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

$ man sigtimedwait.2

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

NAME

sigwaitinfo, sigtimedwait, rt_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 pending (If one of the signals in set is already pending for the calling thread, sigwaitinfo() will return immediately.) sigwaitinfo() removes the signal from the set of pending signals and returns the signal number as its function result. If the info argument is not NULL, then the buffer that it points to is used to return a

structure of type siginfo_t (see sigaction(2)) containing information about the signal.

If multiple signals in set are pending for the caller, the signal that is retrieved by sigwaitinfo() is determined according to the usual or? dering rules; see signal(7) for further details.

sigtimedwait() operates in exactly the same way as sigwaitinfo() except that it has an additional argument, timeout, which specifies the inter? val for which the thread is suspended waiting for a signal. (This in? terval will be rounded up to the system clock granularity, and kernel scheduling delays mean that the interval may overrun by a small amount.) 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 per? formed: 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 num? ber (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 became pending 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, POSIX.1-2008.

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 become pending 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, in order to prevent the signal being treated according to its default disposition in 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 sig? nal(7)).

Attempts to wait for SIGKILL and SIGSTOP are silently ignored.

If multiple threads of a process are blocked waiting for the same sig? nal(s) in sigwaitinfo() or sigtimedwait(), then exactly one of the threads will actually receive the signal if it becomes pending for the process as a whole; which of the threads receives the signal is inde? terminate.

sigwaitinfo() or sigtimedwait(), can't be used to receive signals that are synchronously generated, such as the SIGSEGV signal that results from accessing an invalid memory address or the SIGFPE signal that re? sults from an arithmetic error. Such signals can be caught only via signal handler.

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.

C library/kernel differences

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

The glibc wrapper functions for sigwaitinfo() and sigtimedwait() silently ignore attempts to wait for the two real-time signals that are used internally by the NPTL threading implementation. See nptl(7) for details.

The original Linux system call was named sigtimedwait(). However, with the addition of real-time signals in Linux 2.2, the fixed-size, 32-bit sigset_t type supported by that system call was no longer fit for pur? pose. Consequently, a new system call, rt_sigtimedwait(), was added to support an enlarged sigset_t type. The new system call takes a fourth argument, size_t sigsetsize, which specifies the size in bytes of the signal set in set. This argument is currently required to have the value sizeof(sigset_t) (or the error EINVAL results). The glibc sig? timedwait() wrapper function hides these details from us, transparently calling rt_sigtimedwait() when the kernel provides it.

SEE ALSO

kill(2), sigaction(2), signal(2), signalfd(2), sigpending(2), sigproc? mask(2), sigqueue(3), sigsetops(3), sigwait(3), signal(7), 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/.