Subject: [PATCH 1/2] sysfs: Shadow directory support
Posted by ebiederm on Wed, 24 Jan 2007 19:35:52 GMT
View Forum Message <> Reply to Message

The problem. When implementing a network namespace I need to be able to have multiple network devices with the same name. Currently this is a problem for /sys/class/net/*.

What I want is a separate /sys/class/net directory in sysfs for each network namespace, and I want to name each of them /sys/class/net.

I looked and the VFS actually allows that. All that is needed is for /sys/class/net to implement a follow link method to redirect lookups to the real directory you want.

Implementing a follow link method that is sensitive to the current network namespace turns out to be 3 lines of code so it looks like a clean approach. Modifying sysfs so it doesn't get in my was is a bit trickier.

I am calling the concept of multiple directories all at the same path in the filesystem shadow directories. With the directory entry really at that location the shadow master.

The following patch modifies sysfs so it can handle a directory structure slightly different from the kobject tree so I can implement the shadow directories for handling /sys/class/net/.

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

```
 fs/sysfs/dir.c          |  207 ++++++++++++++++++++++++++++++++++++++++--------
 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   |   23 +++++-
 lib/kobject.c           |   42 +++++++++-
 8 files changed, 249 insertions(+), 45 deletions(-)

diff --git a/fs/sysfs/dir.c b/fs/sysfs/dir.c
index 511edef..dd23903 100644
--- a/fs/sysfs/dir.c
+++ b/fs/sysfs/dir.c
@@ -32,8 +32,7 @@ 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(struct sysfs_dirent * parent_sd,
-      void * element)
+static struct sysfs_dirent * __sysfs_new_dirent(void * element)
 {
  struct sysfs_dirent * sd;

@@ -45,12 +44,28 @@ static struct sysfs_dirent * sysfs_new_dirent(struct sysfs_dirent *
parent_sd,
  atomic_set(&sd->s_count, 1);
  atomic_set(&sd->s_event, 1);
  INIT_LIST_HEAD(&sd->s_children);
- list_add(&sd->s_sibling, &parent_sd->s_children);
+ INIT_LIST_HEAD(&sd->s_sibling);
  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
@@ -77,14 +92,14 @@ int sysfs_dirent_exist(struct sysfs_dirent *parent_sd,
 }


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

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

  sd->s_mode = mode;
  sd->s_type = type;
@@ -94,7 +109,19 @@ int sysfs_make_dirent(struct sysfs_dirent * parent_sd, struct dentry *
dentry,
   dentry->d_op = &sysfs_dentry_ops;
  }

- return 0;
+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;
 }

 static int init_dir(struct inode * inode)
@@ -165,11 +192,11 @@ int sysfs_create_subdir(struct kobject * k, const char * n, struct dentry
** d)

 /**
  * sysfs_create_dir - create a directory for an object.
- * @parent: parent parent object.
  * @kobj:  object we're creating directory for.
+ * @shadow_parent: parent parent object.
  */

-int sysfs_create_dir(struct kobject * kobj)
+int sysfs_create_dir(struct kobject * kobj, struct dentry *shadow_parent)
 {
  struct dentry * dentry = NULL;
  struct dentry * parent;
@@ -177,7 +204,9 @@ int sysfs_create_dir(struct kobject * kobj)

  BUG_ON(!kobj);
```

```
- if (kobj->parent)
+ if (shadow_parent)
+  parent = shadow_parent;
+ else if (kobj->parent)
   parent = kobj->parent->dentry;
  else if (sysfs_mount && sysfs_mount->mnt_sb)
   parent = sysfs_mount->mnt_sb->s_root;
@@ -298,21 +327,12 @@ void sysfs_remove_subdir(struct dentry * d)
 }


-/**
- * 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)
+static void __sysfs_remove_dir(struct dentry *dentry)
 {
- struct dentry * dentry = dget(kobj->dentry);
  struct sysfs_dirent * parent_sd;
  struct sysfs_dirent * sd, * tmp;

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

@@ -333,32 +353,60 @@ void sysfs_remove_dir(struct kobject * kobj)
   * 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, const char *new_name)
+int sysfs_rename_dir(struct kobject * kobj, struct dentry *new_parent,
+       const char *new_name)
 {
  int error = 0;
- struct dentry * new_dentry, * parent;
-
- if (!strcmp(kobject_name(kobj), new_name))
-  return -EINVAL;
+ struct dentry * new_dentry;

- if (!kobj->parent)
-  return -EINVAL;
+ if (!new_parent)
+  return -EFAULT;

  down_write(&sysfs_rename_sem);
- parent = kobj->parent->dentry;
-
- mutex_lock(&parent->d_inode->i_mutex);
+ mutex_lock(&new_parent->d_inode->i_mutex);

- new_dentry = lookup_one_len(new_name, parent, strlen(new_name));
+ new_dentry = lookup_one_len(new_name, new_parent, strlen(new_name));
