How to write your first kernel selftest

Michael Ellerman LCA 2021

IBM

`https://michael.ellerman.id.au/files/lca2021.pdf`

What are they

Small userspace programs, that test part of the kernel.

Mostly written in C, but some are just shell scripts.

Wide range of types & complexity.

Where

Kept in the kernel tree.

linux/tools/testing/selftests/*

Not an external project: make it easy to add tests.

Cover many different subsystems and architectures.

Why

```
Author: Frederic Weisbecker <fweisbec@gmail.com>
Date:   Thu Jan 12 17:20:44 2012 -0800

selftests: new very basic kernel selftests directory

Bring a new kernel selftests directory in
tools/testing/selftests.
...

This can help centralizing and maintaining any useful selftest
that developers usually tend to let rust in peace on some random
server.
```

Why

Make it easy for kernel developers to add tests.

Allows kernel devs to merge tests simultaneously with fixes.

Low barrier to entry, collect tests that don't fit elsewhere.

How to build them

```
$ git clone https://git.kernel.org/.../torvalds/linux.git
$ cd linux/tools/testing/selftests
$ make install
...
{ probably some errors, but build continues }
$ cd kselftest_install
```

How to run a test

```
$ ./run_kselftest.sh -t proc:self
TAP version 13
1..1
# selftests: proc: self
ok 1 selftests: proc: self

Or
$ cd proc; ./self
```

How to run them all

Run all the tests that were successfully built:

```
$ ./run_kselftest.sh -s
```

Some could crash your machine (unlikely). Run in a VM to be safe.

Some won’t run unless you’re root.

Some require certain config options or hardware etc.

Some can take quite a while.

A simple selftest

```
int main(void)
{
    char buf1[64], buf2[64];
    pid_t pid;
    ssize_t rv;

    pid = sys_getpid();
    snprintf(buf1, sizeof(buf1), "%u", pid);

    rv = readlink("/proc/self", buf2, sizeof(buf2));
    assert(rv == strlen(buf1));
    buf2[rv] = '\0';
    assert(streq(buf1, buf2));

    return 0;
}
```

Another short but useful test (from fa10fed30f25)

```
int main(void)
{
	struct stat proc_st1, proc_st2;
	char procbuff[] = "/tmp/proc.XXXXXX/meminfo";
	char procdir1[] = "/tmp/proc.XXXXXX";
	char procdir2[] = "/tmp/proc.XXXXXX";

	assert(mkdtemp(procdir1) != NULL);
	assert(mkdtemp(procdir2) != NULL);

	assert(!mount("proc", procdir1, "proc", 0, "hidepid=1"));
	assert(!mount("proc", procdir2, "proc", 0, "hidepid=2"));

	snprintf(procbuff, sizeof(procbuff), "%s/meminfo", procdir1);
	assert(!stat(procbuff, &proc_st1));

	snprintf(procbuff, sizeof(procbuff), "%s/meminfo", procdir2);
	assert(!stat(procbuff, &proc_st2));
	 ...
	assert(proc_st1.st_dev != proc_st2.st_dev);

	return 0;
}
```

Adding a simple selftest

```
$ cd linux/tools/testing/selftests

$ mkdir lca2021

$ cd lca2021

$ cat > Makefile <<EOF
TEST_GEN_PROGS := lca2021
CFLAGS += -Wall

include ../lib.mk
EOF

$ echo lca2021 > .gitignore
```

```
$ cat > lca2021.c <<EOF
#include <stdio.h>

int main(void) {
        printf("LCA 2021!\n");
        return 0;
}
EOF


$ make
gcc -Wall   lca2021.c  ...

$ ./lca2021
LCA 2021!
```

Integrating a simple selftest

Add `lca2021` to the list of targets in the selftests Makefile:

```
$ cd linux/tools/testing/selftests

$ patch Makefile
27a28
> TARGETS += lca2021

$ make install
...
```

```
$ cd kselftest_install

$ ./run_kselftest.sh -t lca2021:lca2021

TAP version 13

1..1

# selftests: lca2021: lca2021

# LCA 2021!

ok 1 selftests: lca2021: lca2021
```

Ideas for tests

- Every syscall could/should have at least one test?
- /sys and /proc - eg 4f134b89a24b ("lib/syscall: fix syscall registers retrieval on 32-bit platforms")
    - Content of files and relationships between objects?
    - Permission checks on sensitive files?
- Older features tend not to have tests
- Some newer features have no or few tests
    - IPC namespaces - currently only 1 or 2 tests
    - CGROUP namespaces - no tests?
- Newish syscalls with no selftests
    - process_madvise(), faccessat2(), fspick(), fsmount(), move_mount(), open_tree()
- Corner case / stress tests - think of a corner case and try to exercise it

Summing up

- Kernel selftests are not scary
  - If you can write C or shell, you can probably write a kernel selftest.
- Integrating a test should be quite simple
  - You may need to add your test to an existing directory of tests
- How to submit a test upstream?
  - It’s complicated 😁
  - Use `get_maintainer.pl`
  - Send them to `linux-kselftest@vger.kernel.org`
  - I’m happy to help, send me email: `selftests@ellerman.id.au`

