Rocky Enterprise Linux 9.2 Manual Pages on command 'delete_module.2'

$ man delete_module.2

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

NAME

delete_module - unload a kernel module

SYNOPSIS

int delete_module(const char *name, int flags);

Note: No declaration of this system call is provided in glibc headers; see NOTES.

DESCRIPTION

The delete_module() system call attempts to remove the unused loadable module entry iden?

tified by name. If the module has an exit function, then that function is executed before

unloading the module. The flags argument is used to modify the behavior of the system

call, as described below. This system call requires privilege.

Module removal is attempted according to the following rules:

1. If there are other loaded modules that depend on (i.e., refer to symbols defined in)

this module, then the call fails.

2. Otherwise, if the reference count for the module (i.e., the number of processes cur?

rently using the module) is zero, then the module is immediately unloaded.

3. If a module has a nonzero reference count, then the behavior depends on the bits set

in flags. In normal usage (see NOTES), the O_NONBLOCK flag is always specified, and

the O_TRUNC flag may additionally be specified.

The various combinations for flags have the following effect:

flags == O_NONBLOCK

The call returns immediately, with an error.

flags == (O_NONBLOCK | O_TRUNC)

The module is unloaded immediately, regardless of whether it has a nonzero ref?

erence count.

(flags & O_NONBLOCK) == 0

If flags does not specify O_NONBLOCK, the following steps occur:

* The module is marked so that no new references are permitted.

* If the module's reference count is nonzero, the caller is placed in an unin?
terruptible sleep state (TASK_UNINTERRUPTIBLE) until the reference count is
zero, at which point the call unblocks.

* The module is unloaded in the usual way.

The O_TRUNC flag has one further effect on the rules described above. By default, if a module has an init function but no exit function, then an attempt to remove the module fails. However, if O_TRUNC was specified, this requirement is bypassed.

Using the O_TRUNC flag is dangerous! If the kernel was not built with CONFIG_MOD?
ULE_FORCE_UNLOAD, this flag is silently ignored. (Normally, CONFIG_MODULE_FORCE_UNLOAD is enabled.) Using this flag taints the kernel (TAINT_FORCED_RMMOD).

RETURN VALUE

On success, zero is returned. On error, -1 is returned and errno is set appropriately.

ERRORS

EBUSY The module is not "live" (i.e., it is still being initialized or is already marked for removal); or, the module has an init function but has no exit function, and O_TRUNC was not specified in flags.

EFAULT name refers to a location outside the process's accessible address space.

ENOENT No module by that name exists.

EPERM The caller was not privileged (did not have the CAP_SYS_MODULE capability), or mod?
ule unloading is disabled (see /proc/sys/kernel/modules_disabled in proc(5)).

EWOULDBLOCK

Other modules depend on this module; or, O_NONBLOCK was specified in flags, but the reference count of this module is nonzero and O_TRUNC was not specified in flags.

CONFORMING TO

delete_module() is Linux-specific.

NOTES

The delete_module() system call is not supported by glibc. No declaration is provided in glibc headers, but, through a quirk of history, glibc versions before 2.23 did export an

ABI for this system call. Therefore, in order to employ this system call, it is (before glibc 2.23) sufficient to manually declare the interface in your code; alternatively, you can invoke the system call using syscall(2).

The uninterruptible sleep that may occur if O_NONBLOCK is omitted from flags is considered undesirable, because the sleeping process is left in an unkillable state. As at Linux 3.7, specifying O_NONBLOCK is optional, but in future kernels it is likely to become mandatory.

Linux 2.4 and earlier

In Linux 2.4 and earlier, the system call took only one argument:

```
int delete_module(const char *name);
```

If name is NULL, all unused modules marked auto-clean are removed.

Some further details of differences in the behavior of delete_module() in Linux 2.4 and earlier are not currently explained in this manual page.

SEE ALSO

create_module(2), init_module(2), query_module(2), lsmod(8), modprobe(8), rmmod(8)

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/.