

**NAME**

pthread\_attr\_setstack, pthread\_attr\_getstack – set/get stack attributes in thread attributes object

**SYNOPSIS**

```
#include <pthread.h>
```

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

Compile and link with `-pthread`.

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 by glibc since version 2.2.

**ATTRIBUTES**

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

Interface	Attribute	Value
<b>pthread_attr_setstack()</b> , <b>pthread_attr_getstack()</b>	Thread safety	MT-Safe

**CONFORMING TO**

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.

### EXAMPLE

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)

### COLOPHON

This page is part of release 5.02 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at <https://www.kernel.org/doc/man-pages/>.