Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'futimens.3p' command

$ man futimens.3p

FUTIMENS(3P) POSIX Programmer's Manual FUTIMENS(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

futimens, utimensat, utimes ? set file access and modification times

SYNOPSIS

```
#include <sys/stat.h>
int futimens(int fd, const struct timespec times[2]);
#include <fcntl.h>
int utimensat(int fd, const char *path, const struct timespec times[2],
    int flag);
#include <sys/time.h>
int utimes(const char *path, const struct timeval times[2]);
```

DESCRIPTION

The futimens() and utimensat() functions shall set the access and modi? fication times of a file to the values of the times argument. The futi? mens() function changes the times of the file associated with the file descriptor fd. The utimensat() function changes the times of the file pointed to by the path argument, relative to the directory associated with the file descriptor fd. Both functions allow time specifications

accurate to the nanosecond.

For futimens() and utimensat(), the times argument is an array of two timespec structures. The first array member represents the date and time of last access, and the second member represents the date and time of last modification. The times in the timespec structure are measured in seconds and nanoseconds since the Epoch. The file's relevant time? stamp shall be set to the greatest value supported by the file system that is not greater than the specified time.

If the tv_nsec field of a timespec structure has the special value UTIME_NOW, the file's relevant timestamp shall be set to the greatest value supported by the file system that is not greater than the current time. If the tv_nsec field has the special value UTIME_OMIT, the file's relevant timestamp shall not be changed. In either case, the tv_sec field shall be ignored.

If the times argument is a null pointer, both the access and modifica? tion timestamps shall be set to the greatest value supported by the file system that is not greater than the current time. If utimensat() is passed a relative path in the path argument, the file to be used shall be relative to the directory associated with the file descriptor fd instead of the current working directory. If the access mode of the open file description associated with the file descriptor is not O_SEARCH, the function shall check whether directory searches are per? mitted using the current permissions of the directory underlying the file descriptor. If the access mode is O_SEARCH, the function shall not perform the check.

If utimensat() is passed the special value AT_FDCWD in the fd parame? ter, the current working directory shall be used.

Only a process with the effective user ID equal to the user ID of the file, or with write access to the file, or with appropriate privileges may use futimens() or utimensat() with a null pointer as the times ar? gument or with both tv_nsec fields set to the special value UTIME_NOW.

Only a process with the effective user ID equal to the user ID of the file or with appropriate privileges may use futimens() or utimensat()

with a non-null times argument that does not have both tv_nsec fields set to UTIME_NOW and does not have both tv_nsec fields set to UTIME_OMIT. If both tv_nsec fields are set to UTIME_OMIT, no ownership or permissions check shall be performed for the file, but other error conditions may still be detected (including [EACCES] errors related to the path prefix).

Values for the flag argument of utimensat() are constructed by a bit?wise-inclusive OR of flags from the following list, defined in <fc?ntl.h>:

AT_SYMLINK_NOFOLLOW

If path names a symbolic link, then the access and modification times of the symbolic link are changed.

Upon successful completion, futimens() and utimensat() shall mark the last file status change timestamp for update, with the exception that if both tv_nsec fields are set to UTIME_OMIT, the file status change timestamp need not be marked for update.

The utimes() function shall be equivalent to the utimensat() function with the special value AT_FDCWD as the fd argument and the flag argu?ment set to zero, except that the times argument is a timeval structure rather than a timespec structure, and accuracy is only to the microsec?ond, not nanosecond, and rounding towards the nearest second may occur.

RETURN VALUE

Upon successful completion, these functions shall return 0. Otherwise, these functions shall return -1 and set errno to indicate the error. If -1 is returned, the file times shall not be affected.

ERRORS

These functions shall fail if:

EACCES The times argument is a null pointer, or both tv_nsec values are UTIME_NOW, and the effective user ID of the process does not match the owner of the file and write access is denied.

EINVAL Either of the times argument structures specified a tv_nsec value that was neither UTIME_NOW nor UTIME_OMIT, and was a value less than zero or greater than or equal to 1000 million.

EINVAL A new file timestamp would be a value whose tv_sec component is not a value supported by the file system.

EPERM The times argument is not a null pointer, does not have both tv_nsec fields set to UTIME_NOW, does not have both tv_nsec fields set to UTIME_OMIT, the calling process' effective user ID does not match the owner of the file, and the calling process does not have appropriate privileges.

EROFS The file system containing the file is read-only.

The futimens() function shall fail if:

EBADF The fd argument is not a valid file descriptor.

The utimensat() function shall fail if:

EACCES The access mode of the open file description associated with fd is not O_SEARCH and the permissions of the directory underlying fd do not permit directory searches.

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 reading or searching.

ENOTDIR

The path argument is not an absolute path and fd is a file de? scriptor associated with a non-directory file.

The utimensat() and utimes() functions shall fail if:

EACCES Search permission is denied by a component of the path prefix.

ELOOP A loop exists in symbolic links encountered during resolution of the path argument.

ENAMETOOLONG

The length of a component of a pathname is longer than {NAME_MAX}.

ENOENT A component of path does not name an existing file or path is an empty string.

ENOTDIR

A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the path argument contains at least one non-<slash> character and

ends with one or more trailing <slash> characters and the last pathname component names an existing file that is neither a di? rectory nor a symbolic link to a directory.

The utimensat() and utimes() functions may fail if:

ELOOP More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the path argument.

ENAMETOOLONG

The length of a pathname exceeds {PATH_MAX}, or pathname resolu? tion of a symbolic link produced an intermediate result with a length that exceeds {PATH_MAX}.

The utimensat() function may fail if:

EINVAL The value of the flag argument is not valid.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

The purpose of the utimensat() function is to set the access and modi? fication time of files in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to utimes(), resulting in unspecified behavior. By opening a file descriptor for the target di? rectory and using the utimensat() function it can be guaranteed that the changed file is located relative to the desired directory.

The standard developers considered including a special case for the permissions required by utimensat() when one tv_nsec field is UTIME_NOW and the other is UTIME_OMIT. One possibility would be to include this case in with the cases where times is a null pointer or both fields are UTIME_NOW, where the call is allowed if the process has write permis? sion for the file. However, associating write permission with an update to just the last data access timestamp (which is normally updated by read()) did not seem appropriate. The other possibility would be to

specify that this one case is allowed if the process has read permis?
sion, but this was felt to be too great a departure from the utime()
and utimes() functions on which utimensat() is based. If an application
needs to set the last data access timestamp to the current time for a
file on which it has read permission but is not the owner, it can do so
by opening the file, reading one or more bytes (or reading a directory
entry, if the file is a directory), and then closing it.

FUTURE DIRECTIONS

None.

SEE ALSO

read(), utime()

The Base Definitions volume of POSIX.1?2017, <fcntl.h>, <sys_stat.h>,
<sys_time.h>

COPYRIGHT