
Subject: [PATCH 20/25] sysfs: Rename Support multiple superblocks

Posted by [ebiederm](#) on Tue, 07 Aug 2007 21:31:18 GMT

[View Forum Message](#) <> [Reply to Message](#)

This patch modifies the sysfs_rename_dir and sysfs_move_dir to support multiple sysfs dentry trees rooted in different sysfs superblocks.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

fs/sysfs/dir.c | 193 ++++++-----
1 files changed, 136 insertions(+), 57 deletions(-)

diff --git a/fs/sysfs/dir.c b/fs/sysfs/dir.c

index ac45523..34eabf4 100644

--- a/fs/sysfs/dir.c

+++ b/fs/sysfs/dir.c

```
@@ -855,42 +855,113 @@ struct dentry *sysfs_get_dentry(struct super_block *sb, struct
sysfs_dirent *sd)
    return dentry;
}
```

```
+struct sysfs_rename_struct {
```

```
+ struct list_head list;
```

```
+ struct dentry *old_dentry;
```

```
+ struct dentry *new_dentry;
```

```
+ struct dentry *old_parent;
```

```
+ struct dentry *new_parent;
```

```
+};
```

```
+
```

```
+static void post_rename(struct list_head *head)
```

```
+{
```

```
+ struct sysfs_rename_struct *srs;
```

```
+ while (!list_empty(head)) {
```

```
+ srs = list_entry(head->next, struct sysfs_rename_struct, list);
```

```
+ dput(srs->old_dentry);
```

```
+ dput(srs->new_dentry);
```

```
+ dput(srs->old_parent);
```

```
+ dput(srs->new_parent);
```

```
+ list_del(&srs->list);
```

```
+ kfree(srs);
```

```
+ }
```

```
+ }
```

```
+
```

```
+static int prep_rename(struct list_head *head,
```

```
+ struct sysfs_dirent *sd, struct sysfs_dirent *new_parent_sd,
```

```
+ const char *name)
```

```
+{
```

```

+ struct sysfs_rename_struct *srs;
+ struct super_block *sb;
+ struct dentry *dentry;
+ int error;
+
+ list_for_each_entry(sb, &sysfs_fs_type.fs_supers, s_instances) {
+     dentry = sysfs_get_dentry(sb, sd);
+     if (!dentry)
+         continue;
+     if (IS_ERR(dentry)) {
+         error = PTR_ERR(dentry);
+         goto err_out;
+     }
+
+     srs = kzalloc(sizeof(*srs), GFP_KERNEL);
+     if (!srs) {
+         dput(dentry);
+         goto err_out;
+     }
+
+     INIT_LIST_HEAD(&srs->list);
+     list_add(head, &srs->list);
+     srs->old_dentry = dentry;
+     srs->old_parent = dget(dentry->d_parent);
+
+     dentry = sysfs_get_dentry(sb, new_parent_sd);
+     if (IS_ERR(dentry)) {
+         error = PTR_ERR(dentry);
+         goto err_out;
+     }
+     srs->new_parent = dentry;
+
+     error = -ENOMEM;
+     dentry = d_alloc_name(srs->new_parent, name);
+     if (!dentry)
+         goto err_out;
+     srs->new_dentry = dentry;
+ }
+ return 0;
+
+err_out:
+ post_rename(head);
+ return error;
+}
+
+int sysfs_rename_dir(struct kobject *kobj, const char *new_name)
+{

```

```

    struct sysfs_dirent *sd = kobj->sd;
- struct dentry *parent = NULL;
- struct dentry *old_dentry = NULL, *new_dentry = NULL;
+ struct list_head todo;
+ struct sysfs_rename_struct *srs;
+ struct inode *parent_inode = NULL;
    const char *dup_name = NULL;
    int error;

+ INIT_LIST_HEAD(&todo);
    mutex_lock(&sysfs_rename_mutex);

    error = 0;
    if (strcmp(sd->s_name, new_name) == 0)
        goto out; /* nothing to rename */

- /* get the original dentry */
- old_dentry = sysfs_get_dentry(sysfs_sb, sd);
- if (IS_ERR(old_dentry)) {
-     error = PTR_ERR(old_dentry);
-     goto out;
- }
+ sysfs_grab_supers();
+ error = prep_rename(&todo, sd, sd->s_parent, new_name);
+ if (error)
+     goto out_release;

- parent = old_dentry->d_parent;
+ error = -ENOMEM;
+ mutex_lock(&sysfs_mutex);
+ parent_inode = sysfs_get_inode(sd->s_parent);
+ mutex_unlock(&sysfs_mutex);
+ if (!parent_inode)
+     goto out_release;

- /* lock parent and get dentry for new name */
- mutex_lock(&parent->d_inode->i_mutex);
+ mutex_lock(&parent_inode->i_mutex);
    mutex_lock(&sysfs_mutex);

    error = -EEXIST;
    if (sysfs_find_dirent(sd->s_parent, new_name))
        goto out_unlock;

- error = -ENOMEM;
- new_dentry = d_alloc_name(parent, new_name);
- if (!new_dentry)
-     goto out_unlock;

