Using coccinelle to find (and fix!) nested execution context violations

Julia Cartwright

National Instruments

linux-rt-users@vger.kernel.org ?

```
BUG: sleeping function called from invalid context at kernel/locking/rtmutex.c:993
in_atomic(): 0, irqs_disabled(): 1, pid: 58, name: kworker/u12:1
CPU: 5 PID: 58 Comm: kworker/u12:1 Tainted: G        W       4.9.20-rt16-stand6-686 #1
Hardware name: Supermicro SYS-5027R-WRF/X9SRW-F, BIOS 3.2a 10/28/2015
Workqueue: writeback wb_workfn (flush-253:0)
 edf21a34 c12cdc6f c1078f0a edf08c40 edf21a64 c10787c1 c1668778 00000000
 00000001 0000003a edf09144 00000000 80000000 ec694b18 ec775748 ec694ad0
 edf21a70 c1584587 ec775790 edf21ac4 f893e8a3 00000000 c1078e52 00000000
Call Trace:
 [<c12cdc6f>] dump_stack+0x47/0x68
 [<c1078f0a>] ? migrate_enable+0x4a/0xf0
 [<c10787c1>] ___might_sleep+0x101/0x180
 [<c1584587>] rt_spin_lock+0x17/0x40
 [<f893e8a3>] add_stripe_bio+0x4e3/0x6c0 [raid456]
 [<c1078e52>] ? preempt_count_add+0x42/0xb0
 [<f89408f7>] raid5_make_request+0x737/0xdd0 [raid456]
```

```
static void lock_all(/* … */)
{
     int i;
     local_irq_disable();
     for (i = 0; i < NUM_LOCKS; i++)
       spin_lock(&locks[i]);
}

static int add_stripe_bio(/* ... */)
{
     lock_all();
}
```

Problem summary (for RT Summit audience):

Code executing within a local_irq_disable()/local_irq_enable() region runs with interrupts disabled on the local CPU.

Invoking schedule() implicitly or explicitly when interrupts are disabled is a context violation. (“sleeping in atomic context” per CONFIG_DEBUG_ATOMIC_SLEEP).

spin_lock() implicitly invokes schedule() on RT w/ “sleeping spinlocks”.

Code executing within a irqs-disabled region must not use spin_lock().

```
kernel BUG at kernel/locking/rtmutex.c:1014!
Internal error: Oops - BUG: 0 [#1] PREEMPT SMP
Modules linked in: spidev_irq(O) smsc75xx wcn36xx [last unloaded: spidev]
CPU: 0 PID: 1163 Comm: irq/144-mmc0 Tainted: G        W  O    4.4.9-linaro-lt-
qcom #1
PC is at rt_spin_lock_slowlock+0x80/0x2d8
LR is at rt_spin_lock_slowlock+0x68/0x2d8
[..]
Call trace:
  rt_spin_lock_slowlock
  rt_spin_lock
  msm_gpio_irq_ack
  handle_edge_irq
  generic_handle_irq
  msm_gpio_irq_handler
  generic_handle_irq
  __handle_domain_irq
  gic_handle_irq
```

```
static void msm_gpio_irq_ack(struct irq_data *d)
{
      struct gpio_chip *gc = irq_data_get_irq_chip_data(d);
      struct msm_pinctrl *pctrl = gpiochip_get_data(gc);

      /* ... */


      spin_lock_irqsave(&pctrl->lock, flags);
      /* ... */
      spin_lock_irqrestore(&pctrl->lock, flags);
}
```

Problem summary (for RT Summit audience):

Any code involved in interrupt dispatch must execute in hardirq context.

Invoking schedule() implicitly or explicitly in hardirq context is a context violation. (“sleeping in atomic context” per CONFIG_DEBUG_ATOMIC_SLEEP).

spin_lock() implicitly invokes schedule() on RT w/ “sleeping spinlocks”.

Code involved in interrupt dispatch must not use spin_lock().

- Driver developers don’t understand when to use spinlock_t vs. raw_spinlock_t.

From: Arnaldo Carvalho de Melo <acme@kernel.org>:

In drivers/infiniband/hw/hfi1/pio.c sc_buffer_alloc() disables preemption that will be reenabled by either pio_copy() or seg_pio_copy_end().

But before disabling preemption it grabs a spin lock that will be dropped after it disables preemption, which ends up triggering a warning in migrate_disable() later on.

```
  spin_lock_irqsave(&sc->alloc_lock)
    migrate_disable() ++p->migrate_disable -> 2
  preempt_disable()
  spin_unlock_irqrestore(&sc->alloc_lock)
    migrate_enable() in_atomic(), so just returns, migrate_disable stays at 2
  spin_lock_irqsave(some other lock) -> b00m
```

drivers/crypto/caam/jr.c:

```
static irqreturn_t caam_jr_interrupt(int irq, void *st_dev)
{
	/* ... */

	preempt_disable();
	tasklet_schedule(&jrp->irqtask);
	preempt_enable();
}

request_irq(jrp->irq, caam_jr_interrupt, IRQF_SHARED,
		dev_name(dev), dev);
```

