

Testing klp-build

Unit tests to CI



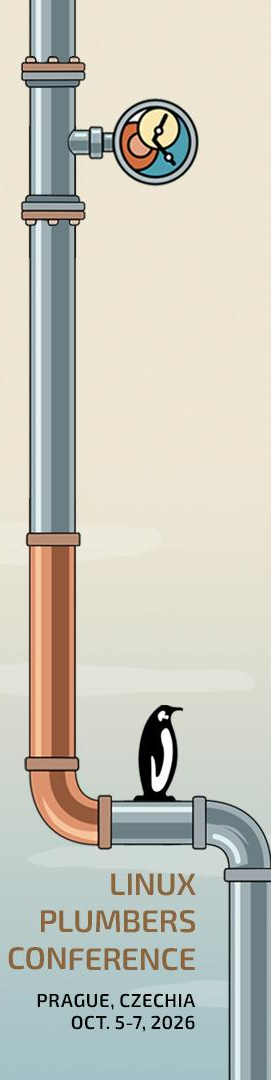
LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

Song Liu (Meta)
Joe Lawrence (Red Hat)

Agenda

- klp-build unit testing
- Integration testing
- Continuous Integration



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

klp-build unit testing



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

You changed objtool or klp-build

Now what?

```
linux/  
├── tools/objtool/      < you changed something here  
├── scripts/livepatch/  < or here  
└── ...
```

You want to know: **does it still work?**

The In-tree klp-build Test Suite

```
linux/
├── tools/objtool/tests/
│   ├── generic/
│   │   ├── fixtures/*.c
│   │   └── test-*.sh
│   ├── lib.sh
│   ├── run-tests.sh
│   └── x86/
│       ├── fixtures/*.c
│       └── test-*.sh
```

- < arch-independent
- < freestanding C sources
- < one test each
- < harness API
- < runner + TAP output
- < arch-specific (same shape)

Unit tests covering objtool klp diff and objtool klp checksum across x86, GCC, Clang, UBSAN, kCFI, ThinLTO.

Unit Test Scope

In scope

```
objtool klp checksum  
objtool klp diff  
objtool klp post-link  
objtool --klp-symids
```

Out of scope

Full klp-build pipeline,
Module loader / boot-time
behavior,
Integration tests (planned)

Preflight Harness Checks

```
TAP version 13
# preflight
#  objtool    .../tools/objtool/objtool (klp: yes)
#  compiler   gcc (GCC) 15.2.1 20260123 (Red Hat 15.2.1-7)
#  arch       x86
#  tmpdir     /tmp (each test builds in a fresh directory here)
```

- objtool exists and was built with klp support
- \$CC, objcopy, ld, readelf are present
- objtool accepts an object built by \$CC
- ARCH matches what the compiler actually emits

A failed preflight aborts the whole run.

Test TAP Output

Each test prints one result line.
Diagnostics from objtool appear as `#` comments.

```
1..52
ok - test-basic: changed function cloned, unchanged function left alone
ok - test-changed-data: changed data symbol rejected
...
ok - test-kcfi # SKIP (declared) clang only: kCFI is a Clang feature
...
```

Ten seconds later...

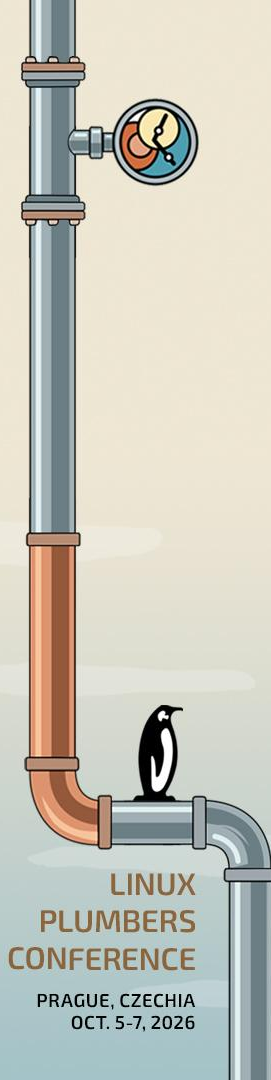
TAP Report Summary

```
# pass:48 fail:0 static-skip:4 probe-skip:0 xfail:0 xpass:0
```

