw--- App1.shared
0000aaaafe503000    132K rw---   [ anon ]
0000ffff7eb50000     64K -----   [ anon ]
0000ffff7eb60000   8192K rw---   [ anon ]
```

```
0000ffff7f360000     64K -----   [ anon ]
0000ffff7f370000   8192K rw---   [ anon ]
0000ffff7fb70000     64K -----   [ anon ]
0000ffff7fb80000   8192K rw---   [ anon ]
0000ffff80380000     64K -----   [ anon ]
0000ffff80390000   8192K rw---   [ anon ]
0000ffff80b90000     64K -----   [ anon ]
0000ffff80ba0000   8192K rw---   [ anon ]
0000ffff813a0000   1572K r-x-- libc.so.6
0000ffff81529000     60K ----- libc.so.6
0000ffff81538000     16K r---- libc.so.6
0000ffff8153c000      8K rw--- libc.so.6
0000ffff8153e000     48K rw---   [ anon ]
0000ffff81550000    172K r-x-- ld-linux-aarch64.so.1
0000ffff81585000      8K rw---   [ anon ]
0000ffff81587000      8K r----   [ anon ]
0000ffff81589000      4K r-x--   [ anon ]
0000ffff8158a000      8K r---- ld-linux-aarch64.so.1
0000ffff8158c000      8K rw--- ld-linux-aarch64.so.1
0000ffffc23c8000    132K rw---   [ stack ]
 total            43468K
```

Exercise A1 (x64, WinDbg)

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

Patterns: Manual Dump; Stack Trace; Incorrect Stack Trace; Unrecognizable Symbolic Information; Stack Trace Collection; Annotated Disassembly; Paratext; Not My Version; Environment Hint.

1. Launch WinDbg.

2. Load the core dump *App1.core.253* from the x64\App1 folder:

```
Microsoft (R) Windows Debugger Version 10.0.27725.1000 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.


Loading Dump File [C:\ALCDA2\x64\App1\App1.core.253]
64-bit machine not using 64-bit API

************* Path validation summary **************
Response                         Time (ms)     Location
Deferred                                       srv*
Symbol search path is: srv*
Executable search path is:
Generic Unix Version 0 UP Free x64
System Uptime: not available
Process Uptime: not available
..
*** WARNING: Unable to verify timestamp for App1
App1+0x41a10:
00000000`00441a10 cmp     rax,0FFFFFFFFFFFFF000h
```

3. Set logging to a file in case of lengthy output from some commands:

```
0:000> .logopen C:\ALCDA2\x64\App1\App1-WinDbg.log
Opened log file 'C:\ALCDA2\x64\App1\App1-WinDbg.log'
```

4. Specify the dump folder as the symbol path and reload symbols:

```
0:000> .symfix; .sympath+ C:\ALCDA2\x64\App1\
Symbol search path is: srv*;C:\ALCDA2\x64\App1\
Expanded Symbol search path is:
cache*;SRV*https://msdl.microsoft.com/download/symbols;c:\alcda2\x64\app1\

************* Path validation summary **************
Response                         Time (ms)     Location
Deferred                                       srv*
OK                                             C:\ALCDA2\x64\App1\
*** WARNING: Unable to verify timestamp for App1

0:000> .reload
..
*** WARNING: Unable to verify timestamp for App1

************* Symbol Loading Error Summary **************
Module name            Error
App1                   The system cannot find the file specified
```

```
You can troubleshoot most symbol related issues by turning on symbol loading diagnostics (!sym
noisy) and repeating the command that caused symbols to be loaded.
You should also verify that your symbol search path (.sympath) is correct.
```

Note: We ignore warnings and errors as they are not relevant for now.

5. List all threads:

```
0:000> ~
Unable to get thread data for thread 0
.  0  Id: fd.fd Suspend: 0 Teb: 00000000`00000000 Unfrozen
Unable to get thread data for thread 1
   1  Id: fd.fe Suspend: 0 Teb: 00000000`00000000 Unfrozen
Unable to get thread data for thread 2
   2  Id: fd.ff Suspend: 0 Teb: 00000000`00000000 Unfrozen
Unable to get thread data for thread 3
   3  Id: fd.100 Suspend: 0 Teb: 00000000`00000000 Unfrozen
Unable to get thread data for thread 4
   4  Id: fd.101 Suspend: 0 Teb: 00000000`00000000 Unfrozen
Unable to get thread data for thread 5
   5  Id: fd.102 Suspend: 0 Teb: 00000000`00000000 Unfrozen
```

Note: WinDbg uses the same output format as for Windows memory dumps. Therefore, some data is either reported as errors or shows 0 or NULL pointer values. However, we see process and thread IDs in the format PID.TID:

```
0:000> .formats fd
Evaluate expression:
  Hex:     00000000`000000fd
  Decimal: 253
  Decimal (unsigned) : 253
  Octal:   0000000000000000375
  Binary:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 11111101
  Chars:   ........
  Time:    Thu Jan  1 00:04:13 1970
  Float:   low 3.54529e-043 high 0
  Double:  1.24999e-321

0:000> ? fd
Evaluate expression: 253 = 00000000`000000fd
```

6. Get the current thread stack trace:

```
0:000> k
# Child-SP          RetAddr               Call Site
00 00007ffd`f4563610 00000000`00000000     App1+0x41a10
```

Note: We see the truncated stack trace because WinDbg couldn't get symbols for some reason. We direct it to try again:

```
0:000> .symopt+ 0x40
Symbol options are 0x30377:
  0x00000001 - SYMOPT_CASE_INSENSITIVE
  0x00000002 - SYMOPT_UNDNAME
  0x00000004 - SYMOPT_DEFERRED_LOADS
  0x00000010 - SYMOPT_LOAD_LINES
  0x00000020 - SYMOPT_OMAP_FIND_NEAREST
  0x00000040 - SYMOPT_LOAD_ANYTHING
```

```
  0x00000100 - SYMOPT_NO_UNQUALIFIED_LOADS
  0x00000200 - SYMOPT_FAIL_CRITICAL_ERRORS
  0x00010000 - SYMOPT_AUTO_PUBLICS
  0x00020000 - SYMOPT_NO_IMAGE_SEARCH
*** WARNING: Unable to verify timestamp for App1

0:000> .reload
..
*** WARNING: Unable to verify timestamp for App1

************* Symbol Loading Error Summary **************
Module name            Error
App1                   The system cannot find the file specified

You can troubleshoot most symbol related issues by turning on symbol loading diagnostics (!sym
noisy) and repeating the command that caused symbols to be loaded.
You should also verify that your symbol search path (.sympath) is correct.

0:000> k
 # Child-SP          RetAddr               Call Site
00 00007ffd`f4563610 00000000`0044199a     App1!nanosleep+0x40
01 00007ffd`f4563640 00000000`00401d92     App1!sleep+0x3a
02 00007ffd`f4563680 00000000`00407581     App1!main+0xaa
03 00007ffd`f45636d0 00000000`00401aba     App1!_libc_start_main+0x3d1
04 00007ffd`f45637d0 ffffffff`ffffffff     App1!start+0x2a
05 00007ffd`f45637d8 00000000`00000000     0xffffffff`ffffffff
```

7. Get all thread stack traces without source code references (**L**):

```
0:000> ~*kL

Unable to get thread data for thread 0
.  0  Id: fd.fd Suspend: 0 Teb: 00000000`00000000 Unfrozen
 # Child-SP          RetAddr               Call Site
00 00007ffd`f4563610 00000000`0044199a     App1!nanosleep+0x40
01 00007ffd`f4563640 00000000`00401d92     App1!sleep+0x3a
02 00007ffd`f4563680 00000000`00407581     App1!main+0xaa
03 00007ffd`f45636d0 00000000`00401aba     App1!_libc_start_main+0x3d1
04 00007ffd`f45637d0 ffffffff`ffffffff     App1!start+0x2a
05 00007ffd`f45637d8 00000000`00000000     0xffffffff`ffffffff

Unable to get thread data for thread 1
   1  Id: fd.fe Suspend: 0 Teb: 00000000`00000000 Unfrozen
 # Child-SP          RetAddr               Call Site
00 00007f0f`c16fad10 00000000`0044199a     App1!nanosleep+0x40
01 00007f0f`c16fad40 00000000`00401bbb     App1!sleep+0x3a
02 00007f0f`c16fad80 00000000`00401bcc     App1!bar_one+0xe
03 00007f0f`c16fad90 00000000`00401be5     App1!foo_one+0xe
04 00007f0f`c16fada0 00000000`004030d3     App1!thread_one+0x16
05 00007f0f`c16fadc0 00000000`0044426f     App1!start_thread+0xf3
06 00007f0f`c16fae80 ffffffff`ffffffff     App1!_clone+0x3f
07 00007f0f`c16fae88 00000000`00000000     0xffffffff`ffffffff

Unable to get thread data for thread 2
   2  Id: fd.ff Suspend: 0 Teb: 00000000`00000000 Unfrozen
 # Child-SP          RetAddr               Call Site
00 00007f0f`c0ef9d10 00000000`0044199a     App1!nanosleep+0x40
01 00007f0f`c0ef9d40 00000000`00401bfa     App1!sleep+0x3a
02 00007f0f`c0ef9d80 00000000`00401c0b     App1!bar_two+0xe
03 00007f0f`c0ef9d90 00000000`00401c24     App1!foo_two+0xe
```

```
04 00007f0f`c0ef9da0 00000000`004030d3     App1!thread_two+0x16
05 00007f0f`c0ef9dc0 00000000`0044426f     App1!start_thread+0xf3
06 00007f0f`c0ef9e80 ffffffff`ffffffff     App1!_clone+0x3f
07 00007f0f`c0ef9e88 00000000`00000000     0xffffffff`ffffffff

Unable to get thread data for thread 3
   3  Id: fd.100 Suspend: 0 Teb: 00000000`00000000 Unfrozen
 # Child-SP          RetAddr               Call Site
00 00007f0f`c06f8d10 00000000`0044199a     App1!nanosleep+0x40
01 00007f0f`c06f8d40 00000000`00401c39     App1!sleep+0x3a
02 00007f0f`c06f8d80 00000000`00401c4a     App1!bar_three+0xe
03 00007f0f`c06f8d90 00000000`00401c63     App1!foo_three+0xe
04 00007f0f`c06f8da0 00000000`004030d3     App1!thread_three+0x16
05 00007f0f`c06f8dc0 00000000`0044426f     App1!start_thread+0xf3
06 00007f0f`c06f8e80 ffffffff`ffffffff     App1!_clone+0x3f
07 00007f0f`c06f8e88 00000000`00000000     0xffffffff`ffffffff

Unable to get thread data for thread 4
   4  Id: fd.101 Suspend: 0 Teb: 00000000`00000000 Unfrozen
 # Child-SP          RetAddr               Call Site
00 00007f0f`bfef7d10 00000000`0044199a     App1!nanosleep+0x40
01 00007f0f`bfef7d40 00000000`00401c78     App1!sleep+0x3a
02 00007f0f`bfef7d80 00000000`00401c89     App1!bar_four+0xe
03 00007f0f`bfef7d90 00000000`00401ca2     App1!foo_four+0xe
04 00007f0f`bfef7da0 00000000`004030d3     App1!thread_four+0x16
05 00007f0f`bfef7dc0 00000000`0044426f     App1!start_thread+0xf3
06 00007f0f`bfef7e80 ffffffff`ffffffff     App1!_clone+0x3f
07 00007f0f`bfef7e88 00000000`00000000     0xffffffff`ffffffff

Unable to get thread data for thread 5
   5  Id: fd.102 Suspend: 0 Teb: 00000000`00000000 Unfrozen
 # Child-SP          RetAddr               Call Site
00 00007f0f`bf6f6d10 00000000`0044199a     App1!nanosleep+0x40
01 00007f0f`bf6f6d40 00000000`00401cb7     App1!sleep+0x3a
02 00007f0f`bf6f6d80 00000000`00401cc8     App1!bar_five+0xe
03 00007f0f`bf6f6d90 00000000`00401ce1     App1!foo_five+0xe
04 00007f0f`bf6f6da0 00000000`004030d3     App1!thread_five+0x16
05 00007f0f`bf6f6dc0 00000000`0044426f     App1!start_thread+0xf3
06 00007f0f`bf6f6e80 ffffffff`ffffffff     App1!_clone+0x3f
07 00007f0f`bf6f6e88 00000000`00000000     0xffffffff`ffffffff
```

8. Switch to thread #1 (threads are numbered from 0) and get its stack trace:

```
0:000> ~1s
App1!nanosleep+0x40:
00000000`00441a10 cmp     rax,0FFFFFFFFFFFFF000h

0:001> kL
# Child-SP          RetAddr               Call Site
00 00007f0f`c16fad10 00000000`0044199a     App1!nanosleep+0x40
01 00007f0f`c16fad40 00000000`00401bbb     App1!sleep+0x3a
02 00007f0f`c16fad80 00000000`00401bcc     App1!bar_one+0xe
03 00007f0f`c16fad90 00000000`00401be5     App1!foo_one+0xe
04 00007f0f`c16fada0 00000000`004030d3     App1!thread_one+0x16
05 00007f0f`c16fadc0 00000000`0044426f     App1!start_thread+0xf3
06 00007f0f`c16fae80 ffffffff`ffffffff     App1!_clone+0x3f
07 00007f0f`c16fae88 00000000`00000000     0xffffffff`ffffffff
```

9. Check that *bar_one* called the *sleep* function by comparing the return address on the call stack from the disassembly output:

```
0:001> uf bar_one
App1!bar_one:
00000000`00401bad push    rbp
00000000`00401bae mov     rbp,rsp
00000000`00401bb1 mov     edi,0FFFFFFFFh
00000000`00401bb6 call    App1!sleep (00000000`00441960)
00000000`00401bbb nop
00000000`00401bbc pop     rbp
00000000`00401bbd ret
```

Another way to do that is to disassemble backward the return address and check if the last instruction is CALL:

```
0:001> ub 00000000`00401bbb
App1!frame_dummy+0x1d:
00000000`00401b9d pop     rbp
00000000`00401b9e jmp     App1!register_tm_clones (00000000`00401b00)
00000000`00401ba3 nop     dword ptr [rax+rax]
00000000`00401ba8 jmp     App1!register_tm_clones (00000000`00401b00)
App1!bar_one:
00000000`00401bad push    rbp
00000000`00401bae mov     rbp,rsp
00000000`00401bb1 mov     edi,0FFFFFFFFh
00000000`00401bb6 call    App1!sleep (00000000`00441960)
```

10. Get *App1* data section from the contents of pmap (*App1.pmap.253*):

```
253:   ./App1
0000000000400000      4K r---- App1
0000000000401000    588K r-x-- App1
0000000000494000    156K r---- App1
00000000004bc000     24K rw--- App1
00000000004c2000     24K rw---   [ anon ]
00000000021b3000    140K rw---   [ anon ]
00007f0fbeef7000      4K -----   [ anon ]
00007f0fbeef8000   8192K rw---   [ anon ]
00007f0fbf6f8000      4K -----   [ anon ]
00007f0fbf6f9000   8192K rw---   [ anon ]
00007f0fbfef9000      4K -----   [ anon ]
00007f0fbfefa000   8192K rw---   [ anon ]
00007f0fc06fa000      4K -----   [ anon ]
00007f0fc06fb000   8192K rw---   [ anon ]
00007f0fc0efb000      4K -----   [ anon ]
00007f0fc0efc000   8192K rw---   [ anon ]
00007ffdf4545000    132K rw---   [ stack ]
00007ffdf45c6000     16K r----   [ anon ]
00007ffdf45ca000      4K r-x--   [ anon ]
 total            42068K
```

11. Compare with the region information in the core dump:

```
0:001> !address

Mapping file section regions...
Mapping module regions...
Mapping heap regions...

