Subject: Re: [PATCH 2/6] IPC namespace - utils
Posted by dev on Wed, 14 Jun 2006 11:14:49 GMT
View Forum Message <> Reply to Message

>>Agreed. Hmm. I bet I didn't see this one earlier because it is specific
>>to the unshare case. In this case I guess we should either deny the unshare
>>or simply undo all of the semaphores. Because we will never be able to
>>talk to them again.
>
> So aren't we reaching the unshare() limits ? Shouldn't we be using the
> exec() principle for the sysvipc namespace ? clear it all and start from
> scratch.
there will be more such issues with more complex namespaces. That's why
I proposed to use containers.
Any way, right now, I don't think this should be urgently fixed as there
are many other ways to crash the node when you are a root.
The rule is simple - the process changing the namespace should be
simple, w/o IPCs. For more complex namespaces (e.g. resource management
namespaces) a fork() can be required after unshare().

>>Thinking about this some more we need to unsharing the semaphore undo semantics
>>when we create a new instances of the sysvipc namespace. Which means that
>>until that piece is implemented we can't unshare the sysvipc namespace.
>
>
> no big issue I think. exit_sem() does it already. it would end up coding
> the yet unsupported unshare_semundo().
do we need really it?
my point is that while with IPCs it maybe quite easy since semundo code
already exists it can be still very hard for other namespaces which do
not track its objects associated with the task.

BTW, do we have the same problem with shm? What will happen if the task
having shm segment will change it's IPC namespace? Besides the possible
crash/stability bugs there can be more interesting effects, for example:
"user will be able to do operations on existing objects with new
namespace limitations?"...

Thanks,
Kirill