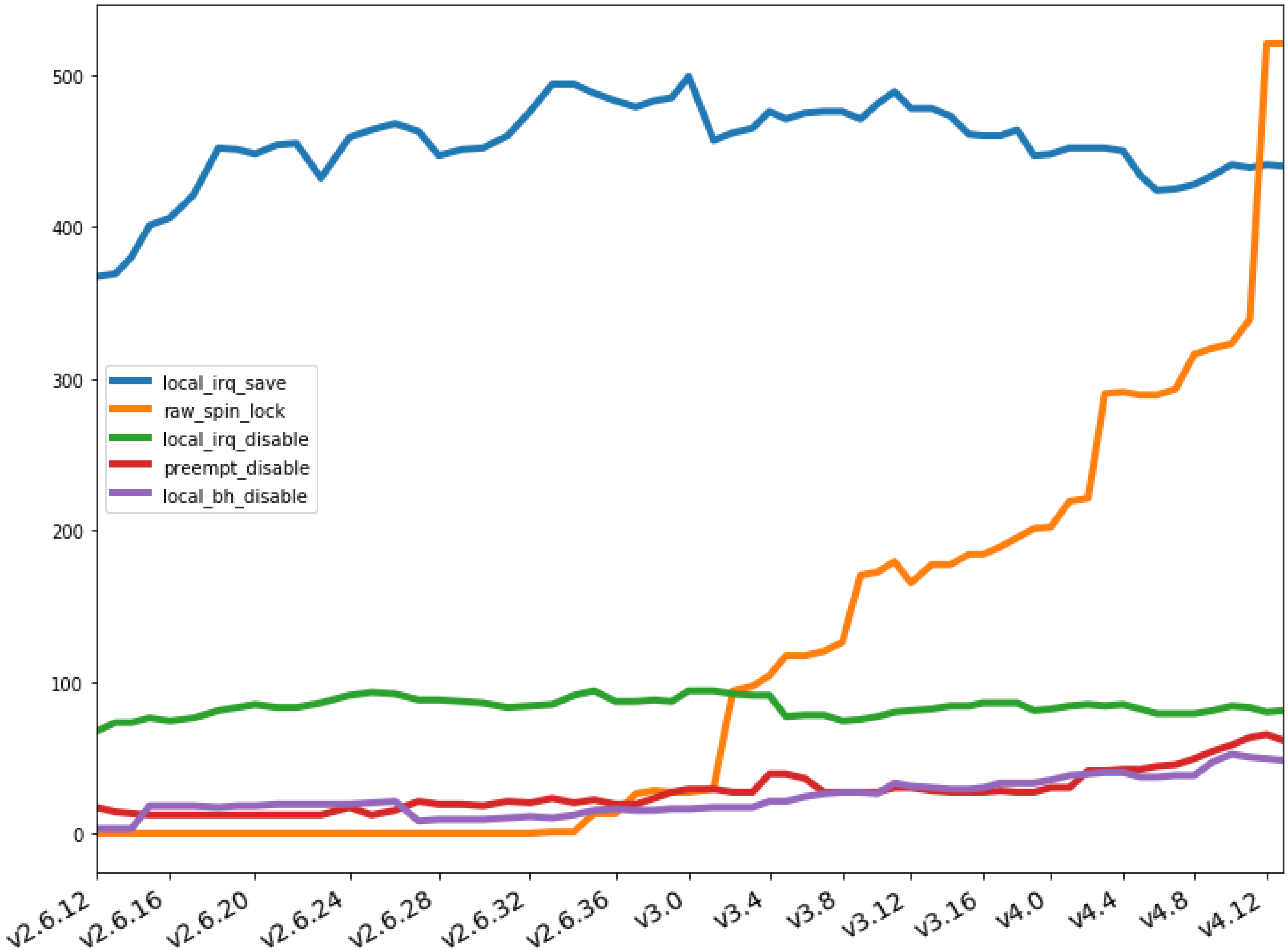
- Driver developers don’t understand when to use spinlock_t vs. raw_spinlock_t.

- Driver developers don’t understand when to use preempt_disable().

- Driver developers don’t understand [illegible] use spinlock_t vs. raw_spinloc[illegible]

- D[illegible] developers don’t understand when to use preempt_disable().

Driver developers don’t understand RT.



500
400
300
200
100
0
local_irq_save
raw_spin_lock
local_irq_disable
preempt_disable
local_bh_disable
v2.6.12
v2.6.16
v2.6.20
v2.6.24
v2.6.28
v2.6.32
v2.6.36
v3.0
v3.4
v3.8
v3.12
v3.16
v4.0
v4.4
v4.8
v4.12

- Static tools – coccinelle, others?
- Run-time tools – augment lockdep in mainline?
- Education:
  - “What every driver developer should know about RT”

- Static tools – coccinelle, others?
- Run-time tools – augment lockdep in mainline?
- Education:
  - “What every driver developer should know about RT”

```
static void lock_all(/* … */)
{
     int i;
     local_irq_disable();
     for (i = 0; i < NUM_LOCKS; i++)
        spin_lock(&locks[i]);
}

static int add_stripe_bio(/* ... */)
{
     lock_all();
}
```

```
virtual report

@r exists@
position p1;
position p2;
@@

(
local_irq_disable@p1
|
local_irq_save@p1
)(...);
... when != local_irq_enable(...)
    when != local_irq_restore(...)
(
spin_lock@p2
|
spin_lock_irq@p2
|
spin_lock_irqsave@p2
)(...);

@script:python depends on r && report@
p1 << r.p1;
p2 << r.p2;
@@

msg="ERROR: sleeping spinlock acquired, though interrupts disabled on line %s" % (p1[0].line)
coccilib.report.print_report(p2[0], msg)
```

