
Subject: [RFC][PATCH 3/4] create fill_if_new_sb() helper
Posted by [Dave Hansen](#) on Fri, 26 Jan 2007 22:42:11 GMT
[View Forum Message](#) <> [Reply to Message](#)

As I was migrating /proc away from get_sb_single() use, I noticed I was duplicating a lot of functionality in the /proc code from get_sb_single(). I then noticed that get_sb_nodev() does basically the same thing.

So, I created a fill_if_new_sb() helper to work on the superblock after sget() to fill it only when necessary and handle the work with sb->s_flags and the s->s_umount semaphore.

I'll use this in a moment in /proc as well.

```
lxc-dave/fs/super.c      | 43 ++++++-----  
lxc-dave/include/linux/fs.h | 2 ++  
2 files changed, 25 insertions(+), 20 deletions(-)
```

```
diff -puN fs/super.c~A2-fill_if_new_sb fs/super.c  
--- lxc/fs/super.c~A2-fill_if_new_sb 2007-01-26 14:29:17.000000000 -0800  
+++ lxc-dave/fs/super.c 2007-01-26 14:29:17.000000000 -0800  
@@ -810,6 +810,23 @@ void kill_block_super(struct super_block  
EXPORT_SYMBOL(kill_block_super);  
#endif
```

```
+int fill_if_new_sb(struct super_block *s, void *data, int flags,  
+ int (*fill_super)(struct super_block *, void *, int))  
+{  
+ int error;  
+ if (s->s_root)  
+ return 0;  
+ s->s_flags = flags;  
+ error = fill_super(s, data, flags & MS_SILENT ? 1 : 0);  
+ if (error) {  
+ up_write(&s->s_umount);  
+ deactivate_super(s);  
+ return error;  
+ }  
+ s->s_flags |= MS_ACTIVE;  
+ return error;  
+}  
+  
int get_sb_nodev(struct file_system_type *fs_type,  
int flags, void *data,  
int (*fill_super)(struct super_block *, void *, int),
```

@@ -820,16 +837,9 @@ int get_sb_nODEV(struct file_system_type

```
if (IS_ERR(s))
    return PTR_ERR(s);
-
- s->s_flags = flags;
-
- error = fill_super(s, data, flags & MS_SILENT ? 1 : 0);
- if (error) {
-     up_write(&s->s_umount);
-     deactivate_super(s);
+ error = fill_if_new_sb(s, data, flags, fill_super);
+ if (error)
    return error;
- }
- s->s_flags |= MS_ACTIVE;
    return simple_set_mnt(mnt, s);
}
```

@@ -845,22 +855,15 @@ int get_sb_single(struct file_system_type
 int (*fill_super)(struct super_block *, void *, int),
 struct vfsmount *mnt)
{

```
- struct super_block *s;
    int error;
+ struct super_block *s;

    s = sget(fs_type, compare_single, set_anon_super, NULL);
    if (IS_ERR(s))
        return PTR_ERR(s);
- if (!s->s_root) {
-     s->s_flags = flags;
-     error = fill_super(s, data, flags & MS_SILENT ? 1 : 0);
-     if (error) {
-         up_write(&s->s_umount);
-         deactivate_super(s);
-         return error;
-     }
-     s->s_flags |= MS_ACTIVE;
- }
+ error = fill_if_new_sb(s, data, flags, fill_super);
+ if (error)
+     return error;
    do_remount_sb(s, flags, data, 0);
    return simple_set_mnt(mnt, s);
}
```

diff -puN include/linux/fs.h~A2-fill_if_new_sb include/linux/fs.h

--- lxc/include/linux/fs.h~A2-fill_if_new_sb 2007-01-26 14:29:17.000000000 -0800

```
+++ lxc-dave/include/linux/fs.h 2007-01-26 14:29:17.000000000 -0800
@@ -1416,6 +1416,8 @@ extern int get_sb_nodev(struct file_syst
    int flags, void *data,
    int (*fill_super)(struct super_block *, void *, int),
    struct vfsmount *mnt);
+extern int fill_if_new_sb(struct super_block *s, void *data, int flags,
+ int (*fill_super)(struct super_block *, void *, int));
void generic_shutdown_super(struct super_block *sb);
void kill_block_super(struct super_block *sb);
void kill_anon_super(struct super_block *sb);
```

—

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