Memfault

Utilizing Memfault to Debug Your Embedded Devices

Eric Johnson
Noah Pendleton

Today's Presenters

Eric Johnson

Firmware Solutions Engineer, Memfault

/in/ejohnso49

interrupt.memfault.com

Noah Pendleton

Firmware Solutions Engineer, Memfault

@nlpendleton

/noah-pendleton-71437843

interrupt.memfault.com

Agenda

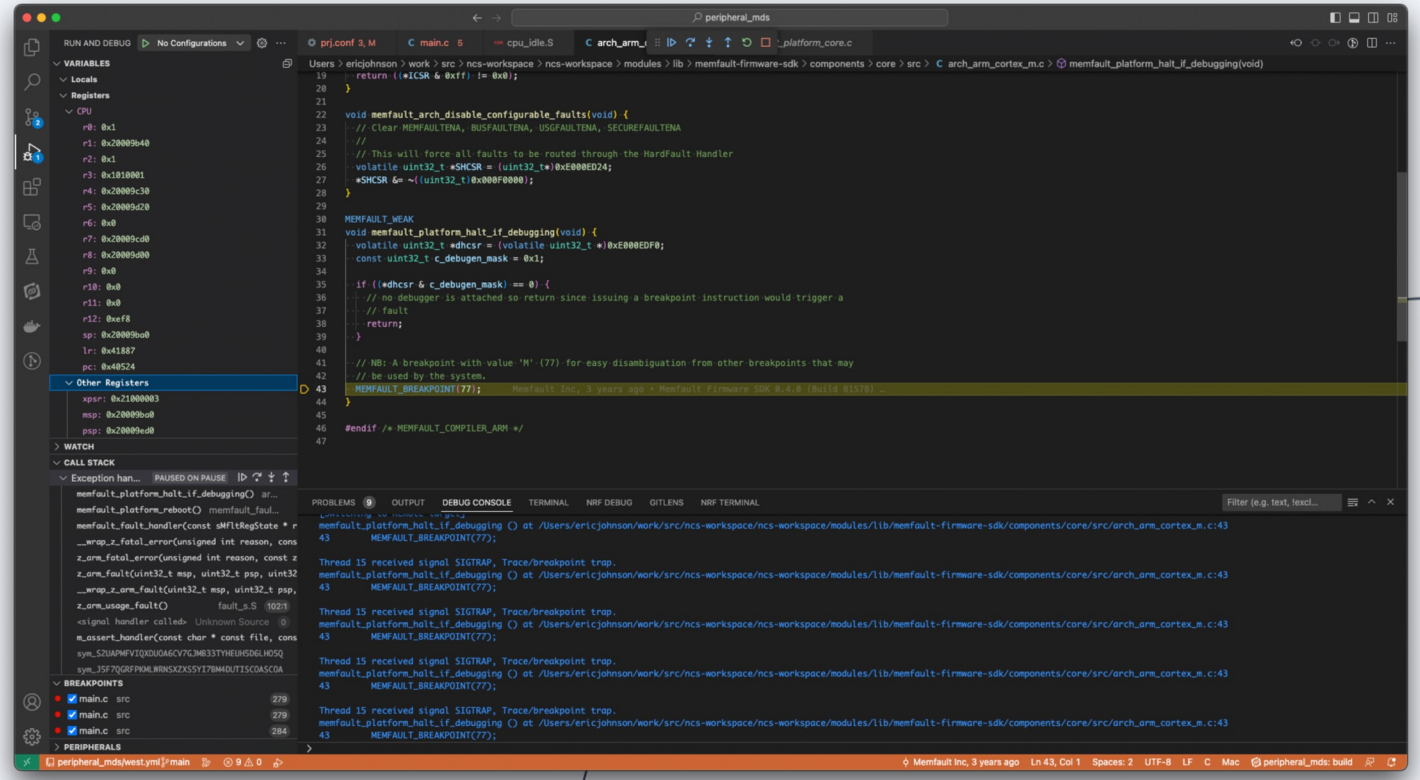
1. Traditional Debugging Workflow
2. Memfault to the Rescue
3. Going Deeper- Memory and Watchdogs
4. Offensive Programming with Memfault

