
Subject: [PATCH 1/5] Use existing macros for distinguishing mandatory locks
Posted by [Pavel Emelianov](#) on Wed, 12 Sep 2007 11:17:58 GMT

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

The combination of S_ISGID bit set and S_IXGRP bit unset is used to mark the inode as "mandatory lockable" and there's a macro for this check called MANDATORY_LOCK(inode). However, fs/locks.c and some filesystems still perform the explicit i_mode checking.

Switch the fs/locks.c to macro making the code shorter and more readable.

The __MANDATORY_LOCK() macro is to be used in places where the IS_MANDLOCK() for superblock is already known to be true.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

```
fs/locks.c      | 14 +++++-----
include/linux/fs.h | 4 +---
2 files changed, 6 insertions(+), 12 deletions(-)
```

```
diff --git a/include/linux/fs.h b/include/linux/fs.h
index 291d40b..035ffda 100644
```

```
--- a/include/linux/fs.h
```

```
+++ b/include/linux/fs.h
```

```
@@ -1488,8 +1488,8 @@ extern int locks_mandatory_area(int, str
 * Candidates for mandatory locking have the setgid bit set
 * but no group execute bit - an otherwise meaningless combination.
 */
```

```
+#define MANDATORY_LOCK(inode) \
```

```
- (IS_MANDLOCK(inode) && ((inode)->i_mode & (S_ISGID | S_IXGRP)) == S_ISGID)
```

```
+#define __MANDATORY_LOCK(ino) (((ino)->i_mode & (S_ISGID | S_IXGRP)) == S_ISGID)
```

```
+#define MANDATORY_LOCK(inode) (IS_MANDLOCK(inode) &&
__MANDATORY_LOCK(inode))
```

```
static inline int locks_verify_locked(struct inode *inode)
```

```
{
```

```
diff --git a/fs/locks.c b/fs/locks.c
```

```
index 0db1a14..f90fe6d 100644
```

```
--- a/fs/locks.c
```

```
+++ b/fs/locks.c
```

```
@@ -1112,7 +1112,7 @@ int locks_mandatory_area(int read_write,
 * If we've been sleeping someone might have
 * changed the permissions behind our back.
 */
```

```

- if ((inode->i_mode & (S_ISGID | S_IXGRP)) == S_ISGID)
+ if (__MANDATORY_LOCK(inode))
    continue;
}

@@ -1751,9 +1751,7 @@ int fcntl_setlk(unsigned int fd, struct
/* Don't allow mandatory locks on files that may be memory mapped
 * and shared.
 */
- if (IS_MANDLOCK(inode) &&
-     (inode->i_mode & (S_ISGID | S_IXGRP)) == S_ISGID &&
-     mapping_writably_mapped(filp->f_mapping)) {
+ if (MANDATORY_LOCK(inode) && mapping_writably_mapped(filp->f_mapping)) {
    error = -EAGAIN;
    goto out;
}

@@ -1877,9 +1875,7 @@ int fcntl_setlk64(unsigned int fd, struc
/* Don't allow mandatory locks on files that may be memory mapped
 * and shared.
 */
- if (IS_MANDLOCK(inode) &&
-     (inode->i_mode & (S_ISGID | S_IXGRP)) == S_ISGID &&
-     mapping_writably_mapped(filp->f_mapping)) {
+ if (MANDATORY_LOCK(inode) && mapping_writably_mapped(filp->f_mapping)) {
    error = -EAGAIN;
    goto out;
}

@@ -2073,9 +2069,7 @@ static void lock_get_status(char* out, s
out += sprintf(out, "%6s %s ",
    (fl->fl_flags & FL_ACCESS) ? "ACCESS" : "POSIX ",
    (inode == NULL) ? "*NOINODE*" :
-     (IS_MANDLOCK(inode) &&
-      (inode->i_mode & (S_IXGRP | S_ISGID)) == S_ISGID) ?
-     "MANDATORY" : "ADVISORY ");
+     MANDATORY_LOCK(inode) ? "MANDATORY" : "ADVISORY ");
} else if (IS_FLOCK(fl)) {
    if (fl->fl_type & LOCK_MAND) {
        out += sprintf(out, "FLOCK MSNFS  ");

```

Subject: Re: [PATCH 1/5] Use existing macros for distinguishing mandatory locks
 Posted by [akpm](#) on Fri, 14 Sep 2007 23:42:15 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Wed, 12 Sep 2007 15:17:58 +0400
 Pavel Emelyanov <xemul@openvz.org> wrote:

> The combination of S_ISGID bit set and S_IXGRP bit unset is

> used to mark the inode as "mandatory lockable" and there's a
> macro for this check called MANDATORY_LOCK(inode). However,
> fs/locks.c and some filesystems still perform the explicit
> i_mode checking.
>
> Switch the fs/locks.c to macro making the code shorter and
> more readable.
>
> The __MANDATORY_LOCK() macro is to be used in places where
> the IS_MANDLOCK() for superblock is already known to be true.
>

If we're going to churn this code then it would be better to switch from ugly-upper-case-macro to nice-lower-case-C-function while we're doing it, please.

```
>
> diff --git a/include/linux/fs.h b/include/linux/fs.h
> index 291d40b..035ffda 100644
> --- a/include/linux/fs.h
> +++ b/include/linux/fs.h
> @@ -1488,8 +1488,8 @@ extern int locks_mandatory_area(int, str
> * Candidates for mandatory locking have the setgid bit set
> * but no group execute bit - an otherwise meaningless combination.
> */
> -#define MANDATORY_LOCK(inode) \
> - (IS_MANDLOCK(inode) && ((inode)->i_mode & (S_ISGID | S_IXGRP)) == S_ISGID)
> +#define __MANDATORY_LOCK(ino) (((ino)->i_mode & (S_ISGID | S_IXGRP)) == S_ISGID)
> +#define MANDATORY_LOCK(inode) (IS_MANDLOCK(inode) &&
__MANDATORY_LOCK(inode))
```

Especially as the macro is a buggy one which references its argument more than once.
