NAME

getitimer, **setitimer** - get/set value of interval timer

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

```
#include <sys/time.h>

#define ITIMER_REAL     0
#define ITIMER_VIRTUAL  1
#define ITIMER_PROF     2

int
getitimer(int which, struct itimerval *value);

int
setitimer(int which, const struct itimerval *value, struct itimerval *ovalue);
```

DESCRIPTION

The system provides each process with three interval timers, defined in <*sys/time.h*>. The **getitimer**() system call returns the current value for the timer specified in *which* in the structure at *value*. The **setitimer**() system call sets a timer to the specified *value* (returning the previous value of the timer if *ovalue* is not a null pointer).

A timer value is defined by the *itimerval* structure:

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

If *it_value* is non-zero, it indicates the time to the next timer expiration. If *it_interval* is non-zero, it specifies a value to be used in reloading *it_value* when the timer expires. Setting *it_value* to 0 disables a timer, regardless of the value of *it_interval*. Setting *it_interval* to 0 causes a timer to be disabled after its next expiration (assuming *it_value* is non-zero).

Time values smaller than the resolution of the system clock are rounded up to this resolution (typically 10 milliseconds).

The ITIMER_REAL timer decrements in real time. A SIGALRM signal is delivered when this timer expires.

The ITIMER_VIRTUAL timer decrements in process virtual time. It runs only when the process is executing. A SIGVTALRM signal is delivered when it expires.

The ITIMER_PROF timer decrements both in process virtual time and when the system is running on behalf of the process. It is designed to be used by interpreters in statistically profiling the execution of interpreted programs. Each time the ITIMER_PROF timer expires, the SIGPROF signal is delivered. Because this signal may interrupt in-progress system calls, programs using this timer must be prepared to restart interrupted system calls.

The maximum number of seconds allowed for *it_interval* and *it_value* in **setitimer**() is 100000000.

NOTES

Three macros for manipulating time values are defined in <*sys/time.h*>. The **timerclear**() macro sets a time value to zero, **timerisset**() tests if a time value is non-zero, and **timercmp**() compares two time values.

RETURN VALUES

Upon successful completion, the value 0 is returned; otherwise the value -1 is returned and the global variable *errno* is set to indicate the error.

ERRORS

The **getitimer**() and **setitimer**() system calls will fail if:

[EFAULT] The *value* argument specified a bad address.

[EINVAL] The *value* argument specified a time that was too large to be handled.

SEE ALSO

gettimeofday(2), select(2), sigaction(2), clocks(7)

STANDARDS

The **getitimer**() and **setitimer**() functions conform to IEEE Std 1003.1-2001 ("POSIX.1"). The later IEEE Std 1003.1-2008 ("POSIX.1") revision however marked both functions as obsolescent, recommending the use of timer_gettime(2) and timer_settime(2) instead.

HISTORY

The **getitimer**() system call appeared in 4.2BSD.