NAME

utimes, **lutimes**, **futimes**, **futimesat** - set file access and modification times

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <sys/time.h>

int
utimes(*const char *path, const struct timeval *times*);

int
lutimes(*const char *path, const struct timeval *times*);

int
futimes(*int fd, const struct timeval *times*);

int
futimesat(*int fd, const char *path, const struct timeval times[2]*);

DESCRIPTION

These interfaces are obsoleted by futimens(2) and utimensat(2) because they are not accurate to nanoseconds.

The access and modification times of the file named by *path* or referenced by *fd* are changed as specified by the argument *times*.

If *times* is NULL, the access and modification times are set to the current time. The caller must be the owner of the file, have permission to write the file, or be the super-user.

If *times* is non-NULL, it is assumed to point to an array of two timeval structures. The access time is set to the value of the first element, and the modification time is set to the value of the second element. For file systems that support file birth (creation) times (such as UFS2), the birth time will be set to the value of the second element if the second element is older than the currently set birth time. To set both a birth time and a modification time, two calls are required; the first to set the birth time and the second to set the (presumably newer) modification time. Ideally a new system call will be added that allows the setting of all three times at once. The caller must be the owner of the file or be the super-user.

In either case, the inode-change-time of the file is set to the current time.

The **lutimes**() system call is like **utimes**() except in the case where the named file is a symbolic link, in which case **lutimes**() changes the access and modification times of the link, while **utimes**() changes the times of the file the link references.

The **futimesat**() system call is equivalent to **utimes**() except in the case where *path* specifies a relative path. In this case the access and modification time is set to that of a file relative to the directory associated with the file descriptor *fd* instead of the current working directory. If **futimesat**() is passed the special value AT_FDCWD in the *fd* parameter, the current working directory is used and the behavior is identical to a call to **utimes**().

RETURN VALUES

Upon successful completion, the value 0 is returned; otherwise the value -1 is returned and the global variable *errno* is set to indicate the error.

ERRORS

All of the system call will fail if:

[EACCES] Search permission is denied for a component of the path prefix.

[EACCES] The *times* argument is NULL and the effective user ID of the process does not match the owner of the file, and is not the super-user, and write access is denied.

[EFAULT] The *path* or *times* argument points outside the process's allocated address space.

[EFAULT] The *times* argument points outside the process's allocated address space.

[EINVAL] The *tv_usec* component of at least one of the values specified by the *times* argument has a value less than 0 or greater than 999999.

[EIO] An I/O error occurred while reading or writing the affected inode.

[EINTEGRITY] Corrupted data was detected while reading from the file system.

[ELOOP] Too many symbolic links were encountered in translating the pathname.

[ENAMETOOLONG] A component of a pathname exceeded NAME_MAX characters, or an entire path name exceeded PATH_MAX characters.

[ENOENT] The named file does not exist.

[ENOTDIR] A component of the path prefix is not a directory.

[EPERM] The *times* argument is not NULL and the calling process's effective user ID does not match the owner of the file and is not the super-user.

[EPERM] The named file has its immutable or append-only flags set. See the chflags(2) manual page for more information.

[EROFS] The file system containing the file is mounted read-only.

The **futimes**() system call will fail if:

[EBADF] The *fd* argument does not refer to a valid descriptor.

In addition to the errors returned by the **utimes**(), the **futimesat**() may fail if:

[EBADF] The *path* argument does not specify an absolute path and the *fd* argument is neither AT_FDCWD nor a valid file descriptor open for searching.

[ENOTDIR] The *path* argument is not an absolute path and *fd* is neither AT_FDCWD nor a file descriptor associated with a directory.

SEE ALSO

chflags(2), stat(2), utimensat(2), utime(3)

STANDARDS

The **utimes**() function is expected to conform to X/Open Portability Guide Issue 4, Version 2 ("XPG4.2"). The **futimesat**() system call follows The Open Group Extended API Set 2 specification but was replaced by **utimensat**() in IEEE Std 1003.1-2008 ("POSIX.1").

HISTORY

The **utimes**() system call appeared in 4.2BSD. The **futimes**() and **lutimes**() system calls first appeared in FreeBSD 3.0. The **futimesat**() system call appeared in FreeBSD 8.0.