RMM unit test framework

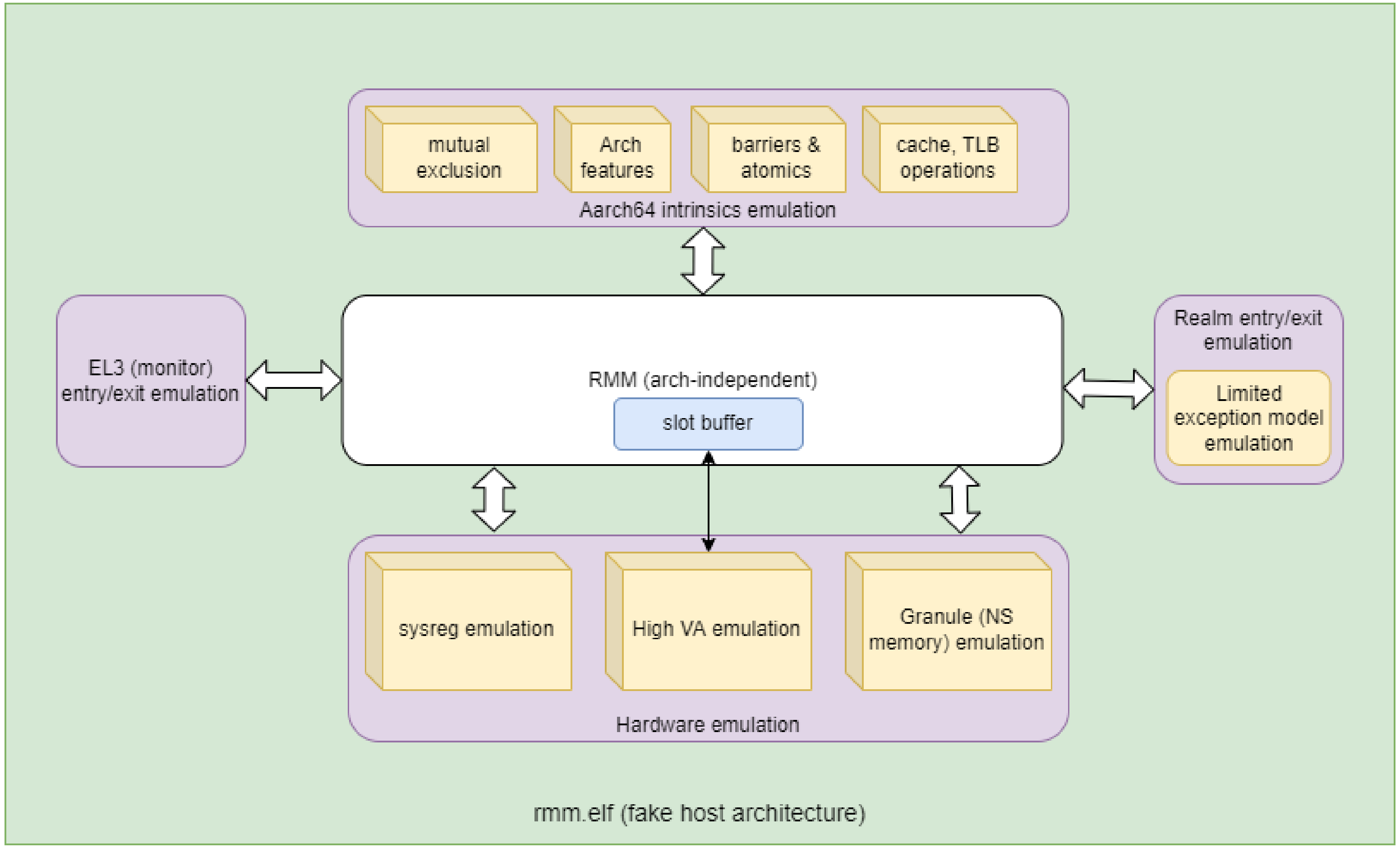
Javier Almansa Sobrino
11/17/2022

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

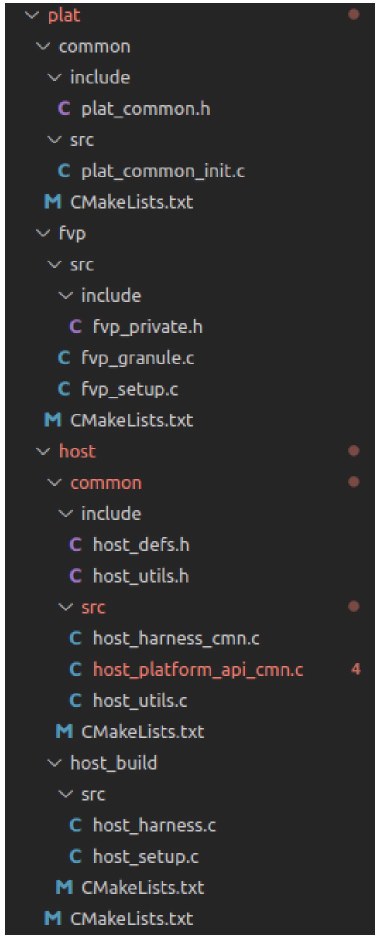
TF-RMM fake_host architecture

Fake_host architecture for RMM


mutual exclusion
Arch features
barriers & atomics
cache, TLB operations
Aarch64 intrinsics emulation
EL3 (monitor) entry/exit emulation
RMM (arch-independent)
slot buffer
Realm entry/exit emulation
Limited exception model emulation
sysreg emulation
High VA emulation
Granule (NS memory) emulation
Hardware emulation
rmm.elf (fake host architecture)

- 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.

