

---

Subject: Re: [PATCH] Protect tty drivers list a little  
Posted by [Andrew Morton](#) on Thu, 22 Mar 2007 17:29:05 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On Thu, 22 Mar 2007 14:25:42 +0300 Alexey Dobriyan <[adobriyan@sw.ru](mailto:adobriyan@sw.ru)> wrote:

> Additions and removal from tty\_drivers list were just done as well as  
> iterating on it for /proc/tty/drivers generation.  
>  
> testing: modprobe/rmmod loop of simple module which does nothing but  
> tty\_register\_driver() vs cat /proc/tty/drivers loop  
>  
> BUG: unable to handle kernel paging request at virtual address 6b6b6b6b  
> printing eip:  
> c01cefa7  
> \*pde = 00000000  
> Oops: 0000 [#1]  
> PREEMPT  
> last sysfs file: devices/pci0000:00/0000:00:1d.7/usb5/5-0:1.0/bInterfaceProto col  
> Modules linked in: ohci\_hcd af\_packet e1000 ehci\_hcd uhci\_hcd usbcore xfs  
> CPU: 0  
> EIP: 0060:[<c01cefa7>] Not tainted VLI  
> EFLAGS: 00010297 (2.6.21-rc4-mm1 #4)  
> EIP is at vsnprintf+0x3a4/0x5fc  
> eax: 6b6b6b6b ebx: f6cb50f2 ecx: 6b6b6b6b edx: ffffffe  
> esi: c0354700 edi: f6cb6000 ebp: 6b6b6b6b esp: f31f5e68  
> ds: 007b es: 007b fs: 00d8 gs: 0033 ss: 0068  
> Process cat (pid: 31864, ti=f31f4000 task=c1998030 task.ti=f31f4000)  
> Stack: 00000000 c0103f20 c013003a c0103f20 00000000 f6cb50da 0000000a 00000f0e  
> f6cb50f2 00000010 00000014 ffffffff ffffffff 00000007 c0354753 f6cb50f2  
> f73e39dc f73e39dc 00000001 c0175416 f31f5ed8 f31f5ed4 0ee00000 f32090bc  
> Call Trace:  
> [<c0103f20>] restore\_nocheck+0x12/0x15  
> [<c013003a>] mark\_held\_locks+0x6d/0x86  
> [<c0103f20>] restore\_nocheck+0x12/0x15  
> [<c0175416>] seq\_printf+0x2e/0x52  
> [<c0192895>] show\_tty\_range+0x35/0x1f3  
> [<c0175416>] seq\_printf+0x2e/0x52  
> [<c0192add>] show\_tty\_driver+0x8a/0x1d9  
> [<c01758f6>] seq\_read+0x70/0x2ba  
> [<c0175886>] seq\_read+0x0/0x2ba  
> [<c018d8e6>] proc\_reg\_read+0x63/0x9f  
> [<c015e764>] vfs\_read+0x7d/0xb5  
> [<c018d883>] proc\_reg\_read+0x0/0x9f  
> [<c015eab1>] sys\_read+0x41/0x6a  
> [<c0103e4e>] sysenter\_past\_esp+0x5f/0x99  
> ======  
> Code: 00 8b 4d 04 e9 44 ff ff 8d 4d 04 89 4c 24 50 8b 6d 00 81 fd ff 0f 00 00 b8 a4 c1 35 c0 0f

```
46 e8 8b 54 24 2c 89 e9 89 c8 eb 06 <80> 38 00 74 07 40 4a 83 fa ff 75 f4 29 c8 89 c6 8b 44 24  
28 89  
> EIP: [<c01cefa7>] vsnprintf+0x3a4/0x5fc SS:ESP 0068:f31f5e68  
>
```

nice.

```
> ---  
>  
> drivers/char/tty_io.c | 9 ++++++++-  
> fs/proc/proc_tty.c | 3 +++  
> include/linux/tty_driver.h | 2 ++  
> 3 files changed, 13 insertions(+), 1 deletion(-)  
>  
> --- a/drivers/char/tty_io.c  
> +++ b/drivers/char/tty_io.c  
> @@ -127,6 +127,7 @@ EXPORT_SYMBOL(tty_std_termios);  
>   into this file */  
>  
> LIST_HEAD(tty_drivers); /* linked list of tty drivers */  
> +DEFINE_SPINLOCK(tty_drivers_lock);  
>  
> /* Mutex to protect creating and releasing a tty. This is shared with  
> vt.c for deeply disgusting hack reasons */  
> @@ -1086,13 +1087,16 @@ static struct tty_driver *get_tty_driver  
> {  
>   struct tty_driver *p;  
>  
> + spin_lock(&tty_drivers_lock);  
>   list_for_each_entry(p, &tty_drivers, tty_drivers) {  
>     dev_t base = MKDEV(p->major, p->minor_start);  
>     if (device < base || device >= base + p->num)  
>       continue;  
> + spin_unlock(&tty_drivers_lock);  
>   *index = device - base;  
>   return p;  
> }  
> + spin_unlock(&tty_drivers_lock);  
>   return NULL;  
> }
```

The locking in here is kinda meaningless: we drop the lock and return an unrefcounted something which really should have been covered by that lock. Or refcounted.

The reason is that get\_tty\_driver() and its return value are already covered by tty\_mutex.

So can we use tty\_mutex to fix this race rather than adding a new lock?

---