Subject: [PATCH 10/15] sysfs: Merge sysfs_rename_dir and sysfs_move_dir
Posted by ebiederm on Fri, 04 Jul 2008 01:17:30 GMT
View Forum Message <> Reply to Message

These two functions do 90% of the same work and it doesn't
significantly obfuscate the function to allow both the parent dir and
the name to change at the same time.  So merge them together to
simplify maintenance, and increase testing.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 fs/sysfs/dir.c |  121 ++++++++++++++++++++++++++++++++----------------------
 1 files changed, 38 insertions(+), 83 deletions(-)

```
diff --git a/fs/sysfs/dir.c b/fs/sysfs/dir.c
index 6dc3376..fe2bb1c 100644
--- a/fs/sysfs/dir.c
+++ b/fs/sysfs/dir.c
@@ -924,44 +924,57 @@ err_out:
  return error;
 }

-int sysfs_rename_dir(struct kobject * kobj, const char *new_name)
+static int sysfs_mv_dir(struct sysfs_dirent *sd,
+ struct sysfs_dirent *new_parent_sd, const char *new_name)
 {
- struct sysfs_dirent *sd = kobj->sd;
  struct list_head todo;
  struct sysfs_rename_struct *srs;
- struct inode *parent_inode = NULL;
+ struct inode *old_parent_inode = NULL, *new_parent_inode = NULL;
  const char *dup_name = NULL;
  const void *old_tag, *tag;
  int error;

  INIT_LIST_HEAD(&todo);
+ BUG_ON(!sd->s_parent);
  mutex_lock(&sysfs_rename_mutex);
+ if (!new_parent_sd)
+  new_parent_sd = &sysfs_root;
+
  old_tag = sd->s_tag;
  tag = sysfs_creation_tag(sd->s_parent, sd);

  error = 0;
- if ((old_tag == tag) && (strcmp(sd->s_name, new_name) == 0))
-  goto out; /* nothing to rename */
+ if ((sd->s_parent == new_parent_sd) && (old_tag == tag) &&
```

```
+     (strcmp(sd->s_name, new_name) == 0))
+  goto out; /* nothing to do */

  sysfs_grab_supers();
  if (old_tag == tag) {
-  error = prep_rename(&todo, sd, sd->s_parent, new_name);
+  error = prep_rename(&todo, sd, new_parent_sd, new_name);
    if (error)
     goto out_release;
  }

  error = -ENOMEM;
  mutex_lock(&sysfs_mutex);
- parent_inode = sysfs_get_inode(sd->s_parent);
+ old_parent_inode = sysfs_get_inode(sd->s_parent);
+ new_parent_inode = sysfs_get_inode(new_parent_sd);
  mutex_unlock(&sysfs_mutex);
- if (!parent_inode)
+ if (!old_parent_inode || !new_parent_inode)
    goto out_release;

- mutex_lock(&parent_inode->i_mutex);
+again:
+ mutex_lock(&old_parent_inode->i_mutex);
+ if (old_parent_inode != new_parent_inode) {
+  if (!mutex_trylock(&new_parent_inode->i_mutex)) {
+   mutex_unlock(&old_parent_inode->i_mutex);
+   goto again;
+  }
+ }
  mutex_lock(&sysfs_mutex);

  error = -EEXIST;
- if (sysfs_find_dirent(sd->s_parent, tag, new_name))
+ if (sysfs_find_dirent(new_parent_sd, tag, new_name))
    goto out_unlock;

  /* rename sysfs_dirent */
@@ -974,7 +987,7 @@ int sysfs_rename_dir(struct kobject * kobj, const char *new_name)
  sd->s_name = new_name;
  sd->s_tag = tag;

- /* rename */
+ /* rename dcache entries */
  list_for_each_entry(srs, &todo, list) {
    d_add(srs->new_dentry, NULL);
    d_move(srs->old_dentry, srs->new_dentry);
@@ -994,77 +1007,6 @@ int sysfs_rename_dir(struct kobject * kobj, const char *new_name)
```

```
  }
 }

- error = 0;
-out_unlock:
- mutex_unlock(&sysfs_mutex);
- mutex_unlock(&parent_inode->i_mutex);
- kfree(dup_name);
-out_release:
- iput(parent_inode);
- post_rename(&todo);
- sysfs_release_supers();
-out:
- mutex_unlock(&sysfs_rename_mutex);
- return error;
-}
-
-int sysfs_move_dir(struct kobject *kobj, struct kobject *new_parent_kobj)
-{
- struct sysfs_dirent *sd = kobj->sd;
- struct sysfs_dirent *new_parent_sd;
- struct list_head todo;
- struct sysfs_rename_struct *srs;
- struct inode *old_parent_inode = NULL, *new_parent_inode = NULL;
- int error;
- const void *tag;
-
- INIT_LIST_HEAD(&todo);
- mutex_lock(&sysfs_rename_mutex);
- BUG_ON(!sd->s_parent);
- new_parent_sd = new_parent_kobj->sd ? new_parent_kobj->sd : &sysfs_root;
- tag = sd->s_tag;
-
- error = 0;
- if (sd->s_parent == new_parent_sd)
-  goto out; /* nothing to move */
-
- sysfs_grab_supers();
- error = prep_rename(&todo, sd, new_parent_sd, sd->s_name);
- if (error)
-  goto out_release;
-
- error = -ENOMEM;
- mutex_lock(&sysfs_mutex);
- old_parent_inode = sysfs_get_inode(sd->s_parent);
- mutex_unlock(&sysfs_mutex);
- if (!old_parent_inode)
-  goto out_release;
```

```
-
- error = -ENOMEM;
- mutex_lock(&sysfs_mutex);
- new_parent_inode = sysfs_get_inode(new_parent_sd);
- mutex_unlock(&sysfs_mutex);
- if (!new_parent_inode)
-  goto out_release;
-
-again:
- mutex_lock(&old_parent_inode->i_mutex);
- if (!mutex_trylock(&new_parent_inode->i_mutex)) {
-  mutex_unlock(&old_parent_inode->i_mutex);
-  goto again;
- }
- mutex_lock(&sysfs_mutex);
-
- error = -EEXIST;
- if (sysfs_find_dirent(new_parent_sd, tag, sd->s_name))
-  goto out_unlock;
-
- error = 0;
- list_for_each_entry(srs, &todo, list) {
-  d_add(srs->new_dentry, NULL);
-  d_move(srs->old_dentry, srs->new_dentry);
- }
-
  /* Remove from old parent's list and insert into new parent's list. */
  sysfs_unlink_sibling(sd);
  sysfs_get(new_parent_sd);
@@ -1072,10 +1014,13 @@ again:
  sd->s_parent = new_parent_sd;
  sysfs_link_sibling(sd);

+ error = 0;
 out_unlock:
  mutex_unlock(&sysfs_mutex);
- mutex_unlock(&new_parent_inode->i_mutex);
+ if (new_parent_inode != old_parent_inode)
+  mutex_unlock(&new_parent_inode->i_mutex);
  mutex_unlock(&old_parent_inode->i_mutex);
+ kfree(dup_name);

 out_release:
  iput(new_parent_inode);
@@ -1087,6 +1032,16 @@ out:
  return error;
 }
```

```
+int sysfs_rename_dir(struct kobject * kobj, const char *new_name)
+{
+ return sysfs_mv_dir(kobj->sd, kobj->sd->s_parent, new_name);
+}
+
+int sysfs_move_dir(struct kobject *kobj, struct kobject *new_parent_kobj)
+{
+ return sysfs_mv_dir(kobj->sd, new_parent_kobj->sd, kobj->sd->s_name);
+}
+
 /* Relationship between s_mode and the DT_xxx types */
 static inline unsigned char dt_type(struct sysfs_dirent *sd)
 {
--
1.5.3.rc6.17.g1911
```