  if (!IS_ERR(new_dentry)) {
-    if (!new_dentry->d_inode) {
+  /* 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)
+   error = -EINVAL;
+  else if (!new_dentry->d_inode) {
    error = kobject_set_name(kobj, "%s", new_name);
    if (!error) {
+    struct sysfs_dirent *sd, *parent_sd;
+
     d_add(new_dentry, NULL);
     d_move(kobj->dentry, new_dentry);
```

```
+
+    sd = kobj->dentry->d_fsdata;
+    parent_sd = new_parent->d_fsdata;
+
+    list_del_init(&sd->s_sibling);
+    list_add(&sd->s_sibling, &parent_sd->s_children);
    }
    else
     d_drop(new_dentry);
@@ -366,7 +414,7 @@ int sysfs_rename_dir(struct kobject * kobj, const char *new_name)
    error = -EEXIST;
   dput(new_dentry);
  }
- mutex_unlock(&parent->d_inode->i_mutex);
+ mutex_unlock(&new_parent->d_inode->i_mutex);
  up_write(&sysfs_rename_sem);

  return error;
@@ -547,6 +595,95 @@ 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 122145b..050d23d 100644
--- a/fs/sysfs/group.c
+++ b/fs/sysfs/group.c
@@ -13,6 +13,7 @@
 #include <linux/dcache.h>
 #include <linux/namei.h>
 #include <linux/err.h>
+#include <linux/fs.h>
 #include "sysfs.h"


diff --git a/fs/sysfs/inode.c b/fs/sysfs/inode.c
index e79e38d..b94b9d3 100644
--- a/fs/sysfs/inode.c
+++ b/fs/sysfs/inode.c
@@ -32,6 +32,16 @@ static 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 e503f85..8be465b 100644
--- a/fs/sysfs/mount.c
+++ b/fs/sysfs/mount.c
@@ -20,7 +20,7 @@ struct kmem_cache *sysfs_dir_cachep;

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

 static struct sysfs_dirent sysfs_root = {
diff --git a/fs/sysfs/sysfs.h b/fs/sysfs/sysfs.h
index bd7cec2..908f05b 100644
--- a/fs/sysfs/sysfs.h
+++ b/fs/sysfs/sysfs.h
@@ -2,6 +2,7 @@
 extern struct vfsmount * sysfs_mount;
 extern struct kmem_cache *sysfs_dir_cachep;

+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 *));

@@ -96,3 +97,7 @@ 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 76538fc..b850e03 100644
--- a/include/linux/kobject.h
+++ b/include/linux/kobject.h
@@ -74,9 +74,13 @@ 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 2129d1b..6573a5d 100644
--- a/include/linux/sysfs.h
+++ b/include/linux/sysfs.h
@@ -15,6 +15,7 @@

 struct kobject;
 struct module;
+struct nameidata;

 struct attribute {
  const char  * name;
@@ -88,13 +89,13 @@ struct sysfs_dirent {
 #ifdef CONFIG_SYSFS

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

 extern void
 sysfs_remove_dir(struct kobject *);

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

 extern int __must_check
 sysfs_move_dir(struct kobject *, struct kobject *);
@@ -126,11 +127,17 @@ int __must_check sysfs_create_group(struct kobject *,
 void sysfs_remove_group(struct kobject *, const struct attribute_group *);
 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 */
```

```
-static inline int sysfs_create_dir(struct kobject * k)
+static inline int sysfs_create_dir(struct kobject * k, struct dentry *shadow)
 {
  return 0;
 }
@@ -140,7 +147,9 @@ static inline void sysfs_remove_dir(struct kobject * k)
  ;
 }

-static inline int sysfs_rename_dir(struct kobject * k, const char *new_name)
+static inline int sysfs_rename_dir(struct kobject * k,
+     struct dentry *new_parent,
+     const char *new_name)
 {
  return 0;
 }
@@ -204,6 +213,12 @@ 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 7ce6dc1..a260aea 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)
+static int create_dir(struct kobject * kobj, struct dentry *shadow_parent)
 {
  int error = 0;
  if (kobject_name(kobj)) {
-  error = sysfs_create_dir(kobj);
+  error = sysfs_create_dir(kobj, shadow_parent);
   if (!error) {
    if ((error = populate_dir(kobj)))
     sysfs_remove_dir(kobj);
@@ -156,9 +156,10 @@ 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_add(struct kobject * kobj)
+int kobject_shadow_add(struct kobject * kobj, struct dentry *shadow_parent)
 {
  int error = 0;
  struct kobject * parent;
@@ -189,7 +190,7 @@ int kobject_add(struct kobject * kobj)
  }
  kobj->parent = parent;

- error = create_dir(kobj);
+ error = create_dir(kobj, shadow_parent);
  if (error) {
   /* unlink does the kobject_put() for us */
   unlink(kobj);
@@ -211,6 +212,15 @@ int kobject_add(struct kobject * kobj)
  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.
@@ -303,7 +313,29 @@ int kobject_rename(struct kobject * kobj, const char *new_name)
  kobj = kobject_get(kobj);
  if (!kobj)
   return -EINVAL;
- error = sysfs_rename_dir(kobj, new_name);
+ 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_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;
--
1.4.4.1.g278f
```