1. The Traditional Debugging Workflow

Spending some quality time with haunted software

Local Debugging

- Great for stepping through difficult problems at your desk
- Full visibility into device state



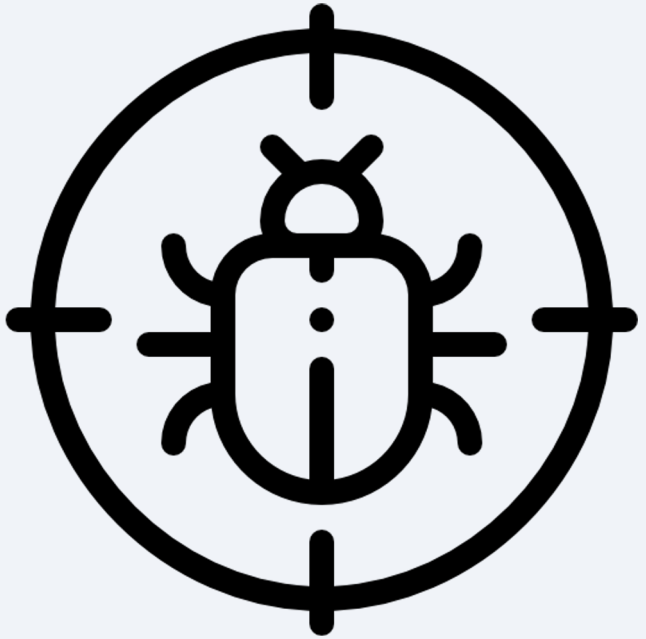
Logging

- Print information about the run-time state of the device
- Simple as LOG_INF and connect with your favorite serial terminal

```
*** Booting Zephyr OS build v3.2.99-ncs1 ***
[00:00:00.001,525] <inf> mflt: Reset Reason, RESETREAS=0x4
[00:00:00.001,586] <inf> mflt: Reset Causes:
[00:00:00.001,617] <inf> mflt:  Software
[00:00:00.002,471] <inf> mflt: GNU Build ID: 98347d7adcd877e05162217bd67fc01d9202d257
Starting Bluetooth Memfault example
[00:00:00.009,033] <inf> fs_nvs: 6 Sectors of 4096 bytes
[00:00:00.009,033] <inf> fs_nvs: alloc wra: 0, f80
[00:00:00.009,063] <inf> fs_nvs: data wra: 0, d0
[00:00:00.009,765] <inf> sdc_hci_driver: SoftDevice Controller build revision:
                                         6d 90 41 2a 38 e8 ad 17  29 a5 03 38 39 27 d7 85 |m.A*8... )..89'..
                                         1f 85 d8 e1                                     |....
[00:00:00.014,709] <inf> bt_hci_core: No ID address. App must call settings_load()
Bluetooth initialized
Advertising successfully started
[00:00:31.760,375] <inf> bas: BAS Notifications enabled
Connected 24:29:34:82:4B:95 (public)
[00:00:32.263,061] <dbg> mds: data_export_ccc_changed: MDS Data Export CCCD changed, handle: 31, value: 0x0001
Security changed: 24:29:34:82:4B:95 (public) level 2
[00:00:33.612,945] <dbg> mds: device_identifier_read: MDS Device Identifier characteristic read, handle: 24, conn: 0x20002610
[00:00:33.747,955] <dbg> mds: data_uri_read: MDS Data URI characteristic read, handle: 26, conn: 0x20002610
[00:00:33.837,982] <dbg> mds: authorization_read: MDS Authorization characteristic read, handle: 28, conn: 0x20002610
[00:00:34.017,944] <dbg> mds: data_export_write: MDS Data Export characteristic write, handle 30, conn: 0x20002610
[00:00:34.032,836] <dbg> mds: mds_data_send: Memfault diagnostic data chunk 0 successfully sent
[00:00:50.082,763] <wrn> bt_att: Not connected
[00:00:50.082,824] <wrn> mds: Failed to send Memfault diagnostic chunk, err -128
[00:00:50.095,520] <wrn> bt_att: Not connected
[00:00:50.095,550] <err> mds: MTU value too low: 0 or link is disconnected
[00:00:50.095,672] <err> mflt: Buffer of 0 bytes too small to packetize data
[00:00:50.095,703] <wrn> mds: Failed to send Memfault diagnostic chunk, err -128
[00:00:50.108,459] <inf> bas: BAS Notifications disabled
[00:00:50.108,489] <dbg> mds: data_export_ccc_changed: MDS Data Export CCCD changed, handle: 31, value: 0x0000
Disconnected (reason 19)
uart:~$
```