Result	Meaning
pass	test passed
fail	test failed, or skipped without declaring why
static-skip	declared up front <code>gcc_only</code> , <code>clang_only</code> , etc.
probe-skip	compiler did not produce expected output
xfail	expected test failure
xpass	unexpected test success

Lightweight Framework

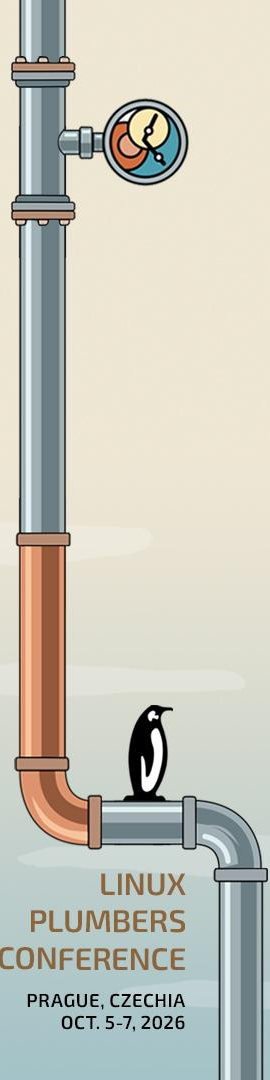
- Based on Puranjay Mohan's work
 - https://github.com/puranjaymohan/bpf/tree/klp_build_tests
- No kernel build
- objtool never reads .config
- objtool klp only cares what's in the object file
- match that with a freestanding fixture + compiler flags



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

Whole suite runs in seconds!



One run = one (arch, compiler)

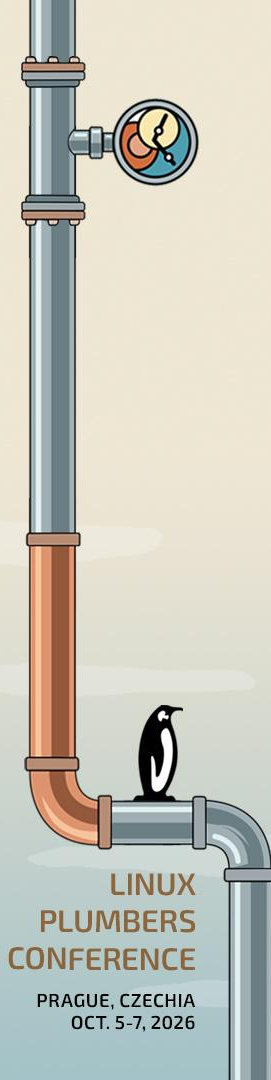
Each invocation covers an architecture/toolchain.
(CI [later] runs the matrix natively)

```
Runner A: x86_64 + gcc      (clang-only tests > static-skip)
Runner B: x86_64 + clang    (gcc-only tests > static-skip)
Runner C: arm64  + gcc      (x86/ tests not run at all)
...
```

How every test runs

`setup > build_pair > run_diff > ELF assertions`

Each test prints one TAP result line and the runner summarizes pass, fail, and skips.

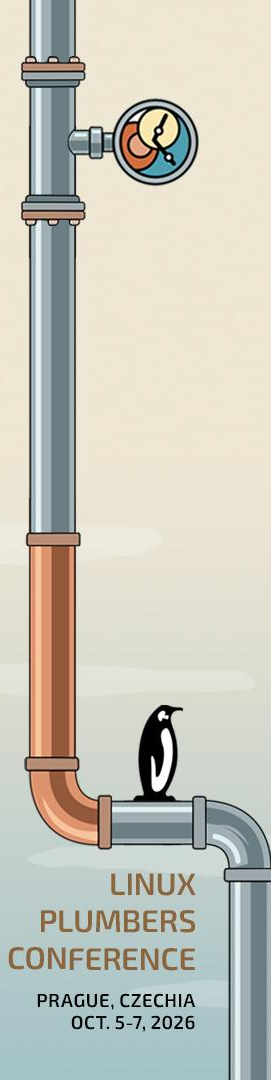


LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

ELF lib.sh Library Examples

Helper	Checks
<code>assert_patched / assert_not_patched</code>	function cloned, or left alone
<code>assert_section / assert_no_section</code>	section present in out.o
<code>assert_symbol / assert_reloc_sym</code>	symbol or reloc target in out.o
<code>assert_klp_sym / assert_klp_rela</code>	post-link klp symbols and relocs
...	...



