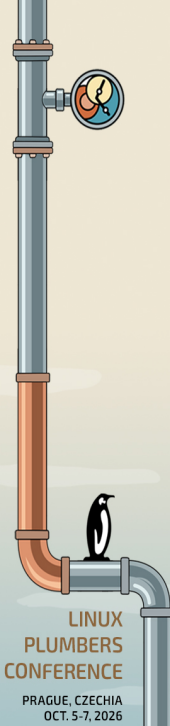


Rethinking KUnit Configuration

David Gow [david@davidgow.net]

6th October 2026

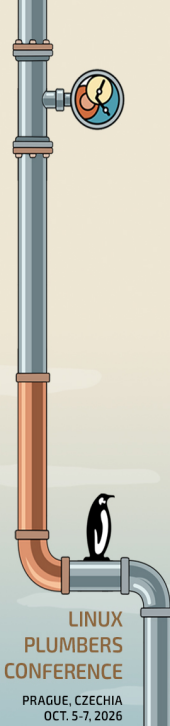


LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

What is KUnit?

- An in-kernel unit test framework.
- A tool — `kunit.py` — which configures, compiles, and runs tests (and parses the results)

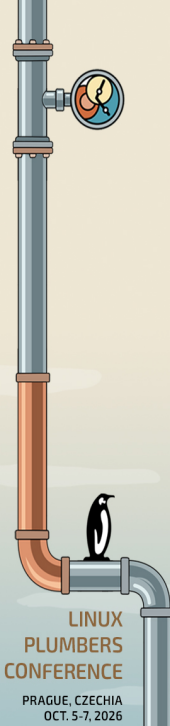


LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

A quick show of hands...

- Do you use KUnit?

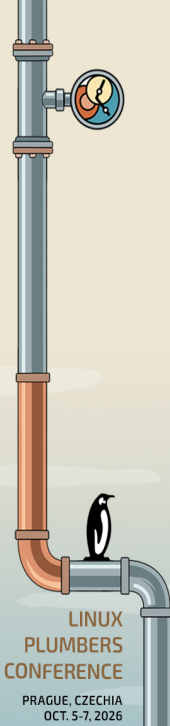


LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

A quick show of hands...

- Do you use KUnit?
- Do you use kunit.py?

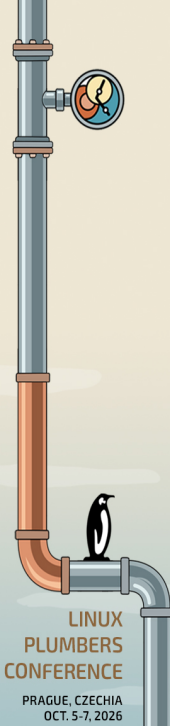


LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

A quick show of hands...

- Do you use KUnit?
- Do you use `kunit.py`?
- Do you use a `.kunitconfig` file?

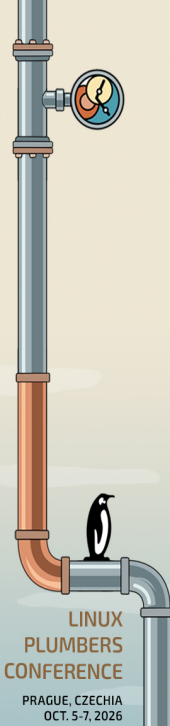


LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

A quick show of hands...

- Do you use KUnit?
- Do you use `kunit.py`?
- Do you use a `.kunitconfig` file?
- Do you share a configuration (`.kunitconfig` or `kunit.py` invocation) with others?

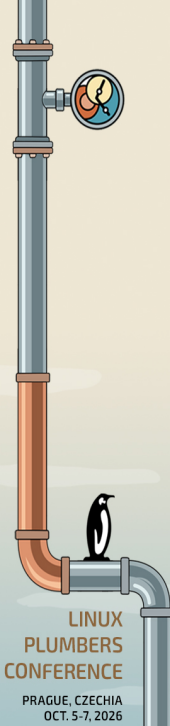


LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

A quick show of hands...

