



Zephyr[®] Project

Developer Summit

Adding Coredumps To Your Debugging Toolkit

Eric Johnson, *Memfault*
@ejohnso49

About Me:

- Firmware Solutions Engineer @
Memfault
- Previously: Walgreens Health, Athos,
Acuity Brands, Lexmark
- First encountered Zephyr at OSS 2018
- Met Tyler Hoffman at ZDS 2022
- Contributor of humble bug fixes to BLE,
Kernel, Shell subsystems



Memfault

- Coredumps Overview
- Coredumps + Zephyr
 - Zephyr Asserts and Fault Handling
 - Subsystem Components
 - Host Tools
 - Scripting
- Demo!
- Future Work

Logging is great, until...



```
[61:55:52.180,000] <err> sensor: Could not insert data into ring buffer
[61:55:52.280,000] <err> sensor: Could not insert data into ring buffer
[61:55:52.380,000] <err> sensor: Could not insert data into ring buffer
[61:55:52.480,000] <err> sensor: Could not insert data into ring buffer
[61:55:52.580,000] <err> sensor: Could not insert data into ring buffer
[61:55:52.680,000] <err> sensor: Could not insert data into ring buffer
[61:55:52.780,000] <err> sensor: Could not insert data into ring buffer
[61:55:52.880,000] <err> sensor: Could not insert data into ring buffer
[61:55:52.980,000] <err> sensor: Could not insert data into ring buffer
[61:55:53.080,000] <err> sensor: Could not insert data into ring buffer
[61:55:53.180,000] <err> sensor: Could not insert data into ring buffer
[61:55:53.280,000] <err> sensor: Could not insert data into ring buffer
[61:55:53.380,000] <err> sensor: Could not insert data into ring buffer
[61:55:53.480,000] <err> sensor: Could not insert data into ring buffer
[61:55:53.580,000] <err> sensor: Could not insert data into ring buffer
```

Panics are great, but only show 1 frame

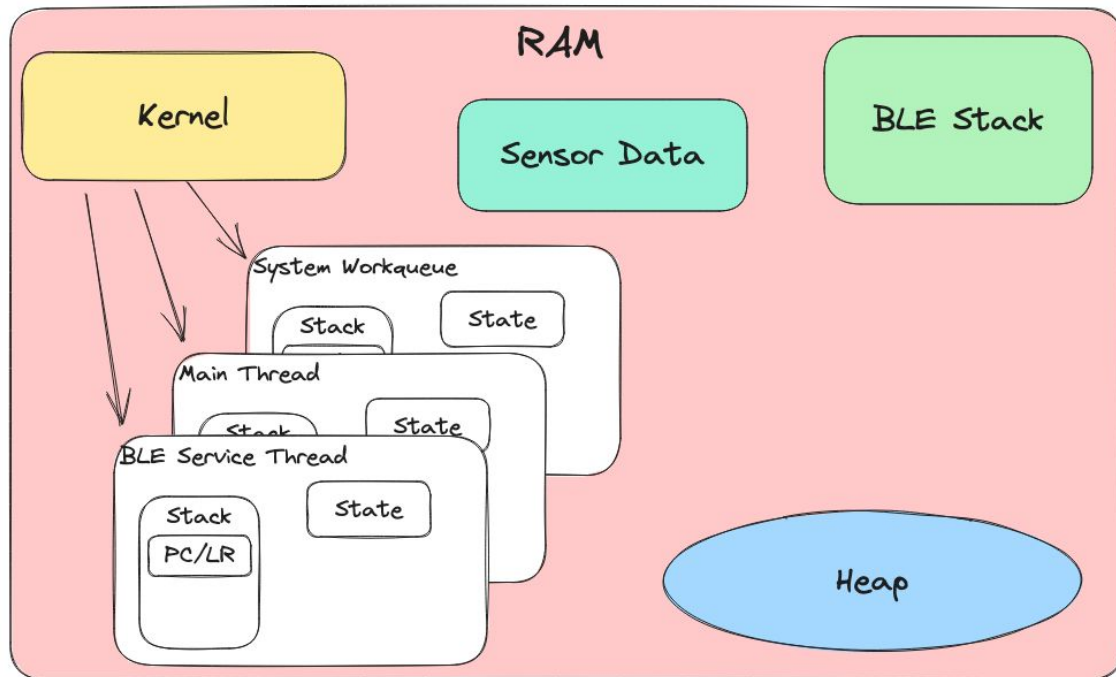


```
[70:52:25.480,000] <err> sensor: Could not insert data into ring buffer
ASSERTION FAIL [ret != size] @ WEST_TOPDIR/eoss-app/src/sensor.c:16
[70:52:25.480,000] <err> os: r0/a1: 0x00000004 r1/a2: 0x00000010 r2/a3: 0x00000002
[70:52:25.480,000] <err> os: r3/a4: 0x20000200 r12/ip: 0x200029c0 r14/lr: 0x00000407
[70:52:25.480,000] <err> os: xpsr: 0x4100000f
[70:52:25.480,000] <err> os: Faulting instruction address (r15/pc): 0x0000a13c
[70:52:25.480,000] <err> os: >>> ZEPHYR FATAL ERROR 4: Kernel panic on CPU 0
[70:52:25.480,000] <err> os: Fault during interrupt handling
```