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

Full API at `tools/objtool/tests/lib.sh`

Designed for AI & humans

The framework was built so AI can write these tests (humans, too!)

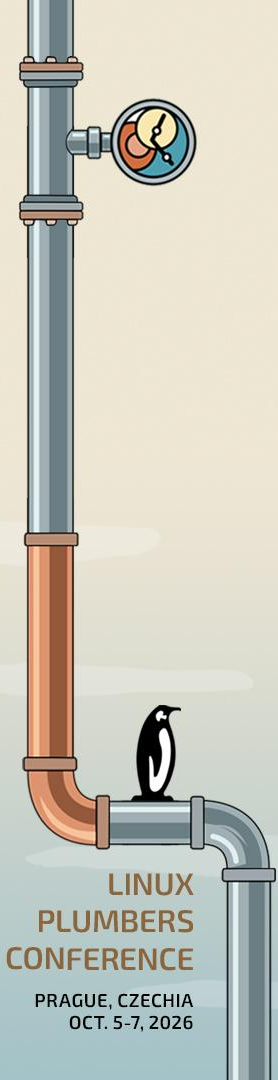
How to write a test:

`tools/objtool/Documentation/klp-write-tests.txt`

How the harness works:

`tools/objtool/Documentation/klp-test-design.txt`

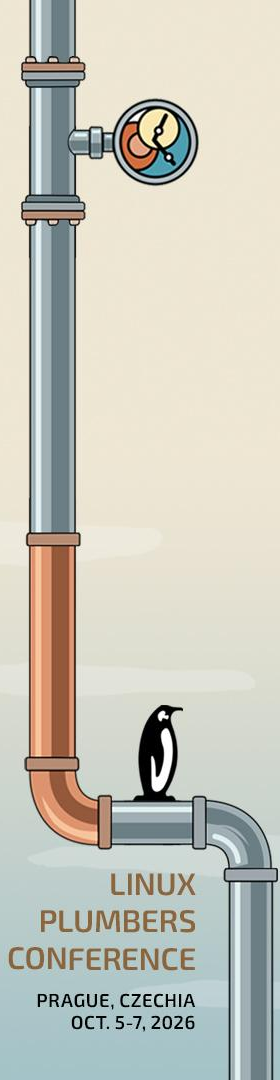
Policy: objtool klp fixes should land with unit tests.



Example: klp-build arm64 patchset

Josh Poimboeuf's WIP klp-build for arm64 patchset adds behaviors with no x86 equivalent:

arm64 behavior	What needs testing
Don't correlate mapping symbols (\$x, \$d, \$a)	Wrong symbol pairing = broken patch
Prefix/PFE detection (kCFI \$d, ftrace CALL_OPS)	Patched function loses type hash or call ops
Clone inline alternative replacements	Patched function must include its arm64 alt replacement insns



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

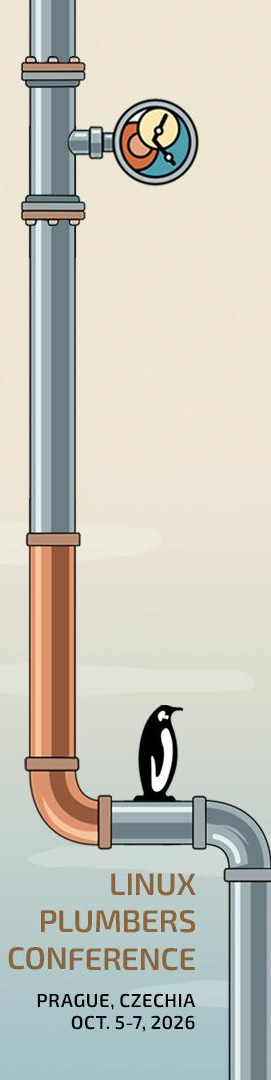
Example Prompt, Part I

Write a unit test for:

```
objtool/klp: Don't correlate arm64 mapping symbols  
(commit 64926c0f5914 on josh/klp-build-arm64-latest)
```

Read first:

```
tools/objtool/Documentation/klp-write-tests.txt  
tools/objtool/Documentation/klp-test-design.txt
```



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

Example Prompt, Part II

The fix adds `is_mapping_sym(sym)` to `dont_correlate()`.

