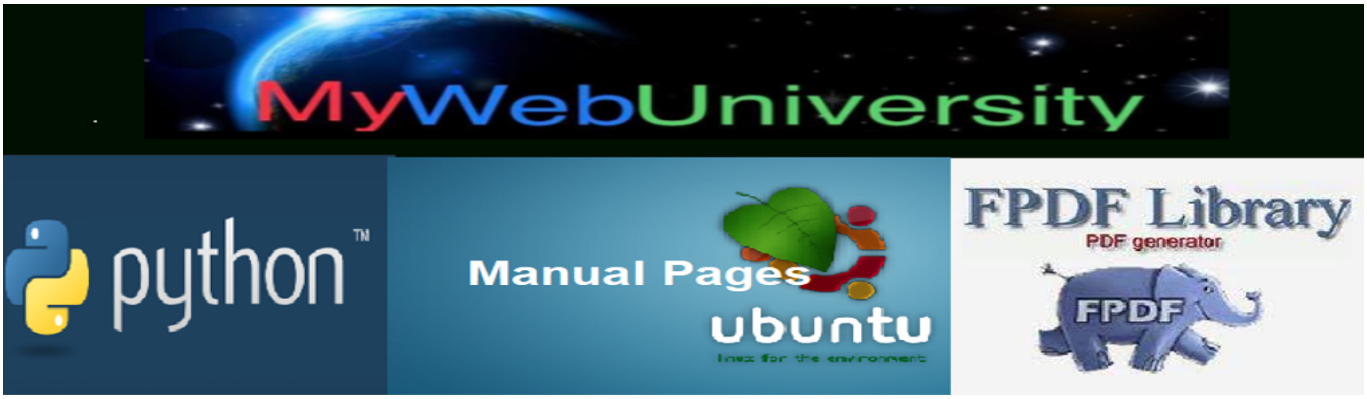

Rocky Enterprise Linux 9.2 Manual Pages on command 'dlinfo.3'

$ man dlinfo.3

DLINFO(3) Linux Programmer's Manual DLINFO(3)

NAME

dlinfo - obtain information about a dynamically loaded object

SYNOPSIS

```
#define _GNU_SOURCE
#include <link.h>
#include <dlfcn.h>
int dlinfo(void *handle, int request, void *info);
```

Link with -ldl.

DESCRIPTION

The dlinfo() function obtains information about the dynamically loaded object referred to by handle (typically obtained by an earlier call to dlopen(3) or dlmopen(3)). The request argument specifies which information is to be returned. The info argument is a pointer to a buffer used to store information returned by the call; the type of this argument depends on request.

The following values are supported for request (with the corresponding type for info shown in parentheses):

RTLD_DI_LMID (Lmid_t *)

Obtain the ID of the link-map list (namespace) in which handle is loaded.

RTLD_DI_LINKMAP (struct link_map **)

Obtain a pointer to the link_map structure corresponding to handle. The info argu? ment points to a pointer to a link_map structure, defined in <link.h> as:

```
struct link_map {
```

```
        ElfW(Addr) l_addr;  /* Difference between the
                               address in the ELF file and
                               the address in memory */
        char      *l_name;  /* Absolute pathname where
                               object was found */
        ElfW(Dyn) *l_ld;    /* Dynamic section of the
                               shared object */
        struct link_map *l_next, *l_prev;
                            /* Chain of loaded objects */
        /* Plus additional fields private to the
           implementation */
    };
```

RTLD_DI_ORIGIN (char *)

Copy the pathname of the origin of the shared object corresponding to handle to the location pointed to by info.

RTLD_DI_SERINFO (Dl_serinfo *)

Obtain the library search paths for the shared object referred to by handle. The info argument is a pointer to a Dl_serinfo that contains the search paths. Because the number of search paths may vary, the size of the structure pointed to by info can vary. The RTLD_DI_SERINFOSIZE request described below allows applications to size the buffer suitably. The caller must perform the following steps:

1. Use a RTLD_DI_SERINFOSIZE request to populate a Dl_serinfo structure with the size (dls_size) of the structure needed for the subsequent RTLD_DI_SERINFO re? quest.
2. Allocate a Dl_serinfo buffer of the correct size (dls_size).
3. Use a further RTLD_DI_SERINFOSIZE request to populate the dls_size and dls_cnt fields of the buffer allocated in the previous step.
4. Use a RTLD_DI_SERINFO to obtain the library search paths.