---

## Subject: [PATCH 11/15] sysfs: Implement sysfs_delete_link and sysfs_rename_link
Posted by ebiederm on Fri, 04 Jul 2008 01:18:29 GMT
View Forum Message <> Reply to Message

When removing a symlink sysfs_remove_link does not provide
enough information to figure out which tagged directory the symlink
falls in.  So I need sysfs_delete_link which is passed the target
of the symlink to delete.

Further half the time when we are removing a symlink the code is
actually renaming the symlink but not doing so explicitly because
we don't have a symlink rename method. So I have added sysfs_rename_link
as well.

Both of these functions now have enough information to find a symlink
in a tagged directory.  The only restriction is that they must be called
before the target kobject is renamed or deleted.  If they are called
later I loose track of which tag the target kobject was marked with
and can no longer find the old symlink to remove it.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
Signed-off-by: Benjamin Thery <benjamin.thery@bull.net>
Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>
Acked-by: Tejun Heo <tj@kernel.org>
---
 fs/sysfs/symlink.c    |   31 ++++++++++++++++++++++++++++++

```
 include/linux/sysfs.h |   17 ++++++++++++++++
 2 files changed, 48 insertions(+), 0 deletions(-)

diff --git a/fs/sysfs/symlink.c b/fs/sysfs/symlink.c
index de9a5c0..ed9c52c 100644
--- a/fs/sysfs/symlink.c
+++ b/fs/sysfs/symlink.c
@@ -80,6 +80,21 @@ int sysfs_create_link(struct kobject * kobj, struct kobject * target, const
char
 }

 /**
+ * sysfs_delete_link - remove symlink in object's directory.
+ * @kobj: object we're acting for.
+ * @targ: object we're pointing to.
+ * @name: name of the symlink to remove.
+ *
+ * Unlike sysfs_remove_link sysfs_delete_link has enough information
+ * to successfully delete symlinks in tagged directories.
+ */
+void sysfs_delete_link(struct kobject *kobj, struct kobject *targ,
+   const char *name)
+{
+ sysfs_hash_and_remove(targ, kobj->sd, name);
+}
+
+/**
  * sysfs_remove_link - remove symlink in object's directory.
  * @kobj: object we're acting for.
  * @name: name of the symlink to remove.
@@ -97,6 +112,22 @@ void sysfs_remove_link(struct kobject * kobj, const char * name)
  sysfs_hash_and_remove(kobj, parent_sd, name);
 }

+/**
+ * sysfs_rename_link - rename symlink in object's directory.
+ * @kobj: object we're acting for.
+ * @targ: object we're pointing to.
+ * @old: previous name of the symlink.
+ * @new: new name of the symlink.
+ *
+ * A helper function for the common rename symlink idiom.
+ */
+int sysfs_rename_link(struct kobject *kobj, struct kobject *targ,
+   const char *old, const char *new)
+{
+ sysfs_delete_link(kobj, targ, old);
+ return sysfs_create_link(kobj, targ, new);
```

```
+}
+
 static int sysfs_get_target_path(struct sysfs_dirent *parent_sd,
      struct sysfs_dirent *target_sd, char *path)
 {
diff --git a/include/linux/sysfs.h b/include/linux/sysfs.h
index 8fa97f0..c3a30ce 100644
--- a/include/linux/sysfs.h
+++ b/include/linux/sysfs.h
@@ -112,6 +112,12 @@ int __must_check sysfs_create_link(struct kobject *kobj, struct kobject
*target,
        const char *name);
 void sysfs_remove_link(struct kobject *kobj, const char *name);

+int sysfs_rename_link(struct kobject *kobj, struct kobject *target,
+   const char *old_name, const char *new_name);
+
+void sysfs_delete_link(struct kobject *dir, struct kobject *targ,
+   const char *name);
+
 int __must_check sysfs_create_group(struct kobject *kobj,
        const struct attribute_group *grp);
 int sysfs_update_group(struct kobject *kobj,
@@ -198,6 +204,17 @@ static inline void sysfs_remove_link(struct kobject *kobj, const char
*name)
 {
 }

+static inline int sysfs_rename_link(struct kobject *k, struct kobject *t,
+        const char *old_name, const char *new_name)
+{
+ return 0;
+}
+
+static inline void sysfs_delete_link(struct kobject *k, struct kobject *t,
+         const char *name)
+{
+}
+
 static inline int sysfs_create_group(struct kobject *kobj,
         const struct attribute_group *grp)
 {
--
1.5.3.rc6.17.g1911
```