Requirements:

- Fixture in `tools/objtool/tests/arm64/fixtures/`
- Test in `tools/objtool/tests/arm64/test-mapping-symbols.sh`
- Assert premise before result
- Must fail when `is_mapping_sym()` is removed from `dont_correlate()`
- One test, one commit; say how you verified

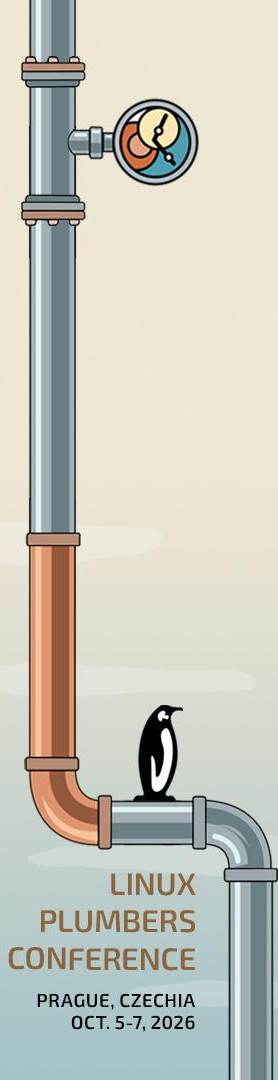
A few tokens later...

Example Prompt, Part III

Commit message: “Arm64 ELF files mark code/data transitions with \$x/\$d mapping symbols. Thousands share the same few names; correlating them is meaningless and, when a patch adds or removes a function, produces "no correlation: \$x" warnings that bury real correlation failures. Commit b889e71c7017 adds `is_mapping_sym()` to `dont_correlate()`.”

The fixture's patched build introduces an extra function so the two sides have unequal \$x counts -- equal counts match positionally and removing the fix would change nothing observable. The test asserts that premise, then that `klp diff` emits no mapping-symbol correlation warning.

Verified by removing `is_mapping_sym()` from `dont_correlate()`: the test fails with "mapping symbol was subjected to correlation". Restored, it passes under `aarch64-linux-gnu-gcc` and `clang --target=aarch64-linux-gnu`.”



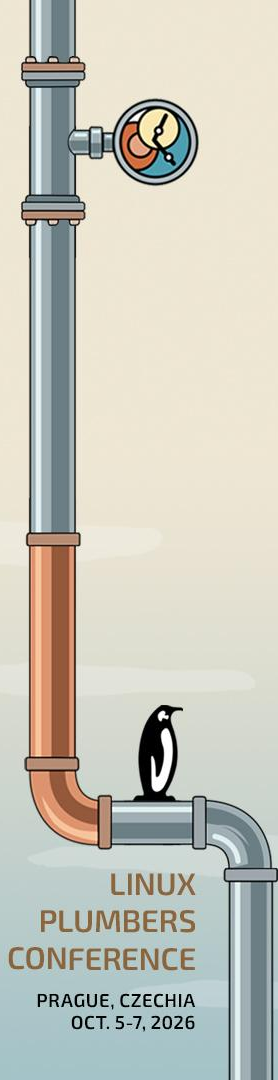
LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

Example Prompt, Part IV

AI generated still needs guidance and review!

