NAME

sigstack – set and/or get signal stack context

SYNOPSIS

/usr/ucb/cc [*flag* ...] *file* ...

#include <signal.h>

int sigstack(*nss*, *oss***)**
struct sigstack **nss*, **oss***;**

DESCRIPTION

sigstack( ) allows users to define an alternate stack, called the ''signal stack'', on which signals are to be processed. When a signal's action indicates its handler should execute on the signal stack (specified with a **sigvec**(3B) call), the system checks to see if the process is currently executing on that stack. If the process 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.

A signal stack is specified by a **sigstack( )** structure, which includes the following members:

```
char    *ss_sp;        /* signal stack pointer */
int     ss_onstack;    /* current status */
```

ss_sp is the initial value to be assigned to the stack pointer when the system switches the process to the signal stack. Note that, on machines where the stack grows downwards in memory, this is *not* the address of the beginning of the signal stack area. **ss_onstack** field is zero or non-zero depending on whether the process is currently executing on the signal stack or not.

If *nss* is not a **NULL** pointer, **sigstack( )** sets the signal stack state to the value in the **sigstack( )** structure pointed to by *nss*. Note: if **ss_onstack** is non-zero, the system will think that the process is executing on the signal stack. If *nss* is a **NULL** pointer, the signal stack state will be unchanged. If *oss* is not a **NULL** pointer, the current signal stack state is stored in the **sigstack( )** structure pointed to by *oss*.

RETURN VALUES

Upon successful completion, a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

ERRORS

sigstack( ) will fail and the signal stack context will remain unchanged if one of the following occurs.

EFAULT Either *nss* or *oss* points to memory that is not a valid part of the process address space.

SEE ALSO

sigaltstack(2), **sigvec**(3B), **signal**(3C)

NOTES

Use of these interfaces should be restricted to only applications written on BSD platforms. Use of these interfaces with any of the system libraries or in multi-thread applications is unsupported.

WARNINGS

Signal stacks are not "grown" automatically, as is done for the normal stack. If the stack overflows unpredictable results may occur.