Coredumps

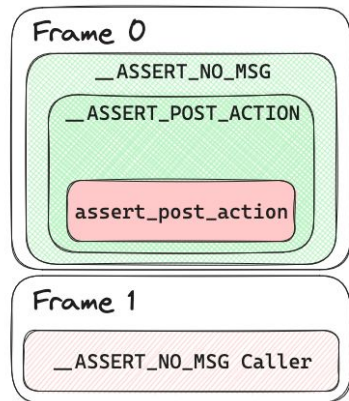
- Triggered by faults, kernel panics, asserts
- Captures registers and memory to allow for later analysis
- Data can be streamed out immediately or stored in non-volatile memory

Coredump Components



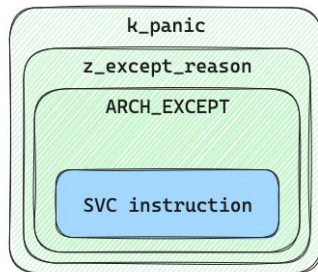
Zephyr Assertion Call Graph

Assertion Backtrace



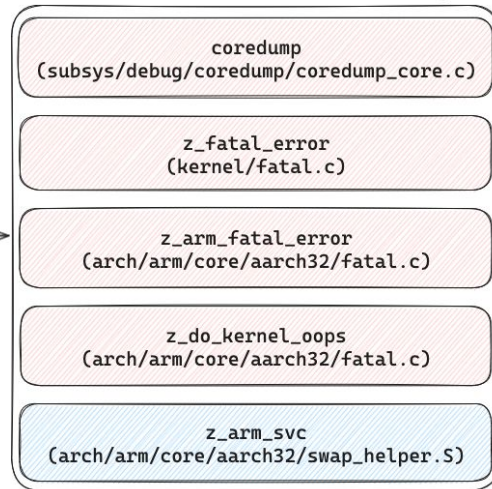
— Calls `k_panic` —>

Fault Context Initiation

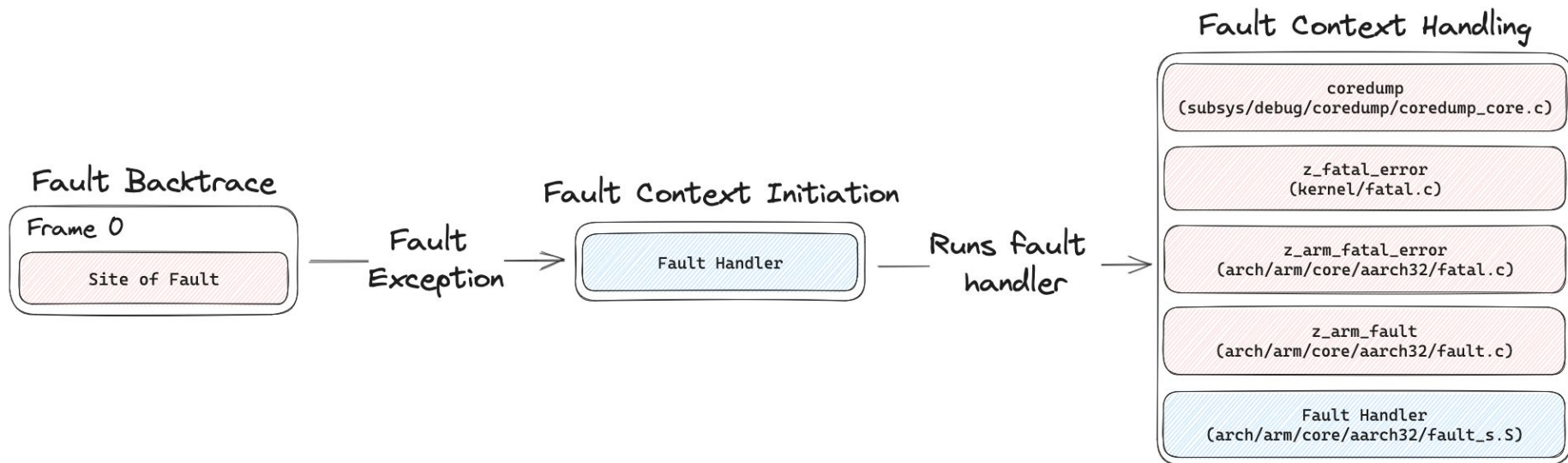


— SVC Interrupt —>

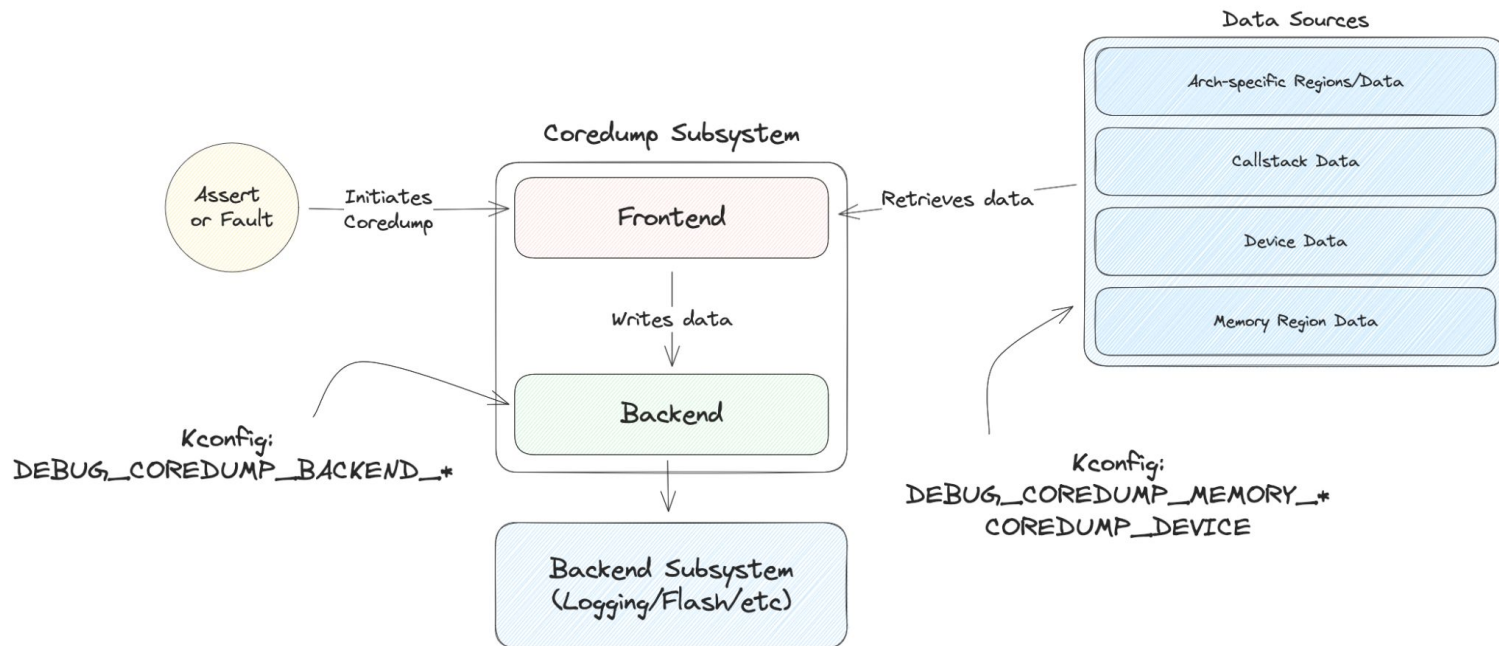
Fault Context Handling



Zephyr Fault Handling Call Graph



Coredump Subsystem



Device Region Example

```

/ {
    coredump_gpio: coredump-gpio {
        compatible = "zephyr,coredump";
        coredump-type = "COREDUMP_TYPE_MEMCPY";
        status = "okay";
        memory-regions = <0x40004000 0x1000>;
    };
};

```



```
uart:~$ sensor enable
```

```
[70:52:25.480,000] <err> sensor: Could not insert data into ring buffer
```

```
ASSERTION FAIL [ret != size] @ WEST_TOPDIR/eoss-app/src/sensor.c:16
```

```
[70:52:25.480,000] <err> os: r0/a1: 0x00000004 r1/a2: 0x00000010 r2/a3: 0x00000002
```

```
[70:52:25.480,000] <err> os: r3/a4: 0x20000200
```

```
[70:52:25.480,000] <err> os: xpsr: 0x4100000f
```

```
[70:52:25.480,000] <err> os: Faulting instruction address (r15/pc): 0x0000a13c
```

```
[70:52:25.480,000] <err> os: >>> ZEPHYR FATAL ERROR 4: Kernel panic on CPU 0
```

```
[70:52:25.480,000] <err> os: Fault during interrupt handling
```

```
[70:52:25.480,000] <err> os: Current thread: 0x2000009e8 (idle)
```

```
[70:52:25.480.000] <err> coredump: #CD:BEGIN#
```

```
[70:52:25.480,000] <err> coredump: #CD:5a4501000300050004000000
```

```
[70:52:25.480,000] <err> coredump: #CD:4102004400
```

```
[70:52:25.480,000] <err> coredump: #CD:0400000001000000000200000000020020c0290020070400003ca1000000f000041
```

[illegible]

```
[70:52:25.480.000] <err> coredump: #CD:00000000
```

```
[70:52:25.480.000] <err> coredump: #CD:4d01000000002088390020
```

```
[70:52:25.480,000] <err> coredump: #CD:a5b92dedc233c34c403500200400000000400000000000000000000000000000
```

[illegible]

```
[70:52:25.480.000] <err> coredump: #CD:0100000001000000e9660000d7a30000000000000010000006800002040330020
```

```
[70:52:25.480.000] <err> coredump: #CD:403300200000000000403300201c00000001c0000000000000000000000000000000
