Subject: release_task(), procfs dependency
Posted by Sukadev Bhattiprolu on Tue, 05 Jun 2007 04:51:11 GMT
View Forum Message <> Reply to Message

Like we discussed earlier and Pavel/others had pointed out,
proc_flush_task() in its current place in release_task() is
useless with the new pid namespace code, because task_pid()
for the task is already NULL before the call to proc_flush_task().

So as a simple change I tried to move proc_flush_task() up (see
below for the patch).

This seems to fix the leak of 'struct pids' we were running into,
but there is a silly circular dependency.

Lets say I execute "pidns_exec ./test1.sh" where test1.sh is:

```
#!/bin/sh

mount -t proc lxcproc /proc
./pidns_exec -F 10 /bin/true
ps -ef
umount /proc
```

When the above 'umount /proc' process exits, /proc is no longer
mounted in the child namespace. And when release_task() tries to
flush the 'umount' task from the child /proc, the proc_mnt->mnt_root
is NULL (its just been unmounted).

A similar problem occurs when test1.sh itself is exiting (both the
'umount' and 'test1.sh' processes are primarily in child namespace).

Not sure yet how to resolve this. Any ideas ?

Suka

```
Index: lx26-21-mm2/kernel/exit.c
===================================================================
--- lx26-21-mm2.orig/kernel/exit.c      2007-06-04 20:04:35.000000000 -0700
+++ lx26-21-mm2/kernel/exit.c   2007-06-04 20:21:35.000000000 -0700
@@ -154,7 +154,9 @@ void release_task(struct task_struct * p
 {
        struct task_struct *leader;
        int zap_leader;
+
 repeat:
+       proc_flush_task(p);
```

```
        atomic_dec(&p->user->processes);
        write_lock_irq(&tasklist_lock);
        ptrace_unlink(p);
@@ -184,7 +186,6 @@ repeat:

        sched_exit(p);
        write_unlock_irq(&tasklist_lock);
-       proc_flush_task(p);
        release_thread(p);
        call_rcu(&p->rcu, delayed_put_task_struct);
```

_____

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