NAME

dlsym, dlvsym − obtain address of a symbol in a shared object or executable

SYNOPSIS

#include <dlfcn.h>

void *dlsym(void *_handle_, const char *_symbol_);

#define _GNU_SOURCE
#include <dlfcn.h>

void *dlvsym(void *_handle_, char *_symbol_, char *_version_);

Link with *−ldl*.

DESCRIPTION

The function **dlsym**() takes a "handle" of a dynamic loaded shared object returned by **dlopen**(3) along with a null-terminated symbol name, and returns the address where that symbol is loaded into memory. If the symbol is not found, in the specified object or any of the shared objects that were automatically loaded by **dlopen**(3) when that object was loaded, **dlsym**() returns NULL. (The search performed by **dlsym**() is breadth first through the dependency tree of these shared objects.)

In unusual cases (see NOTES) the value of the symbol could actually be NULL. Therefore, a NULL return from **dlsym**() need not indicate an error. The correct way to distinguish an error from a symbol whose value is NULL is to call **dlerror**(3) to clear any old error conditions, then call **dlsym**(), and then call **dlerror**(3) again, saving its return value into a variable, and check whether this saved value is not NULL.

There are two special pseudo-handles that may be specified in *handle*:

RTLD_DEFAULT
: Find the first occurrence of the desired symbol using the default shared object search order. The search will include global symbols in the executable and its dependencies, as well as symbols in shared objects that were dynamically loaded with the **RTLD_GLOBAL** flag.

RTLD_NEXT
: Find the next occurrence of the desired symbol in the search order after the current object. This allows one to provide a wrapper around a function in another shared object, so that, for example, the definition of a function in a preloaded shared object (see **LD_PRELOAD** in **ld.so**(8)) can find and invoke the "real" function provided in another shared object (or for that matter, the "next" definition of the function in cases where there are multiple layers of preloading).

The **_GNU_SOURCE** feature test macro must be defined in order to obtain the definitions of **RTLD_DEFAULT** and **RTLD_NEXT** from *<dlfcn.h>*.

The function **dlvsym**() does the same as **dlsym**() but takes a version string as an additional argument.

RETURN VALUE

On success, these functions return the address associated with *symbol*. On failure, they return NULL; the cause of the error can be diagnosed using **dlerror**(3).

VERSIONS

dlsym() is present in glibc 2.0 and later. **dlvsym**() first appeared in glibc 2.1.

ATTRIBUTES

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

Interface	Attribute	Value
dlsym(), **dlvsym**()	Thread safety	MT-Safe

CONFORMING TO

POSIX.1-2001 describes **dlsym**(). The **dlvsym**() function is a GNU extension.

NOTES

The value of a symbol returned by **dlsym**() will never be NULL if the shared object is the result of normal compilation, since a global symbol is never placed at the NULL address. There are nevertheless cases

where a lookup using **dlsym**() may return NULL as the value of a symbol. For example, the symbol value may be the result of a GNU indirect function (IFUNC) resolver function that returns NULL as the resolved value.

History

The **dlsym**() function is part of the dlopen API, derived from SunOS. That system does not have **dlvsym**().

EXAMPLE

See **dlopen**(3).

SEE ALSO

dl_iterate_phdr(3), **dladdr**(3), **dlerror**(3), **dlinfo**(3), **dlopen**(3), **ld.so**(8)

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/.