Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'modify_ldt.2' command

$ man modify_ldt.2

MODIFY_LDT(2) Linux Programmer's Manual MODIFY_LDT(2)

NAME

modify_ldt - get or set a per-process LDT entry

SYNOPSIS

#include <sys/types.h>

int modify_ldt(int func, void *ptr, unsigned long bytecount);

Note: There is no glibc wrapper for this system call; see NOTES.

DESCRIPTION

modify_ldt() reads or writes the local descriptor table (LDT) for a process. The LDT is an array of segment descriptors that can be refer? enced by user code. Linux allows processes to configure a per-process (actually per-mm) LDT. For more information about the LDT, see the In? tel Software Developer's Manual or the AMD Architecture Programming Manual.

When func is 0, modify_ldt() reads the LDT into the memory pointed to by ptr. The number of bytes read is the smaller of bytecount and the actual size of the LDT, although the kernel may act as though the LDT is padded with additional trailing zero bytes. On success, mod? ify_ldt() will return the number of bytes read.

When func is 1 or 0x11, modify_ldt() modifies the LDT entry indicated by ptr->entry_number. ptr points to a user_desc structure and byte? count must equal the size of this structure.

The user_desc structure is defined in <asm/ldt.h> as:

```
struct user_desc {
    unsigned int  entry_number;
    unsigned int  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;
};
```

In Linux 2.4 and earlier, this structure was named modify_ldt_ldt_s.

The contents field is the segment type (data, expand-down data, non-conforming code, or conforming code). The other fields match their de? scriptions in the CPU manual, although modify_ldt() cannot set the hardware-defined "accessed" bit described in the CPU manual.

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. An LDT entry can be cleared by setting it to an "empty" user_desc or, if func is 1, by set? ting both base and limit to 0.

A conforming code segment (i.e., one with contents==3) will be rejected if func is 1 or if seg_not_present is 0.

When func is 2, modify_ldt() will read zeros. This appears to be a leftover from Linux 2.4.

RETURN VALUE

On success, modify_ldt() returns either the actual number of bytes read (for reading) or 0 (for writing). On failure, modify_ldt() returns -1 and sets errno to indicate the error.

ERRORS

EFAULT ptr points outside the address space.

EINVAL ptr is 0, or func is 1 and bytecount is not equal to the size of the structure user_desc, or func is 1 or 0x11 and the new LDT entry has invalid values.

ENOSYS func is neither 0, 1, 2, nor 0x11.

CONFORMING TO

This call is Linux-specific and should not be used in programs intended to be portable.

NOTES

Glibc does not provide a wrapper for this system call; call it using syscall(2).

modify_ldt() should not be used for thread-local storage, as it slows down context switches and only supports a limited number of threads. Threading libraries should use set_thread_area(2) or arch_prctl(2) in? stead, except on extremely old kernels that do not support those system calls.

The normal use for modify_ldt() is to run legacy 16-bit or segmented 32-bit code. Not all kernels allow 16-bit segments to be installed, however.

Even on 64-bit kernels, modify_ldt() cannot be used to create a long mode (i.e., 64-bit) code segment. The undocumented field "lm" in user_desc is not useful, and, despite its name, does not result in a long mode segment.

BUGS

On 64-bit kernels before Linux 3.19, setting the "lm" bit in user_desc prevents the descriptor from being considered empty. Keep in mind that the "lm" bit does not exist in the 32-bit headers, but these buggy ker? nels will still notice the bit even when set in a 32-bit process.

SEE ALSO

arch_prctl(2), set_thread_area(2), vm86(2)

COLOPHON

This page is part of release 5.10 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/.