
Subject: Re: [PATCH 1/4] sysfs: Remove first pass at shadow directory support
Posted by [Benjamin Thery](#) on Fri, 22 Jun 2007 14:42:01 GMT
[View Forum Message](#) <> [Reply to Message](#)

Eric,

Thanks a lot for the updated patchset.

I haven't look at the details yet, but I've already integrated it to your network namespaces patchset and ported the whole thing to 2.6.22-rc4-mm2 today.

It is far easier to do when you don't have to think about the sysfs stuff :)

I only performed some basic testing on it so far as I'm home today, but it already seems to work better than the patchset I ported to 2.6.21-mm2. It solves the issues I still had with sysfs.

Thanks again.

Regards,
Benjamin

Eric W. Biederman wrote:

> 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>

> ---
> These patches are against 2.6.22-rc4-mm2 Hopefully that is new enough
> to catch all of the in flight sysfs patches.

>
> fs/sysfs/dir.c | 172 ++++++-----
> fs/sysfs/group.c | 1 -
> fs/sysfs/inode.c | 10 ---
> fs/sysfs/mount.c | 2 +-
> fs/sysfs/sysfs.h | 6 --
> include/linux/kobject.h | 4 -

```

> include/linux/sysfs.h | 24 +-----
> lib/kobject.c | 44 ++-----
> 8 files changed, 48 insertions(+), 215 deletions(-)
>
> diff --git a/fs/sysfs/dir.c b/fs/sysfs/dir.c
> index b4074ad..b1da4fc 100644
> --- a/fs/sysfs/dir.c
> +++ b/fs/sysfs/dir.c
> @@ -415,10 +415,9 @@ 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;
> @@ -426,9 +425,7 @@ 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;
> @@ -516,12 +513,26 @@ 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 = kobj->dentry;
> struct sysfs_dirent *removed = NULL;
> struct sysfs_dirent *parent_sd;
> struct sysfs_dirent **pos;
>
> + spin_lock(&kobj_sysfs_assoc_lock);
> + kobj->dentry = NULL;
> + spin_unlock(&kobj_sysfs_assoc_lock);
> +
> if (!dentry)
> return;
>
> @@ -555,55 +566,35 @@ static void __sysfs_remove_dir(struct dentry *dentry)
> remove_dir(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 *d = kobj->dentry;
> -
> - spin_lock(&kobj_sysfs_assoc_lock);
> - kobj->dentry = NULL;
> - spin_unlock(&kobj_sysfs_assoc_lock);
> -
> - __sysfs_remove_dir(d);
> -}
> -
> -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)
> {
> - struct sysfs_dirent *sd = kobj->dentry->d_fsdata;
> - struct sysfs_dirent *parent_sd = new_parent->d_fsdata;
> - struct dentry *new_dentry;
> + struct sysfs_dirent *sd;
> + struct sysfs_dirent *parent_sd;
> + struct inode *inode;
> + struct dentry *new_dentry, *parent;

```

```

> char *dup_name;
> int error;
>
> - if (!new_parent)
> - return -EFAULT;
> + if (!kobj->parent)
> + return -EINVAL;
> +
> + parent = dget(kobj->dentry->d_parent);
> + inode = parent->d_inode;
> +
> + sd = kobj->dentry->d_fsdata;
>
> down_write(&sysfs_rename_sem);
> - mutex_lock(&new_parent->d_inode->i_mutex);
> + mutex_lock(&inode->i_mutex);
>
> - new_dentry = lookup_one_len(new_name, new_parent, strlen(new_name));
> + parent_sd = parent->d_fsdata;
> + new_dentry = lookup_one_len(new_name, parent_sd->s_dentry, 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 (kobj->dentry->d_parent->d_inode != new_parent->d_inode ||
> -     new_dentry->d_parent->d_inode != new_parent->d_inode ||
> -     new_dentry == kobj->dentry)
> + if (new_dentry == kobj->dentry)
>     goto out_dput;
>
> error = -EEXIST;
> @@ -643,8 +634,11 @@ int sysfs_rename_dir(struct kobject * kobj, struct dentry *new_parent,
> out_dput:
> dput(new_dentry);
> out_unlock:
> - mutex_unlock(&new_parent->d_inode->i_mutex);
> + mutex_unlock(&inode->i_mutex);
> up_write(&sysfs_rename_sem);
> +
> + dput(parent);
> +
> return error;

