

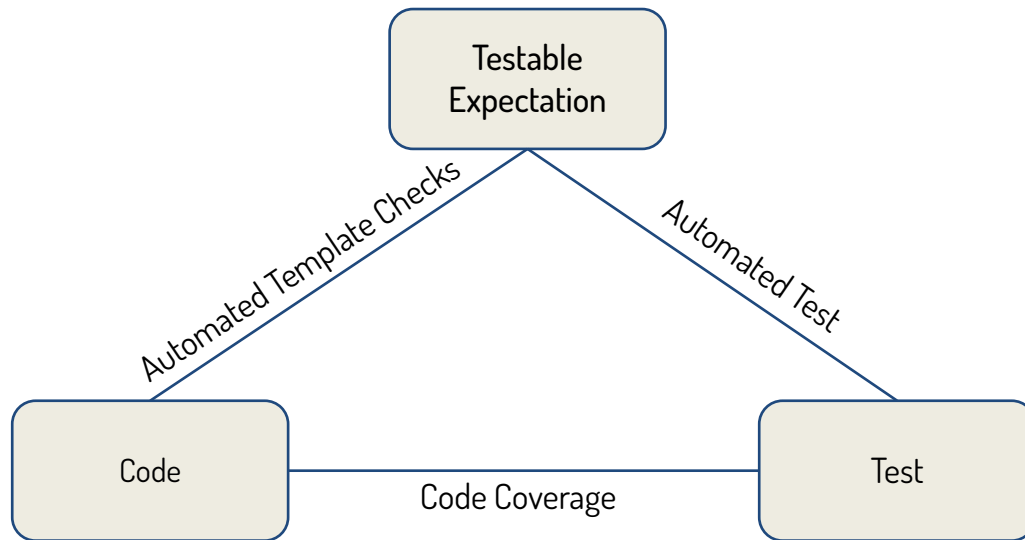
Scaling the Design Definition

Six examples where test and implementation did not reflect the same idea.

<https://github.com/chuckwolber/silent-regressions>

Chuck Wolber <chuck@wolber.net>

Virtuous Cycle



Bug Discovery (Validation Failure)

- Invalid or missing testable expectation.
- Invalid or missing test case.

Test Failure (Verification Failure)

- Invalid testable expectation.
- Code bug.

Verification: Does it do the thing right?

Validation: Does it do the right thing?

With requirement, test, and coverage in equilibrium, it becomes possible to assert that the code is an accurate reflection of developer intent.

Code and test silently diverge. How do we prevent this?

Two ways unchanged tests kept passing

271

days on average across all six cases

Range: 28 to 587 days

Misleading filesystem information

The test accepted an answer without checking the meaning of the returned information.

Inotify cookies, overlayfs hole discovery, overlayfs block size

Required operations never occurred

The test reached a passing result without proving the required operation took place.

DAX huge pages, tunnel PMTU feedback, pKVM single-stepping

Timing: Git committer dates, introduction or activation to the kernel fix.

Six selected cases. Rounded mean. This is not an estimate for the kernel as a whole.

Impact: misleading filesystem information

140

days on average for these three cases

Range: 28 to 292 days

Case	Impact of the silent regression	Days to fix
Inotify cookies	Uninitialized cookies undermined reliable matching of rename events.	28
Overlayfs hole discovery	Sparse-file tools lost the backing filesystem's ability to identify holes.	292
Overlayfs block size	A zero block size could crash filefrag. The fiemap test still reported success.	100

Timing uses commit dates from introduction or activation to the kernel fix.

The two overlayfs cases share a refactoring series but break different interfaces.

Impact: required operations never occurred

401

days on average for these three cases

Range: 98 to 587 days

Case	Impact of the silent regression	Days to fix
DAX huge pages	Huge-page mappings fell back to base pages, losing their performance benefit.	98
Tunnel PMTU feedback	Lost or misrouted ICMP feedback caused network timeouts in affected deployments.	519
pKVM single-stepping	Host single-stepping stopped working. Its logical subtest accepted zero debug exits.	587

Timing uses commit dates from introduction or activation to the kernel fix.

