Subject: [PATCH 1/4] sysfs: Remove first pass at shadow directory support
Posted by ebiederm on Thu, 19 Jul 2007 04:43:46 GMT
View Forum Message <> Reply to Message

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

This patch removes everything from the initial shadow directory support
that allowed the shadow directory creation to be controlled at a higher
level. So except for a few bits of sysfs_rename_dir everything from
commit b592fcfe7f06c15ec11774b5be7ce0de3aa86e73 is now gone.

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

This patchset is against 589f1e81bde732dd0b1bc5d01b6bddd4bcb4527b
in Linus's tree. I took a quick skim through your tree and all
of the sysfs patches appear to be merged.

Sorry for taking so long getting this done but the locking
changes in the last round for Tejun were significant. Plus
there was some additional work needed to actually test the
shadow directory support.

```
 fs/sysfs/dir.c          |  167 ++++++----------------------------------------
 fs/sysfs/group.c        |    1 -
 fs/sysfs/inode.c        |   10 ---
 fs/sysfs/mount.c        |    2 +-
 fs/sysfs/sysfs.h        |    6 --
 include/linux/kobject.h |    5 --
 include/linux/sysfs.h   |   27 +-------
 lib/kobject.c           |   44 ++-----------
 8 files changed, 33 insertions(+), 229 deletions(-)

diff --git a/fs/sysfs/dir.c b/fs/sysfs/dir.c
index 048e605..f88130c 100644
--- a/fs/sysfs/dir.c
+++ b/fs/sysfs/dir.c
@@ -569,9 +569,6 @@ static void sysfs_drop_dentry(struct sysfs_dirent *sd)
  spin_unlock(&dcache_lock);
  spin_unlock(&sysfs_assoc_lock);

- /* dentries for shadowed inodes are pinned, unpin */
- if (dentry && sysfs_is_shadowed_inode(dentry->d_inode))
-  dput(dentry);
```

```
  dput(dentry);

  /* adjust nlink and update timestamp */
@@ -723,19 +720,15 @@ int sysfs_create_subdir(struct kobject *kobj, const char *name,
 /**
  * sysfs_create_dir - create a directory for an object.
  * @kobj:  object we're creating directory for.
- * @shadow_parent: parent object.
  */
-int sysfs_create_dir(struct kobject *kobj,
-       struct sysfs_dirent *shadow_parent_sd)
+int sysfs_create_dir(struct kobject * kobj)
 {
  struct sysfs_dirent *parent_sd, *sd;
  int error = 0;

  BUG_ON(!kobj);

- if (shadow_parent_sd)
-  parent_sd = shadow_parent_sd;
- else if (kobj->parent)
+ if (kobj->parent)
   parent_sd = kobj->parent->sd;
  else if (sysfs_mount && sysfs_mount->mnt_sb)
   parent_sd = sysfs_mount->mnt_sb->s_root->d_fsdata;
@@ -887,45 +880,44 @@ void sysfs_remove_dir(struct kobject * kobj)
  __sysfs_remove_dir(sd);
 }

-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;
+
  /* 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;
  }

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

- new_dentry = lookup_one_len(new_name, new_parent, strlen(new_name));
+ new_dentry = lookup_one_len(new_name, parent, strlen(new_name));
  if (IS_ERR(new_dentry)) {
   error = PTR_ERR(new_dentry);
   goto out_unlock;
  }

- /* By allowing two different directories with the same
-  * d_parent we allow this routine to move between different
-  * shadows of the same directory
-  */
  error = -EINVAL;
- if (old_dentry->d_parent->d_inode != new_parent->d_inode ||
-     new_dentry->d_parent->d_inode != new_parent->d_inode ||
-     old_dentry == new_dentry)
+ if (old_dentry == new_dentry)
   goto out_unlock;

  error = -EEXIST;
@@ -952,9 +944,9 @@ int sysfs_rename_dir(struct kobject *kobj, struct sysfs_dirent
*new_parent_sd,
  mutex_lock(&sysfs_mutex);

  sysfs_unlink_sibling(sd);
- sysfs_get(new_parent_sd);
+ sysfs_get(parent_sd);
  sysfs_put(sd->s_parent);
- sd->s_parent = new_parent_sd;
```

```
+ sd->s_parent = parent_sd;
  sysfs_link_sibling(sd);

  mutex_unlock(&sysfs_mutex);
@@ -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);
  dput(old_dentry);
  dput(new_dentry);
  return error;
@@ -1189,121 +1181,6 @@ static loff_t sysfs_dir_lseek(struct file * file, loff_t offset, int origin)
  return offset;
 }

-
-/**
- * sysfs_make_shadowed_dir - Setup so a directory can be shadowed
- * @kobj: object we're creating shadow of.