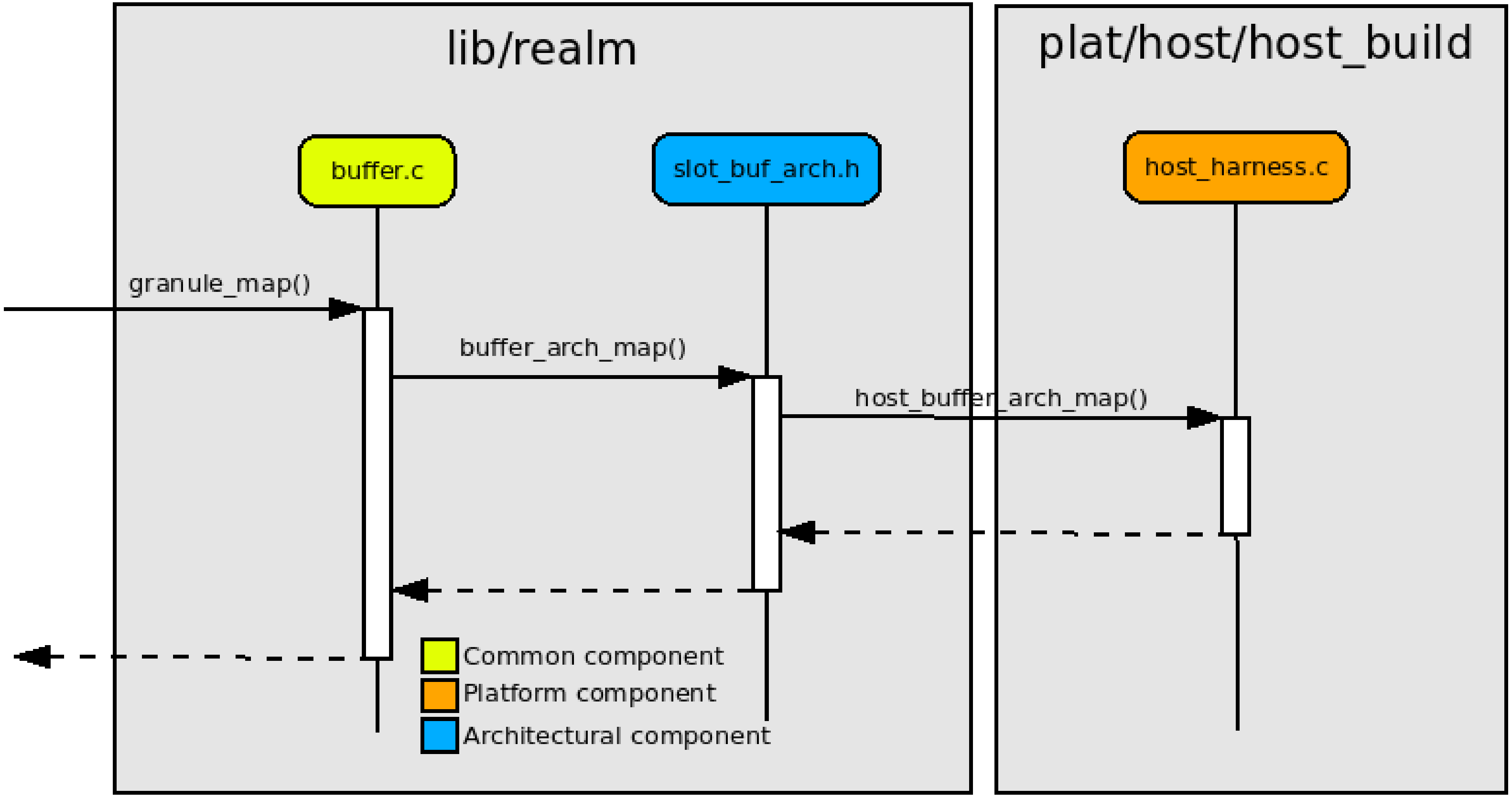
host platform components



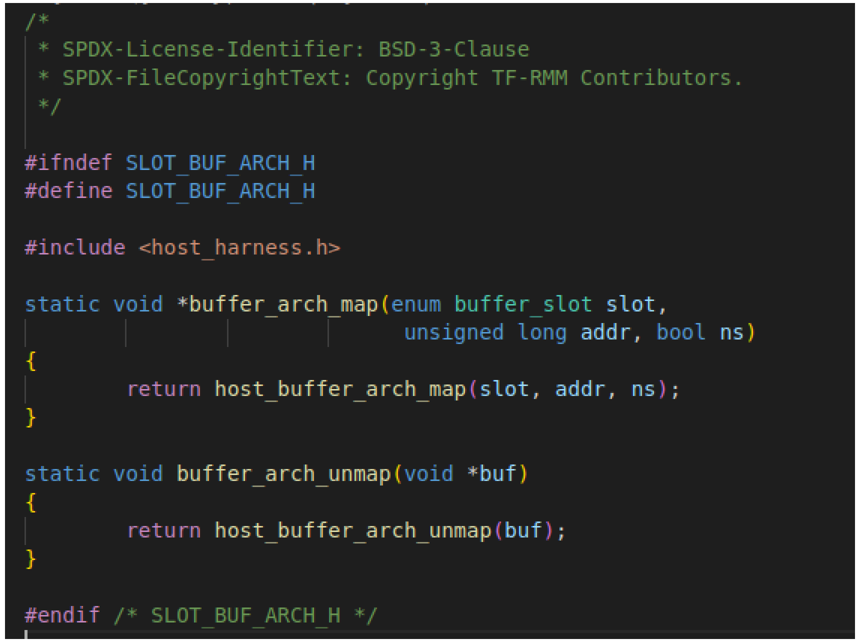
```
plat
  common
    include
      plat_common.h
    src
      plat_common_init.c
    CMakeLists.txt
  fvp
    src
      include
        fvp_private.h
      fvp_granule.c
      fvp_setup.c
    CMakeLists.txt
  host
    common
      include
        host_defs.h
        host_utils.h
      src
        host_harness_cmn.c
        host_platform_api_cmn.c      4
        host_utils.c
      CMakeLists.txt
    host_build
      src
        host_harness.c
        host_setup.c
      CMakeLists.txt
    CMakeLists.txt
```

- *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

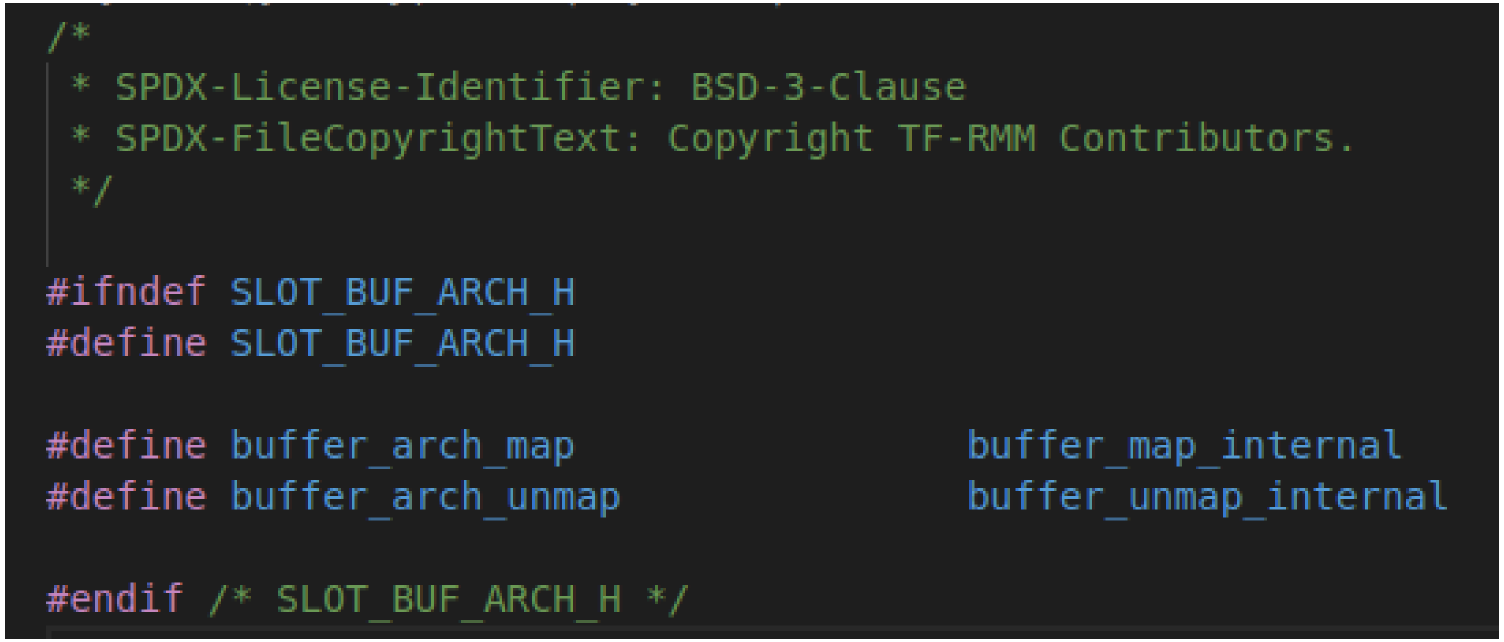


fake_host harness mechanism



```
/*
 * 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 */
```

fake_host



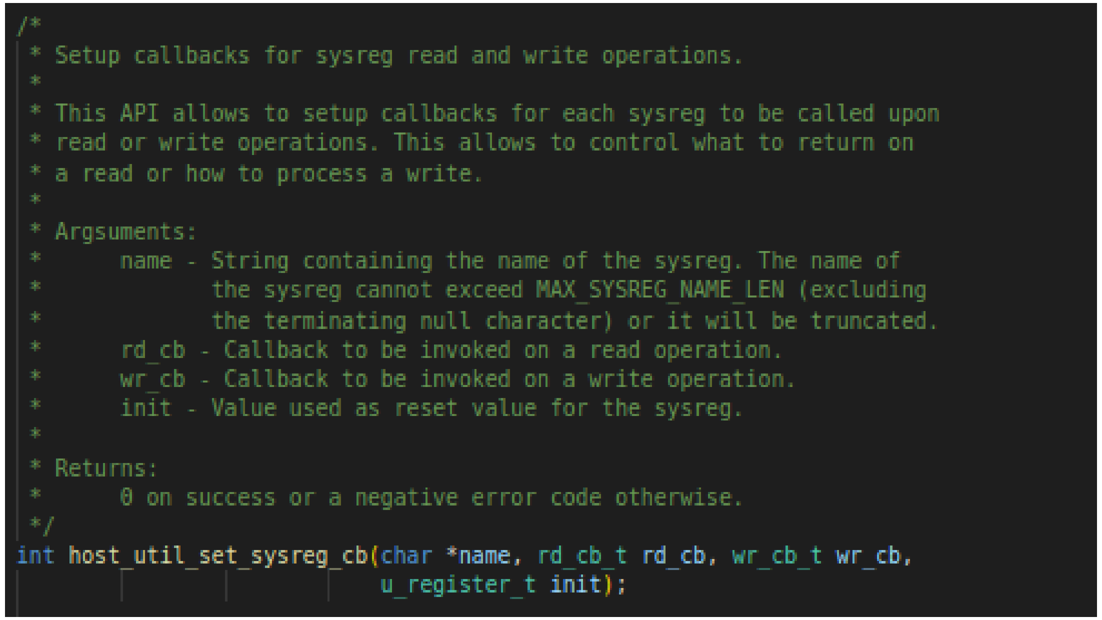
```
/*
 * SPDX-License-Identifier: BSD-3-Clause
 * SPDX-FileCopyrightText: Copyright TF-RMM Contributors.
 */

#ifndef SLOT_BUF_ARCH_H
#define SLOT_BUF_ARCH_H

#define buffer_arch_map                     buffer_map_internal
#define buffer_arch_unmap                   buffer_unmap_internal

#endif /* SLOT_BUF_ARCH_H */
```

