



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

\$ man rt_sigprocmask.2

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

NAME

sigprocmask, rt_sigprocmask - examine and change blocked signals

SYNOPSIS

```
#include <signal.h>

/* Prototype for the glibc wrapper function */
int sigprocmask(int how, const sigset_t *set, sigset_t *oldset);

/* Prototype for the underlying system call */
int rt_sigprocmask(int how, const kernel_sigset_t *set,
                   kernel_sigset_t *oldset, size_t sigsetsize);

/* Prototype for the legacy system call (deprecated) */
int sigprocmask(int how, const old_kernel_sigset_t *set,
                old_kernel_sigset_t *oldset);
```

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

```
sigprocmask(): _POSIX_C_SOURCE
```

DESCRIPTION

sigprocmask() is used to fetch and/or change the signal mask of the calling thread. The signal mask is the set of signals whose delivery

is currently blocked for the caller (see also `signal(7)` for more details).

The behavior of the call is dependent on the value of `how`, as follows.

SIG_BLOCK

The set of blocked signals is the union of the current set and the set argument.

SIG_UNBLOCK

The signals in set are removed from the current set of blocked signals. It is permissible to attempt to unblock a signal which is not blocked.

SIG_SETMASK

The set of blocked signals is set to the argument set.

If `oldset` is non-NULL, the previous value of the signal mask is stored in `oldset`.

If `set` is NULL, then the signal mask is unchanged (i.e., how is ignored), but the current value of the signal mask is nevertheless returned in `oldset` (if it is not NULL).

A set of functions for modifying and inspecting variables of type `sigset_t` ("signal sets") is described in `sigsetops(3)`.

The use of `sigprocmask()` is unspecified in a multithreaded process; see `pthread_sigmask(3)`.

RETURN VALUE

`sigprocmask()` returns 0 on success and -1 on error. In the event of an error, `errno` is set to indicate the cause.

ERRORS

EFAULT The `set` or `oldset` argument points outside the process's allocated address space.

EINVAL Either the value specified in `how` was invalid or the kernel does not support the size passed in `sigsetsize`.

CONFORMING TO

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

NOTES

It is not possible to block SIGKILL or SIGSTOP. Attempts to do so are

silently ignored.

Each of the threads in a process has its own signal mask.

A child created via `fork(2)` inherits a copy of its parent's signal mask; the signal mask is preserved across `execve(2)`.

If `SIGBUS`, `SIGFPE`, `SIGILL`, or `SIGSEGV` are generated while they are blocked, the result is undefined, unless the signal was generated by `kill(2)`, `sigqueue(3)`, or `raise(3)`.

See `sigsetops(3)` for details on manipulating signal sets.

Note that it is permissible (although not very useful) to specify both set and oldset as `NULL`.

C library/kernel differences

The kernel's definition of `sigset_t` differs in size from that used by the C library. In this manual page, the former is referred to as `kernel_sigset_t` (it is nevertheless named `sigset_t` in the kernel sources).

The glibc wrapper function for `sigprocmask()` silently ignores attempts to block 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 `sigprocmask()`. However, with the addition of real-time signals in Linux 2.2, the fixed-size, 32-bit `sigset_t` (referred to as `old_kernel_sigset_t` in this manual page) type supported by that system call was no longer fit for purpose. Consequently, a new system call, `rt_sigprocmask()`, was added to support an enlarged `sigset_t` type (referred to as `kernel_sigset_t` in this manual page). The new system call takes a fourth argument, `size_t sigsetsize`, which specifies the size in bytes of the signal sets in set and oldset.

This argument is currently required to have a fixed architecture specific value (equal to `sizeof(kernel_sigset_t)`).

The glibc `sigprocmask()` wrapper function hides these details from us, transparently calling `rt_sigprocmask()` when the kernel provides it.

SEE ALSO

`kill(2)`, `pause(2)`, `sigaction(2)`, `signal(2)`, `sigpending(2)`, `sigsuspend(2)`, `pthread_sigmask(3)`, `sigqueue(3)`, `sigsetops(3)`, `signal(7)`

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/>.

Linux

2017-09-15

SIGPROCMASK(2)