Subject: [PATCH 12/15] driver core: Implement tagged directory support for device classes.
Posted by ebiederm on Fri, 04 Jul 2008 01:20:09 GMT
View Forum Message <> Reply to Message

This patch enables tagging on every class directory if struct class
has a tag_type.

In addition device_del and device_rename were modified to uses
sysfs_delete_link and sysfs_rename_link respectively to ensure
when these operations happen on devices whose classes have
tag_ops that they work properly.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
Signed-off-by: Benjamin Thery <benjamin.thery@bull.net>
---
 drivers/base/class.c   |   30 ++++++++++++++++++++---
 drivers/base/core.c    |   56 ++++++++++++++++++++++++++++++++++++++----------
 include/linux/device.h |    3 ++
 3 files changed, 68 insertions(+), 21 deletions(-)

diff --git a/drivers/base/class.c b/drivers/base/class.c
index 839d27c..cf4e03f 100644
--- a/drivers/base/class.c
+++ b/drivers/base/class.c
@@ -135,6 +135,17 @@ static void remove_class_attrs(struct class *cls)
  }
 }

+static int class_setup_tagging(struct class *cls)
+{
+ enum sysfs_tag_type type;
+
+ type = cls->tag_type;
+ if (type == SYSFS_TAG_TYPE_NONE)
+  return 0;
+
+ return sysfs_make_tagged_dir(&cls->p->class_subsys.kobj, type);
+}
+
 int __class_register(struct class *cls, struct lock_class_key *key)
 {
  struct class_private *cp;
@@ -171,13 +182,24 @@ int __class_register(struct class *cls, struct lock_class_key *key)
  cls->p = cp;

```
  error = kset_register(&cp->class_subsys);
- if (error) {
-  kfree(cp);
-  return error;
- }
+ if (error)
+  goto out_free_cp;
+
+ error = class_setup_tagging(cls);
+ if (error)
+  goto out_unregister;
+
  error = add_class_attrs(class_get(cls));
  class_put(cls);
+ if (error)
+  goto out_unregister;
+out:
  return error;
+out_unregister:
+ kset_unregister(&cp->class_subsys);
+out_free_cp:
+ kfree(cp);
+ goto out;
 }
 EXPORT_SYMBOL_GPL(__class_register);

diff --git a/drivers/base/core.c b/drivers/base/core.c
index 90621a4..b009d5b 100644
--- a/drivers/base/core.c
+++ b/drivers/base/core.c
@@ -124,9 +124,21 @@ static void device_release(struct kobject *kobj)
  }
 }

+static const void *device_sysfs_tag(struct kobject *kobj)
+{
+ struct device *dev = to_dev(kobj);
+ const void *tag = NULL;
+
+ if (dev->class && dev->class->tag_type)
+  tag = dev->class->sysfs_tag(dev);
+
+ return tag;
+}
+
 static struct kobj_type device_ktype = {
  .release = device_release,
```

```
  .sysfs_ops = &dev_sysfs_ops,
+ .sysfs_tag = device_sysfs_tag,
 };


@@ -619,6 +631,10 @@ static struct kobject *get_device_parent(struct device *dev,
    kobject_put(k);
    return NULL;
   }
+  /* If we created a new class-directory setup tagging */
+  if (dev->class->tag_type)
+   sysfs_make_tagged_dir(k, dev->class->tag_type);
+
   /* do not emit an uevent for this simple "glue" directory */
   return k;
  }
@@ -709,7 +725,7 @@ out_device:
 out_busid:
  if (dev->kobj.parent != &dev->class->p->class_subsys.kobj &&
      device_is_not_partition(dev))
-  sysfs_remove_link(&dev->class->p->class_subsys.kobj,
+  sysfs_delete_link(&dev->class->p->class_subsys.kobj, &dev->kobj,
       dev->bus_id);
 #else
  /* link in the class directory pointing to the device */
@@ -727,7 +743,7 @@ out_busid:
  return 0;

 out_busid:
- sysfs_remove_link(&dev->class->p->class_subsys.kobj, dev->bus_id);
+ sysfs_delete_link(&dev->class->p->class_subsys.kobj, &dev->kobj, dev->bus_id);
 #endif

 out_subsys:
@@ -755,13 +771,13 @@ static void device_remove_class_symlinks(struct device *dev)

  if (dev->kobj.parent != &dev->class->p->class_subsys.kobj &&
      device_is_not_partition(dev))
-  sysfs_remove_link(&dev->class->p->class_subsys.kobj,
+  sysfs_delete_link(&dev->class->p->class_subsys.kobj, &dev->kobj,
       dev->bus_id);
 #else
  if (dev->parent && device_is_not_partition(dev))
   sysfs_remove_link(&dev->kobj, "device");

- sysfs_remove_link(&dev->class->p->class_subsys.kobj, dev->bus_id);
+ sysfs_delete_link(&dev->class->p->class_subsys.kobj, &dev->kobj, dev->bus_id);
 #endif
```

```
  sysfs_remove_link(&dev->kobj, "subsystem");
@@ -1344,6 +1360,16 @@ int device_rename(struct device *dev, char *new_name)
  strlcpy(old_device_name, dev->bus_id, BUS_ID_SIZE);
  strlcpy(dev->bus_id, new_name, BUS_ID_SIZE);

+#ifndef CONFIG_SYSFS_DEPRECATED
+ if (dev->class &&
+     (dev->kobj.parent != &dev->class->p->class_subsys.kobj)) {
+  error = sysfs_rename_link(&dev->class->p->class_subsys.kobj,
+   &dev->kobj, old_device_name, new_name);
+  if (error)
+   goto out;
+ }
+#endif
+
  error = kobject_rename(&dev->kobj, new_name);
  if (error) {
   strlcpy(dev->bus_id, old_device_name, BUS_ID_SIZE);
@@ -1352,23 +1378,19 @@ int device_rename(struct device *dev, char *new_name)

 #ifdef CONFIG_SYSFS_DEPRECATED
  if (old_class_name) {
+  error = -ENOMEM;
   new_class_name = make_class_name(dev->class->name, &dev->kobj);
-  if (new_class_name) {
-   error = sysfs_create_link(&dev->parent->kobj,
-        &dev->kobj, new_class_name);
-   if (error)
-    goto out;
-   sysfs_remove_link(&dev->parent->kobj, old_class_name);
-  }
+  if (new_class_name)
+   error = sysfs_rename_link(&dev->parent->kobj,
+        &dev->kobj,
+        old_class_name,
+        new_class_name);
  }
 #else
  if (dev->class) {
-  error = sysfs_create_link(&dev->class->p->class_subsys.kobj,
-       &dev->kobj, dev->bus_id);
-  if (error)
-   goto out;
-  sysfs_remove_link(&dev->class->p->class_subsys.kobj,
-      old_device_name);
+  error = sysfs_rename_link(&dev->class->p->class_subsys.kobj,
+       &dev->kobj, old_device_name,
```

```
+       dev->bus_id);
 }
#endif

diff --git a/include/linux/device.h b/include/linux/device.h
index d9886a6..8e84539 100644
--- a/include/linux/device.h
+++ b/include/linux/device.h
@@ -191,6 +191,9 @@ struct class {
  int (*suspend)(struct device *dev, pm_message_t state);
  int (*resume)(struct device *dev);

+ enum sysfs_tag_type tag_type;
+ const void *(*sysfs_tag)(struct device *dev);
+
  struct class_private *p;
 };

--
1.5.3.rc6.17.g1911
```



---

Subject: Re: [PATCH 12/15] driver core: Implement tagged directory support for device classes.
Posted by Tejun Heo on Fri, 04 Jul 2008 07:50:40 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> This patch enables tagging on every class directory if struct class
> has a tag_type.
>
> In addition device_del and device_rename were modified to uses
> sysfs_delete_link and sysfs_rename_link respectively to ensure
> when these operations happen on devices whose classes have
> tag_ops that they work properly.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
> Signed-off-by: Benjamin Thery <benjamin.thery@bull.net>

Okay, I went through the users this time but I still think
determine-tags-by-callbacks is a bad idea.  Please just add const void
*tag to kobject and set it during initialization.  If you want to move a
device from one tag to another, implement kobject_rename_tagged(kobj,

new_name, new_tag).

The determine-tag-by-callback basically multiplexes basic functions to do tag-specific things which are determined by ktype callback called back from down the layer. It's simply a bad interface. Those operations become something else depending on how those callbacks behave. That's unnecessarily subtle. Advertising what it's gonna do in the function name and as arguments is way more straight forward and it's not like determining or renaming tags should be done asynchronously.

I personally think it would be better to make tags explicit in the mount interface too but if extracting ns information from the mounting process is what's currently being done, well...

I'm sorry but Nacked-by: Tejun Heo <tj@kernel.org>

Thanks.

