
Subject: [patch 08/22] r/o bind mounts: elevate write count for some ioctls

Posted by [Cedric Le Goater](#) on Thu, 07 Jun 2007 15:25:35 GMT

[View Forum Message](#) <> [Reply to Message](#)

From: Dave Hansen <hansendc@us.ibm.com>

Some ioctl()s can cause writes to the filesystem. Take these, and make them use mnt_want/drop_write() instead.

We need to pass the filp one layer deeper in XFS, but somebody `_just_` pulled it out in February because nobody was using it, so I don't feel guilty for adding it back.

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

```
fs/ext2/ioctl.c      | 67 ++++++++-----
fs/ext3/ioctl.c      | 100 ++++++++-----
fs/ext4/ioctl.c      | 105 ++++++++-----
fs/fat/file.c        | 9 +-
fs/hfsplus/ioctl.c   | 39 ++++++-----
fs/jfs/ioctl.c       | 33 ++++++-----
fs/ocfs2/ioctl.c     | 11 +-
fs/reiserfs/ioctl.c  | 52 ++++++++-----
fs/xfs/linux-2.6/xfs_ioctl.c | 12 +++-
fs/xfs/linux-2.6/xfs_iops.c | 7 --
fs/xfs/linux-2.6/xfs_lrw.c | 8 +-
11 files changed, 281 insertions(+), 162 deletions(-)
```

Index: 2.6.22-rc4-mm2-robindmount/fs/ext2/ioctl.c

=====

--- 2.6.22-rc4-mm2-robindmount.orig/fs/ext2/ioctl.c

+++ 2.6.22-rc4-mm2-robindmount/fs/ext2/ioctl.c

@@ -12,6 +12,7 @@

#include <linux/time.h>

#include <linux/sched.h>

#include <linux/compat.h>

+#include <linux/mount.h>

#include <linux/smp_lock.h>

#include <asm/current.h>

#include <asm/uaccess.h>

@@ -23,6 +24,7 @@ int ext2_ioctl (struct inode * inode, st

struct ext2_inode_info *ei = EXT2_I(inode);

unsigned int flags;

unsigned short rsv_window_size;

+ int ret;

```
ext2_debug ("cmd = %u, arg = %lu\n", cmd, arg);
```

```
@@ -34,14 +36,19 @@ int ext2_ioctl (struct inode * inode, st
```

```
case EXT2_IOC_SETFLAGS: {  
    unsigned int oldflags;
```

```
- if (IS_RDONLY(inode))  
- return -EROFS;  
-  
- if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER))  
- return -EACCES;  
+ ret = mnt_want_write(filp->f_vfsmnt);  
+ if (ret)  
+ return ret;  
+  
+ if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER)) {  
+ ret = -EACCES;  
+ goto setflags_out;  
+ }  
  
- if (get_user(flags, (int __user *) arg))  
- return -EFAULT;  
+ if (get_user(flags, (int __user *) arg)) {  
+ ret = -EFAULT;  
+ goto setflags_out;  
+ }
```

```
    if (!S_ISDIR(inode->i_mode))  
        flags &= ~EXT2_DIRSYNC_FL;
```

```
@@ -58,7 +65,8 @@ int ext2_ioctl (struct inode * inode, st  
    if ((flags ^ oldflags) & (EXT2_APPEND_FL | EXT2_IMMUTABLE_FL)) {  
        if (!capable(CAP_LINUX_IMMUTABLE)) {  
            mutex_unlock(&inode->i_mutex);  
- return -EPERM;  
+ ret = -EPERM;  
+ goto setflags_out;  
        }  
    }
```

```
@@ -70,19 +78,25 @@ int ext2_ioctl (struct inode * inode, st
```

```
    ext2_set_inode_flags(inode);  
    inode->i_ctime = CURRENT_TIME_SEC;  
    mark_inode_dirty(inode);
```

```
- return 0;  
+ setflags_out:  
+ mnt_drop_write(filp->f_vfsmnt);  
+ return ret;  
}
```

```

case EXT2_IOC_GETVERSION:
    return put_user(inode->i_generation, (int __user *) arg);
case EXT2_IOC_SETVERSION:
    if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER))
        return -EPERM;
-   if (IS_RDONLY(inode))
-   return -EROFS;
-   if (get_user(inode->i_generation, (int __user *) arg))
-   return -EFAULT;
-   inode->i_ctime = CURRENT_TIME_SEC;
-   mark_inode_dirty(inode);
+   ret = mnt_want_write(filp->f_vfsmnt);
+   if (ret)
+   return ret;
+   if (get_user(inode->i_generation, (int __user *) arg)) {
+   ret = -EFAULT;
+   } else {
+   inode->i_ctime = CURRENT_TIME_SEC;
+   mark_inode_dirty(inode);
+   }
+   mnt_drop_write(filp->f_vfsmnt);
    return 0;
case EXT2_IOC_GETRSVSZ:
    if (test_opt(inode->i_sb, RESERVATION)
@@ -97,14 +111,19 @@ int ext2_ioctl (struct inode * inode, st
    if (!test_opt(inode->i_sb, RESERVATION) || !S_ISREG(inode->i_mode))
        return -ENOTTY;

-   if (IS_RDONLY(inode))
-   return -EROFS;
-
-   if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER))
-   return -EACCES;
+   ret = mnt_want_write(filp->f_vfsmnt);
+   if (ret)
+   return ret;
+
+   if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER)) {
+   ret = -EACCES;
+   goto setrsvsz_out;
+   }

-   if (get_user(rsv_window_size, (int __user *)arg))
-   return -EFAULT;
+   if (get_user(rsv_window_size, (int __user *)arg)) {
+   ret = -EFAULT;
+   goto setrsvsz_out;
+   }

