NAME

dlsym – get the address of a symbol in a shared object

SYNOPSIS

cc [*flag* ...] *file* ... **–ldl** [*library* ...]

#include <dlfcn.h>

void ∗**dlsym(void** ∗*handle*, **const char** ∗*name***);**

MT-LEVEL

MT-Safe

DESCRIPTION

dlsym( ) is one of a family of routines that give the user direct access to the dynamic linking facilities. (SEE *Linker and Libraries Guide*). These routines are made available via the library loaded when the option **-ldl** is passed to the link-editor.

These routines are available to dynamically linked processes ONLY.

dlsym( ) allows a process to obtain the address of a symbol defined within a shared object. *handle* is either the value returned from a call to **dlopen( )**, or the special flag **RTLD_NEXT**. *name* is the symbol's name as a character string.

In the former case the corresponding shared object must not have been closed using **dlclose( )**. **dlsym( )** will search for the named symbol in all shared objects loaded automatically as a result of loading the object referenced by *handle* (see **dlopen( )).**

In the latter case **dlsym( )** will search for the named symbol in the objects that were loaded following the object from which the **dlsym( )** call is being made. If these subsequent objects were loaded from **dlopen( )** calls, **dlsym( )** will search the object only if the caller is part of the same **dlopen( )** dependency hierarchy, or the object was given global search access (see **dlopen( )** with reference to the mode **RTLD_GLOBAL).**

RETURN VALUES

If *handle* does not refer to a valid object opened by **dlopen( )**, is not the special flag **RTLD_NEXT**, or if the named symbol cannot be found within any of the objects associated with *handle*, **dlsym( )** will return **NULL**. More detailed diagnostic information will be available through **dlerror( )**.

EXAMPLES

The following example shows how one can use **dlopen( )** and **dlsym( )** to access either function or data objects. For simplicity, error checking has been omitted.

```
void    *handle;
int     *iptr, (*fptr)(int);

/* open the needed object */
handle = dlopen("/usr/home/me/libfoo.so.1", RTLD_LAZY);

/* find the address of function and data objects */
fptr = (int (*)(int))dlsym(handle, "my_function");
iptr = (int *)dlsym(handle, "my_object");

/* invoke function, passing value of integer as a parameter */
(*fptr)(*iptr);
```

SEE ALSO

ld(1), **dladdr**(3X), **dlclose**(3X), **dlerror**(3X), **dlopen**(3X)
Linker and Libraries Guide