TOKYO, JAPAN / DECEMBER 11-13, 2025

Automating Scope-Based Resource Cleanup with Coccinelle

Author: Erick Muthama
Co-Author: Julia Lawall

Agenda

- Introduction
- SmPL Script
- Excluded transformation
- Transformation Examples
- Conclusion
- Q&A

東京 2025
LINUX
PLUMBERS CONFERENCE
TOKYO, JAPAN / DEC. 11-13, 2025

Introduction

- Guard and scoped_guard are cleanup-based locking macros in the Linux kernel that automatically release a lock when execution leaves a scope or function.
- Guard: Acquires a lock and holds it until the end of the enclosing function.

  Ideal when most of the function should run under lock protection.

Source: fs/iomap/buffered-io.c: 148

```
static void ifs_set_range_dirty(struct folio *folio,
                struct iomap_folio_state *ifs, size_t off, size_t len)
{
        struct inode *inode = folio->mapping->host;
        unsigned int blks_per_folio = i_blocks_per_folio(inode, folio);
        unsigned int first_blk = (off >> inode->i_blkbits);
        unsigned int last_blk = (off + len - 1) >> inode->i_blkbits;
        unsigned int nr_blks = last_blk - first_blk + 1;

        guard(spinlock_irqsave)(&ifs->state_lock);
        bitmap_set(ifs->state, first_blk + blks_per_folio, nr_blks);
}
```

- Scoped_guard: A macro that acquires a lock at the beginning of a lexical scope and automatically releases it when the scope ends.

  Useful for wrapping critical sections with clear boundaries.

Source: fs/smb/client/smbdirect.c

```
static void put_receive_buffer(
        struct smbdirect_socket *sc, struct smbdirect_recv_io *response)
{
        if (likely(response->sge.length != 0)) {
                ib_dma_unmap_single(sc->ib.dev,
                                    response->sge.addr,
                                    response->sge.length,
                                    DMA_FROM_DEVICE);
                response->sge.length = 0;
        }

        scoped_guard (spinlock_irqsave, &sc->recv_io.free.lock) {
                list_add_tail(&response->list, &sc->recv_io.free.list);
                sc->statistics.put_receive_buffer++;
        }

        queue_work(sc->workqueue, &sc->recv_io.posted.refill_work);
}
```

Why is this useful?

- Reduces code clutter — Removes explicit unlock calls and makes the lock's lifetime structurally clear.
- Safer control flow — Ensures that early returns or other jumps cannot skip unlocks, preventing deadlocks and making lock lifetimes easier to audit.
- Clearer intent — Makes it obvious whether the lock is meant to cover a small scoped region (scoped_guard) or the entire function (guard).

Example

git checkout ca081fff6ecc6
source: drivers/gpu/drm/nouveau/nvkm/engine/gr/fr100.c:398

```
/* Generate golden context image. */
mutex_lock(&gr->fecs.mutex);
if (gr->data == NULL) {
        ret = gf100_grctx_generate(gr, chan, fifoch->inst);
        if (ret) {
                nvkm_error(&base->engine.subdev, "failed to construct context\n");
                return ret;
        }
}
mutex_unlock(&gr->fecs.mutex);
```

SmPL script

```
@s_g@
expression list sg_initial_lock.es;
position sg_last_unlock.p, lock_unlock_order.lp;
identifier label;
identifier virtual.lock;
identifier virtual.unlock;
identifier virtual.lock_type;
@@
+scoped_guard(lock_type, es) {
-lock@lp(es);
<...
(
   if(...) {
     ...
-    unlock(es);
     return ...;
  }
|
    if(...){
     ...
-    unlock(es);
     continue;
    }
|
  if(...){
     ...
-    unlock(es);
     goto label;
    }
)
  ...>
-unlock@p(es);
+}
```

SmPL script

```
@scoped_guard_to_guard@
position p != bad_guard.p;
expression list es;
identifier virtual.lock_type;
@@
-scoped_guard@p(lock_type, es) {
+ guard(lock_type)(es);
  ...
-}
return ...;
```

- This semantic patch provides a conservative transformation that introduces scoped_guard and guard only where it is clearly safe, avoiding problematic cases.
- The script is available on GitHub
- Link to the semantic patch

https://github.com/Erickkaranja/scope_based_cleanup.git

SmPL Script

- The semantic patch can be broken down to several sections.
  - Enforcing lock/unlock order.
  - Excluded transformation nodes.
  - Transformation .

