Subject: Re: [PATCH 14/15] Destroy pid namespace on init's death
Posted by Oleg Nesterov on Thu, 02 Aug 2007 15:39:34 GMT
View Forum Message <> Reply to Message

On 08/02, sukadev@us.ibm.com wrote:
>
> Oleg Nesterov [oleg@tv-sign.ru] wrote:
> | On 07/31, sukadev@us.ibm.com wrote:
> | >
> | > Oleg Nesterov [oleg@tv-sign.ru] wrote:
> | > | >
> | > | > @@ -925,9 +926,10 @@ fastcall NORET_TYPE void do_exit(long co
> | > | > if (unlikely(!tsk->pid))
> | > | > panic("Attempted to kill the idle task!");
> | > | > if (unlikely(tsk == task_child_reaper(tsk))) {
> | > | > - if (task_active_pid_ns(tsk) != &init_pid_ns)
> | > | > - task_active_pid_ns(tsk)->child_reaper =
> | > | > - init_pid_ns.child_reaper;
> | > | > + if (pid_ns != &init_pid_ns) {
> | > | > + zap_pid_ns_processes(pid_ns);
> | > | > + pid_ns->child_reaper = init_pid_ns.child_reaper;

OOPS. I didn't notice this before, but this is not right too (regardless
of multi-threaded init problems).

We should not "reset" ->child_reaper here, we may have exiting tasks
which will re-parent their ->children to global init.

No, we are still /sbin/init of this namespace even if we are exiting,
->child_reaper should point to us, at least until zap_pid_ns_processes()
completes.

> | > Our current definition of is_container_init() and task_child_reaper()
> | > refer only to the main-thread of the container-init (since they check
> | > for pid_t == 1)
> | >
> | Yes.
> >
> This means that we cannot have a check like "tsk == task_child_reaper(tsk)"
> to properly detect the child reaper process right ?

Yes, we should use "tsk->group_leader == task_child_reaper(tsk)"

> Its basically a very dumb question - How do we detect a container_init()
> in the multi-threaded case ?

Good point. I think is_container_init(tsk) needs a fix:

```
 - pid = task_pid(tsk);
 + pid = task_pid(tsk->group_leader);
```

Or, perhaps better, change the callers to use tsk->group_leader.

> Should we use "task->tgid == 1" ?

No, no, "task->tgid == 1" means "global" init.

> IOW to identify if the last thread of a child reaper is exiting, should we
> check "task->tgid == 1" and the "group_dead" flag in do_exit() ?

See above, but yes, as I said before I think we should do this under
the "if (group_dead)" check below.

> | > If the main thread is exiting, but is not the last thread in the
> | > group, should we let it exit and let the next thread in the group
> | > the reaper of the pid ns ?
> |
> | We can, but why? The main thread's task_struct can't go away until all
> | sub-threads exit. Its ->nsproxy will be NULL, but this doesn't matter.
>
> After the main thread exits task_child_reaper() would still refer to
> the main thread right ? So when one of the other processes in the
> namespace calls forget_original_parent(), it would reparent the process
> to the main thread - no ? The main thread still has a valid task_struct,
> but it has exited and cannot adapt children...

Yes it can't, and yes, this is somewhat against the rules.

But, afaics, this should work. Because do_wait() from the alive sub-thread
still can reap the child, note that do_wait() iterates over all sub-threads
->children lists. Please note also that do_notify_parent() uses group
signal, so it will wake up some alive sub-thread.

This is wrong for the "normal" process (because when the last thread exits
main_thread->children is lost), but this seems to be OK for the /sbin/init,
exactly because we are doing zap_pid_ns_processes().

Sukadev, may I ask you to add a fat comment about this in your patch?

> BTW, are there any actual users of multi-threaded init ? Or is this
> something that can be considered outside the "core" patchset and
> addressed soon, but separately like the signalling-container-init issue ?

Well, I don't know. Please also see the reply to Kirill's message...

Oleg.