        BaseAddress      EndAddress+1        RegionSize     Type       State                 Protect             Usage
----------------------------------------------------------------------------------------------------------------------
+        0`00000000        0`00400000        0`00400000                                                       <unknown>
+        0`00400000        0`00401000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_EXECUTE_READ           Image      [App1; "/home/coredump/ALCDA/App1/App1"]
+        0`00401000        0`00494000        0`00093000 MEM_PRIVATE MEM_COMMIT  PAGE_EXECUTE_READ           Image      [App1; "/home/coredump/ALCDA/App1/App1"]
+        0`00494000        0`004bc000        0`00028000                                                       Image      [App1; "/home/coredump/ALCDA/App1/App1"]
+        0`004bc000        0`004c2000        0`00006000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE              Image      [App1; "/home/coredump/ALCDA/App1/App1"]
+        0`004c2000        0`004c8000        0`00006000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE              <unknown>  [................]
+        0`004c8000        0`021b3000        0`01ceb000                                                       <unknown>
+        0`021b3000        0`021d6000        0`00023000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE              <unknown>  [................]
```

```
+        0`021d6000     7f0f`beef7000     7f0f`bcd21000                                                    <unknown>
+     7f0f`beef7000     7f0f`beef8000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY              <unknown>  [................]
+     7f0f`beef8000     7f0f`bf6f8000        0`00800000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE             <unknown>  [................]
+     7f0f`bf6f8000     7f0f`bf6f9000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY              <unknown>  [................]
+     7f0f`bf6f9000     7f0f`bfef9000        0`00800000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE             <unknown>  [................]
+     7f0f`bfef9000     7f0f`bfefa000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY              <unknown>  [................]
+     7f0f`bfefa000     7f0f`c06fa000        0`00800000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE             <unknown>  [................]
+     7f0f`c06fa000     7f0f`c06fb000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY              <unknown>  [................]
+     7f0f`c06fb000     7f0f`c0efb000        0`00800000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE             <unknown>  [................]
+     7f0f`c0efb000     7f0f`c0efc000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY              <unknown>  [................]
+     7f0f`c0efc000     7f0f`c16fc000        0`00800000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE             <unknown>  [................]
+     7f0f`c16fc000     7ffd`f4545000      ee`32e49000                                                    <unknown>
+     7ffd`f4545000     7ffd`f4566000        0`00021000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE             <unknown>  [................]
+     7ffd`f4566000     7ffd`f45ca000        0`00064000                                                    <unknown>
+     7ffd`f45ca000     7ffd`f45cb000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_EXECUTE_READ          Image      [linux_vdso_so; "linux-vdso.so.1"]
```

12. Dump the data region with possible symbolic information (we truncated the output):

```
0:001> dps 0`004bc000 0`004c2000
[...]
00000000`004bea88  00000000`00000000
00000000`004bea90  00000000`00000000
00000000`004bea98  00000000`00000000
00000000`004beaa0  00007ffd`f4563a09
00000000`004beaa8  00000000`004c01c8 App1!_progname
00000000`004beab0  00007ffd`f45637d8
00000000`004beab8  00000000`00000000
[...]
00000000`004bf868  00010000`00000000
00000000`004bf870  00000000`00000000
00000000`004bf878  00000000`00000000
00000000`004bf880  00000000`00000000
00000000`004bf888  00000000`021b41c0
00000000`004bf890  00000000`00000040
00000000`004bf898  00000000`00000408
00000000`004bf8a0  00000000`00000007
00000000`004bf8a8  00000000`00000000
00000000`004bf8b0  00000000`00000000
00000000`004bf8b8  00000000`00000000
00000000`004bf8c0  00000000`0041aad0 App1!memalign_hook_ini
00000000`004bf8c8  00000000`0041b0e0 App1!realloc_hook_ini
00000000`004bf8d0  00000000`00000000
00000000`004bf8d8  00000000`00000000
00000000`004bf8e0  00000000`00000000
00000000`004bf8e8  00000000`00000000
00000000`004bf8f0  00000000`00000000
00000000`004bf8f8  00000000`00000000
00000000`004bf900  00000000`00000000
00000000`004bf908  00000000`00000000
[...]
```

The output is in the following format:

```
address  value
```

Some values may have associated symbols in the format module!name+offset:

```
address  value  symbol
```

For example, from the output above:

```
00000000`004d1110  00000000`004d1068 App1!_progname
```

To list all values with symbols, we can use the **dpS** command (it doesn't show the value addresses):

```
0:001> dpS 0`004bc000 0`004c2000
 00000000`004c65a0 App1!res
 00000000`004c0aa0 App1!nl_global_locale
 00000000`004c0aa0 App1!nl_global_locale
 00000000`004c0ac0 App1!nl_global_locale+0x20
 00000000`004c0aa8 App1!nl_global_locale+0x8
[...]
 00007ffd`f45ca168 linux_vdso_so!LINUX_2.6+0x168
 00000000`004c01c8 App1!_progname
 00000000`00498f28 App1!_PRETTY_FUNCTION__.9662+0x58
[...]
 00000000`0044fbc0 App1!_gconv_transform_internal_ucs2reverse
 00000000`004530c0 App1!nl_postload_ctype
 00000000`004bc1c0 App1!nl_C_LC_CTYPE
 00000000`004bd5c0 App1!nl_C_LC_NUMERIC
 00000000`004bd640 App1!nl_C_LC_TIME
 00000000`004bdec0 App1!nl_C_LC_COLLATE
 00000000`004bd400 App1!nl_C_LC_MONETARY
 00000000`004bd380 App1!nl_C_LC_MESSAGES
 00000000`004bdb80 App1!nl_C_LC_PAPER
 00000000`004bdbe0 App1!nl_C_LC_NAME
 00000000`004bdc60 App1!nl_C_LC_ADDRESS
 00000000`004bdd20 App1!nl_C_LC_TELEPHONE
 00000000`004bdda0 App1!nl_C_LC_MEASUREMENT
 00000000`004bde00 App1!nl_C_LC_IDENTIFICATION
 00000000`004bc1c0 App1!nl_C_LC_CTYPE
 00000000`004bd5c0 App1!nl_C_LC_NUMERIC
 00000000`004bd640 App1!nl_C_LC_TIME
 00000000`004bdec0 App1!nl_C_LC_COLLATE
 00000000`004bd400 App1!nl_C_LC_MONETARY
 00000000`004bd380 App1!nl_C_LC_MESSAGES
 00000000`004bdb80 App1!nl_C_LC_PAPER
 00000000`004bdbe0 App1!nl_C_LC_NAME
 00000000`004bdc60 App1!nl_C_LC_ADDRESS
 00000000`004bdd20 App1!nl_C_LC_TELEPHONE
 00000000`004bdda0 App1!nl_C_LC_MEASUREMENT
 00000000`004bde00 App1!nl_C_LC_IDENTIFICATION
 00000000`0049bc00 App1!nl_C_LC_CTYPE_class+0x100
 00000000`0049ad00 App1!nl_C_LC_CTYPE_tolower+0x200
 00000000`0049b300 App1!nl_C_LC_CTYPE_toupper+0x200
 00000000`00498c88 App1!nl_C_name
 00000000`00498c88 App1!nl_C_name
[...]
 00000000`004c0e20 App1!IO_wfile_jumps
 00000000`004bf1a0 App1!IO_2_1_stderr_
 00000000`004bf3c0 App1!IO_2_1_stdout_
 00000000`004bf5e0 App1!IO_2_1_stdin_
 00000000`0041aad0 App1!memalign_hook_ini
 00000000`0041b0e0 App1!realloc_hook_ini
 00000000`004bf940 App1!main_arena+0x60
 00000000`004bf940 App1!main_arena+0x60
 00000000`004bf950 App1!main_arena+0x70
 00000000`004bf950 App1!main_arena+0x70
[...]
 00000000`00414ef0 App1!IO_default_imbue
 00000000`004145a0 App1!IO_cleanup
 00000000`004ae798 App1!_EH_FRAME_BEGIN__
```

13. Explore the contents of memory pointed to by *App1!memalign_hook_ini* and *App1!_progname* addresses:

```
0:001> u 00000000`0041aad0
App1!memalign_hook_ini:
00000000`0041aad0 sub     rsp,18h
00000000`0041aad4 mov     eax,dword ptr [App1!_libc_malloc_initialized (00000000`004bf824)]
00000000`0041aada mov     qword ptr [App1!_memalign_hook (00000000`004bf8c0)],0
00000000`0041aae5 test    eax,eax
00000000`0041aae7 jns     App1!memalign_hook_ini+0x30 (00000000`0041ab00)
00000000`0041aae9 mov     qword ptr [rsp+8],rsi
00000000`0041aaee mov     qword ptr [rsp],rdi
00000000`0041aaf2 call    App1!ptmalloc_init.part.0 (00000000`00416570)

0:001> dp App1!_progname
00000000`004c01c8  00007ffd`f4565751 00007ffd`f456574f
00000000`004c01d8  00000000`00000000 00000000`00000000
00000000`004c01e8  00000000`00000000 00000000`00000001
00000000`004c01f8  00000000`00000000 00000000`00000000
00000000`004c0208  00000000`00000000 00000000`00000000
00000000`004c0218  00000000`00000000 00000000`00000000
00000000`004c0228  00000000`00000000 00000000`00000001
00000000`004c0238  00000000`00000000 00000000`00000000

0:001> dc 00007ffd`f4565751
00007ffd`f4565751  31707041 45485300 2f3d4c4c 2f6e6962  App1.SHELL=/bin/
00007ffd`f4565761  68736162 53494800 4e4f4354 4c4f5254  bash.HISTCONTROL
00007ffd`f4565771  6e67693d 6265726f 0068746f 5f4c5357  =ignoreboth.WSL_
00007ffd`f4565781  54534944 4e5f4f52 3d454d41 69626544  DISTRO_NAME=Debi
00007ffd`f4565791  4e006e61 3d454d41 4b534544 2d504f54  an.NAME=DESKTOP-
00007ffd`f45657a1  56365349 00304c32 3d445750 6d6f682f  IS6V2L0.PWD=/hom
00007ffd`f45657b1  6f632f65 75646572 412f706d 4144434c  e/coredump/ALCDA
00007ffd`f45657c1  7070412f 4f4c0031 4d414e47 6f633d45  /App1.LOGNAME=co

0:001> da 00007ffd`f4565751
00007ffd`f4565751  "App1"

0:001> db 00007ffd`f4565751
00007ffd`f4565751  41 70 70 31 00 53 48 45-4c 4c 3d 2f 62 69 6e 2f  App1.SHELL=/bin/
00007ffd`f4565761  62 61 73 68 00 48 49 53-54 43 4f 4e 54 52 4f 4c  bash.HISTCONTROL
00007ffd`f4565771  3d 69 67 6e 6f 72 65 62-6f 74 68 00 57 53 4c 5f  =ignoreboth.WSL_
00007ffd`f4565781  44 49 53 54 52 4f 5f 4e-41 4d 45 3d 44 65 62 69  DISTRO_NAME=Debi
00007ffd`f4565791  61 6e 00 4e 41 4d 45 3d-44 45 53 4b 54 4f 50 2d  an.NAME=DESKTOP-
00007ffd`f45657a1  49 53 36 56 32 4c 30 00-50 57 44 3d 2f 68 6f 6d  IS6V2L0.PWD=/hom
00007ffd`f45657b1  65 2f 63 6f 72 65 64 75-6d 70 2f 41 4c 43 44 41  e/coredump/ALCDA
00007ffd`f45657c1  2f 41 70 70 31 00 4c 4f-47 4e 41 4d 45 3d 63 6f  /App1.LOGNAME=co
```

Note: We see that a hook function is installed for *memalign* and *realloc*. Please find the following documentation for hook functions here:

https://www.gnu.org/software/libc/manual/html_node/Hooks-for-Malloc.html

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

