LF *live*

MENTORSHIP SERIES

LF live MENTORSHIP SERIES

Kernel Validation With Kselftest

Shuah Khan, Kernel Maintainer and Fellow,
The Linux Foundation

LF Live MENTORSHIP SERIES

- Why do we test?
- Kinds of testing/tests ...
  - Unit, developer, regression, integration

- Linux kernel testing philosophy
  - Developer and community driven testing
  - Reliance on community and users

- Linux kernel release cycle
  - Time based - not feature based
  - Continuous and parallel development/testing model

- Linux kernel testing and validation
  - Writing tests
    - Kernel test frameworks - Kselftest & KUnit
  - Developer testing
    - Kselftest, KUnit and others.
  - Regression testing
    - Kselftest, KUnit and others.

- Linux kernel testing and validation
  - Continuous Integration testing
    - Static analysis tools (sparse, smatch, coccicheck etc.)
    - Dynamic analysis tools (fuzzers, syzbot etc.)

- Where does this all happen?
  - Developer test systems
  - Continuous Integration Rings
    - Kernel CI Dashboard — Home
    - 0-Day - Boot and Performance issues
    - 0-Day - Build issues
    - Linaro QA
    - Buildbot
    - Hulk Robot

- What is tested?
  - Kernel repositories:
    - linux mainline
    - linux-next
    - developer git repositories
  - Active kernel releases

- Basic testing
  - Boot and usage test
  - Run basic sanity tests

- Basic sanity tests
  - Does networking (wifi/wired) work correctly?
  - Does ssh work?
  - rsync a large file(s) from another system
  - Download files: wget, ftp, git clone etc.
  - Play audio/video

- Examine kernel logs
  - Look for new critical and error messages
  - Check for new warning messages
  - Check for panic traces

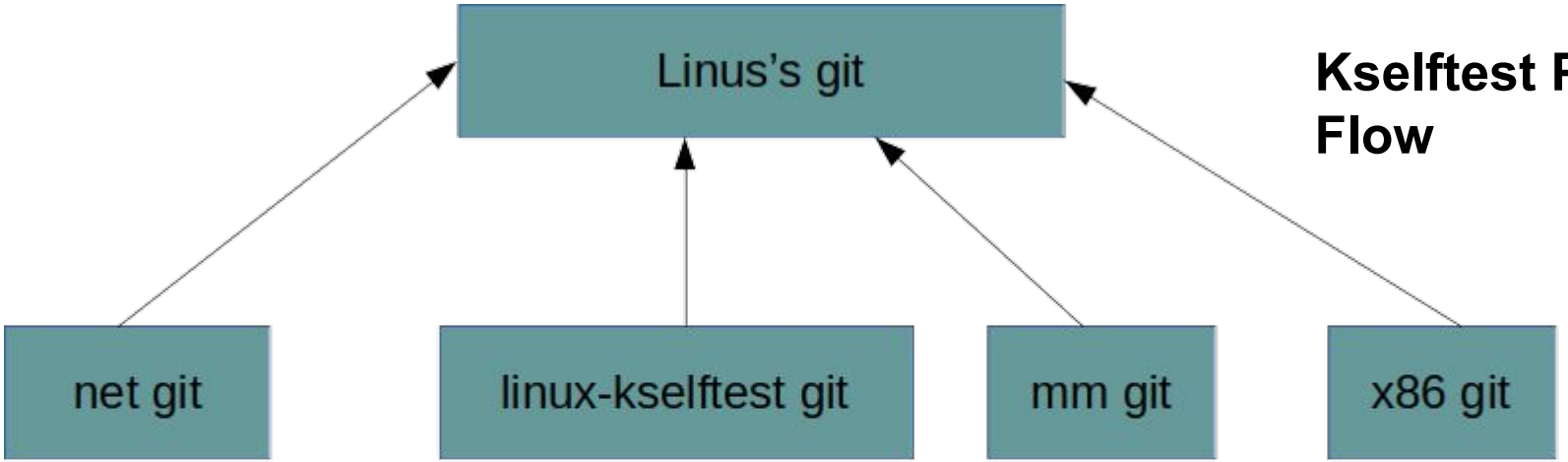
- Kernel selftest (Kselftest)
  - Regression test suite
    - Kernel developers and users
  - User-space applications
    - Shell scripts
    - C programs
    - Test modules

- Kernel selftest (Kselftest)
  - White and black box tests
  - Unit and functional tests
  - Hardware dependent
  - Stress and performance

- Kernel selftest (Kselftest)
  - Feature functional and regression tests
  - Bug fix focused regression tests
  - Subsystem tests
  - Breadth and depth coverage (error paths etc.)
  - No workload or application tests

