

Upgrading Runtime Leaks to Compile-Time Errors

Adopting Clang's *require_explicit_initialization* in the Linux
Kernel

Ian Rogers, Google

Toolchain Track



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

Introduction

- Talk based on an observation that Clang has a thing, Google uses the thing but the Linux kernel isn't - is this an opportunity?
- There's no good reason I should give this talk, other than that I dug into the issue a little. I have no special Clang knowledge or secret skills. I'm just hoping the conversation will be useful.
- "Commits" talked about are at:
 - https://github.com/captain5050/linux/tree/require_explicit_initialization



Clang's `require_explicit_initialization` Attribute

- Clang 14+ attribute: `__attribute__((__require_explicit_initialization__))`
- Enforces explicit init in designated initializers (`{ .a = 1 }`), empty braces (`{}`), & stack vars.
- Triggers `-Wuninitialized-explicit-init` (in `-Wall` / `-Wextra`) if a marked field is omitted.
- Explicitly writing `.field = NULL` or `0` is allowed — forcing author & reviewer acknowledgment.

```
$ make LLVM=1 drivers/spi/spi-virtio.o
drivers/spi/spi-virtio.c:417:49: error: field 'remove' requires explicit initialization
      but is not explicitly initialized [-Werror,-Wuninitialized-explicit-init]
  417 | static struct virtio_driver virtio_spi_driver = {
      |                                     ^
./include/linux/virtio.h:260:9: note: 'remove' declared here
  260 |         void (*remove)(struct virtio_device *dev) __require_explicit_initialization;
      |         ^
```



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Header & Tooling Integration

- Defined in `include/linux/compiler_attributes.h`
- Updated `scripts/kernel-doc` (`kdoc_parser.py`) to strip the attribute when parsing struct members.
- Graceful no-op fallback on compilers lacking the attribute.

```
#if __has_attribute(__require_explicit_initialization__)\n#define __require_explicit_initialization\\\n    __attribute__((__require_explicit_initialization__))\n#else\n#define __require_explicit_initialization\n#endif
```



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



An example problem - kobject release

“... Once you registered your kobject via `kobject_add()`, you must never use `kfree()` to free it directly. ...

One important point cannot be overstated: every kobject must have a `release()` method, ...”

<https://www.kernel.org/doc/Documentation/core-api/kobject.rst>

```
116 struct kobj_type {  
117 + void (*release)(struct kobject *kobj)  
    __require_explicit_initialization;
```



Is annotating kobject release of any use?

- 22 commits on 30 kobj_types, each commit with a Fixes tag

An example:

- [arm64: cpuinfo: Properly release and reinitialize cpuregs kobject on hotplug](#)
- **Files:** [arch/arm64/kernel/cpuinfo.c](#) , [arch/arm64/include/asm/cpu.h](#)
- **Fixes:** f8d9f924526a ("arm64: cpuinfo: Expose MIDR_EL1 and REVIDR_EL1 to sysfs")
- **Explanation:** cpuregs_kobj_type had no .release, cpuid_cpu_offline() only called kobject_del(&info->kobj) without kobject_put(), and cpuid_cpu_online() called kobject_add(&info->kobj, ...) on an already-initialized kobject on CPU re-online (leaking a reference and kobj->name on every CPU hotplug cycle). Fixed by using kobject_init_and_add() on online and kobject_del() + kobject_put() + wait_for_completion() on offline.



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Is annotating kobject release of any use?

- 22 commits on 30 kobj_types, each commit with a Fixes tag

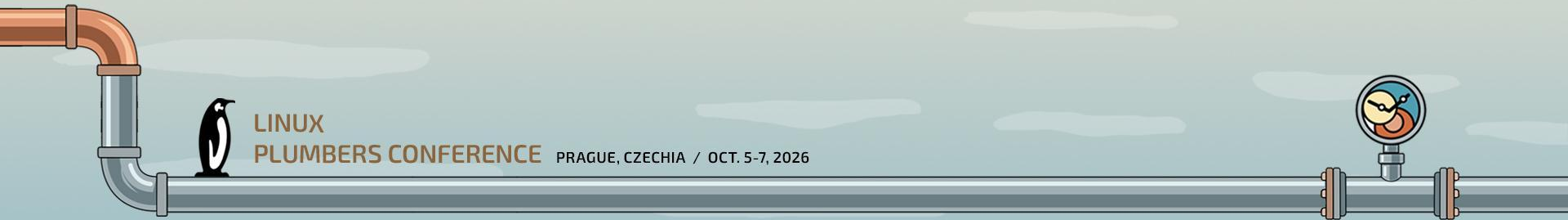
An example:

- [arm64: ...](#)
 - **Files:** ...
 - **Fixes:** ...
 - **Expla** ...
- Fix a memory leak for every hotplug on ARM64
- ... on hotplug ...
...m/cpu.h
... REVIDR_EL1 to sysfs")
... offline() only called
... online() called
... kobject ...
... kobject_add(&info->... ..) on an already-initialized kobject on CPU re-online (leaking a reference and kobj->name on every CPU hotplug cycle). Fixed by using kobject_init_and_add() on online and kobject_del() + kobject_put() + wait_for_completion() on offline.



Finding other cases of requiring explicit initialization

- Search for registration checks like `BUG_ON(!ops->cb)`
- Find 66 structs to annotate across the kernel and drivers



Other cases

Category of Annotated Structs	Structs Annotated	Non-Bug Fix-ups Needed	Bug Instances Caught
Mandatory .owner fields (watchdog_ops, block_device_operations, phy_ops, v4l2_file_operations, pci_epc_ops, framer_ops, ptp_clock_info, pps_source_info, liveupdate_*, tcf_ematch_ops, vfio_iommu_driver_ops)	12	2 files (radio-tea5777.c, tea575x.c template structs)	31 ops instances across 5 subsystems (watchdog, block/swim3, phy, media/v4l2, pci/endpoint)
Mandatory Ops & Driver Callbacks (seq_operations, dma_buf_ops, virtio_driver, vfio_device_ops, etc.)	54	1 file (drivers/ptp/ptp_clock.c stack declaration converted to designated initializer)	3 bugs (vfio/cdx missing .detach_ioas, spi-virtio missing .remove, pps-ldisc uninitialized stack info.echo leak)



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Thinking More Widely

- **1. Stack Configs & Cleanup Guards:** Stack structs (pps_source_info) & cleanup.h lock guards force = { ... } init (only 1 fix needed in kernel/irq/internals.h).
- **2. Non-Zero Invariant Types:** Catching broken BSS zero-init on `struct static_key_true` (.enabled = 1) & static_key_true_deferred (0 false positives).
- **3. Security Policies & Userspace Tools:** LSM hooks/IDs, tools/perf structs, and explicit `sysrq_key_op` masks (found 4 broken dump keys: 'q', 'd', sparc 'y', 'x').



Thinking more widely cases

Category of Annotated Structs	Structs Annotated	Non-Bug Fix-ups Needed	Bug Instances Caught
Bitmask, Lock Guard, Jump Label & LSM Invariants (sysrq_key_op, __DEFINE_UNLOCK_GUARD, static_key_true, lsm_id, lsm_info, security_hook_list)	7	4 files (3 explicit .enable_mask = 0 in sysrq_key_op, 1 designated initializer in kernel/irq/internals.h)	4 sysrq_key_op bugs ('q', 'd', and sparc64 'y', 'x' missing SYSRQ_ENABLE_DUMP)
tools/perf Command & Test Tables (cmd_struct, test_suite, test_workload)	3	0	0



Lessons learned

- **Selection Criteria:**
 - **Best Fit:** Brace-initialized ops & driver structs (seq_operations, dma_buf_ops, vfiio_device_ops, virtio_driver).
 - **Avoid:** Sentinel-terminated arrays (ctl_table, device_id) ending in {} or structs zeroed with = {} and filled later.
- **Handling Static Objects:** Subsystems can either refactor lifetime or write .release = NULL explicitly.
- **Code smells:** Warnings often speak to issues beyond just the missing initializer.



Conclusions

- Summary CL stats:
 - 77 structs (167 fields) annotated
 - 7 files (9 lines) non-bug fixups
 - 68 buggy struct definitions across 32 commits
- `__require_explicit_initialization` seems to be useful in more cases than not
- Clang support but no GCC