```
0:001> dp environ
00000000`004c5f48  00007ffd`f45637f8 00000000`00000000
00000000`004c5f58  00000000`00000000 00000001`00000000
00000000`004c5f68  00000000`00000000 00000000`00000000
00000000`004c5f78  00000000`00000000 00000000`00000001
00000000`004c5f88  00000000`00000030 00000000`004c5f98
```

```
00000000`004c5f98  00000000`00000000 00000000`00000000
00000000`004c5fa8  00000000`00402590 00000000`00000000
00000000`004c5fb8  00000000`00000000 00000000`00000000

0:001> dp 00007ffd`f45637f8
00007ffd`f45637f8  00007ffd`f4565756 00007ffd`f4565766
00007ffd`f4563808  00007ffd`f456577d 00007ffd`f4565794
00007ffd`f4563818  00007ffd`f45657a9 00007ffd`f45657c7
00007ffd`f4563828  00007ffd`f45657d8 00007ffd`f45657f3
00007ffd`f4563838  00007ffd`f45657fe 00007ffd`f4565812
00007ffd`f4563848  00007ffd`f4565823 00007ffd`f4565844
00007ffd`f4563858  00007ffd`f4565e26 00007ffd`f4565e40
00007ffd`f4563868  00007ffd`f4565e54 00007ffd`f4565e62

0:001> da 00007ffd`f4565756
00007ffd`f4565756  "SHELL=/bin/bash"

0:001> dpa 00007ffd`f45637f8
00007ffd`f45637f8  00007ffd`f4565756 "SHELL=/bin/bash"
00007ffd`f4563800  00007ffd`f4565766 "HISTCONTROL=ignoreboth"
00007ffd`f4563808  00007ffd`f456577d "WSL_DISTRO_NAME=Debian"
00007ffd`f4563810  00007ffd`f4565794 "NAME=DESKTOP-IS6V2L0"
00007ffd`f4563818  00007ffd`f45657a9 "PWD=/home/coredump/ALCDA/App1"
00007ffd`f4563820  00007ffd`f45657c7 "LOGNAME=coredump"
00007ffd`f4563828  00007ffd`f45657d8 "MC_TMPDIR=/tmp/mc-coredump"
00007ffd`f4563830  00007ffd`f45657f3 "MC_SID=192"
00007ffd`f4563838  00007ffd`f45657fe "HOME=/home/coredump"
00007ffd`f4563840  00007ffd`f4565812 "LANG=en_US.UTF-8"
00007ffd`f4563848  00007ffd`f4565823 "WSL_INTEROP=/run/WSL/117_interop"
00007ffd`f4563850  00007ffd`f4565844 "LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;3"
00007ffd`f4563858  00007ffd`f4565e26 "WAYLAND_DISPLAY=wayland-0"
00007ffd`f4563860  00007ffd`f4565e40 "TERM=xterm-256color"
00007ffd`f4563868  00007ffd`f4565e54 "USER=coredump"
00007ffd`f4563870  00007ffd`f4565e62 "DISPLAY=:0"
```

18. Now we look at how to perform a memory search.

```
0:001> s 0`004bc000 0`004fc000 6
00000000`004bcc31  06 46 00 00 00 00 00 56-06 46 00 00 00 00 00 4b  .F.....V.F.....K
00000000`004bcc39  06 46 00 00 00 00 00 4b-06 46 00 00 00 00 00 40  .F.....K.F.....@
00000000`004bcc41  06 46 00 00 00 00 00 40-06 46 00 00 00 00 00 f8  .F.....@.F......
00000000`004bcc49  06 46 00 00 00 00 00 f8-03 46 00 00 00 00 00 a8  .F.......F......
00000000`004bd5f8  06 00 00 00 00 00 00 00-d9 7d 49 00 00 00 00 00  .........}I.....
00000000`004bde40  06 86 4a 00 00 00 00 00-70 86 4a 00 00 00 00 00  ..J.....p.J.....
00000000`004be880  06 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
00000000`004bea40  06 00 00 00 00 00 00 00-b8 a1 5c f4 fd 7f 00 00  ..........\.....
00000000`004beb5b  06 00 00 00 00 96 8f 49-00 00 00 00 00 c0 da 44  .......I.......D
00000000`004beb79  06 04 04 00 00 00 00 b6-8f 49 00 00 00 00 00 f0  .........I......
00000000`004bf110  06 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
00000000`004c0248  06 00 00 00 7f 03 00 00-03 03 00 00 02 00 00 00  ................
00000000`004c02d1  06 4c 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .L..............
00000000`004c06c1  06 4c 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .L..............
00000000`004c06e9  06 4c 00 00 00 00 00 20-06 4c 00 00 00 00 00 00  .L..... .L......
00000000`004c06f1  06 4c 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .L..............
00000000`004c6829  06 08 00 00 08 08 05 23-32 da fe ff fb 8b 1f 00  .......#2.......
00000000`004c6858  06 00 00 00 8e 00 00 00-80 03 00 00 00 00 00 00  ................
```

Note: It is possible to search through non-accessible regions as well; they are ignored:

```
0:001> s-q 0 Lffffff 6
00000000`0049a308  00000000`00000006 00000000`0000009f
00000000`0049a7c8  00000000`00000006 00000018`00000001
00000000`0049a8e8  00000000`00000006 00000018`00000001
00000000`0049a9a8  00000000`00000006 00000018`00000001
00000000`004a8d08  00000000`00000006 00000000`00000002
00000000`004b17e8  00000000`00000006 00003088`00000010
00000000`004bd5f8  00000000`00000006 00000000`00497dd9
00000000`004be880  00000000`00000006 00000000`00000000
00000000`004bea40  00000000`00000006 00007ffd`f45ca1b8
00000000`004bf110  00000000`00000006 00000000`00000000

0:001> s-a 00007ffd`f45637f8 L100000 "bin"
00007ffd`f456575d  62 69 6e 2f 62 61 73 68-00 48 49 53 54 43 4f 4e  bin/bash.HISTCON
00007ffd`f4565e9d  62 69 6e 2d 68 61 64 6f-6f 70 32 2e 37 00 58 44  bin-hadoop2.7.XD
00007ffd`f4565ef4  62 69 6e 3a 2f 75 73 72-2f 6c 6f 63 61 6c 2f 62  bin:/usr/local/b
00007ffd`f4565f03  62 69 6e 3a 2f 75 73 72-2f 62 69 6e 3a 2f 62 69  bin:/usr/bin:/bi
00007ffd`f4565f0c  62 69 6e 3a 2f 62 69 6e-3a 2f 75 73 72 2f 6c 6f  bin:/bin:/usr/lo
00007ffd`f4565f11  62 69 6e 3a 2f 75 73 72-2f 6c 6f 63 61 6c 2f 67  bin:/usr/local/g
00007ffd`f4565f4e  62 69 6e 2d 68 61 64 6f-6f 70 32 2e 37 2f 62 69  bin-hadoop2.7/bi
00007ffd`f4565f5c  62 69 6e 3a 2f 75 73 72-2f 73 68 61 72 65 2f 73  bin:/usr/share/s
00007ffd`f4565f7d  62 69 6e 2d 68 61 64 6f-6f 70 32 2e 37 2f 62 69  bin-hadoop2.7/bi
00007ffd`f4565f8b  62 69 6e 00 48 4f 53 54-54 59 50 45 3d 78 38 36  bin.HOSTTYPE=x86
```

Note: It is also possible to show all possible string fragments if any:

```
0:001> s-sa 0 Lfffffff
00000000`00400001  "ELF"
00000000`0040020c  "GNU"
00000000`0040022c  "GNU"
00000000`00400236  "Y_I"
00000000`004011f4  "uAH"
00000000`0040121f  "=,H"
[...]
00000000`0049750f  "../sysdeps/x86/cacheinfo.c"
00000000`0049752a  "! "cannot happen""
00000000`0049753c  "offset == 2"
00000000`00497840  "handle_amd"
00000000`00497850  "intel_check_word"
00000000`00497861  "ANSI_X3.4-1968//TRANSLIT"
00000000`0049787a  "mbsrtowcs_l.c"
00000000`00497888  "result > 0"
00000000`00497893  "__mbsinit (data.__statep)"
00000000`004978b0  "((wchar_t *) data.__outbuf)[-1] "
00000000`004978d0  "== L'\0'"
00000000`004978e0  "status == __GCONV_OK || status ="
00000000`00497900  "= __GCONV_EMPTY_INPUT || status "
00000000`00497920  "== __GCONV_ILLEGAL_INPUT || stat"
00000000`00497940  "us == __GCONV_INCOMPLETE_INPUT |"
00000000`00497960  "| status == __GCONV_FULL_OUTPUT"
00000000`00497980  "__mbsrtowcs_l"
00000000`0049798e  "/usr/lib/getconf"
00000000`0049799f  "GETCONF_DIR"
00000000`004979ab  "/proc/sys/kernel/ngroups_max"
00000000`004979c8  "ILP32_OFF32"
00000000`004979d4  "ILP32_OFFBIG"
00000000`004979e1  "/proc/sys/kernel/rtsig-max"
[...]
```

```
00000000`021b4420  "/home/coredump/ALCDA/App1"
00000000`021b4900  "linux-vdso.so.1"
00000000`021b49a0  "tls//x86_64/tls/x86_64/"
00000000`021b4a08  " uJ"
00000000`021b4a58  "7uJ"
00000000`021b4aa8  "RuJ"
00000000`021b4af8  "XuJ"
```

15. To get process uid/gid and other useful data use this WinDbg extension command:

```
0:000> !ntprpsinfo
NT_PRPSINFO (process info):
    state: 0, sname: t, zomb: 0, nice: 0, flag: 0x4040400000000000
    uid: 1000, gid: 1000, pid: 253, ppid: 192, pgrp: 253, sid: 192
    fname: App1
    psargs: ./App1
```

16. Get the list of loaded modules:

```
0:001> lm
start             end                 module name
00000000`00400000 00000000`004c2000   App1     T (service symbols: DWARF Private Symbols)
c:\alcda2\x64\app1\App1
00007ffd`f45ca000 00007ffd`f45cb000   linux_vdso_so T (service symbols: ELF In Memory Symbols)

0:001> lmv
start             end                 module name
00000000`00400000 00000000`004c2000   App1     T (service symbols: DWARF Private Symbols)
c:\alcda2\x64\app1\App1
    Loaded symbol image file: App1
    Image path: /home/coredump/ALCDA/App1/App1
    Image name: App1
    Browse all global symbols  functions  data  Symbol Reload
    Timestamp:        unavailable (FFFFFFFE)
    CheckSum:         missing
    ImageSize:        000C2000
    Details:
00007ffd`f45ca000 00007ffd`f45cb000   linux_vdso_so T (service symbols: ELF In Memory Symbols)
    Loaded symbol image file: linux-vdso.so.1
    Image path: linux-vdso.so.1
    Image name: linux-vdso.so.1
    Browse all global symbols  functions  data  Symbol Reload
    Timestamp:        unavailable (FFFFFFFE)
    CheckSum:         missing
    ImageSize:        00001000
    Index Key:        (Build Id) 8dd3bd6bb4492768a24ad26802eb38105a15ffe7
    Details:
```

Note: We don't see shared libraries except *vdso* (https://man7.org/linux/man-pages/man7/vdso.7.html) because they were statically linked. We also created the version of a dynamically linked *App1.shared* executable. If we load its core dump *App1.shared.core.275* in the new instance of WinDbg, we see the list of shared libraries:

```
Microsoft (R) Windows Debugger Version 10.0.27725.1000 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.


Loading Dump File [C:\ALCDA2\x64\App1\App1.shared.core.275]
64-bit machine not using 64-bit API
```

```
************* Path validation summary **************
Response                         Time (ms)     Location
Deferred                                       srv*
Symbol search path is: srv*
Executable search path is:
Generic Unix Version 0 UP Free x64
System Uptime: not available
Process Uptime: not available
.....
*** WARNING: Unable to verify timestamp for libc-2.28.so
libc_2_28+0xc5720:
00007f1a`e471e720 ???

0:000> .logappend C:\ALCDA2\x64\App1\App1-WinDbg.log
Opened log file 'C:\ALCDA2\x64\App1\App1-WinDbg.log'

0:000> lm
start             end                 module name
0000557e`17348000 0000557e`1734c000   App1       (deferred)
00007f1a`e4659000 00007f1a`e4815000   libc_2_28 T (no symbols)
00007f1a`e481a000 00007f1a`e4836000   libpthread_2_28   (deferred)
00007f1a`e4847000 00007f1a`e4870000   ld_2_28    (deferred)
00007ffc`749b0000 00007ffc`749b1000   linux_vdso_so   (deferred)
```

17. Let's set symbols to get the correct stack trace:

```
0:000> k
 # Child-SP          RetAddr               Call Site
00 00007ffc`74957a60 00000000`00000000     libc_2_28+0xc5720

0:000> uf bar_one
Couldn't resolve error at 'bar_one'

0:000> .sympath+ C:\ALCDA2\x64\App1
Symbol search path is: srv*;C:\ALCDA2\x64\App1
Expanded Symbol search path is:
cache*;SRV*https://msdl.microsoft.com/download/symbols;c:\alcda2\x64\app1

************* Path validation summary **************
Response                         Time (ms)     Location
Deferred                                       srv*
OK                                             C:\ALCDA2\x64\App1
*** WARNING: Unable to verify timestamp for libc-2.28.so

0:000> .reload
.....
*** WARNING: Unable to verify timestamp for libc-2.28.so

************* Symbol Loading Error Summary **************
Module name            Error
libc-2.28              The system cannot find the file specified

You can troubleshoot most symbol related issues by turning on symbol loading diagnostics (!sym
noisy) and repeating the command that caused symbols to be loaded.
You should also verify that your symbol search path (.sympath) is correct.
```

```
0:000> .symopt+ 0x40
Symbol options are 0x30377:
  0x00000001 - SYMOPT_CASE_INSENSITIVE
  0x00000002 - SYMOPT_UNDNAME
  0x00000004 - SYMOPT_DEFERRED_LOADS
  0x00000010 - SYMOPT_LOAD_LINES
  0x00000020 - SYMOPT_OMAP_FIND_NEAREST
  0x00000040 - SYMOPT_LOAD_ANYTHING
  0x00000100 - SYMOPT_NO_UNQUALIFIED_LOADS
  0x00000200 - SYMOPT_FAIL_CRITICAL_ERRORS
  0x00010000 - SYMOPT_AUTO_PUBLICS
  0x00020000 - SYMOPT_NO_IMAGE_SEARCH
*** WARNING: Unable to verify timestamp for libc-2.28.so

0:000> k
 # Child-SP          RetAddr               Call Site
00 00007ffc`74957a60 00000000`00000000     libc_2_28+0xc5720

0:000> uf bar_one
Couldn't resolve error at 'bar_one'
```

Note: We see WinDbg is not able to get the symbols even for the App1 module, probably due to the way the core dump was saved. We created another core dump, this time saved with the 0x3F option:

```
~/ALCDA2/x64/App1$ ./App1.shared &
[1] 28

~/ALCDA2/x64/App1$ echo 0x3F > /proc/28/coredump_filter

~/ALCDA2/x64/App1$ gcore -o App1.shared.core 28
```

Now, we load the core dump *App1.shared.core.28* in the new instance of WinDbg:

```
Microsoft (R) Windows Debugger Version 10.0.27725.1000 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.


