



Rocky Enterprise Linux 9.2 Manual Pages on command 'futimens.3'

C:\>man futimens.3

UTIMENSAT(2) Linux Programmer's Manual UTIMENSAT(2)

NAME

utimensat, futimens - change file timestamps with nanosecond precision

SYNOPSIS

```
#include <fcntl.h> /* Definition of AT_* constants */
```

```
#include <sys/stat.h>
```

```
int utimensat(int dirfd, const char *pathname,  
              const struct timespec times[2], int flags);
```

```
int futimens(int fd, const struct timespec times[2]);
```

Feature Test Macro Requirements for glibc (see `feature_test_macros(7)`):

utimensat():

Since glibc 2.10:

```
  _POSIX_C_SOURCE >= 200809L
```

Before glibc 2.10:

```
  _ATFILE_SOURCE
```

futimens():

Since glibc 2.10:

```
  _POSIX_C_SOURCE >= 200809L
```

Before glibc 2.10:

```
  _GNU_SOURCE
```

DESCRIPTION

utimensat() and futimens() update the timestamps of a file with nanosecond preci?

sion. This contrasts with the historical `utime(2)` and `utimes(2)`, which permit only second and microsecond precision, respectively, when setting file timestamps.

With `utimensat()` the file is specified via the pathname given in `pathname`. With `futimens()` the file whose timestamps are to be updated is specified via an open file descriptor, `fd`.

For both calls, the new file timestamps are specified in the array `times`: `times[0]` specifies the new "last access time" (`atime`); `times[1]` specifies the new "last modification time" (`mtime`). Each of the elements of `times` specifies a time as the number of seconds and nanoseconds since the Epoch, 1970-01-01 00:00:00 +0000 (UTC).

This information is conveyed in a structure of the following form:

```
struct timespec {
    time_t tv_sec;    /* seconds */
    long tv_nsec;    /* nanoseconds */
};
```

Updated file timestamps are set to the greatest value supported by the filesystem that is not greater than the specified time.

If the `tv_nsec` field of one of the `timespec` structures has the special value `UTIME_NOW`, then the corresponding file timestamp is set to the current time. If the `tv_nsec` field of one of the `timespec` structures has the special value `UTIME_OMIT`, then the corresponding file timestamp is left unchanged. In both of these cases, the value of the corresponding `tv_sec` field is ignored.

If `times` is `NULL`, then both timestamps are set to the current time.

Permissions requirements

To set both file timestamps to the current time (i.e., `times` is `NULL`, or both `tv_nsec` fields specify `UTIME_NOW`), either:

1. the caller must have write access to the file;
2. the caller's effective user ID must match the owner of the file; or
3. the caller must have appropriate privileges.

To make any change other than setting both timestamps to the current time (i.e., `times` is not `NULL`, and neither `tv_nsec` field is `UTIME_NOW` and neither `tv_nsec` field is `UTIME_OMIT`), either condition 2 or 3 above must apply.

If both `tv_nsec` fields are specified as `UTIME_OMIT`, then no file ownership or permission checks are performed, and the file timestamps are not modified, but other

error conditions may still be detected.

utimensat() specifics

If `pathname` is relative, then by default it is interpreted relative to the directory referred to by the open file descriptor, `dirfd` (rather than relative to the current working directory of the calling process, as is done by `utimes(2)` for a relative pathname). See `openat(2)` for an explanation of why this can be useful.

If `pathname` is relative and `dirfd` is the special value `AT_FDCWD`, then `pathname` is interpreted relative to the current working directory of the calling process (like `utimes(2)`).

If `pathname` is absolute, then `dirfd` is ignored.

The flags field is a bit mask that may be 0, or include the following constant, defined in `<fcntl.h>`:

AT_SYMLINK_NOFOLLOW

If `pathname` specifies a symbolic link, then update the timestamps of the link, rather than the file to which it refers.

RETURN VALUE

On success, `utimensat()` and `futimens()` return 0. On error, -1 is returned and `errno` is set to indicate the error.

ERRORS

EACCES `times` is NULL, or both `tv_nsec` values are `UTIME_NOW`, and either:

- * the effective user ID of the caller does not match the owner of the file, the caller does not have write access to the file, and the caller is not privileged (Linux: does not have either the `CAP_FOWNER` or the `CAP_DAC_OVERRIDE` capability); or,
- * the file is marked immutable (see `chattr(1)`).

EBADF (`futimens()`) `fd` is not a valid file descriptor.

EBADF (`utimensat()`) `pathname` is a relative pathname, but `dirfd` is neither `AT_FDCWD` nor a valid file descriptor.