fvp

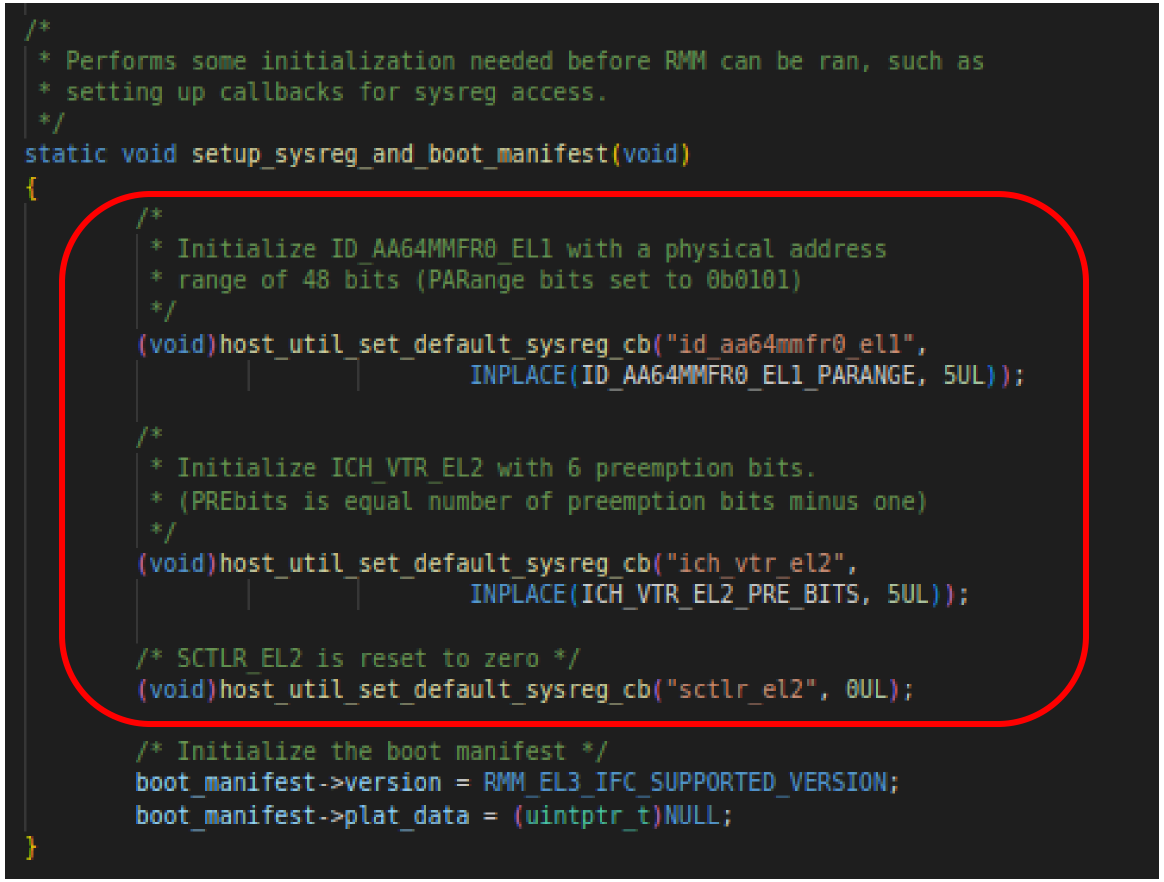
Sysreg access emulation on fake_host

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.



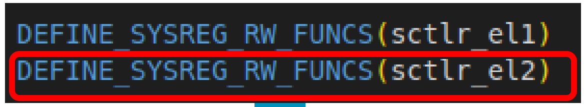
```c
/*
 * Setup callbacks for sysreg read and write operations.
 *
 * This API allows to setup callbacks for each sysreg to be called upon
 * read or write operations. This allows to control what to return on
 * a read or how to process a write.
 *
 * Argsuments:
 *      name - String containing the 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(char *name, rd_cb_t rd_cb, wr_cb_t wr_cb,
                            u_register_t init);
```



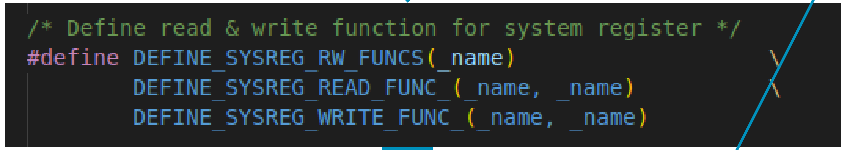
```c
/*
 * Performs some initialization needed before RMM can be ran, such as
 * setting up callbacks for sysreg access.
 */
static void setup_sysreg_and_boot_manifest(void)
{
        /*
         * Initialize ID_AA64MMFR0_EL1 with a physical address
         * range of 48 bits (PARange bits set to 0b0101)
         */
        (void)host_util_set_default_sysreg_cb("id_aa64mmfr0_el1",
                                INPLACE(ID_AA64MMFR0_EL1_PARANGE, 5UL));

        /*
         * 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, 5UL));

        /* 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

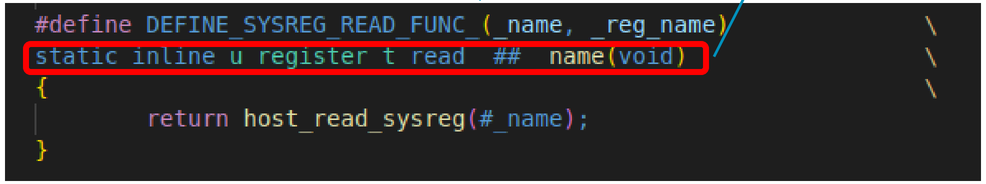
Callbacks for sysreg access



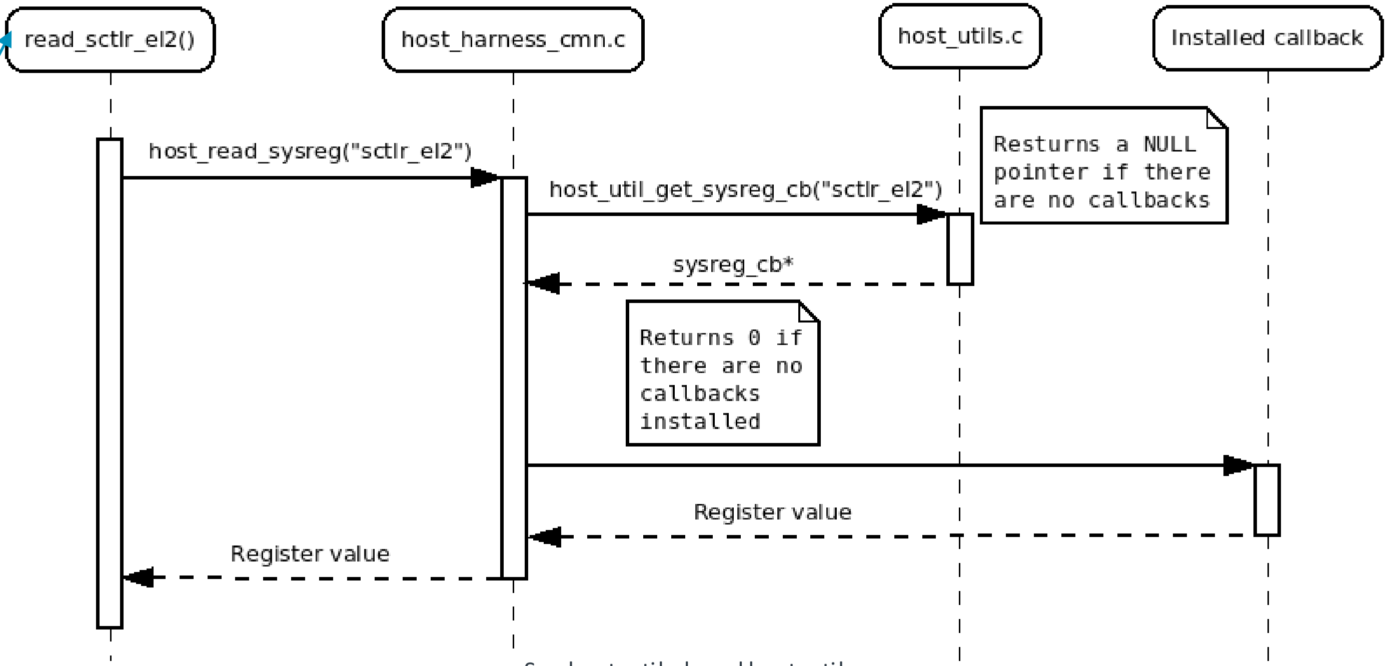
```
DEFINE_SYSREG_RW_FUNCS(sctlr_el1)
DEFINE_SYSREG_RW_FUNCS(sctlr_el2)
```