KVM refers to the single-step logical subtest. The six cases are a selected sample.

Inotify: rename cookies lost their meaning

Intended change

Separate notification events by consumer to reduce memory use and simplify event handling.

Silent regression and impact

The refactor omitted cookie initialization. Rename events could carry unrelated values, undermining the link between the old and new filename.

Why the unchanged test passed

LTP checked event type and name. It printed the cookie on its PASS line without checking it.

Resolution

28

calendar days

21 Jan 2014 to
18 Feb 2014

Initialize the cookie and propagate it through the event callbacks.

Sources: Inotify cookies

Maintainer diagnosis - Jan Kara - 17 Feb 2014

Identifies the introducing change and explains why LTP missed the bug.

<https://lore.kernel.org/all/20140217125954.GD3686@quack.suse.cz/>

Kernel repair - 45a22f4c11fe

“inotify: Fix reporting of cookies for inotify events” — restores initialization.

<https://git.kernel.org/linus/45a22f4c11fef4ecd5c61c0a299cd3f23d77be8e>

LTP test repair - 1ea2821124f3

Adds cookie assertions after the regression; exposes the old coverage gap.

<https://github.com/linux-test-project/ltptest/commit/1ea2821124f38305fcf40b109839748b7ea8e9f9>

Pre-regression test - LTP 20140115 - inotify02.c

Inspect the rename-event checks: event type and name were checked; cookie was not.

<https://github.com/linux-test-project/ltptest/blob/c8b3e28097e7d3208df9daceaf92c25eae87ebf0/testcases/kernel/syscalls/inotify/inotify02.c>

Introduced / activated: 2014-01-21

<https://git.kernel.org/linus/7053aee26a3548ebaba046ae2e52396ccf56ac6c>

Old LTP tests printed cookies but did not validate them.

Fixed: 2014-02-18 (28 days)

<https://git.kernel.org/linus/45a22f4c11fef4ecd5c61c0a299cd3f23d77be8e>

Regression code: Inotify cookie initialization

Change that caused the regression

fs/notify/inotify/inotify_fsnotify.c
7053aee26a35

```
event = kmalloc(alloc_len, GFP_KERNEL);  
...  
fsnotify_init_event(  
    fsn_event, inode, mask);  
event->wd = i_mark->wd;  
event->name_len = len;
```

The introducing refactor allocated an event without initializing `sync_cookie`. The old allocator set that field.

Missing check: a rename pair must carry matching cookies.

Excerpts reflowed. ... marks omissions. + / - mark added / removed code.

Test that missed the regression

LTP: inotify02.c
20140115 / c8b3e28097e7

```
} else if (  
    (event_set[test_num].mask ==  
    event->mask)  
    &&  
    (!strcmp(event_set[test_num].name,  
    event->name, event->len))) {  
    tst_resm(TPASS, ...);  
}
```

The PASS branch compares mask and name. It prints the cookie without checking it.

Repair: Inotify cookies

Actual kernel fix

fs/notify/inotify/inotify_fsnotify.c

Fixed in 45a22f4c11fe

```
fsnotify_init_event(  
    fsn_event, inode, mask);  
event->wd = i_mark->wd;  
event->sync_cookie = cookie;  
event->name_len = len;
```

The fix passes the cookie through the callbacks and initializes sync_cookie before delivery.

What the original test needed

LTP: cookie semantics

Proposed pseudocode

```
events = rename_and_collect(old, new)  
from = events.require(IN_MOVED_FROM)  
to   = events.require(IN_MOVED_TO)  
assert(from.cookie != 0)  
assert(to.cookie == from.cookie)  
other = create_and_require(IN_CREATE)  
assert(other.cookie == 0)
```

Keep the event type and name checks. Also check the cookie's value and rename-pair relationship.

Required behavior: rename events carry a shared, nonzero cookie.

Overlayfs: hole discovery disappeared

Intended change

Give overlayfs its own file operations so a reader sees changes made through another writable open.

Silent regression and impact

The new seek operation treated the file as data up to EOF. Overlayfs stopped exposing the backing filesystem's real holes.

Why the unchanged test passed

The seek test probed the kernel, saw EOF, and relaxed its expected result to accept that fallback.

Resolution

292

calendar days

18 Jul 2018 to
6 May 2019

Delegate SEEK_HOLE
and SEEK_DATA to the
backing file.

Sources: Overlayfs hole discovery

fstests diagnosis and patch - Amir Goldstein - 3 Apr 2019

“Add more sanity to seek_sanity_test” explains how accepting EOF hid the regression.

<https://lore.kernel.org/all/20190403055926.23253-1-amir73il@gmail.com/>

Kernel repair - 9e46b840c705

“ovl: support stacked SEEK_HOLE/SEEK_DATA” delegates seeking to the backing file.

<https://git.kernel.org/linus/9e46b840c7053b5f7a245e98cd239b60d189a96c>

Merged test repair - fstests 340810877483

Requires real hole detection on filesystems expected to provide it.

<https://git.kernel.org/pub/scm/fs/xfs/xfstests-dev.git/commit/?id=34081087748374c5fd2a4056d6d832762c763b>

Pre-regression helper - fstests c432589da355

See test_basic_support() and default_behavior; generic/285 selects tests 1–12.

https://git.kernel.org/pub/scm/fs/xfs/xfstests-dev.git/tree/src/seek_sanity_test.c?id=c432589da355e3bc2016c1ddec7fcdb963f2f382

Introduced / activated: 2018-07-18

<https://git.kernel.org/linus/a6518f73e60e5044656d1ba587e7463479a9381a>

Fixed: 2019-05-06 (292 days)

<https://git.kernel.org/linus/9e46b840c7053b5f7a245e98cd239b60d189a96c>

The regression lost backing-filesystem capability; generic SEEK_HOLE permits an EOF fallback.

Regression code: Overlayfs hole discovery

Change that caused the regression

fs/overlayfs/file.c: ovl_llseek()
d1d04ef8572b / active at a6518f73e60e

```
return generic_file_llseek_size(  
    file, offset, whence,  
    realinode->i_sb->s_maxbytes,  
    i_size_read(realinode));
```

After activation, the generic helper returned EOF for SEEK_HOLE, losing the backing file's hole map.

Test that missed the regression

fstests: src/seek_sanity_test.c
c432589da355

```
exp2 = exp;  
if (origin == SEEK_HOLE && exp2 != -1)  
    exp2 = filsz;  
...  
pos = lseek(fd, set, origin);  
...  
ret = !(pos == exp || pos == exp2);
```

The test accepts EOF as an alternate answer even when it expected a real hole.

Missing check: preserve hole discovery where the backing filesystem supports it.

Repair: Overlayfs hole discovery

Actual kernel fix

fs/overlayfs/file.c: ovl_llseek()

Fixed in 9e46b840c705

```
ret = ovl_real_fdget(file, &real);
if (ret)
    return ret;
...
real.file->f_pos = file->f_pos;
...
ret = vfs_llseek(
    real.file, offset, whence);
...
file->f_pos = real.file->f_pos;
```

The fix asks the real backing file to seek and keeps the overlay file position synchronized.

What the original test needed

fstests: a known sparse-file fixture

Proposed pseudocode

```
require_hole_aware_backing_fs()
make_aligned_sparse_fixture()
assert(expected_hole < file_size)

hole = lseek(overlay_fd, 0, SEEK_HOLE)
assert(hole == expected_hole)
```

On a known hole-aware filesystem, require the fixture's hole offset. An EOF answer must fail.

Required behavior: overlayfs preserves the backing filesystem's hole discovery.

Overlays: a zero block size hid failed work

Intended change

Use overlays file operations while preserving access to the underlying file and its behavior.

Silent regression and impact

FIGETBSZ returned success with a block size of zero. filefrag could divide by zero, and the fiemap test could fail before useful work.

Why the unchanged test passed

The helper trusted the returned size. The shell pipeline could hide its failure behind tee's successful exit.

Resolution

100

calendar days

18 Jul 2018 to
26 Oct 2018

Reject zero block sizes with EINVAL so userspace can use a fallback.

Sources: Overlayfs block size

fstests diagnosis and repair - 6fe6c772f66a

“fix getting blocksize on overlayfs” records tests succeeding after doing little work.

<https://git.kernel.org/pub/scm/fs/xfs/xfstests-dev.git/commit/?id=6fe6c772f66a7c056b5005be4f4b73410b67d89f>

Kernel repair - 8f97d1e99149

Documents filefrag’s divide-by-zero and returns EINVAL for an unset block size.

<https://git.kernel.org/linus/8f97d1e99149a7f1aa19e47a51b09764382a482e>

Original refactoring discussion - Miklos Szeredi - May 2018

Explains stacked file operations and reports no regressions from several test suites.

<https://lore.kernel.org/all/20180507083807.28792-1-mszeredi@redhat.com/>

Pre-regression test - fstests c432589da355 - generic/225

Inspect the fiemap-tester → tee pipeline; the helper and generic/094 are linked in notes.

<https://git.kernel.org/pub/scm/fs/xfs/xfstests-dev.git/tree/tests/generic/225?id=c432589da355e3bc2016c1ddec7fcdb963f2f382>

Introduced / activated: 2018-07-18

<https://git.kernel.org/linus/a6518f73e60e5044656d1ba587e7463479a9381a>

A separate interface failure from the same overlayfs refactoring series.

Fixed: 2018-10-26 (100 days)

<https://git.kernel.org/linus/8f97d1e99149a7f1aa19e47a51b09764382a482e>

Regression code: Overlayfs block size

Change that caused the regression

fs/open.c and fs/ioctl.c
a6518f73e60e (activation)

```
- return do_dentry_open(file,  
-     d_backing_inode(dentry), NULL);  
+ return do_dentry_open(file,  
+     d_backing_inode(path->dentry),  
+     NULL);
```

Unchanged FIGETBSZ handler:

```
case FIGETBSZ:  
    return put_user(  
        inode->i_sb->s_blocksize, argp);
```

The introducing change opened the overlay inode. FIGETBSZ then read its unset block size and returned zero.

Missing check: require a valid block size and propagate the helper's failure.

Excerpts reflowed. ... marks omissions. + / - mark added / removed code.

Test that missed the regression

fstests: tests/generic/225
c432589da355

```
$here/src/fiemap-tester -q -S -s $seed \  
-p 0 -r 200 $fiemapfile 2>&1 |  
tee -a $fiemaplog  
if [ $? -ne 0 ]; then  
    ...  
fi  
...  
status=0  
exit
```

The helper accepts size zero. This pipeline checks tee's status and can hide a helper failure.

Repair: Overlayfs block size

Actual kernel fix

fs/ioctl.c

Fixed in 8f97d1e99149

```
case FIGETBSZ:
    if (!inode->i_sb->s_blocksize)
        return -EINVAL;
    return put_user(
        inode->i_sb->s_blocksize, argp);
```

The fix returns EINVAL when the filesystem has no block size, allowing a caller to use a fallback.

What the original test needed

fstests: helper result and exit status

Proposed pseudocode

```
rc, size = FIGETBSZ(fd)
if rc == SUCCESS:
    assert(size > 0)
else:
    require_errno(EINVAL)
    size = statfs_block_size(fd)
    assert(size > 0)
    assert(run_fiemap_helper(size).exit == 0)
```

Reject a successful zero result. Check the helper's own exit status, even when logging through tee.

Required behavior: valid geometry and a helper failure that reaches the test verdict.

DAX: the huge-page test used ordinary pages

Intended change

Prevent DAX hangs by falling back when a smaller mapping conflicts with a requested huge-page fault.

Silent regression and impact

The new check mistook an empty lookup for a conflict.
Huge-page faults fell back to base pages while data access still worked.

Why the unchanged test passed

The ndctl test verified reads and writes. It never checked that the kernel actually installed a huge-page mapping.

Resolution

98

calendar days

16 Jul 2019 to
22 Oct 2019

Check for an empty
lookup before checking
for a size conflict.

Sources: DAX huge pages

ndctl test-series explanation - Dan Williams - 19 Oct 2019

“Add huge page fault validation” explicitly says the existing unit test missed the regression.

<https://www.mail-archive.com/linux-nvdimmm%40lists.01.org/msg17108.html>

Kernel repair discussion - 21 Oct 2019

“fs/dax: Fix pmd vs pte conflict detection” identifies the bad change and performance impact.

<https://lore.kernel.org/all/157167532455.3945484.11971474077040503994.stgit@dwillia2-desk3.amr.corp.intel.com/>

ndctl test repair - 17e13ab193a1

Adds tracepoint validation that a huge-page fault actually succeeded.

<https://github.com/pmem/ndctl/commit/17e13ab193a131ff3c966f4393fcdab8e3d92294>

Pre-regression test - ndctl 44addc119d2c - dax.sh

The driver and dax-pmd.c are unchanged across the introduction; comparison is in notes.

<https://github.com/pmem/ndctl/blob/44addc119d2ce1a88ffe00fd7e0a189e65534ee4/test/dax.sh>

Introduced / activated: 2019-07-16

<https://git.kernel.org/linus/23c84eb7837514e16d79ed6d849b13745e0ce688>

Fixed: 2019-10-22 (98 days)

<https://git.kernel.org/linus/6370740e5f8ef12de7f9a9bf48a0393d202cd827>

Reads and writes worked; the missing assertion concerned actual huge-page mappings.

Regression code: DAX huge-page fallback

Change that caused the regression

fs/dax.c: get_unlocked_entry()
23c84eb78375

```
entry = xas_find_conflict(xas);  
+ if (dax_entry_order(entry) < order)  
+   return XA_RETRY_ENTRY;  
if (!entry ||  
    WARN_ON_ONCE(!xa_is_value(entry)) ||  
    !dax_is_locked(entry))  
    return entry;
```

The introducing patch put the order check before the NULL check. An empty lookup triggered fallback.

Test that missed the regression

ndctl: test/dax-pmd.c
44addc119d2c

```
case 3: /* convert ro mapping to rw */  
    rc = *(volatile int *) addr;  
    *(volatile int *) addr = rc;  
    rc = 0;  
    break;
```

This access succeeds with base pages too. The test never requires a successful PMD fault.

Missing check: verify the mapping size, beyond successful data access.

Repair: DAX huge-page faults

Actual kernel fix

fs/dax.c: get_unlocked_entry()

Fixed in 6370740e5f8e

```
entry = xas_find_conflict(xas);
if (!entry ||
    WARN_ON_ONCE(!xa_is_value(entry)))
    return entry;
if (dax_entry_order(entry) < order)
    return XA_RETRY_ENTRY;
if (!dax_is_locked(entry))
    return entry;
```

The fix handles an empty lookup before checking entry order. A missing entry no longer forces fallback.

What the original test needed

ndctl: PMD-eligible DAX fixture

Proposed pseudocode

```
require_pmd_eligible_mapping()
trace = record_faults_for_test()
run_existing_data_checks()
faults = trace.target_pmd_faults()
assert(expected_faults > 0)
assert(len(faults) == expected_faults)
assert(all(f.result == VM_FAULT_NOPAGE
           for f in faults))
```

Count the fixture's expected successful PMD faults. Reads and writes alone cannot prove huge-page use.

Required behavior: the eligible mapping actually takes the huge-page fault path.

PMTU: the missing error kept the test green

Intended change

Restore PMTU feedback for host-directed tunnel traffic without changing its packet type in Open vSwitch.