Logging & Local Debug Deficiencies

- Why these methods fall short:
  - No physical access to UART/Debug port
  - Log analysis is difficult and inefficient
  - Invasive to system behavior



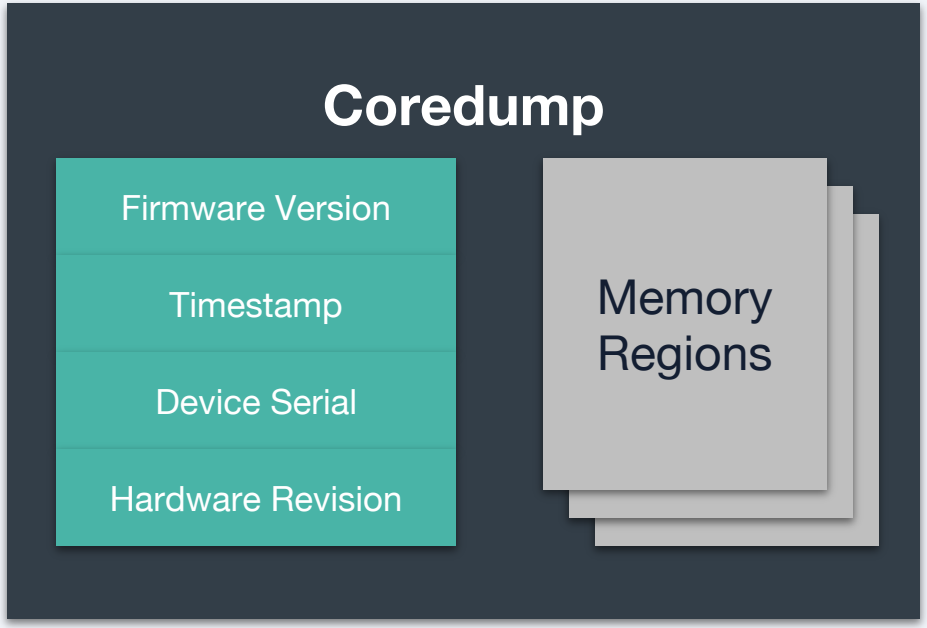
2. Memfault to the Rescue

Software should self-report problems!

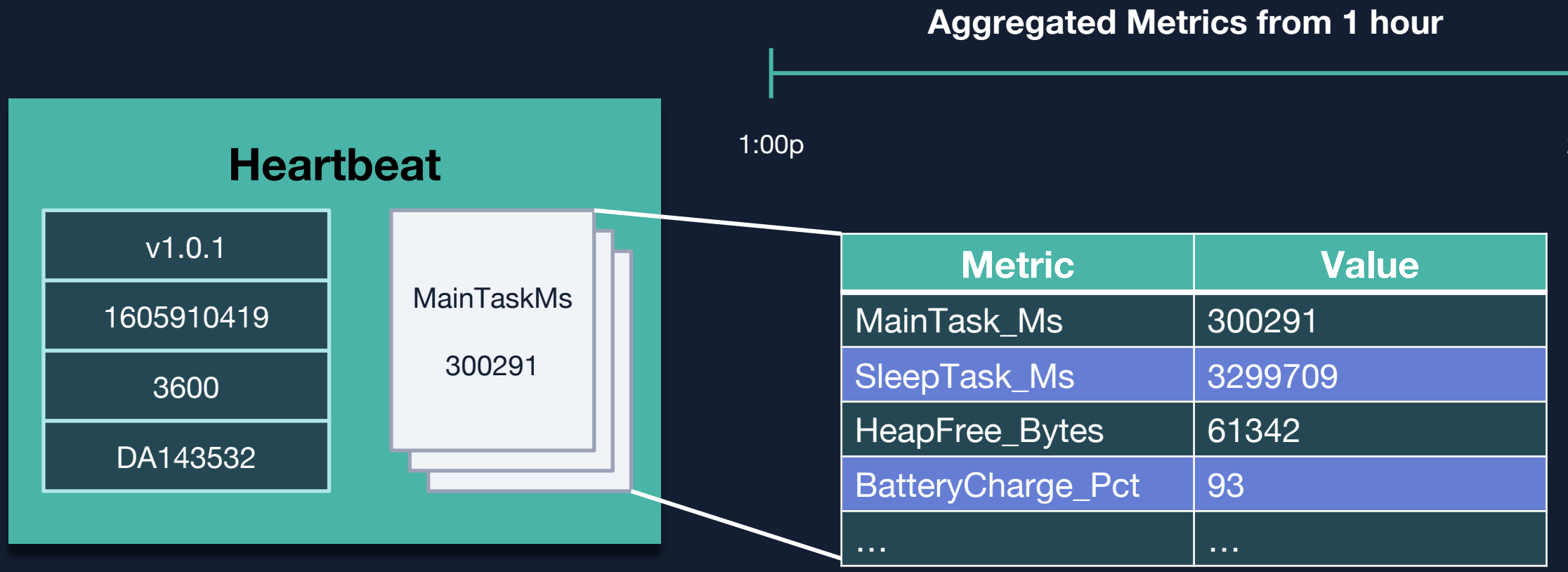
What is a core dump?

```
void HardFault_Handler(void) {
    // Capture core dump FIRST
    capture_coredump();

    // Then reset
    NVIC_SystemReset();
}
```



