

NAME

siginfo - signal generation information

SYNOPSIS

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

DESCRIPTION

A process may request signal information when it is catching a signal. The information specifies why the system generated that signal. To request signal information in a signal handler, the user can set SA_SIGINFO in *sa_flags* before `sigaction(2)` is called, otherwise the user can use `sigwaitinfo(2)` and `sigtimedwait(2)` to get signal information. In either case, the system returns the information in a structure of type *siginfo_t*, which includes the following information:

Type	Member	Description
<i>int</i>	<i>si_signo</i>	signal number
<i>int</i>	<i>si_errno</i>	error number
<i>int</i>	<i>si_code</i>	signal code
<i>union sigval</i>	<i>si_value</i>	signal value
<i>pid_t</i>	<i>si_pid</i>	sending process ID
<i>uid_t</i>	<i>si_uid</i>	sending process's real user ID
<i>void</i>	<i>*si_addr</i>	virtual address
<i>int</i>	<i>si_status</i>	exit value or signal
<i>long</i>	<i>si_band</i>	band event for SIGPOLL
<i>int</i>	<i>si_trapno</i>	machine trap code
<i>int</i>	<i>si_timerid</i>	POSIX timer ID
<i>int</i>	<i>si_overrun</i>	POSIX timer overrun count
<i>int</i>	<i>si_mqd</i>	POSIX message queue ID
<i>int</i>	<i>si_syscall</i>	system-call number for system calls blocked by Capsicum

The *si_signo* member contains the signal number.

The *si_errno* member contains an error number defined in the file *<errno.h>*.

The *si_code* member contains a code which describes the cause of the signal. The macros specified in the **Code** column of the following table are defined for use as values of *si_code* that are signal-specific or non-signal-specific reasons why the signal was generated:

Signal	Code	Reason
SIGILL	ILL_ILLOPC	illegal opcode
	ILL_ILLOPN	illegal operand

	ILL_ILADR	illegal addressing mode
	ILL_ILLTRP	illegal trap
	ILL_PRVOPC	illegal privileged opcode
	ILL_PRVREG	illegal privileged register
	ILL_COPROC	coprocessor error
	ILL_BADSTK	internal stack error
SIGFPE	FPE_INTDIV	integer divide by zero
	FPE_INTOVF	integer overflow
	FPE_FLTDIV	floating-point divide by zero
	FPE_FLTOVF	floating-point overflow
	FPE_FLTUND	floating-point underflow
	FPE_FLTRES	floating-point inexact result
	FPE_FLTINV	invalid floating-point operation
	FPE_FLTSUB	subscript out of range
SIGSEGV	SEGV_MAPERR	address not mapped to object
	SEGV_ACCERR	invalid permissions for mapped object
SIGBUS	BUS_ADRALN	invalid address alignment
	BUS_ADRERR	nonexistent physical address
	BUS_OBJERR	object-specific hardware error
	BUS_OOMERR	cannot alloc a page to map at fault
SIGTRAP	TRAP_BRKPT	process breakpoint
	TRAP_TRACE	process trace trap
	TRAP_DTRACE	DTrace induced trap
	TRAP_CAP	capabilities protective trap
SIGCHLD	CLD_EXITED	child has exited
	CLD_KILLED	child has terminated abnormally and did not create a core file
	CLD_DUMPED	child has terminated abnormally and created a core file
	CLD_TRAPPED	traced child has trapped
	CLD_STOPPED	child has stopped
	CLD_CONTINUED	stopped child has continued
SIGPOLL	POLL_IN	data input available
	POLL_OUT	output buffers available
	POLL_MSG	input message available
	POLL_ERR	I/O error
	POLL_PRI	high priority input available
	POLL_HUP	device disconnected
Any	SI_NOINFO	Only the <i>si_signo</i> member is meaningful; the value of all other members is unspecified.
	SI_USER	signal sent by kill(2)

SI_QUEUE	signal sent by sigqueue(2)
SI_TIMER	signal generated by expiration of a timer set by timer_settime(2)
SI_ASYNCIO	signal generated by completion of an asynchronous I/O request
SI_MESGQ	signal generated by arrival of a message on an empty message queue
SI_KERNEL	signal generated by miscellaneous parts of the kernel
SI_LWP	signal sent by pthread_kill(3)

For synchronous signals, *si_addr* is generally set to the address of the faulting instruction. However, synchronous signals raised by a faulting memory access such as SIGSEGV and SIGBUS may report the address of the faulting memory access (if available) in *si_addr* instead. Additionally SIGTRAP raised by a hardware watchpoint exception may report the data address that triggered the watchpoint in *si_addr*.

Synchronous signals set *si_trapno* to a machine-dependent trap number.

In addition, the following signal-specific information is available:

Signal	Member	Value
SIGCHLD	<i>si_pid</i>	child process ID
	<i>si_status</i>	exit value or signal; if <i>si_code</i> is equal to CLD_EXITED, then it is equal to the exit value of the child process, otherwise, it is equal to a signal that caused the child process to change state.
	<i>si_uid</i>	real user ID of the process that sent the signal
SIGPOLL	<i>si_band</i>	band event for POLL_IN, POLL_OUT, or POLL_MSG

Finally, the following code-specific information is available:

Code	Member	Value
SI_USER	<i>si_pid</i>	the process ID that sent the signal
	<i>si_uid</i>	real user ID of the process that sent the signal
SI_QUEUE	<i>si_value</i>	the value passed to sigqueue(2) system call
	<i>si_pid</i>	the process ID that sent the signal
	<i>si_uid</i>	real user ID of the process that sent the signal
SI_TIMER	<i>si_value</i>	the value passed to timer_create(2) system call
	<i>si_timerid</i>	the timer ID returned by timer_create(2) system call
	<i>si_overrun</i>	timer overrun count corresponding to the signal
	<i>si_errno</i>	If timer overrun will be {DELAYTIMER_MAX}, an error code

		defined in <i><errno.h></i> is set
SI_ASYNCIO	<i>si_value</i>	the value passed to aio system calls
SI_MESGQ	<i>si_value</i>	the value passed to mq_notify(2) system call
	<i>si_mqd</i>	the ID of the message queue which generated the signal
SI_LWP	<i>si_pid</i>	the process ID that sent the signal
	<i>si_uid</i>	real user ID of the process that sent the signal

NOTES

Currently, the kernel never generates the SIGPOLL signal. SIGCHLD signal is queued when a process changed its status or exited. POSIX Realtime Extensions like aio, timer, and message queue also queue signals. Signals with code SI_USER, SI_KERNEL or SI_LWP are only queued if there are sufficient resources; otherwise, SI_NOINFO results. For some hardware architectures, the exact value of *si_addr* might not be available.

SEE ALSO

aio_read(2), kill(2), mq_notify(2), sigaction(2), sigqueue(2), sigwaitinfo(2), timer_create(2), timer_settime(2), waitpid(2), pthread_kill(3)

STANDARDS

The *siginfo_t* type conforms to IEEE Std 1003.1-2004 ("POSIX.1").

HISTORY

Full support for POSIX signal information first appeared in FreeBSD 7.0. The codes SI_USER and SI_KERNEL can be generated as of FreeBSD 8.1. The code SI_LWP can be generated as of FreeBSD 9.0.

AUTHORS

This manual page was written by David Xu <davidxu@FreeBSD.org>.