Building a test for diverse fleet of platforms - how hard that could be?

Evgeniy Paltsev
senior software engineer

Zephyr Developer Summit, 2022

Evgeniy Paltsev

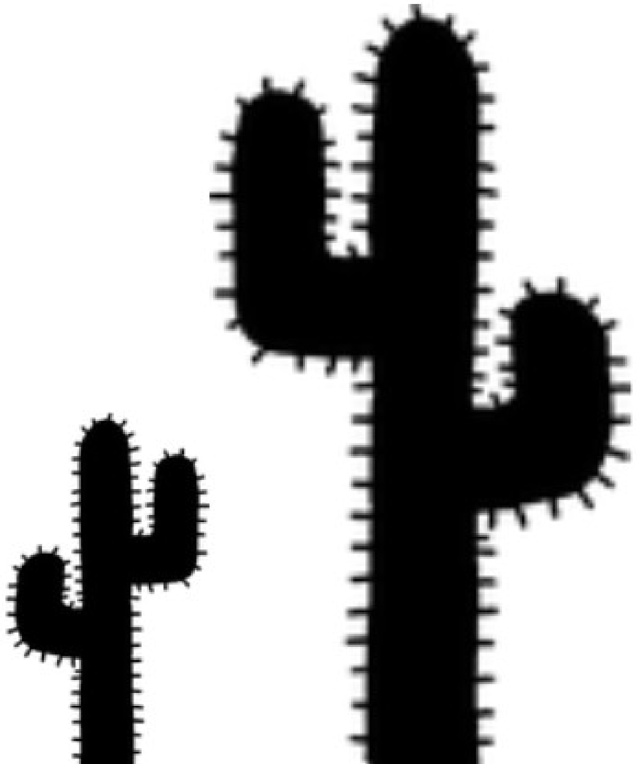
- Develop, port and maintain OSS projects for Synopsys ARC processors architecture
- Maintainer of ARC architecture in Zephyr RTOS
- Main focus - Zephyr RTOS, Linux kernel, U-Boot

Zephyr - diverse test environment

- Wide range of platforms with various ISAs, compilers, target speeds, memory layouts, SW features selection
- Creating a robust test is challenging
- We’ll discuss common test issues

Multiprocessing

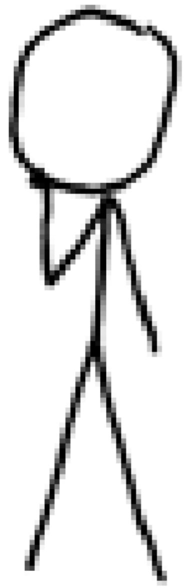
- Lots of zephyr kernel tests assume interaction between threads
- Zephyr supports SMP (since v1.11.0, 2018)



Multiprocessing

If you don't plan to support test for SMP

- ztest_1cpu_unit_test – lock all cores except one
- CONFIG_MP_NUM_CPUS=1 – fake SMP system with 1 core



Multiprocessing

If you plan to support test for SMP

- Prefer synchronization primitives to k_sleep when waiting for event
- Prefer synchronization primitives and atomic variables to global variables

Timeouts in tests

Existing tests use timeouts a lot

```
git grep '\bK_CYC\b|\bK_TICKS\b|\b
K_SECONDS \b|\bK_MSEC\b|\b
K_USEC\b' tests/ | wc -l

1055
```

There were lots of timeouts fixes / tweaks

Timeouts in tests

- Various timeout units - seconds-based, ticks, cycles
- Value may look valid in one unit and be incorrect in another