```
/* 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 u register t read  ##  name(void)             \
{                                                           \
        return host_read_sysreg(#_name);
}
```

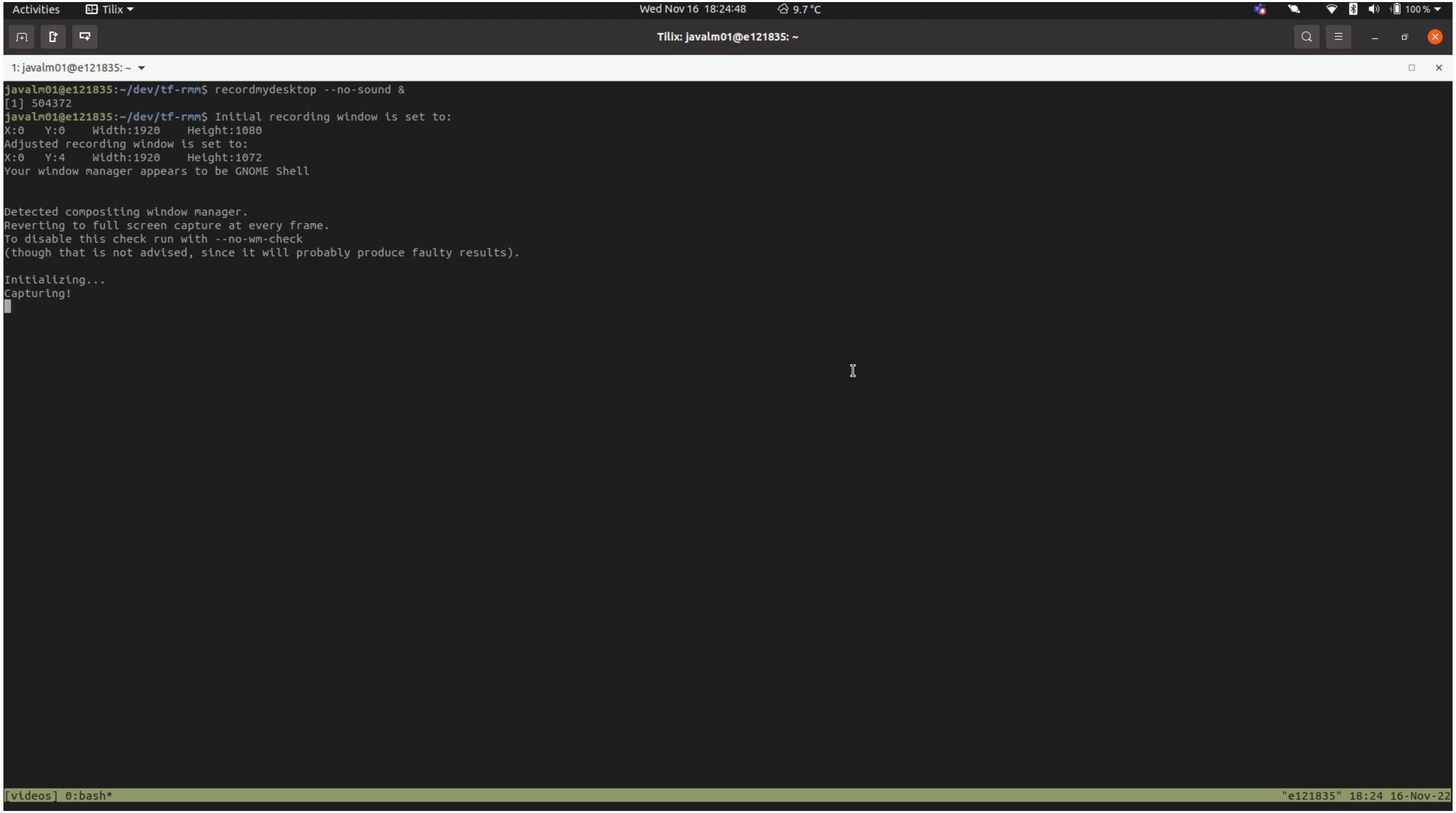


Resturns a NULL pointer if there are no callbacks

Returns 0 if there are no callbacks installed

See host_utils.h and host_utils.c

Fake_host build and run

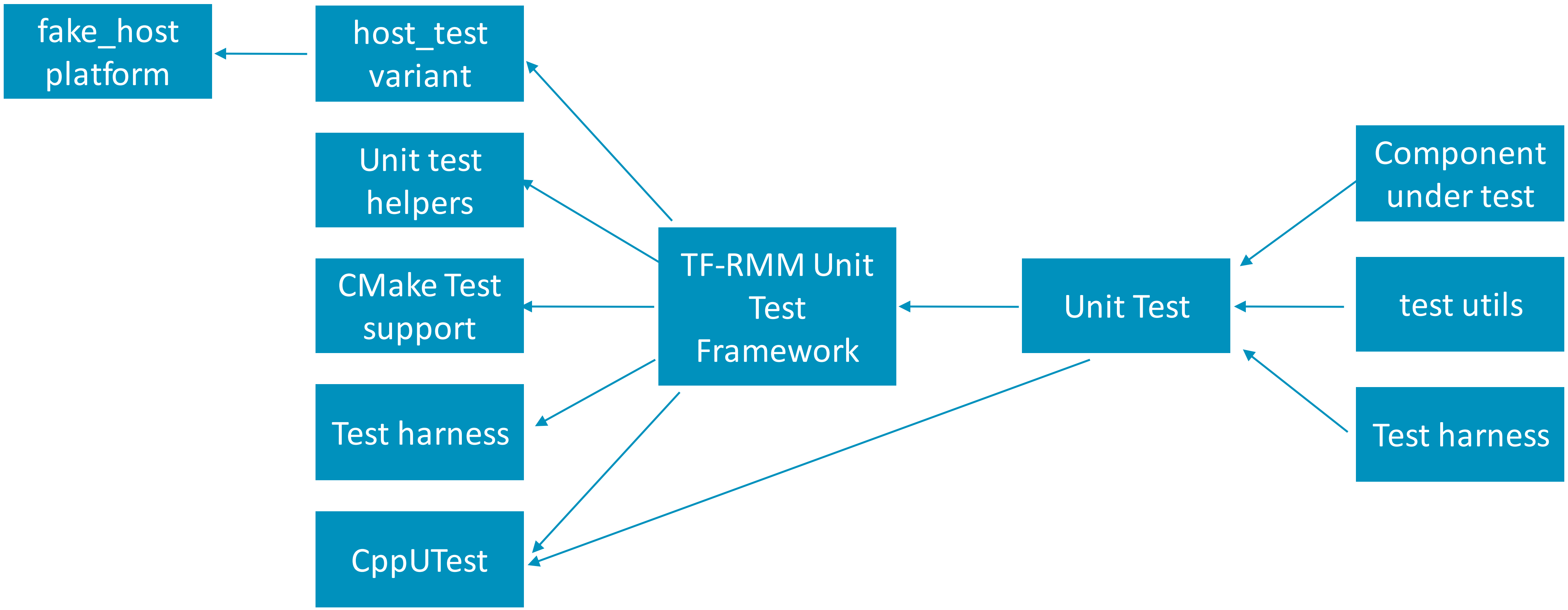


```
javalm01@e121835:~/dev/tf-rmm$ recordmydesktop --no-sound &
[1] 504372
javalm01@e121835:~/dev/tf-rmm$ Initial recording window is set to:
X:0   Y:0    Width:1920    Height:1080
Adjusted recording window is set to:
X:0   Y:4    Width:1920    Height:1072
Your window manager appears to be GNOME Shell


Detected compositing window manager.
Reverting to full screen capture at every frame.
To disable this check run with --no-wm-check
(though that is not advised, since it will probably produce faulty results).

Initializing...
Capturing!
```

```
[videos] 0:bash*                                                    "e121835" 18:24 16-Nov-22
```

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

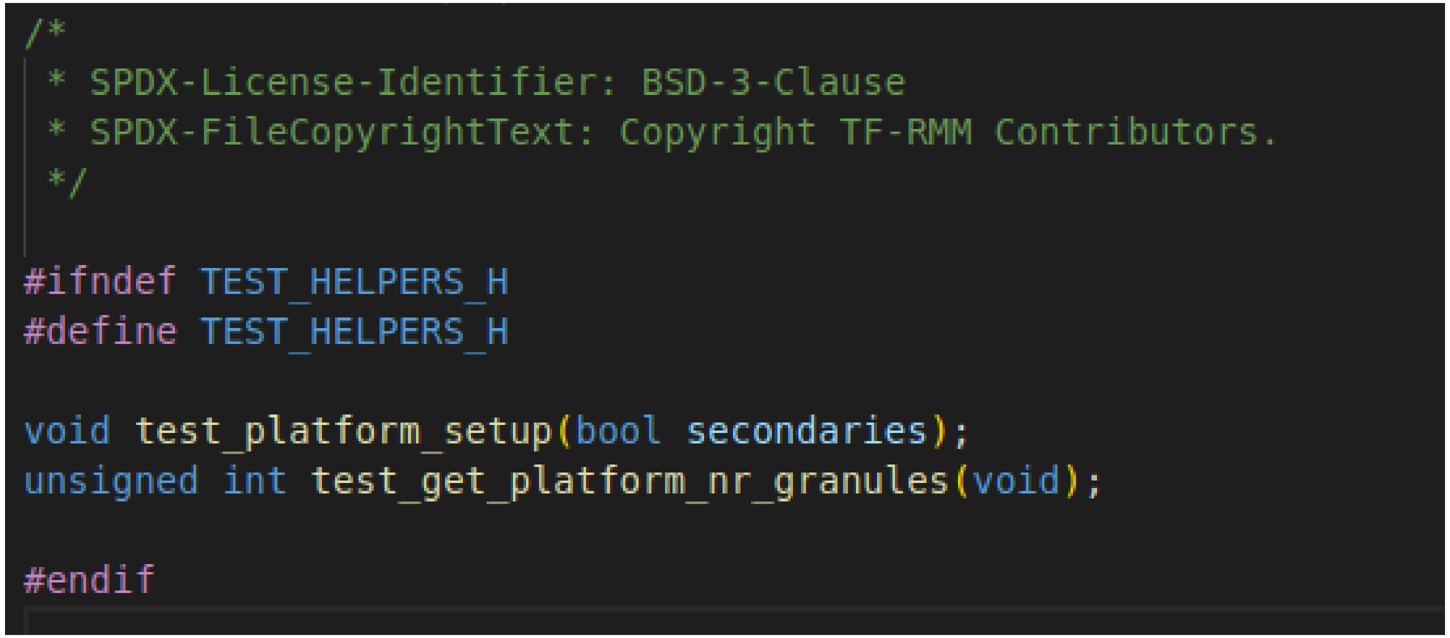
Components: CppUtest Framework

- 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/

Unittest 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

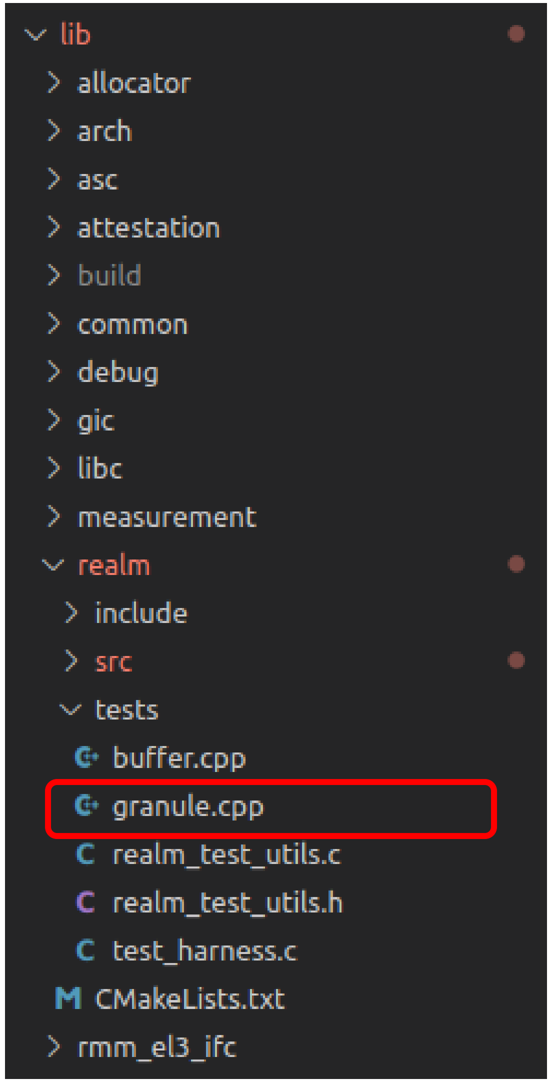