LF live MENTORSHIP SERIES

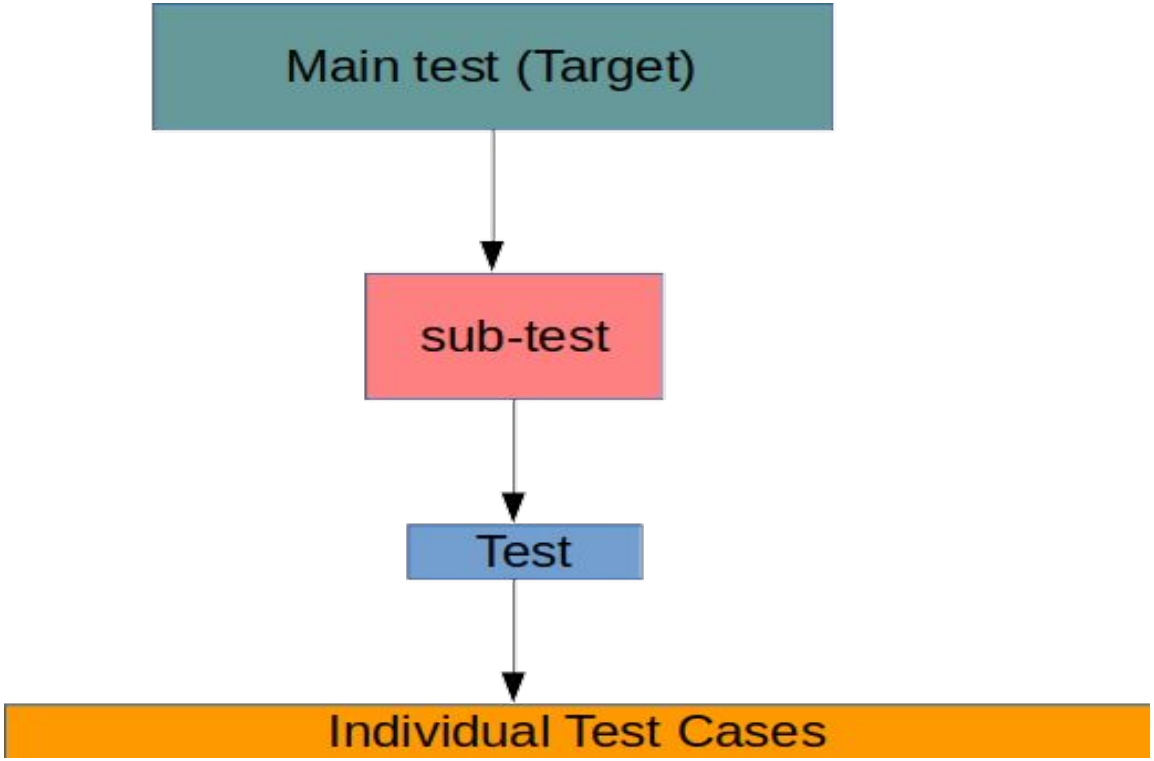
Kselftest Patch Flow



Patches

- Kselftest components
  - Kselftest framework
  - Common build/install/run infrastructure
  - Individual tests

Individual test

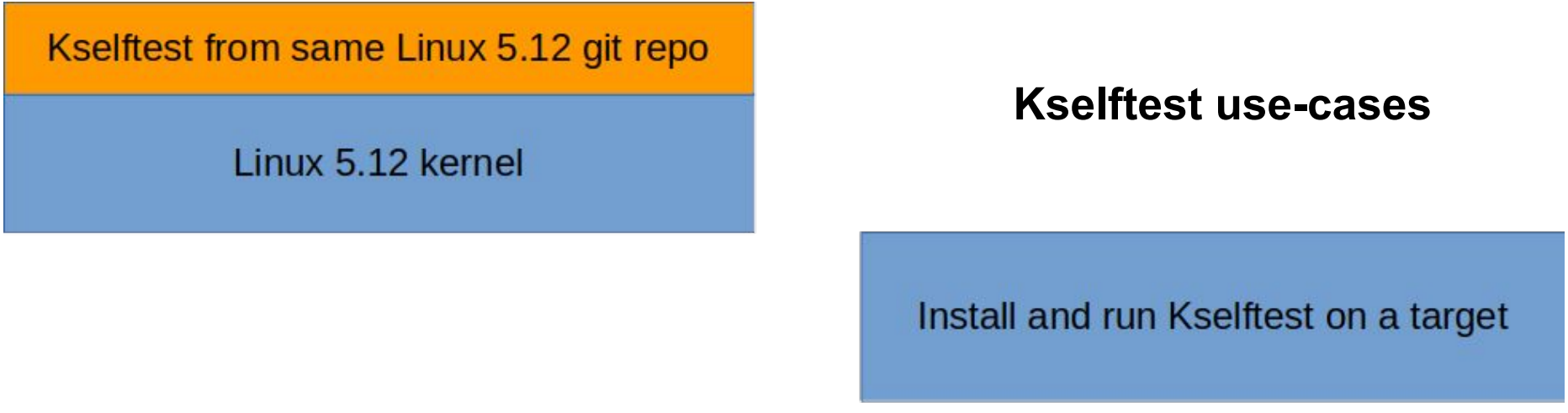


- Kselftest framework
  - C Interfaces for reporting test results
    - Pass/Fail/Skip counts
    - Pass/Fail/Skip messages
    - Uses Test Anything Protocol (Simple Text based)
  - Test harness for running tests
    - Run test and compare for expected result

