NAME

elf32_xlate, **elf64_xlate**, **gelf_xlate** - translate data between files and memory

LIBRARY

ELF Access Library (libelf, -lelf)

SYNOPSIS

#include <libelf.h>

*Elf_Data **
elf32_xlatetof(*Elf_Data *dst, Elf_Data *src, unsigned int file_encoding*);

*Elf_Data **
elf32_xlatetom(*Elf_Data *dst, Elf_Data *src, unsigned int file_encoding*);

*Elf_Data **
elf64_xlatetof(*Elf_Data *dst, Elf_Data *src, unsigned int file_encoding*);

*Elf_Data **
elf64_xlatetom(*Elf_Data *dst, Elf_Data *src, unsigned int file_encoding*);

#include <gelf.h>

*Elf_Data **
gelf_xlatetof(*Elf *elf, Elf_Data *dst, Elf_Data *src, unsigned int file_encoding*);

*Elf_Data **
gelf_xlatetom(*Elf *elf, Elf_Data *dst, Elf_Data *src, unsigned int file_encoding*);

DESCRIPTION

These functions translate between the file and memory representations of ELF data structures. The in-memory representation of an ELF data structure would conform to the byte ordering and data alignment restrictions dictated by the host processor. As described in elf(3), the file representation of this data structure could use a different byte ordering from that of the host, or could use a different layout within the file.

Functions **elf32_xlatetom**(), **elf64_xlatetom**(), and **gelf_xlatetom**() translate data from file representations to native, in-memory representations. Functions **elf32_xlatetof**(), **elf64_xlatetof**(), and **gelf_xlatetof**() translate data from in-memory representations to file representations.

Argument *src* denotes an *Elf_Data* descriptor describing the source to be translated. The following elements of the descriptor need to be set before invoking these functions:

d_buf Set to a valid pointer value denoting the beginning of the data area to be translated.

d_size Set to the total size in bytes of the source data area to be translated.

d_type Set to the type of the source data being translated. This value is one of the values defined in the *Elf_Type* enumeration. The *Elf_Type* enumeration is described in elf(3).

d_version Set to the version number of the ELF data structures being translated. Currently only version EV_CURRENT is supported.

Argument *dst* describes the destination buffer. The following elements of the *Elf_Data* descriptor need to be set before invoking these functions:

d_buf Set to a valid pointer value that denotes the start of the destination buffer that will hold translated data. This value may be the same as that of the source buffer, in which case an in-place conversion will be attempted.

d_size Set to the size of the destination buffer in bytes. This value will be modified if the function call succeeds.

d_version Set to the desired version number of the destination. Currently only version EV_CURRENT is supported.

These translations routines allow the source and destination buffers to coincide, in which case an in-place translation will be done if the destination is large enough to hold the translated data. Other kinds of overlap between the source and destination buffers are not permitted.

On successful completion of the translation request the following fields of the *dst* descriptor would be modified:

d_size Set to the size in bytes of the translated data.

d_type Set to the *d_type* value of the source data descriptor.

Argument *file_encoding* specifies the encoding in which the file objects are represented. It must be one of:

ELFDATANONEFile objects use the library's native byte ordering.

ELFDATA2LSBFile objects use a little-endian ordering.

ELFDATA2MSBFile objects use a big-endian ordering.

The functions **gelf_xlatetof**() and **gelf_xlatetom**() select the appropriate translation scheme based on the properties of argument *elf*.

RETURN VALUES

These functions return argument *dst* if successful, or NULL in case of an error.

EXAMPLES

To translate a *GElf_Rel* structure to its LSB file representation use:

```
Elf_Data dst, src;
GElf_Rel rel;
Elf *e;

e = ...; /* See elf_begin(3). */

/* Set up the 'src' descriptor. */
memset(&src, 0, sizeof src);
src.d_buf = &rel;
src.d_size = sizeof(rel);
src.d_type = ELF_T_REL;
src.d_version = EV_CURRENT;

/* Set up the 'dst' descriptor. */
memset(&dst, 0, sizeof dst);
dst.d_buf = filebuf;
dst.d_size = gelf_fsize(e, ELF_T_REL, 1, EV_CURRENT);
dst.d_version = EV_CURRENT;

if (gelf_xlatetof(e, &dst, &src, ELFDATA2LSB) == NULL) {
        printf("error: %s", elf_errmsg(0));
}
```

ERRORS

These functions may fail with the following errors:

[ELF_E_ARGUMENT]
One of arguments *src*, *dst* or *elf* was NULL.

[ELF_E_ARGUMENT]
Arguments *src* and *dst* were equal.

[ELF_E_ARGUMENT]
The desired encoding parameter was not one of ELFDATANONE, ELFDATA2LSB or ELFDATA2MSB.

[ELF_E_ARGUMENT]
The *d_type* field of argument *src* specified an unsupported type.

[ELF_E_DATA] The *src* argument specified a buffer size that was not an integral multiple of its underlying type.

[ELF_E_DATA] The *dst* argument specified a buffer size that was too small.

[ELF_E_DATA] Argument *dst* specified a destination buffer that overlaps with the source buffer.

[ELF_E_DATA] The destination buffer for a conversion to memory had an alignment inappropriate for the underlying ELF type.

[ELF_E_DATA] The source buffer for a conversion to file had an alignment inappropriate for the underlying ELF type.

[ELF_E_UNIMPL] The version numbers for arguments *dst* and *src* were not identical.

[ELF_E_UNIMPL] The argument *src* requested conversion for a type which is not currently supported.

[ELF_E_VERSION] Argument *src* specified an unsupported version number.

SEE ALSO

elf(3), elf_getdata(3), gelf(3)