Loading Dump File [C:\ALCDA2\x64\App1\App1.shared.core.28]
64-bit machine not using 64-bit API

************* Path validation summary **************
Response                         Time (ms)     Location
Deferred                                       srv*
Symbol search path is: srv*
Executable search path is:
Generic Unix Version 0 UP Free x64
System Uptime: not available
Process Uptime: not available
.....
*** WARNING: Unable to verify timestamp for libc-2.28.so
*** WARNING: Unable to verify timestamp for App1.shared
libc_2_28!nanosleep+0x40:
00007ff6`429dc5c0 cmp     rax,0FFFFFFFFFFFFF000h

0:000> .logappend C:\ALCDA2\x64\App1\App1-WinDbg.log
Opened log file 'C:\ALCDA2\x64\App1\App1-WinDbg.log'
```

```
0:000> lm
start             end                 module name
000055b1`b4e04000 000055b1`b4e09000   App1     T (service symbols: ELF In Memory Symbols)
00007ff6`42916000 00007ff6`42ad2000   libc_2_28 T (service symbols: ELF In Memory Symbols)
00007ff6`42ad6000 00007ff6`42af3000   libpthread_2_28   (deferred)
00007ff6`42b04000 00007ff6`42b2d000   ld_2_28    (deferred)
00007fff`cc76d000 00007fff`cc76e000   linux_vdso_so   (deferred)

0:000> k
 # Child-SP          RetAddr               Call Site
00 00007fff`cc685f70 00007ff6`429dc4ca     libc_2_28!nanosleep+0x40
01 00007fff`cc685fa0 000055b1`b4e0532a     libc_2_28!sleep+0x3a
02 00007fff`cc685fe0 00007ff6`4293a09b     App1!pthread_create+0x132a
03 00007fff`cc686030 000055b1`b4e0508a     libc_2_28!_libc_start_main+0xeb
04 00007fff`cc6860f0 ffffffff`ffffffff     App1!pthread_create+0x108a
05 00007fff`cc6860f8 00000000`00000000     0xffffffff`ffffffff

0:000> uf bar_one
Couldn't resolve error at 'bar_one'
```

Note: Although the stack trace looks better, we need to specify the App1 symbols:

```
0:000> .sympath+ C:\ALCDA2\x64\App1
*** WARNING: Unable to verify timestamp for libc-2.28.so
Symbol search path is: srv*;C:\ALCDA2\x64\App1
Expanded Symbol search path is:
cache*;SRV*https://msdl.microsoft.com/download/symbols;c:\alcda2\x64\app1

************* Path validation summary **************
Response                         Time (ms)     Location
Deferred                                       srv*
OK                                             C:\ALCDA2\x64\App1
*** WARNING: Unable to verify timestamp for App1.shared

0:000> .reload
...*** WARNING: Unable to verify timestamp for libc-2.28.so
..
*** WARNING: Unable to verify timestamp for App1.shared

************* Symbol Loading Error Summary **************
Module name            Error
App1                   The system cannot find the file specified
libc-2.28              The system cannot find the file specified

You can troubleshoot most symbol related issues by turning on symbol loading diagnostics (!sym
noisy) and repeating the command that caused symbols to be loaded.
You should also verify that your symbol search path (.sympath) is correct.

0:000> k
 # Child-SP          RetAddr               Call Site
00 00007fff`cc685f70 00007ff6`429dc4ca     libc_2_28!nanosleep+0x40
01 00007fff`cc685fa0 000055b1`b4e0532a     libc_2_28!sleep+0x3a
02 00007fff`cc685fe0 00007ff6`4293a09b     App1!main+0xaa
03 00007fff`cc686030 000055b1`b4e0508a     libc_2_28!_libc_start_main+0xeb
04 00007fff`cc6860f0 ffffffff`ffffffff     App1!start+0x2a
05 00007fff`cc6860f8 00000000`00000000     0xffffffff`ffffffff
```

```
0:000> uf bar_one
App1!bar_one:
000055b1`b4e05145 push    rbp
000055b1`b4e05146 mov     rbp,rsp
000055b1`b4e05149 mov     edi,0FFFFFFFFh
000055b1`b4e0514e call    App1!sleep$plt (000055b1`b4e05040)
000055b1`b4e05153 nop
000055b1`b4e05154 pop     rbp
000055b1`b4e05155 ret

0:000> u 000055b1`b4e05040
App1!sleep$plt:
000055b1`b4e05040 jmp     qword ptr [App1!GLOBAL_OFFSET_TABLE_+0x20 (000055b1`b4e08020)]
000055b1`b4e05046 push    1
000055b1`b4e0504b jmp     App1+0x1020 (000055b1`b4e05020)
App1!:
000055b1`b4e05050 jmp     qword ptr [App1!+0x20 (000055b1`b4e07ff8)]
000055b1`b4e05056 nop
000055b1`b4e05058 add     byte ptr [rax],al
000055b1`b4e0505a add     byte ptr [rax],al
000055b1`b4e0505c add     byte ptr [rax],al

0:000> dps 000055b1`b4e08020 L1
000055b1`b4e08020  00007ff6`429dc490 libc_2_28!sleep
```

18. *App1.shared.pmap.28* also shows library memory regions:

```
28:   ./App1.shared
000055b1b4e04000      4K r---- App1.shared
000055b1b4e05000      4K r-x-- App1.shared
000055b1b4e06000      4K r---- App1.shared
000055b1b4e07000      4K r---- App1.shared
000055b1b4e08000      4K rw--- App1.shared
000055b1b92e4000    132K rw---   [ anon ]
00007ff64010e000      4K -----   [ anon ]
00007ff64010f000   8192K rw---   [ anon ]
00007ff64090f000      4K -----   [ anon ]
00007ff640910000   8192K rw---   [ anon ]
00007ff641110000      4K -----   [ anon ]
00007ff641111000   8192K rw---   [ anon ]
00007ff641911000      4K -----   [ anon ]
00007ff641912000   8192K rw---   [ anon ]
00007ff642112000      4K -----   [ anon ]
00007ff642113000   8204K rw---   [ anon ]
00007ff642916000    136K r---- libc-2.28.so
00007ff642938000   1308K r-x-- libc-2.28.so
00007ff642a7f000    304K r---- libc-2.28.so
00007ff642acb000      4K ----- libc-2.28.so
00007ff642acc000     16K r---- libc-2.28.so
00007ff642ad0000      8K rw--- libc-2.28.so
00007ff642ad2000     16K rw---   [ anon ]
00007ff642ad6000     24K r---- libpthread-2.28.so
00007ff642adc000     60K r-x-- libpthread-2.28.so
00007ff642aeb000     24K r---- libpthread-2.28.so
00007ff642af1000      4K r---- libpthread-2.28.so
00007ff642af2000      4K rw--- libpthread-2.28.so
00007ff642af3000     24K rw---   [ anon ]
00007ff642b04000      4K r---- ld-2.28.so
00007ff642b05000    120K r-x-- ld-2.28.so
00007ff642b23000     32K r---- ld-2.28.so
```

```
00007ff642b2b000      4K r---- ld-2.28.so
00007ff642b2c000      4K rw--- ld-2.28.so
00007ff642b2d000      4K rw---   [ anon ]
00007fffcc667000    132K rw---   [ stack ]
00007fffcc769000     16K r----   [ anon ]
00007fffcc76d000      8K r-x--   [ anon ]
 total            43400K
```

Note: We can also see shared library mappings in the output of the **!address** command:

```
0:000> !address

Mapping file section regions...
Mapping module regions...
Mapping heap regions...

        BaseAddress      EndAddress+1        RegionSize     Type       State                 Protect             Usage
-------------------------------------------------------------------------------------------------------------------
+        0`00000000     55b1`b4e04000     55b1`b4e04000                                                       <unknown>
+     55b1`b4e04000     55b1`b4e05000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY                 Image      [App1; "/home/coredump/ALCDA2/x64/App1/App1.shared"]
+     55b1`b4e05000     55b1`b4e06000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_EXECUTE_READ            Image      [App1; "/home/coredump/ALCDA2/x64/App1/App1.shared"]
+     55b1`b4e06000     55b1`b4e07000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY                 Image      [App1; "/home/coredump/ALCDA2/x64/App1/App1.shared"]
+     55b1`b4e07000     55b1`b4e08000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY                 Image      [App1; "/home/coredump/ALCDA2/x64/App1/App1.shared"]
+     55b1`b4e08000     55b1`b4e09000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE                Image      [App1; "/home/coredump/ALCDA2/x64/App1/App1.shared"]
+     55b1`b4e09000     55b1`b92e4000        0`044db000                                                       <unknown>
+     55b1`b92e4000     55b1`b9305000        0`00021000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE                <unknown>  [........Q.......]
+     55b1`b9305000     7ff6`4010e000     2a44`86e09000                                                       <unknown>
+     7ff6`4010e000     7ff6`4010f000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY                 <unknown>  [................]
+     7ff6`4010f000     7ff6`4090f000        0`00800000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE                <unknown>  [................]
+     7ff6`4090f000     7ff6`40910000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY                 <unknown>  [................]
+     7ff6`40910000     7ff6`41110000        0`00800000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE                <unknown>  [................]
+     7ff6`41110000     7ff6`41111000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY                 <unknown>  [................]
+     7ff6`41111000     7ff6`41911000        0`00800000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE                <unknown>  [................]
+     7ff6`41911000     7ff6`41912000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY                 <unknown>  [................]
+     7ff6`41912000     7ff6`42112000        0`00800000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE                <unknown>  [................]
+     7ff6`42112000     7ff6`42113000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY                 <unknown>  [................]
+     7ff6`42113000     7ff6`42916000        0`00803000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE                <unknown>  [................]
+     7ff6`42916000     7ff6`42938000        0`00022000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY                 Image      [libc_2_28; "/lib/x86_64-linux-gnu/libc-2.28.so"]
+     7ff6`42938000     7ff6`42a7f000        0`00147000 MEM_PRIVATE MEM_COMMIT  PAGE_EXECUTE_READ            Image      [libc_2_28; "/lib/x86_64-linux-gnu/libc-2.28.so"]
+     7ff6`42a7f000     7ff6`42acb000        0`0004c000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY                 Image      [libc_2_28; "/lib/x86_64-linux-gnu/libc-2.28.so"]
+     7ff6`42acb000     7ff6`42acc000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY                 Image      [libc_2_28; "/lib/x86_64-linux-gnu/libc-2.28.so"]
+     7ff6`42acc000     7ff6`42ad0000        0`00004000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY                 Image      [libc_2_28; "/lib/x86_64-linux-gnu/libc-2.28.so"]
+     7ff6`42ad0000     7ff6`42ad2000        0`00002000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE                Image      [libc_2_28; "/lib/x86_64-linux-gnu/libc-2.28.so"]
+     7ff6`42ad2000     7ff6`42ad6000        0`00004000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE                <unknown>  [................]
+     7ff6`42ad6000     7ff6`42adc000        0`00006000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY                 Image      [libpthread_2_28; "/lib/x86_64-linux-gnu/libpthread-2.28.so"]
+     7ff6`42adc000     7ff6`42aeb000        0`0000f000 MEM_PRIVATE MEM_COMMIT  PAGE_EXECUTE_READ            Image      [libpthread_2_28; "/lib/x86_64-linux-gnu/libpthread-2.28.so"]
+     7ff6`42aeb000     7ff6`42af1000        0`00006000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY                 Image      [libpthread_2_28; "/lib/x86_64-linux-gnu/libpthread-2.28.so"]
+     7ff6`42af1000     7ff6`42af2000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY                 Image      [libpthread_2_28; "/lib/x86_64-linux-gnu/libpthread-2.28.so"]
+     7ff6`42af2000     7ff6`42af3000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE                Image      [libpthread_2_28; "/lib/x86_64-linux-gnu/libpthread-2.28.so"]
+     7ff6`42af3000     7ff6`42af9000        0`00006000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE                <unknown>  [................]
+     7ff6`42af9000     7ff6`42b04000        0`0000b000                                                       <unknown>
+     7ff6`42b04000     7ff6`42b05000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY                 Image      [ld_2_28; "/lib/x86_64-linux-gnu/ld-2.28.so"]
+     7ff6`42b05000     7ff6`42b23000        0`0001e000 MEM_PRIVATE MEM_COMMIT  PAGE_EXECUTE_READ            Image      [ld_2_28; "/lib/x86_64-linux-gnu/ld-2.28.so"]
+     7ff6`42b23000     7ff6`42b2b000        0`00008000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY                 Image      [ld_2_28; "/lib/x86_64-linux-gnu/ld-2.28.so"]
+     7ff6`42b2b000     7ff6`42b2c000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY                 Image      [ld_2_28; "/lib/x86_64-linux-gnu/ld-2.28.so"]
+     7ff6`42b2c000     7ff6`42b2d000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE                Image      [ld_2_28; "/lib/x86_64-linux-gnu/ld-2.28.so"]
+     7ff6`42b2d000     7ff6`42b2e000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE                <unknown>  [AI.B...........]
+     7ff6`42b2e000     7fff`cc667000        9`89b39000                                                       <unknown>
+     7fff`cc667000     7fff`cc688000        0`00021000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE                <unknown>  [................]
+     7fff`cc688000     7fff`cc76d000        0`000e5000                                                       <unknown>
+     7fff`cc76d000     7fff`cc76e000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_EXECUTE_READ            Image      [linux_vdso_so; "linux-vdso.so.1"]
      7fff`cc76e000     7fff`cc76f000        0`00001000 MEM_PRIVATE MEM_COMMIT  PAGE_EXECUTE_READ            <unknown>  [................]
```

19. We close logging before exiting WinDbg sessions:

```
0:000> .logclose
Closing open log file 'C:\ALCDA2\x64\App1\App1-WinDbg.log'
```

We recommend exiting WinDbg after each exercise.

Exercise A1 (A64, WinDbg)

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

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

1. Launch WinDbg.

2. Load the core dump *App1.core.21174* from the A64\App1 folder:

```
Microsoft (R) Windows Debugger Version 10.0.27725.1000 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.


