NAME

signal.h − signals

SYNOPSIS

#include <signal.h>

DESCRIPTION

Some of the functionality described on this reference page extends the ISO C standard. Applications shall define the appropriate feature test macro (see the System Interfaces volume of IEEE Std 1003.1-2001, Section 2.2, The Compilation Environment) to enable the visibility of these symbols in this header.

The *<signal.h>* header shall define the following symbolic constants, each of which expands to a distinct constant expression of the type:

void (*)(int)

whose value matches no declarable function.

SIG_DFL
: Request for default signal handling.

SIG_ERR
: Return value from *signal*() in case of error.

SIG_HOLD
: Request that signal be held.

SIG_IGN
: Request that signal be ignored.

The following data types shall be defined through **typedef**:

sig_atomic_t
: Possibly volatile-qualified integer type of an object that can be accessed as an atomic entity, even in the presence of asynchronous interrupts.

sigset_t
: Integer or structure type of an object used to represent sets of signals.

pid_t As described in *<sys/types.h>* .

The *<signal.h>* header shall define the **sigevent** structure, which has at least the following members:

int	**sigev_notify**	Notification type.
int	**sigev_signo**	Signal number.
union sigval	**sigev_value**	Signal value.
void(*)(union sigval)	**sigev_notify_function**	Notification function.
(pthread_attr_t *)	**sigev_notify_attributes**	Notification attributes.

The following values of *sigev_notify* shall be defined:

SIGEV_NONE
: No asynchronous notification is delivered when the event of interest occurs.

SIGEV_SIGNAL
: A queued signal, with an application-defined value, is generated when the event of interest occurs.

SIGEV_THREAD
: A notification function is called to perform notification.

The **sigval** union shall be defined as:

int sival_int Integer signal value.
void *sival_ptr Pointer signal value.

This header shall also declare the macros SIGRTMIN and SIGRTMAX, which evaluate to integer expressions, and specify a range of signal numbers that are reserved for application use and for which the realtime signal behavior specified in this volume of IEEE Std 1003.1-2001 is supported. The signal numbers in this range do not overlap any of the signals specified in the following table.

The range SIGRTMIN through SIGRTMAX inclusive shall include at least {RTSIG_MAX} signal numbers.

It is implementation-defined whether realtime signal behavior is supported for other signals.

This header also declares the constants that are used to refer to the signals that occur in the system. Signals defined here begin with the letters SIG. Each of the signals have distinct positive integer values. The value 0 is reserved for use as the null signal (see *kill*()). Additional implementation-defined signals may occur in the system.

The ISO C standard only requires the signal names SIGABRT, SIGFPE, SIGILL, SIGINT, SIGSEGV, and SIGTERM to be defined.

The following signals shall be supported on all implementations (default actions are explained below the table):

Signal	Default Action	Description
SIGABRT	A	Process abort signal.
SIGALRM	T	Alarm clock.
SIGBUS	A	Access to an undefined portion of a memory object.
SIGCHLD	I	Child process terminated, stopped, or continued.
SIGCONT	C	Continue executing, if stopped.
SIGFPE	A	Erroneous arithmetic operation.
SIGHUP	T	Hangup.
SIGILL	A	Illegal instruction.
SIGINT	T	Terminal interrupt signal.
SIGKILL	T	Kill (cannot be caught or ignored).
SIGPIPE	T	Write on a pipe with no one to read it.
SIGQUIT	A	Terminal quit signal.
SIGSEGV	A	Invalid memory reference.
SIGSTOP	S	Stop executing (cannot be caught or ignored).
SIGTERM	T	Termination signal.
SIGTSTP	S	Terminal stop signal.
SIGTTIN	S	Background process attempting read.
SIGTTOU	S	Background process attempting write.
SIGUSR1	T	User-defined signal 1.
SIGUSR2	T	User-defined signal 2.
SIGPOLL	T	Pollable event.
SIGPROF	T	Profiling timer expired.
SIGSYS	A	Bad system call.
SIGTRAP	A	Trace/breakpoint trap.
SIGURG	I	High bandwidth data is available at a socket.
SIGVTALRM	T	Virtual timer expired.
SIGXCPU	A	CPU time limit exceeded.
SIGXFSZ	A	File size limit exceeded.

The default actions are as follows:

T Abnormal termination of the process. The process is terminated with all the consequences of *_exit*() except that the status made available to *wait*() and *waitpid*() indicates abnormal termination by the specified signal.

A Abnormal termination of the process.
Additionally, implementation-defined abnormal termination actions, such as creation of a **core** file, may occur.

I Ignore the signal.

S Stop the process.

C Continue the process, if it is stopped; otherwise, ignore the signal.

The header shall provide a declaration of **struct sigaction**, including at least the following members:

void (*sa_handler)(int) Pointer to a signal-catching function or one of the macros SIG_IGN or SIG_DFL.
sigset_t sa_mask Set of signals to be blocked during execution of the signal handling function.
int sa_flags Special flags.
void (*sa_sigaction)(int, siginfo_t *, void *)
Pointer to a signal-catching function.

The storage occupied by *sa_handler* and *sa_sigaction* may overlap, and a conforming application shall not use both simultaneously.

The following shall be declared as constants:

SA_NOCLDSTOP
Do not generate SIGCHLD when children stop or stopped children continue.

SIG_BLOCK
The resulting set is the union of the current set and the signal set pointed to by the argument *set*.

SIG_UNBLOCK
The resulting set is the intersection of the current set and the complement of the signal set pointed to by the argument *set*.