./net/core/drop_monitor.c:166:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 164
./kernel/signal.c:1262:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 1243
./drivers/tty/serial/sh-sci.c:2856:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 2847
./drivers/tty/serial/bcm63xx_uart.c:718:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 711
./drivers/tty/serial/amba-pl011.c:2229:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 2223
./arch/x86/kernel/reboot.c:97:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 85
./arch/parisc/kernel/smp.c:130:2-19: ERROR: sleeping spinlock acquired, though interrupts disabled on line 181
./arch/mn10300/kernel/mn10300-serial.c:1594:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 1587
./mm/workingset.c:486:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 485
./lib/percpu_ida.c:163:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 152
./lib/percpu_ida.c:163:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 200
./lib/percpu_ida.c:352:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 349
./lib/percpu_ida.c:363:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 349
./lib/percpu_ida.c:228:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 225
./kernel/workqueue.c:2830:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 2823
./kernel/workqueue.c:1239:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 1212
./kernel/workqueue.c:4271:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 4268
./drivers/tty/serial/owl-uart.c:520:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 513
./drivers/tty/serial/omap-serial.c:1320:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 1314
./drivers/tty/serial/ar933x_uart.c:555:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 548
./drivers/scsi/libsas/sas_ata.c:254:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 193
./drivers/md/raid5-ppl.c:549:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 547
./drivers/infiniband/ulp/ipoib/ipoib_multicast.c:900:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 898
./drivers/ide/ide-io.c:682:2-15: ERROR: sleeping spinlock acquired, though interrupts disabled on line 663
./block/blk-core.c:3376:3-12: ERROR: sleeping spinlock acquired, though interrupts disabled on line 3363
./arch/x86/kvm/vmx.c:11772:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 11769
./arch/alpha/kernel/srmcons.c:81:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 74
./drivers/usb/gadget/udc/dummy_hcd.c:763:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 762
./drivers/tty/serial/stm32-usart.c:990:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 984
./drivers/tty/serial/pxa.c:659:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 653
./drivers/tty/serial/pch_uart.c:1666:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 1656
./drivers/tty/serial/lpc32xx_hs.c:153:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 147
./drivers/net/ethernet/tehuti/tehuti.c:1624:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 1623
./arch/powerpc/kvm/book3s_hv.c:2845:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 2709
./mm/kasan/quarantine.c:193:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 186
./drivers/tty/serial/meson_uart.c:477:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 471
./arch/parisc/kernel/traps.c:432:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 431
./arch/metag/kernel/smp.c:453:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 448

Mainline: 38 violations as of v4.14-rc5

RT: 22 violations as of v4.13.7-rt1

```
./drivers/tty/serial/lpc32xx_hs.c:153:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 147
./drivers/net/ethernet/tehuti/tehuti.c:1624:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 1623
./drivers/md/raid5-ppl.c:531:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 529
./arch/powerpc/kvm/book3s_hv.c:2828:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 2692
./mm/kasan/quarantine.c:193:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 186
./drivers/tty/serial/sh-sci.c:2856:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 2847
./drivers/tty/serial/ar933x_uart.c:555:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 548
./arch/x86/kvm/vmx.c:11470:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 11467
./net/core/drop_monitor.c:166:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 164
./drivers/usb/gadget/udc/dummy_hcd.c:761:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 760
./drivers/tty/serial/pch_uart.c:1671:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 1661
./arch/x86/kernel/reboot.c:99:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 87
./arch/parisc/kernel/traps.c:432:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 431
./arch/mn10300/kernel/mn10300-serial.c:1594:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 1587
./arch/metag/kernel/smp.c:453:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 448
./arch/alpha/kernel/srmcons.c:81:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 74
./drivers/tty/serial/stm32-usart.c:940:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 934
./drivers/tty/serial/pxa.c:659:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 653
./drivers/tty/serial/owl-uart.c:84:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 77
./drivers/tty/serial/meson_uart.c:477:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 471
./drivers/tty/serial/bcm63xx_uart.c:713:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 706
./arch/parisc/kernel/smp.c:130:2-19: ERROR: sleeping spinlock acquired, though interrupts disabled on line 181
```

```
kernel BUG at kernel/locking/rtmutex.c:1014!
Internal error: Oops - BUG: 0 [#1] PREEMPT SMP
Modules linked in: spidev_irq(O) smsc75xx wcn36xx [last unloaded: spidev]
CPU: 0 PID: 1163 Comm: irq/144-mmc0 Tainted: G        W  O    4.4.9-linaro-lt-
qcom #1
PC is at rt_spin_lock_slowlock+0x80/0x2d8
LR is at rt_spin_lock_slowlock+0x68/0x2d8
[..]
Call trace:
  rt_spin_lock_slowlock
  rt_spin_lock
  msm_gpio_irq_ack
  handle_edge_irq
  generic_handle_irq
  msm_gpio_irq_handler
  generic_handle_irq
  __handle_domain_irq
  gic_handle_irq
```

```
static void msm_gpio_irq_ack(struct irq_data *d)
{
      struct gpio_chip *gc = irq_data_get_irq_chip_data(d);
      struct msm_pinctrl *pctrl = gpiochip_get_data(gc);

      /* ... */


      spin_lock_irqsave(&pctrl->lock, flags);
      /* ... */
      spin_lock_irqrestore(&pctrl->lock, flags);
}
```

```
virtual report

@r exists@
position p1;
position p2;
@@

(
local_irq_disable@p1
|
local_irq_save@p1
)(...);
... when != local_irq_enable(...)
    when != local_irq_restore(...)
(
spin_lock@p2
|
spin_lock_irq@p2
|
spin_lock_irqsave@p2
)(...);

@script:python depends on r && report@
p1 << r.p1;
p2 << r.p2;
@@

msg="ERROR: sleeping spinlock acquired, though interrupts disabled on line %s" % (p1[0].line)
coccilib.report.print_report(p2[0], msg)
```

No explicit local_irq_disable()!

- Coccinelle allows for specifying dependent rules.
- Captured metavariables within a rule, are made available to dependent rules.