Loading Dump File [C:\ALCDA2\A64\App1\App1.core.21174]
64-bit machine not using 64-bit API

************* Path validation summary **************
Response                         Time (ms)     Location
Deferred                                       srv*
Symbol search path is: srv*
Executable search path is:
Generic Unix Version 0 UP Free ARM 64-bit (AArch64)
System Uptime: not available
Process Uptime: not available
..
*** WARNING: Unable to verify timestamp for App1
App1+0xc9b4:
00000000`0040c9b4 svc         #0
```

3. Set logging to a file in case of lengthy output from some commands:

```
0:000> .logopen C:\ALCDA2\A64\App1\App1-WinDbg.log
Opened log file 'C:\ALCDA2\A64\App1\App1-WinDbg.log'
```

4. Specify the dump folder as the symbol path and reload symbols:

```
0:000> .symfix; .sympath+ C:\ALCDA2\A64\App1\
Symbol search path is: srv*;C:\ALCDA2\A64\App1\
Expanded Symbol search path is:
cache*;SRV*https://msdl.microsoft.com/download/symbols;c:\alcda2\a64\app1\

************* Path validation summary **************
Response                         Time (ms)     Location
Deferred                                       srv*
OK                                             C:\ALCDA2\A64\App1\
*** WARNING: Unable to verify timestamp for App1
```

```
0:000> .reload
..
*** WARNING: Unable to verify timestamp for App1

************* Symbol Loading Error Summary **************
Module name            Error
App1                   The system cannot find the file specified

You can troubleshoot most symbol related issues by turning on symbol loading diagnostics (!sym
noisy) and repeating the command that caused symbols to be loaded.
You should also verify that your symbol search path (.sympath) is correct.
```

Note: We ignore warnings and errors as they are not relevant for now.

5. List all threads:

```
0:000> ~
Unable to get thread data for thread 0
.  0  Id: 52b6.52b7 Suspend: 0 Teb: 00000000`00000000 Unfrozen
Unable to get thread data for thread 1
   1  Id: 52b6.52b8 Suspend: 0 Teb: 00000000`00000000 Unfrozen
Unable to get thread data for thread 2
   2  Id: 52b6.52b9 Suspend: 0 Teb: 00000000`00000000 Unfrozen
Unable to get thread data for thread 3
   3  Id: 52b6.52ba Suspend: 0 Teb: 00000000`00000000 Unfrozen
Unable to get thread data for thread 4
   4  Id: 52b6.52bb Suspend: 0 Teb: 00000000`00000000 Unfrozen
Unable to get thread data for thread 5
   5  Id: 52b6.52b6 Suspend: 0 Teb: 00000000`00000000 Unfrozen
```

Note: WinDbg uses the same output format as for Windows memory dumps. Therefore, some data is either reported as errors or shows 0 or NULL pointer values. However, we see process and thread IDs in the format PID.TID:

```
0:000> .formats 52b6
Evaluate expression:
  Hex:     00000000`000052b6
  Decimal: 21174
  Octal:   0000000000000051266
  Binary:  00000000 00000000 00000000 00000000 00000000 00000000 01010010 10110110
  Chars:   ......R.
  Time:    Thu Jan  1 05:52:54 1970
  Float:   low 2.96711e-041 high 0
  Double:  1.04613e-319

0:000> ? 52b6
Evaluate expression: 21174 = 00000000`000052b6
```

6. Get the current thread stack trace:

```
0:000> k
 # Child-SP          RetAddr               Call Site
00 0000fffc`cd38e5f0 00000000`00424cb4     App1!_libc_nanosleep+0x24
01 0000fffc`cd38e630 00000000`004031f8     App1!sleep+0x110
02 0000fffc`cd38e820 00000000`0040320c     App1!bar_one+0x10
03 0000fffc`cd38e830 00000000`00403224     App1!foo_one+0xc
04 0000fffc`cd38e840 00000000`00404c34     App1!thread_one+0x10
05 0000fffc`cd38e860 00000000`00429b60     App1!start_thread+0xb4
06 0000fffc`cd38e990 ffffffff`ffffffff     App1!thread_start+0x30
07 0000fffc`cd38e990 00000000`00000000     0xffffffff`ffffffff
```

7. Get all thread stack traces:

```
0:000> ~*k

Unable to get thread data for thread 0
.  0  Id: 52b6.52b7 Suspend: 0 Teb: 00000000`00000000 Unfrozen
 # Child-SP          RetAddr               Call Site
00 0000fffc`cd38e5f0 00000000`00424cb4     App1!_libc_nanosleep+0x24
01 0000fffc`cd38e630 00000000`004031f8     App1!sleep+0x110
02 0000fffc`cd38e820 00000000`0040320c     App1!bar_one+0x10
03 0000fffc`cd38e830 00000000`00403224     App1!foo_one+0xc
04 0000fffc`cd38e840 00000000`00404c34     App1!thread_one+0x10
05 0000fffc`cd38e860 00000000`00429b60     App1!start_thread+0xb4
06 0000fffc`cd38e990 ffffffff`ffffffff     App1!thread_start+0x30
07 0000fffc`cd38e990 00000000`00000000     0xffffffff`ffffffff

Unable to get thread data for thread 1
   1  Id: 52b6.52b8 Suspend: 0 Teb: 00000000`00000000 Unfrozen
 # Child-SP          RetAddr               Call Site
00 0000fffc`ccb7e5f0 00000000`00424cb4     App1!_libc_nanosleep+0x24
01 0000fffc`ccb7e630 00000000`00403240     App1!sleep+0x110
02 0000fffc`ccb7e820 00000000`00403254     App1!bar_two+0x10
03 0000fffc`ccb7e830 00000000`0040326c     App1!foo_two+0xc
04 0000fffc`ccb7e840 00000000`00404c34     App1!thread_two+0x10
05 0000fffc`ccb7e860 00000000`00429b60     App1!start_thread+0xb4
06 0000fffc`ccb7e990 ffffffff`ffffffff     App1!thread_start+0x30
07 0000fffc`ccb7e990 00000000`00000000     0xffffffff`ffffffff

Unable to get thread data for thread 2
   2  Id: 52b6.52b9 Suspend: 0 Teb: 00000000`00000000 Unfrozen
 # Child-SP          RetAddr               Call Site
00 0000fffc`cc36e5f0 00000000`00424cb4     App1!_libc_nanosleep+0x24
01 0000fffc`cc36e630 00000000`00403288     App1!sleep+0x110
02 0000fffc`cc36e820 00000000`0040329c     App1!bar_three+0x10
03 0000fffc`cc36e830 00000000`004032b4     App1!foo_three+0xc
04 0000fffc`cc36e840 00000000`00404c34     App1!thread_three+0x10
05 0000fffc`cc36e860 00000000`00429b60     App1!start_thread+0xb4
06 0000fffc`cc36e990 ffffffff`ffffffff     App1!thread_start+0x30
07 0000fffc`cc36e990 00000000`00000000     0xffffffff`ffffffff

Unable to get thread data for thread 3
   3  Id: 52b6.52ba Suspend: 0 Teb: 00000000`00000000 Unfrozen
 # Child-SP          RetAddr               Call Site
00 0000fffc`cbb5e5f0 00000000`00424cb4     App1!_libc_nanosleep+0x24
01 0000fffc`cbb5e630 00000000`004032d0     App1!sleep+0x110
02 0000fffc`cbb5e820 00000000`004032e4     App1!bar_four+0x10
03 0000fffc`cbb5e830 00000000`004032fc     App1!foo_four+0xc
04 0000fffc`cbb5e840 00000000`00404c34     App1!thread_four+0x10
05 0000fffc`cbb5e860 00000000`00429b60     App1!start_thread+0xb4
06 0000fffc`cbb5e990 ffffffff`ffffffff     App1!thread_start+0x30
07 0000fffc`cbb5e990 00000000`00000000     0xffffffff`ffffffff

Unable to get thread data for thread 4
   4  Id: 52b6.52bb Suspend: 0 Teb: 00000000`00000000 Unfrozen
 # Child-SP          RetAddr               Call Site
00 0000fffc`cb34e5f0 00000000`00424cb4     App1!_libc_nanosleep+0x24
01 0000fffc`cb34e630 00000000`00403318     App1!sleep+0x110
02 0000fffc`cb34e820 00000000`0040332c     App1!bar_five+0x10
03 0000fffc`cb34e830 00000000`00403344     App1!foo_five+0xc
04 0000fffc`cb34e840 00000000`00404c34     App1!thread_five+0x10
```

```
05 0000fffc`cb34e860 00000000`00429b60     App1!start_thread+0xb4
06 0000fffc`cb34e990 ffffffff`ffffffff     App1!thread_start+0x30
07 0000fffc`cb34e990 00000000`00000000     0xffffffff`ffffffff

Unable to get thread data for thread 5
   5  Id: 52b6.52b6 Suspend: 0 Teb: 00000000`00000000 Unfrozen
 # Child-SP          RetAddr               Call Site
00 0000ffff`d30b8490 00000000`00424cb4     App1!_libc_nanosleep+0x24
01 0000ffff`d30b84d0 00000000`004033e0     App1!sleep+0x110
02 0000ffff`d30b86c0 00000000`0040ec4c     App1!main+0x90
03 0000ffff`d30b8710 00000000`00403090     App1!_libc_start_main+0x304
04 0000ffff`d30b8870 00000000`00000000     App1!start+0x4c
```

8. Switch to thread #1 (threads are numbered from 0) and get its stack trace:

```
0:000> ~1s
App1!_libc_nanosleep+0x24:
00000000`0040c9b4 d4000001 svc         #0

0:001> k
 # Child-SP          RetAddr               Call Site
00 0000fffc`ccb7e5f0 00000000`00424cb4     App1!_libc_nanosleep+0x24
01 0000fffc`ccb7e630 00000000`00403240     App1!sleep+0x110
02 0000fffc`ccb7e820 00000000`00403254     App1!bar_two+0x10
03 0000fffc`ccb7e830 00000000`0040326c     App1!foo_two+0xc
04 0000fffc`ccb7e840 00000000`00404c34     App1!thread_two+0x10
05 0000fffc`ccb7e860 00000000`00429b60     App1!start_thread+0xb4
06 0000fffc`ccb7e990 ffffffff`ffffffff     App1!thread_start+0x30
07 0000fffc`ccb7e990 00000000`00000000     0xffffffff`ffffffff
```

9. Check that *bar_two* called the *sleep* function by comparing the return address on the call stack from the disassembly output:

```
0:001> uf bar_two
App1!bar_two:
00000000`00403230 a9bf7bfd stp         fp,lr,[sp,#-0x10]!
00000000`00403234 910003fd mov         fp,sp
00000000`00403238 12800000 mov         w0,#-1
00000000`0040323c 9400865a bl          App1!sleep (00000000`00424ba4)
00000000`00403240 a8c17bfd ldp         fp,lr,[sp],#0x10
00000000`00403244 d65f03c0 ret
```

Another way to do that is to disassemble backward the return address and check if the last instruction is BL:

```
0:001> ub 00000000`00403240
App1!thread_one+0xc:
00000000`00403220 97fffff8 bl          App1!foo_one (00000000`00403200)
00000000`00403224 d2800000 mov         x0,#0
00000000`00403228 a8c27bfd ldp         fp,lr,[sp],#0x20
00000000`0040322c d65f03c0 ret
App1!bar_two:
00000000`00403230 a9bf7bfd stp         fp,lr,[sp,#-0x10]!
00000000`00403234 910003fd mov         fp,sp
00000000`00403238 12800000 mov         w0,#-1
00000000`0040323c 9400865a bl          App1!sleep (00000000`00424ba4)
```

10. Get the *App1* data section from the contents of pmap (*App1.pmap.21174*):

```
21174:   ./App1
0000000000400000    768K r-x-- App1
00000000004c0000    128K rw--- App1
0000000001fa0000    256K rw---   [ anon ]
0000fffccab40000     64K -----   [ anon ]
0000fffccab50000   8192K rw---   [ anon ]
0000fffccb350000     64K -----   [ anon ]
0000fffccb360000   8192K rw---   [ anon ]
0000fffccbb60000     64K -----   [ anon ]
0000fffccbb70000   8192K rw---   [ anon ]
0000fffccc370000     64K -----   [ anon ]
0000fffccc380000   8192K rw---   [ anon ]
0000fffcccb80000     64K -----   [ anon ]
0000fffcccb90000   8192K rw---   [ anon ]
0000fffccd390000     64K r----   [ anon ]
0000fffccd3a0000     64K r-x--   [ anon ]
0000ffffd3090000    192K rw---   [ stack ]
 total            42752K
```

11. Compare with the region information in the core dump:

```
0:001> !address

Mapping file section regions...
Mapping module regions...

        BaseAddress      EndAddress+1        RegionSize     Type       State                 Protect             Usage
-------------------------------------------------------------------------------------------------------------------------
+       0`00000000       0`00400000       0`00400000                                                <unknown>
+       0`00400000       0`004c0000       0`000c0000 MEM_PRIVATE MEM_COMMIT  PAGE_EXECUTE_READ      Image      [App1; "/home/opc/ALCDA2/App1/App1"]
+       0`004c0000       0`004e0000       0`00020000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE         Image      [App1; "/home/opc/ALCDA2/App1/App1"]
+       0`004e0000       0`01fa0000       0`01ac0000                                                <unknown>
+       0`01fa0000       0`01fe0000       0`00040000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE         <unknown>  [................]
+       0`01fe0000     fffc`cab40000    fffc`c8b60000                                               <unknown>
+     fffc`cab40000    fffc`cab50000       0`00010000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY        <unknown>  [................]
+     fffc`cab50000    fffc`cb350000       0`00800000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE       <unknown>  [................]
+     fffc`cb350000    fffc`cb360000       0`00010000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY        <unknown>  [................]
+     fffc`cb360000    fffc`cbb60000       0`00800000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE       <unknown>  [................]
+     fffc`cbb60000    fffc`cbb70000       0`00010000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY        <unknown>  [................]
+     fffc`cbb70000    fffc`cc370000       0`00800000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE       <unknown>  [................]
+     fffc`cc370000    fffc`cc380000       0`00010000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY        <unknown>  [................]
+     fffc`cc380000    fffc`ccb80000       0`00800000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE       <unknown>  [................]
+     fffc`ccb80000    fffc`ccb90000       0`00010000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY        <unknown>  [................]
+     fffc`ccb90000    fffc`cd390000       0`00800000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE       <unknown>  [................]
+     fffc`cd390000    fffc`cd3a0000       0`00010000                                                <unknown>
+     fffc`cd3a0000    fffc`cd3b0000       0`00010000 MEM_PRIVATE MEM_COMMIT  PAGE_EXECUTE_READ      Image      [linux_vdso_so; "linux-vdso.so.1"]
+     fffc`cd3b0000    ffff`d3090000       3`05ce0000                                                <unknown>
+     ffff`d3090000    ffff`d30c0000       0`00030000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE       <unknown>  [................]
```

12. Dump the data region with possible symbolic information (we truncated the output):

```
0:001> dps 0`004c0000 0`004e0000
[...]
00000000`004d0fe8  00000000`00000000
00000000`004d0ff0  00000000`00000000
00000000`004d0ff8  00000000`004d0788 App1!main_arena
00000000`004d1000  00000000`00000000
00000000`004d1008  00000000`00000001
00000000`004d1010  00000000`0003f078
00000000`004d1018  00000000`0003f078
00000000`004d1020  00000000`00421c08 App1!_default_morecore
00000000`004d1028  00000000`00000001
00000000`004d1030  ffffffff`00000001
00000000`004d1038  00000000`0041cc00 App1!memalign_hook_ini
00000000`004d1040  00000000`0041d688 App1!realloc_hook_ini
00000000`004d1048  00000000`00000000
00000000`004d1050  ffffffff`00000008
00000000`004d1058  000000ff`00000002
00000000`004d1060  00000000`ffffffff
```

```
00000000`004d1068  0000ffff`d30bf6dd
00000000`004d1070  0000ffff`d30bf6db
00000000`004d1078  00000000`00010000
00000000`004d1080  00000000`00000006
00000000`004d1088  00000000`00000000
00000000`004d1090  00000000`00000000
00000000`004d1098  00000000`00000001
00000000`004d10a0  00000000`00000000
00000000`004d10a8  00000000`00000000
00000000`004d10b0  00000000`00000000
00000000`004d10b8  00000000`00000000
00000000`004d10c0  00000000`00000000
00000000`004d10c8  00000000`00000001
00000000`004d10d0  00000000`00000000
00000000`004d10d8  00000000`00000000
00000000`004d10e0  00000000`00000000
00000000`004d10e8  00000000`0042c6a0 App1!dl_make_stack_executable
00000000`004d10f0  00000002`00000a03
00000000`004d10f8  00000000`004045a8 App1!_pthread_init_static_tls
00000000`004d1100  00000000`00000001
00000000`004d1108  ffffffff`fffffffe
00000000`004d1110  00000000`004d1068 App1!_progname
00000000`004d1118  00000000`00000000
00000000`004d1120  00000000`0048ad20 App1!$d+0xe0
00000000`004d1128  00000000`0048ac30 App1!$d+0x38
00000000`004d1130  7fffffff`00000001
00000000`004d1138  00000000`0048ac40 App1!$d
00000000`004d1140  00000000`00000000
00000000`004d1148  00000000`00000000
00000000`004d1150  00000000`00000000
[...]
```

The output is in the following format:

```
address  value
```

Some values may have associated symbols in the format module!name+offset:

```
address  value  symbol
```

For example, from the output above:

```
00000000`004d1110  00000000`004d1068 App1!_progname
```

To list all values with symbols, we can use the **dpS** command (it doesn't show the value addresses):

```
0:001> dpS 0`004c0000 0`004e0000
 00000000`004d6e70 App1!res
 00000000`004d13c0 App1!nl_global_locale
 00000000`004d13c0 App1!nl_global_locale
 00000000`004d13e0 App1!nl_global_locale+0x20
 00000000`004d13c8 App1!nl_global_locale+0x8
 00000000`00403190 App1!frame_dummy
 00000000`00403140 App1!__do_global_dtors_aux
 00000000`00402ffc App1!fini
 00000000`0048a2d0 App1!$d+0x20
 00000000`0048a2f0 App1!$d+0x40
 00000000`0048a308 App1!$d+0x58
 00000000`0048a320 App1!$d+0x70
 00000000`0048a330 App1!$d+0x80
```

```
00000000`0048a348 App1!$d+0x98
00000000`0048a358 App1!$d+0xa8
00000000`0048a368 App1!$d+0xb8
00000000`0048a380 App1!$d+0xd0
00000000`0048a398 App1!$d+0xe8
00000000`0048a3c0 App1!$d+0x110
00000000`0048a3d8 App1!$d+0x128
00000000`0048a3e8 App1!$d+0x138
00000000`0048a400 App1!$d+0x150
00000000`0048a418 App1!$d+0x168
00000000`0048a438 App1!$d+0x188
00000000`0048a450 App1!$d+0x1a0
00000000`0048a470 App1!$d+0x1c0
00000000`0048a488 App1!$d+0x1d8
00000000`0048a4a0 App1!$d+0x1f0
00000000`0048a4b8 App1!$d+0x208
00000000`0048a4d0 App1!$d+0x220
00000000`00409a50 App1!_pthread_key_create
00000000`004231c0 App1!_memmove_generic
00000000`004231d0 App1!_memcpy_generic
00000000`00423fc0 App1!_memset_generic
00000000`00424480 App1!_strlen_generic
00000000`00424480 App1!_strlen_generic
00000000`004d0038 App1!stack_cache
00000000`004d0038 App1!stack_cache
00000000`004d5eb0 App1!initial
00000000`00486b88 App1!_gcc_personality_v0
00000000`004d0088 App1!IO_2_1_stderr_
00000000`004d02b0 App1!IO_2_1_stdout_
00000000`004d6428 App1!IO_stdfile_2_lock
00000000`004d0168 App1!IO_wide_data_2
00000000`004a1950 App1!IO_file_jumps
00000000`004a1800 App1!IO_wfile_jumps
00000000`004d04d8 App1!IO_2_1_stdin_
00000000`004d6438 App1!IO_stdfile_1_lock
00000000`004d0390 App1!IO_wide_data_1
00000000`004a1950 App1!IO_file_jumps
00000000`004a1800 App1!IO_wfile_jumps
00000000`004d6448 App1!IO_stdfile_0_lock
00000000`004d05b8 App1!IO_wide_data_0
00000000`004a1950 App1!IO_file_jumps
00000000`004a1800 App1!IO_wfile_jumps
00000000`004d0088 App1!IO_2_1_stderr_
00000000`004d02b0 App1!IO_2_1_stdout_
00000000`004d04d8 App1!IO_2_1_stdin_
00000000`004d07e8 App1!main_arena+0x60
00000000`004d07e8 App1!main_arena+0x60
00000000`004d07f8 App1!main_arena+0x70
00000000`004d07f8 App1!main_arena+0x70
00000000`004d0808 App1!main_arena+0x80
00000000`004d0808 App1!main_arena+0x80
00000000`004d0818 App1!main_arena+0x90
00000000`004d0818 App1!main_arena+0x90
00000000`004d0828 App1!main_arena+0xa0
00000000`004d0828 App1!main_arena+0xa0
00000000`004d0838 App1!main_arena+0xb0
00000000`004d0838 App1!main_arena+0xb0
00000000`004d0848 App1!main_arena+0xc0
00000000`004d0848 App1!main_arena+0xc0
00000000`004d0858 App1!main_arena+0xd0
```

```
00000000`004d0858 App1!main_arena+0xd0
00000000`004d0868 App1!main_arena+0xe0
00000000`004d0868 App1!main_arena+0xe0
00000000`004d0878 App1!main_arena+0xf0
00000000`004d0878 App1!main_arena+0xf0
00000000`004d0888 App1!main_arena+0x100
00000000`004d0888 App1!main_arena+0x100
00000000`004d0898 App1!main_arena+0x110
[...]
00000000`004d0fc8 App1!main_arena+0x840
00000000`004d0788 App1!main_arena
00000000`00421c08 App1!_default_morecore
00000000`0041cc00 App1!memalign_hook_ini
00000000`0041d688 App1!realloc_hook_ini
00000000`0042c6a0 App1!dl_make_stack_executable
00000000`004045a8 App1!_pthread_init_static_tls
00000000`004d1068 App1!_progname
00000000`0048ad20 App1!$d+0xe0
00000000`0048ac30 App1!$d+0x38
00000000`0048ac40 App1!$d
00000000`0048ac30 App1!$d+0x38
00000000`0048ad20 App1!$d+0xe0
00000000`0048ac50 App1!$d+0x10
00000000`0048ad20 App1!$d+0xe0
00000000`0048ac60 App1!$d+0x20
00000000`0048ac70 App1!$d+0x30
00000000`0048ac60 App1!$d+0x20
00000000`0048ad20 App1!$d+0xe0
00000000`0048ac88 App1!$d+0x48
00000000`0048ad20 App1!$d+0xe0
00000000`0048aca0 App1!$d+0x60
00000000`0048acb0 App1!$d+0x70
00000000`0048aca0 App1!$d+0x60
00000000`0048ad20 App1!$d+0xe0
00000000`0048acc0 App1!$d+0x80
00000000`0048acd0 App1!$d+0x90
00000000`0048ad20 App1!$d+0xe0
00000000`0048ace0 App1!$d+0xa0
00000000`0048ad20 App1!$d+0xe0
00000000`0048acd0 App1!$d+0x90
00000000`0048acf0 App1!$d+0xb0
00000000`0048ad00 App1!$d+0xc0
00000000`0048ad20 App1!$d+0xe0
00000000`0048ad18 App1!$d+0xd8
00000000`0048ad20 App1!$d+0xe0
00000000`0048ad00 App1!$d+0xc0
00000000`0048ad30 App1!$d+0xf0
00000000`0048ad48 App1!$d+0x108
00000000`0048ad20 App1!$d+0xe0
00000000`0048ad58 App1!$d+0x118
00000000`0048ad20 App1!$d+0xe0
00000000`0048ad48 App1!$d+0x108
00000000`0048ad70 App1!$d+0x130
00000000`0048b888 App1!nl_C_LC_CTYPE
00000000`00499f18 App1!nl_C_LC_NUMERIC
00000000`00499f88 App1!nl_C_LC_TIME
00000000`0049aec0 App1!nl_C_LC_COLLATE
00000000`00499d58 App1!nl_C_LC_MONETARY
00000000`00499ce0 App1!nl_C_LC_MESSAGES
00000000`0049a9e0 App1!nl_C_LC_PAPER
```

