NAME

getitimer, setitimer − get or set value of an interval timer

SYNOPSIS

#include <sys/time.h>

int getitimer(int *which***, struct itimerval ****curr_value***);**
int setitimer(int *which***, const struct itimerval ****new_value***,**
struct itimerval ****old_value*);**

DESCRIPTION

The system provides each process with three interval timers, each decrementing in a distinct time domain. When any timer expires, a signal is sent to the process, and the timer (potentially) restarts.

ITIMER_REAL decrements in real time, and delivers **SIGALRM** upon expiration.

ITIMER_VIRTUAL decrements only when the process is executing, and delivers **SIGVTALRM** upon expiration.

ITIMER_PROF decrements both when the process executes and when the system is executing on behalf of the process. Coupled with **ITIMER_VIRTUAL**, this timer is usually used to profile the time spent by the application in user and kernel space. **SIGPROF** is delivered upon expiration.

Timer values are defined by the following structures:

```
struct itimerval {
   struct timeval it_interval; /* next value */
   struct timeval it_value;    /* current value */
};

struct timeval {
   long tv_sec;             /* seconds */
   long tv_usec;             /* microseconds */
};
```

The function **getitimer**() fills the structure pointed to by *curr_value* with the current setting for the timer specified by *which* (one of **ITIMER_REAL**, **ITIMER_VIRTUAL**, or **ITIMER_PROF**). The element *it_value* is set to the amount of time remaining on the timer, or zero if the timer is disabled. Similarly, *it_interval* is set to the reset value.

The function **setitimer**() sets the specified timer to the value in *new_value*. If *old_value* is non-NULL, the old value of the timer is stored there.

Timers decrement from *it_value* to zero, generate a signal, and reset to *it_interval*. A timer which is set to zero (*it_value* is zero or the timer expires and *it_interval* is zero) stops.

Both *tv_sec* and *tv_usec* are significant in determining the duration of a timer.

Timers will never expire before the requested time, but may expire some (short) time afterwards, which depends on the system timer resolution and on the system load; see **time**(7). (But see BUGS below.) Upon expiration, a signal will be generated and the timer reset. If the timer expires while the process is active (always true for **ITIMER_VIRTUAL**) the signal will be delivered immediately when generated. Otherwise the delivery will be offset by a small time dependent on the system loading.

RETURN VALUE

On success, zero is returned. On error, −1 is returned, and *errno* is set appropriately.

ERRORS

EFAULT

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

EINVAL

which is not one of **ITIMER_REAL**, **ITIMER_VIRTUAL**, or **ITIMER_PROF**; or (since Linux 2.6.22) one of the *tv_usec* fields in the structure pointed to by *new_value* contains a value outside the range 0 to 999999.

CONFORMING TO

POSIX.1-2001, SVr4, 4.4BSD (this call first appeared in 4.2BSD). POSIX.1-2008 marks **getitimer**() and **setitimer**() obsolete, recommending the use of the POSIX timers API (**timer_gettime**(2), **timer_settime**(2), etc.) instead.

NOTES

A child created via **fork**(2) does not inherit its parent's interval timers. Interval timers are preserved across an **execve**(2).

POSIX.1 leaves the interaction between **setitimer()** and the three interfaces **alarm**(2), **sleep**(3), and **usleep**(3) unspecified.

BUGS

The generation and delivery of a signal are distinct, and only one instance of each of the signals listed above may be pending for a process. Under very heavy loading, an **ITIMER_REAL** timer may expire before the signal from a previous expiration has been delivered. The second signal in such an event will be lost.

On Linux kernels before 2.6.16, timer values are represented in jiffies. If a request is made set a timer with a value whose jiffies representation exceeds **MAX_SEC_IN_JIFFIES** (defined in *include/linux/jiffies.h*), then the timer is silently truncated to this ceiling value. On Linux/i386 (where, since Linux 2.6.13, the default jiffy is 0.004 seconds), this means that the ceiling value for a timer is approximately 99.42 days. Since Linux 2.6.16, the kernel uses a different internal representation for times, and this ceiling is removed.

On certain systems (including i386), Linux kernels before version 2.6.12 have a bug which will produce premature timer expirations of up to one jiffy under some circumstances. This bug is fixed in kernel 2.6.12.

POSIX.1-2001 says that **setitimer**() should fail if a *tv_usec* value is specified that is outside of the range 0 to 999999. However, in kernels up to and including 2.6.21, Linux does not give an error, but instead silently adjusts the corresponding seconds value for the timer. From kernel 2.6.22 onwards, this non-conformance has been repaired: an improper *tv_usec* value results in an **EINVAL** error.

SEE ALSO

gettimeofday(2), **sigaction**(2), **signal**(2), **timer_create**(2), **timerfd_create**(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/.