The Integration of Rust with Kernel Testing Service

Yujie Liu, Philip Li
0-Day CI Team, Intel

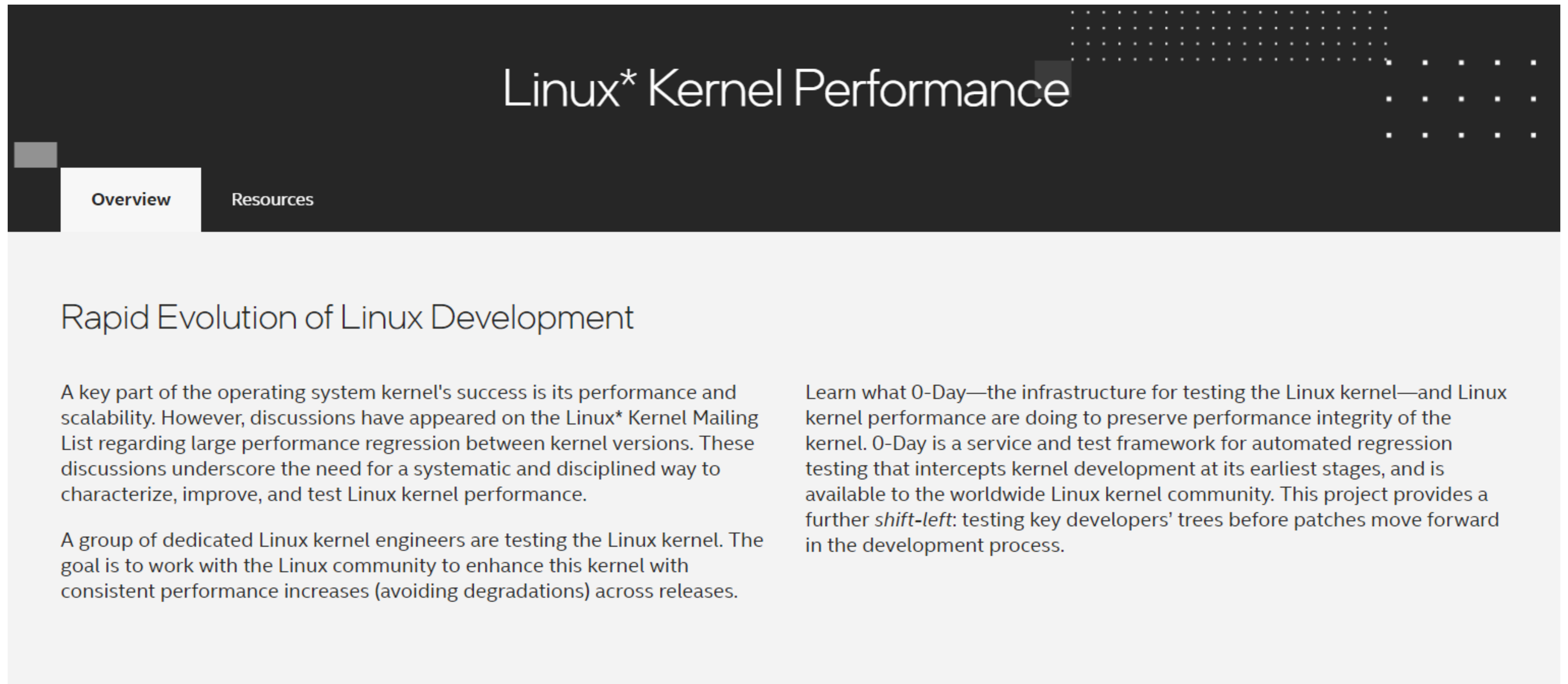
Linux Plumbers Conference | Dublin, Ireland Sept. 12-14, 2022

Contents

- Brief Introduction of 0-Day CI
- Rust Toolchain Setup
- Linux Repo Coverage
- Build Testing
- Future work to be done

Brief Introduction of 0-Day CI

- **An automated test service for Linux kernel, also known as LKP, kernel test robot**
- **Comprehensive test coverage**
  - Build, boot, functional, performance, static analysis, fuzzing
  - Compiler: gcc, clang, rustc
- **One of the top bug reporters**
- **Website**
  - **https://01.org/blogs/2018/0-day-ci-test**
  - **https://01.org/lkp**