```

```

    if (rsv_window_size > EXT2_MAX_RESERVE_BLOCKS)
        rsv_window_size = EXT2_MAX_RESERVE_BLOCKS;
@@ -126,7 +145,9 @@ int ext2_ioctl (struct inode * inode, st
    rsv->rsv_goal_size = rsv_window_size;
}
mutex_unlock(&ei->truncate_mutex);
- return 0;
+ setrsvsz_out:
+ mnt_drop_write(filp->f_vfsmnt);
+ return ret;
}
default:
    return -ENOTTY;

```

Index: 2.6.22-rc4-mm2-robindmount/fs/ext3/ioctl.c

=====

--- 2.6.22-rc4-mm2-robindmount.orig/fs/ext3/ioctl.c

+++ 2.6.22-rc4-mm2-robindmount/fs/ext3/ioctl.c

@@ -12,6 +12,7 @@

#include <linux/capability.h>

#include <linux/ext3_fs.h>

#include <linux/ext3_jbd.h>

+#include <linux/mount.h>

#include <linux/time.h>

#include <linux/compat.h>

#include <linux/smp_lock.h>

@@ -38,14 +39,19 @@ int ext3_ioctl (struct inode * inode, st

unsigned int oldflags;

unsigned int jflag;

- if (IS_RDONLY(inode))

- return -EROFS;

+ err = mnt_want_write(filp->f_vfsmnt);

+ if (err)

+ return err;

- if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER))

- return -EACCES;

+ if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER)) {

+ err = -EACCES;

+ goto flags_out;

+ }

- if (get_user(flags, (int __user *) arg))

- return -EFAULT;

+ if (get_user(flags, (int __user *) arg)) {

+ err = -EFAULT;

+ goto flags_out;

```

+ }

    if (!S_ISDIR(inode->i_mode))
        flags &= ~EXT3_DIRSYNC_FL;
@@ -65,7 +71,8 @@ int ext3_ioctl (struct inode * inode, st
    if ((flags ^ oldflags) & (EXT3_APPEND_FL | EXT3_IMMUTABLE_FL)) {
        if (!capable(CAP_LINUX_IMMUTABLE)) {
            mutex_unlock(&inode->i_mutex);
-         return -EPERM;
+         err = -EPERM;
+         goto flags_out;
        }
    }

@@ -76,7 +83,8 @@ int ext3_ioctl (struct inode * inode, st
    if ((jflag ^ oldflags) & (EXT3_JOURNAL_DATA_FL)) {
        if (!capable(CAP_SYS_RESOURCE)) {
            mutex_unlock(&inode->i_mutex);
-         return -EPERM;
+         err = -EPERM;
+         goto flags_out;
        }
    }

@@ -84,7 +92,8 @@ int ext3_ioctl (struct inode * inode, st
    handle = ext3_journal_start(inode, 1);
    if (IS_ERR(handle)) {
        mutex_unlock(&inode->i_mutex);
-         return PTR_ERR(handle);
+         err = PTR_ERR(handle);
+         goto flags_out;
    }
    if (IS_SYNC(inode))
        handle->h_sync = 1;
@@ -110,6 +119,8 @@ flags_err:
    if ((jflag ^ oldflags) & (EXT3_JOURNAL_DATA_FL))
        err = ext3_change_inode_journal_flag(inode, jflag);
    mutex_unlock(&inode->i_mutex);
+ flags_out:
+ mnt_drop_write(filp->f_vfsmnt);
    return err;
}
case EXT3_IOC_GETVERSION:
@@ -124,14 +135,18 @@ flags_err:

    if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER))
        return -EPERM;
-    if (IS_RDONLY(inode))

