
Subject: Re: [PATCH] Rewrite proc seq operations via seq_list_xxx ones

Posted by [Mathieu Desnoyers](#) on Wed, 10 Oct 2007 15:10:49 GMT

[View Forum Message](#) <> [Reply to Message](#)

* Pavel Emelyanov (xemul@openvz.org) wrote:

> Mathieu Desnoyers wrote:

> > * Pavel Emelyanov (xemul@openvz.org) wrote:

> >> Some time ago I posted a set of patches that consolidated

> >> the seq files, walking the list_head-s. This set was merged

> >> in the mm tree. /proc/diskstats and /proc/partitions files

> >> were included in this set, but a bit later this part was

> >> dropped because of conflicts with some other changes.

> >>

> >> Here's the fixed version against the latest git block repo.

> >>

> >

> > Hi Pavel,

> >

> > You might want to try implementing this patch on top of the sorted seq

> > list patch I proposed there. It deals with a race between proc file read

> > and list modification between iterations.

>

> AFAIS there's no race - everything is protected with the mutex.

>

> OTOH, If you mean the problem that the list can change while the

> seq file reallocates its buffers, then this problem exists for all

> the kernel and I don't think it's a good idea to stick to the block

> proc files only.

>

The problem is that the list can change between two consecutive reads.

And yes, all proc files using lists are affected, but changing every
proc file users is a quite large task.

A problematic sequence would be:

1st procfs read()

*_start()

- mutex down

- seq list start

*_next()

- seq list next

*_stop()

- mutex up

end of 1st procfs read()

mutex down

-> Removing a list entry

mutex up

2nd procfs read()

*_start()

- mutex down

... we now skip a list entry depending on the relative position of our iterator and the deletion.

The mutex is taken/released at each and every read, therefore not protecting against modifications across reads.

Mathieu

> > <http://lkml.org/lkml/2007/9/6/175>

> >

> > See this patch for a user implementation example and test code

> > (/proc/modules) :

> >

> > <http://lkml.org/lkml/2007/9/6/176>

> >

> > Mathieu

> >

> >> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

> >>

> >> ---

> >>

> >> diff --git a/block/genhd.c b/block/genhd.c

> >> index 3af1e7a..27f7061 100644

> >> --- a/block/genhd.c

> >> +++ b/block/genhd.c

> >> @@ -264,39 +264,34 @@ void __init printk_all_partitions(void)

> >>

> >> #ifdef CONFIG_PROC_FS

> >> /* iterator */

> >> -static void *part_start(struct seq_file *part, loff_t *pos)

> >> +static void *disk_start(struct seq_file *part, loff_t *pos)

> >> {

> >> - struct list_head *p;

> >> - loff_t l = *pos;

> >> -

> >> mutex_lock(&block_subsys_lock);

> >> - list_for_each(p, &block_subsys.list)

> >> - if (!--)

> >> - return list_entry(p, struct gendisk, kobj.entry);

> >> - return NULL;

```

>>> + return seq_list_start_head(&block_subsys.list, *pos);
>>> }
>>>
>>> -static void *part_next(struct seq_file *part, void *v, loff_t *pos)
>>> +static void *disk_next(struct seq_file *part, void *v, loff_t *pos)
>>> {
>>> - struct list_head *p = ((struct gendisk *)v)->kobj.entry.next;
>>> - ++*pos;
>>> - return p==&block_subsys.list ? NULL :
>>> - list_entry(p, struct gendisk, kobj.entry);
>>> + return seq_list_next(v, &block_subsys.list, pos);
>>> }
>>>
>>> -static void part_stop(struct seq_file *part, void *v)
>>> +static void disk_stop(struct seq_file *part, void *v)
>>> {
>>> mutex_unlock(&block_subsys_lock);
>>> }
>>>
>>> static int show_partition(struct seq_file *part, void *v)
>>> {
>>> - struct gendisk *sgp = v;
>>> + struct gendisk *sgp;
>>> int n;
>>> char buf[BDEVNAME_SIZE];
>>>
>>> - if (&sgp->kobj.entry == block_subsys.list.next)
>>> + if (v == &block_subsys.list) {
>>> seq_puts(part, "major minor #blocks name\n\n");
>>> + return 0;
>>> + }
>>> +
>>> + sgp = list_entry(v, struct gendisk, kobj.entry);
>>>
>>> /* Don't show non-partitionable removeable devices or empty devices */
>>> if (!get_capacity(sgp) ||
>>> @@ -325,9 +320,9 @@ static int show_partition(struct seq_fil
>>> }
>>>
>>> struct seq_operations partitions_op = {
>>> - .start = part_start,
>>> - .next = part_next,
>>> - .stop = part_stop,
>>> + .start = disk_start,
>>> + .next = disk_next,
>>> + .stop = disk_stop,
>>> .show = show_partition
>>> };

```

```

>>> #endif
>>> @@ -616,44 +611,22 @@ decl_subsys(block, &ktype_block, &block_
>>> */
>>>
>>> /* iterator */
>>> -static void *diskstats_start(struct seq_file *part, loff_t *pos)
>>> -{
>>> - loff_t k = *pos;
>>> - struct list_head *p;
>>> -
>>> - mutex_lock(&block_subsys_lock);
>>> - list_for_each(p, &block_subsys.list)
>>> - if (!k--)
>>> - return list_entry(p, struct gendisk, kobj.entry);
>>> - return NULL;
>>> -}
>>> -
>>> -static void *diskstats_next(struct seq_file *part, void *v, loff_t *pos)
>>> -{
>>> - struct list_head *p = ((struct gendisk *)v)->kobj.entry.next;
>>> - ++*pos;
>>> - return p==&block_subsys.list ? NULL :
>>> - list_entry(p, struct gendisk, kobj.entry);
>>> -}
>>> -
>>> -static void diskstats_stop(struct seq_file *part, void *v)
>>> -{
>>> - mutex_unlock(&block_subsys_lock);
>>> -}
>>> -
>>> static int diskstats_show(struct seq_file *s, void *v)
>>> {
>>> - struct gendisk *gp = v;
>>> + struct gendisk *gp;
>>> char buf[BDEVNAME_SIZE];
>>> int n = 0;
>>>
>>> + if (v == &block_subsys.list)
>>> /*
>>> - if (&sgp->kobj.entry == block_subsys.kset.list.next)
>>> seq_puts(s, "major minor name"
>>> "    rio rmerge rsect ruse wio wmerge "
>>> "wsect wuse running use aveq"
>>> "\n\n");
>>> */
>>> + return 0;
>>> +
>>> + gp = list_entry(v, struct gendisk, kobj.entry);

```

```

>>>
>>> preempt_disable();
>>> disk_round_stats(gp);
>>> @@ -686,9 +659,9 @@ static int diskstats_show(struct seq_fil
>>> }
>>>
>>> struct seq_operations diskstats_op = {
>>> - .start = diskstats_start,
>>> - .next = diskstats_next,
>>> - .stop = diskstats_stop,
>>> + .start = disk_start,
>>> + .next = disk_next,
>>> + .stop = disk_stop,
>>> .show = diskstats_show
>>> };
>>>
>>> -
>>> To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
>>> the body of a message to majordomo@vger.kernel.org
>>> More majordomo info at http://vger.kernel.org/majordomo-info.html
>>> Please read the FAQ at http://www.tux.org/lkml/
>>>
>>
>

```

--

Mathieu Desnoyers
 Computer Engineering Ph.D. Student, Ecole Polytechnique de Montreal
 OpenPGP key fingerprint: 8CD5 52C3 8E3C 4140 715F BA06 3F25 A8FE 3BAE 9A68
