NAME

timer_settime, timer_gettime − arm/disarm and fetch state of POSIX per-process timer

SYNOPSIS

```
#include <time.h>

int timer_settime(timer_t timerid, int flags,
                  const struct itimerspec *new_value,
                  struct itimerspec * old_value);
int timer_gettime(timer_t timerid, struct itimerspec *curr_value);
```

Link with −*lrt*.

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

timer_settime(), **timer_gettime**(): _POSIX_C_SOURCE >= 199309

DESCRIPTION

timer_settime() arms or disarms the timer identified by *timerid*. The *new_value* argument is an *itimerspec* structure that specifies the new initial value and the new interval for the timer. The *itimerspec* structure is defined as follows:

```
struct timespec {
   time_t tv_sec;            /* Seconds */
   long   tv_nsec;           /* Nanoseconds */
};

struct itimerspec {
   struct timespec it_interval;  /* Timer interval */
   struct timespec it_value;     /* Initial expiration */
};
```

Each of the substructures of the *itimerspec* structure is a *timespec* structure that allows a time value to be specified in seconds and nanoseconds. These time values are measured according to the clock that was specified when the timer was created by **timer_create**()

If *new_value->it_value* specifies a non-zero value (i.e., either subfield is non-zero), then **timer_settime**() arms (starts) the timer, setting it to initially expire at the given time. (If the timer was already armed, then the previous settings are overwritten.) If *new_value->it_value* specifies a zero value (i.e., both subfields are zero), then the timer is disarmed.

The *new_value->it_interval* field specifies the period of the timer, in seconds and nanoseconds. If this field is non-zero, then each time that an armed timer expires, the timer is reloaded from the value specified in *new_value->it_interval*. If *new_value->it_interval* specifies a zero value then the timer expires just once, at the time specified by *it_value*.

By default, the initial expiration time specified in *new_value->it_value* is interpreted relative to the current time on the timer's clock at the time of the call. This can be modified by specifying **TIMER_ABSTIME** in *flags*, in which case *new_value->it_value* is interpreted as an absolute value as measured on the timer's clock; that is, the timer will expire when the clock value reaches the value specified by *new_value->it_value*. If the specified absolute time has already passed, then the timer expires immediately, and the overrun count (see **timer_getoverrun**(2)) will be set correctly.

If the value of the **CLOCK_REALTIME** clock is adjusted while an absolute timer based on that clock is armed, then the expiration of the timer will be appropriately adjusted. Adjustments to the **CLOCK_REALTIME** clock have no effect on relative timers based on that clock.

If *old_value* is not NULL, then it returns the previous interval of the timer (in *old_value->it_interval*) and the amount of time until the timer would previously have next expired (in *old_value->it_value*).

timer_gettime() returns the time until next expiration, and the the interval, for the timer specified by *timerid*, in the buffer pointed to by *curr_value*. The time remaining until the next timer expiration is returned in *curr_value.it_value*; this is always a relative value, regardless of whether the **TIMER_ABSTIME** flag was used when arming the timer. If the value returned in *curr_value.it_value* is zero, then the timer is currently disarmed. The timer interval is returned in *curr_value.it_interval*. If the value returned in *curr_value.it_interval* is zero, then this is a "one-shot" timer.

RETURN VALUE

On success, **timer_settime**() and **timer_gettime**() return 0. On error, −1 is returned, and *errno* is set to indicate the error.

ERRORS

These functions may fail with the following errors:

EFAULT
new_value, *old_value*, or *curr_value* is not valid a pointer.

EINVAL
timerid is invalid.

timer_settime() may fail with the following errors:

EINVAL
new_value.it_value is negative; or *new_value.it_value.tv_nsec* is negative or greater than 999,999,999.

VERSIONS

These system calls are available since Linux 2.6.

CONFORMING TO

POSIX.1-2001

EXAMPLE

See **timer_create**(2).

SEE ALSO

timer_create(2), **timer_settime**(2), **timer_getoverrun**(2), **time**(7)

COLOPHON

This page is part of release 3.22 of the Linux *man-pages* project. A description of the project, and information about reporting bugs, can be found at http://www.kernel.org/doc/man-pages/.