Contribution ID: **131** Type: **not specified**

How do we make KUnit work for us?

KUnit is the only unit testing framework in the Linux kernel, but Android kernel changes are rarely accompanied by KUnit tests. Aside from the relative monotony associated with writing tests, one of the main barriers to more widespread KUnit testing seems to its inability (perceived or actual) to accommodate the complex use cases that we are developing features for.

When test code doesn't fit neatly around these changes, sometimes that's a limitation of the tooling, but other times it may signal a need to restructure the feature (or even the whole subsystem) under test. Refactoring an existing feature purely to accommodate testing may seem misguided, especially if doing so increases the memory footprint of the associated structures. Still, in many areas of software engineering, code's "testability" is one measure of its quality.

The goal of this talk will be to cover some patterns where KUnit falls short to facilitate a discussion about when the "correct" solution is to add functionality to KUnit and when it's to modify the code under test. Finally, I hope that having people talk about this will trigger them to consider how they might test their in-flight features, and maybe improve kernel test coverage as a result.

Primary author: YANG, Tiffany (Google)

Presenter: YANG, Tiffany (Google)

Session Classification: Android MC

Track Classification: Android MC