

---

Subject: [RFC][PATCH 3/6] pid namespace : use struct pid\_nr  
Posted by [Sukadev Bhattiprolu](#) on Sat, 10 Mar 2007 03:59:04 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

From: Cedric Le Goater <clg@fr.ibm.com>  
Subject: [RFC][PATCH 3/6] pid namespace : use struct pid\_nr

Allocate and attach a struct pid nr to the struct pid. When freeing the pid, free the attached struct pid nrs.

Changelog:

- [Serge Hallyn's comment]: Add comments on what pid->lock protects and that pid->nr will eventually go away.

Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>  
Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>

---  
include/linux/pid.h | 3 +++  
kernel/pid.c | 44 ++++++++++++++++++++++++++++++++++++++-----  
2 files changed, 38 insertions(+), 9 deletions(-)

Index: lx26-20-mm2b/include/linux/pid.h

```
=====
--- lx26-20-mm2b.orig/include/linux/pid.h 2007-03-09 15:29:21.000000000 -0800
+++ lx26-20-mm2b/include/linux/pid.h 2007-03-09 15:49:29.000000000 -0800
@@ -66,6 +66,9 @@ struct pid
 /* lists of tasks that use this pid */
 struct hlist_head tasks[PIDTYPE_MAX];
 struct rcu_head rcu;
+ struct hlist_head pid_nrs;
+ spinlock_t lock;
+ /* protects pid_nrs list */
};
```

extern struct pid init\_struct\_pid;

Index: lx26-20-mm2b/kernel/pid.c

```
=====
--- lx26-20-mm2b.orig/kernel/pid.c 2007-03-09 15:29:21.000000000 -0800
+++ lx26-20-mm2b/kernel/pid.c 2007-03-09 15:29:23.000000000 -0800
@@ -180,8 +180,19 @@ fastcall void put_pid(struct pid *pid)
 if (!pid)
 return;
 if ((atomic_read(&pid->count) == 1) ||
- atomic_dec_and_test(&pid->count))
+ atomic_dec_and_test(&pid->count)) {
+ struct pid_nr* pid_nr;
+ struct hlist_node *pos, *next;
```

```

+
+ /*
+  * rcu is not needed anymore
+  */
+ hlist_for_each_entry_safe(pid_nr, pos, next, &pid->pid_nrs, node) {
+  hlist_del_init(&pid_nr->node);
+  free_pid_nr(pid_nr);
+ }
+ kmem_cache_free(pid_cachep, pid);
+ }
+ }
EXPORT_SYMBOL_GPL(put_pid);

```

```

@@ -202,6 +213,9 @@ void free_pid_nr(struct pid_nr *pid_nr)

```

```

fastcall void free_pid(struct pid *pid)
{
+ struct pid_nr* pid_nr;
+ struct hlist_node *pos;
+
+ /* We can be called with write_lock_irq(&tasklist_lock) held */
+ unsigned long flags;

```

```

@@ -209,7 +223,8 @@ fastcall void free_pid(struct pid *pid)
+ hlist_del_rcu(&pid->pid_chain);
+ spin_unlock_irqrestore(&pidmap_lock, flags);

```

```

- free_pidmap(&init_pid_ns, pid->nr);
+ hlist_for_each_entry(pid_nr, pos, &pid->pid_nrs, node)
+ free_pidmap(pid_nr->pid_ns, pid_nr->nr);
+ call_rcu(&pid->rcu, delayed_put_pid);
+ }

```

```

@@ -290,31 +305,42 @@ struct pid *alloc_pid(void)

```

```

+ struct pid *pid;
+ enum pid_type type;
+ int nr = -1;
+ struct pid_nr *pid_nr;

+ pid = kmem_cache_alloc(pid_cachep, GFP_KERNEL);
+ if (!pid)
+ goto out;
+ return NULL;

```

```

+ nr = alloc_pidmap(task_pid_ns(current));
+ if (nr < 0)
+ goto out_free;
+ goto out_free_pid;