Silent regression and impact

The expanded path built an ICMP reply but kept the original route. Replies disappeared or went the wrong way, causing network timeouts.

Why the unchanged test passed

The route-cache check still passed. Losing the error kept the old ping command successful after the behavior changed.

Resolution

519

calendar days

2 Apr 2025 to
3 Sep 2026

Drop the stale route
before reinjecting the
ICMP reply.

Sources: Tunnel PMTU feedback

Why the tests stayed green - Yaroslav Dudkov - 1 Sep 2026

Explains the route-cache check and how a lost ICMP error leaves ping successful.

<https://lore.kernel.org/all/20260901081825.287173-1-aroslavdudkov622@gmail.com/>

Kernel repair and discussion - Ido Schimmel - 2 Sep 2026

“Drop stale dst when building an ICMP error for PMTUD” traces the new broken path.

<https://lore.kernel.org/all/20260902190112.4126199-1-idosch@nvidia.com/>

Production regression report - Debian bug #1108860

Records affected deployments and behavior with and without the stable backport.

<https://bugs.debian.org/1108860>

Unchanged selftest - introducing revision - pmtu.sh

Compare with the parent: the whole net selftest tree is identical across this change.

<https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/tools/testing/selftests/net/pmtu.sh?id=8930424777e43257f5bf6f0f0f53defd0d30415c>

Introduced / activated: 2025-04-02

<https://git.kernel.org/linus/8930424777e43257f5bf6f0f0f53defd0d30415c>

Fixed: 2026-09-03 (519 days)

<https://git.kernel.org/linus/b58d749633203d92c265317b45fccee555090352>

The introducing patch leaves the tests untouched; the later repair also adjusts ping.

Regression code: Tunnel PMTU feedback

Change that caused the regression

net/ipv4/ip_tunnel_core.c
8930424777e4

```
skb_dst_update_pmtu_no_confirm(  
    skb, mtu);  
  
- if (!reply ||  
-     skb->pkt_type == PACKET_HOST)  
+ if (!reply)  
    return 0;
```

The introducing patch admitted PACKET_HOST traffic with a saved route. The ICMP builder kept that stale route.

Missing check: confirm that the generated ICMP error reaches the sender.

Excerpts reflowed. ... marks omissions. + / - mark added / removed code.

Test that missed the regression

selftests/net/pmtu.sh
unchanged across 8930424777e4

```
run_cmd ${ns_a} ${ping} -q -M want \  
-i 0.1 -w 1 \  
-s $(( ${ll_mtu} + 500 )) ${dst} \  
|| return 1  
...  
pmtu="$(route_get_dst_pmtu_from_exception \  
    "${ns_a}" ${dst})"  
check_pmtu_value ${exp_mtu} "${pmtu}" ...
```

PMTU cache updates still pass. With the ICMP error lost, deadline-based ping can succeed.

Repair: Tunnel PMTU feedback

Actual kernel fix

net/ipv4/ip_tunnel_core.c

Fixed in b58d74963320

```
skb_reset_mac_header(skb);
```

```
if (skb_valid_dst(skb))  
    skb_dst_drop(skb);
```

```
return skb->len;
```

Both ICMP builders drop the stale route. The IP layer then looks up a route back to the sender.

What the original test needed

selftests: ICMP delivery at the sender

Proposed pseudocode

```
capture = watch_icmp_at_sender()  
probe = send_oversize_routed_packet()  
error = capture.require_reply(probe)  
assert(error.type == expected_pmtu_type)  
assert(error.code == expected_code)  
assert(error.mtu == expected_mtu)  
assert(error.quotes(probe))
```

Require a matching PMTU error at the sender within a timeout. A correct route cache alone is insufficient.

Required behavior: the generated PMTU error reaches the packet's original sender.

pKVM: zero single steps satisfied the test

Intended change

Separate host-requested debug state from the guest's own debug register context.

