Subject: [PATCH 10/15] sysfs: Merge sysfs_rename_dir and sysfs_move_dir
Posted by ebiederm on Fri, 04 Jul 2008 01:17:30 GMT
View Forum Message <> Reply to Message

These two functions do 90% of the same work and it doesn't
significantly obfuscate the function to allow both the parent dir and
the name to change at the same time. So merge them together to
simplify maintenance, and increase testing.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 fs/sysfs/dir.c | 121 ++++++++++++++++++++++++++++++++----------------------
 1 files changed, 38 insertions(+), 83 deletions(-)

```
diff --git a/fs/sysfs/dir.c b/fs/sysfs/dir.c
index 6dc3376..fe2bb1c 100644
--- a/fs/sysfs/dir.c
+++ b/fs/sysfs/dir.c
@@ -924,44 +924,57 @@ err_out:
  return error;
 }

-int sysfs_rename_dir(struct kobject * kobj, const char *new_name)
+static int sysfs_mv_dir(struct sysfs_dirent *sd,
+ struct sysfs_dirent *new_parent_sd, const char *new_name)
 {
- struct sysfs_dirent *sd = kobj->sd;
  struct list_head todo;
  struct sysfs_rename_struct *srs;
- struct inode *parent_inode = NULL;
+ struct inode *old_parent_inode = NULL, *new_parent_inode = NULL;
  const char *dup_name = NULL;
  const void *old_tag, *tag;
  int error;

  INIT_LIST_HEAD(&todo);
+ BUG_ON(!sd->s_parent);
  mutex_lock(&sysfs_rename_mutex);
+ if (!new_parent_sd)
+  new_parent_sd = &sysfs_root;
+
  old_tag = sd->s_tag;
  tag = sysfs_creation_tag(sd->s_parent, sd);

  error = 0;
- if ((old_tag == tag) && (strcmp(sd->s_name, new_name) == 0))
-  goto out; /* nothing to rename */
+ if ((sd->s_parent == new_parent_sd) && (old_tag == tag) &&
```

```
+     (strcmp(sd->s_name, new_name) == 0))
+  goto out; /* nothing to do */

  sysfs_grab_supers();
  if (old_tag == tag) {
-  error = prep_rename(&todo, sd, sd->s_parent, new_name);
+  error = prep_rename(&todo, sd, new_parent_sd, new_name);
    if (error)
     goto out_release;
  }

  error = -ENOMEM;
  mutex_lock(&sysfs_mutex);
- parent_inode = sysfs_get_inode(sd->s_parent);
+ old_parent_inode = sysfs_get_inode(sd->s_parent);
+ new_parent_inode = sysfs_get_inode(new_parent_sd);
  mutex_unlock(&sysfs_mutex);
- if (!parent_inode)
+ if (!old_parent_inode || !new_parent_inode)
    goto out_release;

- mutex_lock(&parent_inode->i_mutex);
+again:
+ mutex_lock(&old_parent_inode->i_mutex);
+ if (old_parent_inode != new_parent_inode) {
+  if (!mutex_trylock(&new_parent_inode->i_mutex)) {
+   mutex_unlock(&old_parent_inode->i_mutex);
+   goto again;
+  }
+ }
  mutex_lock(&sysfs_mutex);

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

  /* rename sysfs_dirent */
@@ -974,7 +987,7 @@ int sysfs_rename_dir(struct kobject * kobj, const char *new_name)
  sd->s_name = new_name;
  sd->s_tag = tag;

- /* rename */
+ /* rename dcache entries */
  list_for_each_entry(srs, &todo, list) {
    d_add(srs->new_dentry, NULL);
    d_move(srs->old_dentry, srs->new_dentry);
@@ -994,77 +1007,6 @@ int sysfs_rename_dir(struct kobject * kobj, const char *new_name)
```

```
  }
 }

- error = 0;
-out_unlock:
- mutex_unlock(&sysfs_mutex);
- mutex_unlock(&parent_inode->i_mutex);
- kfree(dup_name);
-out_release:
- iput(parent_inode);
- post_rename(&todo);
- sysfs_release_supers();
-out:
- mutex_unlock(&sysfs_rename_mutex);
- return error;
-}
-
-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 list_head todo;
- struct sysfs_rename_struct *srs;
- struct inode *old_parent_inode = NULL, *new_parent_inode = NULL;
- int error;
- const void *tag;
-
- 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;
- tag = sd->s_tag;
-
- error = 0;
- if (sd->s_parent == new_parent_sd)
-  goto out; /* nothing to move */
-
- sysfs_grab_supers();
- error = prep_rename(&todo, sd, new_parent_sd, sd->s_name);
- if (error)
-  goto out_release;
-
- 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_inode->i_mutex);
- if (!mutex_trylock(&new_parent_inode->i_mutex)) {
-  mutex_unlock(&old_parent_inode->i_mutex);
-  goto again;
- }
- mutex_lock(&sysfs_mutex);
-
- error = -EEXIST;
- if (sysfs_find_dirent(new_parent_sd, tag, sd->s_name))
-  goto out_unlock;
-
- error = 0;
- 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);
  sysfs_get(new_parent_sd);
@@ -1072,10 +1014,13 @@ again:
  sd->s_parent = new_parent_sd;
  sysfs_link_sibling(sd);

+ error = 0;
 out_unlock:
  mutex_unlock(&sysfs_mutex);
- mutex_unlock(&new_parent_inode->i_mutex);
+ if (new_parent_inode != old_parent_inode)
+  mutex_unlock(&new_parent_inode->i_mutex);
  mutex_unlock(&old_parent_inode->i_mutex);
+ kfree(dup_name);

 out_release:
  iput(new_parent_inode);
@@ -1087,6 +1032,16 @@ out:
  return error;
 }
```

```
+int sysfs_rename_dir(struct kobject * kobj, const char *new_name)
+{
+ return sysfs_mv_dir(kobj->sd, kobj->sd->s_parent, new_name);
+}
+
+int sysfs_move_dir(struct kobject *kobj, struct kobject *new_parent_kobj)
+{
+ return sysfs_mv_dir(kobj->sd, new_parent_kobj->sd, kobj->sd->s_name);
+}
+
 /* Relationship between s_mode and the DT_xxx types */
 static inline unsigned char dt_type(struct sysfs_dirent *sd)
 {
--
1.5.3.rc6.17.g1911
```