```
/*
 * SPDX-License-Identifier: BSD-3-Clause
 * SPDX-FileCopyrightText: Copyright TF-RMM Contributors.
 */

#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

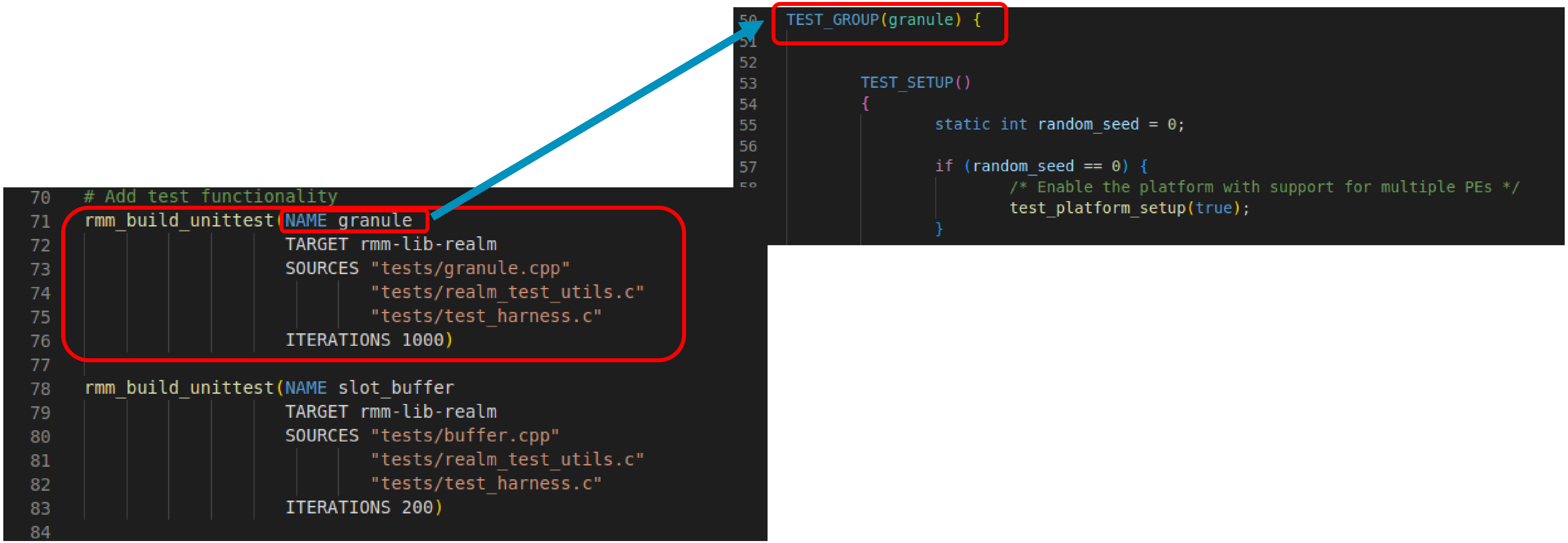
- Include a "*tests*" directory to hold all the tests for a given library along with ancillary sources.
- granule.c is part of lib/realm (src/granule.c), therefore granule.cpp will be inside the "*tests"* directory for lib/realm
  - We are only interested in the public API for this component
- CMakelists.txt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



```
v lib
  > allocator
  > arch
  > asc
  > attestation
  > build
  > common
  > debug
  > gic
  > libc
  > measurement
  v realm
    > include
    > src
    v tests
      buffer.cpp
      granule.cpp
      C realm_test_utils.c
      C realm_test_utils.h
      C test_harness.c
    M CMakeLists.txt
  > rmm_el3_ifc
```

Test example

Granule test module

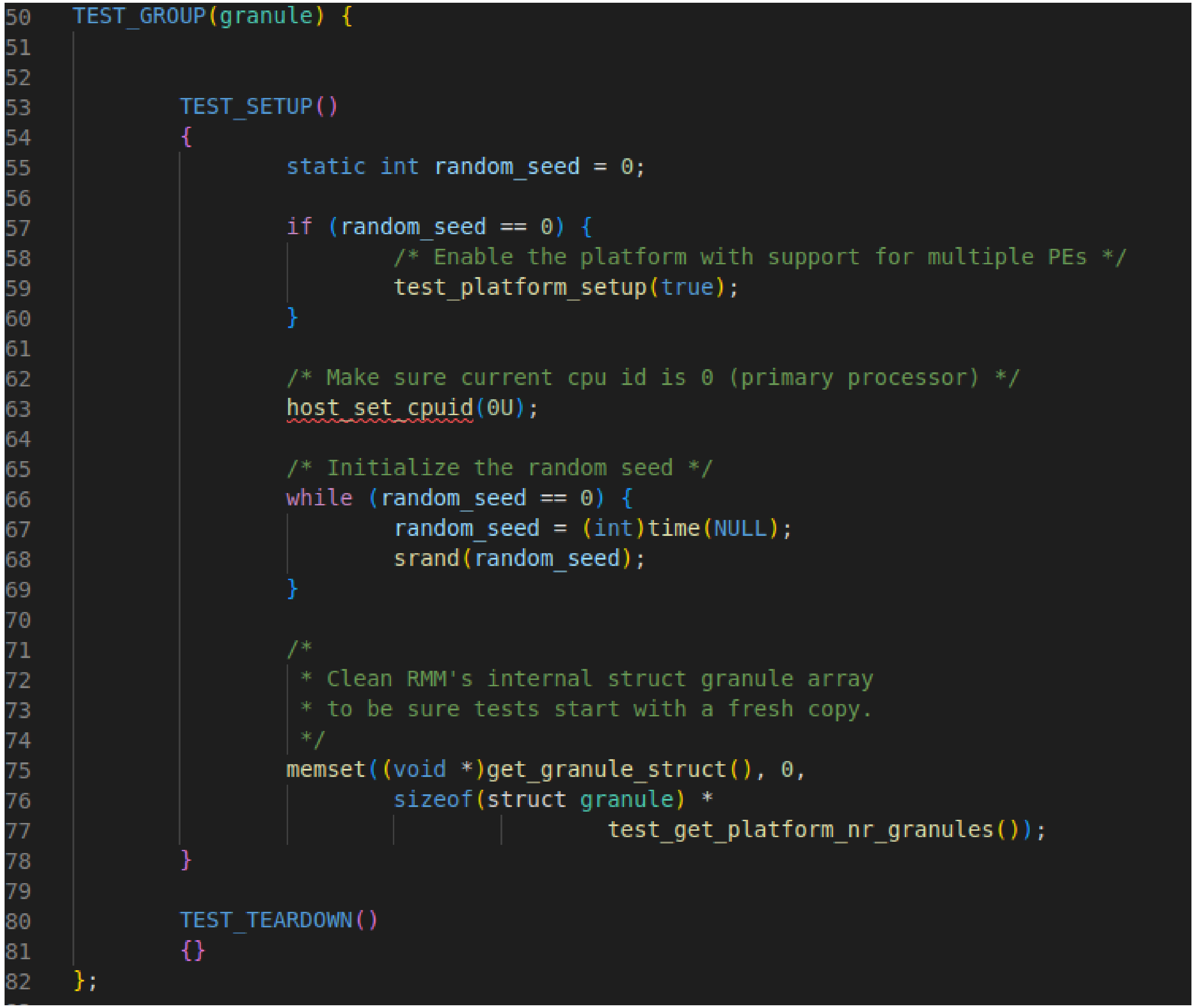


```
# Add test functionality
rmm_build_unittest(NAME granule
                   TARGET rmm-lib-realm
                   SOURCES "tests/granule.cpp"
                           "tests/realm_test_utils.c"
                           "tests/test_harness.c"
                   ITERATIONS 1000)

rmm_build_unittest(NAME slot_buffer
                   TARGET rmm-lib-realm
                   SOURCES "tests/buffer.cpp"
                           "tests/realm_test_utils.c"
                           "tests/test_harness.c"
                   ITERATIONS 200)

```

```
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);
                }
```