- Models may “optimize” their work effort and shortcut required steps
- Previous context / training on kernel, livepatching, etc. can effect output
- Goal is test coverage



klp-build integration testing

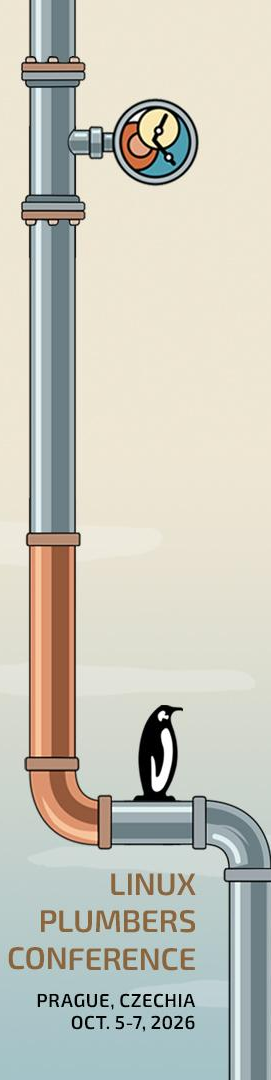


LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

kdp-build integration testing

- Test kdp-build end-to-end
- Not backward compatible
- Can be tested in VMs
- Dedicated test target code
 - No need to worry about `cmdline_proc_show` changes
 - `CONFIG_KDP_TEST`, `CONFIG_KDP_TEST_MODULE`
- Commit `*.patch` files to the repository
 - AI agents have no problem keeping them up to date



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

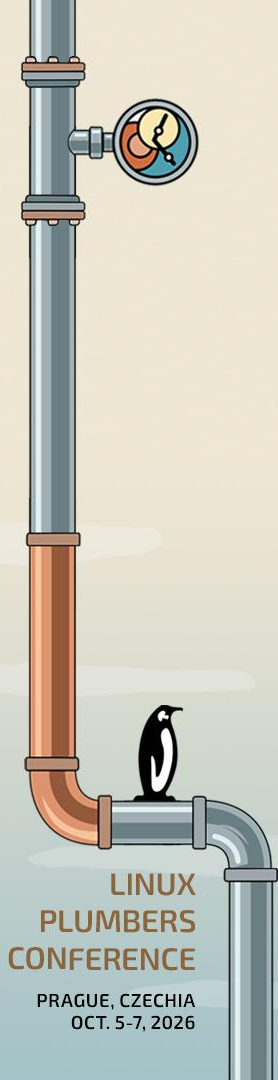
klp-build integration testing

```
linux/
├── lib/livepatch/                < test target
│   ├── Makefile
│   └── *.c|h
└── tools/testing/selftests/livepatch
    ├── *.sh                     < test scripts
    ├── config.*                 < configs to test
    └── test_patches/*.patch     < patches
```

Build kernels based on `config.*`; run `klp-build` with `*.patch`, verifier results with `*.sh`

klp-build integration testing

- Easy to add tests (Assisted-by some AI)
- Easy to run in CI
- More than a testing framework
 - Examples for human and AI
 - Playground for the developer
- Already helped finding bugs



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

What's not ideal?

- Missing negative tests
- Scale to many tests
 - Each klp-build takes minutes
 - Positive tests can be merged
 - How to merge negative tests?
 - Maybe use `--keep-tmp`?