--
tejun

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

---

Subject: Re: [PATCH 12/15] driver core: Implement tagged directory support for device classes.
Posted by ebiederm on Fri, 04 Jul 2008 13:31:29 GMT
View Forum Message <> Reply to Message

Thank you for your opinion.

Incremental patches to make things more beautiful are welcome.

Please remember we are not building lisp. The goal is code that works today.

Since we are not talking about correctness of the code. Since we are not talking about interfaces with user space. Since we are talking something that is currently about 100 lines of code, and so will be easy to change even after it is merged. I don't understand how discussing this further is useful. Especially when I get a NAK based on the feel that the code is ugly.

As for your main objection. Adding a accessor method to an object versus adding a data field that contain the same thing. The two are effectively identical. With the practical difference in my eyes that an accessor method prevents data duplication which reduces maintenance and reduces skew problems,

and it keeps the size of struct kobject small.  Since you think methods are
horrible I must respectfully disagree with you.

Eric

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

---

Subject: Re: [PATCH 12/15] driver core: Implement tagged directory support for device classes.
Posted by Tejun Heo on Fri, 04 Jul 2008 13:57:15 GMT
View Forum Message <> Reply to Message

Hello, Eric.

Eric W. Biederman wrote:
> Thank you for your opinion.
>
> Incremental patches to make things more beautiful are welcome.
>
> Please remember we are not building lisp.  The goal is code that works today.
>
> Since we are not talking about correctness of the code.  Since we are not
> talking about interfaces with user space.  Since we are talking something
> that is currently about 100 lines of code, and so will be easy to change
> even after it is merged.  I don't understand how discussing this further
> is useful.  Especially when I get a NAK  based on the feel that the code
> is ugly.

I'm sorry if I gave you the impression of being draconian.  Explanations
below.

> As for your main objection.  Adding a accessor method to an object versus
> adding a data field that contain the same thing.  The two are effectively
> identical.  With the practical difference in my eyes that an accessor method
> prevents data duplication which reduces maintenance and reduces skew problems,
> and it keeps the size of struct kobject small.  Since you think methods are
> horrible I must respectfully disagree with you.

Yeah, it seems we should agree to disagree here.  I think using callback
for static values is a really bad idea.  It obfuscates the code and
opens up a big hole for awful misuses.  Greg, what do you think?

As we're very close to rc1 window, I think we can work out a solution
here.  The reason why I nack'd was because the change wouldn't take too
much effort and I thought it could be done before -rc1.  Unless you

disagree with making tags static values, I'll try to write up a patch to
do so.  If you (and Greg) think the callback interface is better, we can
merge the code as-is and update (or not) later.

Thanks.

--
tejun

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

---

Subject: Re: [PATCH 12/15] driver core: Implement tagged directory support for device classes.
Posted by gregkh on Fri, 04 Jul 2008 16:12:00 GMT
View Forum Message <> Reply to Message

On Fri, Jul 04, 2008 at 10:57:15PM +0900, Tejun Heo wrote:
> Hello, Eric.
>
> Eric W. Biederman wrote:
> > Thank you for your opinion.
> >
> > Incremental patches to make things more beautiful are welcome.
> >
> > Please remember we are not building lisp.  The goal is code that works today.
> >
> > Since we are not talking about correctness of the code.  Since we are not
> > talking about interfaces with user space.  Since we are talking something
> > that is currently about 100 lines of code, and so will be easy to change
> > even after it is merged.  I don't understand how discussing this further
> > is useful.  Especially when I get a NAK  based on the feel that the code
> > is ugly.
>
> I'm sorry if I gave you the impression of being draconian.  Explanations
> below.
>
> > As for your main objection.  Adding a accessor method to an object versus
> > adding a data field that contain the same thing.  The two are effectively
> > identical.  With the practical difference in my eyes that an accessor method
> > prevents data duplication which reduces maintenance and reduces skew problems,
> > and it keeps the size of struct kobject small.  Since you think methods are
> > horrible I must respectfully disagree with you.
>
> Yeah, it seems we should agree to disagree here.  I think using callback
> for static values is a really bad idea.  It obfuscates the code and

> opens up a big hole for awful misuses.  Greg, what do you think?

Sorry, Greg is walking out the door in 30 minutes for a much needed week
long vacation and can't look into this right now :(

I'll be able to review it next weekend, sorry for the delay.

greg k-h

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

---

Subject: Re: [PATCH 12/15] driver core: Implement tagged directory support for device classes.
Posted by ebiederm on Fri, 04 Jul 2008 21:49:05 GMT
View Forum Message <> Reply to Message

Greg KH <gregkh@suse.de> writes:

> Sorry, Greg is walking out the door in 30 minutes for a much needed week
> long vacation and can't look into this right now :(
>
> I'll be able to review it next weekend, sorry for the delay.

Understood and no problem.

Eric

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

---

Subject: Re: [PATCH 12/15] driver core: Implement tagged directory support for device classes.
Posted by ebiederm on Fri, 04 Jul 2008 22:00:36 GMT
View Forum Message <> Reply to Message

Tejun Heo <htejun@gmail.com> writes:

> Yeah, it seems we should agree to disagree here.  I think using callback
> for static values is a really bad idea.  It obfuscates the code and
> opens up a big hole for awful misuses.  Greg, what do you think?

The misuse argument is small because currently all users must be compiled

into the kernel and must add to the static enumeration.  I'm afraid we
are making the facility over general for the problem at hand.