SIG_SETMASK
The resulting set is the signal set pointed to by the argument *set*.

SA_ONSTACK
Causes signal delivery to occur on an alternate stack.

SA_RESETHAND
Causes signal dispositions to be set to SIG_DFL on entry to signal handlers.

SA_RESTART
Causes certain functions to become restartable.

SA_SIGINFO
Causes extra information to be passed to signal handlers at the time of receipt of a signal.

SA_NOCLDWAIT
Causes implementations not to create zombie processes on child death.

SA_NODEFER
: Causes signal not to be automatically blocked on entry to signal handler.

SS_ONSTACK
: Process is executing on an alternate signal stack.

SS_DISABLE
: Alternate signal stack is disabled.

MINSIGSTKSZ
: Minimum stack size for a signal handler.

SIGSTKSZ
: Default size in bytes for the alternate signal stack.

The **ucontext_t** structure shall be defined through **typedef** as described in <*ucontext.h*>.

The **mcontext_t** type shall be defined through **typedef** as described in <*ucontext.h*>.

The <*signal.h*> header shall define the **stack_t** type as a structure that includes at least the following members:

void ***ss_sp** Stack base or pointer.
size_t **ss_size** Stack size.
int **ss_flags** Flags.

The <*signal.h*> header shall define the **sigstack** structure that includes at least the following members:

int **ss_onstack** Non-zero when signal stack is in use.
void ***ss_sp** Signal stack pointer.

The <*signal.h*> header shall define the **siginfo_t** type as a structure that includes at least the following members:

int **si_signo** Signal number.

int **si_errno** If non-zero, an *errno* value associated with this signal, as defined in **<errno.h>.**

int **si_code** Signal code.

pid_t **si_pid** Sending process ID.
uid_t **si_uid** Real user ID of sending process.
void ***si_addr** Address of faulting instruction.
int **si_status** Exit value or signal.
long **si_band** Band event for SIGPOLL.

union sigval **si_value** Signal value.

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_ILLADR	Illegal addressing mode.
	ILL_ILLTRP	Illegal trap.
	ILL_PRVOPC	Privileged opcode.
	ILL_PRVREG	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.
SIGTRAP	TRAP_BRKPT	Process breakpoint.
	TRAP_TRACE	Process trace 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_USER	Signal sent by *kill*().
	SI_QUEUE	Signal sent by the *sigqueue*().
	SI_TIMER	Signal generated by expiration of a timer set by *timer_settime*().
	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.

Implementations may support additional *si_code* values not included in this list, may generate values included in this list under circumstances other than those described in this list, and may contain extensions or limitations that prevent some values from being generated. Implementations do not generate a different value from the ones described in this list for circumstances described in this list.

In addition, the following signal-specific information shall be available:

Signal	Member	Value

SIGILL	**void *** *si_addr*	*Address of faulting instruction.*
SIGFPE		
SIGSEGV	**void *** *si_addr*	*Address of faulting memory reference.*
SIGBUS		
SIGCHLD	**pid_t** *si_pid*	*Child process ID.*
	int *si_status*	*Exit value or signal.*
	uid_t *si_uid*	*Real user ID of the process that sent the signal.*
SIGPOLL	**long** *si_band*	*Band event for POLL_IN, POLL_OUT, or POLL_MSG.*

For some implementations, the value of *si_addr* may be inaccurate.

The following shall be declared as functions and may also be defined as macros:

```
void (*bsd_signal(int, void (*)(int)))(int);

int    kill(pid_t, int);

int    killpg(pid_t, int);

int    pthread_kill(pthread_t, int);
int    pthread_sigmask(int, const sigset_t *, sigset_t *);

int    raise(int);

int    sigaction(int, const struct sigaction *restrict,
         struct sigaction *restrict);
int    sigaddset(sigset_t *, int);

int    sigaltstack(const stack_t *restrict, stack_t *restrict);

int    sigdelset(sigset_t *, int);
int    sigemptyset(sigset_t *);
int    sigfillset(sigset_t *);

int    sighold(int);
int    sigignore(int);
int    siginterrupt(int, int);

int    sigismember(const sigset_t *, int);

void (*signal(int, void (*)(int)))(int);

int    sigpause(int);

int    sigpending(sigset_t *);
int    sigprocmask(int, const sigset_t *restrict, sigset_t *restrict);
```

```
int    sigqueue(pid_t, int, const union sigval);


int    sigrelse(int);
void (*sigset(int, void (*)(int)))(int);


int    sigsuspend(const sigset_t *);


int    sigtimedwait(const sigset_t *restrict, siginfo_t *restrict,
         const struct timespec *restrict);


int    sigwait(const sigset_t *restrict, int *restrict);


int    sigwaitinfo(const sigset_t *restrict, siginfo_t *restrict);
```

Inclusion of the <*signal.h*> header may make visible all symbols from the <*time.h*> header.

The following sections are informative.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

<*errno.h*>, <*stropts.h*>, <*sys/types.h*>, <*time.h*>, <*ucontext.h*>, the System Interfaces volume of IEEE Std 1003.1-2001, *alarm*(), *bsd_signal*(), *ioctl*(), *kill*(), *killpg*(), *raise*(), *sigaction*(), *sigaddset*(), *sigaltstack*(), *sigdelset*(), *sigemptyset*(), *sigfillset*(), *siginterrupt*(), *sigismember*(), *signal*(), *sigpending*(), *sigprocmask*(), *sigqueue*(), *sigsuspend*(), *sigwaitinfo*(), *wait*(), *waitid*()

COPYRIGHT