- Kselftest common infrastructure
  - Build, run, install interfaces (Makefile based)
  - Run script generation (run and install)
  - Install and packaging tools

- Kselftest default run
  - Runs all targets slated for default run
  - Runs stress tests that support default mode
    - e.g: hotplug (cpu, memory) and pstore test support default and stress modes.
  - Run as root for best coverage
  - Run mainline tests on stable (if possible)

Kselftest use-cases



- Kselftest stress tests
  - Could change system state
  - Could require reboot in some cases
  - Experiment on a virtual machine
  - Some tests e.g: hotplug (cpu, memory) and pstore test support default and stress modes.

- Kselftest challenges
  - Developer and user requirements
  - Coverage and run-time
  - Driver coverage
  - Error path coverage
  - Improve common framework & infrastructure

- Kselftest git
  - Branches:
    - next
    - Fixes
  - linux-kselftest@vger.kernel.org
  - linux-kselftest git
  - Patchwork - linux-kselftest

- Sources
  - tools/testing/selftests
- Documentation
  - Documentation/dev-tools/kselftest.rst

- Building tests
  - make --silent kselftest-all
  - make --silent kselftest-all TARGETS=”timers size”
  - Installs kernel headers in repo for building tests
  - Supports cross-compile and relocatable builds
  - Build continues even when some tests fail to build

- kselftest_deps.sh
  - Usage: ./kselftest_deps.sh -[p] <compiler> [test_name]
  - kselftest_deps.sh [-p] gcc
  - kselftest_deps.sh [-p] gcc vm
  - kselftest_deps.sh [-p] aarch64-linux-gnu-gcc
  - kselftest_deps.sh [-p] aarch64-linux-gnu-gcc vm

- kselftest_deps.sh
  - Should be run in selftests directory in the kernel repo.
  - Checks if Kselftests can be built/cross-built on a system.
  - Parses all test/sub-test Makefile to find library dependencies.
  - Runs compile test on a trivial C file with LDLIBS specified in the test Makefiles to identify missing library dependencies.

- kselftest_deps.sh
  - Prints suggested target list for a system filtering out tests failed the build dependency check from the TARGETS in Selftests main Makefile when optional -p is specified.
  - Prints pass/fail dependency check for each tests/sub-test.
  - Prints pass/fail targets and libraries.
  - Default: runs dependency checks on all tests.
  - Optional test name can be specified to check dependencies for it.

- Running tests
  - make --silent kselftest
  - make --silent kselftest TARGETS=”timers size”

- Running stress tests
  - make –silent -C tools/testing/selftests run_hotplug
  - make –silent -C tools/testing/selftests run_pstore_crash
  - make –silent -C tools/testing/selftests/timers run_destructive_tests

- Reporting - detailed vs. summary
  - Default mode is detailed.
    - Includes individual test results
  - Summary mode reports just the pass/fail status
    - make –silent summary=1 kselftest

- Install
  - make kselftest-install
- Clean
  - make kselftest-clean
- Packaging
  - make gen_tar (in tools/testing/sefltests dir)

LF Live MENTORSHIP SERIES

- Let’s look at how it works ...

- Contributing new tests
  - Report pass/fail/skip conditions
    - Leverage kselftest framework
  - Skip and keep going
    - Unsupported features and configs

- Hooking into framework
  - Add new test to selftests/Makefile target list
  - Test: Makefile, config, shell script or C
  - Leverage framework for build/run/install
    - Check lib.mk for details
  - Override and custom builds are supported
    - Use them only if necessary

- What can you do to help:
  - Run kselftest to validate kernels
  - Write new tests
  - Enhance existing tests
  - Review tests

LF live MENTORSHIP SERIES

Thank you for joining us today!

We hope it will be helpful in your journey to learning more about effective and productive participation in open source projects. We will leave you with a few additional resources for your continued learning:

- The LF Mentoring Program is designed to help new developers with necessary skills and resources to experiment, learn and contribute effectively to open source communities.
- Outreachy remote internships program supports diversity in open source and free software
- Linux Foundation Training offers a wide range of free courses, webinars, tutorials and publications to help you explore the open source technology landscape.
- Linux Foundation Events also provide educational content across a range of skill levels and topics, as well as the chance to meet others in the community, to collaborate, exchange ideas, expand job opportunities and more. You can find all events at events.linuxfoundation.org.