- Strategy
  - rule1: match functions which execute with interrupts disabled
  - rule2: use matched function in rule1 and seek patterns within it’s body

```
static void msm_gpio_irq_ack(struct irq_data *d);

static struct irq_chip msm_gpio_irq_chip = {
     .name            = "msmgpio",
     /* … */
     .irq_ack         = msm_gpio_irq_ack,
     /* … */
};
```

```
virtual report

@rule1@
identifier __irqchip;
identifier __irq_ack;
@@
static struct irq_chip __irqchip = {
      .irq_ack   = __irq_ack,
};
```

```
@rule2 depends on rule1 exists@
identifier rule1.__irq_ack;
identifier data;
position j0;
@@
static void __irq_ack(struct irq_data *data)
{
	... when any
(
	spin_lock_irqsave@j0
|
	spin_lock_irq@j0
|
	spin_lock@j0
)(...);
	... when any
}

@script:python simple depends on rule2 && report@
j0 << rule2.j0;
@@

msg = "Use of non-raw spinlock is illegal in this context"
coccilib.report.print_report(j0[0], msg)
```

./drivers/pinctrl/sirf/pinctrl-sirf.c:432:1-18: Use of non-raw spinlock is illegal in this context

./arch/mips/bcm63xx/irq.c:265:1-18: Use of non-raw spinlock is illegal in this context

./drivers/pinctrl/pinctrl-adi2.c:262:1-18: Use of non-raw spinlock is illegal in this context

./drivers/pinctrl/pinctrl-adi2.c:263:1-10: Use of non-raw spinlock is illegal in this context

./drivers/gpio/gpio-aspeed.c:352:1-18: Use of non-raw spinlock is illegal in this context

Mainline & RT:
5 violations

That’s all with coccinelle’s *report* mode. What about *patch* mode?

Strategy

- rule1: match functions which execute with interrupts disabled
- rule2: use matched function in rule1 and seek patterns within it’s body

```
static void msm_gpio_irq_ack(struct irq_data *d)
{
	struct gpio_chip *gc = irq_data_get_irq_chip_data(d);
	struct msm_pinctrl *pctrl = gpiochip_get_data(gc);

	/* ... */


	spin_lock_irqsave(&pctrl->lock, flags);
	/* ... */
	spin_lock_irqrestore(&pctrl->lock, flags);
}
```

Strategy

- rule1: match functions which execute with interrupts disabled
- **augmented** rule2: use matched function in rule1 and seek patterns within it’s body, **capturing the type and member name of the relevant lock**
- rule3: use captured type and member name to generate hunk for changing lock type
- rule4: use type and member name to generate hunks for updating spin_lock*() callers

```
@rule2 depends on rule1 exists@
identifier rule1.__irq_ack;
identifier data, x, l;
type T;
position j0;
expression flags;
@@
static void __irq_ack(struct irq_data *data)
{
     ... when any
     T *x;
     ... when any
(
     spin_lock_irqsave@j0(&x->l, flags);
|
     spin_lock_irq@j0(&x->l);
|
     spin_lock@j0(&x->l);
)
     ... when any
}

@script:python simple depends on rule2 && report@
j0 << rule2.j0;
t << rule2.T;
l << rule2.l;
@@

msg = "Use of non-raw spinlock is illegal in this context (%s::%s)" % (t, l)
coccilib.report.print_report(j0[0], msg)
```

```
@rule3 depends on rule2 && patch@
type rule2.T;
identifier rule2.l;
@@
T {
    ...
-   spinlock_t l;
+   raw_spinlock_t l;
    ...
};
```

```
@rule4 depends on rule2 && patch@
type rule2.T;
identifier rule2.l;
expression flags;
T *x;
@@

(
-spin_lock(&x->l)
+raw_spin_lock(&x->l)
|
-spin_lock_irqsave(&x->l, flags)
+raw_spin_lock_irqsave(&x->l, flags)
|
-spin_lock_irq(&x->l)
+raw_spin_lock_irq(&x->l)
|
-spin_unlock(&x->l)
+raw_spin_unlock(&x->l)
|
-spin_unlock_irq(&x->l)
+raw_spin_unlock_irq(&x->l)
|
-spin_unlock_irqrestore(&x->l, flags)
+raw_spin_unlock_irqrestore(&x->l, flags)
|
-spin_lock_init(&x->l)
+raw_spin_lock_init(&x->l)
)
```

Limitations

- Doesn’t really eliminate the “hard work”
- Right now: only looks at irq_chip callbacks

?

julia@ni.com

julia_c

linux-rt-users

What every driver developer should know about RT.

All code executes within a *context.*

The *context* in which code executes is determined by it’s caller, and by explicit action, namely:

Code may enter a more restrictive *context* by invoking a *context enter* function, and may exit via the invocation of a *context exit* function.

NMI context

hardirq context

softirq context

preemption disabled context

migration disabled context

```
static irqreturn_t foo_handler(int, void *)
{
   /* executes in a context */
   return IRQ_HANDLED;
}

static int foo_probe(...)
{
   /* ... */
   ret = request_irq(irq, foo_handler,
     NULL, 0);
      /* ... */
}
```

thread context

- “interrupts disabled”
- “preemption disabled”
- “migration disabled”

- **“interrupts disabled”**

Per-CPU bit indicating whether or not hardware-triggered interrupts can be serviced.

