Test Style and Nomenclature

To make finding, writing, and using KUnit tests as simple as possible, it is strongly encouraged that they are named and written according to the guidelines below. While it is possible to write KUnit tests which do not follow these rules, they may break some tooling, may conflict with other tests, and may not be run automatically by testing systems.

It is recommended that you only deviate from these guidelines when:

1. Porting tests to KUnit which are already known with an existing name.
2. Writing tests which would cause serious problems if automatically run. For example, non-deterministically producing false positives or negatives, or taking a long time to run.

Subsystems, Suites, and Tests

To make tests easy to find, they are grouped into suites and subsystems. A test suite is a group of tests which test a related area of the kernel. A subsystem is a set of test suites which test different parts of a kernel subsystem or a driver.

Subsystems

Every test suite must belong to a subsystem. A subsystem is a collection of one or more KUnit test suites which test the same driver or part of the kernel. A test subsystem should match a single kernel module. If the code being tested cannot be compiled as a module, in many cases the subsystem should correspond to a directory in the source tree or an entry in the `MAINTAINERS` file. If unsure, follow the conventions set by tests in similar areas.

Test subsystems should be named after the code being tested, either after the module (wherever possible), or after the directory or files being tested. Test subsystems should be named to avoid ambiguity where necessary.

If a test subsystem name has multiple components, they should be separated by underscores. *Do not* include "test" or "kunit" directly in the subsystem name unless we are actually testing other tests or the kunit framework itself. For example, subsystems could be called:

`ext4`
: Matches the module and filesystem name.

`apparmor`
: Matches the module name and LSM name.

`kasan`
: Common name for the tool, prominent part of the path `mm/kasan`

`snd_hda_codec_hdmi`
: Has several components (`snd`, `hda`, `codec`, `hdmi`) separated by underscores. Matches the module name.

Avoid names as shown in examples below:

`linear-ranges`
: Names should use underscores, not dashes, to separate words. Prefer `linear_ranges`.

`qos-kunit-test`
: This name should use underscores, and not have "kunit-test" as a suffix. `qos` is also ambiguous as a subsystem name, because several parts of the kernel have a `qos` subsystem. `power_qos` would be a better name.

`pc_parallel_port`
: The corresponding module name is `parport_pc`, so this subsystem should also be named `parport_pc`.

Note

The KUnit API and tools do not explicitly know about subsystems. They are a way of categorizing test suites and naming modules which provides a simple, consistent way for humans to find and run tests. This may change in the future.

Suites

KUnit tests are grouped into test suites, which cover a specific area of functionality being tested. Test suites can have shared initialization and shutdown code which is run for all tests in the suite. Not all subsystems need to be split into multiple test suites (for example, simple drivers).

Test suites are named after the subsystem they are part of. If a subsystem contains several suites, the specific area under test should be appended to the subsystem name, separated by an underscore.

In the event that there are multiple types of test using KUnit within a subsystem (for example, both unit tests and integration tests), they should be put into separate suites, with the type of test as the last element in the suite name. Unless these tests are actually present, avoid using `_test`, `_unittest` or similar in the suite name.

The full test suite name (including the subsystem name) should be specified as the `.name` member of the `kunit_suite` struct, and forms the base for the module name. For example, test suites could include:

`ext4_inode`
: Part of the `ext4` subsystem, testing the `inode` area.

`kunit_try_catch`
: Part of the `kunit` implementation itself, testing the `try_catch` area.

`apparmor_property_entry`
: Part of the `apparmor` subsystem, testing the `property_entry` area.

`kasan`
: The `kasan` subsystem has only one suite, so the suite name is the same as the subsystem name.

Avoid names, for example:

`ext4_ext4_inode`
: There is no reason to state the subsystem twice.

`property_entry`
: The suite name is ambiguous without the subsystem name.

`kasan_integration_test`
: Because there is only one suite in the `kasan` subsystem, the suite should just be called as `kasan`. Do not redundantly add `integration_test`. It should be a separate test suite. For example, if the unit tests are added, then that suite could be named as `kasan_unittest` or similar.

Test Cases

Individual tests consist of a single function which tests a constrained codepath, property, or function. In the test output, an individual test's results will show up as subtests of the suite's results.

Tests should be named after what they are testing. This is often the name of the function being tested, with a description of the input or codepath being tested. As tests are C functions, they should be named and written in accordance with the kernel coding style.

> **Note**
>
> As tests are themselves functions, their names cannot conflict with other C identifiers in the kernel. This may require some creative naming. It is a good idea to make your test functions *static* to avoid polluting the global namespace.

Example test names include:

`unpack_u32_with_null_name`
: Tests the `unpack_u32` function when a NULL name is passed in.

`test_list_splice`
: Tests the `list_splice` macro. It has the prefix `test_` to avoid a name conflict with the macro itself.

Should it be necessary to refer to a test outside the context of its test suite, the *fully-qualified* name of a test should be the suite name followed by the test name, separated by a colon (i.e. `suite:test`).

Test Kconfig Entries

Every test suite should be tied to a Kconfig entry.

This Kconfig entry must:

- be named `CONFIG_<name>_KUNIT_TEST`: where <name> is the name of the test suite.
- be listed either alongside the config entries for the driver/subsystem being tested, or be under [Kernel Hacking]->[Kernel Testing and Coverage]
- depend on `CONFIG_KUNIT`.
- be visible only if `CONFIG_KUNIT_ALL_TESTS` is not enabled.
- have a default value of `CONFIG_KUNIT_ALL_TESTS`.
- have a brief description of KUnit in the help text.

If we are not able to meet above conditions (for example, the test is unable to be built as a module), Kconfig entries for tests should be tristate.

For example, a Kconfig entry might look like:

> **System Message: WARNING/2 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\dev-tools\kunit\(linux-master) (Documentation) (dev-tools) (kunit)style.rst`, line 173)**
>
> Cannot analyze code. No Pygments lexer found for "none".
>
> ```
> .. code-block:: none
>
>         config FOO_KUNIT_TEST
> ```

```
        tristate "KUnit test for foo" if !KUNIT_ALL_TESTS
        depends on KUNIT
        default KUNIT_ALL_TESTS
        help
          This builds unit tests for foo.

          For more information on KUnit and unit tests in general,
          please refer to the KUnit documentation in Documentation/dev-tools/kunit/.

          If unsure, say N.
```

Test File and Module Names

KUnit tests can often be compiled as a module. These modules should be named after the test suite, followed by `_test`. If this is likely to conflict with non-KUnit tests, the suffix `_kunit` can also be used.

The easiest way of achieving this is to name the file containing the test suite `<suite>_test.c` (or, as above, `<suite>_kunit.c`). This file should be placed next to the code under test.

If the suite name contains some or all of the name of the test's parent directory, it may make sense to modify the source filename to reduce redundancy. For example, a `foo_firmware` suite could be in the `foo/firmware_test.c` file.