utimensat(2) — Linux manual page

NAME top

utimensat, futimens - change file timestamps with nanosecond precision

SYNOPSIS top

```
#include <fcntl.h> /* Определение констант AT_* */
#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]);
```

Требования к макросу Feature Test для glibc (см. feature_test_macros(7)):

utimensat():
Начиная с glibc 2.10:
_POSIX_C_SOURCE >= 200809L

До glibc 2.10:
_ATFILE_SOURCE

futimens():
Начиная с glibc 2.10:
_POSIX_C_SOURCE >= 200809L
До glibc 2.10:
_GNU_SOURCE

ОПИСАНИЕ top

utimensat() и **futimens**() обновляют временные метки файла с точностью до наносекунды. Это контрастирует с историческим utime(2) иutimes(2), которые допускают только секундную и микросекундную точность соответственно при установке меток времени файла.

С **помощью utimensat**() файл указывается через путь, указанный в поле *pathname*. С **помощью**функции futimens() файл, временные метки которого должны быть обновлены, указывается через открытый файловый дескриптор *fd*.

Для обоих вызовов новые метки времени файла указываются в массиве *times*: *times[0]* задает новое "время последнего доступа" (*atime*); *times[1]* указывает новое "время последней модификации". (*mtime*). Каждый из элементов *времен* определяет время как количество секунд и наносекунд с момента эпохи, 1970-01-01 00:00:00 +0000 (UTC). Эта информация передается в следующей форме:

```
struct timespec {
time_t tv_sec; /* секунды */
long tv_nsec; /* наносекунды */
};
```

Обновленные временные метки файлов устанавливаются на наибольшее значение, поддерживаемое файловой системой, которое не превышает указанное время.

Если *поле* tv_nsec одной из структур *timespec* имеет специальное значение **UTIME_NOW**затем соответствующая метка времени файла устанавливается на текущее время. Если *поле* tv_nsec одной из структур *timespec* имеет специальное значение **UTIME_OMIT**, то соответствующая метка времени файла остается неизменной. В обоих случаях значение соответствующего поля tv_sec игнорируется.

Если *times* равно NULL, то обе метки времени устанавливаются на текущее время.

Требования к разрешениям
Чтобы установить обе метки времени файла на текущее время (*т.Е.* NULL или оба поля tv_nsec указывают **UTIME_NOW**), либо:

1. вызывающий абонент должен иметь доступ на запись к файлу;

2. эффективный идентификатор вызывающего абонента должен совпадать с владельцем файла; или

3. вызывающий абонент должен иметь соответствующие привилегии.

Чтобы внести какие-либо изменения, кроме установки обеих меток времени на текущее время (т.Е. *times* не равно NULL , и ни одно поле tv_nsec не является **UTIME_NOW,** и ни одно поле tv_nsec не является **UTIME_OMIT**), должно применяться условие 2 или 3 выше.

Если оба поля tv_nsec заданы как **UTIME_OMIT**, тогда никакие проверки владения файлом или разрешений не выполняются, и временные метки файла не изменяются, но другие условия ошибки все еще могут быть обнаружены.

особенности utimensat()
Если *путь* относительный, то по умолчанию он интерпретируется относительно каталога, на который ссылается открытый файловый

дескриптор, *dirfd* (а не относительно текущего рабочего каталога вызывающего процесса, как это делается utimes(2) для относительного пути). См. openat(2) для объяснения того, почему это может быть полезно.

Если *путь* является относительным и *dirfd* - это специальное значение **AT_FDCWD**, тогда *путь* интерпретируется относительно текущего рабочего каталога вызывающего процесса (например, utimes(2)).

Если *pathname* является абсолютным, то *dirfd* игнорируется.

Поле *flags* - это битовая маска, которая может быть равна 0 или включать следующую константу, определенную в *<fcntl.h>*:

AT_SYMLINK_NOFOLLOW
Если *pathname* указывает символическую ссылку, обновите временные метки ссылки, а не файл, на который она ссылается.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ top

При успешном **выполнении utimensat**() и **futimens**() возвращают 0. При ошибке возвращается значение -1, а для указания ошибки устанавливается значение errno.

ОШИБКИ сверху

EACCES *times* равно NULL , или оба значения tv_nsec равны **UTIME_NOW**, а эффективный идентификатор пользователя вызывающего абонента не соответствует владельцу файла, вызывающий абонент не имеет доступа на запись к файлу, а вызывающий абонент не имеет привилегий (Linux: не имеет ни **CAP_FOWNER**, ни **CAP_DAC_OVERRIDE** возможность).

