

---

Subject: [PATCH 2/5] sysfs: Remove first pass at shadow directory support

Posted by [ebiederm](#) on Fri, 06 Apr 2007 16:48:42 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](mailto:ebiederm@xmission.com)>

```
---
fs/sysfs/dir.c      | 196 ++++++-----
fs/sysfs/group.c   |  1 -
fs/sysfs/inode.c   | 10 ---
fs/sysfs/mount.c   |  2 +-
fs/sysfs/sysfs.h   |  5 -
include/linux/kobject.h |  4 -
include/linux/sysfs.h | 24 +-----
lib/kobject.c      | 43 +-----
8 files changed, 45 insertions(+), 240 deletions(-)
```

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

```
index 85a6686..d6ab015 100644
```

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

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

```
@@ -33,7 +33,8 @@ static struct dentry_operations sysfs_dentry_ops = {
/*
```

```
 * Allocates a new sysfs_dirent and links it to the parent sysfs_dirent
*/
```

```
-static struct sysfs_dirent * __sysfs_new_dirent(void * element)
```

```
+static struct sysfs_dirent * sysfs_new_dirent(struct sysfs_dirent * parent_sd,
+ void * element)
```

```
{
 struct sysfs_dirent * sd;
```

```
@@ -44,28 +45,12 @@ static struct sysfs_dirent * __sysfs_new_dirent(void * element)
```

```
 atomic_set(&sd->s_count, 1);
```

```
 atomic_set(&sd->s_event, 1);
```

```
 INIT_LIST_HEAD(&sd->s_children);
```

```
- INIT_LIST_HEAD(&sd->s_sibling);
```

```
+ list_add(&sd->s_sibling, &parent_sd->s_children);
```

```
 sd->s_element = element;
```

```

return sd;
}

-static void __sysfs_list_dirent(struct sysfs_dirent *parent_sd,
- struct sysfs_dirent *sd)
- {
- if (sd)
- list_add(&sd->s_sibling, &parent_sd->s_children);
- }
-
-static struct sysfs_dirent * sysfs_new_dirent(struct sysfs_dirent *parent_sd,
- void * element)
- {
- struct sysfs_dirent *sd;
- sd = __sysfs_new_dirent(element);
- __sysfs_list_dirent(parent_sd, sd);
- return sd;
- }
-
/*
*
* Return -EEXIST if there is already a sysfs element with the same name for
@@ -92,14 +77,14 @@ int sysfs_dirent_exist(struct sysfs_dirent *parent_sd,
}

-static struct sysfs_dirent *
-__sysfs_make_dirent(struct dentry *dentry, void *element, mode_t mode, int type)
+int sysfs_make_dirent(struct sysfs_dirent * parent_sd, struct dentry * dentry,
+ void * element, umode_t mode, int type)
{
struct sysfs_dirent * sd;

- sd = __sysfs_new_dirent(element);
+ sd = sysfs_new_dirent(parent_sd, element);
if (!sd)
- goto out;
+ return -ENOMEM;

sd->s_mode = mode;
sd->s_type = type;
@@ -109,19 +94,7 @@ __sysfs_make_dirent(struct dentry *dentry, void *element, mode_t mode,
int type)
dentry->d_op = &sysfs_dentry_ops;
}

-out:

```

```

- return sd;
-}
-
-int sysfs_make_dirent(struct sysfs_dirent * parent_sd, struct dentry * dentry,
- void * element, umode_t mode, int type)
- {
- struct sysfs_dirent *sd;
-
- sd = __sysfs_make_dirent(dentry, element, mode, type);
- __sysfs_list_dirent(parent_sd, sd);
-
- return sd ? 0 : -ENOMEM;
+ return 0;
}

static int init_dir(struct inode * inode)
@@ -193,10 +166,9 @@ int sysfs_create_subdir(struct kobject * k, const char * n, struct dentry **
d)
/**
 * sysfs_create_dir - create a directory for an object.
 * @kobj: object we're creating directory for.
- * @shadow_parent: parent parent object.
 */

-int sysfs_create_dir(struct kobject * kobj, struct dentry *shadow_parent)
+int sysfs_create_dir(struct kobject * kobj)
{
struct dentry * dentry = NULL;
struct dentry * parent;
@@ -204,9 +176,7 @@ int sysfs_create_dir(struct kobject * kobj, struct dentry *shadow_parent)

BUG_ON(!kobj);

- if (shadow_parent)
- parent = shadow_parent;
- else if (kobj->parent)
+ if (kobj->parent)
parent = kobj->parent->dentry;
else if (sysfs_mount && sysfs_mount->mnt_sb)
parent = sysfs_mount->mnt_sb->s_root;
@@ -327,12 +297,21 @@ void sysfs_remove_subdir(struct dentry * d)
}

-static void __sysfs_remove_dir(struct dentry *dentry)
+/**
+ * sysfs_remove_dir - remove an object's directory.
+ * @kobj: object.

```

```

+ *
+ * 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_dir(struct kobject * kobj)
{
+ struct dentry * dentry = dget(kobj->dentry);
  struct sysfs_dirent * parent_sd;
  struct sysfs_dirent * sd, * tmp;

- dget(dentry);
  if (!dentry)
    return;

@@ -353,46 +332,28 @@ static void __sysfs_remove_dir(struct dentry *dentry)
  * Drop reference from dget() on entrance.
  */
  dput(dentry);
-}
-
-/**
- * sysfs_remove_dir - remove an object's directory.
- * @kobj: object.
- *
- * 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_dir(struct kobject * kobj)
-{
- __sysfs_remove_dir(kobj->dentry);
  kobj->dentry = NULL;
}

-int sysfs_rename_dir(struct kobject * kobj, struct dentry *new_parent,
- const char *new_name)
+int sysfs_rename_dir(struct kobject * kobj, const char *new_name)
{
  int error = 0;
- struct dentry * new_dentry;
+ struct inode * inode;
+ struct dentry * new_dentry, * old_parent, * new_parent;

- if (!new_parent)
- return -EFAULT;

```

```

+ if (!kobj->parent)
+ return -EINVAL;
+
+ old_parent = dget(kobj->dentry->d_parent);
+ inode = old_parent->d_inode;

    down_write(&sysfs_rename_sem);
- mutex_lock(&new_parent->d_inode->i_mutex);
+ mutex_lock(&inode->i_mutex);

+ new_parent = kobj->dentry->d_parent;
  new_dentry = lookup_one_len(new_name, new_parent, strlen(new_name));
  if (!IS_ERR(new_dentry)) {
- /* By allowing two different directories with the
- * same d_parent we allow this routine to move
- * between different shadows of the same directory
- */
- if (kobj->dentry->d_parent->d_inode != new_parent->d_inode)
- return -EINVAL;
- else if (new_dentry->d_parent->d_inode != new_parent->d_inode)
- error = -EINVAL;
- else if (new_dentry == kobj->dentry)
+ if (new_dentry == kobj->dentry)
    error = -EINVAL;
    else if (!new_dentry->d_inode) {
        error = kobject_set_name(kobj, "%s", new_name);
@@ -414,9 +375,11 @@ int sysfs_rename_dir(struct kobject * kobj, struct dentry *new_parent,
        error = -EEXIST;
        dput(new_dentry);
    }
- mutex_unlock(&new_parent->d_inode->i_mutex);
+ mutex_unlock(&inode->i_mutex);
    up_write(&sysfs_rename_sem);

+ dput(old_parent);
+
    return error;
}

@@ -595,95 +558,6 @@ 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 inode *inode;
- struct inode_operations *i_op;
-
- inode = kobj->dentry->d_inode;
- if (inode->i_op != &sysfs_dir_inode_operations)
- 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 dentry *sysfs_create_shadow_dir(struct kobject *kobj)
- {
- struct sysfs_dirent *sd;
- struct dentry *parent, *dir, *shadow;
- struct inode *inode;
-
- dir = kobj->dentry;
- inode = dir->d_inode;
- parent = dir->d_parent;
- shadow = ERR_PTR(-EINVAL);
- if (!sysfs_is_shadowed_inode(inode))
- goto out;
-
- shadow = d_alloc(parent, &dir->d_name);

```

```

- if (!shadow)
- goto nomem;
-
- sd = __sysfs_make_dirent(shadow, kobj, inode->i_mode, SYSFS_DIR);
- if (!sd)
- goto nomem;
-
- d_instantiate(shadow, igrab(inode));
- inc_nlink(inode);
- inc_nlink(parent->d_inode);
- shadow->d_op = &sysfs_dentry_ops;
-
- dget(shadow); /* Extra count - pin the dentry in core */
-
-out:
- return shadow;
-nomem:
- dput(shadow);
- shadow = ERR_PTR(-ENOMEM);
- goto out;
-}
-
-/**
- * sysfs_remove_shadow_dir - remove an object's directory.
- * @shadow: dentry 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 dentry *shadow)
-{
- __sysfs_remove_dir(shadow);
-}
-
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 b20951c..46a277b 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 4de5c6b..69e6e9b 100644
--- a/fs/sysfs/inode.c
+++ b/fs/sysfs/inode.c
@@ -33,16 +33,6 @@ 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 23a48a3..7978cf2 100644
--- a/fs/sysfs/mount.c
+++ b/fs/sysfs/mount.c
@@ -23,7 +23,7 @@ static void sysfs_clear_inode(struct inode *inode);

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

diff --git a/fs/sysfs/sysfs.h b/fs/sysfs/sysfs.h
index a77c57e..ce6b3ec 100644
--- a/fs/sysfs/sysfs.h
+++ b/fs/sysfs/sysfs.h
@@ -13,7 +13,6 @@ struct sysfs_dirent {
extern struct vfsmount * sysfs_mount;
extern struct kmem_cache *sysfs_dir_cache;

-extern void sysfs_delete_inode(struct inode *inode);
extern struct inode * sysfs_new_inode(mode_t mode, struct sysfs_dirent *);
extern int sysfs_create(struct dentry *, int mode, int (*init)(struct inode *));

```



```

@@ -124,7 +123,3 @@ static inline void sysfs_put(struct sysfs_dirent * sd)
    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 d37cd7f..c96d243 100644
--- a/include/linux/kobject.h
+++ b/include/linux/kobject.h
@@ -74,13 +74,9 @@ 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 *, struct dentry *);
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 dentry *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 fea9a6b..1f1bb45 100644
--- a/include/linux/sysfs.h
+++ b/include/linux/sysfs.h
@@ -17,8 +17,6 @@

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

struct attribute {
    const char * name;
@@ -83,13 +81,13 @@ extern int sysfs_schedule_callback(struct kobject *kobj,
    void (*func)(void *), void *data);

extern int __must_check
-sysfs_create_dir(struct kobject *, struct dentry *);
+sysfs_create_dir(struct kobject *);

extern void

```

```

sysfs_remove_dir(struct kobject *);

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

extern int __must_check
sysfs_move_dir(struct kobject *, struct kobject *);
@@ -126,12 +124,6 @@ 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 dentry *sysfs_create_shadow_dir(struct kobject *kobj);
-extern void sysfs_remove_shadow_dir(struct dentry *dir);
-
extern int __must_check sysfs_init(void);

#else /* CONFIG_SYSFS */
@@ -142,7 +134,7 @@ static inline int sysfs_schedule_callback(struct kobject *kobj,
return -ENOSYS;
}

-static inline int sysfs_create_dir(struct kobject * k, struct dentry *shadow)
+static inline int sysfs_create_dir(struct kobject * k)
{
return 0;
}
@@ -152,9 +144,7 @@ static inline void sysfs_remove_dir(struct kobject * k)
;
}

-static inline int sysfs_rename_dir(struct kobject * k,
- struct dentry *new_parent,
- const char *new_name)
+static inline int sysfs_rename_dir(struct kobject * k, const char *new_name)
{
return 0;
}
@@ -229,12 +219,6 @@ 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 09cb276..f590ad8 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 dentry *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);
@@ -159,10 +159,9 @@ static void unlink(struct kobject * kobj)
/**
 * 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 dentry *shadow_parent)
+int kobject_add(struct kobject * kobj)
{
    int error = 0;
    struct kobject * parent;
@@ -193,7 +192,7 @@ int kobject_shadow_add(struct kobject * kobj, struct dentry
*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);
@@ -214,15 +213,6 @@ int kobject_shadow_add(struct kobject * kobj, struct dentry
*shadow_parent)

```

```

    return error;
}

-/**
- * 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.
@@ -315,30 +305,7 @@ int kobject_rename(struct kobject * kobj, const char *new_name)
    kobj = kobject_get(kobj);
    if (!kobj)
        return -EINVAL;
- if (!kobj->parent)
- return -EINVAL;
- error = sysfs_rename_dir(kobj, kobj->parent->dentry, new_name);
- kobject_put(kobj);
-
- return error;
- }
-
-/**
- * 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 dentry *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);
+ error = sysfs_rename_dir(kobj, new_name);
    kobject_put(kobj);

    return error;
--

```

1.5.0.g53756

---

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>

---