
Subject: [PATCH 15/25] elevate write count files are open()ed
Posted by [Dave Hansen](#) on Mon, 11 Dec 2006 22:30:12 GMT
[View Forum Message](#) <> [Reply to Message](#)

This is the first really tricky patch in the series. It elevates the writer count on a mount each time a non-special file is opened for write.

This is not completely apparent in the patch because the two if() conditions in may_open() above the mnt_want_write() call are, combined, equivalent to special_file().

There is also an elevated count around the vfs_create() call in open_namei(). The count needs to be kept elevated all the way into the may_open() call. Otherwise, when the write is dropped, a ro->rw transision could occur. This would lead to having rw access on the newly created file, while the vfsmount is ro. That is bad.

Signed-off-by: Dave Hansen <haveblue@us.ibm.com>

```
lxc-dave/fs/file_table.c | 5 ++++
lxc-dave/fs/namei.c      | 22 ++++++-----
lxc-dave/ipc/mqueue.c    | 3 +++
3 files changed, 25 insertions(+), 5 deletions(-)
```

```
diff -puN fs/file_table.c~14-24-tricky-elevate-write-count-files-are-open-ed fs/file_table.c
--- lxc/fs/file_table.c~14-24-tricky-elevate-write-count-files-are-open-ed 2006-12-11
14:22:06.000000000 -0800
+++ lxc-dave/fs/file_table.c 2006-12-11 14:22:06.000000000 -0800
@@ -202,8 +202,11 @@ void fastcall __fput(struct file *file)
     if (unlikely(S_ISCHR(inode->i_mode) && inode->i_cdev != NULL))
         cdev_put(inode->i_cdev);
     fops_put(file->f_op);
- if (file->f_mode & FMODE_WRITE)
+ if (file->f_mode & FMODE_WRITE) {
     put_write_access(inode);
+ if(!special_file(inode->i_mode))
+ mnt_drop_write(mnt);
+ }
     put_pid(file->f_owner.pid);
     file_kill(file);
     file->f_path.dentry = NULL;
diff -puN fs/namei.c~14-24-tricky-elevate-write-count-files-are-open-ed fs/namei.c
--- lxc/fs/namei.c~14-24-tricky-elevate-write-count-files-are-open-ed 2006-12-11
14:22:06.000000000 -0800
```

```

+++ lxc-dave/fs/namei.c 2006-12-11 14:22:06.000000000 -0800
@@ -1544,8 +1544,17 @@ int may_open(struct nameidata *nd, int a
    return -EACCES;

    flag &= ~O_TRUNC;
- } else if (IS_RDONLY(inode) && (flag & FMODE_WRITE))
- return -EROFS;
+ } else if (flag & FMODE_WRITE) {
+ /*
+  * effectively: !special_file()
+  * balanced by __fput()
+  */
+ error = mnt_want_write(nd->mnt);
+ if (error)
+ return error;
+ if (IS_RDONLY(inode))
+ return -EROFS;
+ }
+ /*
+  * An append-only file must be opened in append mode for writing.
+  */
@@ -1684,14 +1693,17 @@ do_last:
}

if (IS_ERR(nd->intent.open.file)) {
- mutex_unlock(&dir->d_inode->i_mutex);
- error = PTR_ERR(nd->intent.open.file);
- goto exit_dput;
+ goto exit_mutex_unlock;
}

/* Negative dentry, just create the file */
if (!path.dentry->d_inode) {
+ error = mnt_want_write(nd->mnt);
+ if (error)
+ goto exit_mutex_unlock;
+ error = open_namei_create(nd, &path, flag, mode);
+ mnt_drop_write(nd->mnt);
+ if (error)
+ goto exit;
+ return 0;
@@ -1729,6 +1741,8 @@ ok:
goto exit;
return 0;

+exit_mutex_unlock:
+ mutex_unlock(&dir->d_inode->i_mutex);
+ exit_dput:

```

```
dput_path(&path, nd);
exit:
diff -puN ipc/mqueue.c~14-24-tricky-elevate-write-count-files-are-open-ed ipc/mqueue.c
--- lxc/ipc/mqueue.c~14-24-tricky-elevate-write-count-files-are-open-ed 2006-12-11
14:22:06.000000000 -0800
+++ lxc-dave/ipc/mqueue.c 2006-12-11 14:22:06.000000000 -0800
@@ -687,6 +687,9 @@ asmlinkage long sys_mq_open(const char _
    goto out;
    filp = do_open(dentry, oflag);
} else {
+ error = mnt_want_write(mqueue_mnt);
+ if (error)
+ goto out;
    filp = do_create(mqueue_mnt->mnt_root, dentry,
        oflag, mode, u_attr);
}
```

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