NAME

timerfd, **timerfd_create**, **timerfd_gettime**, **timerfd_settime** - timers with file descriptor semantics

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <sys/timerfd.h>

int
timerfd_create(*int clockid, int flags*);

int
timerfd_gettime(*int fd, struct itimerspec *curr_value*);

int
timerfd_settime(*int fd, int flags, const struct itimerspec *new_value, struct itimerspec *old_value*);

DESCRIPTION

The **timerfd** system calls operate on timers, identified by special **timerfd** file descriptors. These calls are analogous to **timer_create**(), **timer_gettime**(), and **timer_settime**() per-process timer functions, but use a **timerfd** descriptor in place of *timerid*.

All **timerfd** descriptors possess traditional file descriptor semantics; they may be passed to other processes, preserved across fork(2), and monitored via kevent(2), poll(2), or select(2). When a **timerfd** descriptor is no longer needed, it may be disposed of using close(2).

timerfd_create() Initialize a **timerfd** object and return its file descriptor. The *clockid* argument specifies the clock used as a timing base and may be:

CLOCK_REALTIME	Increments as a wall clock should.
CLOCK_MONOTONIC	Increments monotonically in SI seconds.

The *flags* argument may contain the result of *or'*ing the following values:

TFD_CLOEXEC	The newly generated file descriptor will close-on-exec.
TFD_NONBLOCK	Do not block on read/write operations.

timerfd_gettime() Retrieve the current state of the timer denoted by *fd*. The result is stored in *curr_value* as a struct itimerspec. The *it_value* and *it_interval* members of

curr_value represent the relative time until the next expiration and the interval reload value last set by **timerfd_settime**(), respectively.

timerfd_settime() Update the timer denoted by *fd* with the struct itimerspec in *new_value*. The *it_value* member of *new_value* should contain the amount of time before the timer expires, or zero if the timer should be disarmed. The *it_interval* member should contain the reload time if an interval timer is desired.

The previous timer state will be stored in *old_value* given *old_value* is not NULL.

The *flags* argument may contain the result of *or*'ing the following values:

TFD_TIMER_ABSTIME	Expiration will occur at the absolute time provided in *new_value*. Normally, *new_value* represents a relative time compared to the timer's *clockid* clock.
TFD_TIMER_CANCEL_ON_SET	If *clockid* has been set to CLOCK_REALTIME and the realtime clock has experienced a discontinuous jump, then the timer will be canceled and the next read(2) will fail with ECANCELED.

File operations have the following semantics:

read(2)

Transfer the number of timer expirations that have occurred since the last successful read(2) or **timerfd_settime**() into the output buffer of size *uint64_t*. If the expiration counter is zero, read(2) blocks until a timer expiration occurs unless TFD_NONBLOCK is set, where EAGAIN is returned.

poll(2)

The file descriptor is readable when its timer expiration counter is greater than zero.

ioctl(2)

FIOASYNC int

A non-zero input will set the FASYNC flag. A zero input will clear the FASYNC flag.

FIONBIO int

A non-zero input will set the FNONBLOCK flag. A zero input will clear the FNONBLOCK flag.

RETURN VALUES

The **timerfd_create**() system call creates a **timerfd** object and returns its file descriptor. If an error occurs, -1 is returned and the global variable *errno* is set to indicate the error.

The **timerfd_gettime**() and **timerfd_settime**() system calls return 0 on success. If an error occurs, -1 is returned and the global variable *errno* is set to indicate the error.

ERRORS

The **timerfd_create**() system call fails if:

[EINVAL] The specified *clockid* is not supported.

[EINVAL] The provided *flags* are invalid.

[EMFILE] The per-process descriptor table is full.

[ENFILE] The system file table is full.

[ENOMEM] The kernel failed to allocate enough memory for the timer.

Both **timerfd_gettime**() and **timerfd_settime**() system calls fail if:

[EBADF] The provided *fd* is invalid.

[EFAULT] The addresses provided by *curr_value*, *new_value*, or *old_value* are invalid.

[EINVAL] The provided *fd* is valid, but was not generated by **timerfd_create**().

The following errors only apply to **timerfd_settime**():

[EINVAL] The provided *flags* are invalid.

[EINVAL] A nanosecond field in the *new_value* argument specified a value less than zero, or greater than or equal to 10^9.

[ECANCELED] The timer was created with the clock ID CLOCK_REALTIME, was configured with the TFD_TIMER_CANCEL_ON_SET flag, and the system realtime clock

experienced a discontinuous change without being read.

A read from a **timerfd** object fails if:

[EAGAIN] The timer's expiration counter is zero and the **timerfd** object is is set for non-blocking I/O.

[ECANCELED] The timer was created with the clock ID CLOCK_REALTIME, was configured with the TFD_TIMER_CANCEL_ON_SET flag, and the system realtime clock experienced a discontinuous change.

[EINVAL] The size of the read buffer is not large enough to hold the *uint64_t* sized timer expiration counter.

SEE ALSO

eventfd(2), kqueue(2), poll(2), read(2), timer_create(2), timer_gettime(2), timer_settime(2)

STANDARDS

The **timerfd** system calls originated from Linux and are non-standard.

HISTORY

The **timerfd** facility was originally ported to FreeBSD's Linux compatibility layer by Dmitry Chagin <*dchagin@FreeBSD.org*> in FreeBSD 12.0. It was revised and adapted to be native by Jake Freeland <*jfree@FreeBSD.org*> in FreeBSD 14.0.