setitimer()--Set Value for Interval Timer

Last Updated: 2021-04-14

Syntax

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

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

Service Program Name: QP0SSRV1

Default Public Authority: *USE

Threadsafe: Yes

The **setitimer()** function sets the timer specified by *which* to the value in the structure pointed to by *value* and stores the previous value of the timer in the structure pointed to by *ovalue*.

Authorities and Locks

None.

Parameters

which

(Input) The interval timer type.

The possible values for *which*, which are defined in the <**sys/time.h**> header file, are as follows:

ITIMER_REAL	The interval timer value is decremented in real time. The SIGALRM signal is generated for the process when this timer expires.
ITIMER_VIRTUAL	The interval timer value is only decremented when the process is running. The SIGVTALRM signal is generated for the process when this timer expires.
ITIMER_PROF	The interval timer value is only decremented when the process is running or when the system is running on behalf of the process. The SIGPROF signal is generated for the process when this timer expires.

elapses. If *it_value* is zero, the timer is disabled and the value of *it_interval* is ignored. If *it_interval* is zero, the timer is disabled after the next timer expiration.

ovalue

(Output) A pointer to the space where the previous interval timer value is stored. This value may be NULL.

Return Value

0	**setitimer()** was successful.
-1	**setitimer()** was not successful. The *errno* variable is set to indicate the error.

Error Conditions

If **setitimer()** is not successful, *errno* usually indicates the following error. Under some conditions, *errno* could indicate an error other than that listed here.

[EINVAL]

The value specified for the argument is not correct.

A function was passed incorrect argument values, or an operation was attempted on an object and the operation specified is not supported for that type of object.

An argument value is not valid, out of range, or NULL.

- The value of *which* is not equal to one of the defined values.
- The *tv_usec* member of the *it_value* structure has a value greater than or equal to 1,000,000.
- The *tv_usec* member of the *it_interval* structure has a value greater than or equal to 1,000,000.

[ENOSYSRSC]

System resources not available to complete request.

- The ITIMER_VIRTUAL value for *which* is not supported on this implementation.
- The ITIMER_PROF value for *which* is not supported on this implementation.

[ENOTSIGINIT]

Process not enabled for signals.

An attempt was made to call a signal function under one of the following conditions:

- The signal function is being called for a process that is not enabled for asynchronous signals.
- The signal function is being called for a system signal that has not been initialized.

Usage Notes

The **setitimer()** function enables a process for signals if the process is not already enabled for signals. For details, see Qp0sEnableSignals()--Enable Process for Signals. If the system has not been enabled for signals, **setitimer()** is not successful, and an [ENOTSIGINIT] error is returned.

Related Information

- The <**sys/time.h**> file (see Header Files for UNIX®-Type Functions)
- alarm()--Set Schedule for Alarm Signal
- setitimer()--Set Value for Interval Timer
- sleep()--Suspend Processing for Interval of Time
- usleep()--Suspend Processing for Interval of Time

Example

The following example returns the current interval timer value using the **setitimer()** function.

Note: By using the code examples, you agree to the terms of the Code license and disclaimer information.

```
#include <sys/time.h>
#include <signal.h>
#include <unistd.h>
#include <stdio.h>
#include <time.h>
#include <errno.h>

#define LOOP_LIMIT  1E12

volatile int sigcount=0;

void catcher( int sig ) {

   struct itimerval value;
   int which = ITIMER_REAL;

   printf( "Signal catcher called for signal %d\n", sig );
   sigcount++;
```

```
      getitimer( which, &value );

      value.it_value.tv_sec = 0;
      value.it_value.tv_usec = 0;

      setitimer( which, &value, NULL );
    }
  }

  int main( int argc, char *argv[] ) {

    int result = 0;

    struct itimerval value, ovalue, pvalue;
    int which = ITIMER_REAL;

    struct sigaction sact;
    volatile double count;
    time_t t;

    sigemptyset( &sact.sa_mask );
    sact.sa_flags = 0;
    sact.sa_handler = catcher;
    sigaction( SIGALRM, &sact, NULL );

    getitimer( which, &pvalue );

    /*
     * Set a real time interval timer to repeat every 200 milliseconds
     */

    value.it_interval.tv_sec = 0;       /* Zero seconds */
    value.it_interval.tv_usec = 200000;  /* Two hundred milliseconds */
    value.it_value.tv_sec = 0;          /* Zero seconds */
    value.it_value.tv_usec = 500000;    /* Five hundred milliseconds */

    result = setitimer( which, &value, &ovalue );

    /*
     * The interval timer value returned by setitimer() should be
     * identical to the timer value returned by getitimer().
     */
```

```
   time( &t );
   printf( "Before loop, time is %s", ctime(&t) );

   for( count=0; ((count<LOOP_LIMIT) && (sigcount<2)); count++ );

   time( &t );
   printf( "After loop, time is %s\n", ctime(&t) );


   if( sigcount == 0 )
      printf( "The signal catcher never gained control\n" );
   else
      printf( "The signal catcher gained control\n" );

   printf( "The value of count is %.0f\n", count );

   return( result );
}
```

Output:

```
Before loop, time is Sun Jun 15 10:14:00 1997
Signal catcher called for signal 14
Signal catcher called for signal 14
After loop, time is Sun Jun 15 10:14:01 1997
The signal catcher gained control
The value of count is 702943
```

API introduced: V4R2

[Back to top | UNIX-Type APIs | APIs by category]