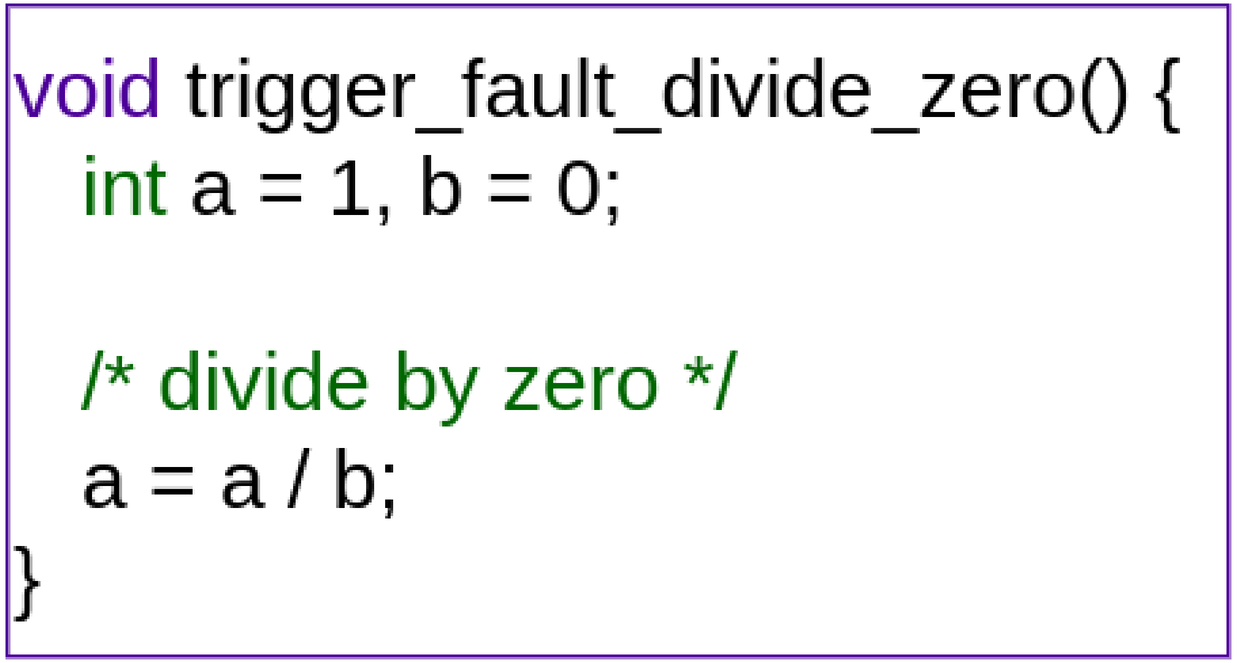
Timeouts in tests

Possible solution?

- Check why do we need timeout
- Prefer synchronization primitives to k_sleep when waiting for event

Undefined behavior in tests

- Bad idea in general
- Some tests do it intentionally
- E.g: divide by zero test in tests/ztest/error_hook – the division was removed by GCC



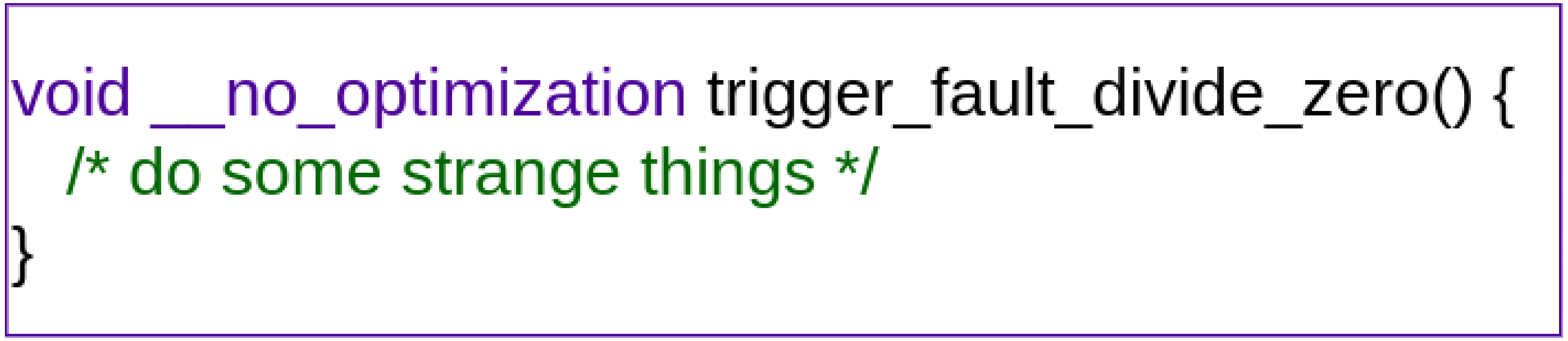
```c
void trigger_fault_divide_zero() {
   int a = 1, b = 0;

   /* divide by zero */
   a = a / b;
}
```

Undefined behavior in tests

Possible solution?

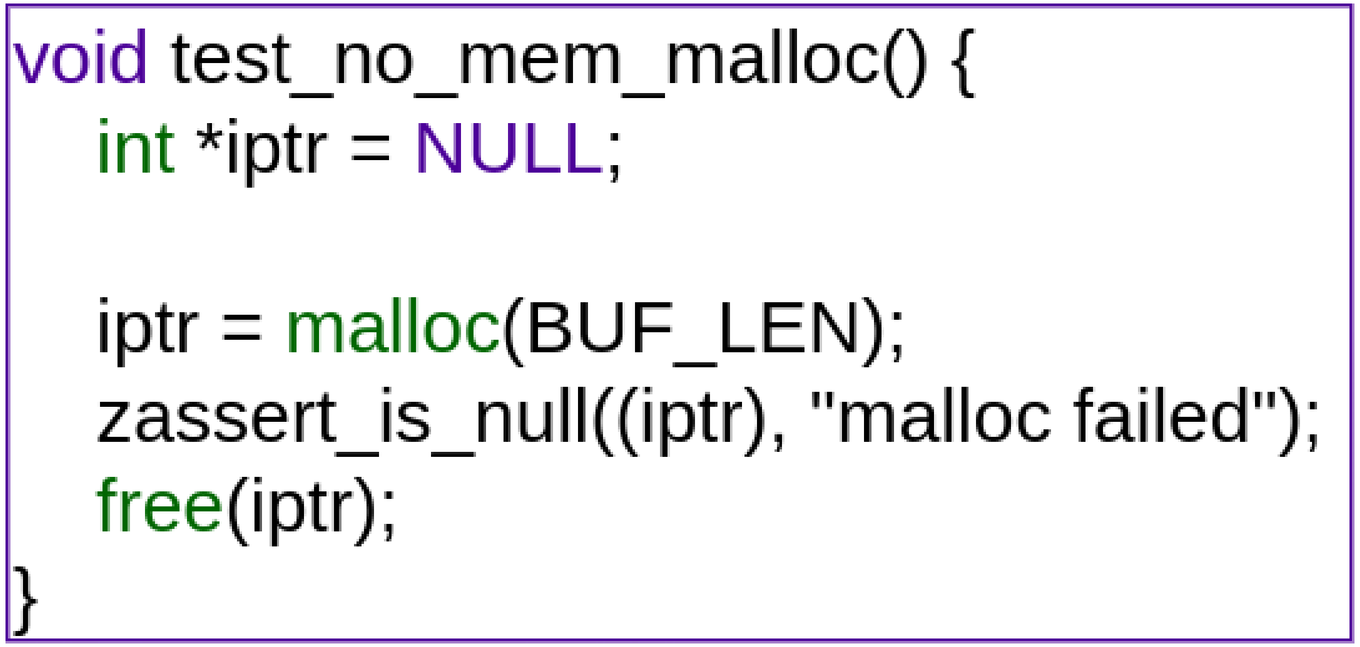
- Disable optimization for function with __no_optimization



```
void __no_optimization trigger_fault_divide_zero() {
   /* do some strange things */
}
```

Compiler optimizations

- Test may break due to aggressive compiler optimizations
- E.g: tests/lib/mem_alloc test case - what can go wrong here?

