Subject: Re: [PATCH 0/13] Pid namespaces (OpenVZ view)
Posted by Pavel Emelianov on Thu, 24 May 2007 16:26:10 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> Pavel Emelianov <xemul@openvz.org> writes:
>
>> That's how OpenVZ sees the pid namespaces.
>>
>> The main idea is that kernel keeps operating with tasks pid
>> as it did before, but each task obtains one more pid for each
>> pid type - the virtual pid. When putting the pid to user or
>> getting the pid from it kernel operates with the virtual ones.
>
> Just a quick reaction.
>
> - I would very much like to see a minimum of 3 levels of pids,

Why not 4? From my part, I would like to know, why such nesting
is important. We have plain IPC namespaces and nobody cares.
We will have isolated network namespaces, why pids are exception?

>  being supported.  Otherwise it is easy to overlook some of the
>  cases that are required to properly support nesting, which long
>  terms seems important.
>
> - Semantically fork is easier then unshare.  Unshare can mean

This is not. When you fork, the kid shares the session and the
group with its parent, but moving this pids to new ns is bad - the
parent will happen to be half-moved. Thus you need to break the
session and the group in fork(), but this is extra complexity.

>  a lot of things, and it is easy to pick a meaning that has weird
>  side effects.  Your implementation has a serious problem in that you
>  change the value of getpid() at runtime.  Glibc does not know how to
>  cope with the value of getpid() changing.

This pid changing happens only once per task lifetime. Though I haven't
seen any problems with glibc for many years running OpenVZ and I think,
that if glibc will want to cache this getpid() value we can teach it to
uncache this value in case someone called unshare() with CLONE_NEWPIDS.

> Eric
>

Thanks,
Pavel.