NAME

get_thread_area, set_thread_area - set a GDT entry for thread-local storage

SYNOPSIS

#include <linux/unistd.h>
#include <asm/ldt.h>

int get_thread_area(struct user_desc *u_info**);**
int set_thread_area(struct user_desc *u_info**);**

Note: There are no glibc wrappers for these system calls; see NOTES.

DESCRIPTION

Linux dedicates three global descriptor table (GDT) entries for thread-local storage. For more information about the GDT, see the Intel Software Developer's Manual or the AMD Architecture Programming Manual.

Both of these system calls take an argument that is a pointer to a structure of the following type:

```
struct user_desc {
unsigned int entry_number;
unsigned long base_addr;
unsigned int limit;
unsigned int seg_32bit:1;
unsigned int contents:2;
unsigned int read_exec_only:1;
unsigned int limit_in_pages:1;
unsigned int seg_not_present:1;
unsigned int useable:1;
};
```

get_thread_area() reads the GDT entry indicated by *u_info->entry_number* and fills in the rest of the fields in *u_info*.

set_thread_area() sets a TLS entry in the GDT.

The TLS array entry set by **set_thread_area**() corresponds to the value of *u_info->entry_number* passed in by the user. If this value is in bounds, **set_thread_area**() writes the TLS descriptor pointed to by *u_info* into the thread's TLS array.

When **set_thread_area**() is passed an *entry_number* of -1, it searches for a free TLS entry. If **set_thread_area**() finds a free TLS entry, the value of *u_info->entry_number* is set upon return to show which entry was changed.

A *user_desc* is considered "empty" if *read_exec_only* and *seg_not_present* are set to 1 and all of the other fields are 0. If an "empty" descriptor is passed to **set_thread_area,** the corresponding TLS entry will be cleared. See BUGS for additional details.

Since Linux 3.19, **set_thread_area**() cannot be used to write non-present segments, 16-bit segments, or code segments, although clearing a segment is still acceptable.

RETURN VALUE

These system calls return 0 on success, and -1 on failure, with *errno* set appropriately.

ERRORS

EFAULT

u_info is an invalid pointer.

EINVAL

u_info->entry_number is out of bounds.

ENOSYS

get_thread_area() or **set_thread_area**() was invoked as a 64-bit system call.

ESRCH
    (**set_thread_area**()) A free TLS entry could not be located.

VERSIONS
set_thread_area() first appeared in Linux 2.5.29. **get_thread_area**() first appeared in Linux 2.5.32.

CONFORMING TO
set_thread_area() is Linux-specific and should not be used in programs that are intended to be portable.

NOTES
Glibc does not provide wrappers for these system calls, since they are generally intended for use only by threading libraries. In the unlikely event that you want to call them directly, use syscall(2).

arch_prctl(2) can interfere with **set_thread_area**(). See arch_prctl(2) for more details. This is not normally a problem, as arch_prctl(2) is normally used only by 64-bit programs.

BUGS
On 64-bit kernels before Linux 3.19, one of the padding bits in *user_desc*, if set, would prevent the descriptor from being considered empty (see modify_ldt(2)). As a result, the only reliable way to clear a TLS entry is to use memset(3) to zero the entire *user_desc* structure, including padding bits, and then to set the *read_exec_only* and *seg_not_present* bits. On Linux 3.19, a *user_desc* consisting entirely of zeros except for *entry_number* will also be interpreted as a request to clear a TLS entry, but this behaved differently on older kernels.

Prior to Linux 3.19, the DS and ES segment registers must not reference TLS entries.

SEE ALSO
arch_prctl(2), modify_ldt(2)

COLOPHON
This page is part of release 4.10 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/.