

[PATCH] open returns ENFILE but creates file anyway

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-08/7674.html>

From: Peter Staubach (staubach_at_redhat.com)

Date: 08/30/05

Date: Tue, 30 Aug 2005 14:30:46 -0400

To: Linux Kernel Mailing List <linux-kernel@vger.kernel.org>

Hi.

When `open(O_CREAT)` is called and the error, `ENFILE`, is returned, the file may be created anyway. This is counter intuitive, against the SUS V3 specification, and may cause applications to misbehave if they are not coded correctly to handle this semantic. The SUS V3 specification explicitly states "No files shall be created or modified if the function returns `-1`".

The error, `ENFILE`, is used to indicate the system wide open file table is full and no more file structs can be allocated.

This is due to an ordering problem. The entry in the directory is created before the file struct is allocated. If the allocation for the file struct fails, then the system call must return an error, but the directory entry was already created and can not be safely removed.

The solution to this situation is relatively easy. The file struct should be allocated before the directory entry is created. If the allocation fails, then the error can be returned directly. If the creation of the directory entry fails, then the file struct can be easily freed.

Thanx...

ps

Signed-off-by: Peter Staubach <staubach@redhat.com>

--- linux-2.6.12/fs/open.c.org 2005-08-24 10:51:00.000000000 -0400

+++ linux-2.6.12/fs/open.c 2005-08-24 11:28:59.000000000 -0400

@@ -741,6 +741,9 @@ asmlinkage long sys_fchown(unsigned int

Linux–Kernel: [PATCH] open returns ENFILE but creates file anyway

```
    return error;
}

+static struct file *__dentry_open(struct dentry *, struct vfsmount *, int,
+ struct file *);
+
+/*
+ * Note that while the flag value (low two bits) for sys_open means:
+ * 00 – read-only
+@@ -759,6 +762,7 @@ struct file *filp_open(const char * file
+ {
+     int namei_flags, error;
+     struct nameidata nd;
+ struct file *f;

+     namei_flags = flags;
+     if ((namei_flags+1) & O_ACCMODE)
+@@ -766,10 +770,16 @@ struct file *filp_open(const char * file
+         if (namei_flags & O_TRUNC)
+             namei_flags |= 2;

+ error = -ENFILE;
+ f = get_empty_filp();
+ if (f == NULL)
+ return ERR_PTR(error);
+
+     error = open_namei(filename, namei_flags, mode, &nd);
+     if (!error)
+ – return dentry_open(nd.dentry, nd.mnt, flags);
+ return __dentry_open(nd.dentry, nd.mnt, flags, f);

+ put_filp(f);
+     return ERR_PTR(error);
+ }

+@@ -777,14 +787,24 @@ EXPORT_SYMBOL(filp_open);

+ struct file *dentry_open(struct dentry *dentry, struct vfsmount *mnt, int flags)
+ {
+ – struct file * f;
+ – struct inode *inode;
+     int error;
+ struct file *f;

+     error = -ENFILE;
+     f = get_empty_filp();
+ – if (!f)
+ – goto cleanup_dentry;
+ + if (f == NULL)
+ + return ERR_PTR(error);
+
+ }
```

Linux-Kernel: [PATCH] open returns ENFILE but creates file anyway

```
+ return __dentry_open(dentry, mnt, flags, f);
+}
+
+EXPORT_SYMBOL(dentry_open);
+
+static struct file *__dentry_open(struct dentry *dentry, struct vfsmount *mnt, int flags, struct file *f)
+{
+ struct inode *inode;
+ int error;
+
+     f->f_flags = flags;
+     f->f_mode = ((flags+1) & O_ACCMODE) | FMODE_LSEEK | FMODE_PREAD | FMODE_PWRITE;
+     inode = dentry->d_inode;
@@ -831,14 +851,11 @@ cleanup_all:
+     f->f_vfsmnt = NULL;
cleanup_file:
+     put_filp(f);
-cleanup_dentry:
+     dput(dentry);
+     mntput(mnt);
+     return ERR_PTR(error);
+}

-EXPORT_SYMBOL(dentry_open);
-
-/*
+ * Find an empty file descriptor entry, and mark it busy.
+ */
```

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@vger.kernel.org
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>