```

```

- return -EROFS;
- if (get_user(generation, (int __user *) arg))
- return -EFAULT;
-
+ err = mnt_want_write(filp->f_vfsmnt);
+ if (err)
+ return err;
+ if (get_user(generation, (int __user *) arg)) {
+ err = -EFAULT;
+ goto setversion_out;
+ }
    handle = ext3_journal_start(inode, 1);
- if (IS_ERR(handle))
- return PTR_ERR(handle);
+ if (IS_ERR(handle)) {
+ err = PTR_ERR(handle);
+ goto setversion_out;
+ }
    err = ext3_reserve_inode_write(handle, inode, &iloc);
    if (err == 0) {
        inode->i_ctime = CURRENT_TIME_SEC;
@@ -139,6 +154,8 @@ flags_err:
        err = ext3_mark_iloc_dirty(handle, inode, &iloc);
    }
    ext3_journal_stop(handle);
+ setversion_out:
+ mnt_drop_write(filp->f_vfsmnt);
    return err;
}
#ifdef CONFIG_JBD_DEBUG
@@ -174,18 +191,24 @@ flags_err:
    }
    return -ENOTTY;
    case EXT3_IOC_SETTRSVSZ: {
+ int err;

    if (!test_opt(inode->i_sb, RESERVATION) || !S_ISREG(inode->i_mode))
        return -ENOTTY;

- if (IS_RDONLY(inode))
- return -EROFS;
+ err = mnt_want_write(filp->f_vfsmnt);
+ if (err)
+ return err;

- if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER))
- return -EACCES;
+ if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER)) {

```

```

+   err = -EACCES;
+   goto setrsvsz_out;
+ }

- if (get_user(rsv_window_size, (int __user *)arg))
-   return -EFAULT;
+ if (get_user(rsv_window_size, (int __user *)arg)) {
+   err = -EFAULT;
+   goto setrsvsz_out;
+ }

    if (rsv_window_size > EXT3_MAX_RESERVE_BLOCKS)
        rsv_window_size = EXT3_MAX_RESERVE_BLOCKS;
@@ -203,7 +226,9 @@ flags_err:
    rsv->rsv_goal_size = rsv_window_size;
}
mutex_unlock(&ei->truncate_mutex);
- return 0;
+ setrsvsz_out:
+ mnt_drop_write(filp->f_vfsmnt);
+ return err;
}
case EXT3_IOC_GROUP_EXTEND: {
    ext3_fsblk_t n_blocks_count;
@@ -213,17 +238,20 @@ flags_err:
    if (!capable(CAP_SYS_RESOURCE))
        return -EPERM;

- if (IS_RDONLY(inode))
-   return -EROFS;
-
- if (get_user(n_blocks_count, (__u32 __user *)arg))
-   return -EFAULT;
+ err = mnt_want_write(filp->f_vfsmnt);
+ if (err)
+   return err;

+ if (get_user(n_blocks_count, (__u32 __user *)arg)) {
+   err = -EFAULT;
+   goto group_extend_out;
+ }
    err = ext3_group_extend(sb, EXT3_SB(sb)->s_es, n_blocks_count);
    journal_lock_updates(EXT3_SB(sb)->s_journal);
    journal_flush(EXT3_SB(sb)->s_journal);
    journal_unlock_updates(EXT3_SB(sb)->s_journal);
-
+ group_extend_out:
+ mnt_drop_write(filp->f_vfsmnt);

```

```

    return err;
}
case EXT3_IOC_GROUP_ADD: {
@@ -234,18 +262,22 @@ flags_err:
    if (!capable(CAP_SYS_RESOURCE))
        return -EPERM;

- if (IS_RDONLY(inode))
- return -EROFS;
+ err = mnt_want_write(filp->f_vfsmnt);
+ if (err)
+ return err;

    if (copy_from_user(&input, (struct ext3_new_group_input __user *)arg,
- sizeof(input)))
- return -EFAULT;
+ sizeof(input))) {
+ err = -EFAULT;
+ goto group_add_out;
+ }

    err = ext3_group_add(sb, &input);
    journal_lock_updates(EXT3_SB(sb)->s_journal);
    journal_flush(EXT3_SB(sb)->s_journal);
    journal_unlock_updates(EXT3_SB(sb)->s_journal);
-
+ group_add_out:
+ mnt_drop_write(filp->f_vfsmnt);
    return err;
}

```

Index: 2.6.22-rc4-mm2-robindmount/fs/ext4/ioctl.c

```

=====
--- 2.6.22-rc4-mm2-robindmount.orig/fs/ext4/ioctl.c
+++ 2.6.22-rc4-mm2-robindmount/fs/ext4/ioctl.c
@@ -12,6 +12,7 @@
#include <linux/capability.h>
#include <linux/ext4_fs.h>
#include <linux/ext4_jbd2.h>
+#include <linux/mount.h>
#include <linux/time.h>
#include <linux/compat.h>
#include <linux/smp_lock.h>
@@ -38,15 +39,19 @@ int ext4_ioctl (struct inode * inode, st
    unsigned int oldflags;
    unsigned int jflag;

- if (IS_RDONLY(inode))

```



```

- return -EROFS;
-
- if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER))
- return -EACCES;
+ err = mnt_want_write(filp->f_vfsmnt);
+ if (err)
+ return err;

- if (get_user(flags, (int __user *) arg))
- return -EFAULT;
+ if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER)) {
+ err = -EACCES;
+ goto flags_out;
+ }

+ if (get_user(flags, (int __user *) arg)) {
+ err = -EFAULT;
+ goto flags_out;
+ }
    if (!S_ISDIR(inode->i_mode))
        flags &= ~EXT4_DIRSYNC_FL;

@@ -65,7 +70,8 @@ int ext4_ioctl (struct inode * inode, st
    if ((flags ^ oldflags) & (EXT4_APPEND_FL | EXT4_IMMUTABLE_FL)) {
        if (!capable(CAP_LINUX_IMMUTABLE)) {
            mutex_unlock(&inode->i_mutex);
- return -EPERM;
+ err = -EPERM;
+ goto flags_out;
        }
    }

@@ -76,7 +82,8 @@ int ext4_ioctl (struct inode * inode, st
    if ((jflag ^ oldflags) & (EXT4_JOURNAL_DATA_FL)) {
        if (!capable(CAP_SYS_RESOURCE)) {
            mutex_unlock(&inode->i_mutex);
- return -EPERM;
+ err = -EPERM;
+ goto flags_out;
        }
    }

@@ -84,7 +91,8 @@ int ext4_ioctl (struct inode * inode, st
    handle = ext4_journal_start(inode, 1);
    if (IS_ERR(handle)) {
        mutex_unlock(&inode->i_mutex);
- return PTR_ERR(handle);
+ err = PTR_ERR(handle);

```

```

+ goto flags_out;
}
if (IS_SYNC(inode))
    handle->h_sync = 1;
@@ -104,12 +112,14 @@ flags_err:
    ext4_journal_stop(handle);
    if (err) {
        mutex_unlock(&inode->i_mutex);
- return err;
+ goto flags_out;
    }

    if ((jflag ^ oldflags) & (EXT4_JOURNAL_DATA_FL))
        err = ext4_change_inode_journal_flag(inode, jflag);
    mutex_unlock(&inode->i_mutex);
+flags_out:
+ mnt_drop_write(filp->f_vfsmnt);
    return err;
}
case EXT4_IOC_GETVERSION:
@@ -124,14 +134,18 @@ flags_err:

    if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER))
        return -EPERM;
- if (IS_RDONLY(inode))
- return -EROFS;
- if (get_user(generation, (int __user *) arg))
- return -EFAULT;
-
+ err = mnt_want_write(filp->f_vfsmnt);
+ if (err)
+ return err;
+ if (get_user(generation, (int __user *) arg)) {
+ err = -EFAULT;
+ goto setversion_out;
+ }
    handle = ext4_journal_start(inode, 1);
- if (IS_ERR(handle))
- return PTR_ERR(handle);
+ if (IS_ERR(handle)) {
+ err = PTR_ERR(handle);
+ goto setversion_out;
+ }
    err = ext4_reserve_inode_write(handle, inode, &iloc);
    if (err == 0) {
        inode->i_ctime = ext4_current_time(inode);
@@ -139,6 +153,8 @@ flags_err:
        err = ext4_mark_iloc_dirty(handle, inode, &iloc);