> As we're very close to rc1 window, I think we can work out a solution
> here.  The reason why I nack'd was because the change wouldn't take too
> much effort and I thought it could be done before -rc1.  Unless you
> disagree with making tags static values, I'll try to write up a patch to
> do so.  If you (and Greg) think the callback interface is better, we can
> merge the code as-is and update (or not) later.

Making a change and pushing down into the patches is much more time intensive
then I would like.  The last round of changes simple as they were took something
between 16 and 30 hours, and has left me sapped.  Keeping all of the other
pieces in flight in all of the other patches so I can't just focus on the
change at hand is what makes it difficult at this point.

Adding an additional patch on top isn't too bad, but my creativity is sapped on
this right now.  I agree that a function called device_rename isn't the best
possible name when we are changing tags, but I can't think of anything that
seems better.

I know in the users that the tags are already quite static and that I call
kobject_rename in the one case where they change (which is a significant
exception).  So that part doesn't concern me as I have not intention of using
the interface like that.  Ultimately I don't care as long as we have code
that works.

Eric

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

---

Subject: Re: [PATCH 12/15] driver core: Implement tagged directory support for device classes.
Posted by ebiederm on Mon, 14 Jul 2008 01:54:10 GMT
View Forum Message <> Reply to Message

Greg KH <gregkh@suse.de> writes:
>
> Sorry, Greg is walking out the door in 30 minutes for a much needed week
> long vacation and can't look into this right now :(
>
> I'll be able to review it next weekend, sorry for the delay.

Any progress in reviewing these changes, and seeing if you can stand
to merge them?

Eric

_____

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

---

Subject: Re: [PATCH 12/15] driver core: Implement tagged directory support for device classes.
Posted by Tejun Heo on Wed, 16 Jul 2008 03:25:24 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> Greg KH <gregkh@suse.de> writes:
>> Sorry, Greg is walking out the door in 30 minutes for a much needed week
>> long vacation and can't look into this right now :(
>>
>> I'll be able to review it next weekend, sorry for the delay.
>
> Any progress in reviewing these changes, and seeing if you can stand
> to merge them?

Greg, please disregard my earlier NACKs and commit the patches if you're okay with them. I'm working on cleaning it up but I don't think I'll be able to make it in time for merge window and as Eric said getting the functionality in place is more important at this point as it doesn't affect user visible interface.

Eric, with the multiple superblocks, sysfs now uses inode from the default sysfs_sb with dentries from other sb's. Is this okay? Are there any other filesystems which do this?

Thanks.

--
tejun

_____

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

---

Subject: Re: [PATCH 12/15] driver core: Implement tagged directory support for device classes.
Posted by ebiederm on Wed, 16 Jul 2008 05:41:14 GMT
View Forum Message <> Reply to Message

Tejun Heo <htejun@gmail.com> writes:

> Greg, please disregard my earlier NACKs and commit the patches if you're
> okay with them.  I'm working on cleaning it up but I don't think I'll be
> able to make it in time for merge window and as Eric said getting the
> functionality in place is more important at this point as it doesn't
> affect user visible interface.
>
> Eric, with the multiple superblocks, sysfs now uses inode from the
> default sysfs_sb with dentries from other sb's.  Is this okay?  Are
> there any other filesystems which do this?

I don't know of any other filesystems where this unique challenge arises.
/proc almost qualifies but it never needs to be modified.

It is certainly ok to go from multiple dentries to a single inode.
I'm trying to remember why I choose to do that.  I think both because it simplifies
the locking and keeps us more efficient in the icache.

Eric

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

---

Subject: Re: [PATCH 12/15] driver core: Implement tagged directory support for device classes.
Posted by Tejun Heo on Wed, 16 Jul 2008 05:50:02 GMT
View Forum Message <> Reply to Message

Hello,

Eric W. Biederman wrote:
>> Eric, with the multiple superblocks, sysfs now uses inode from the
>> default sysfs_sb with dentries from other sb's.  Is this okay?  Are
>> there any other filesystems which do this?
>
> I don't know of any other filesystems where this unique challenge arises.
> /proc almost qualifies but it never needs to be modified.
>
> It is certainly ok to go from multiple dentries to a single inode.
> I'm trying to remember why I choose to do that.  I think both because it simplifies
> the locking and keeps us more efficient in the icache.

It's a bit scary tho.  Working inode->i_dentry or dentry->d_alias
crosses multiple sb's.  sysfs isn't too greedy about dcache/icache.
Only open files and directories hold them and only single copy of

sysfs_dirent is there for most nodes.  Wouldn't it be better to stay on
the safer side and use separate inode hierarchy?

Thanks.

--
tejun

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

---

Subject: Re: [PATCH 12/15] driver core: Implement tagged directory support for device classes.
Posted by ebiederm on Wed, 16 Jul 2008 06:32:24 GMT
View Forum Message <> Reply to Message

Tejun Heo <htejun@gmail.com> writes:

> It's a bit scary tho.  Working inode->i_dentry or dentry->d_alias
> crosses multiple sb's.  sysfs isn't too greedy about dcache/icache.
> Only open files and directories hold them and only single copy of
> sysfs_dirent is there for most nodes.  Wouldn't it be better to stay on
> the safer side and use separate inode hierarchy?