Explicit:
local_irq_{disable,enable}()
local_irq_{save,restore}(flags)

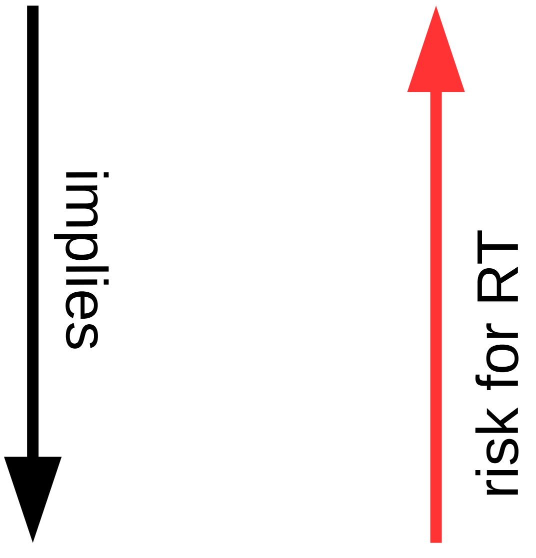
Implicit:
raw_spin_{lock,unlock}_irq(lock)
raw_spin_{,un}lock_irq{save,restore}(lock, flags)
~~spin_{lock,unlock}_irq(lock)~~
~~spin_{,un}lock_irq{save,restore}(lock, flags)~~

On RT, spin_lock critical sections do <u>not</u> execute with interrupts disabled.

- “interrupts disabled”
- “preemption disabled”
- “migration disabled”

implies

- “interrupts disabled”
- “preemption disabled”
- “migration disabled”



- Driver developers don’t understand when to use spinlock_t vs. raw_spinlock_t.

- Driver developers don’t understand when to use spinlock_t vs. raw_spinlock_t.

- Driver developers don’t understand the impact of using preempt_disable().

- raw vs. atomic notifiers

```
@rule2 depends on rule1 exists@
identifier rule1.__irq_ack;
identifier data;
position j0;
expression flags;
@@
static void __irq_ack(struct irq_data *data)
{
     ... when any
(
     spin_lock_irqsave@j0
|
     spin_lock_irq@j0
|
     spin_lock@j0
)(...);
     ... when any
}

@script:python simple depends on rule2 && report@
j0 << rule2.j0;
@@

msg = "Use of non-raw spinlock is illegal in this context"
coccilib.report.print_report(j0[0], msg)
```

Using coccinelle to find (and fix!) nested execution context violations

Julia Cartwright

National Instruments

linux-rt-users@vger.kernel.org ?

```
BUG: sleeping function called from invalid context at kernel/locking/rtmutex.c:993
in_atomic(): 0, irqs_disabled(): 1, pid: 58, name: kworker/u12:1
CPU: 5 PID: 58 Comm: kworker/u12:1 Tainted: G        W       4.9.20-rt16-stand6-686 #1
Hardware name: Supermicro SYS-5027R-WRF/X9SRW-F, BIOS 3.2a 10/28/2015
Workqueue: writeback wb_workfn (flush-253:0)
 edf21a34 c12cdc6f c1078f0a edf08c40 edf21a64 c10787c1 c1668778 00000000
 00000001 0000003a edf09144 00000000 80000000 ec694b18 ec775748 ec694ad0
 edf21a70 c1584587 ec775790 edf21ac4 f893e8a3 00000000 c1078e52 00000000
Call Trace:
 [<c12cdc6f>] dump_stack+0x47/0x68
 [<c1078f0a>] ? migrate_enable+0x4a/0xf0
 [<c10787c1>] ___might_sleep+0x101/0x180
 [<c1584587>] rt_spin_lock+0x17/0x40
 [<f893e8a3>] add_stripe_bio+0x4e3/0x6c0 [raid456]
 [<c1078e52>] ? preempt_count_add+0x42/0xb0
 [<f89408f7>] raid5_make_request+0x737/0xdd0 [raid456]
```

```
static void lock_all(/* … */)
{
     int i;
     local_irq_disable();
     for (i = 0; i < NUM_LOCKS; i++)
        spin_lock(&locks[i]);
}

static int add_stripe_bio(/* ... */)
{
     lock_all();
}
```

Problem summary (for RT Summit audience):

Code executing within a local_irq_disable()/local_irq_enable() region runs with interrupts disabled on the local CPU.

Invoking schedule() implicitly or explicitly when interrupts are disabled is a context violation. (“sleeping in atomic context” per CONFIG_DEBUG_ATOMIC_SLEEP).

spin_lock() implicitly invokes schedule() on RT w/ “sleeping spinlocks”.

Code executing within a irqs-disabled region must not use spin_lock().

```
kernel BUG at kernel/locking/rtmutex.c:1014!
Internal error: Oops - BUG: 0 [#1] PREEMPT SMP
Modules linked in: spidev_irq(O) smsc75xx wcn36xx [last unloaded: spidev]
CPU: 0 PID: 1163 Comm: irq/144-mmc0 Tainted: G        W  O    4.4.9-linaro-lt-
qcom #1
PC is at rt_spin_lock_slowlock+0x80/0x2d8
LR is at rt_spin_lock_slowlock+0x68/0x2d8
[..]
Call trace:
  rt_spin_lock_slowlock
  rt_spin_lock
  msm_gpio_irq_ack
  handle_edge_irq
  generic_handle_irq
  msm_gpio_irq_handler
  generic_handle_irq
  __handle_domain_irq
  gic_handle_irq
```

```
static void msm_gpio_irq_ack(struct irq_data *d)
{
	struct gpio_chip *gc = irq_data_get_irq_chip_data(d);
	struct msm_pinctrl *pctrl = gpiochip_get_data(gc);

	/* ... */


	spin_lock_irqsave(&pctrl->lock, flags);
	/* ... */
	spin_lock_irqrestore(&pctrl->lock, flags);
}
```

<u>Problem summary (for RT Summit audience):</u>

Any code involved in interrupt dispatch must execute in hardirq context.

Invoking schedule() implicitly or explicitly in hardirq context is a context violation. (“sleeping in atomic context” per CONFIG_DEBUG_ATOMIC_SLEEP).

spin_lock() implicitly invokes schedule() on RT w/ “sleeping spinlocks”.

Code involved in interrupt dispatch must not use spin_lock().

- Driver developers don’t understand when to use spinlock_t vs. raw_spinlock_t.

