Subject: Re: [PATCH 2/2] pidns: Support unsharing the pid namespace.
Posted by Greg Kurz on Thu, 17 Feb 2011 22:35:59 GMT
View Forum Message <> Reply to Message

On 02/17/2011 09:29 PM, Oleg Nesterov wrote:
> On 02/17, Daniel Lezcano wrote:
>>
>> On 02/15/2011 08:01 PM, Oleg Nesterov wrote:
>>>
>>> I have to admit, I can't say I like this very much. OK, if we need
>>> this, can't we just put something into, say, signal->flags so that
>>> copy_process can check and create the new namespace.
>>>
>>> Also. I remember, I already saw something like this and google found
>>> my questions. I didn't actually read the new version, perhaps my
>>> concerns were already answered...
>>>
>>> But what if the task T does unshare(CLONE_NEWPID) and then, say,
>>> pthread_create() ? Unless I missed something, the new thread won't
>>> be able to see T ?
>>
>> Right. Is it really a problem ? I mean it is a weird use case where we
>> fall in a weird situation.
>
> But this is really weird! How it is possible that the parent can't see
> its own child? No matter which thread did fork(), the new process is

Hmmm... I guess you mean the opposite. The way pid namespaces are
nested, parents always see their children. But indeed, the child thread
can't see its group leader and that's kind of unusual. Unshare a pid
namespace at your own risk. :)

> the child of any sub-thread. More precisely, it is the child of thread
> group.
>
>> I suppose we can do the same weird combination with clone.
>
> No, or we have the bug. If nothing else, kill() or wait() should work
> equally for any sub-thread. (OK, __WNOTHREAD hack is the only exception).
>
>>> and, in this case the exiting sub-namespace init also kills its
>>> parent?
>>
>> I don't think so because the zap_pid_ns_processes does not hit the
>> parent process when it browses the pidmap.
>
> OK... Honestly, right now I can't understand my own question, it was
> written a long ago. Probably I missed something.... but I'll recheck ;)

>
>>> OK, suppose it does fork() after unshare(), then another fork().
>>> In this case the second child lives in the same namespace with
>>> init created by the 1st fork, but it is not descendant ? This means
>>> in particular that if the new init exits, zap_pid_ns_processes()->
>>> do_wait() can't work.
>>
>> Hmm, good question. IMO, we should prevent such case for now in the same
>> way we added the flag 'dead', IOW adding a flag 'busy' for example.
>
> I dunno.
>
> As I said, I do not like this approach at all. But please feel free to
> ignore, it is very easy to blaim somebody else's code without suggesting
> the alternative ;)
>
> Oleg.
>
> _____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containe rs

--
Gregory Kurz gkurz@fr.ibm.com
Software Engineer @ IBM/Meiosys http://www.ibm.com
Tel +33 (0)534 638 479 Fax +33 (0)561 400 420

"Anarchy is about taking complete responsibility for yourself."
Alan Moore.

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