NAME

pthread_attr_setstack, pthread_attr_getstack − set/get stack attributes in thread attributes object

LIBRARY

POSIX threads library (*libpthread*, *−lpthread*)

SYNOPSIS

```
#include <pthread.h>

int pthread_attr_setstack(pthread_attr_t *attr,
                          void stackaddr[.stacksize],
                          size_t stacksize);
int pthread_attr_getstack(const pthread_attr_t *restrict attr,
                          void **restrict stackaddr,
                          size_t *restrict stacksize);
```

Feature Test Macro Requirements for glibc (see **feature_test_macros**(7)):

pthread_attr_getstack(), **pthread_attr_setstack**():
  _POSIX_C_SOURCE >= 200112L

DESCRIPTION

The **pthread_attr_setstack**() function sets the stack address and stack size attributes of the thread attributes object referred to by *attr* to the values specified in *stackaddr* and *stacksize*, respectively. These attributes specify the location and size of the stack that should be used by a thread that is created using the thread attributes object *attr*.

stackaddr should point to the lowest addressable byte of a buffer of *stacksize* bytes that was allocated by the caller. The pages of the allocated buffer should be both readable and writable.

The **pthread_attr_getstack**() function returns the stack address and stack size attributes of the thread attributes object referred to by *attr* in the buffers pointed to by *stackaddr* and *stacksize*, respectively.

RETURN VALUE

On success, these functions return 0; on error, they return a nonzero error number.

ERRORS

pthread_attr_setstack() can fail with the following error:

EINVAL
: *stacksize* is less than **PTHREAD_STACK_MIN** (16384) bytes. On some systems, this error may also occur if *stackaddr* or *stackaddr + stacksize* is not suitably aligned.

POSIX.1 also documents an **EACCES** error if the stack area described by *stackaddr* and *stacksize* is not both readable and writable by the caller.

VERSIONS

These functions are provided since glibc 2.2.

ATTRIBUTES

For an explanation of the terms used in this section, see **attributes**(7).

Interface	**Attribute**	**Value**
pthread_attr_setstack(), **pthread_attr_getstack**()	Thread safety	MT-Safe

STANDARDS

POSIX.1-2001, POSIX.1-2008.

NOTES

These functions are provided for applications that must ensure that a thread’s stack is placed in a particular location. For most applications, this is not necessary, and the use of these functions should be avoided. (Use **pthread_attr_setstacksize**(3) if an application simply requires a stack size other than the default.)

When an application employs **pthread_attr_setstack**(), it takes over the responsibility of allocating the stack. Any guard size value that was set using **pthread_attr_setguardsize**(3) is ignored. If deemed necessary, it is the application's responsibility to allocate a guard area (one or more pages protected against reading and writing) to handle the possibility of stack overflow.

The address specified in *stackaddr* should be suitably aligned: for full portability, align it on a page boundary (*sysconf(_SC_PAGESIZE)*). **posix_memalign**(3) may be useful for allocation. Probably, *stacksize* should also be a multiple of the system page size.

If *attr* is used to create multiple threads, then the caller must change the stack address attribute between calls to **pthread_create**(3); otherwise, the threads will attempt to use the same memory area for their stacks, and chaos will ensue.

EXAMPLES

See **pthread_attr_init**(3).

SEE ALSO

mmap(2), **mprotect**(2), **posix_memalign**(3), **pthread_attr_init**(3), **pthread_attr_setguardsize**(3), **pthread_attr_setstackaddr**(3), **pthread_attr_setstacksize**(3), **pthread_create**(3), **pthreads**(7)