From: Arnaldo Carvalho de Melo <acme@kernel.org>:

In drivers/infiniband/hw/hfi1/pio.c sc_buffer_alloc() disables preemption that will be reenabled by either pio_copy() or seg_pio_copy_end().

But before disabling preemption it grabs a spin lock that will be dropped after it disables preemption, which ends up triggering a warning in migrate_disable() later on.

```
  spin_lock_irqsave(&sc->alloc_lock)
    migrate_disable() ++p->migrate_disable -> 2
  preempt_disable()
  spin_unlock_irqrestore(&sc->alloc_lock)
    migrate_enable() in_atomic(), so just returns, migrate_disable stays at 2
  spin_lock_irqsave(some other lock) -> b00m
```

```
drivers/crypto/caam/jr.c:

static irqreturn_t caam_jr_interrupt(int irq, void *st_dev)
{
     /* ... */

     preempt_disable();
     tasklet_schedule(&jrp->irqtask);
     preempt_enable();
}

request_irq(jrp->irq, caam_jr_interrupt, IRQF_SHARED,
            dev_name(dev), dev);
```

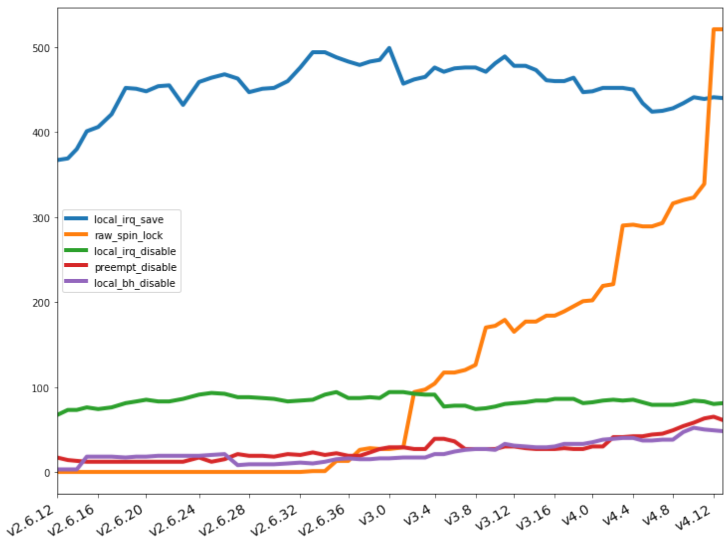
- Driver developers don’t understand when to use spinlock_t vs. raw_spinlock_t.

- Driver developers don’t understand when to use preempt_disable().

- Driver developers don’t understand [illegible] use spinlock_t vs. raw_spinlo[illegible]

- D[illegible] developers don’t understand when to use preempt_disable().

Driver developers don’t understand RT.



local_irq_save
raw_spin_lock
local_irq_disable
preempt_disable
local_bh_disable
0
100
200
300
400
500
v2.6.12
v2.6.16
v2.6.20
v2.6.24
v2.6.28
v2.6.32
v2.6.36
v3.0
v3.4
v3.8
v3.12
v3.16
v4.0
v4.4
v4.8
v4.12

- Static tools – coccinelle, others?
- Run-time tools – augment lockdep in mainline?
- Education:
  - “What every driver developer should know about RT”

- Static tools – coccinelle, others?
- Run-time tools – augment lockdep in mainline?
- Education:
  - “What every driver developer should know about RT”

```
static void lock_all(/* … */)
{
	int i;
	local_irq_disable();
	for (i = 0; i < NUM_LOCKS; i++)
	  spin_lock(&locks[i]);
}

static int add_stripe_bio(/* ... */)
{
	lock_all();
}
```

```
virtual report

@r exists@
position p1;
position p2;
@@

(
local_irq_disable@p1
|
local_irq_save@p1
)(...);
... when != local_irq_enable(...)
    when != local_irq_restore(...)
(
spin_lock@p2
|
spin_lock_irq@p2
|
spin_lock_irqsave@p2
)(...);

@script:python depends on r && report@
p1 << r.p1;
p2 << r.p2;
@@

msg="ERROR: sleeping spinlock acquired, though interrupts disabled on line %s" % (p1[0].line)
coccilib.report.print_report(p2[0], msg)
```

Mainline: 38 violations as of v4.14-rc5

./net/core/drop_monitor.c:166:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 164
./kernel/signal.c:1262:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 1243
./drivers/tty/serial/sh-sci.c:2856:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 2847
./drivers/tty/serial/bcm63xx_uart.c:718:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 711
./drivers/tty/serial/amba-pl011.c:2229:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 2223
./arch/x86/kernel/reboot.c:97:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 85
./arch/parisc/kernel/smp.c:130:2-19: ERROR: sleeping spinlock acquired, though interrupts disabled on line 181
./arch/mn10300/kernel/mn10300-serial.c:1594:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 1587
./mm/workingset.c:486:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 485
./lib/percpu_ida.c:163:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 152
./lib/percpu_ida.c:163:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 200
./lib/percpu_ida.c:352:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 349
./lib/percpu_ida.c:363:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 349
./lib/percpu_ida.c:228:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 225
./kernel/workqueue.c:2830:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 2823
./kernel/workqueue.c:1239:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 1212
./kernel/workqueue.c:4271:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 4268
./drivers/tty/serial/owl-uart.c:520:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 513
./drivers/tty/serial/omap-serial.c:1320:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 1314
./drivers/tty/serial/ar933x_uart.c:555:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 548
./drivers/scsi/libsas/sas_ata.c:254:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 193
./drivers/md/raid5-ppl.c:549:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 547
./drivers/infiniband/ulp/ipoib/ipoib_multicast.c:900:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 898
./drivers/ide/ide-io.c:682:2-15: ERROR: sleeping spinlock acquired, though interrupts disabled on line 663
./block/blk-core.c:3376:3-12: ERROR: sleeping spinlock acquired, though interrupts disabled on line 3363
./arch/x86/kvm/vmx.c:11772:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 11769
./arch/alpha/kernel/srmcons.c:81:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 74
./drivers/usb/gadget/udc/dummy_hcd.c:763:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 762
./drivers/tty/serial/stm32-usart.c:990:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 984
./drivers/tty/serial/pxa.c:659:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 653
./drivers/tty/serial/pch_uart.c:1666:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 1656
./drivers/tty/serial/lpc32xx_hs.c:153:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 147
./drivers/net/ethernet/tehuti/tehuti.c:1624:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 1623
./arch/powerpc/kvm/book3s_hv.c:2845:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 2709
./mm/kasan/quarantine.c:193:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 186
./drivers/tty/serial/meson_uart.c:477:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 471
./arch/parisc/kernel/traps.c:432:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 431
./arch/metag/kernel/smp.c:453:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 448