Rapid Evolution of Linux Development

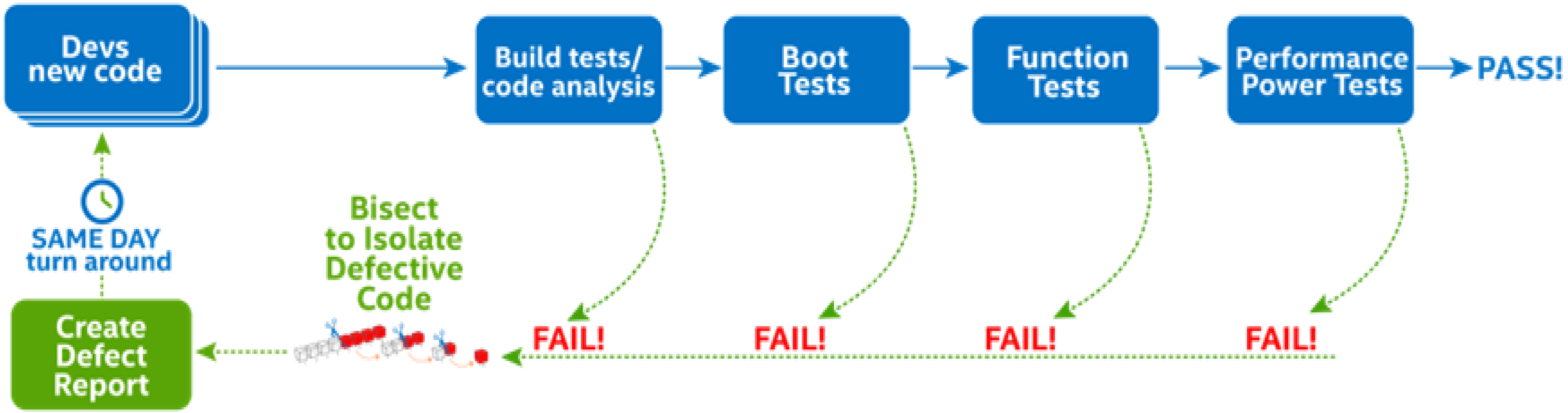
A key part of the operating system kernel's success is its performance and scalability. However, discussions have appeared on the Linux* Kernel Mailing List regarding large performance regression between kernel versions. These discussions underscore the need for a systematic and disciplined way to characterize, improve, and test Linux kernel performance.

A group of dedicated Linux kernel engineers are testing the Linux kernel. The goal is to work with the Linux community to enhance this kernel with consistent performance increases (avoiding degradations) across releases.

Learn what 0-Day—the infrastructure for testing the Linux kernel—and Linux kernel performance are doing to preserve performance integrity of the kernel. 0-Day is a service and test framework for automated regression testing that intercepts kernel development at its earliest stages, and is available to the worldwide Linux kernel community. This project provides a further *shift-left*: testing key developers' trees before patches move forward in the development process.

Brief Introduction of 0-Day CI

- Bisection-oriented Testing
  - Not only catch an issue, but also bisect to find the first commit that introduced the issue



Rust Toolchain Setup

- A combination of Rust compiler (rustc) and binding generator (bindgen)
- Regular upgrade according to min-tool-version.sh
- Adaptive toolchain version switching during bisection process

```
rustc-1.58.0-bindgen-0.56.0
rustc-1.59.0-bindgen-0.56.0
rustc-1.60.0-bindgen-0.56.0
rustc-1.61.0-bindgen-0.56.0
rustc-1.62.0-bindgen-0.56.0
…
```

Linux Repo Coverage

- **Rust-for-Linux repo**
  - https://github.com/Rust-for-Linux/linux
  - Branch: rust, rust-next
- **linux-next repo**
  - https://git.kernel.org/pub/scm/linux/kernel/git/next/linux-next.git
  - Branch: master

Build Testing

- **Random config coverage**
  - Rust toolchain added into PATH, so Rust related configs can be enabled by randconfig to provide partial coverage
- **Rust samples coverage**
  - Enable all the Rust samples in samples/rust

```
config SAMPLE_RUST_MINIMAL
config SAMPLE_RUST_PRINT
config SAMPLE_RUST_MODULE_PARAMETERS
config SAMPLE_RUST_SYNC
config SAMPLE_RUST_CHRDEV
config SAMPLE_RUST_MISCDEV
config SAMPLE_RUST_STACK_PROBING
config SAMPLE_RUST_SEMAPHORE
config SAMPLE_RUST_SEMAPHORE_C
config SAMPLE_RUST_RANDOM
config SAMPLE_RUST_PLATFORM
config SAMPLE_RUST_FS
config SAMPLE_RUST_NETFILTER
config SAMPLE_RUST_ECHO_SERVER
config SAMPLE_RUST_HOSTPROGS
config SAMPLE_RUST_SELFTESTS
```

