Subject: [RFC PATCH 4/6] IPC/sem: next operations for /proc/pid/semundo
Posted by Nadia Derbey on Wed, 25 Jun 2008 13:49:14 GMT
View Forum Message <> Reply to Message

PATCH [04/06]

This patch introduces the .next seq operation for /proc/pid/semundo.

What should be mentioned here is that the undo_list lock is released between
between each iteration.
Doing this, we only guarantee to access some valid data during the .show,
not to have a full coherent view of the whole list. But, oth, this reduces the
the performance impact on the access to the undo_list.

Signed-off-by: Pierre Peiffer <pierre.peiffer@bull.net>
Signed-off-by: Nadia Derbey <Nadia.Derbey@bull.net>

---
 ipc/sem.c | 23 ++++++++++++++++++++++-
 1 file changed, 22 insertions(+), 1 deletion(-)

Index: linux-2.6.26-rc5-mm3/ipc/sem.c
===
--- linux-2.6.26-rc5-mm3.orig/ipc/sem.c 2008-06-24 12:32:36.000000000 +0200
+++ linux-2.6.26-rc5-mm3/ipc/sem.c 2008-06-24 12:54:40.000000000 +0200
@@ -1440,7 +1440,28 @@ static void *semundo_start(struct seq_fi

 static void *semundo_next(struct seq_file *m, void *v, loff_t *ppos)
 {
- return NULL;
+ struct sem_undo *undo = v;
+ struct undo_list_data *data = m->private;
+ struct sem_undo_list *ulp = data->undo_list;
+
+ /*
+ * No need to protect against ulp being NULL, if we are here,
+ * it can't be NULL.
+ */
+ spin_lock(&ulp->lock);
+
+ do {
+ undo = list_entry(rcu_dereference(undo->list_proc.next),
+   struct sem_undo, list_proc);
+
+ } while (&undo->list_proc != &ulp->list_proc && undo->semid == -1);
+
+ ++*ppos;
+ spin_unlock(&ulp->lock);

```
+
+ if (&undo->list_proc == &ulp->list_proc)
+  return NULL;
+ return undo;
 }

 static void semundo_stop(struct seq_file *m, void *v)

--
```

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [RFC PATCH 4/6] IPC/sem: next operations for /proc/pid/semundo
Posted by serue on Wed, 25 Jun 2008 20:57:19 GMT
View Forum Message <> Reply to Message

Quoting Nadia.Derbey@bull.net (Nadia.Derbey@bull.net):
> PATCH [04/06]
>
> This patch introduces the .next seq operation for /proc/pid/semundo.
>
> What should be mentioned here is that the undo_list lock is released between
> between each iteration.
> Doing this, we only guarantee to access some valid data during the .show,

Ok so you count on an item sticking around for the duration of the
rcu_read_cycle(). exit_sem() is therefore not an issue. The other
possible racer is freeary() as called from IPC_RMID, but while that
could remove this entry from the undo_list->list_proc, it will wait
an rcu cycle before it actually frees it.

Am I reading that all right? If so, then:

> not to have a full coherent view of the whole list. But, oth, this reduces the
> the performance impact on the access to the undo_list.
>
> Signed-off-by: Pierre Peiffer <pierre.peiffer@bull.net>
> Signed-off-by: Nadia Derbey <Nadia.Derbey@bull.net>

Acked-by: Serge Hallyn <serue@us.ibm.com>

>
> ---
> ipc/sem.c | 23 ++++++++++++++++++++-
> 1 file changed, 22 insertions(+), 1 deletion(-)

```
>
> Index: linux-2.6.26-rc5-mm3/ipc/sem.c
> ===================================================================
> --- linux-2.6.26-rc5-mm3.orig/ipc/sem.c 2008-06-24 12:32:36.000000000 +0200
> +++ linux-2.6.26-rc5-mm3/ipc/sem.c 2008-06-24 12:54:40.000000000 +0200
> @@ -1440,7 +1440,28 @@ static void *semundo_start(struct seq_fi
>
>  static void *semundo_next(struct seq_file *m, void *v, loff_t *ppos)
>  {
> - return NULL;
> + struct sem_undo *undo = v;
> + struct undo_list_data *data = m->private;
> + struct sem_undo_list *ulp = data->undo_list;
> +
> + /*
> +  * No need to protect against ulp being NULL, if we are here,
> +  * it can't be NULL.
> +  */
> + spin_lock(&ulp->lock);
> +
> + do {
> +  undo = list_entry(rcu_dereference(undo->list_proc.next),
> +    struct sem_undo, list_proc);
> +
> + } while (&undo->list_proc != &ulp->list_proc && undo->semid == -1);
> +
> + ++*ppos;
> + spin_unlock(&ulp->lock);
> +
> + if (&undo->list_proc == &ulp->list_proc)
> +  return NULL;
> + return undo;
>  }
>
>  static void semundo_stop(struct seq_file *m, void *v)
>
> --

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers
```