```

```

-
/* rename kobject and sysfs_dirent */
error = -ENOMEM;
new_name = dup_name = kstrdup(new_name, GFP_KERNEL);
@@ -901,23 +972,25 @@ int sysfs_rename_dir(struct kobject * kobj, const char *new_name)
if (error)
goto out_unlock;

- mutex_lock(&sysfs_mutex);
dup_name = sd->s_name;
sd->s_name = new_name;
- mutex_unlock(&sysfs_mutex);

/* rename */
- d_add(new_dentry, NULL);
- d_move(old_dentry, new_dentry);
+ list_for_each_entry(srs, &todo, list) {
+ d_add(srs->new_dentry, NULL);
+ d_move(srs->old_dentry, srs->new_dentry);
+ }

error = 0;
- out_unlock:
+out_unlock:
mutex_unlock(&sysfs_mutex);
- mutex_unlock(&parent->d_inode->i_mutex);
+ mutex_unlock(&parent_inode->i_mutex);
kfree(dup_name);
- dput(old_dentry);
- dput(new_dentry);
- out:
+out_release:
+ iput(parent_inode);
+ post_rename(&todo);
+ sysfs_release_supers();
+out:
mutex_unlock(&sysfs_rename_mutex);
return error;
}
@@ -926,10 +999,12 @@ int sysfs_move_dir(struct kobject *kobj, struct kobject
*new_parent_kobj)
{
struct sysfs_dirent *sd = kobj->sd;
struct sysfs_dirent *new_parent_sd;
- struct dentry *old_parent, *new_parent = NULL;
- struct dentry *old_dentry = NULL, *new_dentry = NULL;
+ struct list_head todo;
+ struct sysfs_rename_struct *srs;

```

```

+ struct inode *old_parent_inode = NULL, *new_parent_inode = NULL;
  int error;

+ INIT_LIST_HEAD(&todo);
  mutex_lock(&sysfs_rename_mutex);
  BUG_ON(!sd->s_parent);
  new_parent_sd = new_parent_kobj->sd ? new_parent_kobj->sd : &sysfs_root;
@@ -938,24 +1013,29 @@ int sysfs_move_dir(struct kobject *kobj, struct kobject
 *new_parent_kobj)
  if (sd->s_parent == new_parent_sd)
    goto out; /* nothing to move */

- /* get dentries */
- old_dentry = sysfs_get_dentry(sysfs_sb, sd);
- if (IS_ERR(old_dentry)) {
-   error = PTR_ERR(old_dentry);
-   goto out;
- }
- old_parent = old_dentry->d_parent;
+ sysfs_grab_supers();
+ error = prep_rename(&todo, sd, new_parent_sd, sd->s_name);
+ if (error)
+   goto out_release;

- new_parent = sysfs_get_dentry(sysfs_sb, new_parent_sd);
- if (IS_ERR(new_parent)) {
-   error = PTR_ERR(new_parent);
-   goto out;
- }
+ error = -ENOMEM;
+ mutex_lock(&sysfs_mutex);
+ old_parent_inode = sysfs_get_inode(sd->s_parent);
+ mutex_unlock(&sysfs_mutex);
+ if (!old_parent_inode)
+   goto out_release;
+
+ error = -ENOMEM;
+ mutex_lock(&sysfs_mutex);
+ new_parent_inode = sysfs_get_inode(new_parent_sd);
+ mutex_unlock(&sysfs_mutex);
+ if (!new_parent_inode)
+   goto out_release;

again:
- mutex_lock(&old_parent->d_inode->i_mutex);
- if (!mutex_trylock(&new_parent->d_inode->i_mutex)) {
-   mutex_unlock(&old_parent->d_inode->i_mutex);
+ mutex_lock(&old_parent_inode->i_mutex);

```

```

+ if (!mutex_trylock(&new_parent_inode->i_mutex)) {
+ mutex_unlock(&old_parent_inode->i_mutex);
+ goto again;
+ }
+ mutex_lock(&sysfs_mutex);
@@ -964,15 +1044,11 @@ again:
+ if (sysfs_find_dirent(new_parent_sd, sd->s_name))
+ goto out_unlock;

- error = -ENOMEM;
- new_dentry = d_alloc_name(new_parent, sd->s_name);
- if (!new_dentry)
- goto out_unlock;
-
+ error = 0;
- d_add(new_dentry, NULL);
- d_move(old_dentry, new_dentry);
- dput(new_dentry);
+ list_for_each_entry(srs, &todo, list) {
+ d_add(srs->new_dentry, NULL);
+ d_move(srs->old_dentry, srs->new_dentry);
+ }

+ /* Remove from old parent's list and insert into new parent's list. */
+ sysfs_unlink_sibling(sd);
@@ -981,14 +1057,17 @@ again:
+ sd->s_parent = new_parent_sd;
+ sysfs_link_sibling(sd);

- out_unlock:
+out_unlock:
+ mutex_unlock(&sysfs_mutex);
- mutex_unlock(&new_parent->d_inode->i_mutex);
- mutex_unlock(&old_parent->d_inode->i_mutex);
- out:
- dput(new_parent);
- dput(old_dentry);
- dput(new_dentry);
+ mutex_unlock(&new_parent_inode->i_mutex);
+ mutex_unlock(&old_parent_inode->i_mutex);
+
+out_release:
+ iput(new_parent_inode);
+ iput(old_parent_inode);
+ post_rename(&todo);
+ sysfs_release_supers();
+out:
+ mutex_unlock(&sysfs_rename_mutex);

```

```
    return error;
}
--
1.5.1.1.181.g2de0
```

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