RMM unit test framework
Table of contents

- Fake_host architecture for RMM
  - fake_host harness mechanism
  - sysreg access emulation
  - Build & Run

- Unittest framework on RMM
  - Test example
  - Example of use
  - Coverage Report
arm

TF-RMM fake_host architecture
Fake_host architecture for RMM

- Builds and runs the program natively
- Facilitates development of architecture independent parts of RMM on the host
- Not intended to support multi-thread
- Cannot execute AArch64 assembly
- Emulation for AArch64 exceptions is limited
- libc implementation in RMM cannot be verified with this architecture.
host platform components

- **plat/** contains subdirectories for all the different platforms.
  - **fvp/** & **host/**
  - **common/** contains common components for all the platforms, such as platform setup.
- **host/** is the home for the fake_host architecture. It can support different variants
fake_host harness mechanism
fake_host harness mechanism

```c
/*
 * SPDX-License-Identifier: BSD-3-Clause
 * SPDX-FileCopyrightText: Copyright TF-RMM Contributors.
 */

#ifndef SLOT_BUF_ARCH_H
#define SLOT_BUF_ARCH_H

#include <host_harness.h>

static void *buffer_arch_map(enum buffer_slot slot, unsigned long addr, bool ns)
{
    return host_buffer_arch_map(slot, addr, ns);
}

static void buffer_arch_unmap(void *buf)
{
    return host_buffer_arch_unmap(buf);
}
#endif /* SLOT_BUF_ARCH_H */

/*
 * SPDX-License-Identifier: BSD-3-Clause
 * SPDX-FileCopyrightText: Copyright TF-RMM Contributors.
 */

#ifndef SLOT_BUF_ARCH_H
#define SLOT_BUF_ARCH_H

#define buffer_arch_map __attribute__((visibility("default"))) int (*buffer_arch_map)(void *buf)
#define buffer_arch_unmap __attribute__((visibility("default"))) int (*buffer_arch_unmap)(void *buf)
#endif /* SLOT_BUF_ARCH_H */
```

fake_host  

fvp
Sysreg access emulation on fake_host

Callbacks for sysreg access

- Provides support to install callback functions that will be called upon sysreg access
- Separate callbacks for read and write access
- Ability to configure a default (reset) value for the register.

```c
/*
 * Setup callbacks for sysreg read and write operations.
 *
 * This API allows to setup callbacks for each sysreg to be called upon read or write operations. This allows to control what to return on a read or how to process a write.
 *
 * Arguments:
 * name - String containing the name of the sysreg. The name of the sysreg cannot exceed MAX_SYSREG_NAME_LEN (excluding the terminating null character) or it will be truncated.
 * rd_cb - Callback to be invoked on a read operation.
 * wr_cb - Callback to be invoked on a write operation.
 * init - Value used as reset value for the sysreg.
 *
 * Returns:
 * 0 on success or a negative error code otherwise.
 */
int host_util_set_sysreg_cb(scher *name, rd_cb t rd_cb, wr_cb t wr_cb, u_register_t init);
*/

/*
 * Performs some initialization needed before RMM can be ran, such as setting up callbacks for sysreg access.
 */
static void setup_sysreg_and_boot_manifest(void)
{
    /* Initialize ID.AA64MMFR0_EL1 with a physical address
 * range of 48 bits (PARange bits set to 0b0101)
 */
    (void)host_util_set_default_sysreg_cb("id aa64mmfr0 ell",
        INPLACE(ID_AAA64MMFR0_EL1_PARANGE, SUL));

    /* Initialize ICH VTR EL2 with 6 preemption bits.
 * (PREbits is equal number of preemption bits minus one)
 */
    (void)host_util_set_default_sysreg_cb("ich vtr el2",
        INPLACE(ICH_VTR_EL2_PRE_BITS, SUL));

    /* SCTLR EL2 is reset to zero */
    (void)host_util_set_default_sysreg_cb("sctlr el2", 0UL);

    /* Initialize the boot manifest */
    boot_manifest->version = RMM_EL3_IFC_SUPPORTED_VERSION;
    boot_manifest->plat_data = (uintptr_t)NULL;
}
Sysreg access emulation on fake_host

Callbacks for sysreg access