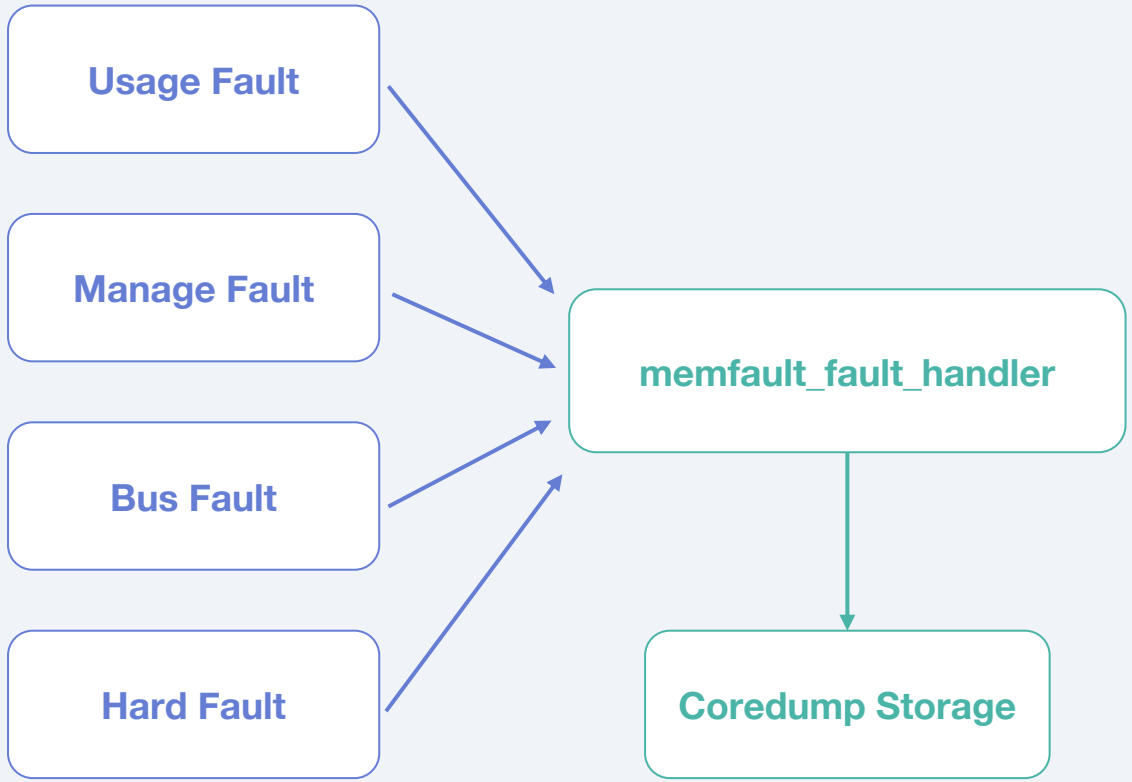
A Metric Heartbeat



https://interrupt.memfault.com/blog/device-heartbeat-metrics

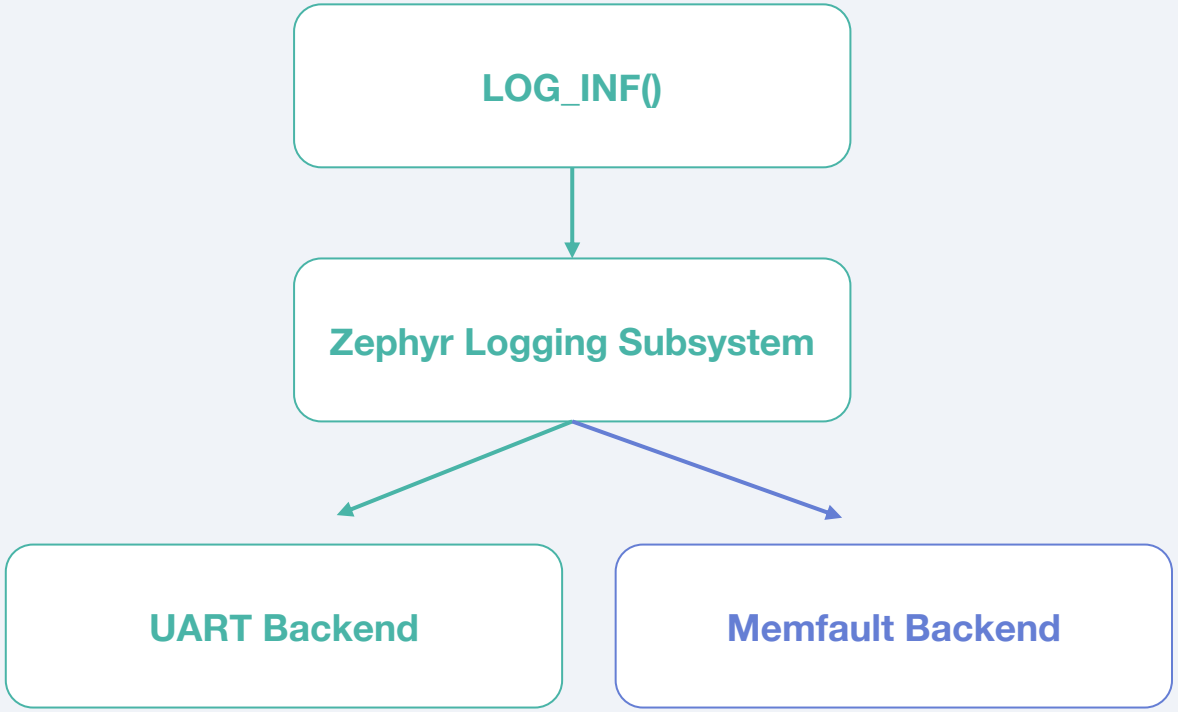
MCU SDK Integration

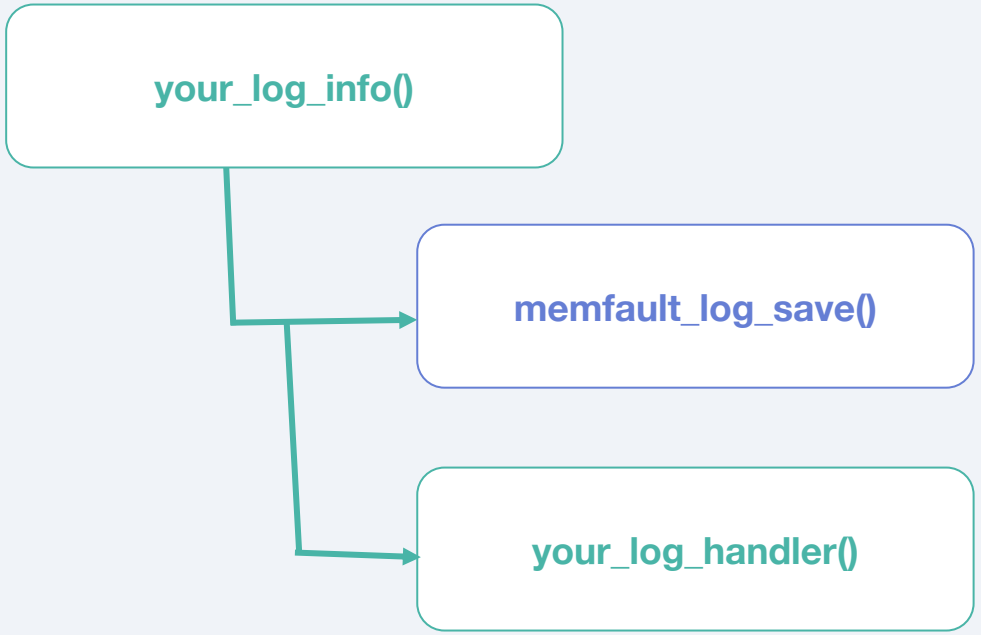
- Coredump Integration
  - Implement ARM fault handlers