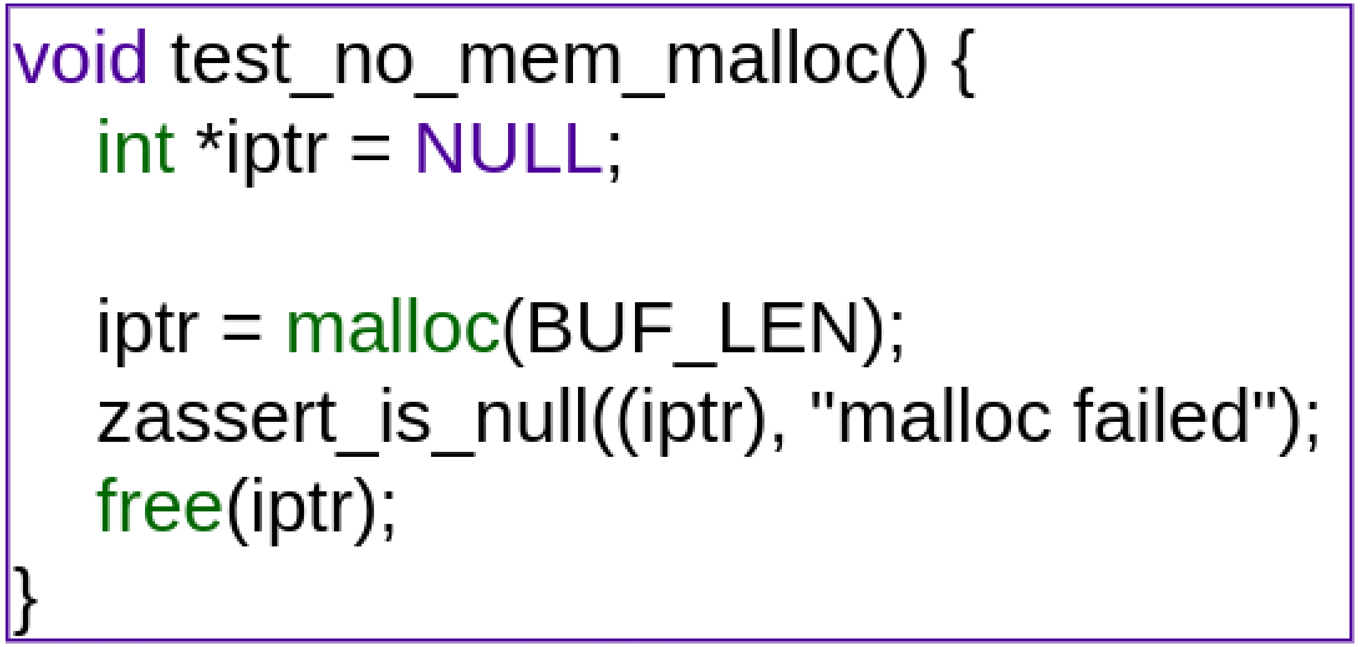


```
void test_no_mem_malloc() {
    int *iptr = NULL;

    iptr = malloc(BUF_LEN);
    zassert_is_null((iptr), "malloc failed");
    free(iptr);
}
```

Compiler optimizations

- The malloc & free pair was removed by compiler
- Compiler can do lots of such tricks



```
void test_no_mem_malloc() {
    int *iptr = NULL;

    iptr = malloc(BUF_LEN);
    zassert_is_null((iptr), "malloc failed");
    free(iptr);
}
```

Compiler optimizations

Possible solution?

- Disable optimization for function with __no_optimization
- Disable variable access optimization with volatile

What else?

- Virtual memory
- Userspace
- Various C libraries implementations
- …?

Thanks!

Questions?

Contacts

Evgeniy Paltsev

Eugeniy.Paltsev@synopsys.com
PaltsevEvgeniy@gmail.com

https://www.linkedin.com/in/evgeniy-paltsev

Images licensing notes