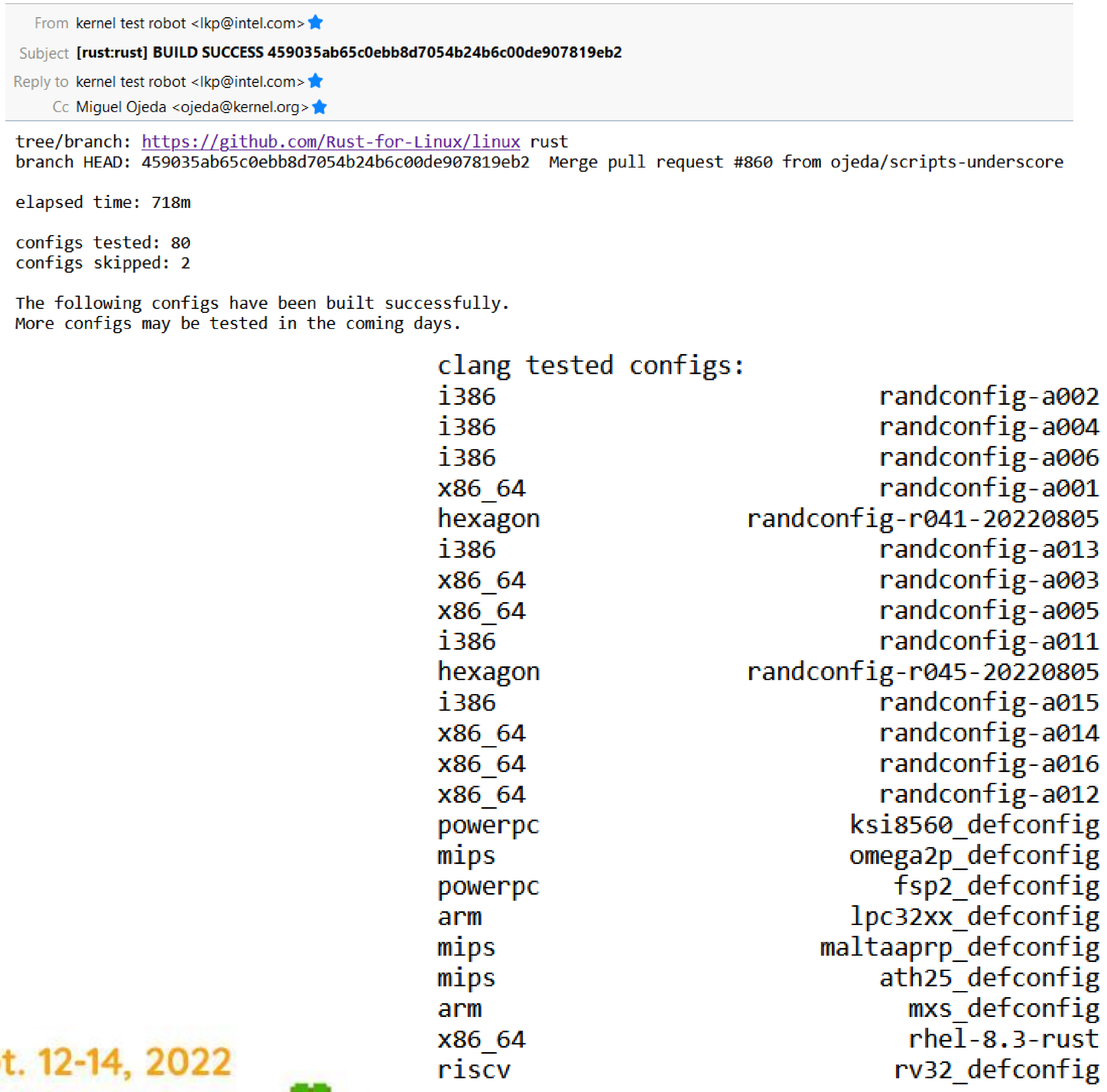
Build Testing

- **Catch rustc, bindgen or C code errors/warnings**

```
rust/helpers.c:22:17: warning: no previous prototype for function 'rust_helper_BUG' [-Wmissing-prototypes], err: false
rust/helpers.c:27:6: warning: no previous prototype for function 'rust_helper_clk_disable_unprepare' [-Wmissing-prototypes], err: false
rust/helpers.c:33:5: warning: no previous prototype for function 'rust_helper_clk_prepare_enable' [-Wmissing-prototypes], err: false
rust/helpers.c:39:15: warning: no previous prototype for function 'rust_helper_copy_from_user' [-Wmissing-prototypes], err: false
…

error[E0428]: the name `maple_enode` is defined multiple times
   --> /kbuild/obj/consumer/x86_64-rhel-8.3-rust/rust/bindings_generated.rs:34604:1
      |
34601 | pub struct maple_enode {
      | ---------------------- previous definition of the type `maple_enode` here
...
34604 | pub type maple_enode = *mut maple_enode;
      | ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ `maple_enode` redefined here
      |
      = note: `maple_enode` must be defined only once in the type namespace of this module
...
```

Build Testing

- Build Summary Report
- Triggered when the branch is updated
- Build the head under various arch and configs (allyes, allno, allmod, def, rand, -rust, …)
- Usually send out the report within one day



From kernel test robot <lkp@intel.com>

Subject **[rust:rust] BUILD SUCCESS 459035ab65c0ebb8d7054b24b6c00de907819eb2**

Reply to kernel test robot <lkp@intel.com>

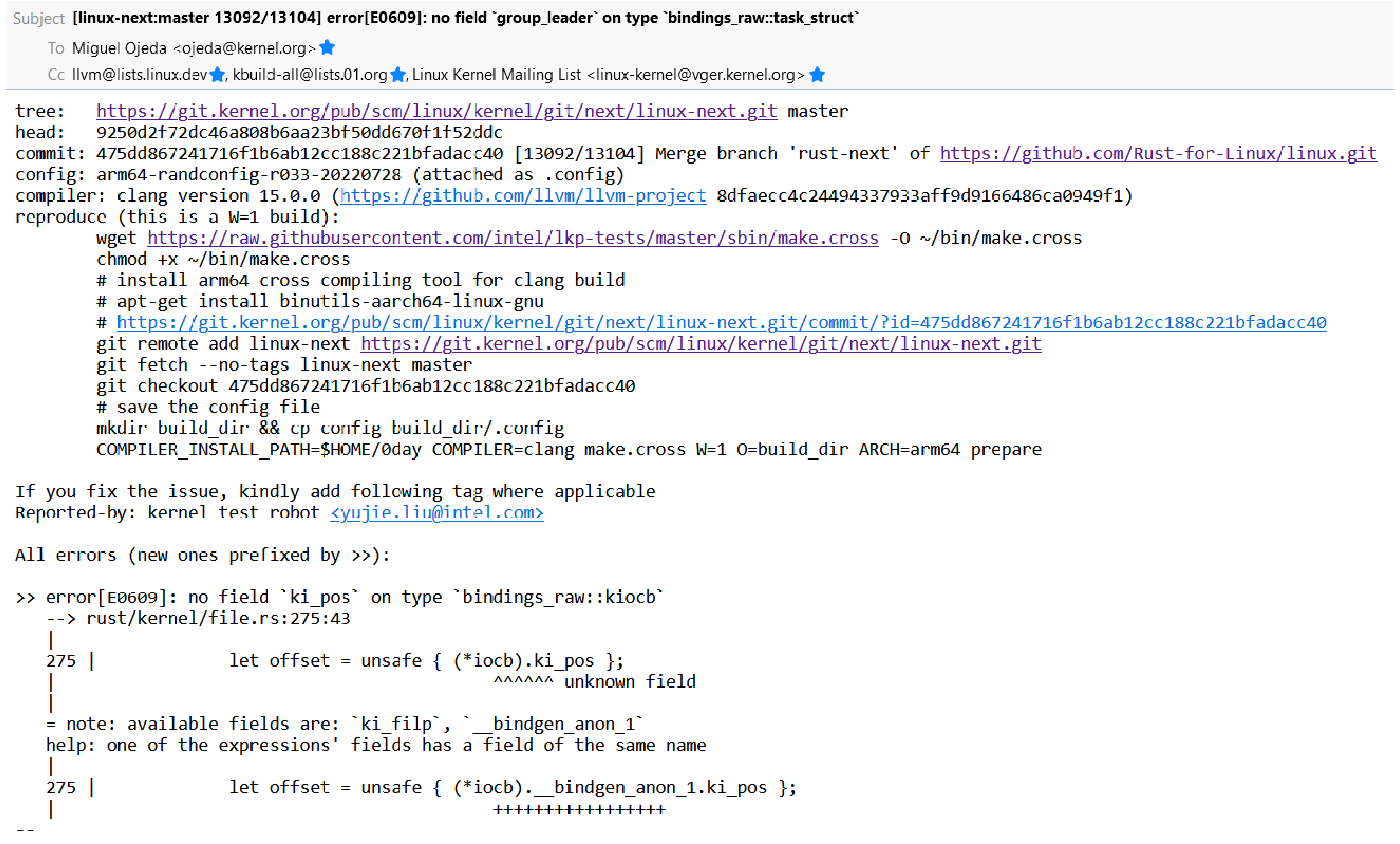
Cc Miguel Ojeda <ojeda@kernel.org>

```
tree/branch: https://github.com/Rust-for-Linux/linux rust
branch HEAD: 459035ab65c0ebb8d7054b24b6c00de907819eb2  Merge pull request #860 from ojeda/scripts-underscore

elapsed time: 718m

configs tested: 80
configs skipped: 2

The following configs have been built successfully.
More configs may be tested in the coming days.
```

```
clang tested configs:
i386                          randconfig-a002
i386                          randconfig-a004
i386                          randconfig-a006
x86_64                        randconfig-a001
hexagon              randconfig-r041-20220805
i386                          randconfig-a013
x86_64                        randconfig-a003
x86_64                        randconfig-a005
i386                          randconfig-a011
hexagon              randconfig-r045-20220805
i386                          randconfig-a015
x86_64                        randconfig-a014
x86_64                        randconfig-a016
x86_64                        randconfig-a012
powerpc                      ksi8560_defconfig
mips                        omega2p_defconfig
powerpc                        fsp2_defconfig
arm                         lpc32xx_defconfig
mips                       maltaaprp_defconfig
mips                          ath25_defconfig
arm                             mxs_defconfig
x86_64                          rhel-8.3-rust
riscv                          rv32_defconfig
```

