Kselftest running in test rings - Where are we?

Shuah Khan
Kernel Maintainer & Fellow
The Linux Foundation

Embedded Linux Conference Europe
October 27 2020

@ShuahKhan

Abstract

Kselftest is a developer test suite which has evolved to run in test rings, and by distributions. This evolution hasn't been an easy one.

In this talk, Shuah shares what it took to get Kselftest running in test rings such as Kernel CI. She will go over the changes necessary to run Kselftests to fully support relocatable builds and enable integration into test rings.

Agenda

Kselftest Overview

Kselftest goals & challenges

Kselftest use-cases

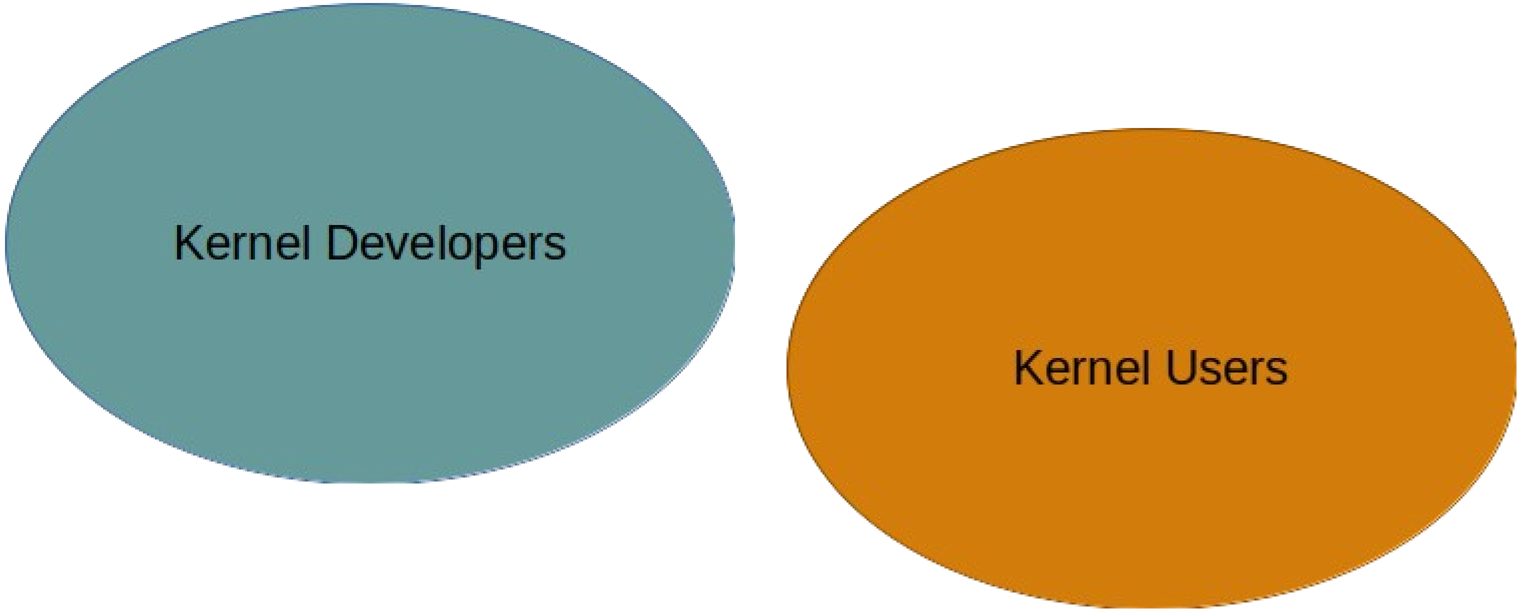
Kselftest running in test rings

Kselftest use-cases for Kernel CI and other test rings (support status)