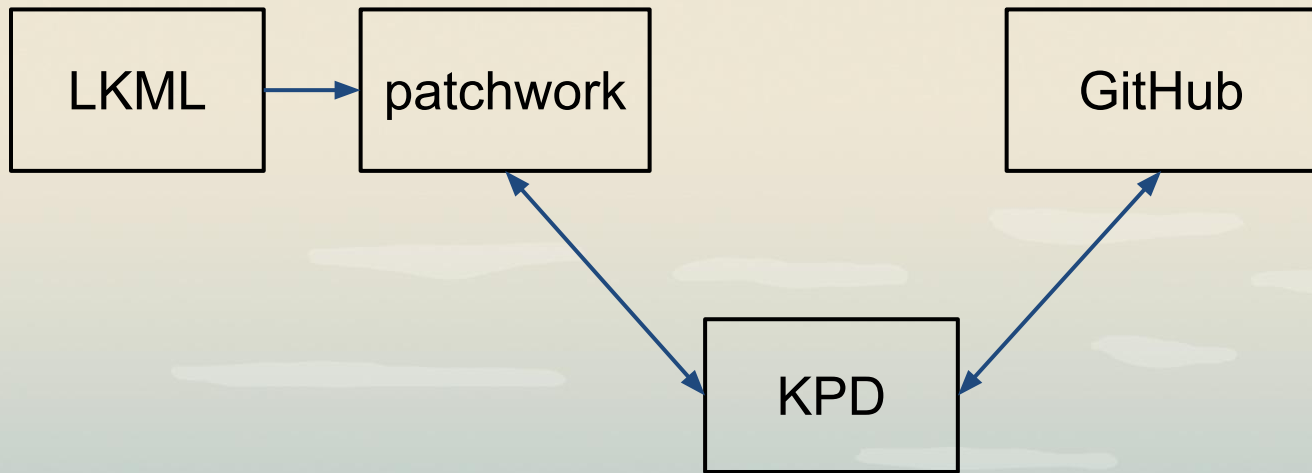
Continuous Integration



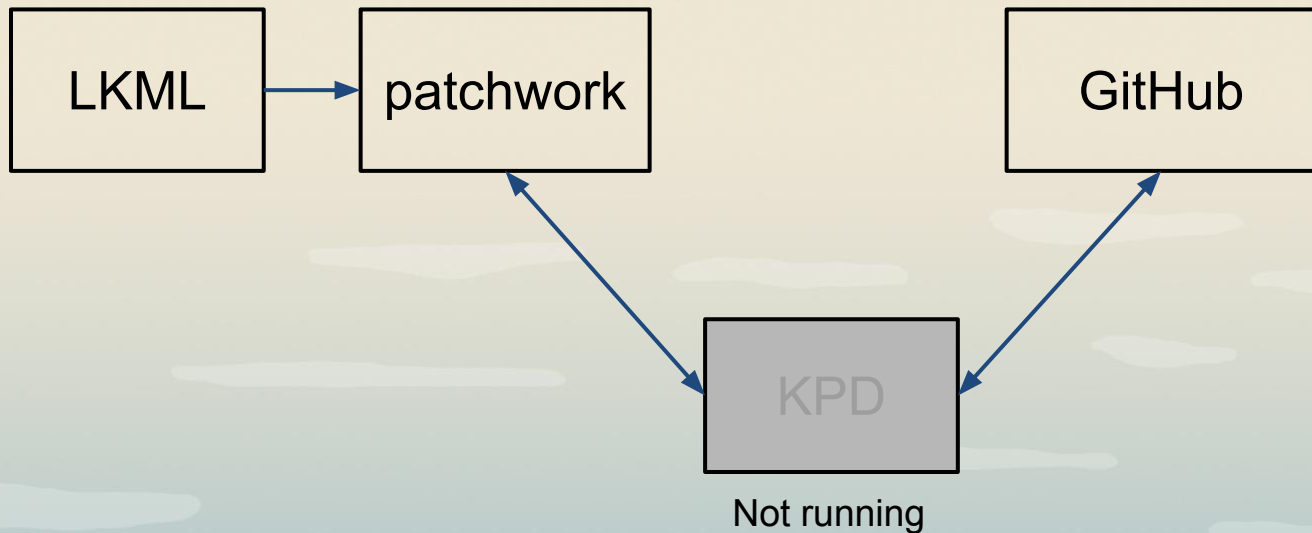
LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

CI: Run tests for all the patches



CI: Run tests for all the patches



Patchwork UI

☐ Patch	Series	A/ R/T	S/W/ F	▲ Date	Submitter	Delegate	State
☐ objtool/klp: Fix detection of static keys defined in other files	objtool/klp: Fix detection of static keys defined in other files	---	---	2026-10-06	Song Liu		New

Patch

```
diff --git a/tools/objtool/klp-diff.c b/tools/objtool/klp-diff.c
index 34da55e45609..cbc475d79156 100644
--- a/tools/objtool/klp-diff.c
+++ b/tools/objtool/klp-diff.c
@@ -1862,8 +1862,13 @@ static int validate_special_section_klp_reloc(struct elfs *e, struct symbol *sym
     if (convert_reloc_sym(e->patched, reloc))
         continue;

-    /* Static branch/call keys are always STT_OBJECT */
-    if (reloc->sym->type != STT_OBJECT) {
+    /*
+     * Static branch/call keys are STT_OBJECT if they're defined
+     * in this object file. Keys defined in another file, whether
+     * in vmlinux, the same module or another module, are undefined
+     * symbols, which are STT_NOTYPE.
+     */
+    if (reloc->sym->type != STT_OBJECT && !is_undef_sym(reloc->sym)) {

        /* Save code location which can be printed below */
        if (reloc->sym->type == STT_FUNC && !code_sym) {
```

<https://patchwork.kernel.org/project/live-patching/list/>

LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

GitHub Tests

- objtool-klp unit tests
- kernel build
- livepatch selftests in a VM
- On test failure, *.o files are uploaded for debugging
- GitHub free runners
- User runners (LoongArch?)