Test example

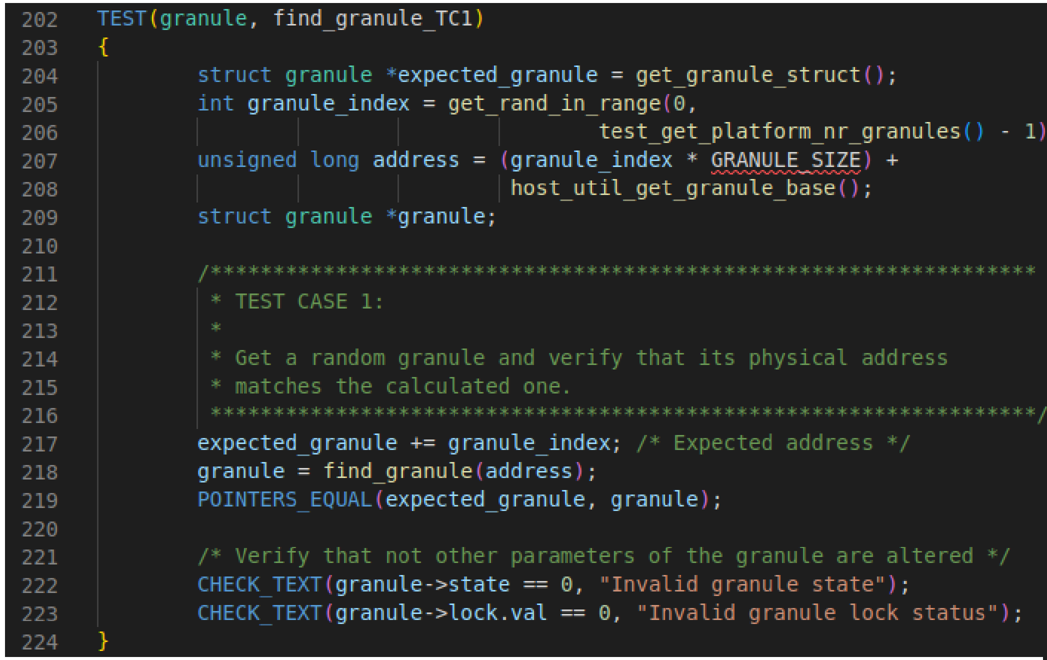
Granule test module



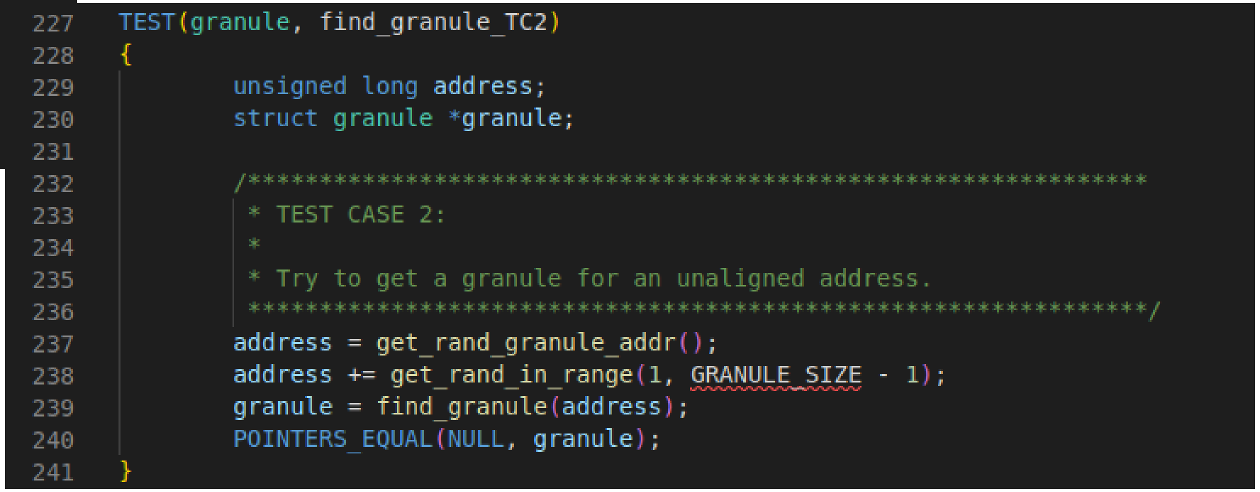
```
TEST_GROUP(granule) {


        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



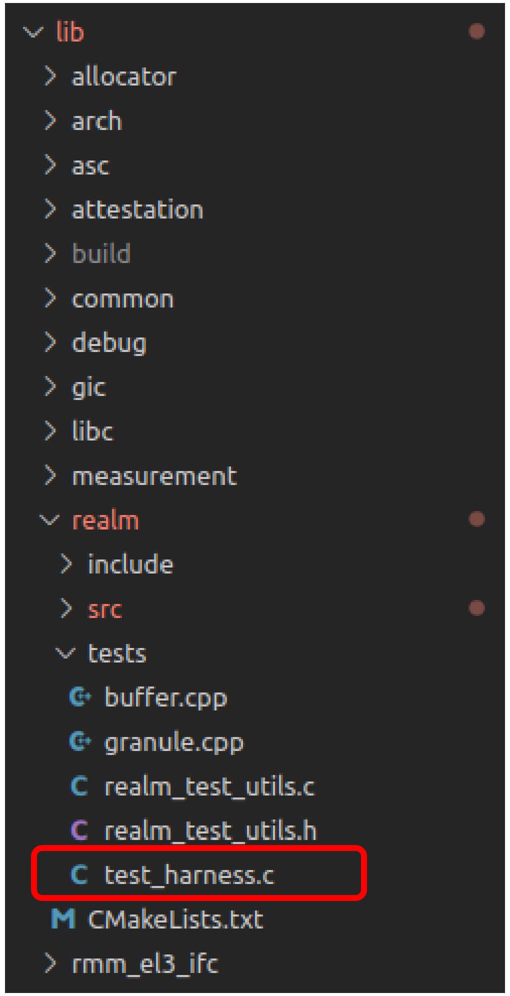
```c
TEST(granule, find_granule_TC1)
{
	struct granule *expected_granule = get_granule_struct();
	int granule_index = get_rand_in_range(0,
					test_get_platform_nr_granules() - 1);
	unsigned long address = (granule_index * GRANULE_SIZE) +
				host_util_get_granule_base();
	struct granule *granule;

	/******************************************************************
	 * 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");
}
```



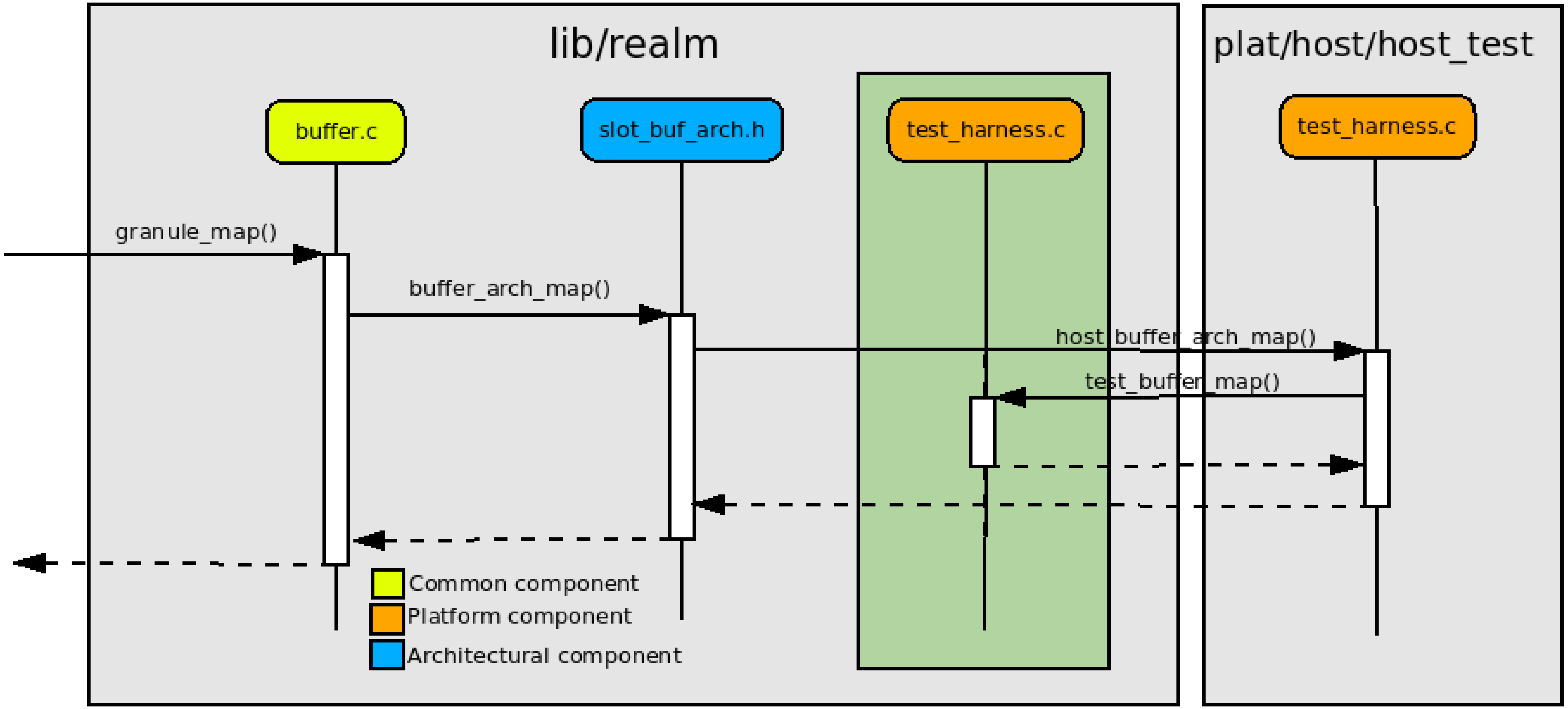
```c
TEST(granule, find_granule_TC2)
{
	unsigned long address;
	struct granule *granule;