- Mark the lock, conditional unlocks(unlocks at ifs or goto) and the last unlock.
- We ensure that the lock is safe to transform by enforcing a strict lock/unlock order.
- Remove the conditional unlocks and replace the outer lock and unlock with an appropriate guard macro.

Enforce lock/unlock order

- For correct transformation we must ensure a strict lock and unlock order I.e in a given node, the lock precedes the unlock

  source: tools/perf/util/intel-tpebs.c:559

```
mutex_lock(tpebs_mtx_get());
t = tpebs_retire_lat__find(evsel);
/*
 * If reading the first tpebs result, send a ping to the record
 * process. Allow the sample reader a chance to read by releasing an
 * reacquiring the lock.
 */
if (t && &t->nd == tpebs_results.next) {
        ret = tpebs_send_record_cmd(EVLIST_CTL_CMD_PING_TAG);
        mutex_unlock(tpebs_mtx_get());
        if (ret)
                return ret;
        mutex_lock(tpebs_mtx_get());
}
```

Excluded Transformation Nodes

Bad break statements

- Guard/scoped_guard implementation wraps around a for loop.

- The problem is that the break would affect the new inner for loop, and not the outer loop as intended.

- Consider the example below:-

```
for(...;...;...){       for(...;...;...){
  lock(...);     scoped_guard(...){
  ...          ...
  if(...)              if(...)
    break;               break;
  unlock(...);             }
}       }
```

source: fs/smb/server/transport_tcp.c:235

```
while (!kthread_should_stop()) {
        mutex_lock(&iface->sock_release_lock);
        if (!iface->ksmbd_socket) {
                mutex_unlock(&iface->sock_release_lock);
                break;
        }
        ret = kernel_accept(iface->ksmbd_socket, &client_sk,
                            SOCK_NONBLOCK);
        mutex_unlock(&iface->sock_release_lock);
```

Excluded Transformation Nodes

Unlock at else

- Conditional unlock at an else may lead to a bad transformation.

- Example:-

```
lock(...);      scoped_guard(...){
if(...)      if(...)
   ...          ...
else{     else{
   ...    ...
   unlock(...);        unlock(...);
}           }
...           ...
unlock(...);  }
```

source: drivers/gpu/drm/msm/dp/dp_display.c:651

```
if (block_len == 0)
        ; /* hole */
else if (unlikely(block_len > 2*PAGE_SIZE ||
                  (uncompressed && block_len > PAGE_SIZE))) {
        mutex_unlock(&read_mutex);
        pr_err("bad data blocksize %u\n", block_len);
        goto err;
} else if (uncompressed) {
        memcpy(pgdata,
               cramfs_read(sb, block_start, block_len),
               block_len);
        bytes_filled = block_len;
```

Excluded Transformation Nodes

Protect the critical section

- We strictly ensure that we cannot introduce new code to the critical section.

- This ensure we don't move code that can sleep into the critical section.

- Example:

```
lock(...);        guard(...)(...);
...          ...
if(...){ if(...){
  unlock(...);     fn_call(...);
  fn_call(...);    return ...;
  return ...;    }
}          return ...;
unlock(...);
return ...;
```

source: fs/btrfs/delayed-inode.c:1193

```
delayed_node = btrfs_get_delayed_node(inode, &delayed_node_tracker);
if (!delayed_node)
        return 0;

mutex_lock(&delayed_node->mutex);
if (!test_bit(BTRFS_DELAYED_NODE_INODE_DIRTY, &delayed_node->flags)) {
        mutex_unlock(&delayed_node->mutex);
        btrfs_release_delayed_node(delayed_node, &delayed_node_tracker);
        return 0;
}
mutex_unlock(&delayed_node->mutex);
```

Excluded Transformation node

No code before unlocks in gotos

- In cases where unlock occurs at goto statement, we must ensure no code exists before the unlock.
- This ensure we don't alter the critical section.