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: [PATCH 2/2] Implement shadow directory support for device classes.
Posted by ebiederm on Wed, 24 Jan 2007 19:37:19 GMT
View Forum Message <> Reply to Message

Modify the device class code so that normal manipulations work
in the presence of shadow directories. Some of the shadow directory
support still needs to be implemented in the implementation of the
class but these modifications are sufficient to make that simple.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 drivers/base/class.c | 34 ++++++++++++++++++++++++++++++++--
 include/linux/device.h | 4 ++++
 2 files changed, 36 insertions(+), 2 deletions(-)

diff --git a/drivers/base/class.c b/drivers/base/class.c
index 8bf2ca2..f72ddc0 100644
--- a/drivers/base/class.c
+++ b/drivers/base/class.c
@@ -136,6 +136,18 @@ static void remove_class_attrs(struct class * cls)

```
 }
}

+static int class_setup_shadow(struct class *cls)
+{
+ if (!cls->class_device_dparent && !cls->class_follow_link)
+  return 0;
+
+ if (!cls->class_device_dparent || !cls->class_device_dparent)
+  return -EINVAL;
+
+ return sysfs_make_shadowed_dir(&cls->subsys.kset.kobj,
+      cls->class_follow_link);
+}
+
 int class_register(struct class * cls)
 {
  int error;
@@ -154,6 +166,11 @@ int class_register(struct class * cls)

  error = subsystem_register(&cls->subsys);
  if (!error) {
+  error = class_setup_shadow(cls);
+  if (error)
+   subsystem_unregister(&cls->subsys);
+ }
+ if (!error) {
   error = add_class_attrs(class_get(cls));
   class_put(cls);
  }
@@ -582,6 +599,7 @@ int class_device_add(struct class_device *class_dev)
  struct class *parent_class = NULL;
  struct class_device *parent_class_dev = NULL;
  struct class_interface *class_intf;
+ struct dentry *dparent;
  int error = -EINVAL;

  class_dev = class_device_get(class_dev);
@@ -610,7 +628,11 @@ int class_device_add(struct class_device *class_dev)
  else
   class_dev->kobj.parent = &parent_class->subsys.kset.kobj;

- error = kobject_add(&class_dev->kobj);
+ if (parent_class->class_device_dparent)
+  dparent = parent_class->class_device_dparent(class_dev);
+ else
+  dparent = parent_class->subsys.kset.kobj.dentry;
+ error = kobject_shadow_add(&class_dev->kobj, dparent);
```

```
  if (error)
   goto out2;

@@ -841,11 +863,15 @@ int class_device_rename(struct class_device *class_dev, char
*new_name)
 {
  int error = 0;
  char *old_class_name = NULL, *new_class_name = NULL;
+ struct class *class;
+ struct dentry *new_parent;

  class_dev = class_device_get(class_dev);
  if (!class_dev)
   return -EINVAL;

+ class = class_dev->class;
+
  pr_debug("CLASS: renaming '%s' to '%s'\n", class_dev->class_id,
    new_name);

@@ -857,7 +883,11 @@ int class_device_rename(struct class_device *class_dev, char
*new_name)

  strlcpy(class_dev->class_id, new_name, KOBJ_NAME_LEN);

- error = kobject_rename(&class_dev->kobj, new_name);
+ if (class->class_device_dparent)
+  new_parent = class->class_device_dparent(class_dev);
+ else
+  new_parent = class->subsys.kset.kobj.dentry;
+ error = kobject_shadow_rename(&class_dev->kobj, new_parent, new_name);

 #ifdef CONFIG_SYSFS_DEPRECATED
  if (class_dev->dev) {
diff --git a/include/linux/device.h b/include/linux/device.h
index f44247f..162e840 100644
--- a/include/linux/device.h
+++ b/include/linux/device.h
@@ -33,6 +33,7 @@ struct device;
 struct device_driver;
 struct class;
 struct class_device;
+struct nameidata;

 struct bus_type {
  const char  * name;
@@ -197,6 +198,9 @@ struct class {
```

```
  int (*suspend)(struct device *, pm_message_t state);
  int (*resume)(struct device *);
+
+ struct dentry *(*class_device_dparent)(struct class_device *);
+ void *(*class_follow_link)(struct dentry *dentry, struct nameidata *nd);
 };

 extern int __must_check class_register(struct class *);
--
1.4.4.1.g278f
```

_____

Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers