

[PATCH] FAT: fix VFAT_IOCTL_READDIR_BOTH ioctl

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2004-10/10329.html>

From: OGAWA Hirofumi (*hirofumi_at_mail.parknet.co.jp*)

Date: 10/31/04

To: Linus Torvalds <torvalds@osdl.org>, Andrew Morton <akpm@osdl.org>

Date: Mon, 01 Nov 2004 03:31:41 +0900

Hi,

Please apply

--

OGAWA Hirofumi <hirofumi@mail.parknet.co.jp>

If "utf8" option was specified and app uses VFAT_IOCTL_READDIR_BOTH ioctl, utf8_wcstombs() is used. utf8_wcstombs() doesn't add the nul terminate. Then fat_ioctl_filldir() uses a wrong length of name by strlen().

This patch passes the correct length to fat_ioctl_filldir(), and doesn't use nul terminate anymore.

Many helps by Alex Villacís Lasso. Thanks.

Signed-off-by: OGAWA Hirofumi <hirofumi@mail.parknet.co.jp>

```
fs/fat/dir.c | 138 ++++++-----
1 files changed, 78 insertions(+), 60 deletions(-)
diff -puN fs/fat/dir.c~fat_ioctl-utf8-fix fs/fat/dir.c
--- linux-2.6.10-rc1/fs/fat/dir.c~fat_ioctl-utf8-fix      2004-10-31 08:15:03.000000000 +0900
+++ linux-2.6.10-rc1-hirofumi/fs/fat/dir.c              2004-10-31 08:15:03.000000000 +0900
@@ -325,16 +325,28 @@ EODir:
     return res;
 }

```

```
+struct fat_ioctl_filldir_callback {
+    struct dirent __user *dirent;
+    int result;
+    /* for dir ioctl */
+    const char *longname;
+    int long_len;
+    const char *shortname;
+    int short_len;
+};
+
+static int fat_readdirx(struct inode *inode, struct file *filp, void *dirent,
-                        filldir_t filldir, int shortnames, int both)
+                        filldir_t filldir, int short_only, int both)
+{
+    struct super_block *sb = inode->i_sb;
+    struct buffer_head *bh;
+    struct msdos_dir_entry *de;
+    struct nls_table *nls_io = MSDOS_SB(sb)->nls_io;
+    struct nls_table *nls_disk = MSDOS_SB(sb)->nls_disk;

```

Linux-Kernel: [PATCH] FAT: fix VFAT_IOCTL_READDIR_BOTH ioctl

```

-     wchar_t bufuname[14];
-     unsigned char long_slots;
+     const char *fill_name;
+     int fill_len;
+     wchar_t bufuname[14];
+     wchar_t *unicode = NULL;
+     unsigned char c, work[8], bufname[56], *ptname = bufname;
+     unsigned long lpos, dummy, *furrfu = &lpos;
@@ -397,8 +409,7 @@ GetNew:
+     unsigned char alias_checksum;

+     if (!unicode) {
-         unicode = (wchar_t *)
-             __get_free_page(GFP_KERNEL);
+         unicode = (wchar_t *)__get_free_page(GFP_KERNEL);
+         if (!unicode) {
+             filp->f_pos = cpos;
+             brelse(bh);
@@ -533,26 +544,35 @@ ParseLong:
+             : unil6_to_x8(bufname, bufuname, uni_xlate, nls_io);
+         }

-     if (!long_slots || shortnames) {
-         if (both)
-             bufname[i] = '\0';
-         if (filldir(dirent, bufname, i, *furrfu, inum,
-             (de->attr & ATTR_DIR) ? DT_DIR : DT_REG) < 0)
-             goto FillFailed;
-     } else {
-         unsigned char longname[275];
+     fill_name = bufname;
+     fill_len = i;
+     if (!short_only && long_slots) {
+         /* convert the unicode long name. 261 is maximum size
+          * of unicode buffer. (13 * slots + nul) */
+         void *longname = unicode + 261;
+         int buf_size = PAGE_SIZE - (261 * sizeof(unicode[0]));
+         int long_len = utf8
-             ? utf8_wcstombs(longname, unicode, sizeof(longname))
-             : unil6_to_x8(longname, unicode, uni_xlate,
-                 nls_io);
+         if (both) {
+             memcpy(&longname[long_len+1], bufname, i);
+             long_len += i;
+             ? utf8_wcstombs(longname, unicode, buf_size)
+             : unil6_to_x8(longname, unicode, uni_xlate, nls_io);
+
+         if (!both) {
+             fill_name = longname;
+             fill_len = long_len;
+         } else {
+             /* hack for fat_ioctl_filldir() */
+             struct fat_ioctl_filldir_callback *p = dirent;
+
+             p->longname = longname;
+             p->long_len = long_len;
+             p->shortname = bufname;
+             p->short_len = i;
+             fill_name = NULL;
+             fill_len = 0;
+         }
-         if (filldir(dirent, longname, long_len, *furrfu, inum,