- */
-
-int sysfs_make_shadowed_dir(struct kobject *kobj,
- void * (*follow_link)(struct dentry *, struct nameidata *))
-{
- struct dentry *dentry;
- struct inode *inode;
- struct inode_operations *i_op;
-
- /* get dentry for @kobj->sd, dentry of a shadowed dir is pinned */
- dentry = sysfs_get_dentry(kobj->sd);
- if (IS_ERR(dentry))
-  return PTR_ERR(dentry);
-
- inode = dentry->d_inode;
- if (inode->i_op != &sysfs_dir_inode_operations) {
-  dput(dentry);
-  return -EINVAL;
- }
-
- i_op = kmalloc(sizeof(*i_op), GFP_KERNEL);
- if (!i_op)
```

```
-  return -ENOMEM;
-
- memcpy(i_op, &sysfs_dir_inode_operations, sizeof(*i_op));
- i_op->follow_link = follow_link;
-
- /* Locking of inode->i_op?
-  * Since setting i_op is a single word write and they
-  * are atomic we should be ok here.
-  */
- inode->i_op = i_op;
- return 0;
-}
-
-/**
- * sysfs_create_shadow_dir - create a shadow directory for an object.
- * @kobj: object we're creating directory for.
- *
- * sysfs_make_shadowed_dir must already have been called on this
- * directory.
- */
-
-struct sysfs_dirent *sysfs_create_shadow_dir(struct kobject *kobj)
-{
- struct sysfs_dirent *parent_sd = kobj->sd->s_parent;
- struct dentry *dir, *parent, *shadow;
- struct inode *inode;
- struct sysfs_dirent *sd;
- struct sysfs_addrm_cxt acxt;
-
- dir = sysfs_get_dentry(kobj->sd);
- if (IS_ERR(dir)) {
-  sd = (void *)dir;
-  goto out;
- }
- parent = dir->d_parent;
-
- inode = dir->d_inode;
- sd = ERR_PTR(-EINVAL);
- if (!sysfs_is_shadowed_inode(inode))
-  goto out_dput;
-
- shadow = d_alloc(parent, &dir->d_name);
- if (!shadow)
-  goto nomem;
-
- sd = sysfs_new_dirent("_SHADOW_", inode->i_mode, SYSFS_DIR);
- if (!sd)
-  goto nomem;
```

```
- sd->s_elem.dir.kobj = kobj;
-
- sysfs_addrm_start(&acxt, parent_sd);
-
- /* add but don't link into children list */
- sysfs_add_one(&acxt, sd);
-
- /* attach and instantiate dentry */
- sysfs_attach_dentry(sd, shadow);
- d_instantiate(shadow, igrab(inode));
- inc_nlink(inode); /* tj: synchronization? */
-
- sysfs_addrm_finish(&acxt);
-
- dget(shadow);  /* Extra count - pin the dentry in core */
-
- goto out_dput;
-
- nomem:
- dput(shadow);
- sd = ERR_PTR(-ENOMEM);
- out_dput:
- dput(dir);
- out:
- return sd;
-}
-
-/**
- * sysfs_remove_shadow_dir - remove an object's directory.
- * @shadow_sd: sysfs_dirent of shadow directory
- *
- * The only thing special about this is that we remove any files in
- * the directory before we remove the directory, and we've inlined
- * what used to be sysfs_rmdir() below, instead of calling separately.
