

---

Subject: [PATCH] Containers: Fix refcount bug  
Posted by [menage](#) on Mon, 10 Sep 2007 21:51:12 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Fix a reference counting bug in containerfs

As part of the extraction of cpusetfs to containerfs, a call to `cpuset_get_dentry()` was lost (justified by the fact that the dentry in question was now being passed down by the caller). Since `cpuset_get_dentry()` called `lookup_one_len()`, this resulted in a reference count being missed from the directory dentry.

This patch removes `container_get_dentry()` and replaces it with direct calls to `lookup_one_len()`; the initialization of containerfs dentry ops is done now in `container_create_file()` at dentry creation time.

Signed-off-by: Paul Menage <[menage@google.com](mailto:menage@google.com)>

---

kernel/container.c | 26 ++++++++-----  
1 file changed, 8 insertions(+), 18 deletions(-)

Index: container-2.6.23-rc3-mm1/kernel/container.c

=====

--- container-2.6.23-rc3-mm1.orig/kernel/container.c

+++ container-2.6.23-rc3-mm1/kernel/container.c

@@ -603,19 +603,6 @@ static void container\_diput(struct dentry \*dentry)  
iput(inode);  
}

-static struct dentry \*container\_get\_dentry(struct dentry \*parent,  
- const char \*name)

-{  
- struct dentry \*d = lookup\_one\_len(name, parent, strlen(name));  
- static struct dentry\_operations container\_dops = {  
- .d\_iput = container\_diput,  
- };  
-

- if (!IS\_ERR(d))  
- d->d\_op = &container\_dops;  
- return d;  
-}

-  
static void remove\_dir(struct dentry \*d)  
{

struct dentry \*parent = dget(d->d\_parent);  
@@ -1506,6 +1493,10 @@ static struct inode\_operations container\_ops = {  
static int container\_create\_file(struct dentry \*dentry, int mode,

```

    struct super_block *sb)
{
+ static struct dentry_operations container_dops = {
+ .d_iput = container_diput,
+ };
+
    struct inode *inode;

    if (!dentry)
@@ -1531,7 +1522,7 @@ static int container_create_file(struct
    inode->i_size = 0;
    inode->i_fop = &container_file_operations;
}
-
+ dentry->d_op = &container_dops;
    d_instantiate(dentry, inode);
    dget(dentry); /* Extra count - pin the dentry in core */
    return 0;
@@ -1552,13 +1543,12 @@ static int container_create_dir(struct c
    int error = 0;

    parent = cont->parent->dentry;
- if (IS_ERR(dentry))
- return PTR_ERR(dentry);
    error = container_create_file(dentry, S_IFDIR | mode, cont->root->sb);
    if (!error) {
        dentry->d_fsdata = cont;
        inc_nlink(parent->d_inode);
        cont->dentry = dentry;
+ dget(dentry);
    }
    dput(dentry);

@@ -1580,7 +1570,7 @@ int container_add_file(struct container
}
    strcat(name, cft->name);
    BUG_ON(!mutex_is_locked(&dir->d_inode->i_mutex));
- dentry = container_get_dentry(dir, name);
+ dentry = lookup_one_len(name, dir, strlen(name));
    if (!IS_ERR(dentry)) {
        error = container_create_file(dentry, 0644 | S_IFREG,
            cont->root->sb);
@@ -2586,7 +2576,7 @@ int container_clone(struct task_struct *
/* Hold the parent directory mutex across this operation to
 * stop anyone else deleting the new container */
    mutex_lock(&inode->i_mutex);
- dentry = container_get_dentry(parent->dentry, nodename);
+ dentry = lookup_one_len(nodename, parent->dentry, strlen(nodename));

```

```
if (IS_ERR(dentry)) {  
    printk(KERN_INFO  
        "Couldn't allocate dentry for %s: %ld\n", nodename,
```

---

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

---