Subject: Re: [RFC PATCH 4/6] IPC/sem: next operations for /proc/pid/semundo
Posted by Nadia Derbey on Thu, 26 Jun 2008 05:35:27 GMT
View Forum Message <> Reply to Message

Serge E. Hallyn wrote:

> Quoting Nadia.Derbey@bull.net (Nadia.Derbey@bull.net):
>
>>PATCH [04/06]
>>
>>This patch introduces the .next seq operation for /proc/pid/semundo.
>>
>>What should be mentioned here is that the undo_list lock is released between
>>between each iteration.
>>Doing this, we only guarantee to access some valid data during the .show,
>
>
> Ok so you count on an item sticking around for the duration of the
> rcu_read_cycle(). exit_sem() is therefore not an issue. The other
> possible racer is freeary() as called from IPC_RMID, but while that
> could remove this entry from the undo_list->list_proc, it will wait
> an rcu cycle before it actually frees it.

Yes, the sem_undo structure is freed in an rcu callback in freeary() too.

>
> Am I reading that all right? If so, then:
>
>
>>not to have a full coherent view of the whole list. But, oth, this reduces the
>>the performance impact on the access to the undo_list.
>>
>>Signed-off-by: Pierre Peiffer <pierre.peiffer@bull.net>
>>Signed-off-by: Nadia Derbey <Nadia.Derbey@bull.net>
>
>
> Acked-by: Serge Hallyn <serue@us.ibm.com>
>
>>---
>> ipc/sem.c | 23 ++++++++++++++++++++++-
>> 1 file changed, 22 insertions(+), 1 deletion(-)
>>
>>Index: linux-2.6.26-rc5-mm3/ipc/sem.c
>>===
>>--- linux-2.6.26-rc5-mm3.orig/ipc/sem.c 2008-06-24 12:32:36.000000000 +0200
>>+++ linux-2.6.26-rc5-mm3/ipc/sem.c 2008-06-24 12:54:40.000000000 +0200
>>@@ -1440,7 +1440,28 @@ static void *semundo_start(struct seq_fi
>>
>> static void *semundo_next(struct seq_file *m, void *v, loff_t *ppos)
>> {
>>- return NULL;
>>+ struct sem_undo *undo = v;
>>+ struct undo_list_data *data = m->private;
>>+ struct sem_undo_list *ulp = data->undo_list;

```
>>+
>>+ /*
>>+  * No need to protect against ulp being NULL, if we are here,
>>+  * it can't be NULL.
>>+  */
>>+ spin_lock(&ulp->lock);
>>+
>>+ do {
>>+  undo = list_entry(rcu_dereference(undo->list_proc.next),
>>+    struct sem_undo, list_proc);
>>+
>>+ } while (&undo->list_proc != &ulp->list_proc && undo->semid == -1);
>>+
>>+ ++*ppos;
>>+ spin_unlock(&ulp->lock);
>>+
>>+ if (&undo->list_proc == &ulp->list_proc)
>>+  return NULL;
>>+ return undo;
>> }
>>
>> static void semundo_stop(struct seq_file *m, void *v)
>>
>>--
>
>
>
```


--
==========
Name..... Nadia DERBEY
Organization.. BULL/DT/OSwR&D/Linux

Email..... mailto:Nadia.Derbey@bull.net
Address..... BULL, B.P. 208, 38432 Echirolles Cedex, France
Tel..... (33) 76 29 77 62 [Internal Bull: (229) 77 62]
Telex,Fax..... 980648 F - (33) 76 29 76 00
Internal Bull. Mail: FREC-B1208
==========