Subject: Re: [PATCH 06/11] sysfs: Implement sysfs tagged directory support.
Posted by Tejun Heo on Tue, 01 Jul 2008 10:30:41 GMT
View Forum Message <> Reply to Message

Hello,

Eric W. Biederman wrote:
>> Having enumed tag types limits that a sb can have map to only one tag
>> but it doesn't really prevent multiple possibly visible entries which is
>> the real unnecessary degrees of freedom. That said, I don't really
>> think it's an issue.
>
> Having a single tag type per directory and thus a single tag visible per
> directory does prevent multiple possible visible entries.
>
> That is we can check when we add the sd if there will be a conflict in
> the directory.

Yeap, that we can do.

>> I still would prefer something which is more generic. The abstraction
>> is clearer too. A sb shows untagged and a set of tags. A sd can either
>> be untagged or tagged (a single tag).
>
> That is the abstraction now.
>
> The only difference is how we represent the set of tags.
> I use and array of the valid tags.
> You use a bitmap.
>
> And array allows the lookup of the tag I am looking for before
> I search for the sd. An bitmap requires me to compare each entry.

How so? sysfs_sb->bitmap which contains enough bits for all the defined
tags and determining whether a sd should be shown or not is as simple as
single test_bit.

> For me that is a deal breaker. Currently in certain pathological
> cases we have scaling issues with sysctl and sysfs that we can
> have enormous directories that start running slowly. To fix
> lookup performance requires that we know the full name before
> we do the directory search which is the name string and the
> tag.
>
> So I having a type of tag as being of fundamental importance in
> the interface now so we don't need to refactor all of the users
> later. In addition to the fact that we need the type to know
> how to set the tags when mounting a superblock and when

> given a new kobject to create an sd for.
>
> We could make the types dynamic rather then a static enumeration but
> that seems needless complexity for now.

What I'm feeling unease about is the extra level of abstraction added by tag types. A sd is given a tag. A sb shows a set of tags. The most straight forward to implement that is to give sd a tag and test the tag against sb's set of tags. The type is added because pointer tag requires sequential matching which is usually best to avoid. It's nothing fundamental. It's an extra baggage.

>> Using ida (or idr if a pointer for private data is necessary) is really
>> easy. It'll probably take a few tens of lines of code. That said, I
>> don't think I have enough rationale to nack what you described. So, as
>> long as the tags are made static, I won't object.
>
> Sounds good. The only justification I can think of for ida tags is that
> they are smaller, and so can keep the sysfs_dirents smaller. Which
> occasionally is a significant concern. Still that should be an optimization
> that we can apply later, as it is not a structural difference in the code.
>
> Just to confirm. Do you the two operations:
> mount_tag - called only when the sb is mounted
> kobject_tag - called when we create new sd or rename an sd
>
> Cause you to view an the tags as dynamic?

The thing is that I don't really see why there's tagged_dir_ops at all. What's needed is tagged sd's and sb's which can show subset of those tags, so adding callback ops for tags just doesn't make much sense to me. The interface should ideally be...

1. alloc/release tag
2. set / change / remove tag on sd
3. enable / disable tag on a sb

This has been my opinion from the beginning. Unless the tags need to be changed dynamically on demand (which I hope is not the case), there just is plainly no reason to have callbacks for tags.

Thanks.

--
tejun

_____
Containers mailing list
Containers@lists.linux-foundation.org