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 'sigaltstack.3p' command

$ man sigaltstack.3p

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

sigaltstack ? set and get signal alternate stack context

SYNOPSIS

```
#include <signal.h>
int sigaltstack(const stack_t *restrict ss, stack_t *restrict oss);
```

DESCRIPTION

The sigaltstack() function allows a process to define and examine the state of an alternate stack for signal handlers for the current thread.

Signals that have been explicitly declared to execute on the alternate stack shall be delivered on the alternate stack.

If ss is not a null pointer, it points to a stack_t structure that specifies the alternate signal stack that shall take effect upon return from sigaltstack(). The ss_flags member specifies the new stack state. If it is set to SS_DISABLE, the stack is disabled and ss_sp and ss_size are ignored. Otherwise, the stack shall be enabled, and the ss_sp and ss_size members specify the new address and size of the stack.

The range of addresses starting at ss_sp up to but not including

ss_sp+ss_size is available to the implementation for use as the stack.

This function makes no assumptions regarding which end is the stack base and in which direction the stack grows as items are pushed.

If oss is not a null pointer, upon successful completion it shall point to a stack_t structure that specifies the alternate signal stack that was in effect prior to the call to sigaltstack(). The ss_sp and ss_size members specify the address and size of that stack. The ss_flags member specifies the stack's state, and may contain one of the following values:

SS_ONSTACK The process is currently executing on the alternate signal stack. Attempts to modify the alternate signal stack while the process is executing on it fail. This flag shall not be modified by processes.

SS_DISABLE The alternate signal stack is currently disabled.

The value SIGSTKSZ is a system default specifying the number of bytes that would be used to cover the usual case when manually allocating an alternate stack area. The value MINSIGSTKSZ is defined to be the mini? mum stack size for a signal handler. In computing an alternate stack size, a program should add that amount to its stack requirements to al? low for the system implementation overhead. The constants SS_ONSTACK, SS_DISABLE, SIGSTKSZ, and MINSIGSTKSZ are defined in <signal.h>.

After a successful call to one of the exec functions, there are no al? ternate signal stacks in the new process image.

In some implementations, a signal (whether or not indicated to execute on the alternate stack) shall always execute on the alternate stack if it is delivered while another signal is being caught using the alter? nate stack.

Use of this function by library threads that are not bound to kernel-scheduled entities results in undefined behavior.

RETURN VALUE

Upon successful completion, sigaltstack() shall return 0; otherwise, it shall return -1 and set errno to indicate the error.

ERRORS

The sigaltstack() function shall fail if:

EINVAL The ss argument is not a null pointer, and the ss_flags member pointed to by ss contains flags other than SS_DISABLE.

ENOMEM The size of the alternate stack area is less than MINSIGSTKSZ.

EPERM An attempt was made to modify an active stack.

The following sections are informative.

EXAMPLES

Allocating Memory for an Alternate Stack

The following example illustrates a method for allocating memory for an alternate stack.

```
#include <signal.h>
...
if ((sigstk.ss_sp = malloc(SIGSTKSZ)) == NULL)
    /* Error return. */
sigstk.ss_size = SIGSTKSZ;
sigstk.ss_flags = 0;
if (sigaltstack(&sigstk,(stack_t *)0) < 0)
    perror("sigaltstack");
```

APPLICATION USAGE

On some implementations, stack space is automatically extended as needed. On those implementations, automatic extension is typically not available for an alternate stack. If the stack overflows, the behavior is undefined.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.4, Signal Concepts, exec, sigaction(), sigsetjmp()

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

COPYRIGHT

table Operating System Interface (POSIX), The Open Group Base Specifi? cations Issue 7, 2018 Edition, Copyright (C) 2018 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

Any typographical or formatting errors that appear in this page are most likely to have been introduced during the conversion of the source files to man page format. To report such errors, see https://www.ker? nel.org/doc/man-pages/reporting_bugs.html .