

What NOT to do in kernel drivers

Arjan van de Ven
arjanv@redhat.com



Program

- Allocating memory
- Synchronisation primitives
- SMP issues
- Architecture
- Delays



Goal

- Avoiding future bugs
- No insult intended
- Examples might be fixed in recent trees



Allocating Memory

- Be prepared to handle failure!
- Be careful with abstracting allocation
- THINK about memory allocation

```
static int uhci_submit_iso_urb (urb_t *urb)
{
...
    tdm = kmalloc(some_size, in_interrupt()
        ? GFP_ATOMIC : GFP_KERNEL);
...
}
```

from `usb-uhci.c` in kernel 2.4.9



GFP Flags

- GFP_KERNEL vs GFP_ATOMIC
- GFP_DMA
- GFP_NOIO and GFP_NOFS



Write out path

- Filesystems
- Block device drivers



Stack

- for *small*, temporary data you can use the stack
- Kernel stack is about 4Kb usable
- Don't pass stack variables to registration functions



Synchronisation Primitives

- trust Rusty's Hamster:
<http://netfilter.samba.org/unreliable-guides>



Reinventing the square wheel

- Linux has a complete set of locks
- Linux locks are optimized by the architecture code
- Writing race-free locks is *hard*



Example

```
unsigned long LinuxEnterCriticalSection(DeviceInfo_pt pDevice)
{
    int retval;
#ifdef __SMP__
    for (;;) {
        retval=test_and_set_bit(SEMAPHORE_SRL,&pDevice->Semaphore_SRL);
        if (!retval)
            break;
        sleep_on(&pDevice->WaitQ);
    }
    return(retval);
#else /* __SMP__ */
    if (pDevice->Semaphore_SRL)
        return(-1);
    pDevice->Semaphore_SRL=-1;
    return(0);
#endif
}
```

bcm5820 driver, version 1.11



Example 2: Recursive spinlocks

```
static void
lcs_chan_lock_func(lcs_chan_globals * chan_globals)
    if (chan_globals->lock_owner != smp_processor_id()) {
        spin_lock_irqsave(chan_globals->subchannel,
                           chan_globals->flags);
        chan_globals->lock_cnt = 1;
        chan_globals->lock_owner = smp_processor_id();

    } else
        chan_globals->lock_cnt++;
}
```

IBM S/390 LCS network driver





SMP: Read/Write locks

- Starvation
 - r/w spinlocks
 - r/w semaphores
- Recursion
 - r/w spinlocks
 - r/w semaphores
- r/w locks are slower on some platforms



spinlocks and things that sleep

```
if(copy_to_user(filebuffer,read_position,bytes_to_write))
{
    spin_unlock_irqrestore(&audit_spin_lock, flags);
    return -EFAULT;
}
```

SNARE security audit module 0.9

- copy_to_user() / copy_from_user()
- request_irq()
- obvious: kmalloc(), down(), schedule(), filesystem ops



Other items

- Even if you don't have hardware, test with the SMP kernel!
- spin_trylock is almost always a bug
- keep it simple stupid



cli()

```
int mptscsih_detect(Scsi_Host_Template *tpnt)
{
    unsigned long flags;
    save_flags(flags);
    cli();
    /* do stuff */
    restore_flags(flags);
}
```

LSI Logic Fusion driver



cli() continued

- Using `cli()` is often (hiding) a bug
- Kills SMP performance
- Used to hide lack of locking(design)
- Should be avoided at all cost for new code



sleep_on()

- Popular in other Unices
- Very hard to use right
- Over 99% of current users are buggy
- Should be avoided at all cost for all code



Example

```
static void ms_delay(int ms)
{
    unsigned long flags;
    int ticks;
    if (ms > 0) {
        save_flags(flags);
        cli();
        while (ms_busy == 0)
            sleep_on(&ms_wait);
        ms_busy = 0;
        restore_flags(flags);
        ticks = MS_TICKS*ms-1;
        ciaa.tblo=ticks%256;
        ciaa.tbhi=ticks/256;
        ciaa.crb=0x19;
        sleep_on(&ms_wait);
    }
}
```

2.4.18 drivers/block/amiflop.c



All the world is a VAX

```
typedef unsigned long UWORD32;
```

```
Cisco iSCSI client
```

```
#ifdef ALPHA
```

```
#define U32    unsigned int
```

```
#else
```

```
#define U32    unsigned long
```

```
#endif
```

```
drivers/scsi/inia100.h 2.4.9
```



Read Documentation/DMA-mapping.txt

```
#ifdef __ia64__
    u64    mask = 0xffffffffffff;
#endif
    if (pci_enable_device(pdev))
        return r;
#ifdef __ia64__
    if (!pci_set_dma_mask(pdev, mask)) {
        dprintk((KERN_INFO "64 BIT PCI BUS DMA ADDRESSING SUPPORTED\n"));
    } else if (pci_set_dma_mask(pdev, (u64) 0xffffffff)) {
        printk(KERN_WARNING "32 BIT PCI BUS DMA ADDRESSING NOT SUPPORTED\n");
        return r;
    }
#else
    if (pci_set_dma_mask(pdev, (u64) 0xffffffff)) {
        printk(KERN_WARNING "32 BIT PCI BUS DMA ADDRESSING NOT SUPPORTED\n");
        return r;
    }
#endif
#endif
```