```

```

    }
    ext4_journal_stop(handle);
+setversion_out:
+ mnt_drop_write(filp->f_vfsmnt);
    return err;
}
#ifdef CONFIG_JBD_DEBUG
@@ -174,19 +190,23 @@ flags_err:
    }
    return -ENOTTY;
    case EXT4_IOC_SETRSVSZ: {
+ int err;

    if (!test_opt(inode->i_sb, RESERVATION) || !S_ISREG(inode->i_mode))
        return -ENOTTY;

- if (IS_RDONLY(inode))
- return -EROFS;
-
- if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER))
- return -EACCES;
-
- if (get_user(rsv_window_size, (int __user *)arg))
- return -EFAULT;
+ err = mnt_want_write(filp->f_vfsmnt);
+ if (err)
+ return err;

+ if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER)) {
+ err = -EACCES;
+ goto setrsvsz_out;
+ }
+ if (get_user(rsv_window_size, (int __user *)arg)) {
+ err = -EFAULT;
+ goto setrsvsz_out;
+ }
    if (rsv_window_size > EXT4_MAX_RESERVE_BLOCKS)
        rsv_window_size = EXT4_MAX_RESERVE_BLOCKS;

@@ -203,7 +223,9 @@ flags_err:
    rsv->rsv_goal_size = rsv_window_size;
    }
    mutex_unlock(&ei->truncate_mutex);
- return 0;
+setrsvsz_out:
+ mnt_drop_write(filp->f_vfsmnt);
+ return err;
    }

```

```

case EXT4_IOC_GROUP_EXTEND: {
    ext4_fsblk_t n_blocks_count;
@@ -213,17 +235,21 @@ flags_err:
    if (!capable(CAP_SYS_RESOURCE))
        return -EPERM;

- if (IS_RDONLY(inode))
- return -EROFS;
-
- if (get_user(n_blocks_count, (__u32 __user *)arg))
- return -EFAULT;
+ err = mnt_want_write(filp->f_vfsmnt);
+ if (err)
+ return err;

+ if (get_user(n_blocks_count, (__u32 __user *)arg)) {
+ err = -EFAULT;
+ goto group_extend_out;
+ }
    err = ext4_group_extend(sb, EXT4_SB(sb)->s_es, n_blocks_count);
    jbd2_journal_lock_updates(EXT4_SB(sb)->s_journal);
    jbd2_journal_flush(EXT4_SB(sb)->s_journal);
    jbd2_journal_unlock_updates(EXT4_SB(sb)->s_journal);

+ group_extend_out:
+ mnt_drop_write(filp->f_vfsmnt);
    return err;
}

case EXT4_IOC_GROUP_ADD: {
@@ -234,18 +260,21 @@ flags_err:
    if (!capable(CAP_SYS_RESOURCE))
        return -EPERM;

- if (IS_RDONLY(inode))
- return -EROFS;
+ err = mnt_want_write(filp->f_vfsmnt);
+ if (err)
+ return err;

    if (copy_from_user(&input, (struct ext4_new_group_input __user *)arg,
        sizeof(input)))
- return -EFAULT;
-
+ sizeof(input))) {
+ err = -EFAULT;
+ goto group_add_out;
+ }
    err = ext4_group_add(sb, &input);

```

```

jbd2_journal_lock_updates(EXT4_SB(sb)->s_journal);
jbd2_journal_flush(EXT4_SB(sb)->s_journal);
jbd2_journal_unlock_updates(EXT4_SB(sb)->s_journal);
-
+ group_add_out:
+ mnt_drop_write(filp->f_vfsmnt);
  return err;
}

```

Index: 2.6.22-rc4-mm2-robindmount/fs/fat/file.c

```

=====
--- 2.6.22-rc4-mm2-robindmount.orig/fs/fat/file.c
+++ 2.6.22-rc4-mm2-robindmount/fs/fat/file.c
@@ -46,10 +46,9 @@ int fat_generic_ioctl(struct inode *inode
    mutex_lock(&inode->i_mutex);

- if (IS_RDONLY(inode)) {
-   err = -EROFS;
-   goto up;
- }
+ err = mnt_want_write(filp->f_vfsmnt);
+ if (err)
+   goto up_no_drop_write;

/*
 * ATTR_VOLUME and ATTR_DIR cannot be changed; this also
@@ -106,6 +105,8 @@ int fat_generic_ioctl(struct inode *inode
    MSDOS_I(inode)->i_attrs = attr & ATTR_UNUSED;
    mark_inode_dirty(inode);
up:
+ mnt_drop_write(filp->f_vfsmnt);
+ up_no_drop_write:
    mutex_unlock(&inode->i_mutex);
    return err;
}

