Linux System Calls

Oct 12, 2001

8.13 *setitimer*: Setting Interval Timers

The `setitimer` system call is a generalization of the `alarm` call. It schedules the delivery of a signal at some point in the future after a fixed amount of time has elapsed.

A program can set three different types of timers with `setitimer`:

- If the timer code is `ITIMER_REAL`, the process is sent a `SIGALRM` signal after the specified wall-clock time has elapsed.
- If the timer code is `ITIMER_VIRTUAL`, the process is sent a `SIGVTALRM` signal after the process has executed for the specified time. Time in which the process is not executing (that is, when the kernel or another process is running) is not counted.
- If the timer code is `ITIMER_PROF`, the process is sent a `SIGPROF` signal when the specified time has elapsed either during the process's own execution or the execution of a system call on behalf of the process.

The first argument to `setitimer` is the timer code, specifying which timer to set. The second argument is a pointer to a `struct itimerval` object specifying the new settings for that timer. The third argument, if not null, is a pointer to another `struct itimerval` object that receives the old timer settings.

A `struct itimerval` variable has two fields:

- `it_value` is a `struct timeval` field that contains the time until the timer next expires and a signal is sent. If this is 0, the timer is disabled.
- `it_interval` is another `struct timeval` field containing the value to which the timer will be reset after it expires. If this is 0, the timer will be disabled after it expires. If this is nonzero, the timer is set to expire repeatedly after this interval.

The `struct timeval` type is described in Section 8.7, "`gettimeofday`: Wall-Clock Time."

The program in Listing 8.11 illustrates the use of `setitimer` to track the execution time of a program. A timer is configured to expire every 250 milliseconds and send a `SIGVTALRM` signal.

Listing 8.11 (*itemer.c*) Timer Example

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

void timer_handler (int signum)
{
 static int count = 0;
 printf ("timer expired %d times\n", ++count);
}

int main ()
{
 struct sigaction sa;
 struct itimerval timer;

 /* Install timer_handler as the signal handler for SIGVTALRM. */
```

```
  memset (&sa, 0, sizeof (sa));
  sa.sa_handler = &timer_handler;
  sigaction (SIGVTALRM, &sa, NULL);

  /* Configure the timer to expire after 250 msec... */
  timer.it_value.tv_sec = 0;
  timer.it_value.tv_usec = 250000;
  /* ... and every 250 msec after that. */
  timer.it_interval.tv_sec = 0;
  timer.it_interval.tv_usec = 250000;
  /* Start a virtual timer. It counts down whenever this process is
    executing. */
  setitimer (ITIMER_VIRTUAL, &timer, NULL);

  /* Do busy work. */
  while (1);
}
```

+ Share This Save To Your Account

< Back **Page 13** of 15 Next >