NAME

sigaltstack – set or get signal alternate stack context

SYNOPSIS

#include <signal.h>

int sigaltstack(const stack_t **ss*, **stack_t** **oss***);**

DESCRIPTION

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

The structure **stack_t** includes the following members:

```
int     *ss_sp
long    ss_size
int     ss_flags
```

If *ss* is not **NULL**, it points to a structure specifying the alternate signal stack that will take effect upon successful return from **sigaltstack( )**. The **ss_sp** and **ss_size** fields specify the new base and size of the stack, which is automatically adjusted for direction of growth and alignment. The **ss_flags** field specifies the new stack state and may be set to the following:

SS_DISABLE The stack is to be disabled and **ss_sp** and **ss_size** are ignored. If **SS_DISABLE** is not set, the stack will be enabled.

If *oss* is not **NULL**, it points to a structure specifying the alternate signal stack that was in effect prior to the call to **sigaltstack( )**. The **ss_sp** and **ss_size** fields specify the base and size of that stack. The **ss_flags** field specifies the stack's state, and may contain the following values:

SS_ONSTACK The LWP is currently executing on the alternate signal stack. Attempts to modify the alternate signal stack while the LWP is executing on it will fail.

SS_DISABLE The alternate signal stack is currently disabled.

RETURN VALUES

On success, **sigaltstack( )** returns zero. On failure, it returns −1 and sets **errno** to indicate the error.

ERRORS

sigaltstack( ) fails if any of the following is true:

EFAULT *ss* or *oss* points to an illegal address.

EINVAL An attempt was made to disable an active stack or the **ss_flags** field in *ss* specifies invalid flags.

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

SEE ALSO

getcontext(2), **sigaction**(2), **ucontext**(5)

NOTES

The value **SIGSTKSZ** is defined to be the number of bytes that would be used to cover the usual case when allocating an alternate stack area. The value **MINSIGSTKSZ** is defined to be the minimum stack size for a signal handler. In computing an alternate stack size, a program should add that amount to its stack requirements to allow for the operating system overhead.

The following code fragment is typically used to allocate an alternate stack:

```
if ((sigstk.ss_sp = (char *)malloc(SIGSTKSZ)) == NULL)
        /* error return */;

sigstk.ss_size = SIGSTKSZ;
sigstk.ss_flags = 0;
if (sigaltstack(&sigstk, (stack_t *)0) < 0)
        perror("sigaltstack");
```