LSI Fusion driver



Weakly ordered memory

- the cpu can reorder writes as visible to the outside
- the cpu can delay making writes visible
- solution: use memory barriers: `wmb()` `rmb()`



PCI posting

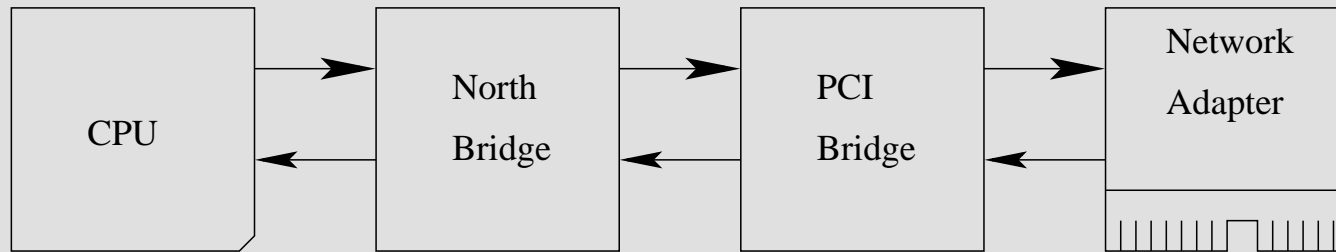


Figure 1: PCI system

- Problem: Writes can get delayed for arbitrarily long times
- Solution: Do a dummy read when you really need a `writel()` to get to your device *now*.
- Important cases that go wrong most often
 - disabling interrupts on the card on `close()`
 - ringbuffer management (NIC's, fibrechannel)



PCI posting continued

```
static void
eeprom_stand_by(struct e100_priv *adapter)
{
    u16 x;

    x = readw(&CSR_EEPROM(adapter));
    x &= ~(EECS | EESK);
    writew(x, &CSR_EEPROM(adapter));

    udelay(EEPROM_STALL_TIME);
    x |= EECS;
    writew(x, &CSR_EEPROM(adapter));

    udelay(EEPROM_STALL_TIME);
}
```

from the Intel e100 driver in kernel 2.5.6



PCI posting continued

```
static void
eeprom_stand_by(struct e100_private *adapter)
{
    u16 x;

    x = readw(&CSR_EEPROM_CONTROL_FIELD(adapter));
    x &= ~(EECS | EESK);
    writew(x, &CSR_EEPROM_CONTROL_FIELD(adapter));
    readw(&(adapter->scb->scb_status)); /* flush command to card */
    udelay(EEPROM_STALL_TIME);
    x |= EECS;
    writew(x, &CSR_EEPROM_CONTROL_FIELD(adapter));
    readw(&(adapter->scb->scb_status)); /* flush command to card */
    udelay(EEPROM_STALL_TIME);
}
```

from the Intel e100 driver in kernel 2.5.20



pci_enable_device

```
mmio_start = pci_resource_start(pdev, 0);
mmio_len = pci_resource_len(pdev, 0);
if (!request_mem_region(mmio_start, mmio_len, "slicoss")) {
    goto err_out_exit_slic_probe;
}
memmapped_ioaddr = (ulong) ioremap_nocache(mmio_start, mmio_len);
...
status = pci_enable_device(pdev);
```

slicoss NIC driver

- pci_enable_device() is the *very* first thing you **MUST** do



Other PCI issues

- assuming ioremap returns a real pointer, eg not readl/writel
- hardcoded PC-isms (port numbers) for, say, a private serial console
- not using PCI DMA API but assuming GFP_KERNEL memory can be dma'd from/to as is
- not registering/requesting resources
- reading PCI resources from PCI config space
- use dma_addr_t, don't try to be smart and do your own type



64 bit

- bitops only work on unsigned long
- only use unsigned long for storing cpu flags
- pci_map_page returns an u64, even on x86



Delays

- The Windows driver model is different with driver threads
- the kernel goal is approx 500usec latency
- avoid `mdelay()` as much as possible



Abstracting delays

```
void msec_delay(int x)
{
    if(in_interrupt()) {
        mdelay(x);
    } else {
        set_current_state(TASK_UNINTERRUPTIBLE);
        schedule_timeout((x * HZ)/1000);
    }
}
```

Intel e1000 driver in 2.5.20



Reading config files

```
set_fs(KERNEL_DS);
if((fd=open(CHANDEV_FILE,O_RDONLY,0))!=-1) {
    curr=0;
    left=statbuf.st_size;
    while((len=read(fd,&buff[curr],left))>0) {
        curr+=len;
        left-=len;
    }
    close(fd);
}
set_fs(USER_DS);
```

[2.4.18] drivers/char/s390/misc.c:chandev_read_conf



Races

- module unload races
- open/close races

```
if(audit_device_open)
    return -EBUSY;
// Ok, we're open.
audit_device_open=1;
// Fetch the task structure of the process that opened the device
auditdaemon_task_struct = current;
```

...

```
if(auditdaemon_task_struct) {
    if(auditdaemon_task_struct->pid && auditdaemon_task_struct->pid
<= 65535
```

...



Other

- Accessing current from IRQ context
- Use of floating point
- Overriding system calls