- Do you use KUnit?
- Do you use `kunit.py`?
- Do you use a `.kunitconfig` file?
- Do you share a configuration (`.kunitconfig` or `kunit.py` invocation) with others?
- Do you have a script which runs KUnit for you?

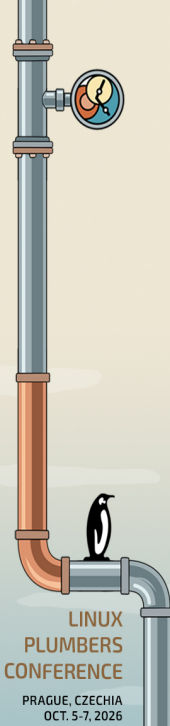


LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

A quick show of hands...

- Do you use KUnit?
- Do you use `kunit.py`?
- Do you use a `.kunitconfig` file?
- Do you share a configuration (`.kunitconfig` or `kunit.py` invocation) with others?
- Do you have a script which runs KUnit for you?
- Do you run multiple different KUnit configurations (for testing the same thing?)



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

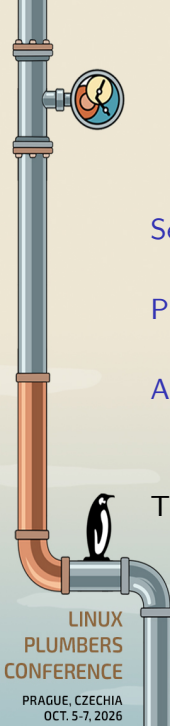
Why do we want to configure it?

Select which tests to run e.g. The **ext4** tests, the **USB** tests, or **all the tests available**.

Provide a test environment e.g. Under **UML**, in a **32-bit powerpc VM**, on a multiprocessor setup, etc.

Adapt the test runs to the user's system and preferences e.g. With results in **XML** format in /tmp, skipping slow tests, with this compiler path, etc.

These can overlap: architecture, for instance, can be important for all three.



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

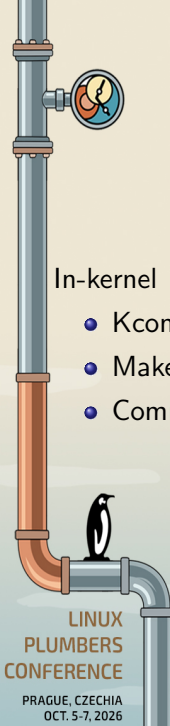
How do we configure it?

In-kernel

- Kconfig options
- Makefile options
- Command-line arguments

kunit.py

- .kunitconfig files
- qemu_configs (architecture config python)
- Command-line arguments

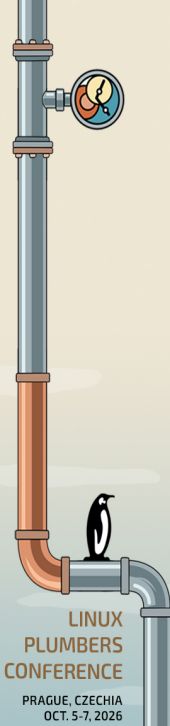


LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

What's wrong with this?

- Not everything can be configured via Kconfig options
- `.kunitconfig` files can *only* change Kconfig options
- `qemu_config` files are rarely used, and difficult to override
- Command-line arguments are difficult to share
- Each config only configures a single run (so running multiple tests requires multiple `kunit.py` invocations)
- All of these formats are more different than necessary

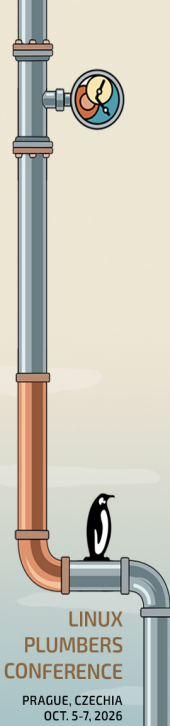


LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

How can we improve it?

