NAME

sigfastblock - controls signals blocking with a simple memory write

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <sys/signalvar.h>

int
sigfastblock(*int cmd, void *ptr*);

DESCRIPTION

This function is not intended for a direct usage by applications. The functionality is provided for implementing some optimizations in ld-elf.so.1(8) and 1:1 Threading Library (libthr, -lthr).

The function configures the kernel facility that allows a thread to block asynchronous signals delivery with a single write to userspace memory, avoiding overhead of system calls like sigprocmask(2) for establishing critical sections. The C runtime uses it to optimize implementation of async-signal-safe functionality.

A thread might register a sigblock variable of type *int* as a location which is consulted by kernel when calculating the blocked signal mask for delivery of asynchronous signals. If the variable indicates that blocking is requested, then the kernel effectively operates as if the mask containing all blockable signals was supplied to sigprocmask(2).

The variable is supposed to be modified only from the owning thread, there is no way to guarantee visibility of update from other thread to kernel when signals are delivered.

Lower bits of the sigblock variable are reserved as flags, which might be set or cleared by kernel at arbitrary moments. Userspace code should use atomic(9) operations of incrementing and decrementing by SIGFASTBLOCK_INC quantity to recursively block or unblock signals delivery.

If a signal would be delivered when unmasked, kernel might set the SIGFASTBLOCK_PEND "pending signal" flag in the sigblock variable. Userspace should perform SIGFASTBLOCK_UNBLOCK operation when clearing the variable if it notes the pending signal bit is set, which would deliver the pending signals immediately. Otherwise, signals delivery might be postponed.

The *cmd* argument specifies one of the following operations:

SIGFASTBLOCK_SETPTR Register the variable of type *int* at location pointed to by the *ptr* argument as sigblock variable for the calling thread.

SIGFASTBLOCK_UNSETPTR Unregister the currently registered sigblock location. Kernel stops inferring the blocked mask from non-zero value of its blocked count. New location can be registered after previous one is deregistered.

SIGFASTBLOCK_UNBLOCK If there are pending signals which should be delivered to the calling thread, they are delivered before returning from the call. The sigblock variable should have zero blocking count, and indicate that the pending signal exists. Effectively this means that the variable should have the value SIGFASTBLOCK_PEND.

RETURN VALUES

Upon successful completion, the value 0 is returned; otherwise the value -1 is returned and the global variable *errno* is set to indicate the error.

ERRORS

The operation may fail with the following errors:

[EBUSY] The SIGFASTBLOCK_SETPTR attempted while the sigblock address was already registered. The SIGFASTBLOCK_UNBLOCK was called while sigblock variable value is not equal to SIGFASTBLOCK_PEND.

[EINVAL] The variable address passed to SIGFASTBLOCK_SETPTR is not aligned naturally. The SIGFASTBLOCK_UNSETPTR operation was attempted without prior successfull call to SIGFASTBLOCK_SETPTR.

[EFAULT] Attempt to read or write to the sigblock variable failed. Note that kernel generates the SIGSEGV signal if an attempt to read from the sigblock variable faulted during implicit accesses from syscall entry.

SEE ALSO

kill(2), signal(2), sigprocmask(2), libthr(3), ld-elf.so.1(8)

STANDARDS

The **sigfastblock** function is non-standard, although a similar functionality is a common optimization provided by several other systems.

HISTORY

The **sigfastblock** function was introduced in FreeBSD 13.0.

BUGS

The **sigfastblock** symbol is currently not exported by libc, on purpose. Consumers should either use the __sys_fast_sigblock symbol from the private libc namespace, or utilize syscall(2).