- */
-
-void sysfs_remove_shadow_dir(struct sysfs_dirent *shadow_sd)
-{
- __sysfs_remove_dir(shadow_sd);
-}
-
 const struct file_operations sysfs_dir_operations = {
  .open  = sysfs_dir_open,
  .release = sysfs_dir_close,
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"

diff --git a/fs/sysfs/inode.c b/fs/sysfs/inode.c
index 10d1b52..9671164 100644
--- a/fs/sysfs/inode.c
+++ b/fs/sysfs/inode.c
@@ -34,16 +34,6 @@ static const struct inode_operations sysfs_inode_operations ={
  .setattr = sysfs_setattr,
 };

-void sysfs_delete_inode(struct inode *inode)
-{
- /* Free the shadowed directory inode operations */
- if (sysfs_is_shadowed_inode(inode)) {
-  kfree(inode->i_op);
-  inode->i_op = NULL;
- }
- return generic_delete_inode(inode);
-}
-
 int sysfs_setattr(struct dentry * dentry, struct iattr * iattr)
 {
  struct inode * inode = dentry->d_inode;
diff --git a/fs/sysfs/mount.c b/fs/sysfs/mount.c
index 60714d0..e2073e6 100644
--- a/fs/sysfs/mount.c
+++ b/fs/sysfs/mount.c
@@ -21,7 +21,7 @@ struct kmem_cache *sysfs_dir_cachep;

 static const struct super_operations sysfs_ops = {
  .statfs  = simple_statfs,
- .drop_inode = sysfs_delete_inode,
+ .drop_inode = generic_delete_inode,
 };

 struct sysfs_dirent sysfs_root = {
diff --git a/fs/sysfs/sysfs.h b/fs/sysfs/sysfs.h
index 6b8c8d7..b55e510 100644
--- a/fs/sysfs/sysfs.h
+++ b/fs/sysfs/sysfs.h
@@ -70,7 +70,6 @@ extern void sysfs_remove_one(struct sysfs_addrm_cxt *acxt,
         struct sysfs_dirent *sd);
```

```
 extern int sysfs_addrm_finish(struct sysfs_addrm_cxt *acxt);

-extern void sysfs_delete_inode(struct inode *inode);
 extern struct inode * sysfs_get_inode(struct sysfs_dirent *sd);
 extern void sysfs_instantiate(struct dentry *dentry, struct inode *inode);

@@ -121,8 +120,3 @@ static inline void sysfs_put(struct sysfs_dirent * sd)
  if (sd && atomic_dec_and_test(&sd->s_count))
   release_sysfs_dirent(sd);
 }
-
-static inline int sysfs_is_shadowed_inode(struct inode *inode)
-{
- return S_ISDIR(inode->i_mode) && inode->i_op->follow_link;
-}
diff --git a/include/linux/kobject.h b/include/linux/kobject.h
index aa2fe22..7108a3e 100644
--- a/include/linux/kobject.h
+++ b/include/linux/kobject.h
@@ -80,14 +80,9 @@ extern void kobject_init(struct kobject *);
 extern void kobject_cleanup(struct kobject *);

 extern int __must_check kobject_add(struct kobject *);
-extern int __must_check kobject_shadow_add(struct kobject *kobj,
-       struct sysfs_dirent *shadow_parent);
 extern void kobject_del(struct kobject *);

 extern int __must_check kobject_rename(struct kobject *, const char *new_name);
-extern int __must_check kobject_shadow_rename(struct kobject *kobj,
-          struct sysfs_dirent *new_parent,
-          const char *new_name);
 extern int __must_check kobject_move(struct kobject *, struct kobject *);

 extern int __must_check kobject_register(struct kobject *);
diff --git a/include/linux/sysfs.h b/include/linux/sysfs.h
index be8228e..c16e4c5 100644
--- a/include/linux/sysfs.h
+++ b/include/linux/sysfs.h
@@ -17,9 +17,6 @@

 struct kobject;
 struct module;
-struct nameidata;
-struct dentry;
-struct sysfs_dirent;

 /* FIXME
  * The *owner field is no longer used, but leave around
```

```
@@ -94,14 +91,13 @@ extern int sysfs_schedule_callback(struct kobject *kobj,
   void (*func)(void *), void *data, struct module *owner);

 extern int __must_check
-sysfs_create_dir(struct kobject *kobj, struct sysfs_dirent *shadow_parent_sd);
+sysfs_create_dir(struct kobject *);

 extern void
 sysfs_remove_dir(struct kobject *);

 extern int __must_check
-sysfs_rename_dir(struct kobject *kobj, struct sysfs_dirent *new_parent_sd,
-   const char *new_name);
+sysfs_rename_dir(struct kobject *kobj, const char *new_name);

 extern int __must_check
 sysfs_move_dir(struct kobject *, struct kobject *);
@@ -138,12 +134,6 @@ void sysfs_remove_file_from_group(struct kobject *kobj,

 void sysfs_notify(struct kobject * k, char *dir, char *attr);

-
-extern int sysfs_make_shadowed_dir(struct kobject *kobj,