- Standardise on a single configuration mechanism which:
 - ▶ Supports more than just Kconfig options
 - ▶ Can be composed
 - ▶ Can be shared between developers
- Support running multiple configurations from a single `kunit.py` invocation
- Provide a way of passing metadata through to the results



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

Format 1: .kunitconfig

fs/fat/.kunitconfig

```
CONFIG_KUNIT=y
CONFIG_FAT_FS=y
CONFIG_MSDOS_FS=y
CONFIG_VFAT_FS=y
CONFIG_FAT_KUNIT_TEST=y
```

- Just Kconfig entries
 - ▶ To enable the feature being tested (FAT_FS et al)
 - ▶ To enable the KUnit framework (KUNIT=y)
 - ▶ To enable the specific test suites (FAT_KUNIT_TEST=y)
- Of the form CONFIG_...=value
- Or a comment for =n: # CONFIG_... is not set
- Easy to combine so long as there aren't conflicting options
- Plenty of scope for other options which don't start with CONFIG_

Format 2: qemu_config.py

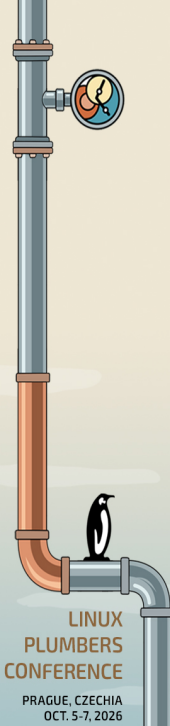
tools/testing/kunit/qemu_configs/m68k.py

```
# SPDX-License-Identifier: GPL-2.0-only
from ..qemu_config import QemuArchParams

QEMU_ARCH = QemuArchParams(
    linux_arch='m68k',
    kconfig='''CONFIG_VIRT=y''',
    qemu_arch='m68k',
    kernel_path='vmlinux',
    kernel_command_line='console=hvc0',
    extra_qemu_params=['-machine', 'virt'])
```

Format 2: qemu_config.py

- Can configure:
 - ▶ Linux architecture (ARCH=)
 - ▶ Kconfig entries
 - ▶ Qemu (arch/binary name) and extra parameters
 - ▶ Kernel path and command-line
- Cannot configure other, more-user-level kunit.py options, e.g.
 - ▶ Output format, path, etc.
 - ▶ Other make options (e.g. LLVM=1)
- Turing-complete, real Python
- Not designed to be combined / overridden.
- UML (--arch=um) special-cased



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

Format 3: kunit.py command-line

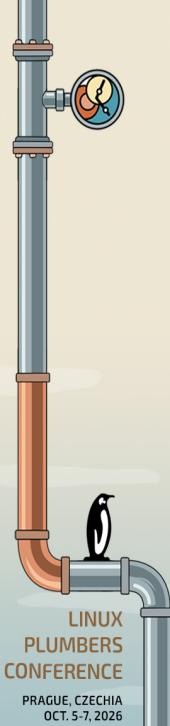
\$

```
./tools/testing/kunit/kunit.py run  
--arch um  
--make_options LLVM=1  
--kconfig_add CONFIG_RUST=y  
--json=/tmp/test_output.json
```

- Capable of handling all different config options
- Available options depend on action (config, build, exec, parse, run)
- Not parsed from a file (but you can call `kunit.py` from a script)

The Proposal

- One configuration format to rule them all
 - ▶ Create a new one?
 - ▶ Extend an existing one?
- Merge / override configurations
 - ▶ Currently possible (to some extent) with the command-line
 - ▶ Support, e.g., Test ext4 on powerpc with David's preferences
- Multiple test runs per `kunit.py` execution
 - ▶ Run the same tests on several different architectures both with and without **KASAN**, etc.



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

A unified format: extend .kunitconfig

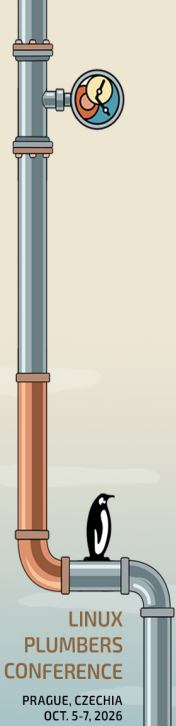
- Backwards-compatible with existing .kunitconfig files
(Just use lines starting with something other than CONFIG_)
- Needs some re-work, as .kunitconfig parser linked specifically to the *config* pass.
- Potentially support for preprocessor-style `#includes`.

