

NAME

sigwaitinfo, sigtimedwait – synchronously wait for queued signals

SYNOPSIS

```
#include <signal.h>
```

```
int sigwaitinfo(const sigset_t *set, siginfo_t *info);
```

```
int sigtimedwait(const sigset_t *set, siginfo_t *info,
                 const struct timespec *timeout);
```

Feature Test Macro Requirements for glibc (see **feature_test_macros(7)**):

```
sigwaitinfo(), sigtimedwait(): _POSIX_C_SOURCE >= 199309L
```

DESCRIPTION

sigwaitinfo() suspends execution of the calling thread until one of the signals in *set* is delivered. (If one of the signals in *set* is already pending for the calling thread, **sigwaitinfo()** will return immediately with information about that signal.)

sigwaitinfo() removes the delivered signal from the set of pending signals and returns the signal number as its function result. If the *info* argument is not NULL, then it returns a structure of type *siginfo_t* (see **sigaction(2)**) containing information about the signal.

Signals returned via **sigwaitinfo()** are delivered in the usual order; see **signal(7)** for further details.

sigtimedwait() operates in exactly the same way as **sigwaitinfo()** except that it has an additional argument, *timeout*, which enables an upper bound to be placed on the time for which the thread is suspended. This argument is of the following type:

```
struct timespec {
    long   tv_sec;      /* seconds */
    long   tv_nsec;     /* nanoseconds */
}
```

If both fields of this structure are specified as 0, a poll is performed: **sigtimedwait()** returns immediately, either with information about a signal that was pending for the caller, or with an error if none of the signals in *set* was pending.

RETURN VALUE

On success, both **sigwaitinfo()** and **sigtimedwait()** return a signal number (i.e., a value greater than zero). On failure both calls return -1 , with *errno* set to indicate the error.

ERRORS**EAGAIN**

No signal in *set* was delivered within the *timeout* period specified to **sigtimedwait()**.

EINTR

The wait was interrupted by a signal handler; see **signal(7)**. (This handler was for a signal other than one of those in *set*.)

EINVAL

timeout was invalid.

CONFORMING TO

POSIX.1-2001.

NOTES

In normal usage, the calling program blocks the signals in *set* via a prior call to **sigprocmask(2)** (so that the default disposition for these signals does not occur if they are delivered between successive calls to

sigwaitinfo() or **sigtimedwait()** and does not establish handlers for these signals. In a multithreaded program, the signal should be blocked in all threads to prevent the signal being delivered to a thread other than the one calling **sigwaitinfo()** or **sigtimedwait()**.

The set of signals that is pending for a given thread is the union of the set of signals that is pending specifically for that thread and the set of signals that is pending for the process as a whole (see **signal(7)**).

If multiple threads of a process are blocked waiting for the same signal(s) in **sigwaitinfo()** or **sigtimedwait()**, then exactly one of the threads will actually receive the signal if it is delivered to the process as a whole; which of the threads receives the signal is indeterminate.

POSIX leaves the meaning of a NULL value for the *timeout* argument of **sigtimedwait()** unspecified, permitting the possibility that this has the same meaning as a call to **sigwaitinfo()**, and indeed this is what is done on Linux.

On Linux, **sigwaitinfo()** is a library function implemented on top of **sigtimedwait()**.

SEE ALSO

kill(2), **sigaction(2)**, **signal(2)**, **signalfd(2)**, **sigpending(2)**, **sigprocmask(2)**, **sigqueue(2)**, **sigsetops(3)**, **sigwait(3)**, **signal(7)**, **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/>.