RT: 22 violations as of v4.13.7-rt1

```
./drivers/tty/serial/lpc32xx_hs.c:153:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 147
./drivers/net/ethernet/tehuti/tehuti.c:1624:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 1623
./drivers/md/raid5-ppl.c:531:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 529
./arch/powerpc/kvm/book3s_hv.c:2828:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 2692
./mm/kasan/quarantine.c:193:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 186
./drivers/tty/serial/sh-sci.c:2856:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 2847
./drivers/tty/serial/ar933x_uart.c:555:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 548
./arch/x86/kvm/vmx.c:11470:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 11467
./net/core/drop_monitor.c:166:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 164
./drivers/usb/gadget/udc/dummy_hcd.c:761:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 760
./drivers/tty/serial/pch_uart.c:1671:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 1661
./arch/x86/kernel/reboot.c:99:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 87
./arch/parisc/kernel/traps.c:432:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 431
./arch/mn10300/kernel/mn10300-serial.c:1594:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 1587
./arch/metag/kernel/smp.c:453:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 448
./arch/alpha/kernel/srmcons.c:81:1-10: ERROR: sleeping spinlock acquired, though interrupts disabled on line 74
./drivers/tty/serial/stm32-usart.c:940:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 934
./drivers/tty/serial/pxa.c:659:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 653
./drivers/tty/serial/owl-uart.c:84:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 77
./drivers/tty/serial/meson_uart.c:477:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 471
./drivers/tty/serial/bcm63xx_uart.c:713:2-11: ERROR: sleeping spinlock acquired, though interrupts disabled on line 706
./arch/parisc/kernel/smp.c:130:2-19: ERROR: sleeping spinlock acquired, though interrupts disabled on line 181
```

```
kernel BUG at kernel/locking/rtmutex.c:1014!
Internal error: Oops - BUG: 0 [#1] PREEMPT SMP
Modules linked in: spidev_irq(O) smsc75xx wcn36xx [last unloaded: spidev]
CPU: 0 PID: 1163 Comm: irq/144-mmc0 Tainted: G        W  O    4.4.9-linaro-lt-
qcom #1
PC is at rt_spin_lock_slowlock+0x80/0x2d8
LR is at rt_spin_lock_slowlock+0x68/0x2d8
[..]
Call trace:
  rt_spin_lock_slowlock
  rt_spin_lock
  msm_gpio_irq_ack
  handle_edge_irq
  generic_handle_irq
  msm_gpio_irq_handler
  generic_handle_irq
  __handle_domain_irq
  gic_handle_irq
```

```
static void msm_gpio_irq_ack(struct irq_data *d)
{
	struct gpio_chip *gc = irq_data_get_irq_chip_data(d);
	struct msm_pinctrl *pctrl = gpiochip_get_data(gc);

	/* ... */


	spin_lock_irqsave(&pctrl->lock, flags);
	/* ... */
	spin_lock_irqrestore(&pctrl->lock, flags);
}
```

```
virtual report

@r exists@
position p1;
position p2;
@@
(
local_irq_disable@p1
|
local_irq_save@p1
)(...);
... when != local_irq_enable(...)
    when != local_irq_restore(...)
(
spin_lock@p2
|
spin_lock_irq@p2
|
spin_lock_irqsave@p2
)(...);

@script:python depends on r && report@
p1 << r.p1;
p2 << r.p2;
@@

msg="ERROR: sleeping spinlock acquired, though interrupts disabled on line %s" % (p1[0].line)
coccilib.report.print_report(p2[0], msg)
```

No explicit local_irq_disable()!

- Coccinelle allows for specifying dependent rules.
- Captured metavariables within a rule, are made available to dependent rules.

- Strategy
  - rule1: match functions which execute with interrupts disabled
  - rule2: use matched function in rule1 and seek patterns within it’s body

```
static void msm_gpio_irq_ack(struct irq_data *d);

static struct irq_chip msm_gpio_irq_chip = {
	.name		= "msmgpio",
	/* … */
	.irq_ack	= msm_gpio_irq_ack,
	/* … */
};
```

```
virtual report

@rule1@
identifier __irqchip;
identifier __irq_ack;
@@
static struct irq_chip __irqchip = {
      .irq_ack   = __irq_ack,
};
```

```
@rule2 depends on rule1 exists@
identifier rule1.__irq_ack;
identifier data;
position j0;
@@
static void __irq_ack(struct irq_data *data)
{
     ... when any
(
     spin_lock_irqsave@j0
|
     spin_lock_irq@j0
|
     spin_lock@j0
)(...);
     ... when any
}

@script:python simple depends on rule2 && report@
j0 << rule2.j0;
@@

msg = "Use of non-raw spinlock is illegal in this context"
coccilib.report.print_report(j0[0], msg)
```

./drivers/pinctrl/sirf/pinctrl-sirf.c:432:1-18: Use of non-raw spinlock is illegal in this context

./arch/mips/bcm63xx/irq.c:265:1-18: Use of non-raw spinlock is illegal in this context

./drivers/pinctrl/pinctrl-adi2.c:262:1-18: Use of non-raw spinlock is illegal in this context

./drivers/pinctrl/pinctrl-adi2.c:263:1-10: Use of non-raw spinlock is illegal in this context

./drivers/gpio/gpio-aspeed.c:352:1-18: Use of non-raw spinlock is illegal in this context

Mainline & RT:
5 violations

That’s all with coccinelle’s *report* mode. What about *patch* mode?