Example:

```
Lock(...);      scoped_guard(...){
If(...)      if(...)
  goto lbl:          goto lbl:
...          ...
unlock(...);  }
lbl:      lbl:
fn_call(...);  fn_call(...);
unlock(...);  return ...;
return ...;
```

source: arch/sparc/kernel/ldc.c:1404

```
        spin_lock_irqsave(&lp->lock, flags);

        err = -ENODEV;
        hv_err = sun4v_ldc_tx_qconf(lp->id, 0, 0);
        if (hv_err)
                goto out_err;

        ...

        spin_unlock_irqrestore(&lp->lock, flags);

        return 0;

out_err:
        sun4v_ldc_tx_qconf(lp->id, 0, 0);
        sun4v_ldc_rx_qconf(lp->id, 0, 0);
        free_irq(lp->cfg.tx_irq, lp);
        free_irq(lp->cfg.rx_irq, lp);
        lp->flags &= ~(LDC_FLAG_REGISTERED_IRQS |
                       LDC_FLAG_REGISTERED_QUEUES);
        ldc_set_state(lp, LDC_STATE_INIT);

        spin_unlock_irqrestore(&lp->lock, flags);

        return err;
```

Transformation Examples

```
+++ b/arch/powerpc/perf/imc-pmu.c
@@ -1636,17 +1636,17 @@ static void imc_common_mem_free(struct i
 static void imc_common_cpuhp_mem_free(struct imc_pmu *pmu_ptr)
 {
        if (pmu_ptr->domain == IMC_DOMAIN_NEST) {
-               mutex_lock(&nest_init_lock);
-               if (nest_pmus == 1) {
-                       cpuhp_remove_state(CPUHP_AP_PERF_POWERPC_NEST_IMC_ONLINE);
-                       kfree(nest_imc_refc);
-                       kfree(per_nest_pmu_arr);
-                       per_nest_pmu_arr = NULL;
-               }
+               scoped_guard (mutex, &nest_init_lock) {
+                       if (nest_pmus == 1) {
+                               cpuhp_remove_state(CPUHP_AP_PERF_POWERPC_NEST_IMC_ONLINE);
+                               kfree(nest_imc_refc);
+                               kfree(per_nest_pmu_arr);
+                               per_nest_pmu_arr = NULL;
+                       }

-                       if (nest_pmus > 0)
-                               nest_pmus--;
-               mutex_unlock(&nest_init_lock);
+                       if (nest_pmus > 0)
+                               nest_pmus--;
+               }
        }
```

Transformation Examples

```
 static void iomap_set_range_dirty(struct folio *folio, size_t off, size_t len)
@@ -326,14 +320,12 @@ static void iomap_finish_folio_read(stru
        bool finished = true;

        if (ifs) {
-               unsigned long flags;
-
-               spin_lock_irqsave(&ifs->state_lock, flags);
-               if (!error)
-                       uptodate = ifs_set_range_uptodate(folio, ifs, off, len);
-               ifs->read_bytes_pending -= len;
-               finished = !ifs->read_bytes_pending;
-               spin_unlock_irqrestore(&ifs->state_lock, flags);
+               scoped_guard (spinlock_irqsave, &ifs->state_lock) {
+                       if (!error)
+                               uptodate = ifs_set_range_uptodate(folio, ifs, off, len);
+                       ifs->read_bytes_pending -= len;
+                       finished = !ifs->read_bytes_pending;
+               }
        }
```

Conclusion

- How do we check for correctness of the transformation?
  - Compiling the kernel with the new changes (with relevant configuration set e.g CONFIG_DEBUG_SPINLOCK=y)
  - Manual diff reviews

- Current transformation opportunities where lock/unlock patterns can be replaced with scoped based macro.
  - Mutex: 21246
  - Spinlock: 9395
  - write_lock: 324
  - read_lock: 343
  - raw_spinlock_irqsave : 1057

    etc

Q&A


東京 2025
LINUX
PLUMBERS CONFERENCE
TOKYO, JAPAN / DEC. 11-13, 2025

References

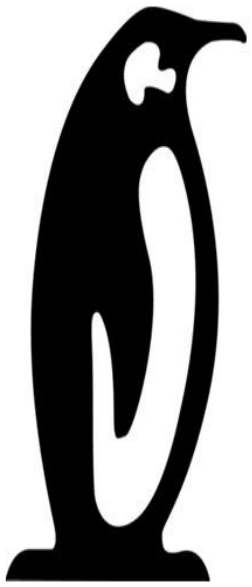
1. https://elixir.bootlin.com/linux/v6.17.9/source/include/linux/cleanup.h

2.Coccinelle: Program Matching and Transformation for C

http://coccinelle.lip6.fr

3.https://hackerbikepacker.com/kernel-auto-cleanup-2

4.https://www.spinics.net/lists/kernel/msg5897485.html



東京 2025
LINUX PLUMBERS CONFERENCE

TOKYO, JAPAN / DECEMBER 11-13, 2025