Subject: [PATCH 3/3] lutimesat: actual syscall and wire-up on i386
Posted by adobriyan on Fri, 26 Jan 2007 11:17:03 GMT
View Forum Message <> Reply to Message

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(AT_FDCWD, filename, utimes)

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 | 9 +++++++++
 include/asm-i386/unistd.h | 3 ++-
 3 files changed, 12 insertions(+), 1 deletion(-)

--- 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
@@ -105,3 +105,12 @@ asmlinkage long sys_utimes(char __user *
 {
  return sys_futimesat(AT_FDCWD, filename, utimes);
 }
+
+asmlinkage long sys_lutimesat(int dfd, char __user *filename, struct timeval __user *utimes)
+{
+ struct timeval times[2];
+
+ if (utimes && copy_from_user(&times, utimes, sizeof(times)))
+ return -EFAULT;
+ return do_utimes(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
```

Subject: Re: [PATCH 3/3] lutimesat: actual syscall and wire-up on i386
Posted by Andrew Morton on Fri, 26 Jan 2007 20:39:03 GMT
View Forum Message <> Reply to Message

On Fri, 26 Jan 2007 14:23:45 +0300
Alexey Dobriyan <adobriyan@openvz.org> wrote:

> 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(AT_FDCWD, filename, utimes)
>
> Closes http://bugme.osdl.org/show_bug.cgi?id=4433
>
efine __ARCH_WANT_IPC_PARSE_VERSION
> #define __ARCH_WANT_OLD_READDIR

OK, but I don't recall having seeing a demand for lutimes(). Opinions
are sought?

Subject: Re: [PATCH 3/3] lutimesat: actual syscall and wire-up on i386
Posted by Ulrich Drepper on Fri, 26 Jan 2007 20:45:20 GMT
View Forum Message <> Reply to Message

Andrew Morton wrote:
> OK, but I don't recall having seeing a demand for lutimes(). Opinions
> are sought?

It's an interface which has been available on other platforms forever (lutimes, not lutimesat). If it can be implemented correctly on the interesting file systems I'd say "go ahead", it can only be useful and have more benefits than the probably small cost of implementing it.

If on the other hand important filesystems cannot support lutimes then I'd wait with introducing the syscall at least until the support is added. It much easier to cope with unavailable syscalls then it is with partially working ones.

--

Subject: Re: [PATCH 3/3] lutimesat: actual syscall and wire-up on i386
Posted by hpa on Sun, 28 Jan 2007 07:59:55 GMT
View Forum Message <> Reply to Message

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.

-hpa

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_usec * 1000;
-  newattrs.ia_mtime.tv_sec = times[1].tv_sec;
-  newattrs.ia_mtime.tv_nsec = times[1].tv_usec * 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_usec * 1000;
+  ts[1].tv_sec = times[1].tv_sec;
+  ts[1].tv_nsec = times[1].tv_usec * 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

Subject: Re: [PATCH 3/3] lutimesat: actual syscall and wire-up on i386
Posted by adobriyan on Mon, 29 Jan 2007 10:59:40 GMT
View Forum Message <> Reply to Message

On Fri, Jan 26, 2007 at 12:45:20PM -0800, Ulrich Drepper wrote:
> Andrew Morton wrote:
> > OK, but I don't recall having seeing a demand for lutimes(). Opinions
> > are sought?
>
> It's an interface which has been available on other platforms forever
> (lutimes, not lutimesat). If it can be implemented correctly on the
> interesting file systems I'd say "go ahead", it can only be useful and
> have more benefits than the probably small cost of implementing it.
>
> If on the other hand important filesystems cannot support lutimes then
> I'd wait with introducing the syscall at least until the support is
> added.

What do you mean by "filesystems cannot support lutimes"? Filesystems
that don't have on-disk timestamps for symlinks?

Subject: Re: [PATCH 3/3] lutimesat: actual syscall and wire-up on i386
Posted by Ulrich Drepper on Mon, 29 Jan 2007 15:07:04 GMT
View Forum Message <> Reply to Message

Alexey Dobriyan wrote:
> What do you mean by "filesystems cannot support lutimes"? Filesystems
> that don't have on-disk timestamps for symlinks?

Yes.

--

Subject: Re: [PATCH 3/3] lutimesat: actual syscall and wire-up on i386
Posted by adobriyan on Mon, 29 Jan 2007 15:55:52 GMT
View Forum Message <> Reply to Message

On Mon, Jan 29, 2007 at 07:07:04AM -0800, Ulrich Drepper wrote:
> Alexey Dobriyan wrote:
> > What do you mean by "filesystems cannot support lutimes"? Filesystems
> > that don't have on-disk timestamps for symlinks?
>
> Yes.

Checked to be sure, on ext2, ext3, reiserfs, XFS symlink timestamps
stick across mounts/umounts.

Also, looking at disk inode structures of UFS, SysV, JFFS2, GFS2, EFS, I
think they should handle lutimesat(2) just fine.