	/******************************************************************
	 * TEST CASE 2:
	 *
	 * Try to get a granule for an unaligned address.
	 ******************************************************************/
	address = get_rand_granule_addr();
	address += get_rand_in_range(1, GRANULE_SIZE - 1);
	granule = find_granule(address);
	POINTERS_EQUAL(NULL, granule);
}
```

Test example

Test harness

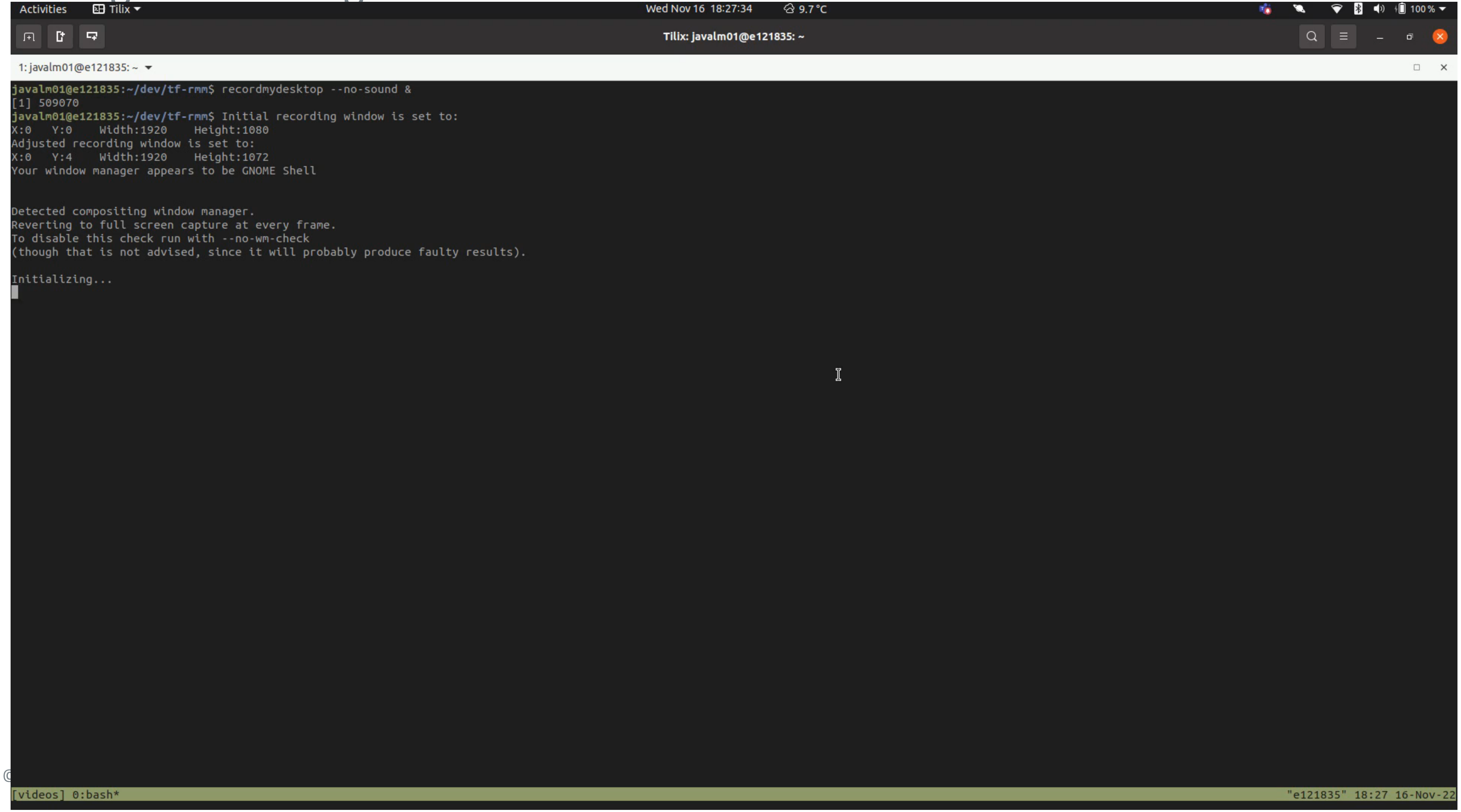


```
v lib
  > allocator
  > arch
  > asc
  > attestation
  > build
  > common
  > debug
  > gic
  > libc
  > measurement
  v realm
    > include
    > src
    v tests
      buffer.cpp
      granule.cpp
      realm_test_utils.c
      realm_test_utils.h
      test_harness.c
    CMakeLists.txt
  > rmm_el3_ifc
```



Example of use

Building and running the tests

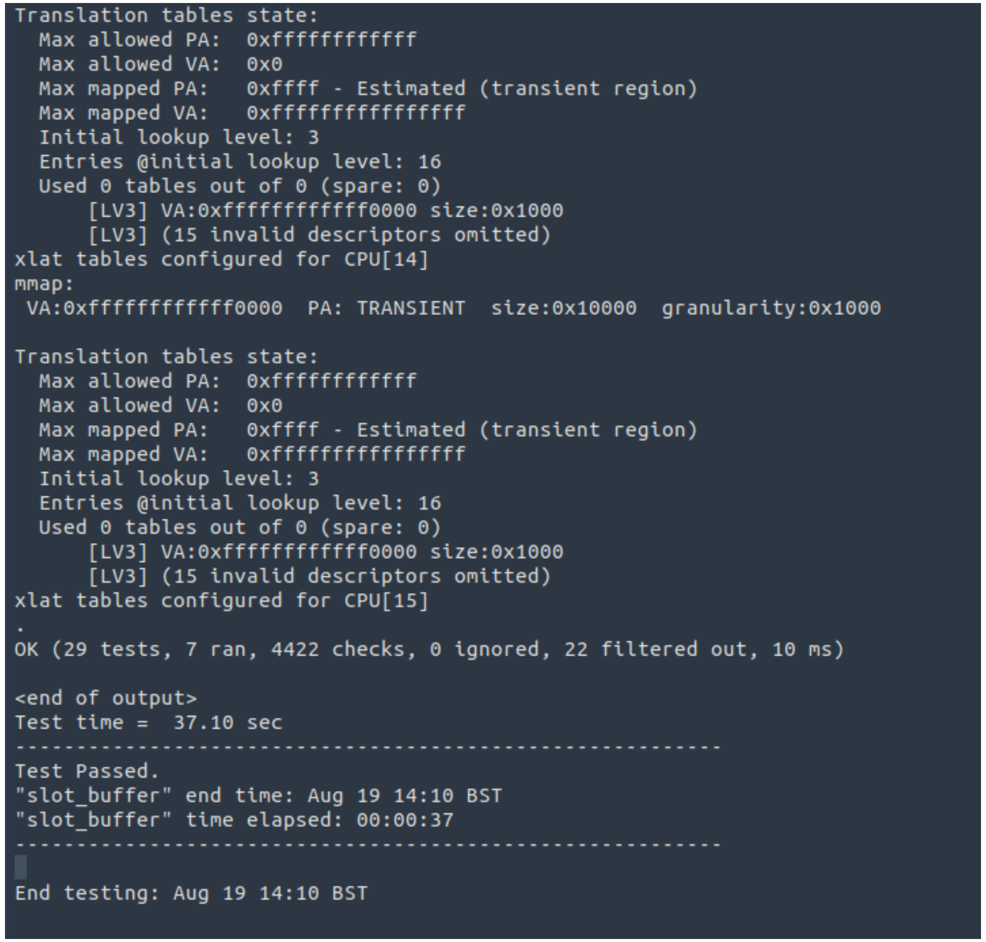


```
javalm01@e121835:~/dev/tf-rmm$ recordmydesktop --no-sound &
[1] 509070
javalm01@e121835:~/dev/tf-rmm$ Initial recording window is set to:
X:0   Y:0    Width:1920    Height:1080
Adjusted recording window is set to:
X:0   Y:4    Width:1920    Height:1072
Your window manager appears to be GNOME Shell


Detected compositing window manager.
Reverting to full screen capture at every frame.
To disable this check run with --no-wm-check
(though that is not advised, since it will probably produce faulty results).

Initializing...

