NAME

dl_iterate_phdr - walk through list of shared objects

SYNOPSIS

#define _GNU_SOURCE /* See feature_test_macros(7) */
#include <link.h>

int dl_iterate_phdr(
int (callback*) (struct dl_phdr_info *_info_,**
size_t _size_**, void ***_data_**),**
void ***_data_);**

DESCRIPTION

The **dl_iterate_phdr**() function allows an application to inquire at run time to find out which shared objects it has loaded.

The **dl_iterate_phdr**() function walks through the list of an application's shared objects and calls the function *callback* once for each object, until either all shared objects have been processed or *callback* returns a nonzero value.

Each call to *callback* receives three arguments: *info*, which is a pointer to a structure containing information about the shared object; *size*, which is the size of the structure pointed to by *info*; and *data*, which is a copy of whatever value was passed by the calling program as the second argument (also named *data*) in the call to **dl_iterate_phdr**().

The *info* argument is a structure of the following type:

```
struct dl_phdr_info {
ElfW(Addr) dlpi_addr; /* Base address of object */
const char *dlpi_name; /* (Null-terminated) name of
object */
const ElfW(Phdr) *dlpi_phdr; /* Pointer to array of
ELF program headers
for this object */
ElfW(Half) dlpi_phnum; /* # of items in dlpi_phdr */
};
```

(The *ElfW*() macro definition turns its argument into the name of an ELF data type suitable for the hardware architecture. For example, on a 32-bit platform, ElfW(Addr) yields the data type name Elf32_Addr. Further information on these types can be found in the <*elf.h*> and <*link.h*> header files.)

The *dlpi_addr* field indicates the base address of the shared object (i.e., the difference between the virtual memory address of the shared object and the offset of that object in the file from which it was loaded). The *dlpi_name* field is a null-terminated string giving the pathname from which the shared object was loaded.

To understand the meaning of the *dlpi_phdr* and *dlpi_phnum* fields, we need to be aware that an ELF shared object consists of a number of segments, each of which has a corresponding program header describing the segment. The*dlpi_phdr* field is a poin ter to an array of the program headers for this shared object. The *dlpi_phnum* field indicates the size of this array.

These program headers are structures of the following form:

```
typedef struct {
Elf32_Word p_type; /* Segment type */
Elf32_Off p_offset; /* Segment file offset */
Elf32_Addr p_vaddr; /* Segment virtual address */
Elf32_Addr p_paddr; /* Segment physical address */
Elf32_Word p_filesz; /* Segment size in file */
```

```
    Elf32_Word p_memsz; /* Segment size in memory */
    Elf32_Word p_flags; /* Segment flags */
    Elf32_Word p_align; /* Segment alignment */
    } Elf32_Phdr;
```

Note that we can calculate the location of a particular program header, *x*, in virtual memory using the formula:

```
addr == info->dlpi_addr + info->dlpi_phdr[x].p_vaddr;
```

RETURN VALUE

The **dl_iterate_phdr**() function returns whatever value was returned by the last call to *callback*.

VERSIONS

dl_iterate_phdr() has been supported in glibc since version 2.2.4.

CONFORMING TO

The **dl_iterate_phdr**() function is Linux-specific and should be avoided in portable applications.

EXAMPLE

The following program displays a list of pathnames of the shared objects it has loaded. For each shared object, the program lists the virtual addresses at which the object's ELF segments are loaded.

```
#define _GNU_SOURCE
#include <link.h>
#include <stdlib.h>
#include <stdio.h>

static int
callback(struct dl_phdr_info *info, size_t size, void *data)
{
int j;

printf(name=%s (%d segments)n, info->dlpi_name,
info->dlpi_phnum);

for (j = 0; j < info->dlpi_phnum; j++)
printf(tt header %2d: address=%10pn, j,
(void *) (info->dlpi_addr + info->dlpi_phdr[j].p_vaddr));
return 0;
}

int
main(int argc, char *argv[])
{
dl_iterate_phdr(callback, NULL);

exit(EXIT_SUCCESS);
}
```

SEE ALSO

ldd(1), objdump(1), readelf(1), dlopen(3), elf(5), ld.so(8)

Executable and Linking Format Specification, available at various locations online.

COLOPHON

This page is part of release 3.74 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 http://www.kernel.org/doc/man-pages/.