To do that I believe we would need to ensure sysfs does not use
the inode->i_mutex lock except to keep the VFS layer out.  Allowing us
to safely change the directory structure, without holding it.

You raise a good point about inode->i_dentry and dentry->d_alias.
Generally they are used by fat like filesystems but I am starting to
see uses in generic pieces of code.  I don't see any problems today
but yes it would be good to do the refactoring to allow us to duplicate
the inodes.

Eric

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

---

Subject: Re: [PATCH 12/15] driver core: Implement tagged directory support for device classes.
Posted by Tejun Heo on Wed, 16 Jul 2008 06:48:39 GMT

Hello, Eric.

Eric W. Biederman wrote:
> Tejun Heo <htejun@gmail.com> writes:
>
>> It's a bit scary tho.  Working inode->i_dentry or dentry->d_alias
>> crosses multiple sb's.  sysfs isn't too greedy about dcache/icache.
>> Only open files and directories hold them and only single copy of
>> sysfs_dirent is there for most nodes.  Wouldn't it be better to stay on
>> the safer side and use separate inode hierarchy?
>
> To do that I believe we would need to ensure sysfs does not use
> the inode->i_mutex lock except to keep the VFS layer out.  Allowing us
> to safely change the directory structure, without holding it.

I don't think sysfs is depending on i_mutex anymore but I need to go
through the code to make sure.

> You raise a good point about inode->i_dentry and dentry->d_alias.
> Generally they are used by fat like filesystems but I am starting to
> see uses in generic pieces of code.  I don't see any problems today
> but yes it would be good to do the refactoring to allow us to duplicate
> the inodes.

Yeah, I can't spot any place which can cause actual problem yet but it's
still scary as we're breaking a vfs assumption and even if it's not a
problem now, future seemingly unrelated changes can break things subtly.

Thanks.

--
tejun

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

Subject: Re: [PATCH 12/15] driver core: Implement tagged directory support for device classes.
Posted by Tejun Heo on Wed, 16 Jul 2008 07:02:36 GMT


Tejun Heo wrote:
> Hello, Eric.
>

> Eric W. Biederman wrote:
>> Tejun Heo <htejun@gmail.com> writes:
>>
>>> It's a bit scary tho.  Working inode->i_dentry or dentry->d_alias
>>> crosses multiple sb's.  sysfs isn't too greedy about dcache/icache.
>>> Only open files and directories hold them and only single copy of
>>> sysfs_dirent is there for most nodes.  Wouldn't it be better to stay on
>>> the safer side and use separate inode hierarchy?
>> To do that I believe we would need to ensure sysfs does not use
>> the inode->i_mutex lock except to keep the VFS layer out.  Allowing us
>> to safely change the directory structure, without holding it.
>
> I don't think sysfs is depending on i_mutex anymore but I need to go
> through the code to make sure.
>
>> You raise a good point about inode->i_dentry and dentry->d_alias.
>> Generally they are used by fat like filesystems but I am starting to
>> see uses in generic pieces of code.  I don't see any problems today
>> but yes it would be good to do the refactoring to allow us to duplicate
>> the inodes.
>
> Yeah, I can't spot any place which can cause actual problem yet but it's
> still scary as we're breaking a vfs assumption and even if it's not a
> problem now, future seemingly unrelated changes can break things subtly.

Okay, one small problem spotted.  It seems invalidate_inodes() can fail
which will make generic_shutdown_super() complain.  It's not a fatal
failure tho.

--
tejun

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

---

Subject: Re: [PATCH 12/15] driver core: Implement tagged directory support for device classes.
Posted by ebiederm on Wed, 16 Jul 2008 19:07:25 GMT
View Forum Message <> Reply to Message

Tejun Heo <teheo@suse.de> writes:

> Okay, one small problem spotted.  It seems invalidate_inodes() can fail
> which will make generic_shutdown_super() complain.  It's not a fatal
> failure tho.

How when the inode list is empty?  We don't unmount the superblock that
has the inodes.

Eric

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

---

Subject: Re: [PATCH 12/15] driver core: Implement tagged directory support for device classes.
Posted by ebiederm on Wed, 16 Jul 2008 21:09:55 GMT
View Forum Message <> Reply to Message

Tejun Heo <htejun@gmail.com> writes:
>> To do that I believe we would need to ensure sysfs does not use
>> the inode->i_mutex lock except to keep the VFS layer out.  Allowing us
>> to safely change the directory structure, without holding it.
>
> I don't think sysfs is depending on i_mutex anymore but I need to go
> through the code to make sure.

The vfs still does. So at least for directory tree manipulation  we
need to hold i_mutex before we grab sysfs_mutex.

I think that means we need to unscramble the whole set of locking
order issues.

In lookup we have:
local_vfs_lock -> fs_global_lock

In modifications we have:
fs_global_lock -> local_vfs_lock

Which is the definition of a lock ordering problem.

Currently we play jump through some significant hoops to keep things
in local_vfs_lock -> fs_global_lock order.

If we also take the rename_mutex on directory adds and deletes we
may be able to keep jumping through those hoops.  However I expect
we would be in a much better situation if we could figure out how
to avoid the problem.

It looks like the easy way to handle this is to make the sysfs_dirent
list rcu protected.  Which means we can fix our lock ordering problem
without VFS modifications.  Allowing the locking to always

be: sysfs_mutex ... i_mutex.

