NAME

dlinfo - information about dynamically loaded object

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <link.h>
#include <dlfcn.h>

int
dlinfo(*void * restrict handle*, *int request*, *void * restrict p*);

DESCRIPTION

The **dlinfo**() function provides information about dynamically loaded object. The action taken by **dlinfo**() and exact meaning and type of *p* argument depend on value of the *request* argument provided by caller.

The *handle* argument is either the value returned from the dlopen(3) function call or special handle RTLD_SELF. If *handle* is the value returned from dlopen(3), the information returned by the **dlinfo**() function pertains to the specified object. If handle is the special handle RTLD_SELF, the information returned pertains to the caller itself.

Possible values for the *request* argument are:

RTLD_DI_LINKMAP

Retrieve the *Link_map* (*struct link_map*) structure pointer for the specified *handle*. On successful return, the *p* argument is filled with the pointer to the *Link_map* structure (*Link_map **p*) describing a shared object specified by the *handle* argument. The *Link_map* structures are maintained as a doubly linked list by ld.so(1), in the same order as dlopen(3) and dlclose(3) are called. See *EXAMPLES*, example 1.

The *Link_map* structure is defined in <*link.h*> and has the following members:

```
caddr_t         l_base;    /* Base Address of library */
const char      *l_name;   /* Absolute Path to Library */
const void      *l_ld;     /* Pointer to .dynamic in memory */
struct link_map *l_next,   /* linked list of mapped libs */
                *l_prev;
caddr_t         l_addr;    /* Load Offset of library */
```

```
        const char      *l_refname; /* Object this one filters for */
```

l_base The base address of the object loaded into memory.

l_name
The full name of the loaded shared object.

l_ld The address of the dynamic linking information segment (PT_DYNAMIC) loaded into memory.

l_next The next *Link_map* structure on the link-map list.

l_prev The previous *Link_map* structure on the link-map list.

l_addr The load offset of the object, that is, the difference between the actual load address and the base virtual address the object was linked at.

l_refname
A name of the object this object filters for, if any. If there are more then one filtee, a name from the first DT_FILTER dynamic entry is supplied.

RTLD_DI_SERINFO
Retrieve the library search paths associated with the given *handle* argument. The *p* argument should point to *Dl_serinfo* structure buffer (*Dl_serinfo *p*). The *Dl_serinfo* structure must be initialized first with the RTLD_DI_SERINFOSIZE request.

The returned *Dl_serinfo* structure contains *dls_cnt Dl_serpath* entries. Each entry's *dlp_name* field points to the search path. The corresponding *dlp_info* field contains one of more flags indicating the origin of the path (see the LA_SER_* flags defined in the <*link.h*> header file). See *EXAMPLES*, example 2, for a usage example.

RTLD_DI_SERINFOSIZE
Initialize a *Dl_serinfo* structure for use in a RTLD_DI_SERINFO request. Both the *dls_cnt* and *dls_size* fields are returned to indicate the number of search paths applicable to the handle, and the total size of a *Dl_serinfo* buffer required to hold *dls_cnt Dl_serpath* entries and the associated search path strings. See *EXAMPLES*, example 2, for a usage example.

RTLD_DI_ORIGIN
Retrieve the origin of the dynamic object associated with the handle. On successful return, *p* argument is filled with the *char* pointer (*char *p*).

RETURN VALUES

The **dlinfo**() function returns 0 on success, or -1 if an error occurred. Whenever an error has been detected, a message detailing it can be retrieved via a call to dlerror(3).

EXAMPLES

Example 1: Using **dlinfo**() to retrieve *Link_map* structure.

The following example shows how dynamic library can detect the list of shared libraries loaded after caller’s one. For simplicity, error checking has been omitted.

```
Link_map *map;

dlinfo(RTLD_SELF, RTLD_DI_LINKMAP, &map);

while (map != NULL) {
        printf("%p: %s\n", map->l_addr, map->l_name);
        map = map->l_next;
}
```

Example 2: Using **dlinfo**() to retrieve the library search paths.

The following example shows how a dynamic object can inspect the library search paths that would be used to locate a simple filename with dlopen(3). For simplicity, error checking has been omitted.

```
Dl_serinfo       _info, *info = &_info;
Dl_serpath      *path;
unsigned int     cnt;

/* determine search path count and required buffer size */
dlinfo(RTLD_SELF, RTLD_DI_SERINFOSIZE, (void *)info);

/* allocate new buffer and initialize */
info = malloc(_info.dls_size);
info->dls_size = _info.dls_size;
info->dls_cnt = _info.dls_cnt;

/* obtain sarch path information */
dlinfo(RTLD_SELF, RTLD_DI_SERINFO, (void *)info);

path = &info->dls_serpath[0];
```

```
      for (cnt = 1; cnt <= info->dls_cnt; cnt++, path++) {
              (void) printf("%2d: %s\n", cnt, path->dls_name);
      }
```

SEE ALSO

rtld(1), dladdr(3), dlopen(3), dlsym(3)

HISTORY

The **dlinfo**() function first appeared in the Solaris operating system. In FreeBSD, it first appeared in FreeBSD 4.8.

AUTHORS

The FreeBSD implementation of the **dlinfo**() function was originally written by Alexey Zelkin <*phantom@FreeBSD.org*> and later extended and improved by Alexander Kabaev <*kan@FreeBSD.org*>.

The manual page for this function was written by Alexey Zelkin <*phantom@FreeBSD.org*>.