Kselftest Kernel CI workflow

Next Steps

Kselftest is a developer test suite for



Kselftest is a collection of tests

Open and closed box

Functional & feature

Hardware and drivers

Stress and performance

IRC: freenode #linux-kselftest

https://git.kernel.org/pub/scm/linux/kernel/git/shuah/linux-kselftest.git/

https://patchwork.kernel.org/project/linux-kselftest/list/

Testing focus is

Features

Functionality

Regressions

Subsystems

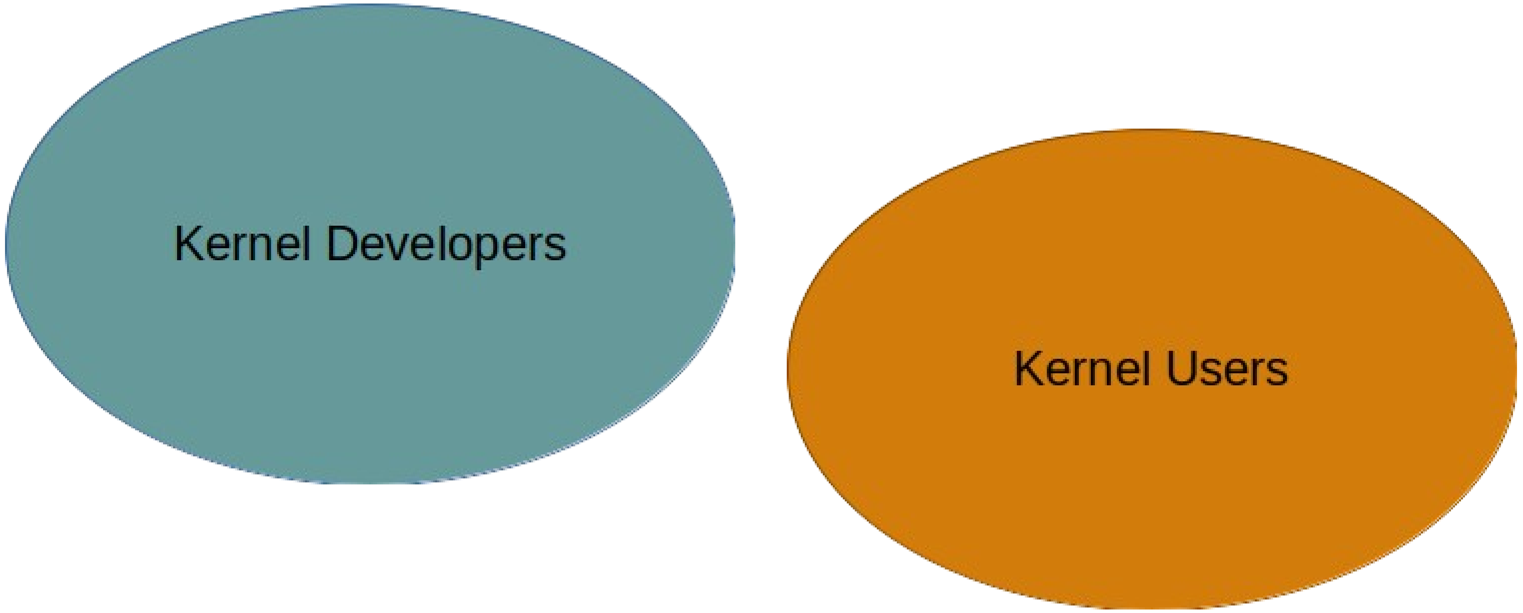
Several new tests and test cases are added every release.