Strategy

- rule1: match functions which execute with interrupts disabled
- rule2: use matched function in rule1 and seek patterns within it’s body

```
static void msm_gpio_irq_ack(struct irq_data *d)
{
	struct gpio_chip *gc = irq_data_get_irq_chip_data(d);
	struct msm_pinctrl *pctrl = gpiochip_get_data(gc);

	/* ... */


	spin_lock_irqsave(&pctrl->lock, flags);
	/* ... */
	spin_lock_irqrestore(&pctrl->lock, flags);
}
```

Strategy

- rule1: match functions which execute with interrupts disabled
- **augmented** rule2: use matched function in rule1 and seek patterns within it’s body, **capturing the type and member name of the relevant lock**
- rule3: use captured type and member name to generate hunk for changing lock type
- rule4: use type and member name to generate hunks for updating spin_lock*() callers

```
@rule2 depends on rule1 exists@
identifier rule1.__irq_ack;
identifier data, x, l;
type T;
position j0;
expression flags;
@@
static void __irq_ack(struct irq_data *data)
{
     ... when any
     T *x;
     ... when any
(
     spin_lock_irqsave@j0(&x->l, flags);
|
     spin_lock_irq@j0(&x->l);
|
     spin_lock@j0(&x->l);
)
     ... when any
}

@script:python simple depends on rule2 && report@
j0 << rule2.j0;
t << rule2.T;
l << rule2.l;
@@

msg = "Use of non-raw spinlock is illegal in this context (%s::%s)" % (t, l)
coccilib.report.print_report(j0[0], msg)
```

```
@rule3 depends on rule2 && patch@
type rule2.T;
identifier rule2.l;
@@
T {
    ...
-   spinlock_t l;
+   raw_spinlock_t l;
    ...
};
```

```
@rule4 depends on rule2 && patch@
type rule2.T;
identifier rule2.l;
expression flags;
T *x;
@@

(
-spin_lock(&x->l)
+raw_spin_lock(&x->l)
|
-spin_lock_irqsave(&x->l, flags)
+raw_spin_lock_irqsave(&x->l, flags)
|
-spin_lock_irq(&x->l)
+raw_spin_lock_irq(&x->l)
|
-spin_unlock(&x->l)
+raw_spin_unlock(&x->l)
|
-spin_unlock_irq(&x->l)
+raw_spin_unlock_irq(&x->l)
|
-spin_unlock_irqrestore(&x->l, flags)
+raw_spin_unlock_irqrestore(&x->l, flags)
|
-spin_lock_init(&x->l)
+raw_spin_lock_init(&x->l)
)
```

Limitations

- Doesn’t really eliminate the “hard work”
- Right now: only looks at irq_chip callbacks

?

julia@ni.com

julia_c

linux-rt-users

What every driver developer should know about RT.

All code executes within a *context.*

The *context* in which code executes is determined by it’s caller, and by explicit action, namely:

Code may enter a more restrictive *context* by invoking a *context enter* function, and may exit via the invocation of a *context exit* function.

NMI context

hardirq context

softirq context

preemption disabled context

migration disabled context

```
static irqreturn_t foo_handler(int, void *)
{
   /* executes in a context */
   return IRQ_HANDLED;
}

static int foo_probe(...)
{
   /* ... */
   ret = request_irq(irq, foo_handler,
     NULL, 0);
     /* ... */
}
```

thread context

- “interrupts disabled”
- “preemption disabled”
- “migration disabled”

- **“interrupts disabled”**

Per-CPU bit indicating whether or not hardware-triggered interrupts can be serviced.

Explicit:
local_irq_{disable,enable}()
local_irq_{save,restore}(flags)

Implicit:
raw_spin_{lock,unlock}_irq(lock)
raw_spin_{,un}lock_irq{save,restore}(lock, flags)
~~spin_{lock,unlock}_irq(lock)~~
~~spin_{,un}lock_irq{save,restore}(lock, flags)~~

On RT, spin_lock critical sections do <u>not</u> execute with interrupts disabled.

- “interrupts disabled”
- “preemption disabled”
- “migration disabled”

implies

- “interrupts disabled”
- “preemption disabled”
- “migration disabled”

implies

risk for RT

- Driver developers don’t understand when to use spinlock_t vs. raw_spinlock_t.

- Driver developers don’t understand when to use spinlock_t vs. raw_spinlock_t.

- Driver developers don’t understand the impact of using preempt_disable().

- raw vs. atomic notifiers

```
@rule2 depends on rule1 exists@
identifier rule1.__irq_ack;
identifier data;
position j0;
expression flags;
@@
static void __irq_ack(struct irq_data *data)
{
     ... when any
(
     spin_lock_irqsave@j0
|
     spin_lock_irq@j0
|
     spin_lock@j0
)(...);
     ... when any
}

@script:python simple depends on rule2 && report@
j0 << rule2.j0;
@@

msg = "Use of non-raw spinlock is illegal in this context"
coccilib.report.print_report(j0[0], msg)
```