Build Testing

- Build Bisection Report
- Once any build errors/warnings are captured, will do bisection to find the first bad commit that introduced the issue
- The response time depends on the bisection process



Subject **[linux-next:master 13092/13104] error[E0609]: no field `group_leader` on type `bindings_raw::task_struct`**

To Miguel Ojeda <ojeda@kernel.org>

Cc llvm@lists.linux.dev, kbuild-all@lists.01.org, Linux Kernel Mailing List <linux-kernel@vger.kernel.org>

```
tree:   https://git.kernel.org/pub/scm/linux/kernel/git/next/linux-next.git master
head:   9250d2f72dc46a808b6aa23bf50dd670f1f52ddc
commit: 475dd867241716f1b6ab12cc188c221bfadacc40 [13092/13104] Merge branch 'rust-next' of https://github.com/Rust-for-Linux/linux.git
config: arm64-randconfig-r033-20220728 (attached as .config)
compiler: clang version 15.0.0 (https://github.com/llvm/llvm-project 8dfaecc4c24494337933aff9d9166486ca0949f1)
reproduce (this is a W=1 build):
        wget https://raw.githubusercontent.com/intel/lkp-tests/master/sbin/make.cross -O ~/bin/make.cross
        chmod +x ~/bin/make.cross
        # install arm64 cross compiling tool for clang build
        # apt-get install binutils-aarch64-linux-gnu
        # https://git.kernel.org/pub/scm/linux/kernel/git/next/linux-next.git/commit/?id=475dd867241716f1b6ab12cc188c221bfadacc40
        git remote add linux-next https://git.kernel.org/pub/scm/linux/kernel/git/next/linux-next.git
        git fetch --no-tags linux-next master
        git checkout 475dd867241716f1b6ab12cc188c221bfadacc40
        # save the config file
        mkdir build_dir && cp config build_dir/.config
        COMPILER_INSTALL_PATH=$HOME/0day COMPILER=clang make.cross W=1 O=build_dir ARCH=arm64 prepare

If you fix the issue, kindly add following tag where applicable
Reported-by: kernel test robot <yujie.liu@intel.com>

All errors (new ones prefixed by >>):

>> error[E0609]: no field `ki_pos` on type `bindings_raw::kiocb`
   --> rust/kernel/file.rs:275:43
       |
   275 |             let offset = unsafe { (*iocb).ki_pos };
       |                                           ^^^^^^ unknown field
       |
       = note: available fields are: `ki_filp`, `__bindgen_anon_1`
   help: one of the expressions' fields has a field of the same name
       |
   275 |             let offset = unsafe { (*iocb).__bindgen_anon_1.ki_pos };
       |                                           +++++++++++++++++
--
```