Kselftest is not for testing

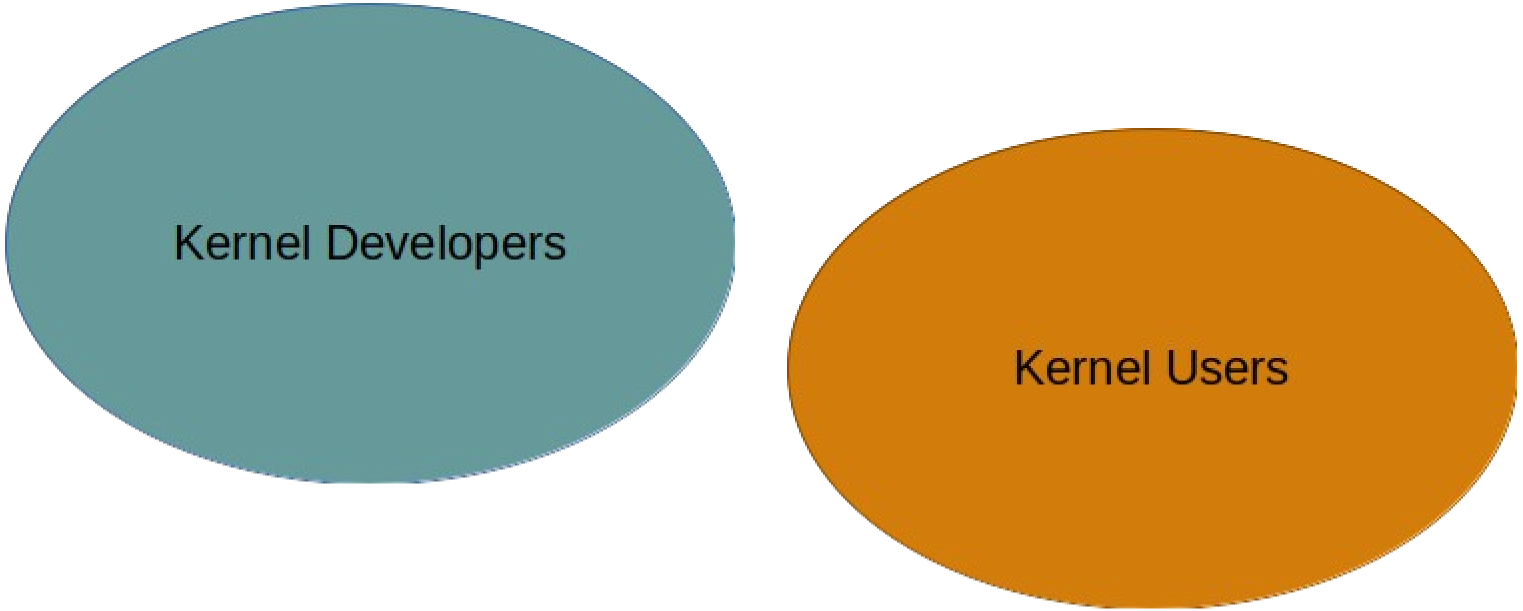
Workloads

Applications

