
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