Future work to be done

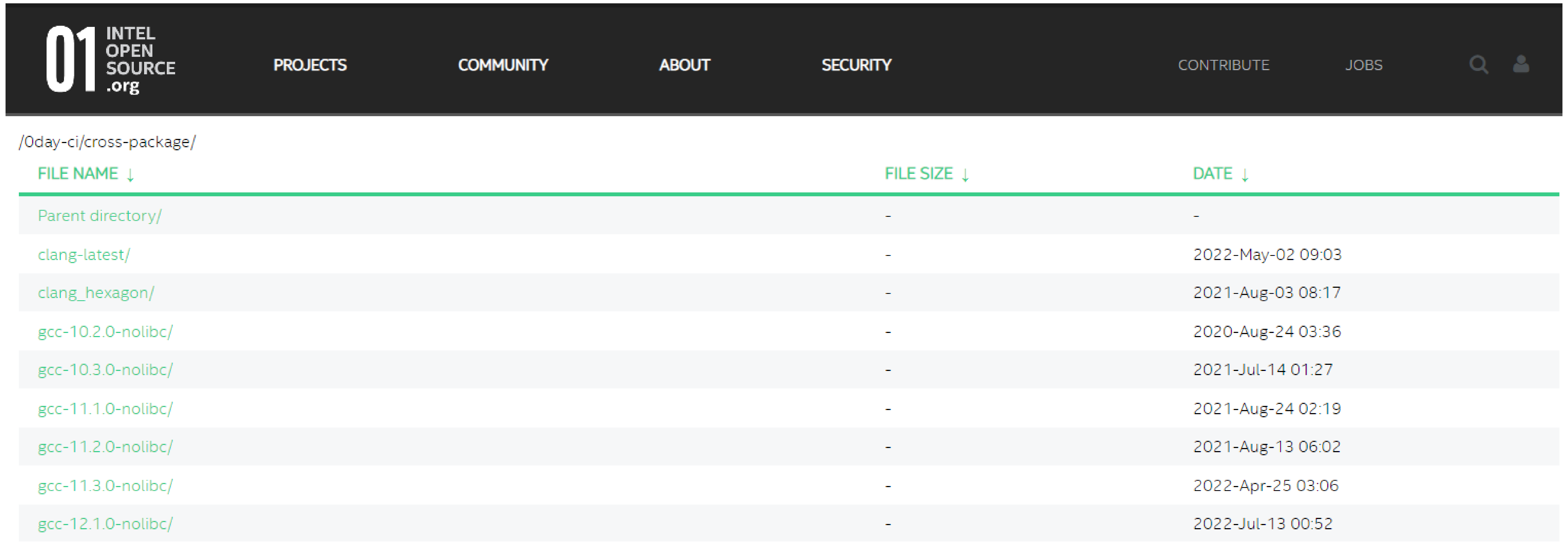
Future work to be done

- **Testing for Rust compiling configs**
  - Optimization level, overflow checks, build assert…

```
config RUST_DEBUG_ASSERTIONS
config RUST_OVERFLOW_CHECKS
config RUST_OPT_LEVEL_SIMILAR_AS_CHOSEN_FOR_C
config RUST_OPT_LEVEL_0
config RUST_OPT_LEVEL_1
config RUST_OPT_LEVEL_2
config RUST_OPT_LEVEL_3
config RUST_OPT_LEVEL_S
config RUST_OPT_LEVEL_Z
config RUST_BUILD_ASSERT_ALLOW
config RUST_BUILD_ASSERT_WARN
config RUST_BUILD_ASSERT_DENY
config RUST_KERNEL_KUNIT_TEST
```

Future work to be done

- **Support Rust in LKP tools to reproduce issues**
  - We have a make.cross tool to help user reproduce the issue locally
  - It will download toolchain from our website https://download.01.org/0day-ci/cross-package/
  - Support Rust in the tool to help user setup Rust env conveniently.