```
/* Define read & write function for system register */
#define DEFINE_SYSREG_RW_FUNCS(name)
    DEFINE_SYSREG_READ_FUNC (name, name)
    DEFINE_SYSREG_WRITE_FUNC (name, name)

#define DEFINE_SYSREG_READ_FUNC (name, _reg_name)
static inline u64 register_read (# name)(void)
{
    return host_read_sysreg(# name);
}

#define DEFINE_SYSREG_WRITE_FUNC (name, _reg_name)
static inline void register_write (# name, val)(void)
{
    return host_write_sysreg(# name, val);
}
```

See host_utils.h and host_utils.c
Fake_host build and run
TF-RMM Unit Test Framework
Unittest framework on RMM

fake_host platform

host_test variant

Unit test helpers

CMake Test support

Test harness

CppUTest

TF-RMM Unit Test Framework

Unit Test

Component under test
test utils
Test harness
Unittest framework on RMM
Components: CppUtest Framework

- C/C++ base unit xUnit test framework for unit testing
  - Simple in design and, more importantly, in use
  - Portable
  - Built with test driven mentality for test driven development
  - Added as a git submodule. Automatically linked when the host_test variant is chosen.
  - For more information: http://cpputest.github.io/
Unit test framework on RMM

Components: test_helpers.h

- Collection of APIs to assist with the tests.
  - Very limited at the moment, can be expanded as needed

```c
/*
 * SPDX-License-Identifier: BSD-3-Clause
 * SPDX-FileCopyrightText: Copyright TF-RMM Contributors.
 */

#ifndef TEST_HELPERS_H
#define TEST_HELPERS_H

void test_platform_setup(bool secondaries);
unsigned int test_get_platform_nr_granules(void);

#endif
```
Test example
Granule test module

- Include a "tests" directory to hold all the tests for a given library along with ancillary sources.
- granule.c is part of lib/realm (src/granule.c), therefore granule.cpp will be inside the "tests" directory for lib/realm
  • We are only interested in the public API for this component
- CMakeLists.txt needs to add support to build and run the tests
Test example
Granule test module
Test example
Granule test module

```c
TEST_GROUP(granule) {
  TEST_SETUP()
  {
    static int random_seed = 0;
    if (random_seed == 0) {
      /* Enable the platform with support for multiple PEs */
      test_platform_setup(true);
    }
    /* Make sure current cpu id is 0 (primary processor) */
    host_set_cpuid(0U);
    /* Initialize the random seed */
    while (random_seed == 0) {
      random_seed = (int)time(NULL);
      srand(random_seed);
    }
    /* Clean RMM's internal struct granule array
      * to be sure tests start with a fresh copy.
      */
    memset((void *)get_granule_struct(), 0,
           sizeof(struct granule) *
           test_get_platform_nr_granules());
  }
  TEST_TEARDOWN()
  {};
```
Test example
Granule test module

```c
// Test example
#include <granule_find.h>

void TEST(granule, find_granule_TC1)
{
    struct granule *expected_granule = get_granule_struct();
    int granule_index = get_randInRange(0,
        test_get_platform_nr_granules() - 1);
    unsigned long address = (granule_index * GRANULE_SIZE) +
        host_get_granule_base();
    struct granule *granule;

    // Test Case 1:
    // Get a random granule and verify that its physical address
    // matches the calculated one.
    expected_granule = granule_index; // Expected address */
    granule = find_granule(address);
    POINTERS_EQUAL(expected_granule, granule);

    // Verify that not other parameters of the granule are altered */
    CHECK_TEXT(granule->state == 0, "Invalid granule state");
    CHECK_TEXT(granule->lock.val == 0, "Invalid granule lock status");
}

void TEST(granule, find_granule_TC2)
{
    unsigned long address;
    struct granule *granule;

    // Test Case 2:
    // Try to get a granule for an unaligned address.
    address = get_rand_granule_addr();
    address += get_randInRange(1, GRANULE_SIZE - 1);
    granule = find_granule(address);
    POINTERS_EQUAL(NULL, granule);
}
```
Test example

Test harness
Example of use

Building and running the tests
Example of use

Test logs

- Test results and logs are stored in `<build_dir>/Testing/Temporary/LastTest.log`
- If LOG_LEVEL is enabled (> 0), the file can be very long
For more information: https://tf-rmm.readthedocs.io/en/latest/
Thank You
Danke
Gracias
Grazie
谢谢
ありがとう
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה