

NAME

pthread_sigmask – examine and change mask of blocked signals

LIBRARY

POSIX threads library (*libpthread*, *-lpthread*)

SYNOPSIS

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

```
int pthread_sigmask(int how, const sigset_t *set, sigset_t *oldset);
```

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

```
pthread_sigmask():
```

```
_POSIX_C_SOURCE >= 199506L || _XOPEN_SOURCE >= 500
```

DESCRIPTION

The **pthread_sigmask()** function is just like **sigprocmask(2)**, with the difference that its use in multi-threaded programs is explicitly specified by POSIX.1. Other differences are noted in this page.

For a description of the arguments and operation of this function, see **sigprocmask(2)**.

RETURN VALUE

On success, **pthread_sigmask()** returns 0; on error, it returns an error number.

ERRORS

See **sigprocmask(2)**.

ATTRIBUTES

For an explanation of the terms used in this section, see **attributes(7)**.

Interface	Attribute	Value
pthread_sigmask()	Thread safety	MT-Safe

STANDARDS

POSIX.1-2001, POSIX.1-2008.

NOTES

A new thread inherits a copy of its creator's signal mask.

The glibc **pthread_sigmask()** function silently ignores attempts to block the two real-time signals that are used internally by the NPTL threading implementation. See **nptl(7)** for details.

EXAMPLES

The program below blocks some signals in the main thread, and then creates a dedicated thread to fetch those signals via **sigwait(3)**. The following shell session demonstrates its use:

```
$ ./a.out &
[1] 5423
$ kill -QUIT %1
Signal handling thread got signal 3
$ kill -USR1 %1
Signal handling thread got signal 10
$ kill -TERM %1
[1]+  Terminated                  ./a.out
```

Program source

```
#include <errno.h>
#include <pthread.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
```

```

#include <unistd.h>

/* Simple error handling functions */

#define handle_error_en(en, msg) \
    do { errno = en; perror(msg); exit(EXIT_FAILURE); } while (0)

static void *
sig_thread(void *arg)
{
    sigset_t *set = arg;
    int s, sig;

    for (;;) {
        s = sigwait(set, &sig);
        if (s != 0)
            handle_error_en(s, "sigwait");
        printf("Signal handling thread got signal %d\n", sig);
    }
}

int
main(void)
{
    pthread_t thread;
    sigset_t set;
    int s;

    /* Block SIGQUIT and SIGUSR1; other threads created by main()
       will inherit a copy of the signal mask. */

    sigemptyset(&set);
    sigaddset(&set, SIGQUIT);
    sigaddset(&set, SIGUSR1);
    s = pthread_sigmask(SIG_BLOCK, &set, NULL);
    if (s != 0)
        handle_error_en(s, "pthread_sigmask");

    s = pthread_create(&thread, NULL, &sig_thread, &set);
    if (s != 0)
        handle_error_en(s, "pthread_create");

    /* Main thread carries on to create other threads and/or do
       other work. */

    pause(); /* Dummy pause so we can test program */
}

```

SEE ALSO

sigaction(2), sigpending(2), sigprocmask(2), pthread_attr_setsigmask_np(3), pthread_create(3), pthread_kill(3), sigsetops(3), pthreads(7), signal(7)