```

Linux-Kernel: [PATCH] FAT: fix VFAT_IOCTL_READDIR_BOTH ioctl

```

-             (de->attr & ATTR_DIR) ? DT_DIR : DT_REG) < 0)
-             goto FillFailed;
-         }
+         if (filldir(dirent, fill_name, fill_len, *furrfu, inum,
+             (de->attr & ATTR_DIR) ? DT_DIR : DT_REG) < 0)
+             goto FillFailed;

RecEnd:
    furrfu = &lpos;
@@ -563,9 +583,8 @@ EODir:
    FillFailed:
        if (bh)
            brelse(bh);
-        if (unicode) {
-            free_page((unsigned long) unicode);
-        }
+        if (unicode)
+            free_page((unsigned long)unicode);
    out:
        unlock_kernel();
        return ret;
@@ -577,49 +596,48 @@ static int fat_readdir(struct file *filp
    return fat_readdirx(inode, filp, dirent, filldir, 0, 0);
}

-struct fat_ioctl_filldir_callback {
-    struct dirent __user *dirent;
-    int result;
-};
-
-static int fat_ioctl_filldir(void *__buf, const char * name, int name_len,
+static int fat_ioctl_filldir(void *__buf, const char *name, int name_len,
                             loff_t offset, ino_t ino, unsigned int d_type)
    {
        struct fat_ioctl_filldir_callback *buf = __buf;
        struct dirent __user *d1 = buf->dirent;
        struct dirent __user *d2 = d1 + 1;
-        int len, slen;
-        int dotdir;

        if (buf->result)
            return -EINVAL;
        buf->result++;

-        if ((name_len == 1 && name[0] == '.') ||
-            (name_len == 2 && name[0] == '.' && name[1] == '.')) {
-            dotdir = 1;
-            len = name_len;
-        } else {
-            dotdir = 0;
-            len = strlen(name);
-        }
-        if (len != name_len) {
-            slen = name_len - len;
-            if (copy_to_user(d2->d_name, name, len) ||
-                put_user(0, d2->d_name + len) ||
-                put_user(len, &d2->d_reclen) ||
-                put_user(ino, &d2->d_ino) ||
-                put_user(offset, &d2->d_off) ||
-                copy_to_user(d1->d_name, name+len+1, slen) ||
-                put_user(0, d1->d_name+slen) ||
-                put_user(slen, &d1->d_reclen) ||

```

Linux-Kernel: [PATCH] FAT: fix VFAT_IOCTL_READDIR_BOTH ioctl

```

-         goto efault;
-     } else {
+     if (name != NULL) {
+         /* dirent has only short name */
+         if (name_len >= sizeof(d1->d_name))
+             name_len = sizeof(d1->d_name) - 1;
+
+         if (put_user(0, d2->d_name)           ||
+             put_user(0, &d2->d_reclen)       ||
-             copy_to_user(d1->d_name, name, len) ||
-             put_user(0, d1->d_name+len)      ||
-             put_user(len, &d1->d_reclen))
+             copy_to_user(d1->d_name, name, name_len) ||
+             put_user(0, d1->d_name + name_len) ||
+             put_user(name_len, &d1->d_reclen))
+                 goto efault;
+     } else {
+         /* dirent has short and long name */
+         const char *longname = buf->longname;
+         int long_len = buf->long_len;
+         const char *shortname = buf->shortname;
+         int short_len = buf->short_len;
+
+         if (long_len >= sizeof(d1->d_name))
+             long_len = sizeof(d1->d_name) - 1;
+         if (short_len >= sizeof(d1->d_name))
+             short_len = sizeof(d1->d_name) - 1;
+
+         if (copy_to_user(d2->d_name, longname, long_len) ||
+             put_user(0, d2->d_name + long_len)           ||
+             put_user(long_len, &d2->d_reclen)           ||
+             put_user(ino, &d2->d_ino)                   ||
+             put_user(offset, &d2->d_off)                 ||
+             copy_to_user(d1->d_name, shortname, short_len) ||
+             put_user(0, d1->d_name + short_len)         ||
+             put_user(short_len, &d1->d_reclen))
+                 goto efault;
+     }
    return 0;
@@ -633,15 +651,15 @@ static int fat_dir_ioctl(struct inode *
{
    struct fat_ioctl_filldir_callback buf;
    struct dirent __user *d1;
-   int ret, shortname, both;
+   int ret, short_only, both;

    switch (cmd) {
    case VFAT_IOCTL_READDIR_SHORT:
-       shortname = 1;
-       both = 1;
+       short_only = 1;
+       both = 0;
        break;
    case VFAT_IOCTL_READDIR_BOTH:
-       shortname = 0;
+       short_only = 0;
        both = 1;
        break;
    default:
@@ -665,7 +683,7 @@ static int fat_dir_ioctl(struct inode *
    ret = -ENOENT;
    if (!IS_DEADDIR(inode)) {

```

Linux-Kernel: [PATCH] FAT: fix VFAT_IOCTL_READDIR_BOTH ioctl

```
ret = fat_readdirx(inode, filp, &buf, fat_ioctl_filldir,  
- shortname, both);  
+ short_only, both);  
}  
up(&inode->i_sem);  
if (ret >= 0)
```

-
-
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/>