Subject: [PATCH 09/25] sysfs: Move sysfs_get_dentry below __sysfs_get_dentry
Posted by ebiederm on Tue, 07 Aug 2007 21:17:33 GMT
View Forum Message <> Reply to Message

sysfs_get_dentry is higher in fs/sysfs/dir.c then is needed and it the
dependencies get simpler if we move it down in the file to where I
have placed __sysfs_get_dentry. So this patch just moves
sysfs_get_dentry so code movement doesn't get confused with later code
changes.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
fs/sysfs/dir.c | 196 ++++++++++++++++++++++++++++++++----------------------
1 files changed, 98 insertions(+), 98 deletions(-)

diff --git a/fs/sysfs/dir.c b/fs/sysfs/dir.c
index 2caa5e0..59a9ce8 100644
--- a/fs/sysfs/dir.c
+++ b/fs/sysfs/dir.c
@@ -73,104 +73,6 @@ static void sysfs_unlink_sibling(struct sysfs_dirent *sd)
}

/**
- * sysfs_get_dentry - get dentry for the given sysfs_dirent
- * @sd: sysfs_dirent of interest
- *
- * Get dentry for @sd. Dentry is looked up if currently not
- * present. This function climbs sysfs_dirent tree till it
- * reaches a sysfs_dirent with valid dentry attached and descends
- * down from there looking up dentry for each step.
- *
- * LOCKING:
- * Kernel thread context (may sleep)
- *
- * RETURNS:
- * Pointer to found dentry on success, ERR_PTR() value on error.
- */
-struct dentry *sysfs_get_dentry(struct sysfs_dirent *sd)
-{
- struct sysfs_dirent *cur;
- struct dentry *parent_dentry, *dentry;
- int i, depth;
-
- /* Find the first parent which has valid s_dentry and get the
- * dentry.
- */
- mutex_lock(&sysfs_mutex);
- restart0:

```
- spin_lock(&sysfs_assoc_lock);
- restart1:
- spin_lock(&dcache_lock);
-
- dentry = NULL;
- depth = 0;
- cur = sd;
- while (!cur->s_dentry || !cur->s_dentry->d_inode) {
-  if (cur->s_flags & SYSFS_FLAG_REMOVED) {
-   dentry = ERR_PTR(-ENOENT);
-   depth = 0;
-   break;
-  }
-  cur = cur->s_parent;
-  depth++;
- }
- if (!IS_ERR(dentry))
-  dentry = dget_locked(cur->s_dentry);
-
- spin_unlock(&dcache_lock);
- spin_unlock(&sysfs_assoc_lock);
-
- /* from the found dentry, look up depth times */
- while (depth--) {
-  /* find and get depth'th ancestor */
-  for (cur = sd, i = 0; cur && i < depth; i++)
-   cur = cur->s_parent;
-
-  /* This can happen if tree structure was modified due
-   * to move/rename.  Restart.
-   */
-  if (i != depth) {
-   dput(dentry);
-   goto restart0;
-  }
-
-  sysfs_get(cur);
-
-  mutex_unlock(&sysfs_mutex);
-
-  /* look it up */
-  parent_dentry = dentry;
-  dentry = lookup_one_len_kern(cur->s_name, parent_dentry,
-          strlen(cur->s_name));
-  dput(parent_dentry);
-
-  if (IS_ERR(dentry)) {
-   sysfs_put(cur);
```

```
-   return dentry;
-  }
-
-  mutex_lock(&sysfs_mutex);
-  spin_lock(&sysfs_assoc_lock);
-
-  /* This, again, can happen if tree structure has
-   * changed and we looked up the wrong thing.  Restart.
-   */
-  if (cur->s_dentry != dentry) {
-   dput(dentry);
-   sysfs_put(cur);
-   goto restart1;
-  }
-
-  spin_unlock(&sysfs_assoc_lock);
-
-  sysfs_put(cur);
- }
-
- mutex_unlock(&sysfs_mutex);
- return dentry;
-}
-
-/**
  * sysfs_get_active - get an active reference to sysfs_dirent
  * @sd: sysfs_dirent to get an active reference to
  *
@@ -886,6 +788,104 @@ static struct dentry *__sysfs_get_dentry(struct super_block *sb, struct
sysfs_di
  return dentry;
 }

+/**
+ * sysfs_get_dentry - get dentry for the given sysfs_dirent
+ * @sd: sysfs_dirent of interest
+ *
+ * Get dentry for @sd.  Dentry is looked up if currently not
+ * present.  This function climbs sysfs_dirent tree till it
+ * reaches a sysfs_dirent with valid dentry attached and descends
+ * down from there looking up dentry for each step.
+ *
+ * LOCKING:
+ * Kernel thread context (may sleep)
+ *
+ * RETURNS:
+ * Pointer to found dentry on success, ERR_PTR() value on error.
+ */
```

```
+struct dentry *sysfs_get_dentry(struct sysfs_dirent *sd)
+{
+ struct sysfs_dirent *cur;
+ struct dentry *parent_dentry, *dentry;
+ int i, depth;
+
+ /* Find the first parent which has valid s_dentry and get the
+  * dentry.
+  */
+ mutex_lock(&sysfs_mutex);
+ restart0:
+ spin_lock(&sysfs_assoc_lock);
+ restart1:
+ spin_lock(&dcache_lock);
+
+ dentry = NULL;
+ depth = 0;
+ cur = sd;
+ while (!cur->s_dentry || !cur->s_dentry->d_inode) {
+  if (cur->s_flags & SYSFS_FLAG_REMOVED) {
+   dentry = ERR_PTR(-ENOENT);
+   depth = 0;
+   break;
+  }
+  cur = cur->s_parent;
+  depth++;
+ }
+ if (!IS_ERR(dentry))
+  dentry = dget_locked(cur->s_dentry);
+
+ spin_unlock(&dcache_lock);
+ spin_unlock(&sysfs_assoc_lock);
+
+ /* from the found dentry, look up depth times */
+ while (depth--) {
+  /* find and get depth'th ancestor */
+  for (cur = sd, i = 0; cur && i < depth; i++)
+   cur = cur->s_parent;
+
+  /* This can happen if tree structure was modified due
+   * to move/rename.  Restart.
+   */
+  if (i != depth) {
+   dput(dentry);
+   goto restart0;
+  }
+
+  sysfs_get(cur);
```

```
+
+  mutex_unlock(&sysfs_mutex);
+
+  /* look it up */
+  parent_dentry = dentry;
+  dentry = lookup_one_len_kern(cur->s_name, parent_dentry,
+          strlen(cur->s_name));
+  dput(parent_dentry);
+
+  if (IS_ERR(dentry)) {
+   sysfs_put(cur);
+   return dentry;
+  }
+
+  mutex_lock(&sysfs_mutex);
+  spin_lock(&sysfs_assoc_lock);
+
+  /* This, again, can happen if tree structure has
+   * changed and we looked up the wrong thing.  Restart.
+   */
+  if (cur->s_dentry != dentry) {
+   dput(dentry);
+   sysfs_put(cur);
+   goto restart1;
+  }
+
+  spin_unlock(&sysfs_assoc_lock);
+
+  sysfs_put(cur);
+ }
+
+ mutex_unlock(&sysfs_mutex);
+ return dentry;
+}
+
 int sysfs_rename_dir(struct kobject * kobj, const char *new_name)
 {
  struct sysfs_dirent *sd;
--
1.5.1.1.181.g2de0
```