EBADF (**futimens**()) *fd* не является допустимым файловым дескриптором.

Путь EBADF (**utimensat**()) относителен, *но dirfd* не является ни **AT_FDCWD**, ни допустимым файловым дескриптором.

EFAULT *times* указывает на недопустимый адрес; или *dirfd* был **AT_FDCWD**, а *pathname* – NULL или недопустимый адрес.

Недопустимое значение EINVAL во *флагах*.

EINVAL Недопустимое значение в одном из полей *tv_nsec* (значение вне диапазона от 0 до 999,999,999, а не **UTIME_NOW** или **UTIME_OMIT**);
или недопустимое значение в одном из полей *tv_sec*.

Путь EINVAL равен NULL, *dirfd* не **AT_FDCWD**, а *флаги*
содержит **AT_SYMLINK_NOFOLLOW**.

ELOOP (**utimensat**()) При разрешении пути было обнаружено слишком много символических ссылок
.

ENAMETOOLONG
(**utimensat**()) *путь* слишком длинный.

ENOENT (**utimensat**()) Компонент *pathname* не ссылается на существующий каталог или файл, или *pathname* является пустой строкой.

ENOTDIR
(**utimensat**()) *pathname* – это относительный путь, но *dirfd*
не является ни **AT_FDCWD**, ни файловым дескриптором, ссылающимся на каталог; или один из префиксных компонентов *pathname* не является каталогом.

EPERM Вызывающий абонент попытался изменить одну или обе метки времени на значение, отличное от текущего времени, или изменить одну из меток времени на текущее время, оставив другую

метку времени неизменной (т.е. *times* не равно NULL, ни
одно поле tv_nsec не является **UTIME_NOW**, ни одно поле tv_nsec не является
UTIME_OMIT).:

* эффективный идентификатор пользователя вызывающего абонента не совпадает с владельцем
файла, и вызывающий абонент не имеет привилегий (Linux:
не имеет **CAP_FOWNER** возможности); или,

* файл помечен как только для добавления или неизменяемый (см.
chattr(1)).

EROFS Файл находится в файловой системе, доступной только для чтения.

ESRCH (**utimensat**()) Отказано в разрешении поиска для одного из
префиксных компонентов *pathname*.

ВЕРСИИ top

utimensat() был добавлен в Linux в ядре 2.6.22; поддержка glibc
была добавлена в версии 2.6.

Поддержка **futimens**() впервые появилась в glibc 2.6.

АТРИБУТЫ сверху

Для объяснения терминов, используемых в этом разделе, см.
атрибуты(7).

Интерфейс	Атрибут	Значение
utimensat(), **futimens**()	Thread safety	MT-Safe

СООТВЕТСТВУЕТ top

futimens() и **utimensat**() указаны в POSIX.1-2008.

ПРИМЕЧАНИЯ вверху

utimensat() устарел futimesat(2).

В Linux временные метки не могут быть изменены для файла с пометкой immutable, и единственное изменение, разрешенное для файлов с пометкой append-only, - это установка временных меток на текущее время. (Это согласуется с историческим поведением utime(2) и utimes(2). в Linux.)

Если оба поля tv_nsec указаны как **UTIME_OMIT**, то реализация **utimensat**() в Linux выполняется успешно, даже если файл , на который ссылаются *dirfd* и *pathname*, не существует.

C библиотека / ядро ABI различия
В Linux, **futimens**() - это библиотечная функция, реализованная поверх системного вызова utimensat(). Чтобы поддержать это, Linux системный вызов utimensat() реализует нестандартную функцию: если *pathname* равно NULL, то вызов изменяет временные метки файла, на который ссылается файловый дескриптор *dirfd* (который может ссылаться на любой тип файла). Используя эту функцию, вызов *futimens(fd, times)* реализован как:

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

Однако обратите внимание, что оболочка glibc для **utimensat**() запрещает передачу NULL в качестве значения для *pathname*:
в этом случае функция-оболочка возвращает ошибку EINVAL.

BUGS top

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 specifying *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 top

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

COLOPHON top

This page is part of release 5.13 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 2021-08-27 UTIMENSAT(2)

Pages that refer to this page: fcntl(2), futimesat(2), open(2), syscalls(2), utime(2), futimes(3), inotify(7), signal-safety(7), symlink(7), xfs_io(8)