
Subject: [PATCH 3/3] lutimesat: actual syscall and wire-up on i386

Posted by [adobriyan](#) on Mon, 29 Jan 2007 09:52:02 GMT

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

On Sat, Jan 27, 2007 at 11:59:55PM -0800, H. Peter Anvin wrote:

> Alexey Dobriyan wrote:

> >+asmlinkage long sys_lutimesat(int dfd, char __user *filename, struct

> >timeval __user *utimes)

>

> Could we get these to take struct timespec instead of struct timeval?

>

> Right now we have a real problem in that the interfaces that *set* times

> take struct timeval (microsecond granularity) but the interfaces that

> *get* times return struct timespec (nanosecond granularity), which means

> information loss on any setting operations.

OK. XFS could use it.

[PATCH 3/3] lutimesat: actual syscall and wire-up on i386

lutimesat(2) does everything futimesat(2) does except it doesn't follow symlinks. It could be used by tar(1) and cp(1).

FreeBSD and NetBSD have lutimes(2) which can be emulated by C library.

lutimesat(2) accepts "struct timespec" which means timestamps with nanosecond granularity. Tested on XFS which has nanosecond timestamps on-disk.

Changes to do_utimes() which is used by all existing utime* syscalls pass LTP utime tests.

Closes http://bugme.osdl.org/show_bug.cgi?id=4433

Signed-off-by: Alexey Dobriyan <adobriyan@openvz.org>

```
arch/i386/kernel/syscall_table.S | 1 +
fs/utimes.c                       | 30 ++++++-----
include/asm-i386/unistd.h         | 4 ++-
3 files changed, 28 insertions(+), 7 deletions(-)
```

```
--- a/arch/i386/kernel/syscall_table.S
+++ b/arch/i386/kernel/syscall_table.S
@@ -319,3 +319,4 @@ ENTRY(sys_call_table)
 .long sys_move_pages
 .long sys_getcpu
 .long sys_epoll_pwait
```

```

+ .long sys_lutimesat /* 320 */
--- a/fs/utimes.c
+++ b/fs/utimes.c
@@ -40,7 +40,7 @@ #endif
 * must be owner or have write permission.
 * Else, update from *times, must be owner or super user.
 */
-long do_utimes(int dfd, char __user *filename, struct timeval *times, int flags)
+static long do_utimes_nsec(int dfd, char __user *filename, struct timespec *times, int flags)
{
    int error = -EINVAL;
    struct nameidata nd;
@@ -69,10 +69,8 @@ long do_utimes(int dfd, char __user *fil
        if (IS_APPEND(inode) || IS_IMMUTABLE(inode))
            goto dput_and_out;

-    newattrs.ia_atime.tv_sec = times[0].tv_sec;
-    newattrs.ia_atime.tv_nsec = times[0].tv_nsec * 1000;
-    newattrs.ia_mtime.tv_sec = times[1].tv_sec;
-    newattrs.ia_mtime.tv_nsec = times[1].tv_nsec * 1000;
+    newattrs.ia_atime = times[0];
+    newattrs.ia_mtime = times[1];
    newattrs.ia_valid |= ATTR_ATIME_SET | ATTR_MTIME_SET;
} else {
    error = -EACCES;
@@ -92,6 +90,19 @@ out:
    return error;
}

+long do_utimes(int dfd, char __user *filename, struct timeval *times, int flags)
+{
+    struct timespec ts[2];
+
+    if (times) {
+        ts[0].tv_sec = times[0].tv_sec;
+        ts[0].tv_nsec = times[0].tv_nsec * 1000;
+        ts[1].tv_sec = times[1].tv_sec;
+        ts[1].tv_nsec = times[1].tv_nsec * 1000;
+    }
+    return do_utimes_nsec(dfd, filename, times ? ts : NULL, flags);
+}
+
+asmlinkage long sys_futimesat(int dfd, char __user *filename, struct timeval __user *utimes)
+{
+    struct timeval times[2];
@@ -105,3 +116,12 @@ asmlinkage long sys_utimes(char __user *
{
    return sys_futimesat(AT_FDCWD, filename, utimes);
}

```

```

}
+
+asmlinkage long sys_lutimesat(int dfd, char __user *filename, struct timespec __user *utimes)
+{
+ struct timespec times[2];
+
+ if (utimes && copy_from_user(&times, utimes, sizeof(times)))
+ return -EFAULT;
+ return do_utimes_nsec(dfd, filename, utimes ? times : NULL, AT_SYMLINK_NOFOLLOW);
+}
--- a/include/asm-i386/unistd.h
+++ b/include/asm-i386/unistd.h
@@ -325,10 +325,11 @@ #define __NR_vmsplice 316
#define __NR_move_pages 317
#define __NR_getcpu 318
#define __NR_epoll_pwait 319
+#define __NR_lutimesat 320

#ifdef __KERNEL__

-#define NR_syscalls 320
+#define NR_syscalls 321

#define __ARCH_WANT_IPC_PARSE_VERSION
#define __ARCH_WANT_OLD_READDIR

```