```

```

> }
>
> @@ -837,98 +831,6 @@ 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 dentry *dir = kobj->dentry;

```

```

> - struct inode *inode = dir->d_inode;
> - struct dentry *parent = dir->d_parent;
> - struct sysfs_dirent *parent_sd = parent->d_fsdata;
> - struct dentry *shadow;
> - struct sysfs_dirent *sd;
> -
> - 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_new_dirent("_SHADOW_", inode->i_mode, SYSFS_DIR);
> - if (!sd)
> - goto nomem;
> - sd->s_elem.dir.kobj = kobj;
> - /* point to parent_sd but don't attach to it */
> - sd->s_parent = sysfs_get(parent_sd);
> - sysfs_attach_dirent(sd, NULL, shadow);
> -
> - d_instantiate(shadow, igrab(inode));
> - inc_nlink(inode);
> - inc_nlink(parent->d_inode);
> -
> - 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 52eed2a..82e0f55 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 b1a4527..8181c49 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 4be9593..c8126ab 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,

```

```

> };
>
> static struct sysfs_dirent sysfs_root = {
> diff --git a/fs/sysfs/sysfs.h b/fs/sysfs/sysfs.h
> index 6f8aaf3..6258462 100644
> --- a/fs/sysfs/sysfs.h
> +++ b/fs/sysfs/sysfs.h
> @@ -53,7 +53,6 @@ extern struct sysfs_dirent *sysfs_get_active_two(struct sysfs_dirent *sd);
> extern void sysfs_put_active_two(struct sysfs_dirent *sd);
> extern void sysfs_deactivate(struct sysfs_dirent *sd);
>
> -extern void sysfs_delete_inode(struct inode *inode);
> extern void sysfs_init_inode(struct sysfs_dirent *sd, struct inode *inode);
> extern struct inode * sysfs_get_inode(struct sysfs_dirent *sd);
> extern void sysfs_instantiate(struct dentry *dentry, struct inode *inode);
> @@ -100,8 +99,3 @@ 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 c288e41..4fa5b88 100644
> --- a/include/linux/kobject.h
> +++ b/include/linux/kobject.h
> @@ -71,13 +71,9 @@ 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 161e19a..a3fa5b9 100644
> --- a/include/linux/sysfs.h
> +++ b/include/linux/sysfs.h
> @@ -17,8 +17,6 @@
>

```



```

> struct kobject;
> struct module;
> -struct nameidata;
> -struct dentry;
>
> /* FIXME
>  * The *owner field is no longer used, but leave around
> @@ -88,13 +86,13 @@ 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 *, 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 *);
> @@ -131,12 +129,6 @@ 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 */
> @@ -147,7 +139,7 @@ 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;
> }
> @@ -157,9 +149,7 @@ 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;
> }
> @@ -234,12 +224,6 @@ 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 cb40a00..738d76f 100644
> --- a/lib/kobject.c
> +++ b/lib/kobject.c
> @@ -46,11 +46,11 @@ 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);
> @@ -201,12 +201,11 @@ 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 dentry *shadow_parent)
> +int kobject_add(struct kobject * kobj)
> {
>     int error = 0;
>     struct kobject * parent;
> @@ -238,7 +237,7 @@ 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);
> @@ -260,16 +259,6 @@ int kobject_shadow_add(struct kobject * kobj, struct dentry
> *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.
>  */
> @@ -382,7 +371,7 @@ 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->dentry, 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
> @@ -399,27 +388,6 @@ 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 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);
> - 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)

```

Containers mailing list

Containers@lists.linux-foundation.org

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