[videos] 0:bash*                                                                    "e121835" 18:27 16-Nov-22
```

Example of use

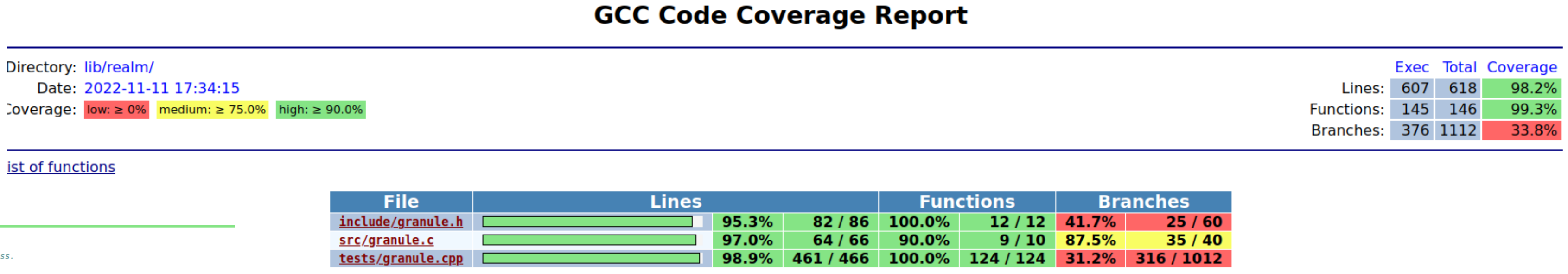
Test logs

- 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



```
Translation tables state:
  Max allowed PA:  0xffffffffffff
  Max allowed VA:  0x0
  Max mapped PA:   0xffff - Estimated (transient region)
  Max mapped VA:   0xffffffffffffffff
  Initial lookup level: 3
  Entries @initial lookup level: 16
  Used 0 tables out of 0 (spare: 0)
      [LV3] VA:0xffffffffffff0000 size:0x1000
      [LV3] (15 invalid descriptors omitted)
xlat tables configured for CPU[14]
mmap:
 VA:0xffffffffffff0000  PA: TRANSIENT  size:0x10000  granularity:0x1000

Translation tables state:
  Max allowed PA:  0xffffffffffff
  Max allowed VA:  0x0
  Max mapped PA:   0xffff - Estimated (transient region)
  Max mapped VA:   0xffffffffffffffff
  Initial lookup level: 3
  Entries @initial lookup level: 16
  Used 0 tables out of 0 (spare: 0)
      [LV3] VA:0xffffffffffff0000 size:0x1000
      [LV3] (15 invalid descriptors omitted)
xlat tables configured for CPU[15]
.
OK (29 tests, 7 ran, 4422 checks, 0 ignored, 22 filtered out, 10 ms)

<end of output>
Test time =  37.10 sec
----------------------------------------------------------
Test Passed.
"slot_buffer" end time: Aug 19 14:10 BST
"slot_buffer" time elapsed: 00:00:37
----------------------------------------------------------

End testing: Aug 19 14:10 BST
```

Coverage Report (WIP)



GCC Code Coverage Report

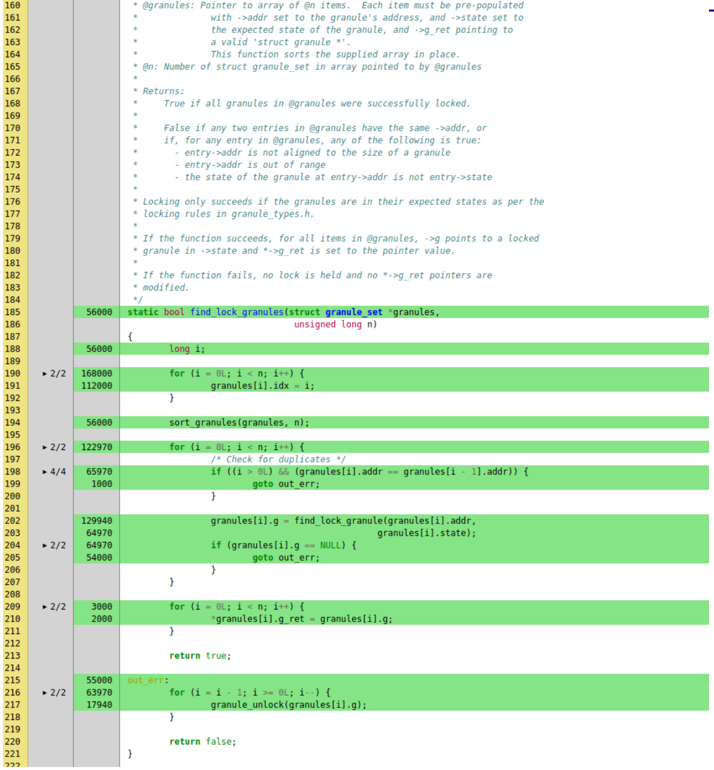
Directory: lib/realm/
Date: 2022-11-11 17:34:15
Coverage: low: ≥ 0% medium: ≥ 75.0% high: ≥ 90.0%

	Exec	Total	Coverage
Lines:	607	618	98.2%
Functions:	145	146	99.3%
Branches:	376	1112	33.8%

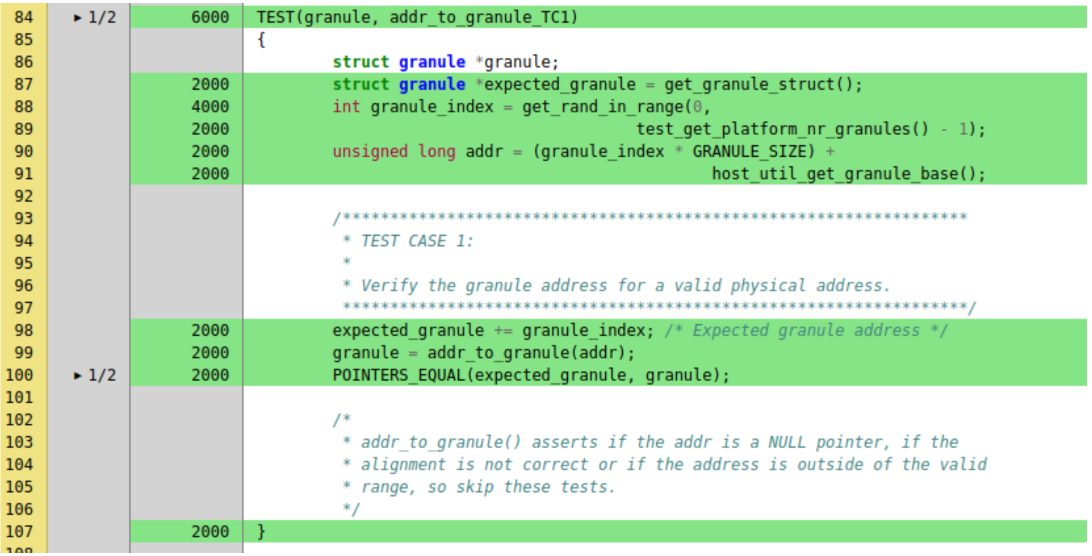
ist of functions

File	Lines			Functions		Branches	
include/granule.h		95.3%	82 / 86	100.0%	12 / 12	41.7%	25 / 60
src/granule.c		97.0%	64 / 66	90.0%	9 / 10	87.5%	35 / 40
tests/granule.cpp		98.9%	461 / 466	100.0%	124 / 124	31.2%	316 / 1012

Generated by: GCOVR (Version 5.2)



```
                /*
                 * Find a set of granules and lock them in order of their address.
                 *
                 * @granules: Pointer to array of @n items.  Each item must be pre-populated
                 *              with ->addr set to the granule's address, and ->state set to
                 *              the expected state of the granule, and ->g_ret pointing to
                 *              a valid 'struct granule *'.
                 *              This function sorts the supplied array in place.
                 * @n: Number of struct granule_set in array pointed to by @granules
                 *
                 * Returns:
                 *     True if all granules in @granules were successfully locked.
                 *
                 *     False if any two entries in @granules have the same ->addr, or
                 *     if, for any entry in @granules, any of the following is true:
                 *       - entry->addr is not aligned to the size of a granule
                 *       - entry->addr is out of range
                 *       - the state of the granule at entry->addr is not entry->state
                 *
                 * Locking only succeeds if the granules are in their expected states as per the
                 * locking rules in granule_types.h.
                 *
                 * If the function succeeds, for all items in @granules, ->g points to a locked
                 * granule in ->state and *->g_ret is set to the pointer value.
                 *
                 * If the function fails, no lock is held and no *->g_ret pointers are
                 * modified.
                 */
          56000 static bool find_lock_granules(struct granule_set *granules,
                                                unsigned long n)
                {
          56000         long i;

▸ 2/2    168000         for (i = 0L; i < n; i++) {
         112000                 granules[i].idx = i;
                        }

          56000         sort_granules(granules, n);

▸ 2/2    122970         for (i = 0L; i < n; i++) {
                                /* Check for duplicates */
▸ 4/4     65970                 if ((i > 0L) && (granules[i].addr == granules[i - 1].addr)) {
           1000                         goto out_err;
                                }

         129940                 granules[i].g = find_lock_granule(granules[i].addr,
          64970                                                   granules[i].state);
▸ 2/2     64970                 if (granules[i].g == NULL) {
          54000                         goto out_err;
                                }
                        }

▸ 2/2      3000         for (i = 0L; i < n; i++) {
           2000                 *granules[i].g_ret = granules[i].g;
                        }

                        return true;

          55000 out_err:
▸ 2/2     63970         for (i = i - 1; i >= 0L; i--) {
          17940                 granule_unlock(granules[i].g);
                        }

                        return false;
                }
```



```
▸ 1/2   6000  TEST(granule, addr_to_granule_TC1)
              {
                      struct granule *granule;
        2000          struct granule *expected_granule = get_granule_struct();
        4000          int granule_index = get_rand_in_range(0,
        2000                                  test_get_platform_nr_granules() - 1);
        2000          unsigned long addr = (granule_index * GRANULE_SIZE) +
        2000                                          host_util_get_granule_base();

                      /***********************************************************************
                       * TEST CASE 1:
                       *
                       * Verify the granule address for a valid physical address.
                       ***********************************************************************/
        2000          expected_granule += granule_index; /* Expected granule address */
        2000          granule = addr_to_granule(addr);
▸ 1/2   2000          POINTERS_EQUAL(expected_granule, granule);

                      /*
                       * addr_to_granule() asserts if the addr is a NULL pointer, if the
                       * alignment is not correct or if the address is outside of the valid
                       * range, so skip these tests.
                       */
        2000  }
```

- For more information: https://tf-rmm.readthedocs.io/en/latest/

Thank You

Danke

Gracias

Grazie

谢谢

ありがとう

Asante

Merci

감사합니다

धन्यवाद

Kiitos

شكرًا

ধন্যবাদ

תודה