```

Index: 2.6.22-rc4-mm2-robindmount/fs/hfsplus/ioctl.c

```

=====
--- 2.6.22-rc4-mm2-robindmount.orig/fs/hfsplus/ioctl.c
+++ 2.6.22-rc4-mm2-robindmount/fs/hfsplus/ioctl.c
@@ -35,25 +35,32 @@ int hfsplus_ioctl(struct inode *inode, s
    flags |= FS_NODUMP_FL; /* EXT2_NODUMP_FL */
    return put_user(flags, (int __user *)arg);
case HFSPLUS_IOC_EXT2_SETFLAGS: {
- if (IS_RDONLY(inode))
-   return -EROFS;
-
- if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER))

```

```

- return -EACCES;
-
- if (get_user(flags, (int __user *)arg))
- return -EFAULT;
-
+ int err = 0;
+ err = mnt_want_write(filp->f_vfsmnt);
+ if (err)
+ return err;
+
+ if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER)) {
+ err = -EACCES;
+ goto setflags_out;
+ }
+ if (get_user(flags, (int __user *)arg)) {
+ err = -EFAULT;
+ goto setflags_out;
+ }
+ if (flags & (FS_IMMUTABLE_FL|FS_APPEND_FL) ||
+     HFSPLUS_I(inode).rootflags & (HFSPLUS_FLG_IMMUTABLE|HFSPLUS_FLG_APPEND)) {
- if (!capable(CAP_LINUX_IMMUTABLE))
- return -EPERM;
+ if (!capable(CAP_LINUX_IMMUTABLE)) {
+ err = -EPERM;
+ goto setflags_out;
+ }
+ }

/* don't silently ignore unsupported ext2 flags */
- if (flags & ~(FS_IMMUTABLE_FL|FS_APPEND_FL|FS_NODUMP_FL))
- return -EOPNOTSUPP;
-
+ if (flags & ~(FS_IMMUTABLE_FL|FS_APPEND_FL|FS_NODUMP_FL)) {
+ err = -EOPNOTSUPP;
+ goto setflags_out;
+ }
+ if (flags & FS_IMMUTABLE_FL) { /* EXT2_IMMUTABLE_FL */
+ inode->i_flags |= S_IMMUTABLE;
+ HFSPLUS_I(inode).rootflags |= HFSPLUS_FLG_IMMUTABLE;
@@ -75,7 +82,9 @@ int hfsplus_ioctl(struct inode *inode, s

inode->i_ctime = CURRENT_TIME_SEC;
mark_inode_dirty(inode);
- return 0;
+ setflags_out:
+ mnt_drop_write(filp->f_vfsmnt);
+ return err;
+ }

```

default:

return -ENOTTY;

Index: 2.6.22-rc4-mm2-robindmount/fs/jfs/ioctl.c

```
=====
--- 2.6.22-rc4-mm2-robindmount.orig/fs/jfs/ioctl.c
+++ 2.6.22-rc4-mm2-robindmount/fs/jfs/ioctl.c
@@ -8,6 +8,7 @@
#include <linux/fs.h>
#include <linux/ctype.h>
#include <linux/capability.h>
+#include <linux/mount.h>
#include <linux/time.h>
#include <linux/sched.h>
#include <asm/current.h>
@@ -65,16 +66,20 @@ int jfs_ioctl(struct inode * inode, stru
    return put_user(flags, (int __user *) arg);
    case JFS_IOC_SETFLAGS: {
        unsigned int oldflags;
+    int err;

-    if (IS_RDONLY(inode))
-        return -EROFS;
-
-    if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER))
-        return -EACCES;
-
-    if (get_user(flags, (int __user *) arg))
-        return -EFAULT;
-
+    err = mnt_want_write(filp->f_vfsmnt);
+    if (err)
+        return err;
+
+    if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER)) {
+        err = -EACCES;
+        goto setflags_out;
+    }
+    if (get_user(flags, (int __user *) arg)) {
+        err = -EFAULT;
+        goto setflags_out;
+    }
        flags = jfs_map_ext2(flags, 1);
        if (!S_ISDIR(inode->i_mode))
            flags &= ~JFS_DIRSYNC_FL;
@@ -89,8 +94,10 @@ int jfs_ioctl(struct inode * inode, stru
    if ((oldflags & JFS_IMMUTABLE_FL) ||
        ((flags ^ oldflags) &
         (JFS_APPEND_FL | JFS_IMMUTABLE_FL))) {
```

```

- if (!capable(CAP_LINUX_IMMUTABLE))
- return -EPERM;
+ if (!capable(CAP_LINUX_IMMUTABLE)) {
+ err = -EPERM;
+ goto setflags_out;
+ }
}

flags = flags & JFS_FL_USER_MODIFIABLE;
@@ -100,7 +107,9 @@ int jfs_iocctl(struct inode * inode, stru
jfs_set_inode_flags(inode);
inode->i_ctime = CURRENT_TIME_SEC;
mark_inode_dirty(inode);
- return 0;
+ setflags_out:
+ mnt_drop_write(filp->f_vfsmnt);
+ return err;
}
default:
return -ENOTTY;
Index: 2.6.22-rc4-mm2-robindmount/fs/ocfs2/iocctl.c
=====
--- 2.6.22-rc4-mm2-robindmount.orig/fs/ocfs2/iocctl.c
+++ 2.6.22-rc4-mm2-robindmount/fs/ocfs2/iocctl.c
@@ -57,10 +57,6 @@ static int ocfs2_set_inode_attr(struct i
goto bail;
}

- status = -EROFS;
- if (IS_RDONLY(inode))
- goto bail_unlock;
-
status = -EACCES;
if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER))
goto bail_unlock;
@@ -128,8 +124,13 @@ int ocfs2_iocctl(struct inode * inode, st
if (get_user(flags, (int __user *) arg))
return -EFAULT;

- return ocfs2_set_inode_attr(inode, flags,
+ status = mnt_want_write(filp->f_vfsmnt);
+ if (status)
+ return status;
+ status = ocfs2_set_inode_attr(inode, flags,
OCFS2_FL_MODIFIABLE);
+ mnt_drop_write(filp->f_vfsmnt);
+ return status;
default:

