Subject: Re: namespace and nsproxy syscalls
Posted by Cedric Le Goater on Tue, 26 Sep 2006 17:51:02 GMT
View Forum Message <> Reply to Message

Herbert Poetzl wrote:
> On Tue, Sep 26, 2006 at 11:42:10AM +0200, Cedric Le Goater wrote:
>> Hello all,
>>
>> A while ago, we expressed the need to have a new syscall specific to
>> namespaces. the clone and unshare are good candidates but we are reaching
>> the limit of the clone flags and clone has been hijacked enough.
>>
>> So, I came up with unshare_ns. the patch for the core feature follows
>> the email. Not much difference with unshare() for the moment but it gives
>> us the freedom to diverge when new namespaces come in. I have faith also !
>> If you feel it's useful, i'll send the full patchset for review on the list.
>>
>> I'd like to discuss of another syscall which would allow a process to
>> bind to a set of namespaces (== nsproxy == container) :
>>
>> bind_ns(ns_id_t id, int flags)
>>
>> bind_ns binds the current nsproxy to an id. You can only bind once and
>> you can use this id to bind another process to the same nsproxy.
>>
>> a few comments :
>>
>> * ns_id_t could be an int, a const char*, a struct ns_addr*. this is
>>   to be defined.
>> * bind_ns applies to nsproxy and not to namespaces. we could bind a
>>   specific namespace to an id using flags but i don't see the need.
>
> what if you have a setup like this:
>
> - guest using a full set of namespaces
> - host admin wants to 'adjust' a mount inside
>
> typically the host admin would enter the filesystem
> namespace (at least it is done so in Linux-VServer)
> but would avoid entering the other namespaces, like
> the pid-space or the container itself, just to do
> the mount without 'appearing' inside the guest and
> more important, with more priviledges than the guest
> processes would have ...

the flags would be useful for such scenarii.

a process unshares some namespaces, this operation creates a new

nsproxy. the process binds (identifies) to someid with

 bind_ns(someid, UNSHARE_NS_ALL)

The process does some other stuff, exec, fork, etc. and dies. but we keep
the nsproxy bound to someid because some processes are still using it.

Now, i'm back in the 'host' and I want to fully bind to the identified
nsproxy, I would use the same command

 bind_ns(someid, UNSHARE_NS_ALL)

The above operation would simply attach the process to the nsproxy.

Now, if only want to bind to the filesystem namespace of this nsproxy, I
would use :

 bind_ns(someid, UNSHARE_NS_MNT)

The above operation creates a new nsproxy object and populates it with
the identified nsproxy namespaces specified by flags (UNSHARE_NS_MNT) and
keeps the current namespaces for the others. It's a reverse unshare but
not exactly a clone().

It does some stuff again like mount a new fs and when the process dies
the nsproxy gets recycled.

you use bind_ns() for 2 different purposes in 2 differents contexts :

* to identify a nsproxy and its set of namespaces in the context of the
  process (guest) which created the nsproxy.
* to bind to some or all of a nsproxy namespaces in the context of another
  process (host)

[the UNSHARE_NS_ prefix is not well chosen. NS_ would be more
appropriate i think]

> IMHO that would require to setup two nsproxies
> which _share_ a certain namespace, but how would
> I do that? the only way I see here is to create some
> kind of nsproxy hierarchy when the guest is started
> like this:
>
> - create filesystem namespace
> - do a bind_ns()
> - create network namespace
> - do another bind_ns()
> - create uts/ipc/pid namespace

> - do another bind_ns()
> ...
>
> what if I only want to enter the network namespace
> at some point?

I think the above sequence i've described covers your need, nop ?

> IMHO some kind of unique (e.g. the virtual address
> of the kernel structure) identifier for every
> namespace would be a very good idea to have, in
> combination with a method to 'build' a proxy on
> the fly (e.g. by simply 'joining' the namespaces
> you want to use) makes the above trivial and simple

what about :

unsigned long bind_ns(unsigned long id, int flags)

when identifying the nsproxy, use the virtual address of the kernel
structure if id == 0 . returns the identifier.

> an alternative would be to have some 'copy partial'
> method to 'pick' certain namespaces from an existing
> nsproxy, but personally I would prefer to keep the
> nsproxy invisible and to work on namespaces only
> (IIRC, that was what we planned at the beginning of
> the nsproxy discussion)

Yes. I think we are reaching that now.

thanks,

C.

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