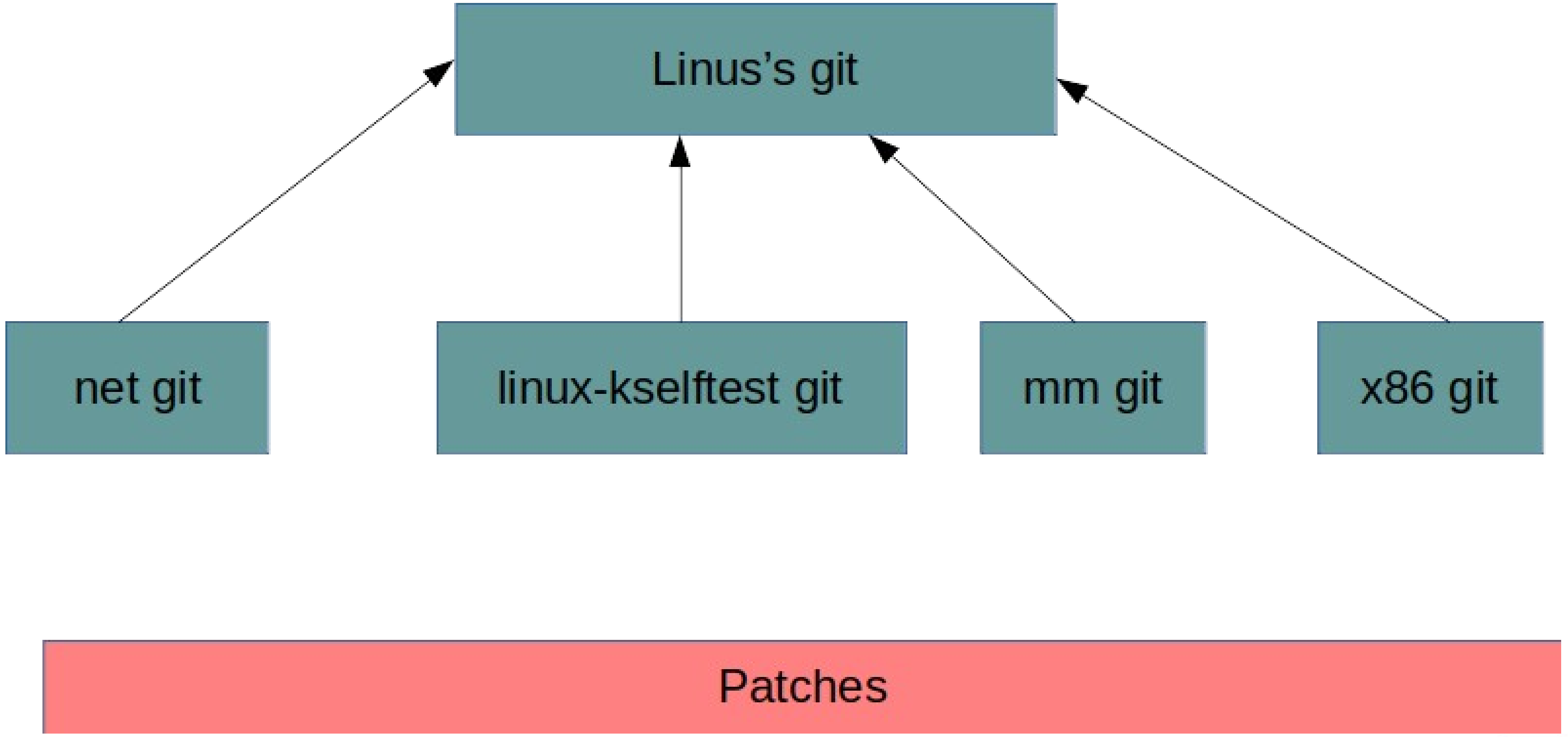
Who are the authors?



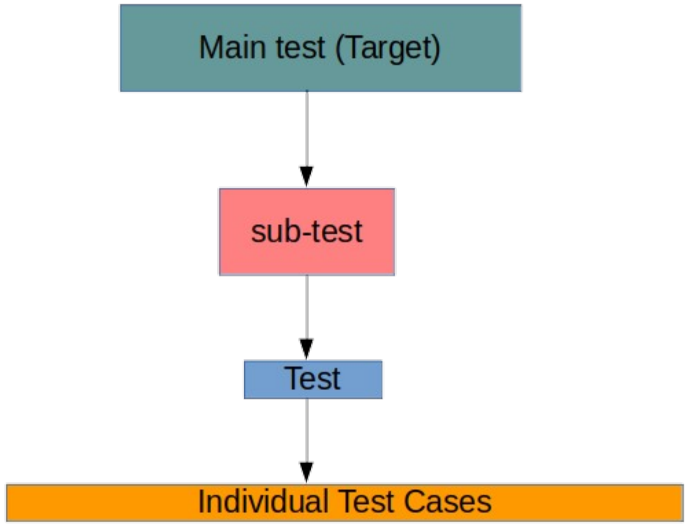
Who are the users?