```
00000000`0049aa38 App1!nl_C_LC_NAME
00000000`0049aac0 App1!nl_C_LC_ADDRESS
00000000`0049ab98 App1!nl_C_LC_TELEPHONE
00000000`0049ac10 App1!nl_C_LC_MEASUREMENT
00000000`0049ad08 App1!nl_C_LC_IDENTIFICATION
00000000`0048d1c0 App1!nl_C_LC_CTYPE_class+0x100
00000000`0048c2c0 App1!nl_C_LC_CTYPE_tolower+0x200
00000000`0048c8c0 App1!nl_C_LC_CTYPE_toupper+0x200
00000000`00497450 App1!nl_C_name
00000000`00497450 App1!nl_C_name
00000000`00497450 App1!nl_C_name
00000000`00497450 App1!nl_C_name
00000000`00497450 App1!nl_C_name
00000000`00497450 App1!nl_C_name
00000000`00497450 App1!nl_C_name
00000000`00497450 App1!nl_C_name
00000000`00497450 App1!nl_C_name
00000000`00497450 App1!nl_C_name
00000000`00497450 App1!nl_C_name
00000000`00497450 App1!nl_C_name
00000000`00497450 App1!nl_C_name
00000000`004975d0 App1!nl_C_locobj+0x158
00000000`00498850 App1!$d+0x30
00000000`00498850 App1!$d+0x30
00000000`00465268 App1!_libc_dlopen_mode
00000000`004651ec App1!_libc_dlsym
00000000`0046517c App1!_libc_dlclose
00000000`00465460 App1!dl_initial_error_catch_tsd
00000000`0049b540 App1!nl_default_default_domain
00000000`0047e9e8 App1!_dlopen
00000000`0047ea3c App1!_dlclose
00000000`0047ea98 App1!_dlsym
00000000`0047eb4c App1!_dlvsym
00000000`00470f60 App1!_dlerror
00000000`00471324 App1!_dladdr
00000000`00471330 App1!_dladdr1
00000000`00471470 App1!_dlinfo
00000000`00471528 App1!_dlmopen
00000000`004d1078 App1!dl_pagesize
00000000`004a1e68 App1!_EH_FRAME_BEGIN__
00000000`004cfb20 App1!
00000000`004d5618 App1!static_map
00000000`00403f44 App1!_reclaim_stacks
00000000`004d1588 App1!object.6205
00000000`004d7d40 App1!_libc_multiple_threads
00000000`004d5a78 App1!static_slotinfo
00000000`004d78e0 App1!_fork_generation
00000000`004d6570 App1!fork_handler_pool+0x8
00000000`0048a618 App1!unsecure_envvars.10865+0x118
00000000`004046f0 App1!_wait_lookup_done
00000000`00400040 App1+0x40
????????`????????
```

13. Explore the contents of memory pointed to by *App1!memalign_hook_ini* and *App1!_progname* addresses:

```
0:001> u 00000000`0041cc00
App1!memalign_hook_ini:
00000000`0041cc00 a9b97bfd stp         fp,lr,[sp,#-0x70]!
00000000`0041cc04 910003fd mov         fp,sp
00000000`0041cc08 a9025bf5 stp         x21,x22,[sp,#0x20]
00000000`0041cc0c 900005b6 adrp        x22,App1!+0x18 (00000000`004d0000)
00000000`0041cc10 58004815 ldr         x21,App1!$d (00000000`0041d510)
00000000`0041cc14 911c62c2 add         x2,x22,#0x718
00000000`0041cc18 a90153f3 stp         x19,x20,[sp,#0x10]
00000000`0041cc1c a90363f7 stp         x23,x24,[sp,#0x30]

0:001> dp App1!_progname
00000000`004d1068  0000ffff`d30bf6dd 0000ffff`d30bf6db
00000000`004d1078  00000000`00010000 00000000`00000006
00000000`004d1088  00000000`00000000 00000000`00000000
00000000`004d1098  00000000`00000001 00000000`00000000
00000000`004d10a8  00000000`00000000 00000000`00000000
00000000`004d10b8  00000000`00000000 00000000`00000000
00000000`004d10c8  00000000`00000001 00000000`00000000
00000000`004d10d8  00000000`00000000 00000000`00000000

0:001> dc 0000ffff`d30bf6dd
0000ffff`d30bf6dd  31707041 47445800 5345535f 4e4f4953  App1.XDG_SESSION
0000ffff`d30bf6ed  3d44495f 30353836 534f4800 4d414e54  _ID=6850.HOSTNAM
0000ffff`d30bf6fd  6e693d45 6e617473 322d6563 31313230  E=instance-20211
0000ffff`d30bf70d  2d393031 34303032 4c455300 58554e49  109-2004.SELINUX
0000ffff`d30bf71d  4c4f525f 45525f45 53455551 3d444554  _ROLE_REQUESTED=
0000ffff`d30bf72d  52455400 74783d4d 2d6d7265 63363532  .TERM=xterm-256c
0000ffff`d30bf73d  726f6c6f 45485300 2f3d4c4c 2f6e6962  olor.SHELL=/bin/
0000ffff`d30bf74d  68736162 53494800 5a495354 30313d45  bash.HISTSIZE=10

0:001> da 0000ffff`d30bf6dd
0000ffff`d30bf6dd  "App1"

0:001> db 0000ffff`d30bf6dd
0000ffff`d30bf6dd  41 70 70 31 00 58 44 47-5f 53 45 53 53 49 4f 4e  App1.XDG_SESSION
0000ffff`d30bf6ed  5f 49 44 3d 36 38 35 30-00 48 4f 53 54 4e 41 4d  _ID=6850.HOSTNAM
0000ffff`d30bf6fd  45 3d 69 6e 73 74 61 6e-63 65 2d 32 30 32 31 31  E=instance-20211
0000ffff`d30bf70d  31 30 39 2d 32 30 30 34-00 53 45 4c 49 4e 55 58  109-2004.SELINUX
0000ffff`d30bf71d  5f 52 4f 4c 45 5f 52 45-51 55 45 53 54 45 44 3d  _ROLE_REQUESTED=
0000ffff`d30bf72d  00 54 45 52 4d 3d 78 74-65 72 6d 2d 32 35 36 63  .TERM=xterm-256c
0000ffff`d30bf73d  6f 6c 6f 72 00 53 48 45-4c 4c 3d 2f 62 69 6e 2f  olor.SHELL=/bin/
0000ffff`d30bf74d  62 61 73 68 00 48 49 53-54 53 49 5a 45 3d 31 30  bash.HISTSIZE=10
```

Note: We see that a hook function is installed for *memalign* and *realloc*. Please find the following documentation for hook functions here:

https://www.gnu.org/software/libc/manual/html_node/Hooks-for-Malloc.html

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

```
0:001> dp environ
00000000`004d64c8  0000ffff`d30b8888 00000000`00000000
00000000`004d64d8  00000000`00000000 00000000`00000000
00000000`004d64e8  00000000`00000000 00000000`00000000
00000000`004d64f8  00000000`00000000 00000000`00000000
00000000`004d6508  00000000`00000000 00000000`00000000
```

```
00000000`004d6518  00000000`00000000 00000000`00000000
00000000`004d6528  00000000`00000000 00000000`00000000
00000000`004d6538  00000000`00000000 00000000`00000000

0:001> dp 0000ffff`d30b8888
0000ffff`d30b8888  0000ffff`d30bf6e2 0000ffff`d30bf6f6
0000ffff`d30b8898  0000ffff`d30bf716 0000ffff`d30bf72e
0000ffff`d30b88a8  0000ffff`d30bf742 0000ffff`d30bf752
0000ffff`d30b88b8  0000ffff`d30bf760 0000ffff`d30bf783
0000ffff`d30b88c8  0000ffff`d30bf79e 0000ffff`d30bf7b1
0000ffff`d30b88d8  0000ffff`d30bf7ba 0000ffff`d30bfe72
0000ffff`d30b88e8  0000ffff`d30bfe8b 0000ffff`d30bfee5
0000ffff`d30b88f8  0000ffff`d30bfeff 0000ffff`d30bff10

0:001> da 0000ffff`d30bf6e2
0000ffff`d30bf6e2  "XDG_SESSION_ID=6850"

0:001> dpa 0000ffff`d30b8888
0000ffff`d30b8888  0000ffff`d30bf6e2 "XDG_SESSION_ID=6850"
0000ffff`d30b8890  0000ffff`d30bf6f6 "HOSTNAME=instance-20211109-2004"
0000ffff`d30b8898  0000ffff`d30bf716 "SELINUX_ROLE_REQUESTED="
0000ffff`d30b88a0  0000ffff`d30bf72e "TERM=xterm-256color"
0000ffff`d30b88a8  0000ffff`d30bf742 "SHELL=/bin/bash"
0000ffff`d30b88b0  0000ffff`d30bf752 "HISTSIZE=1000"
0000ffff`d30b88b8  0000ffff`d30bf760 "SSH_CLIENT=37.228.238.120 61099 22"
0000ffff`d30b88c0  0000ffff`d30bf783 "SELINUX_USE_CURRENT_RANGE="
0000ffff`d30b88c8  0000ffff`d30bf79e "SSH_TTY=/dev/pts/1"
0000ffff`d30b88d0  0000ffff`d30bf7b1 "USER=opc"
0000ffff`d30b88d8  0000ffff`d30bf7ba "LS_COLORS=rs=0:di=38;5;27:ln=38;5;51:mh=44;38;5;15:pi=4"
0000ffff`d30b88e0  0000ffff`d30bfe72 "MAIL=/var/spool/mail/opc"
0000ffff`d30b88e8  0000ffff`d30bfe8b "PATH=/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:"
0000ffff`d30b88f0  0000ffff`d30bfee5 "PWD=/home/opc/ALCDA2/App1"
0000ffff`d30b88f8  0000ffff`d30bfeff "LANG=en_US.UTF-8"
0000ffff`d30b8900  0000ffff`d30bff10 "SELINUX_LEVEL_REQUESTED="
```

19. Now we look at how to perform a memory search.

```
0:001> s 0`004c0000 0`004f0000 6
00000000`004c002a  06 9a 05 9b 04 9c 03 02-49 0a de dd dc db da d9  ........I.......
00000000`004c007e  06 9a 05 45 95 0a 96 09-46 9b 04 9c 03 6c 0a de  ...E....F....l..
00000000`004c012d  06 00 00 00 41 0e a0 01-9d 14 9e 13 41 0d 1d 46  ....A.......A..F
00000000`004c018d  06 9e 05 41 0d 1d 41 93-04 94 03 5b 0a de dd d4  ...A..A....[....
00000000`004c020d  06 9e 05 41 0d 1d 41 93-04 94 03 5b 0a de dd d4  ...A..A....[....
00000000`004c0285  06 9e 05 41 0d 1d 42 93-04 94 03 95 02 96 01 75  ...A..B........u
00000000`004c04fe  06 04 00 00 80 07 88 01-90 0b 00 b0 08 04 00 00  ................
00000000`004cfe78  06 00 00 00 00 00 00 00-50 01 3a cd fc ff 00 00  ........P.:.....
00000000`004d0048  06 00 00 00 00 00 00 00-40 f1 34 cb fc ff 00 00  ........@.4.....
00000000`004d1080  06 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
00000000`004d7e00  06 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
```

Note: It is possible to search through non-accessible regions as well; they are ignored:

```
0:001> s-q 0 Lffffff 6
00000000`0048b208  00000000`00000006 00000000`0000006f
00000000`0048be40  00000000`00000006 00000018`00000001
00000000`0048bf28  00000000`00000006 00000018`00000001
00000000`0048bfc0  00000000`00000006 00000018`00000001
00000000`00499f50  00000000`00000006 00000000`00498240
00000000`0049b728  00000000`00000006 00000000`00000002
00000000`004cfe78  00000000`00000006 0000fffc`cd3a0150
```

```
00000000`004d0048  00000000`00000006 0000fffc`cb34f140
00000000`004d1080  00000000`00000006 00000000`00000000
00000000`004d7e00  00000000`00000006 00000000`00000000
[...]

0:001> s-a 0000ffff`d30b88a8 L100000 "bin"
0000ffff`d30bf749  62 69 6e 2f 62 61 73 68-00 48 49 53 54 53 49 5a  bin/bash.HISTSIZ
0000ffff`d30bfe9b  62 69 6e 3a 2f 75 73 72-2f 62 69 6e 3a 2f 75 73  bin:/usr/bin:/us
0000ffff`d30bfea4  62 69 6e 3a 2f 75 73 72-2f 6c 6f 63 61 6c 2f 73  bin:/usr/local/s
0000ffff`d30bfeb4  62 69 6e 3a 2f 75 73 72-2f 73 62 69 6e 3a 2f 68  bin:/usr/sbin:/h
0000ffff`d30bfebe  62 69 6e 3a 2f 68 6f 6d-65 2f 6f 70 63 2f 2e 6c  bin:/home/opc/.l
0000ffff`d30bfed3  62 69 6e 3a 2f 68 6f 6d-65 2f 6f 70 63 2f 62 69  bin:/home/opc/bi
0000ffff`d30bfee1  62 69 6e 00 50 57 44 3d-2f 68 6f 6d 65 2f 6f 70  bin.PWD=/home/op
0000ffff`d30bffa5  62 69 6e 2f 6c 65 73 73-70 69 70 65 2e 73 68 20  bin/lesspipe.sh
```

Note: It is also possible to show all possible string fragments if any:

```
0:001> s-sa 0 Lfffffff
00000000`00400001  "ELF"
00000000`00400018  "D0@"
00000000`0040019c  "GNU"
00000000`004001bc  "GNU"
00000000`004001d1  "48y"
[...]
00000000`004a12b0  "weak version `"
00000000`004a12c0  "' not found (required by "
00000000`004a12e0  "version `"
00000000`004a12f0  "version lookup error"
00000000`004a1308  "cannot allocate version referenc"
00000000`004a1328  "e table"
00000000`004a1330  " of Verneed record"
00000000`004a1348  "RTLD_NEXT used in code not dynam"
00000000`004a1368  "ically loaded"
[...]
00000000`004d6588  "D?@"
00000000`004d7880  "@}M"
00000000`004d7d08  "xZM"
00000000`004d7d38  "peM"
00000000`01fa0700  "pnM"
00000000`01fa1680  "linux-vdso.so.1"
00000000`01fa16e0  "tls/atomics/"
```

15. To get process uid/gid and other useful data, use this WinDbg extension command:

```
0:000> !ntprpsinfo
NT_PRPSINFO (process info):
    state: 0, sname: t, zomb: 0, nice: 0, flag: 0x40400000
    uid: 1000, gid: 1000, pid: 21174, ppid: 20730, pgrp: 21174, sid: 20730
    fname: App1
    psargs: ./App1
```

16. Get the list of loaded modules:

```
0:001> lm
start             end                 module name
00000000`00400000 00000000`004e0000   App1     T (service symbols: ELF Export Symbols)
c:\alcda2\a64\app1\App1
```

```
0:001> lmv
start             end                 module name
00000000`00400000 00000000`004e0000   App1     T (service symbols: ELF Export Symbols)
c:\alcda2\a64\app1\App1
    Loaded symbol image file: App1
    Image path: /home/opc/ALCDA2/App1/App1
    Image name: App1
    Browse all global symbols  functions  data
    Timestamp:        unavailable (FFFFFFFE)
    CheckSum:         missing
    ImageSize:        000E0000
    Details:
0000fffc`cd3a0000 0000fffc`cd3b0000   linux_vdso_so T (service symbols: ELF In Memory Symbols)
    Loaded symbol image file: linux-vdso.so.1
    Image path: linux-vdso.so.1
    Image name: linux-vdso.so.1
    Browse all global symbols  functions  data
    Timestamp:        unavailable (FFFFFFFE)
    CheckSum:         missing
    ImageSize:        00010000
    Details:
```

Note: We don't see shared libraries except *vdso* (https://man7.org/linux/man-pages/man7/vdso.7.html) because they were statically linked. We also created the version of a dynamically linked *App1.shared* executable. If we load its core dump *App1.shared.core.22442* in the new instance of WinDbg, we see the list of shared libraries:

```
Microsoft (R) Windows Debugger Version 10.0.27725.1000 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.