After that it would be safe and a good idea to have unshared
inodes between superblocks, just so we don't surprise anyone
making generic VFS assumptions.

Eric

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

---

Subject: Re: [PATCH 12/15] driver core: Implement tagged directory support for device classes.
Posted by gregkh on Thu, 17 Jul 2008 23:08:57 GMT
View Forum Message <> Reply to Message

On Wed, Jul 16, 2008 at 12:25:24PM +0900, Tejun Heo wrote:
> Eric W. Biederman wrote:
> > Greg KH <gregkh@suse.de> writes:
> >> Sorry, Greg is walking out the door in 30 minutes for a much needed week
> >> long vacation and can't look into this right now :(
> >>
> >> I'll be able to review it next weekend, sorry for the delay.
> >
> > Any progress in reviewing these changes, and seeing if you can stand
> > to merge them?
>
> Greg, please disregard my earlier NACKs and commit the patches if you're
> okay with them.  I'm working on cleaning it up but I don't think I'll be
> able to make it in time for merge window and as Eric said getting the
> functionality in place is more important at this point as it doesn't
> affect user visible interface.

Ok, I'll work to get these in where applicable.

thanks,

greg k-h

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

---

Subject: Re: [PATCH 12/15] driver core: Implement tagged directory support for

device classes.
Posted by Tejun Heo on Fri, 18 Jul 2008 12:41:29 GMT
View Forum Message <> Reply to Message

Greg KH wrote:
> Ok, I'll work to get these in where applicable.

Did this get into 2.6.27?  Or will this have to wait till .28?

Thanks.

--
tejun

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

Subject: Re: [PATCH 12/15] driver core: Implement tagged directory support for device classes.
Posted by gregkh on Fri, 18 Jul 2008 18:49:36 GMT
View Forum Message <> Reply to Message

On Fri, Jul 18, 2008 at 09:41:29PM +0900, Tejun Heo wrote:
> Greg KH wrote:
> > Ok, I'll work to get these in where applicable.
>
> Did this get into 2.6.27?  Or will this have to wait till .28?

I haven't sent any patches to Linus yet for 2.6.27, so it hasn't gotten there yet.

And due to the intrusiveness, and the fact that this hasn't been tested at all in any build tree yet, I can't in good concious submit this for .27 either.

Part of this is my fault, I know, due to vacation and work, but also lots is due to the fact that the code showed up so late in the development cycle.

I'll add it to my tree, and get it some testing in the next cycle of linux-next until 2.6.27 is out, and then if it's still looking good, go into .28.

Hope this helps,

greg k-h

Subject: Re: [PATCH 12/15] driver core: Implement tagged directory support for device classes.
Posted by ebiederm on Fri, 18 Jul 2008 20:19:55 GMT
View Forum Message <> Reply to Message

Greg KH <gregkh@suse.de> writes:

> On Fri, Jul 18, 2008 at 09:41:29PM +0900, Tejun Heo wrote:
>> Greg KH wrote:
>> > Ok, I'll work to get these in where applicable.
>>
>> Did this get into 2.6.27?  Or will this have to wait till .28?
>
> I haven't sent any patches to Linus yet for 2.6.27, so it hasn't gotten
> there yet.
>
> And due to the intrusiveness, and the fact that this hasn't been tested
> at all in any build tree yet, I can't in good concious submit this for
> .27 either.
>
> Part of this is my fault, I know, due to vacation and work, but also
> lots is due to the fact that the code showed up so late in the
> development cycle.

The code showed up at least by the beginning of may, and the code has
been around for 9 months or more.  I just haven't always had the
energy to retransmit every time it has gotten dropped.  The last spin
of it was very late I agree.

> I'll add it to my tree, and get it some testing in the next cycle of
> linux-next until 2.6.27 is out, and then if it's still looking good, go
> into .28.

I think that is unnecessarily precautions but reasonable.  That should at
least keep other sysfs patches from coming in and causing bit rot.

Eric

Subject: Re: [PATCH 12/15] driver core: Implement tagged directory support for device classes.
Posted by Tejun Heo on Sat, 19 Jul 2008 01:07:55 GMT
View Forum Message <> Reply to Message

Hello, Greg.

Greg KH wrote:
> On Fri, Jul 18, 2008 at 09:41:29PM +0900, Tejun Heo wrote:
>> Greg KH wrote:
>>> Ok, I'll work to get these in where applicable.
>> Did this get into 2.6.27?  Or will this have to wait till .28?
>
> I haven't sent any patches to Linus yet for 2.6.27, so it hasn't gotten
> there yet.
>
> And due to the intrusiveness, and the fact that this hasn't been tested
> at all in any build tree yet, I can't in good concious submit this for
> .27 either.
>
> Part of this is my fault, I know, due to vacation and work, but also
> lots is due to the fact that the code showed up so late in the
> development cycle.

Heh... A lot of it is my fault too.  Probably my share is bigger than anyone else's.  Sorry Eric.

> I'll add it to my tree, and get it some testing in the next cycle of
> linux-next until 2.6.27 is out, and then if it's still looking good, go
> into .28.

I was just curious how the merge will turn out as I'm about to do a refresh pass on the sysfs locking and stuff.  I'm not sure yet whether to put those changes on top of or below ns patches but if I'll always keep the ns patches updated.

Thank you.

--
tejun