```

```

+
+ pid_nr = alloc_pid_nr(task_pid_ns(current));
+ if (!pid_nr)
+ goto out_free_pidmap;

    atomic_set(&pid->count, 1);
- pid->nr = nr;
+ pid->nr = pid_nr->nr = nr; /* pid->nr to be removed soon */
    for (type = 0; type < PIDTYPE_MAX; ++type)
        INIT_HLIST_HEAD(&pid->tasks[type]);

+ spin_lock_init(&pid->lock);
+
+ INIT_HLIST_HEAD(&pid->pid_nrs);
+ hlist_add_head_rcu(&pid_nr->node, &pid->pid_nrs);
+
    spin_lock_irq(&pidmap_lock);
    hlist_add_head_rcu(&pid->pid_chain, &pid_hash[pid_hashfn(pid->nr)]);
    spin_unlock_irq(&pidmap_lock);

-out:
    return pid;

-out_free:
+out_free_pidmap:
+ free_pidmap(task_pid_ns(current), nr);
+
+out_free_pid:
    kmem_cache_free(pid_cachep, pid);
- pid = NULL;
- goto out;
+ return NULL;
}

struct pid * fastcall find_pid(int nr)

```

---

Containers mailing list  
Containers@lists.osdl.org  
<https://lists.osdl.org/mailman/listinfo/containers>

---



---

Subject: Re: [RFC][PATCH 3/6] pid namespace : use struct pid\_nr  
Posted by [ebiederm](#) on Sun, 11 Mar 2007 11:43:20 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

sukadev@us.ibm.com writes:

> From: Cedric Le Goater <clg@fr.ibm.com>

```

> Subject: [RFC][PATCH 3/6] pid namespace : use struct pid_nr
>
> Allocate and attach a struct pid nr to the struct pid. When freeing the
> pid, free the attached struct pid nrs.
>
> Changelog:
> - [Serge Hallyn's comment]: Add comments on what pid->lock protects
>   and that pid->nr will eventually go away.
>
> Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
> Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
>
> ---
> include/linux/pid.h | 3 +++
> kernel/pid.c        | 44 ++++++++++++++++++++++++++++++++++++++-----
> 2 files changed, 38 insertions(+), 9 deletions(-)
>
> Index: lx26-20-mm2b/include/linux/pid.h
> =====
> --- lx26-20-mm2b.orig/include/linux/pid.h 2007-03-09 15:29:21.000000000 -0800
> +++ lx26-20-mm2b/include/linux/pid.h 2007-03-09 15:49:29.000000000 -0800
> @@ -66,6 +66,9 @@ struct pid
>  /* lists of tasks that use this pid */
>  struct hlist_head tasks[PIDTYPE_MAX];
>  struct rcu_head rcu;
> + struct hlist_head pid_nrs;
> + spinlock_t lock;

```

Ah so this is where the lock appears. Definitely not git-bisect safe.

```

> + /* protects pid_nrs list */
> };
>
> extern struct pid init_struct_pid;
> Index: lx26-20-mm2b/kernel/pid.c
> =====
> --- lx26-20-mm2b.orig/kernel/pid.c 2007-03-09 15:29:21.000000000 -0800
> +++ lx26-20-mm2b/kernel/pid.c 2007-03-09 15:29:23.000000000 -0800
> @@ -180,8 +180,19 @@ fastcall void put_pid(struct pid *pid)
>  if (!pid)
>    return;
>  if ((atomic_read(&pid->count) == 1) ||
> -   atomic_dec_and_test(&pid->count))
> +   atomic_dec_and_test(&pid->count)) {
> + struct pid_nr* pid_nr;
> + struct hlist_node *pos, *next;
> +
> + /*

```

```
> + * rcu is not needed anymore
> + */
```

rcu should never be needed...

We should be able to get away with a definition that is immutable for the lifetime of a struct pid.

```
> + hlist_for_each_entry_safe(pid_nr, pos, next, &pid->pid_nrs, node) {
> +   hlist_del_init(&pid_nr->node);
> +   free_pid_nr(pid_nr);
> + }
>   kmem_cache_free(pid_cachep, pid);
> + }
> }
> EXPORT_SYMBOL_GPL(put_pid);
>
> @@ -202,6 +213,9 @@ void free_pid_nr(struct pid_nr *pid_nr)
>
> fastcall void free_pid(struct pid *pid)
> {
> + struct pid_nr* pid_nr;
> + struct hlist_node *pos;
> +
> /* We can be called with write_lock_irq(&tasklist_lock) held */
> unsigned long flags;
>
> @@ -209,7 +223,8 @@ fastcall void free_pid(struct pid *pid)
>   hlist_del_rcu(&pid->pid_chain);
>   spin_unlock_irqrestore(&pidmap_lock, flags);
>
> - free_pidmap(&init_pid_ns, pid->nr);
> + hlist_for_each_entry(pid_nr, pos, &pid->pid_nrs, node)
> +   free_pidmap(pid_nr->pid_ns, pid_nr->nr);
>   call_rcu(&pid->rcu, delayed_put_pid);
> }
>
> @@ -290,31 +305,42 @@ struct pid *alloc_pid(void)
>   struct pid *pid;
>   enum pid_type type;
>   int nr = -1;
> + struct pid_nr *pid_nr;
>
>   pid = kmem_cache_alloc(pid_cachep, GFP_KERNEL);
>   if (!pid)
> -   goto out;
> +   return NULL;
>
>   nr = alloc_pidmap(task_pid_ns(current));
```

```

> if (nr < 0)
> - goto out_free;
> + goto out_free_pid;
> +
> + pid_nr = alloc_pid_nr(task_pid_ns(current));
> + if (!pid_nr)
> + goto out_free_pidmap;
>

```

We should allocate one pid number for each parent pid namespace, not just one. You don't even seem to be keeping track of the parent pid namespaces. We can probably get there by walking the parent processes but it would be easier if we had a pointer in the pid\_namespace...

Eric

---

Containers mailing list  
Containers@lists.osdl.org  
<https://lists.osdl.org/mailman/listinfo/containers>

---



---

Subject: Re: [RFC][PATCH 3/6] pid namespace : use struct pid\_nr  
Posted by [Sukadev Bhattiprolu](#) on Tue, 13 Mar 2007 04:17:15 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Eric W. Biederman [ebiederm@xmission.com] wrote:

| sukadev@us.ibm.com writes:

```

|
| > From: Cedric Le Goater <clg@fr.ibm.com>
| > Subject: [RFC][PATCH 3/6] pid namespace : use struct pid_nr
| >
| > Allocate and attach a struct pid nr to the struct pid. When freeing the
| > pid, free the attached struct pid nrs.
| >
| > Changelog:
| > - [Serge Hallyn's comment]: Add comments on what pid->lock protects
| >   and that pid->nr will eventually go away.
| >
| > Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
| > Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
| >
| > ---
| > include/linux/pid.h | 3 +++
| > kernel/pid.c        | 44 ++++++++++++++++++++++++++++++++++++++-----
| > 2 files changed, 38 insertions(+), 9 deletions(-)
| >

```

```
| > Index: lx26-20-mm2b/include/linux/pid.h
| > =====
| > --- lx26-20-mm2b.orig/include/linux/pid.h 2007-03-09 15:29:21.000000000 -0800
| > +++ lx26-20-mm2b/include/linux/pid.h 2007-03-09 15:49:29.000000000 -0800
| > @@ -66,6 +66,9 @@ struct pid
| > /* lists of tasks that use this pid */
| > struct hlist_head tasks[PIDTYPE_MAX];
| > struct rcu_head rcu;
| > + struct hlist_head pid_nrs;
| > + spinlock_t lock;
|
| Ah so this is where the lock appears. Definitely not git-bisect safe.
```

Yes. I moved some patches around before posting. Will fix it for next pass.

```
|
| > + /* protects pid_nrs list */
| > };
| >
| > extern struct pid init_struct_pid;
| > Index: lx26-20-mm2b/kernel/pid.c
| > =====
| > --- lx26-20-mm2b.orig/kernel/pid.c 2007-03-09 15:29:21.000000000 -0800
| > +++ lx26-20-mm2b/kernel/pid.c 2007-03-09 15:29:23.000000000 -0800
| > @@ -180,8 +180,19 @@ fastcall void put_pid(struct pid *pid)
| > if (!pid)
| > return;
| > if ((atomic_read(&pid->count) == 1) ||
| > - atomic_dec_and_test(&pid->count))
| > + atomic_dec_and_test(&pid->count)) {
| > + struct pid_nr* pid_nr;
| > + struct hlist_node *pos, *next;
| > +
| > + /*
| > + * rcu is not needed anymore
| > + */
|
| rcu should never be needed...
| We should be able to get away with a definition that is immutable for the
| lifetime of a struct pid.
```

Yes.

```
|
| > + hlist_for_each_entry_safe(pid_nr, pos, next, &pid->pid_nrs, node) {
| > + hlist_del_init(&pid_nr->node);
| > + free_pid_nr(pid_nr);
```

```

|> + }
|> kmem_cache_free(pid_cachep, pid);
|> + }
|> }
|> EXPORT_SYMBOL_GPL(put_pid);
|>
|> @@ -202,6 +213,9 @@ void free_pid_nr(struct pid_nr *pid_nr)
|>
|> fastcall void free_pid(struct pid *pid)
|> {
|> + struct pid_nr* pid_nr;
|> + struct hlist_node *pos;
|> +
|> /* We can be called with write_lock_irq(&tasklist_lock) held */
|> unsigned long flags;
|>
|> @@ -209,7 +223,8 @@ fastcall void free_pid(struct pid *pid)
|> hlist_del_rcu(&pid->pid_chain);
|> spin_unlock_irqrestore(&pidmap_lock, flags);
|>
|> - free_pidmap(&init_pid_ns, pid->nr);
|> + hlist_for_each_entry(pid_nr, pos, &pid->pid_nrs, node)
|> + free_pidmap(pid_nr->pid_ns, pid_nr->nr);
|> call_rcu(&pid->rcu, delayed_put_pid);
|> }
|>
|> @@ -290,31 +305,42 @@ struct pid *alloc_pid(void)
|> struct pid *pid;
|> enum pid_type type;
|> int nr = -1;
|> + struct pid_nr *pid_nr;
|>
|> pid = kmem_cache_alloc(pid_cachep, GFP_KERNEL);
|> if (!pid)
|> - goto out;
|> + return NULL;
|>
|> nr = alloc_pidmap(task_pid_ns(current));
|> if (nr < 0)
|> - goto out_free;
|> + goto out_free_pid;
|> +
|> + pid_nr = alloc_pid_nr(task_pid_ns(current));
|> + if (!pid_nr)
|> + goto out_free_pidmap;
|>
|
| We should allocate one pid number for each parent pid namespace,

```



| not just one.

Yes. I will do that for the next pass.

| You don't even seem to be keeping track of the parent  
| pid namespaces. We can probably get there by walking the parent  
| processes but it would be easier if we had a pointer in the  
| pid\_namespace...

I thought about keeping track of parent pid namespace, but did  
find the need for it yet. When do we expect to walk parent/  
ancestor pid namespaces ?

When cloning we can walk the parent pid->pid\_nrs list and duplicate  
them for the child struct pid. If CLONE\_NEWPID is set, then we would  
allocate one more, just for the child. That should give us all our  
ancestor pid namespaces. no ?

|  
|  
| Eric

---

Containers mailing list  
Containers@lists.osdl.org  
<https://lists.osdl.org/mailman/listinfo/containers>

---

---

Subject: Re: [RFC][PATCH 3/6] pid namespace : use struct pid\_nr  
Posted by [Cedric Le Goater](#) on Tue, 13 Mar 2007 08:02:47 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

```
>> Index: lx26-20-mm2b/kernel/pid.c
>> =====
>> --- lx26-20-mm2b.orig/kernel/pid.c 2007-03-09 15:29:21.000000000 -0800
>> +++ lx26-20-mm2b/kernel/pid.c 2007-03-09 15:29:23.000000000 -0800
>> @@ -180,8 +180,19 @@ fastcall void put_pid(struct pid *pid)
>>  if (!pid)
>>    return;
>>  if ((atomic_read(&pid->count) == 1) ||
>>      atomic_dec_and_test(&pid->count))
>>  +  atomic_dec_and_test(&pid->count)) {
>>  +  struct pid_nr* pid_nr;
>>  +  struct hlist_node *pos, *next;
>>  +
>>  +  /*
>>  +   * rcu is not needed anymore
>>  +   */
>>
```