Example

```
QEMU_ARCH=i386
QEMU_OPTIONS+="-smp=4"
MAKE_OPTIONS+="LLVM=1"
KERNEL_CMDLINE+="console=ttyS0"
CONFIG_KUNIT=y
```

A unified format: something with python

- Turing complete for more complicated configs.
- Either build on `qemu_config`, which sets a big struct...
- ...or provide an API with things like `set_kconfig()` or `add_make_option()`
- Harder to parse, more heavily tied to the `kunit.py` implementation



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

Python Example 1

Example 1

```
KUNITCONFIG = KunitRequest(  
    make_options=['LLVM=1', 'ARCH=i386']  
    kconfig_entries=[  
        'CONFIG_KUNIT=y',  
        'CONFIG_KUNIT_ALL_TESTS=y'],  
    qemu_arch='i386',  
    arch_kernel_path='vmlinux',  
    kernel_params=['console=ttyS0'],  
    qemu_params=['-smp=4'])
```

Note that most options are now arrays, which can be combined. There are also a lot of options which will be set to None by default.

Python Example 2

Example 2

```
kunit_config.add_make_option("LLVM", "1")
kunit_config.add_make_option("ARCH", "i386")
kunit_config.set_kconfig("KUNIT", "y")
kunit_config.set_kconfig("KUNIT_ALL_TESTS", "y")
kunit_config.set_qemu_arch("i386")
kunit_config.set_kernel_path("vmlinux")
kunit_config.add_qemu_param("-smp=4")
```

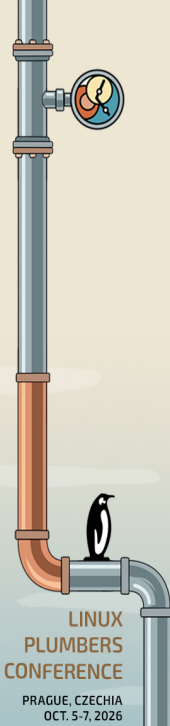
This would make it easier to catch conflicting options when they're set, and avoid having huge partially-filled config structs.

LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

Which of these seems best?

- ① Enhanced `.kunitconfig`
- ② Python script
 - ① Option 1: struct
 - ② Option 2: calls
- ③ Leave it alone / Something else

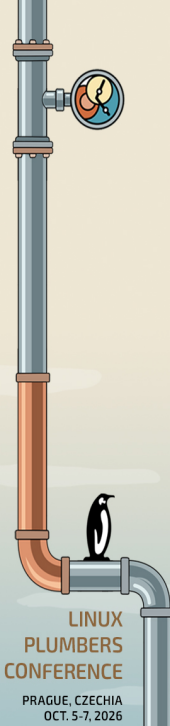


LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

Bonus 1: Running multiple configs

- Two interpretations of being passed multiple configs:
 - ① Combine the options (multiple `--kunitconfig args`)
 - ② Do multiple runs (multiple `kunit.py` executions)
- UI: how do you specify each?
What about multiple runs, each of multiple configs?
- How do you handle results?
Does each config need to specify a separate output file, or do we combine them?

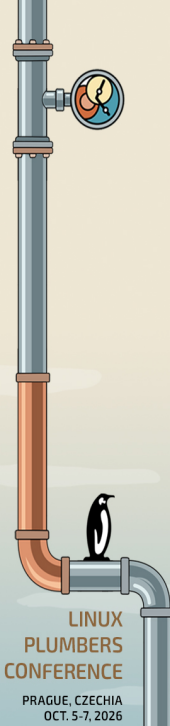


LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

Bonus 2: Passing through metadata

- KTAP (and other format) results can contain metadata
- Some of this is auto-generated, but some could be passed through from the config.
e.g. name of the config / user / branch / etc.
- Some of these could always be applied (config file paths, kernel version, etc)
- This could be a way to differentiate results when multiple runs are happening



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026