  - Save coredump to specified storage (typically Flash)
  - Next reboot, our SDK provides chunks to send to Memfault



MCU SDK Integration

- Logging Integration
  - Register/hook into logging subsystem
  - Implement wrapper to call into SDK and your logging





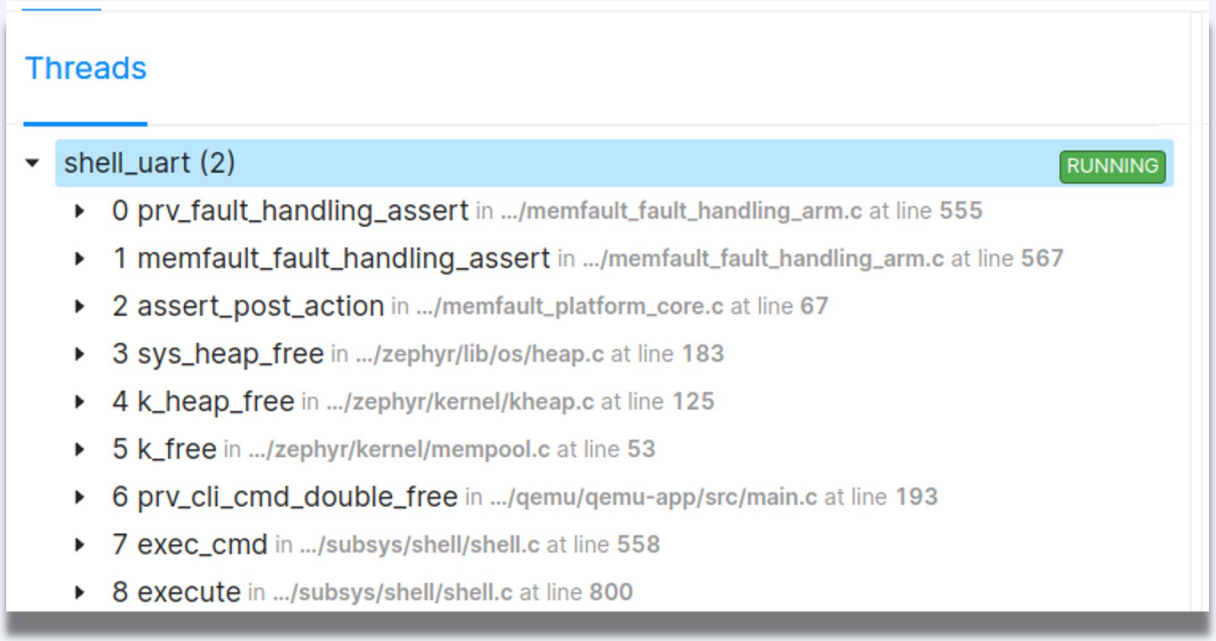
Demos!

Offensive Programming: assert()

Memory leaks!

```
void *malloc_assert(size_t n) {
  void *p = malloc(n)
  ASSERT(p);
  return p;
  ...
}
```

Double free



https://go.memfault.com/proactive-debugging-with-offensive-programming-webinar

THANK YOU!