Loading Dump File [C:\ALCDA2\A64\App1\App1.shared.core.22442]
64-bit machine not using 64-bit API

************* Path validation summary **************
Response                         Time (ms)     Location
Deferred                                       srv*
Symbol search path is: srv*
Executable search path is:
Generic Unix Version 0 UP Free ARM 64-bit (AArch64)
System Uptime: not available
Process Uptime: not available
.....
*** WARNING: Unable to verify timestamp for libc-2.17.so
*** WARNING: Unable to verify timestamp for App1.shared
libc_2_17!nanosleep+0x24:
0000ffff`0496dd64 svc         #0

0:000> .logappend C:\ALCDA2\A64\App1\App1-WinDbg.log
Opened log file 'C:\ALCDA2\A64\App1\App1-WinDbg.log'

0:000> .sympath+ C:\ALCDA2\A64\App1
*** WARNING: Unable to verify timestamp for libc-2.17.so
Symbol search path is: srv*;C:\ALCDA2\A64\App1
Expanded Symbol search path is:
cache*;SRV*https://msdl.microsoft.com/download/symbols;c:\alcda2\a64\app1

************* Path validation summary **************
Response                         Time (ms)     Location
Deferred                                       srv*
```

```
OK                                                  C:\ALCDA2\A64\App1

0:000> .reload
....*** WARNING: Unable to verify timestamp for libc-2.17.so
.
*** WARNING: Unable to verify timestamp for App1.shared

************* Symbol Loading Error Summary **************
Module name            Error
App1                   The system cannot find the file specified
libc-2.17              The system cannot find the file specified

You can troubleshoot most symbol related issues by turning on symbol loading diagnostics (!sym
noisy) and repeating the command that caused symbols to be loaded.
You should also verify that your symbol search path (.sympath) is correct.

0:000> lm
start             end                 module name
00000000`00400000 00000000`00430000   App1     T (service symbols: ELF Export Symbols)
c:\alcda2\a64\app1\App1.shared
0000ffff`048c0000 0000ffff`04a50000   libc_2_17 T (service symbols: ELF In Memory Symbols)
0000ffff`04a50000 0000ffff`04a90000   libpthread_2_17 T (service symbols: ELF In Memory
Symbols)
0000ffff`04ab0000 0000ffff`04ac0000   linux_vdso_so   (deferred)
0000ffff`04ac0000 0000ffff`04af1168   ld_2_17    (deferred)
```

17. Disassemble the *bar_one* function and follow the indirect *sleep* function call:

```
0:000> k
# Child-SP          RetAddr               Call Site
00 0000ffff`048be750 0000ffff`0496da20     libc_2_17!nanosleep+0x24
01 0000ffff`048be790 00000000`00400738     libc_2_17!sleep+0x11c
02 0000ffff`048be990 00000000`0040074c     App1!bar_one+0x10
03 0000ffff`048be9a0 00000000`00400764     App1!foo_one+0xc
04 0000ffff`048be9b0 0000ffff`04a57d40     App1!thread_one+0x10
05 0000ffff`048be9d0 0000ffff`049a2d00     libpthread_2_17!_pthread_get_minstack+0x1394
06 0000ffff`048beb00 ffffffff`ffffffff     libc_2_17!clone+0x80
07 0000ffff`048beb00 00000000`00000000     0xffffffff`ffffffff

0:000> uf bar_one
App1!bar_one:
00000000`00400728 a9bf7bfd stp         fp,lr,[sp,#-0x10]!
00000000`0040072c 910003fd mov         fp,sp
00000000`00400730 12800000 mov         w0,#-1
00000000`00400734 97ffff93 bl          App1!$x+0x30 (00000000`00400580)
00000000`00400738 a8c17bfd ldp         fp,lr,[sp],#0x10
00000000`0040073c d65f03c0 ret

0:000> u 00000000`00400580
App1!$x+0x30:
00000000`00400580 90000110 adrp        xip0,App1!+0x18 (00000000`00420000)
00000000`00400584 f9400611 ldr         xip1,[xip0,#8]
00000000`00400588 91002210 add         xip0,xip0,#8
00000000`0040058c d61f0220 br          xip1
00000000`00400590 90000110 adrp        xip0,App1!+0x18 (00000000`00420000)
00000000`00400594 f9400a11 ldr         xip1,[xip0,#0x10]
00000000`00400598 91004210 add         xip0,xip0,#0x10
00000000`0040059c d61f0220 br          xip1
```

Note: XIP0/XIP1 are mnemonics for X16/X17 registers used for inter-procedure-call.

```
0:000> dp 00000000`00420000 + 8
00000000`00420008  0000ffff`0496d904 0000ffff`04a57fd0
00000000`00420018  00000000`00400550 00000000`00400550
00000000`00420028  00000000`00000000 00000000`00000000
00000000`00420038  00000000`00000000 00000000`00000000
00000000`00420048  00000000`00000000 00000000`00000000
00000000`00420058  00000000`00000000 00000000`00000000
00000000`00420068  00000000`00000000 00000000`00000000
00000000`00420078  00000000`00000000 00000000`00000000

0:000> u 0000ffff`0496d904
libc_2_17!sleep:
0000ffff`0496d904 d106c3ff sub         sp,sp,#0x1B0
0000ffff`0496d908 a9bb7bfd stp         fp,lr,[sp,#-0x50]!
0000ffff`0496d90c 910003fd mov         fp,sp
0000ffff`0496d910 a90153f3 stp         x19,x20,[sp,#0x10]
0000ffff`0496d914 a9025bf5 stp         x21,x22,[sp,#0x20]
0000ffff`0496d918 a90363f7 stp         x23,x24,[sp,#0x30]
0000ffff`0496d91c f90023f9 str         x25,[sp,#0x40]
0000ffff`0496d920 34000e40 cbz         w0,libc_2_17!sleep+0x1e4 (0000ffff`0496dae8)

0:000> ln 0000ffff`0496d904
Browse module
Set bu breakpoint

(0000ffff`0496d904)   libc_2_17!sleep
Exact matches:
    libc_2_17!sleep = <no type information>

0:000> dps 00000000`00420000 + 8
00000000`00420008  0000ffff`0496d904 libc_2_17!sleep
00000000`00420010  0000ffff`04a57fd0 libpthread_2_17!pthread_create
00000000`00420018  00000000`00400550 App1!$x
00000000`00420020  00000000`00400550 App1!$x
00000000`00420028  00000000`00000000
00000000`00420030  00000000`00000000
00000000`00420038  00000000`00000000
00000000`00420040  00000000`00000000
00000000`00420048  00000000`00000000
00000000`00420050  00000000`00000000
00000000`00420058  00000000`00000000
00000000`00420060  00000000`00000000
00000000`00420068  00000000`00000000
00000000`00420070  00000000`00000000
```

18. *App1.shared.pmap.22442* also shows library memory regions:

```
22442:   ./App1.shared
0000000000400000     64K r-x-- App1.shared
0000000000410000     64K r---- App1.shared
0000000000420000     64K rw--- App1.shared
0000000036a80000    192K rw---   [ anon ]
0000ffff02070000     64K -----   [ anon ]
0000ffff02080000   8192K rw---   [ anon ]
0000ffff02880000     64K -----   [ anon ]
0000ffff02890000   8192K rw---   [ anon ]
0000ffff03090000     64K -----   [ anon ]
0000ffff030a0000   8192K rw---   [ anon ]
```

```
0000ffff038a0000     64K -----   [ anon ]
0000ffff038b0000   8192K rw---   [ anon ]
0000ffff040b0000     64K -----   [ anon ]
0000ffff040c0000   8192K rw---   [ anon ]
0000ffff048c0000   1472K r-x-- libc-2.17.so
0000ffff04a30000     64K r---- libc-2.17.so
0000ffff04a40000     64K rw--- libc-2.17.so
0000ffff04a50000    128K r-x-- libpthread-2.17.so
0000ffff04a70000     64K r---- libpthread-2.17.so
0000ffff04a80000     64K rw--- libpthread-2.17.so
0000ffff04aa0000     64K r----   [ anon ]
0000ffff04ab0000     64K r-x--   [ anon ]
0000ffff04ac0000    128K r-x-- ld-2.17.so
0000ffff04ae0000     64K r---- ld-2.17.so
0000ffff04af0000     64K rw--- ld-2.17.so
0000ffffe2fc0000    192K rw---   [ stack ]
 total            44096K
```

Note: We can also see shared library mappings in the output of the **!address** command:

```
0:000> !address

Mapping file section regions...
Mapping module regions...

        BaseAddress      EndAddress+1        RegionSize     Type       State                 Protect             Usage
----------------------------------------------------------------------------------------------------------------------
+        0`00000000        0`00400000        0`00400000                                                       <unknown>
+        0`00400000        0`00410000        0`00010000 MEM_PRIVATE MEM_COMMIT  PAGE_EXECUTE_READ           Image      [App1;
"/home/opc/ALCDA2/App1/App1.shared"]
+        0`00410000        0`00420000        0`00010000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY              Image      [App1;
"/home/opc/ALCDA2/App1/App1.shared"]
+        0`00420000        0`00430000        0`00010000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE             Image      [App1;
"/home/opc/ALCDA2/App1/App1.shared"]
+        0`00430000        0`36a80000        0`36650000                                                       <unknown>
+        0`36a80000        0`36ab0000        0`00030000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE             <unknown>  [........Q.......]
+        0`36ab0000     ffff`02070000     fffe`cb5c0000                                                       <unknown>
+     ffff`02070000     ffff`02080000        0`00010000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY              <unknown>  [................]
+     ffff`02080000     ffff`02880000        0`00800000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE             <unknown>  [................]
+     ffff`02880000     ffff`02890000        0`00010000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY              <unknown>  [................]
+     ffff`02890000     ffff`03090000        0`00800000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE             <unknown>  [................]
+     ffff`03090000     ffff`030a0000        0`00010000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY              <unknown>  [................]
+     ffff`030a0000     ffff`038a0000        0`00800000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE             <unknown>  [................]
+     ffff`038a0000     ffff`038b0000        0`00010000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY              <unknown>  [................]
+     ffff`038b0000     ffff`040b0000        0`00800000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE             <unknown>  [................]
+     ffff`040b0000     ffff`040c0000        0`00010000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY              <unknown>  [................]
+     ffff`040c0000     ffff`048c0000        0`00800000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE             <unknown>  [................]
+     ffff`048c0000     ffff`04a30000        0`00170000 MEM_PRIVATE MEM_COMMIT  PAGE_EXECUTE_READ           Image      [libc_2_17; "/usr/lib64/libc-
2.17.so"]
+     ffff`04a30000     ffff`04a40000        0`00010000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY              Image      [libc_2_17; "/usr/lib64/libc-
2.17.so"]
+     ffff`04a40000     ffff`04a50000        0`00010000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE             Image      [libc_2_17; "/usr/lib64/libc-
2.17.so"]
+     ffff`04a50000     ffff`04a70000        0`00020000 MEM_PRIVATE MEM_COMMIT  PAGE_EXECUTE_READ           Image      [libpthread_2_17;
"/usr/lib64/libpthread-2.17.so"]
+     ffff`04a70000     ffff`04a80000        0`00010000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY              Image      [libpthread_2_17;
"/usr/lib64/libpthread-2.17.so"]
+     ffff`04a80000     ffff`04a90000        0`00010000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE             Image      [libpthread_2_17;
"/usr/lib64/libpthread-2.17.so"]
+     ffff`04a90000     ffff`04ab0000        0`00020000                                                       <unknown>
+     ffff`04ab0000     ffff`04ac0000        0`00010000 MEM_PRIVATE MEM_COMMIT  PAGE_EXECUTE_READ           Image      [linux_vdso_so; "linux-
vdso.so.1"]
+     ffff`04ac0000     ffff`04ae0000        0`00020000 MEM_PRIVATE MEM_COMMIT  PAGE_EXECUTE_READ           Image      [ld_2_17; "/usr/lib64/ld-
2.17.so"]
+     ffff`04ae0000     ffff`04af0000        0`00010000 MEM_PRIVATE MEM_COMMIT  PAGE_READONLY              Image      [ld_2_17; "/usr/lib64/ld-
2.17.so"]
+     ffff`04af0000     ffff`04b00000        0`00010000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE             Image      [ld_2_17; "/usr/lib64/ld-
2.17.so"]
+     ffff`04b00000     ffff`e2fc0000        0`de4c0000                                                       <unknown>
+     ffff`e2fc0000     ffff`e2ff0000        0`00030000 MEM_PRIVATE MEM_COMMIT  PAGE_READWRITE             <unknown>  [................]
```

19. We close logging before exiting WinDbg sessions:

```
0:000> .logclose
Closing open log file 'C:\ALCDA2\A64\App1\App1-WinDbg.log'
```

We recommend exiting WinDbg after each exercise.