Extras

More advanced Makefiles

Using a shell script:

```
TEST_PROGS := lca2021.sh

include ../lib.mk
```

Shell script wrapper with a C program:

```
TEST_PROGS := lca2021.sh
TEST_GEN_FILES := lca2021

include ../lib.mk
```

Per-target CFLAGS:

```
TEST_GEN_PROGS := lca2021

include ../lib.mk

$(OUTPUT)/lca2021: CFLAGS += -g
```

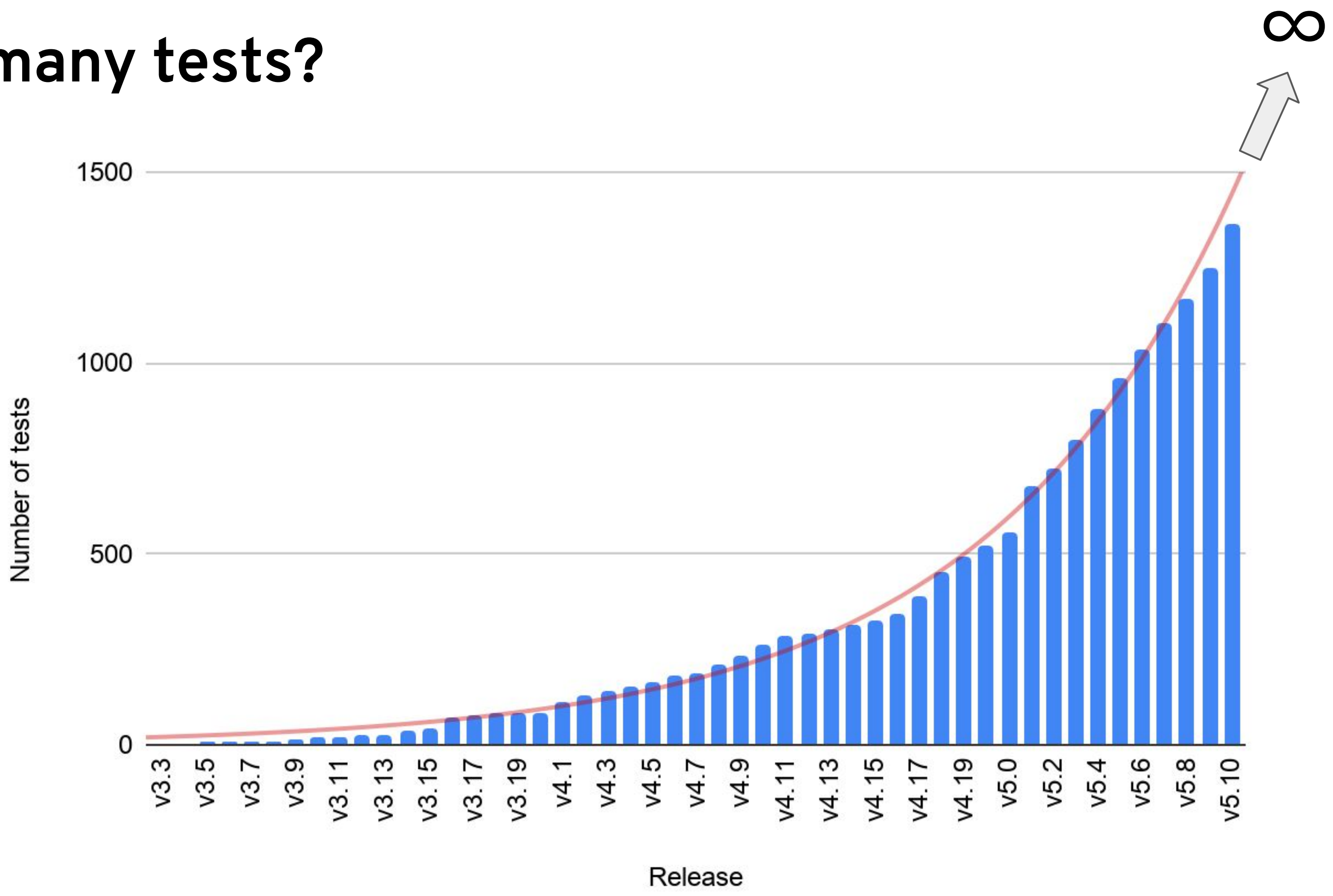
Share some code between all tests:

```
TEST_GEN_PROGS := lca2021 lca2022

include ../lib.mk

$(TEST_GEN_PROGS): helpers.c
```

How many tests?


∞
Number of tests
v3.3
v3.5
v3.7
v3.9
v3.11
v3.13
v3.15
v3.17
v3.19
v4.1
v4.3
v4.5
v4.7
v4.9
v4.11
v4.13
v4.15
v4.17
v4.19
v5.0
v5.2
v5.4
v5.6
v5.8
v5.10
Release