Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'signal.3p' command

$ man signal.3p

SIGNAL(3P) POSIX Programmer's Manual SIGNAL(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

signal ? signal management

SYNOPSIS

#include <signal.h>

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

DESCRIPTION

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1?2017 defers to the ISO C standard.

The signal() function chooses one of three ways in which receipt of the signal number sig is to be subsequently handled. If the value of func is SIG_DFL, default handling for that signal shall occur. If the value of func is SIG_IGN, the signal shall be ignored. Otherwise, the appli? cation shall ensure that func points to a function to be called when that signal occurs. An invocation of such a function because of a sig? nal, or (recursively) of any further functions called by that invoca?

tion (other than functions in the standard library), is called a ``sig?
nal handler''.
When a signal occurs, and func points to a function, it is implementa?
tion-defined whether the equivalent of a:

    signal(sig, SIG_DFL);

is executed or the implementation prevents some implementation-defined
set of signals (at least including sig) from occurring until the cur?
rent signal handling has completed. (If the value of sig is SIGILL, the
implementation may alternatively define that no action is taken.) Next
the equivalent of:

    (*func)(sig);

is executed. If and when the function returns, if the value of sig was
SIGFPE, SIGILL, or SIGSEGV or any other implementation-defined value
corresponding to a computational exception, the behavior is undefined.
Otherwise, the program shall resume execution at the point it was in?
terrupted. The ISO C standard places a restriction on applications re?
lating to the use of raise() from signal handlers. This restriction
does not apply to POSIX applications, as POSIX.1?2008 requires raise()
to be async-signal-safe (see Section 2.4.3, Signal Actions).
If the process is multi-threaded, or if the process is single-threaded
and a signal handler is executed other than as the result of:

* The process calling abort(), raise(), kill(), pthread_kill(), or
  sigqueue() to generate a signal that is not blocked
* A pending signal being unblocked and being delivered before the
  call that unblocked it returns

the behavior is undefined if the signal handler refers to any object
other than errno with static storage duration other than by assigning a
value to an object declared as volatile sig_atomic_t, or if the signal
handler calls any function defined in this standard other than one of
the functions listed in Section 2.4, Signal Concepts.
At program start-up, the equivalent of:

    signal(sig, SIG_IGN);

is executed for some signals, and the equivalent of:

signal(sig, SIG_DFL);

is executed for all other signals (see exec).

The signal() function shall not change the setting of errno if success?

ful.

RETURN VALUE

If the request can be honored, signal() shall return the value of func

for the most recent call to signal() for the specified signal sig.

Otherwise, SIG_ERR shall be returned and a positive value shall be

stored in errno.

ERRORS

The signal() function shall fail if:

EINVAL The sig argument is not a valid signal number or an attempt is

made to catch a signal that cannot be caught or ignore a signal

that cannot be ignored.

The signal() function may fail if:

EINVAL An attempt was made to set the action to SIG_DFL for a signal

that cannot be caught or ignored (or both).

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

The sigaction() function provides a more comprehensive and reliable

mechanism for controlling signals; new applications should use sigac?

tion() rather than signal().

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.4, Signal Concepts, exec, pause(), raise(), sigaction(), sig?

suspend(), waitid()

The Base Definitions volume of POSIX.1?2017, <signal.h>

COPYRIGHT