```



```

    return -ENOTTY;
}
Index: 2.6.22-rc4-mm2-robindmount/fs/reiserfs/ioctl.c
=====
--- 2.6.22-rc4-mm2-robindmount.orig/fs/reiserfs/ioctl.c
+++ 2.6.22-rc4-mm2-robindmount/fs/reiserfs/ioctl.c
@@ -25,6 +25,7 @@ int reiserfs_ioctl(struct inode *inode,
    unsigned long arg)
{
    unsigned int flags;
+ int err = 0;

    switch (cmd) {
    case REISERFS_IOC_UNPACK:
@@ -48,48 +49,61 @@ int reiserfs_ioctl(struct inode *inode,
    if (!reiserfs_attrs(inode->i_sb))
        return -ENOTTY;

- if (IS_RDONLY(inode))
-     return -EROFS;
+ err = mnt_want_write(filp->f_vfsmnt);
+ if (err)
+     return err;

    if ((current->fsuid != inode->i_uid)
        && !capable(CAP_FOWNER))
        return -EPERM;
-
- if (get_user(flags, (int __user *)arg))
-     return -EFAULT;
-
+ && !capable(CAP_FOWNER)) {
+     err = -EPERM;
+     goto setflags_out;
+ }
+ if (get_user(flags, (int __user *)arg)) {
+     err = -EFAULT;
+     goto setflags_out;
+ }
    if (((flags ^ REISERFS_I(inode)->
        i_attrs) & (REISERFS_IMMUTABLE_FL |
        REISERFS_APPEND_FL))
        && !capable(CAP_LINUX_IMMUTABLE))
        return -EPERM;
-
+ && !capable(CAP_LINUX_IMMUTABLE)) {
+     err = -EPERM;
+     goto setflags_out;

```

```

+ }
  if ((flags & REISERFS_NOTAIL_FL) &&
      S_ISREG(inode->i_mode)) {
    int result;

    result = reiserfs_unpack(inode, filp);
-   if (result)
-     return result;
+   if (result) {
+     err = result;
+     goto setflags_out;
+   }
  }
  sd_attrs_to_i_attrs(flags, inode);
  REISERFS_I(inode)->i_attrs = flags;
  inode->i_ctime = CURRENT_TIME_SEC;
  mark_inode_dirty(inode);
-   return 0;
+ setflags_out:
+   mnt_drop_write(filp->f_vfsmnt);
+   return err;
  }
  case REISERFS_IOC_GETVERSION:
    return put_user(inode->i_generation, (int __user *)arg);
  case REISERFS_IOC_SETVERSION:
    if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER))
      return -EPERM;
-   if (IS_RDONLY(inode))
-     return -EROFS;
-   if (get_user(inode->i_generation, (int __user *)arg))
-     return -EFAULT;
+   err = mnt_want_write(filp->f_vfsmnt);
+   if (err)
+     return err;
+   if (get_user(inode->i_generation, (int __user *)arg)) {
+     err = -EFAULT;
+     goto setversion_out;
+   }
  inode->i_ctime = CURRENT_TIME_SEC;
  mark_inode_dirty(inode);
-   return 0;
+ setversion_out:
+   mnt_drop_write(filp->f_vfsmnt);
+   return err;
  default:
    return -ENOTTY;
  }

