Subject: Re: [PATCH 1/4] sysfs: Remove first pass at shadow directory support
Posted by Tejun Heo on Sun, 22 Jul 2007 18:35:27 GMT
View Forum Message <> Reply to Message

Hello,

Hope my late review isn't too late.

Eric W. Biederman wrote:
> While shadow directories appear to be a good idea, the current scheme
> of controlling their creation and destruction outside of sysfs appears
> to be a locking and maintenance nightmare in the face of sysfs
> directories dynamically coming and going.

Yeah, shadow directory was a real PITA while restructuring sysfs. I tried hard to keep it working but, without test case and detailed documentation, I'm pretty sure I broke something.

My feeling was that it was merged too early and implementation was too tightly coupled with other more regular paths. Anyways, glad shadow dirs are getting some loving care. If properly done, we should be able to simplify sysfs_get_dentry(), removal logic and save a pointer in sysfs_dirent.

> Which can now occur for
> directories containing network devices when CONFIG_SYSFS_DEPRECATED is
> not set.

Shadow directories were always pinned - both the shadowed one and shadows. Well, that was the theory at least.

```
> -int sysfs_rename_dir(struct kobject *kobj, struct sysfs_dirent *new_parent_sd,
> -       const char *new_name)
> +int sysfs_rename_dir(struct kobject * kobj, const char *new_name)
>  {
> - struct sysfs_dirent *sd = kobj->sd;
> - struct dentry *new_parent = NULL;
> + struct sysfs_dirent *sd;
> + struct dentry *parent = NULL;
>   struct dentry *old_dentry = NULL, *new_dentry = NULL;
> + struct sysfs_dirent *parent_sd;
>   const char *dup_name = NULL;
>   int error;
>
> + if (!kobj->parent)
> +  return -EINVAL;
```

Please don't do this. The goal is to decouple sysfs and kobj.

```
>   /* get dentries */
> + sd = kobj->sd;
>   old_dentry = sysfs_get_dentry(sd);
>   if (IS_ERR(old_dentry)) {
>    error = PTR_ERR(old_dentry);
>    goto out_dput;
>   }
>
> - new_parent = sysfs_get_dentry(new_parent_sd);
> - if (IS_ERR(new_parent)) {
> -  error = PTR_ERR(new_parent);
> + parent_sd = kobj->parent->sd;
> + parent = sysfs_get_dentry(parent_sd);
> + if (IS_ERR(parent)) {
> +  error = PTR_ERR(parent);
>    goto out_dput;
```

You can simply do parent = old_dentry->d_parent.  parent is guaranteed
to be there as long as we hold old_dentry.  In the original code,
new_parent needed to be looked because new_parent wasn't necessarily
old_dentry's parent.

```
> @@ -965,10 +957,10 @@ int sysfs_rename_dir(struct kobject *kobj, struct sysfs_dirent
*new_parent_sd,
>   out_drop:
>   d_drop(new_dentry);
>   out_unlock:
> - mutex_unlock(&new_parent->d_inode->i_mutex);
> + mutex_unlock(&parent->d_inode->i_mutex);
>   out_dput:
>   kfree(dup_name);
> - dput(new_parent);
> + dput(parent);
```

So, dput(parent) can go away too.

```
> diff --git a/fs/sysfs/group.c b/fs/sysfs/group.c
> index f318b73..4606f7c 100644
> --- a/fs/sysfs/group.c
> +++ b/fs/sysfs/group.c
> @@ -13,7 +13,6 @@
>  #include <linux/dcache.h>
>  #include <linux/namei.h>
>  #include <linux/err.h>
> -#include <linux/fs.h>
>  #include <asm/semaphore.h>
>  #include "sysfs.h"
```

Can you explain this one a bit?

Thanks.

--
tejun

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