- void * (*follow_link)(struct dentry *, struct nameidata *));
-extern struct sysfs_dirent *sysfs_create_shadow_dir(struct kobject *kobj);
-extern void sysfs_remove_shadow_dir(struct sysfs_dirent *shadow_sd);
-
 extern int __must_check sysfs_init(void);

 #else /* CONFIG_SYSFS */
@@ -154,8 +144,7 @@ static inline int sysfs_schedule_callback(struct kobject *kobj,
  return -ENOSYS;
 }

-static inline int sysfs_create_dir(struct kobject *kobj,
-       struct sysfs_dirent *shadow_parent_sd)
+static inline int sysfs_create_dir(struct kobject * kobj)
 {
  return 0;
 }
@@ -165,9 +154,7 @@ static inline void sysfs_remove_dir(struct kobject * k)
  ;
 }

-static inline int sysfs_rename_dir(struct kobject *kobj,
-       struct sysfs_dirent *new_parent_sd,
-       const char *new_name)
+static inline int sysfs_rename_dir(struct kobject * kobj, const char *new_name)
```

```
 {
  return 0;
 }
@@ -242,12 +229,6 @@ static inline void sysfs_notify(struct kobject * k, char *dir, char *attr)
 {
 }

-static inline int sysfs_make_shadowed_dir(struct kobject *kobj,
- void * (*follow_link)(struct dentry *, struct nameidata *))
-{
- return 0;
-}
-
 static inline int __must_check sysfs_init(void)
 {
  return 0;
diff --git a/lib/kobject.c b/lib/kobject.c
index 4b08e0f..2fc9fc6 100644
--- a/lib/kobject.c
+++ b/lib/kobject.c
@@ -44,11 +44,11 @@ static int populate_dir(struct kobject * kobj)
  return error;
 }

-static int create_dir(struct kobject *kobj, struct sysfs_dirent *shadow_parent)
+static int create_dir(struct kobject * kobj)
 {
  int error = 0;
  if (kobject_name(kobj)) {
-  error = sysfs_create_dir(kobj, shadow_parent);
+  error = sysfs_create_dir(kobj);
   if (!error) {
    if ((error = populate_dir(kobj)))
     sysfs_remove_dir(kobj);
@@ -157,12 +157,11 @@ static void unlink(struct kobject * kobj)
 }

 /**
- * kobject_shadow_add - add an object to the hierarchy.
+ * kobject_add - add an object to the hierarchy.
  * @kobj: object.
- * @shadow_parent: sysfs directory to add to.
  */

-int kobject_shadow_add(struct kobject *kobj, struct sysfs_dirent *shadow_parent)
+int kobject_add(struct kobject * kobj)
 {
  int error = 0;
```

```
  struct kobject * parent;
@@ -194,7 +193,7 @@ int kobject_shadow_add(struct kobject *kobj, struct sysfs_dirent
*shadow_parent)
   kobj->parent = parent;
  }

- error = create_dir(kobj, shadow_parent);
+ error = create_dir(kobj);
  if (error) {
   /* unlink does the kobject_put() for us */
   unlink(kobj);
@@ -216,16 +215,6 @@ int kobject_shadow_add(struct kobject *kobj, struct sysfs_dirent
*shadow_parent)
 }

 /**
- * kobject_add - add an object to the hierarchy.
- * @kobj: object.
- */
-int kobject_add(struct kobject * kobj)
-{
- return kobject_shadow_add(kobj, NULL);
-}
-
-
-/**
  * kobject_register - initialize and add an object.
  * @kobj: object in question.
  */
@@ -338,7 +327,7 @@ int kobject_rename(struct kobject * kobj, const char *new_name)
  /* Note : if we want to send the new name alone, not the full path,
   * we could probably use kobject_name(kobj); */

- error = sysfs_rename_dir(kobj, kobj->parent->sd, new_name);
+ error = sysfs_rename_dir(kobj, new_name);

  /* This function is mostly/only used for network interface.
   * Some hotplug package track interfaces by their name and
@@ -355,27 +344,6 @@ out:
 }

 /**
- * kobject_rename - change the name of an object
- * @kobj: object in question.
- * @new_parent: object's new parent
- * @new_name: object's new name
- */
-
```

```
-int kobject_shadow_rename(struct kobject *kobj,
-     struct sysfs_dirent *new_parent, const char *new_name)
-{
- int error = 0;
-
- kobj = kobject_get(kobj);
- if (!kobj)
-  return -EINVAL;
- error = sysfs_rename_dir(kobj, new_parent, new_name);
- kobject_put(kobj);
-
- return error;
-}
-
-/**
  * kobject_move - move object to another parent
  * @kobj: object in question.
  * @new_parent: object's new parent (can be NULL)
--
1.5.1.1.181.g2de0
```

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