The screenshot shows the GitHub Actions interface for a workflow run. The top section is titled 'Summary' and shows 'All jobs'. A list of jobs is displayed, each with a green checkmark indicating success: 'tests gcc-11', 'tests gcc-13', 'tests clang-18', 'kernel gcc-11', 'kernel gcc-13', 'build', 'livepatch selftests' (highlighted with a blue bar), and 'kernel clang-18'. Below the jobs list, there are links for 'Run details', 'Usage', and 'Workflow file'. On the right side, the 'Annotations' section shows '1 warning and 1 notice'. Below that, the specific job 'kernel gcc-13 / livepatch selftests' is detailed, showing it 'succeeded 3 days ago in 2m 13s'. A list of steps for this job is shown, each with a green checkmark: 'Set up job', 'Run actions/checkout@v4', 'Run actions/download-artifact@v4', 'Unpack', 'Run livepatch selftests', 'Upload logs on failure', 'Post Run actions/checkout@v4', and 'Complete job'.

Summary

All jobs

- ✓ tests gcc-11
- ✓ tests gcc-13
- ✓ tests clang-18
- ✓ kernel gcc-11
- ✓ kernel gcc-13
- ✓ build
- ✓ livepatch selftests
- ✓ kernel clang-18

Run details

- 🕒 Usage
- 📄 Workflow file

Annotations

1 warning and 1 notice

kernel gcc-13 / livepatch selftests

succeeded 3 days ago in 2m 13s

- ✓ Set up job
- ✓ Run actions/checkout@v4
- ✓ Run actions/download-artifact@v4
- ✓ Unpack
- ✓ Run livepatch selftests
- 🔄 Upload logs on failure
- ✓ Post Run actions/checkout@v4
- ✓ Complete job



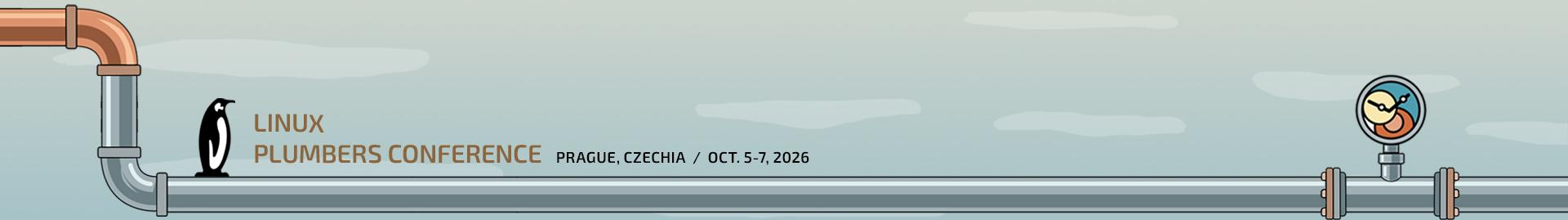
LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



CI next steps

- Start KPD somewhere
 - A free cloud server may just work
- Keep the CI green
- More tests, more fixes, AI reviews
- Send emails to patch authors and maintainers
- Alternative proposals?



The background features a light-colored, stylized illustration of a city skyline. On the left, a large, vertical pipe with a blue body and orange-brown flanges extends from the top to the middle of the frame. Below the pipe, a small black and white penguin stands on a small patch of ground. The city skyline consists of various buildings with different rooflines, windows, and architectural details, rendered in a simple, line-art style. The sky is a pale yellow with a few soft, white clouds. The water in the foreground is a light blue with some darker blue patches.

Thank You! Questions?



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026