Subject: Re: [PATCH 06/11] sysfs: Implement sysfs tagged directory support.
Posted by ebiederm on Tue, 01 Jul 2008 12:30:47 GMT
View Forum Message <> Reply to Message

Tejun Heo <htejun@gmail.com> writes:

> 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.

What we are implementing is not, a sb with a set of tags that are displayed,
but directories with a single tag that is displayed. The sb just happens
to hold the state for the directories.

A directory displaying only a single tag is an necessary constraint for
a large number of reasons.

>> 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.

Yes. The compare happens to be test_bit.

With a bitmap you must visit each dirent with a given name and see if
it has a tag that is displayed.

With an array you can lookup the tag aprori and can potentially do a
hash table lookup or a tree lookup and are not required to visit each
entry.

> 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.

That is just one important aspect of it.   We need a way to describe
which tag a sb,directory pair displays. It is a fundamental concept.

>>> 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.

We need callbacks for interfacing with the kobject layer, and for
selecting our set of tags at mount time. Not tagged_dir_ops so much
as tagged_type_ops.

> 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
    Agreed.

> 2. set / change / remove tag on sd
    Essentially agreed.

    Create an sd with a tag, change the tag on a sd.
    Having an untagged sd in a directory that requires tags should
    not be allowed.

> 3. enable / disable tag on a sb
    Disagree that is too flexible. Tags on a sb need to be
    unchanging or else we get vfs layer issues.

   Further the abstraction is logically exactly one tag on a (sb,directory) pair.

The operations needed are.
- Select the set of tags on a sb (at mount time)
  This requires we call a set of callbacks. [My mount_sb callback]

- release a tag (which implies removing all tagged entries and
  removing the sb reference)

4. Interface with the kobject layer.
   kobject_add calls sysfs_create_dir
   kboject_rename calls sysfs_rename_dir
   kobject_del calls sysfs_remove_dir

   For the first two operations we need a helper function to go from a
   kobject to a tag.

   For the second two operations we need to go from a kobject to a sd.

> 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.

We don't need callbacks to poll to see if the tags on a sd have
changed.

We need helper functions for interfacing with the rest of the kernel.

Eric