
Subject: semantics for namespace naming
Posted by [serue](#) on Wed, 13 Dec 2006 14:35:55 GMT
[View Forum Message](#) <> [Reply to Message](#)

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 (In /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.

-serge

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