

---

Subject: Re: semantics for namespace naming

Posted by [Cedric Le Goater](#) on Thu, 14 Dec 2006 13:58:29 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

Serge E. Hallyn wrote:

> Let's say we have a vserver, from which we start some jobs  
> which we want to checkpoint/restart/migrate. These are two  
> of the usages we currently foresee for the namespaces, though  
> I'd say it's safe to assume there will be more.  
>  
> I'll want to be able to address the c/r jobs by some ID in  
> order to checkpoint and kill them. I'll also want to be  
> able to address the entire vserver by some ID, in order to  
> kill it. In that case the c/r jobs should also be killed.  
> So those jobs are known by at least two id's. Furthermore, I  
> may want two vservers on the same machine, both running a c/r  
> job called 'calculate\_pi'.  
>  
> So we can look at this as a filesystem. In the above scenario,  
> we've got /sergesvserver, /sergesvserver/calculate\_pi,  
> /randomvserver, and /randomvserver/calculate\_pi. And, if  
> user hallyn logs into /sergesvserver using pam\_namespace.so,  
> unsharing his mounts namespace to get a private /tmp and /home,  
> then he ends up in /sergesvserver/unnamed1. So each nsproxy  
> has a node in the namespace id filesystem, with random names  
> unless/until it is renamed to a more meaningful name. This  
> allows us to switch to a vserver by specifying the vserver's  
> name (ln /sys/namespaces/vserver1 /proc/nsproxy or whatever  
> semantics we end up using), kill an entire vserver recursively  
> (rm -rf /sys/namespaces/vserver1), perhaps even checkpoint  
> (tar jcf /tarballs/vserver1 /sys/namespaces/vserver1) and  
> certainly rename (mv /sys/namespaces/unnamed1 /sys/namespaces/sergeprivhome).  
>  
> One key observation which I haven't made explicit is that you  
> never actually leave a nsid ("container"). If you start under  
> /vserver1, you will always be under /vserver1. I don't know of  
> any reason that would not be appropriate. If I start a nested  
> vserver from there, then to me it may be known as  
> 'vserver\_testme', while to the admin of the machine, it would be  
> known as /vserver1/vserver\_testme.  
>  
> This makes one possible implementation of the container struct:  
>  
> struct container {  
> struct container \*parent;  
> char \*name;  
> struct nsproxy \*nsproxy;  
> struct list\_head children;

```
> };  
> struct nsproxy {  
> ...  
> struct container *container;  
> };  
>  
> Plus of course relevant sysfs stuff.
```

I like the naming model. a few questions :

how do you enter only a subset of namespaces of a nsproxy/container  
and not all of it ?

what flexibility the struct container is giving us ? why not have  
container == nsproxy ?

the recursivity model looks like extra overhead. it could be flat.

C.

---

Containers mailing list  
Containers@lists.osdl.org  
<https://lists.osdl.org/mailman/listinfo/containers>

---