

**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 >= 199309L`

**DESCRIPTION**

`timer_settime()` arms or disarms the timer identified by *timerid*. The *new\_value* argument is pointer to 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(2)`.

If *new\_value->it\_value* specifies a nonzero value (i.e., either subfield is nonzero), 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 nonzero, 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 points to a buffer that is used to return 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 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 a valid 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, POSIX.1-2008.

## EXAMPLE

See **timer\_create(2)**.

## SEE ALSO

**timer\_create(2)**, **timer\_getoverrun(2)**, **time(7)**

## COLOPHON

This page is part of release 5.05 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/>.