```

Index: 2.6.22-rc4-mm2-robindmount/fs/xfs/linux-2.6/xfs_ioctl.c

```

=====
--- 2.6.22-rc4-mm2-robindmount.orig/fs/xfs/linux-2.6/xfs_ioctl.c
+++ 2.6.22-rc4-mm2-robindmount/fs/xfs/linux-2.6/xfs_ioctl.c
@@ -526,8 +526,6 @@ xfs_attrmulti_attr_set(
    char *kbuf;
    int error = EFAULT;

- if (IS_RDONLY(&vp->v_inode))
- return -EROFS;
    if (IS_IMMUTABLE(&vp->v_inode) || IS_APPEND(&vp->v_inode))
        return EPERM;
    if (len > XATTR_SIZE_MAX)
@@ -553,8 +551,6 @@ xfs_attrmulti_attr_remove(
    char *name,
    __uint32_t flags)
{
- if (IS_RDONLY(&vp->v_inode))
- return -EROFS;
    if (IS_IMMUTABLE(&vp->v_inode) || IS_APPEND(&vp->v_inode))
        return EPERM;
    return bhv_vop_attr_remove(vp, name, flags, NULL);
@@ -618,13 +614,21 @@ xfs_attrmulti_by_handle(
    &ops[i].am_length, ops[i].am_flags);
    break;
    case ATTR_OP_SET:
+ ops[i].am_error = mnt_want_write(parfilp->f_vfsmnt);
+ if (ops[i].am_error)
+ break;
    ops[i].am_error = xfs_attrmulti_attr_set(vp,
        attr_name, ops[i].am_attrvalue,
        ops[i].am_length, ops[i].am_flags);
+ mnt_drop_write(parfilp->f_vfsmnt);
    break;
    case ATTR_OP_REMOVE:
+ ops[i].am_error = mnt_want_write(parfilp->f_vfsmnt);
+ if (ops[i].am_error)
+ break;
    ops[i].am_error = xfs_attrmulti_attr_remove(vp,
        attr_name, ops[i].am_flags);
+ mnt_drop_write(parfilp->f_vfsmnt);
    break;
    default:
        ops[i].am_error = EINVAL;
Index: 2.6.22-rc4-mm2-robindmount/fs/xfs/linux-2.6/xfs_iops.c
=====
--- 2.6.22-rc4-mm2-robindmount.orig/fs/xfs/linux-2.6/xfs_iops.c
+++ 2.6.22-rc4-mm2-robindmount/fs/xfs/linux-2.6/xfs_iops.c
@@ -156,13 +156,6 @@ xfs_ichgtime_fast(

```

```

*/
ASSERT((flags & XFS_ICHGTIME_ACC) == 0);

- /*
- * We're not supposed to change timestamps in readonly-mounted
- * filesystems. Throw it away if anyone asks us.
- */
- if (unlikely(IS_RDONLY(inode)))
- return;
-
  if (flags & XFS_ICHGTIME_MOD) {
    tvp = &inode->i_mtime;
    ip->i_d.di_mtime.t_sec = (__int32_t)tvp->tv_sec;
Index: 2.6.22-rc4-mm2-robindmount/fs/xfs/linux-2.6/xfs_lrw.c
=====
--- 2.6.22-rc4-mm2-robindmount.orig/fs/xfs/linux-2.6/xfs_lrw.c
+++ 2.6.22-rc4-mm2-robindmount/fs/xfs/linux-2.6/xfs_lrw.c
@@ -750,10 +750,16 @@ start:
     if (new_size > xip->i_size)
         io->io_new_size = new_size;

- if (likely(!(ioflags & IO_INVIS))) {
+ /*
+ * We're not supposed to change timestamps in readonly-mounted
+ * filesystems. Throw it away if anyone asks us.
+ */
+ if (likely(!(ioflags & IO_INVIS) &&
+ !mnt_want_write(file->f_vfsmnt))) {
     file_update_time(file);
     xfs_ichgtime_fast(xip, inode,
        XFS_ICHGTIME_MOD | XFS_ICHGTIME_CHG);
+ mnt_drop_write(file->f_vfsmnt);
+ }

/*

--

```

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