NAME

sigaltstack - set and/or get signal stack context

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

```
#include <signal.h>

typedef struct {
        char    *ss_sp;
        size_t  ss_size;
        int     ss_flags;
} stack_t;
int
sigaltstack(const stack_t * restrict ss, stack_t * restrict oss);
```

DESCRIPTION

The **sigaltstack**() system call allows defining an alternate stack on which signals are to be processed for the current thread. If *ss* is non-zero, it specifies a pointer to and the size of a *signal stack* on which to deliver signals. When a signal's action indicates its handler should execute on the signal stack (specified with a sigaction(2) system call), the system checks to see if the thread is currently executing on that stack. If the thread is not currently executing on the signal stack, the system arranges a switch to the signal stack for the duration of the signal handler's execution.

An active stack cannot be modified.

If SS_DISABLE is set in *ss_flags*, *ss_sp* and *ss_size* are ignored and the signal stack will be disabled. A disabled stack will cause all signals to be taken on the regular user stack. If the stack is later re-enabled then all signals that were specified to be processed on an alternate stack will resume doing so.

If *oss* is non-zero, the current signal stack state is returned. The *ss_flags* field will contain the value SS_ONSTACK if the thread is currently on a signal stack and SS_DISABLE if the signal stack is currently disabled.

NOTES

The value SIGSTKSZ is defined to be the number of bytes/chars that would be used to cover the usual case when allocating an alternate stack area. The following code fragment is typically used to allocate an alternate stack.

```
if ((sigstk.ss_sp = malloc(SIGSTKSZ)) == NULL)
        /* error return */
sigstk.ss_size = SIGSTKSZ;
sigstk.ss_flags = 0;
if (sigaltstack(&sigstk, NULL) < 0)
        perror("sigaltstack");
```

An alternative approach is provided for programs with signal handlers that require a specific amount of stack space other than the default size. The value MINSIGSTKSZ is defined to be the number of bytes/chars that is required by the operating system to implement the alternate stack feature. In computing an alternate stack size, programs should add MINSIGSTKSZ to their stack requirements to allow for the operating system overhead.

Signal stacks are automatically adjusted for the direction of stack growth and alignment requirements. Signal stacks may or may not be protected by the hardware and are not ‘‘grown’’ automatically as is done for the normal stack. If the stack overflows and this space is not protected unpredictable results may occur.

RETURN VALUES

The **sigaltstack**() function returns the value 0 if successful; otherwise the value -1 is returned and the global variable *errno* is set to indicate the error.

ERRORS

The **sigaltstack**() system call will fail and the signal stack context will remain unchanged if one of the following occurs.

[EFAULT]	Either *ss* or *oss* points to memory that is not a valid part of the process address space.
[EPERM]	An attempt was made to modify an active stack.
[EINVAL]	The *ss_flags* field was invalid.
[ENOMEM]	Size of alternate stack area is less than or equal to MINSIGSTKSZ.

SEE ALSO

sigaction(2), setjmp(3)

HISTORY

The predecessor to **sigaltstack**(), the **sigstack**() system call, appeared in 4.2BSD.