NAME

signal - simplified software signal facilities

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <signal.h>

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

or in FreeBSD's equivalent but easier to read typedef'd version:
*typedef void (*sig_t) (int)*;

sig_t
signal(*int sig, sig_t func*);

DESCRIPTION

This **signal**() facility is a simplified interface to the more general sigaction(2) facility.

Signals allow the manipulation of a process from outside its domain as well as allowing the process to manipulate itself or copies of itself (children). There are two general types of signals: those that cause termination of a process and those that do not. Signals which cause termination of a program might result from an irrecoverable error or might be the result of a user at a terminal typing the 'interrupt' character. Signals are used when a process is stopped because it wishes to access its control terminal while in the background (see tty(4)). Signals are optionally generated when a process resumes after being stopped, when the status of child processes changes, or when input is ready at the control terminal. Most signals result in the termination of the process receiving them if no action is taken; some signals instead cause the process receiving them to be stopped, or are simply discarded if the process has not requested otherwise. Except for the SIGKILL and SIGSTOP signals, the **signal**() function allows for a signal to be caught, to be ignored, or to generate an interrupt. These signals are defined in the file *<signal.h>*:

Num

	Name	**Default Action**	**Description**
1	SIGHUP	terminate process	terminal line hangup
2	SIGINT	terminate process	interrupt program
3	SIGQUIT	create core image	quit program
4	SIGILL	create core image	illegal instruction

5	SIGTRAP	create core image	trace trap
6	SIGABRT	create core image	abort program (formerly SIGIOT)
7	SIGEMT	create core image	emulate instruction executed
8	SIGFPE	create core image	floating-point exception
9	SIGKILL	terminate process	kill program
10	SIGBUS	create core image	bus error
11	SIGSEGV	create core image	segmentation violation
12	SIGSYS	create core image	non-existent system call invoked
13	SIGPIPE	terminate process	write on a pipe with no reader
14	SIGALRM	terminate process	real-time timer expired
15	SIGTERM	terminate process	software termination signal
16	SIGURG	discard signal	urgent condition present on socket
17	SIGSTOP	stop process	stop (cannot be caught or ignored)
18	SIGTSTP	stop process	stop signal generated from keyboard
19	SIGCONT	discard signal	continue after stop
20	SIGCHLD	discard signal	child status has changed
21	SIGTTIN	stop process	background read attempted from control terminal
22	SIGTTOU	stop process	background write attempted to control terminal
23	SIGIO	discard signal	I/O is possible on a descriptor (see fcntl(2))
24	SIGXCPU	terminate process	cpu time limit exceeded (see setrlimit(2))
25	SIGXFSZ	terminate process	file size limit exceeded (see setrlimit(2))
26	SIGVTALRM	terminate process	virtual time alarm (see setitimer(2))
27	SIGPROF	terminate process	profiling timer alarm (see setitimer(2))
28	SIGWINCH	discard signal	Window size change
29	SIGINFO	discard signal	status request from keyboard
30	SIGUSR1	terminate process	User defined signal 1
31	SIGUSR2	terminate process	User defined signal 2
32	SIGTHR	terminate process	thread interrupt
33	SIGLIBRT	terminate process	real-time library interrupt

The *sig* argument specifies which signal was received. The *func* procedure allows a user to choose the action upon receipt of a signal. To set the default action of the signal to occur as listed above, *func* should be SIG_DFL. A SIG_DFL resets the default action. To ignore the signal *func* should be SIG_IGN. This will cause subsequent instances of the signal to be ignored and pending instances to be discarded. If SIG_IGN is not used, further occurrences of the signal are automatically blocked and *func* is called.

The handled signal is unblocked when the function returns and the process continues from where it left off when the signal occurred. **Unlike previous signal facilities, the handler func() remains installed after a signal has been delivered.**

For some system calls, if a signal is caught while the call is executing and the call is prematurely terminated, the call is automatically restarted. Any handler installed with signal(3) will have the SA_RESTART flag set, meaning that any restartable system call will not return on receipt of a signal. The affected system calls include read(2), write(2), sendto(2), recvfrom(2), sendmsg(2) and recvmsg(2) on a communications channel or a low speed device and during a ioctl(2) or wait(2). However, calls that have already committed are not restarted, but instead return a partial success (for example, a short read count). These semantics could be changed with siginterrupt(3).

When a process which has installed signal handlers forks, the child process inherits the signals. All caught signals may be reset to their default action by a call to the execve(2) function; ignored signals remain ignored.

If a process explicitly specifies SIG_IGN as the action for the signal SIGCHLD, the system will not create zombie processes when children of the calling process exit. As a consequence, the system will discard the exit status from the child processes. If the calling process subsequently issues a call to wait(2) or equivalent, it will block until all of the calling process's children terminate, and then return a value of -1 with *errno* set to ECHILD.

See sigaction(2) for a list of functions that are considered safe for use in signal handlers.

RETURN VALUES

The previous action is returned on a successful call. Otherwise, SIG_ERR is returned and the global variable *errno* is set to indicate the error.

ERRORS

The **signal**() function will fail and no action will take place if one of the following occur:

[EINVAL] The *sig* argument is not a valid signal number.

[EINVAL] An attempt is made to ignore or supply a handler for SIGKILL or SIGSTOP.

SEE ALSO

kill(1), kill(2), ptrace(2), sigaction(2), sigaltstack(2), sigprocmask(2), sigsuspend(2), wait(2), fpsetmask(3), setjmp(3), siginterrupt(3), tty(4)

HISTORY

The **signal**() function appeared in Version 4 AT&T UNIX. The current **signal** facility appeared in 4.0BSD. The option to avoid the creation of child zombies through ignoring SIGCHLD appeared in FreeBSD 5.0.