Subject: Re: [RFC] [PATCH] Re: Prefixing cgroup generic control filenames with "cgroup."

Posted by Paul Jackson on Fri, 29 Feb 2008 19:20:57 GMT

View Forum Message <> Reply to Message

> >   2) "[a-z]+\.[a-z]+(_[a-z]+)*"
>
> Why make it more complex?

It's a trade-off, between how many names the pattern covers,
and how complicated it is. I don't really have a preference
between "[a-z].*", "[a-z]+\.[a-z_]+", or other variants.

This sort of namespace partitioning is common in other venues,
such as one or two leading underscores in various C or Python
names having particular 'system' uses.

Perhaps 90% of users who ever are in a position to construct
a name in the cgroup namespace won't worry much about this one
way or the other. For them, the strlen() of the regex pattern
describing future possible kernel generated cgroup names won't
matter. A few such constructors of cgroup names will appreciate
the precision of whatever rule you create.

> The main reason it's not my primary choice is that it's an implicit
> rule that the user has to know about or risk getting bitten in future.

Ok - as I suspected.

Note again, you can't keep the user from getting bitten. You can
only avoid being one of the biters.

>From the users perspective, since you can't actually eliminate the
risk of them ever seeing a collision, any complexity that you impose
on their code risks being viewed as your taxing them more for your
benefit (avoiding any blame to you) than for their benefit (actually
avoiding all collision risk, which we can't practically do.)

All name spaces co-operatively maintained in the commons have this
risk of collision.

As such name spaces go, the cgroup name space is easy. It's a
small world. A few simple lexical conventions should suffice.

I'd suggest something like you promise to stay in the "[a-z]+\.[a-z_]+"
space, where the leading "[a-z]+" prefix is "cgroup" or one of a
modest, slowly growing, set of cgroup subsystems, plus the existing
grand-fathered names. Nothing here keeps others from intruding in

that same space; you would just be promising not to go outside of
that space.

Self-imposed restraint like this "sells" better. The vast majority can
just pay no mind; the small minority that care will appreciate that
you're imposing constraints on yourself (on cgroup kernel code) that
make their lives a little safer (one less chance of name collision)
without imposing any additional constraints on what they can do or
complexity for which they must inescapably code.

Such restraint is similar to what I see the major users of the similar
cpuset name space doing. They each pick a distinct branding prefix
with which to start the names they add below /dev/cpuset. They do so
without even a suggestion from myself, and without any consultation
amongst themselves; just seems a convenient and sensible thing to do.

--
I won't rest till it's the best ...
Programmer, Linux Scalability
Paul Jackson <pj@sgi.com> 1.940.382.4214

____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers