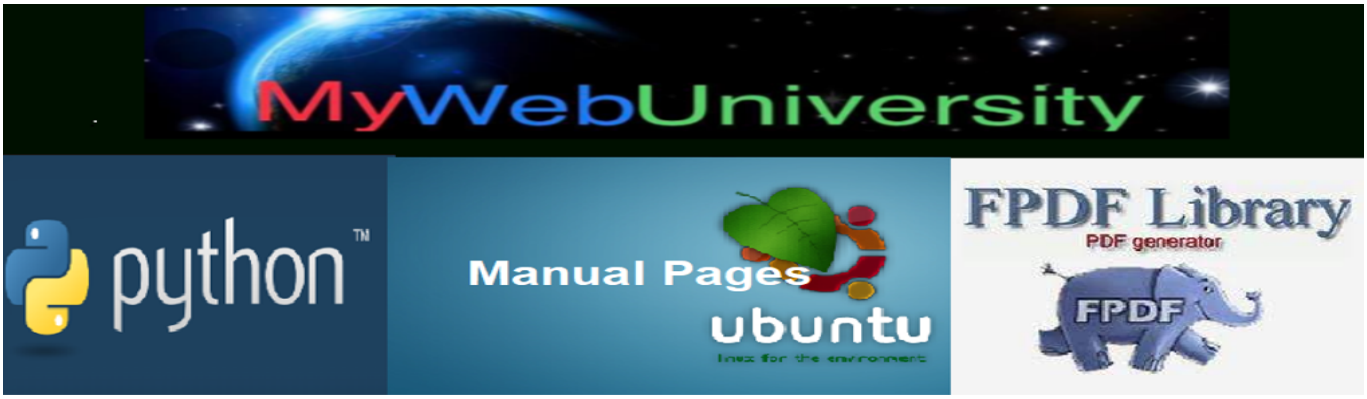

Rocky Enterprise Linux 9.2 Manual Pages on command 'getitimer.2'

$ man getitimer.2

GETITIMER(2) Linux Programmer's Manual GETITIMER(2)

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

These system calls provide access to interval timers, that is, timers that initially ex? pire at some point in the future, and (optionally) at regular intervals after that. When a timer expires, a signal is generated for the calling process, and the timer is reset to the specified interval (if the interval is nonzero).

Three types of timers?specified via the which argument?are provided, each of which counts against a different clock and generates a different signal on timer expiration:

ITIMER_REAL

This timer counts down in real (i.e., wall clock) time. At each expiration, a SIGALRM signal is generated.

ITIMER_VIRTUAL

This timer counts down against the user-mode CPU time consumed by the process. (The measurement includes CPU time consumed by all threads in the process.) At each expiration, a SIGVTALRM signal is generated.

ITIMER_PROF

This timer counts down against the total (i.e., both user and system) CPU time con?
sumed by the process. (The measurement includes CPU time consumed by all threads
in the process.) At each expiration, a SIGPROF signal is generated.
In conjunction with ITIMER_VIRTUAL, this timer can be used to profile user and sys?
tem CPU time consumed by the process.

A process has only one of each of the three types of timers.

Timer values are defined by the following structures:

```
struct itimerval {
    struct timeval it_interval; /* Interval for periodic timer */
    struct timeval it_value;    /* Time until next expiration */
};
struct timeval {
    time_t      tv_sec;         /* seconds */
    suseconds_t tv_usec;        /* microseconds */
};
```

getitimer()

The function getitimer() places the current value of the timer specified by which in the buffer pointed to by curr_value.

The it_value substructure is populated with the amount of time remaining until the next expiration of the specified timer. This value changes as the timer counts down, and will be reset to it_interval when the timer expires. If both fields of it_value are zero, then this timer is currently disarmed (inactive).

The it_interval substructure is populated with the timer interval. If both fields of it_interval are zero, then this is a single-shot timer (i.e., it expires just once).

setitimer()

The function setitimer() arms or disarms the timer specified by which, by setting the timer to the value specified by new_value. If old_value is non-NULL, the buffer it points to is used to return the previous value of the timer (i.e., the same information that is returned by getitimer()).

If either field in new_value.it_value is nonzero, then the timer is armed to initially ex? pire at the specified time. If both fields in new_value.it_value are zero, then the timer is disarmed.

The new_value.it_interval field specifies the new interval for the timer; if both of its

subfields are zero, the timer is single-shot.

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

Timers will never expire before the requested time, but may expire some (short) time af? terward, which depends on the system timer resolution and on the system load; see time(7). (But see BUGS below.) If the timer expires while the process is active (always true for ITIMER_VIRTUAL), the signal will be delivered immediately when generated.

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.

The standards are silent on the meaning of the call:

```
setitimer(which, NULL, &old_value);
```

Many systems (Solaris, the BSDs, and perhaps others) treat this as equivalent to:

```
getitimer(which, &old_value);
```

In Linux, this is treated as being equivalent to a call in which the new_value fields are zero; that is, the timer is disabled. Don't use this Linux misfeature: it is nonportable and unnecessary.

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 deliv? ered. 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 (de? fined 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 onward, this nonconformance 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 5.10 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/.