EFAULT `times` pointed to an invalid address; or, `dirfd` was `AT_FDCWD`, and `pathname` is NULL or an invalid address.

EINVAL Invalid value in flags.

EINVAL Invalid value in one of the `tv_nsec` fields (value outside range 0 to 999,999,999, and not `UTIME_NOW` or `UTIME_OMIT`); or an invalid value in one of

the tv_sec fields.

EINVAL pathname is NULL, dirfd is not AT_FDCWD, and flags contains AT_SYMLINK_NO?

FOLLOW.

ELOOP (utimensat()) Too many symbolic links were encountered in resolving path?

name.

ENAMETOOLONG

(utimensat()) pathname is too long.

ENOENT (utimensat()) A component of pathname does not refer to an existing direc?

tory or file, or pathname is an empty string.

ENOTDIR

(utimensat()) pathname is a relative pathname, but dirfd is neither AT_FDCWD

nor a file descriptor referring to a directory; or, one of the prefix compo?

nents of pathname is not a directory.

EPERM The caller attempted to change one or both timestamps to a value other than

the current time, or to change one of the timestamps to the current time

while leaving the other timestamp unchanged, (i.e., times is not NULL, nei?

ther tv_nsec field is UTIME_NOW, and neither tv_nsec field is UTIME_OMIT)

and either:

- * the caller's effective user ID does not match the owner of file, and the caller is not privileged (Linux: does not have the CAP_FOWNER capabil? ity); or,
- * the file is marked append-only or immutable (see chattr(1)).

EROFS The file is on a read-only filesystem.

ESRCH (utimensat()) Search permission is denied for one of the prefix components

of pathname.

VERSIONS

utimensat() was added to Linux in kernel 2.6.22; glibc support was added with ver?

sion 2.6.

Support for futimens() first appeared in glibc 2.6.

ATTRIBUTES

For an explanation of the terms used in this section, see attributes(7).

??

?Interface ? Attribute ? Value ?

??

?utimensat(), futimens() ? Thread safety ? MT-Safe ?

??

CONFORMING TO

futimens() and utimensat() are specified in POSIX.1-2008.

NOTES

utimensat() obsoletes futimesat(2).

On Linux, timestamps cannot be changed for a file marked immutable, and the only change permitted for files marked append-only is to set the timestamps to the current time. (This is consistent with the historical behavior of utime(2) and utimes(2) on Linux.)

If both tv_nsec fields are specified as UTIME_OMIT, then the Linux implementation of utimensat() succeeds even if the file referred to by dirfd and pathname does not exist.

C library/kernel ABI differences

On Linux, futimens() is a library function implemented on top of the utimensat() system call. To support this, the Linux utimensat() system call implements a non-standard feature: if pathname is NULL, then the call modifies the timestamps of the file referred to by the file descriptor dirfd (which may refer to any type of file). Using this feature, the call futimens(fd, times) is implemented as:

```
utimensat(fd, NULL, times, 0);
```

Note, however, that the glibc wrapper for utimensat() disallows passing NULL as the value for pathname: the wrapper function returns the error EINVAL in this case.

BUGS

Several bugs afflict utimensat() and futimens() on kernels before 2.6.26. These bugs are either nonconformances with the POSIX.1 draft specification or inconsistencies with historical Linux behavior.

* POSIX.1 specifies that if one of the tv_nsec fields has the value UTIME_NOW or UTIME_OMIT, then the value of the corresponding tv_sec field should be ignored. Instead, the value of the tv_sec field is required to be 0 (or the error EINVAL results).

* Various bugs mean that for the purposes of permission checking, the case where both tv_nsec fields are set to UTIME_NOW isn't always treated the same as speci?

fyng times as NULL, and the case where one tv_nsec value is UTIME_NOW and the other is UTIME_OMIT isn't treated the same as specifying times as a pointer to an array of structures containing arbitrary time values. As a result, in some cases: a) file timestamps can be updated by a process that shouldn't have permission to perform updates; b) file timestamps can't be updated by a process that should have permission to perform updates; and c) the wrong errno value is returned in case of an error.

* POSIX.1 says that a process that has write access to the file can make a call with times as NULL, or with times pointing to an array of structures in which both tv_nsec fields are UTIME_NOW, in order to update both timestamps to the current time. However, futimens() instead checks whether the access mode of the file descriptor allows writing.

SEE ALSO

chattr(1), touch(1), futimesat(2), openat(2), stat(2), utimes(2), futimes(3), inode(7), path_resolution(7), symlink(7)

COLOPHON

This page is part of release 5.05 of the Linux man-pages project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at <https://www.kernel.org/doc/man-pages/>.

Linux

2017-09-15

UTIMENSAT(2)