```

```
[70:52:25.480.000] <err> coredump: #CD:00000000000020000000cc80000074c800000403900200e00000000e000000000000000
```

```
[70:52:25.480.000] <err> coredump: #CD:0e00000000e000000000000000400000000803900209400000009400000009000000000
```

```
[70:52:25.480.000] <err> coredump: #CD:9400000009400000009000000008000000e966000000000020883900200000000000
```

```
[70:52:25.480.000] <err> coredump: #CD:00000000a1640000a9650000e5640000316400004564000000c201000000000000
```

```
[70:52:25 480 000] <err> coredump: #CD:000000000000c301000000000000000000d00010001bb700b3bd0000000c30100
```

```
[70:52:25:480,000] <err> coredump: #CD:00000000000c20100000000000000000d000040001bb7900b3bd0000000c20100
```

```
[70:52:25.480,000] <err> coredump: #CD:000200200000100004e00000000c0100200c010020010000000010000000;c010020
```

```
[70:52:25.480,000] <err> coredump: #CD:7c01002000000000001b000000001b000000001b000000001b000000001b00000000
```

```
[70:52:25.480,000] <err> coredump: #CD:00a00000000000000000c80010020c8010020475e00000000000000a000000000000000
```

```
[70:52:25.480,000] <elf> coredump: #CD:010000000000000000000000z00z00000z00z0000000000001000000f80100z0f80100z0
```