Patch flow


Linus's git
net git
linux-kselftest git
mm git
x86 git
Patches

Individual test view



Kselftest goals & challenges

Evolving common framework flexible for customizing tests

Increase coverage (drivers, configs, and features)

Add regression tests for fixed bugs

Common interfaces for Pass/Fail/Skip reporting

Reporting results in simple text based Test Anything Protocol 13

Balance kselftest run-time and coverage

Kselftest - goals & challenges

Balance kernel developer and user use-cases

Evolving common framework to support test ring use-cases

- Framework is well suited for manual testing.
- Needs changes to support auto-test environments.

Kselftest use-cases

Native and cross-build use-cases

- Individual tests
- Subset of tests
- All tests

Relocating native and cross-build objects

Kselftest use-cases

Running tests - use-cases

- Individual tests
- Subset of tests
- All tests

Kselftest use-cases

Generating installable tests and run script - use-cases

- Individual tests
- Subset of tests
- All tests

Support relocating install objects (native & cross-builds)

Kselftest and test rings

[Linux Kernel Functional Testing]

Runs Kselftests on:

[Linux-next]

[Linux-mainline]

[Stable]

[Active kernel Releases]

Kselftests from the same repo are used to rev match kernel. One exception is Kselftest from latest stable is run on all stables.

Kselftest and test rings

[0-Day service]

Runs Kselftests from mainline on several trees and kernel configs.

Kernel CI (test rings) use-cases

Support relocating install objects (native & cross-builds)

- Supported in Linux 5.6 (except bpf)
- Relative path support is work in progress

Dependency checks for build/cross-build - kselftest_deps.sh

- Supported in Linux 5.6.
- Prints test targets that can be built. This output can be used in auto-test frameworks.

Build/cross-build tests for specific subsystems (supported with TARGETS var)

Kernel CI (test rings) use-cases

Build/cross-build tests for specific configs

- Individual tests add config file with required dependencies tools/testing/selftests/*/config
- "**make kselftest-merge**" generates kernel config to include individual test config files

Build/cross-build tests for specific features (this is a bit tricky)

- One single test could cover multiple features for a config or a subsystem.

Kselftest commands

Default builds/runs/installs all TARGETS.

- make kselftest-all
- make kselftest
- make kselftest-install

Using TARGETS helps select a subset of tests to build.

- make kselftest-install TARGETS=”breakpoints timers”

Install generates a script to run tests and report results.

Kselftest Kernel CI workflow

Cross-compile kernel (relocatable):

- make O=/arm64_build ARCH=arm64 HOSTCC=gcc CROSS_COMPILE=aarch64-linux-gnu- defconfig
- make O=/arm64_build ARCH=arm64 HOSTCC=gcc CROSS_COMPILE=aarch64-linux-gnu- all

Kselftest Kernel CI workflow

Cross-compile kselftest-all (relocatable):

- make kselftest-all ARCH=arm64 HOSTCC=gcc CROSS_COMPILE=arm-linux-gnueabi- O=/tmp/kselftest_arm > kselftest_all_arm.log 2>&1
- make -C tools/testing/selftests ARCH=arm64 HOSTCC=gcc CROSS_COMPILE=aarch64-linux-gnu- CC="ccache aarch64-linux-gnu-gcc" O=build-arm64

Kselftest Kernel CI workflow

Cross-compile kselftest-install (relocatable):

- make kselftest-install O=/arm64_build ARCH=arm64 HOSTCC=gcc CROSS_COMPILE=aarch64-linux-gnu- > kselftest_install 2>&1

New in Linux 5.10-rc1

Speed up headers_install done during selftest build

Add generic make nesting support

Add support to select individual tests to run_kselftest.sh script:

With this enhancement, user can select test collections (or tests) individually. e.g:

```
run_kselftest.sh -c seccomp -t timers:posix_timers -t timers:nanosleep
```

Additionally adds a way to list all known tests with "-l", usage with "-h", and perform a dry run without running tests with "-n".

Next Steps

Add support for build/cross-build tests for specific configs

Add support for build/cross-build tests for specific features (this is a bit tricky)

- One single test could cover multiple features for a config or a subsystem.

Thank You

THE LINUX FOUNDATION