The Dl_serinfo structure is defined as follows:

```
    typedef struct {
        size_t dls_size;          /* Size in bytes of
                                     the whole buffer */
        unsigned int dls_cnt;     /* Number of elements
```

```
                                in 'dls_serpath' */
        Dl_serpath dls_serpath[1]; /* Actually longer,
                                'dls_cnt' elements */
    } Dl_serinfo;
```

Each of the dls_serpath elements in the above structure is a structure of the fol? lowing form:

```
    typedef struct {
        char *dls_name;          /* Name of library search
                                path directory */
        unsigned int dls_flags;  /* Indicates where this
                                directory came from */
    } Dl_serpath;
```

The dls_flags field is currently unused, and always contains zero.

RTLD_DI_SERINFOSIZE (Dl_serinfo *)

Populate the dls_size and dls_cnt fields of the Dl_serinfo structure pointed to by info with values suitable for allocating a buffer for use in a subsequent RTLD_DI_SERINFO request.

RTLD_DI_TLS_MODID (size_t *, since glibc 2.4)

Obtain the module ID of this shared object's TLS (thread-local storage) segment, as used in TLS relocations. If this object does not define a TLS segment, zero is placed in *info.

RTLD_DI_TLS_DATA (void **, since glibc 2.4)

Obtain a pointer to the calling thread's TLS block corresponding to this shared ob? ject's TLS segment. If this object does not define a PT_TLS segment, or if the calling thread has not allocated a block for it, NULL is placed in *info.

RETURN VALUE

On success, dlinfo() returns 0. On failure, it returns -1; the cause of the error can be diagnosed using dlerror(3).

VERSIONS

dlinfo() first appeared in glibc 2.3.3.

ATTRIBUTES

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

?Interface ? Attribute ? Value ?

??????????????????????????????????????

?dlinfo() ? Thread safety ? MT-Safe ?

??????????????????????????????????????

CONFORMING TO

This function is a nonstandard GNU extension.

NOTES

This function derives from the Solaris function of the same name and also appears on some other systems. The sets of requests supported by the various implementations overlaps only partially.

EXAMPLES

The program below opens a shared objects using dlopen(3) and then uses the RTLD_DI_SERIN? FOSIZE and RTLD_DI_SERINFO requests to obtain the library search path list for the li? brary. Here is an example of what we might see when running the program:

```
$ ./a.out /lib64/libm.so.6
dls_serpath[0].dls_name = /lib64
dls_serpath[1].dls_name = /usr/lib64
```

Program source

```
#define _GNU_SOURCE
#include <dlfcn.h>
#include <link.h>
#include <stdio.h>
#include <stdlib.h>
int
main(int argc, char *argv[])
{
    void *handle;
    Dl_serinfo serinfo;
    Dl_serinfo *sip;
    if (argc != 2) {
        fprintf(stderr, "Usage: %s <libpath>\n", argv[0]);
        exit(EXIT_FAILURE);
    }
```

```
/* Obtain a handle for shared object specified on command line */
handle = dlopen(argv[1], RTLD_NOW);
if (handle == NULL) {
    fprintf(stderr, "dlopen() failed: %s\n", dlerror());
    exit(EXIT_FAILURE);
}
/* Discover the size of the buffer that we must pass to
   RTLD_DI_SERINFO */
if (dlinfo(handle, RTLD_DI_SERINFOSIZE, &serinfo) == -1) {
    fprintf(stderr, "RTLD_DI_SERINFOSIZE failed: %s\n", dlerror());
    exit(EXIT_FAILURE);
}
/* Allocate the buffer for use with RTLD_DI_SERINFO */
sip = malloc(serinfo.dls_size);
if (sip == NULL) {
    perror("malloc");
    exit(EXIT_FAILURE);
}
/* Initialize the 'dls_size' and 'dls_cnt' fields in the newly
   allocated buffer */
if (dlinfo(handle, RTLD_DI_SERINFOSIZE, sip) == -1) {
    fprintf(stderr, "RTLD_DI_SERINFOSIZE failed: %s\n", dlerror());
    exit(EXIT_FAILURE);
}
/* Fetch and print library search list */
if (dlinfo(handle, RTLD_DI_SERINFO, sip) == -1) {
    fprintf(stderr, "RTLD_DI_SERINFO failed: %s\n", dlerror());
    exit(EXIT_FAILURE);
}
for (int j = 0; j < serinfo.dls_cnt; j++)
    printf("dls_serpath[%d].dls_name = %s\n",
            j, sip->dls_serpath[j].dls_name);
exit(EXIT_SUCCESS);
```

```
}
```

SEE ALSO

dl_iterate_phdr(3), dladdr(3), dlerror(3), dlopen(3), dlsym(3), ld.so(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/.