Silent regression and impact

The new field never reached the hypervisor's vCPU. Host single-step requests produced no expected debug exits.

Why the unchanged test passed

The subtest checked each debug exit only if one occurred. Normal guest completion with zero exits also passed.

Resolution

587

calendar days

20 Dec 2024 to
30 Jul 2026

Copy the new external debug field to the hypervisor's vCPU.

Sources: pKVM single-stepping

Maintainer diagnosis - Fuad Tabba - 26 Jul 2026

Series cover letter explains why the existing single-step subtest passed with zero exits.

<https://lore.kernel.org/all/20260726143643.303456-1-fuad.tabba@linux.dev/>

Kernel repair - external_mdscr_el1 - 26 Jul 2026

Shows the missing copy to the hypervisor vCPU and names the introducing change.

<https://lore.kernel.org/all/20260726143643.303456-2-fuad.tabba@linux.dev/>

Pre-existing single-step test - b18e4d4aebdd - 2022

Inspect test_single_step_from_userspace(): debug exits are checked only when received.

<https://git.kernel.org/linus/b18e4d4aebdd05810ceb2f73d7f72afcd11b41>

Scope qualification - separate debug repair - 0f1a6c5c9784

Other debug subtests had an earlier failure; do not infer continuously passing full-suite CI.

<https://git.kernel.org/linus/0f1a6c5c9784eff7e31e4915e17285fb89ad3644>

Introduced / activated: 2024-12-20

<https://git.kernel.org/linus/4ad3a0b87f2ec4714fd6a6bd5de57b4c30e15753>

Fixed: 2026-07-30 (587 days)

<https://git.kernel.org/linus/119a9a23fbae4fefbb5cd4776547b3b6803a7eaa>

Evidence concerns the logical single-step subtest; the whole executable has a separate history.

Regression code: pKVM single-stepping

Change that caused the regression

debug.c and hyp/nvhe/hyp-main.c
4ad3a0b87f2e

```
+ if (vcpu->guest_debug &  
+   KVM_GUESTDBG_SINGLESTEP)  
+   mdscr |= MDSCR_EL1_SS;  
...  
+ vcpu->arch.external_mdscr_el1 = mdscr;
```

Existing context copy:

```
hyp_vcpu->vcpu.arch.ctx =  
    host_vcpu->arch.ctx;
```

The introducing refactor moved SS outside ctx. The hyp vCPU never received the new field.

Test that missed the regression

KVM: aarch64/debug-exceptions.c
unchanged across 4ad3a0b87f2e

```
if (run->exit_reason != KVM_EXIT_DEBUG) {  
    cmd = get_ucall(vcpu, &uc);  
    if (cmd == UCALL_ABORT) {  
        REPORT_GUEST_ASSERT(uc);  
    } else if (cmd == UCALL_DONE) {  
        break;  
    }  
    ...  
}
```

UCALL_DONE ends the logical subtest without requiring even one KVM_EXIT_DEBUG.

Missing check: require the expected single-step exits before accepting completion.

Repair: pKVM single-stepping

Actual kernel fix

hyp/nvhe/hyp-main.c: flush_debug_state()

Fixed in 119a9a23fbae

```
} else if (kvm_host_owns_debug_regs(
    &hyp_vcpu->vcpu)) {
    ...
    hyp_vcpu->vcpu.arch.external_mdscr_el1 =
        host_vcpu->arch.external_mdscr_el1;
}
```

The fix copies external_mdscr_el1 to the hyp vCPU, including the host's single-step control bit.

What the original test needed

KVM: each single-step test iteration

Proposed pseudocode

```
assert(expected_exits > 0)
debug_exits = 0
for exit in run_to_UCALL_DONE():
    if exit.reason == KVM_EXIT_DEBUG:
        assert_pc_is_expected(exit.pc)
        debug_exits += 1
assert(debug_exits == expected_exits)
```

Require the expected debug exits in every iteration before accepting normal guest completion.

Required behavior: every requested single step produces the expected debug exit.