/0day-ci/cross-package/

FILE NAME ↓	FILE SIZE ↓	DATE ↓
Parent directory/	-	-
clang-latest/	-	2022-May-02 09:03
clang_hexagon/	-	2021-Aug-03 08:17
gcc-10.2.0-nolibc/	-	2020-Aug-24 03:36
gcc-10.3.0-nolibc/	-	2021-Jul-14 01:27
gcc-11.1.0-nolibc/	-	2021-Aug-24 02:19
gcc-11.2.0-nolibc/	-	2021-Aug-13 06:02
gcc-11.3.0-nolibc/	-	2022-Apr-25 03:06
gcc-12.1.0-nolibc/	-	2022-Jul-13 00:52

Future work to be done

- **Support Rust in LKP tools to reproduce issues**
  - We have a make.cross tool to help user reproduce the issue locally
  - It will download toolchain from our website https://download.01.org/0day-ci/cross-package/
  - Support Rust in the tool to help user setup Rust env conveniently.

```
wget https://raw.githubusercontent.com/intel/lkp-tests/master/sbin/make.cross -O ~/bin/make.cross
chmod +x ~/bin/make.cross
git remote add linux-next https://git.kernel.org/pub/scm/linux/kernel/git/next/linux-next.git
git fetch --no-tags linux-next master
git checkout 822a6200734c94ccbfc9ae7acc5da5505b991515
# save the config file
mkdir build_dir && cp config build_dir/.config
COMPILER_INSTALL_PATH=$HOME/0day COMPILER=clang make.cross W=1 O=build_dir ARCH=x86_64 SHELL=/bin/bash
```

Future work to be done

- **Functional testing**
  - Run Rust samples as testcases in CI

```
commit 91fb0182d4db1e0b0b2a1438f6bf699975fca96c
Author: Miguel Ojeda <ojeda@kernel.org>
Date:   Sat Jul 3 17:21:12 2021 +0200

    samples: add Rust examples

    A set of Rust modules that showcase how Rust modules look like
    and how to use the abstracted kernel features, as well as
    an example of a Rust host program with several modules.

    These samples also double as tests in the CI.

    The semaphore sample comes with a C version for comparison.
```

Future work to be done

- **Performance testing**
  - Comparison of Rust samples and corresponding C samples
  - Performance test for hardware drivers written in Rust

```
commit 91fb0182d4db1e0b0b2a1438f6bf699975fca96c
Author: Miguel Ojeda <ojeda@kernel.org>
Date:   Sat Jul 3 17:21:12 2021 +0200

    samples: add Rust examples

    A set of Rust modules that showcase how Rust modules look like
    and how to use the abstracted kernel features, as well as
    an example of a Rust host program with several modules.

    These samples also double as tests in the CI.

    The semaphore sample comes with a C version for comparison.
```

Future work to be done

- **Boot/fuzzing test for kernels including Rust code**
  - Rust code as built-in or module? Any influence on boot process?
  - Utilize fuzzing tools such as syzkaller to spot potential issues?
- **Rust kunit test, Rust kselftests**
  - config RUST_KERNEL_KUNIT_TEST
  - config SAMPLE_RUST_SELFTESTS

Future work to be done

- **Rust code coverage data**
  - gcov for Rust?

Thanks!

Q & A

References and Links

- https://github.com/Rust-for-Linux/linux/blob/rust/Documentation/rust/quick-start.rst
- https://doc.rust-lang.org/book/
- https://doc.rust-lang.org/nightly/rustc/platform-support.html
- https://lore.kernel.org/all/202201310402.vCWP8CUS-lkp@intel.com/
- https://lore.kernel.org/all/383b1045-94c5-c2b0-57db-9f4f4760206c@intel.com/
- https://lore.kernel.org/all/e5c7aa10-590d-0d20-dd3b-385bee2377e7@intel.com/
- https://lore.kernel.org/all/202202081303.QEI35DwC-lkp@intel.com/
- https://lore.kernel.org/all/202202070448.t4dQz3iS-lkp@intel.com/
- https://lore.kernel.org/all/202204030402.Ps8X1oHd-lkp@intel.com/

Notices & Disclaimers