> rcu should never be needed...  
> We should be able to get away with a definition that is immutable for the  
> lifetime of a struct pid.

but struct pid requires to be rcu safe and as the new struct pid\_nr is a member of struct pid, it seems that the same rule should apply. nop ?

Cheers,

C.

---

Containers mailing list  
Containers@lists.osdl.org  
<https://lists.osdl.org/mailman/listinfo/containers>

---

---

Subject: Re: [RFC][PATCH 3/6] pid namespace : use struct pid\_nr  
Posted by [ebiederm](#) on Tue, 13 Mar 2007 08:22:23 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

sukadev@us.ibm.com writes:

> I thought about keeping track of parent pid namespace, but did  
> find the need for it yet. When do we expect to walk parent/  
> ancestor pid namespaces ?  
>  
> When cloning we can walk the parent pid->pid\_nrs list and duplicate  
> them for the child struct pid. If CLONE\_NEWPID is set, then we would  
> allocate one more, just for the child. That should give us all our  
> ancestor pid namespaces. no ?

Sound sane to me, at least as long as the entries are well ordered on that list.

You have missed so many little cases that it is hard for me to tell looking at various patches if you have covered various bits correctly or not.

Going down (if not up) may be one of those things that comes up in kill\_something\_info when we send signals to the list of all processes, in a pid namespace.

By the time we have covered all of the missing little bits it wouldn't surprise me if the choice of data structures changed noticeably, and we need to get all of the bits in the kernel core before we can seriously think about using this code anywhere.

Eric

---

Subject: Re: [RFC][PATCH 3/6] pid namespace : use struct pid\_nr  
Posted by [ebiederm](#) on Tue, 13 Mar 2007 08:52:58 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Cedric Le Goater <[clg@fr.ibm.com](mailto:clg@fr.ibm.com)> writes:

```
>>> Index: lx26-20-mm2b/kernel/pid.c
>>> =====
>>> --- lx26-20-mm2b.orig/kernel/pid.c 2007-03-09 15:29:21.000000000 -0800
>>> +++ lx26-20-mm2b/kernel/pid.c 2007-03-09 15:29:23.000000000 -0800
>>> @@ -180,8 +180,19 @@ fastcall void put_pid(struct pid *pid)
>>>  if (!pid)
>>>    return;
>>>  if ((atomic_read(&pid->count) == 1) ||
>>> -   atomic_dec_and_test(&pid->count))
>>> +   atomic_dec_and_test(&pid->count)) {
>>> +   struct pid_nr* pid_nr;
>>> +   struct hlist_node *pos, *next;
>>> +
>>> +   /*
>>> +    * rcu is not needed anymore
>>> +    */
>>>
>> rcu should never be needed...
>> We should be able to get away with a definition that is immutable for the
>> lifetime of a struct pid.
>
> but struct pid requires to be rcu safe and as the new struct pid_nr is a
> member of struct pid, it seems that the same rule should apply. nop ?
```

I was implying a strong property. A list that is created before we insert into the hash table and is destroyed after the last reference from the hash table is going.

So in that case we would have a property stronger than rcu.

The restrict would be that the list would be completely immutable, for the lifetime of the struct pid.

We may need to modify the list and do the whole rcu thing but I would prefer we not go in with that assumption until we know we need to. Modifying the mapping of struct pid into various pid namespaces adds a complexity to

the user space interface that I would like to avoid.

The only practical use I can currently think of for modifying the struct pid list is when a namespace exists to cleanup our memory of what was in that namespace because pid\_nr will no longer care.

Eric

---

Containers mailing list

Containers@lists.osdl.org

<https://lists.osdl.org/mailman/listinfo/containers>

---