Testing Drivers with KUnit

(Does Hardware have to be Hard?)

David Gow <davidgow@google.com>

What Is KUnit?

What Is KUnit?

- A Unit Testing framework for the Linux Kernel.
- Upstream since 5.5
- Tests are written in C, run in kernel mode, and can call arbitrary kernel functions.
- Tools to run these tests, and parse the results:
  - ./tools/testing/kunit/kunit.py run
  - Uses User-Mode Linux by default, or QEMU for other architectures.
  - ./tools/testing/kunit/kunit.py run --arch x86_64

Recent and Advanced Features

- Test-managed resources: automatically clean things up on test failure/completion.
- Parameterised testing: provide an array of inputs (or a generator function) to create several similar tests.
- Function redirection (mocking)
- A bunch of tooling features:
  - Architecture emulation via kunit.py --arch {x86_64,arm64,s390,etc}
  - Easily add extra kconfig options with --kconfig_add CONFIG_KASAN=y
- For the full list of changes, version by version, see https://kunit.dev/release_notes.html

Some code is easy to test

"Library" code

- Data structures and algorithms
- Helper functions
- Parsers
- Anything 'self-contained' or 'pure'
- Code with explicit abstractions

Some code is really difficult

Global state (in all its forms)

- Anything with global or static state
- High coupling to other systems
- Hardware state is global state
- Big lists of things (e.g. devices) in the kernel
  - 'register' a device / filesystem / etc
- Implicit global state (memory allocations, etc)

Why?

- Need a known starting state.
- Need to mutate that state.
- Need no conflicting mutations.

However,

- Can start in an unknown state.
- Mutating that state can break other parts of the kernel.
- Other parts of the kernel can modify the state.
- Can't just lock it: some of this state may be necessary to run the test.
- How do you handle failed tests, leaks, etc.

What can we do?

Design / refactor code for testing

- Minimise global state.
- Where that's not possible, wrap it.
- Make good use of 'pure' helper functions.
- Goal: swap in fake clients and devices.
- Goal: clean internal API surfaces.

But:

- Lots of existing code.
- Can require more work.
- Can have performance impacts.

Function Redirection (static_stub)

- Redirect calls to a global function during the test.
- Available since 6.3
- Requires adding a macro to the 'target' function:
  - KUNIT_STATIC_STUB_REDIRECT(fn_name, args…)
  - Compiles to 'if (function is redirected) return new_function(args)'
- Redirection only happens from test thread, can be controlled by tests at runtime.
- No performance impact if no KUnit, minimal if redirection not enabled

But,

- Can't redirect things needed for the test to function (e.g. kmalloc())
- May need to export functions if deep in the callstack.
- Multithreaded tests can be fun.

Devices and Drivers

- Most drivers need a device pointer, which needs registering
- In the past: struct root_device & root_device_register()
  - Worked well for simple cases, but caused some horror
- Platform devices
  - Can work, but still need a bus of some kind.
- DeviceTree support
  - Stephen Boyd has some patches.
  - A magic 'linux,kunit' board
  - Switching to DT overlays
- struct kunit_device
  - Patches in progress to have a specific kunit device and kunit bus.
  - Helpers to manage these within the test lifecycle.

Open Questions

LOGIC_IOMEM

- UML feature used for virtio/PCI.
- Callbacks for iomem accesses.
- Do we want this for KUnit hardware mocking?
  - Can intercept register writes.
- If so, do we need to port this to non-UML architectures?
  - How do we handle integration between real iomem and logic iomem?
- Breaks the fallback approach of just passing real memory around and inspecting it.

Variable redirection

- A.K.A static_data_stubbing
- Replace a global variable with another within a test
- (Macro magic and a pointer indirection)
- Prototype exists, but probably over-the-top.

User context / MM context

- Make copy_{to,from}_user() and similar work.
- No easy way of creating a context which works from kernel space (everything's done in execve())
- Some promising prototypes.

Something else?

Questions / Comments?

Or visit kunit.dev/ and subscribe to kunit-dev@googlegroups.com