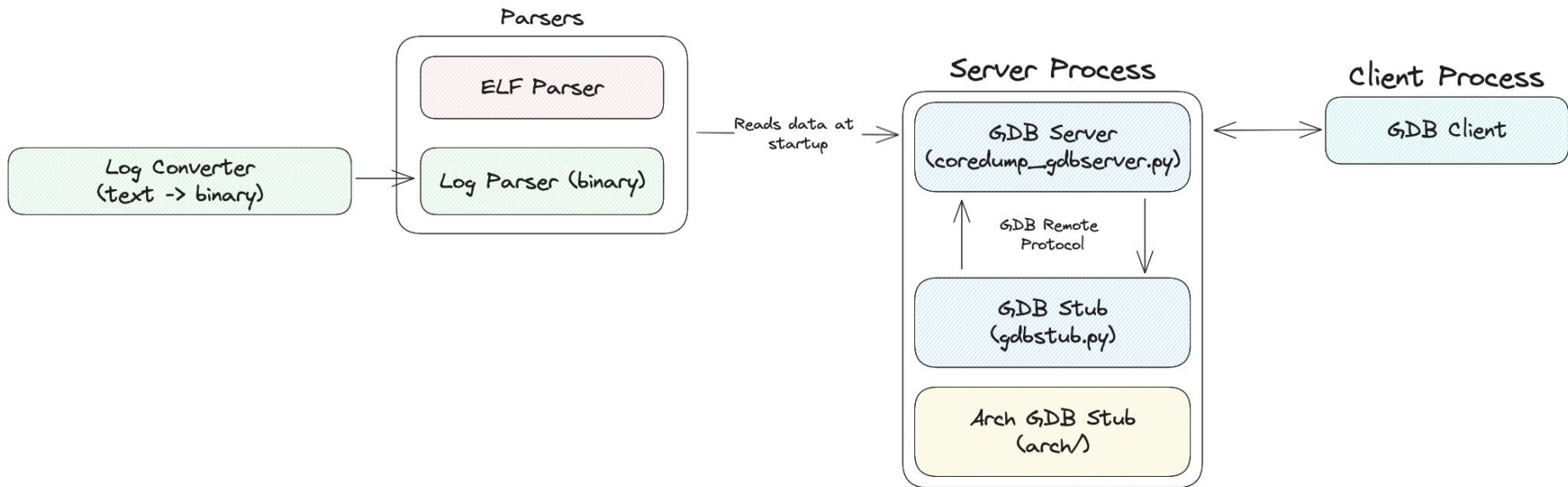
[illegible]

```
[70:32:25.480,000] <err> coredump: #cd:ffffffffffffe8010020000000000000d81900200000000003802002038020020
```

© 2014 Pearson Education, Inc. or its affiliate(s). All rights reserved. Pearson Education, Inc., publishing as Pearson Benjamin Cummings, 101 University Avenue, New York, NY 10017-2423.

#EMBEDDEDOSSUMMIT

Coredump Host Tools



Scripting Extensions

- GDB can be built with Python extension support
 - Zephyr toolchain defaults to no-py version
 - Use Python with “-py”
- Requires matching system Python install
- Use venv + [gdbundle](#) to manage packages

Demo!

Future Work

- ISR-safe Flash backed: See [issue](#)
- Allow application to override linker memory region
- Encourage users to submit coredumps with bugs/issues
- Add example using flash simulation region for backend
- Extend west to convert log output and run coredump server

Where To Find Me

- [linkedin.com/in/ejohnso49/](https://www.linkedin.com/in/ejohnso49/)
- Github: [ejohnso49](https://github.com/ejohnso49)
- interrupt.memfault.com

Coredumps in Memfault

▼ accel-workq (2) STACK OVERFLOW RUNNING

▶ 0 compute_fft in .../src/fft.c at line 10

▶ 1 sleep_algo_compute_sleep_time in .../src/sleep_algo.c at line 12

▶ 2 process_accel_data_worker_task in .../src/accel_data.c at line 106

▶ 3 z_work_q_main in .../zephyr/lib/os/work_q.c at line 32

▶ 4 z_thread_entry in .../lib/os/thread_entry.c at line 29

▶ 5 0xaaaaaaaa

▶ Thread 3 SUSPENDED

▶ idle (4) READY

▶ logging (5) SUSPENDED

▶ net_mgmt (6) BLOCKED

▶ rx_workq (7) BLOCKED

▶ shell_uart (8) BLOCKED

▶ sysworkq (9) BLOCKED

▶ tx_workq (10) BLOCKED

▶ workqueue (11) BLOCKED

A dft_out = 0x2000a900 <my_stack_area+1344>

L i = 400

A num_samples = 536912536

A raw_samples = 0x3128115f

L tmp = {1, 222, 7, 84}

R \$r0 = long 536912536 (0x2000a298)

