Subject: Re: semantics for namespace naming
Posted by serue on Thu, 14 Dec 2006 16:28:33 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):

(still digesting the earlier part of your email, will respond
to that later. I'm not sure whether you're laying out the
purely pid-addressed approach as an alternative, or just
extensively arguing that the container should point to an
init_pid and not an nsproxy, or whether you are saying
something differently entirely which I still need to process.)

> > struct container {
> > struct container *parent;
> > char *name;
> > struct nsproxy *nsproxy;
> > struct list_head children;
> > };
> > struct nsproxy {
> > ...
> > struct container *container;
> > };
>
> For your chosen struct container I guess if the hierarchy are
> struct containers that will work. Going from your struct container
> to anything interesting is currently a walk through the process list
> which is painful. So I would suggest putting a pointer to the
> init process of the container, that is probably better than the
> nsproxy.

Main downside of that is that we then again expect the init
process to stick around.

And since we can have a new container without having a new pidspace,
it's not even limited to one "reserved" process per pidspace.
Imagine a system with 300 users logging in, each with a
polyinstantiated /tmp directory. So now you have 300 implicit
containers due to their sys_unshare(CLONE_NS), and each of
these containers points to and reserves the PAM process which
did the unshare?

> I'm not quite convinced we need the struct container. But I have
> no fundamental objects to it either.

I'm not convinced either. I stuck it in there mainly for
description of the idea.

Though as I mentioned in my response to Suka, there is the
issue of keeping container 'vserver1' around even if both
the original nsproxy and the init process for that vserver
are gone. Because so long as one of it's decendents still
exists, we should still be able to say "kill vserver1", and
kill all it's decendents.

(And here we may want to talk about unsharing the container
namespace so that /vserver1/vserver3 can become independent
of /vserver1, but I don't like the security implications of
that)

> > Plus of course relevant sysfs stuff.
>
> /proc is actually the appropriate filesystem for this sort of
> information not sysfs. Handling the network information that
> is in sysfs is going to be hard enough.

Ok, good point. In addition to actually being process info, that'll
also make it trivial (compared to sysfs) to present different
information depending on a process' container.

thanks,
-serge

_____