R \$r1 = long 1372324912 (0x51cc0430)

R \$r2 = long 1372324919 (0x51cc0437)

R \$r3 = long 536912832 (0x2000a3c0)

R \$r4 = long 536912508 (0x2000a27c)

R \$r5 = long 536914136 (0x2000a8d8)

R \$r6 = long 0 (0x00000000)

R \$r7 = long 536912488 (0x2000a268)

R \$r8 = long 0 (0x00000000)

R \$r9 = long 0 (0x00000000)

R \$r10 = long 0 (0x00000000)

View tasks, stack traces, registers, and local variables

Coredumps in Memfault

The screenshot displays the Memfault debugger interface. On the left, a list of global variables is shown, including `heap_sz`, `_mbedtls_heap`, `heap`, `buf`, `len`, `first`, `first_free`, `*`, `magic1`, `size`, `alloc`, `prev`, `next`, `prev_free`, `next_free`, `magic2`, `verify`, and `heap_sem`. Each variable is associated with a memory address. The right pane shows a memory dump with addresses ranging from `0x20009f58` to `0x20009ff8`. The dump includes hexadecimal values and ASCII representations, such as `ec310020` and `.1.` in the first row, and `69646c65` and `idle` in the 11th row.

Variable	Address
<code>heap_sz = unsigned int 0</code>	@ 0x20002378
<code>▶ _mbedtls_heap = unsigned char[28000] {...}</code>	@ 0x200031ec
<code>▼ heap = buffer_alloc_ctx {...}</code>	@ 0x20009f4c
<code>▶ buf = unsigned char* {...}</code>	@ 0x20009f4c
<code>len = size_t 28000</code>	@ 0x20009f50
<code>▶ first = memory_header* {...}</code>	@ 0x20009f54
<code>▼ first_free = memory_header* {...}</code>	@ 0x20009f58
<code>▼ * = memory_header {...}</code>	@ 0x200031ec
<code>magic1 = size_t 4278233685</code>	@ 0x200031ec
<code>size = size_t 27968</code>	@ 0x200031f0
<code>alloc = size_t 0</code>	@ 0x200031f4
<code>▶ prev = memory_header* {...}</code>	@ 0x200031f8
<code>▶ next = memory_header* {...}</code>	@ 0x200031fc
<code>▶ prev_free = memory_header* {...}</code>	@ 0x20003200
<code>▶ next_free = memory_header* {...}</code>	@ 0x20003204
<code>magic2 = size_t 3994130790</code>	@ 0x20003208
<code>verify = int 0</code>	@ 0x20009f5c
<code>▶ heap_sem = sys_sem {...}</code>	@ 0x2001166c

Address	Hex	ASCII
0x20009f58	ec310020 00000000	.1.
0x20009f60	00000000 00000000
0x20009f68	00000000 00000000
0x20009f70	00000000 00000000
0x20009f78	00000000 00000000
0x20009f80	00000000 00000000
0x20009f88	01000f00 00000000
0x20009f90	00000000 00000000
0x20009f98	00000000 00000000
0x20009fa0	00000000 adc20208
0x20009fa8	00000000 00000000
0x20009fb0	00000000 00000000
0x20009fb8	00000000 00000000
0x20009fc0	00000000 48e20020 H..
0x20009fc8	00000000 00000000
0x20009fd0	adc20208 00000000
0x20009fd8	00000000 00000000
0x20009fe0	00000000 69646c65 idle
0x20009fe8	00000000 00000000
0x20009ff0	00000000 00000000
0x20009ff8	00000000 00000000

View all global variables at time of coredump

Acknowledgements:

- Coredump subsystem maintainers and contributors
- Coredump: A brief introduction and demo by Daniel Leung



- <https://github.com/memfault/gdbundle>
- <https://github.com/ejohnso49/gdbundle-gpio>