NAME

gelf - class-independent API for ELF manipulation

LIBRARY

ELF Access Library (libelf, -lelf)

SYNOPSIS

#include <gelf.h>

DESCRIPTION

This manual page describes a class independent API for manipulating ELF objects. This API allows an application to operate on ELF descriptors without needing to the know the ELF class of the descriptor.

The GElf API may be used alongside the ELF API without restriction.

GElf Data Structures

The GElf API defines the following class-independent data structures:

GElf_Addr A representation of ELF addresses.

GElf_Chdr A class-independent representation of an ELF Compression Header.

GElf_Dyn A class-independent representation of ELF **.dynamic** section entries.

GElf_Ehdr A class-independent representation of an ELF Executable Header.

GElf_Half An unsigned 16 bit quantity.

GElf_Off A class-independent representation of a ELF offset.

GElf_Phdr A class-independent representation of an ELF Program Header Table entry.

GElf_Rel A class-independent representation of an ELF relocation entry.

GElf_Rela A class-independent representation of an ELF relocation entry with addend.

GElf_Shdr A class-independent representation of an ELF Section Header Table entry.

GElf_Sword A signed 32 bit quantity.

GElf_Sxword A signed 64 bit quantity.

GElf_Sym A class-independent representation of an ELF symbol table entry.

GElf_Word An unsigned 32 bit quantity.

GElf_Xword An unsigned 64 bit quantity.

These data structures are sized to be compatible with the corresponding 64 bit ELF structures, and have the same internal structure as their 64 bit class-dependent counterparts. Class-dependent ELF structures are described in elf(5).

GElf Programming Model

GElf functions always return a *copy* of the underlying (class-dependent) ELF data structure. The programming model with GElf is as follows:

1. An application will retrieve data from an ELF descriptor using a **gelf_get_***() function. This will copy out data into a private *GElf_** data structure.

2. The application will work with its private copy of the GElf structure.

3. Once done, the application copies the new values back to the underlying ELF data structure using the **gelf_update_***() functions.

4. The application will then use the **elf_flag***() APIs to indicate to the ELF library that an ELF data structure is dirty.

When updating an underlying 32 bit ELF data structure, the GElf routines will signal an error if a GElf value is out of range for the underlying ELF data type.

Namespace use

The GElf interface uses the following symbols:

GElf_*
: Class-independent data types.

gelf_*
: For functions defined in the API set.

GElf Programming APIs

This section provides an overview of the GElf programming APIs. Further information is provided in the manual page of each function listed here.

Allocating ELF Data Structures

gelf_newehdr()
: Allocate a new ELF Executable Header.

gelf_newphdr()
: Allocate a new ELF Program Header Table.

Data Translation

gelf_xlatetof()
: Translate the native representation of an ELF data structure to its file representation.

gelf_xlatetom()
: Translate from the file representation of an ELF data structure to a native representation.

Retrieving ELF Data

gelf_getchdr()
: Retrieve an ELF Compression Header from the underlying ELF descriptor.

gelf_getdyn()
: Retrieve an ELF **.dynamic** table entry.

gelf_getehdr()
: Retrieve an ELF Executable Header from the underlying ELF descriptor.

gelf_getphdr()
: Retrieve an ELF Program Header Table entry from the underlying ELF descriptor.

gelf_getrel()
: Retrieve an ELF relocation entry.

gelf_getrela()
: Retrieve an ELF relocation entry with addend.

gelf_getshdr()
: Retrieve an ELF Section Header Table entry from the underlying ELF descriptor.

gelf_getsym()
: Retrieve an ELF symbol table entry.

Queries

gelf_checksum()
: Retrieves the ELF checksum for an ELF descriptor.

gelf_fsize()
: Retrieves the size of the file representation of an ELF type.

gelf_getclass()
: Retrieves the ELF class of an ELF descriptor.

Updating ELF Data

gelf_update_dyn()
: Copy back an ELF **.dynamic** Table entry.

gelf_update_phdr()
: Copy back an ELF Program Header Table entry.

gelf_update_rel() Copy back an ELF relocation entry.

gelf_update_rela()
: Copy back an ELF relocation with addend entry.

gelf_update_shdr()
: Copy back an ELF Section Header Table entry.

gelf_update_sym()
: Copy back an ELF symbol table entry.

SEE ALSO

elf(3), elf(5)

HISTORY

The **gelf** API first appeared in AT&T System V Release 4 UNIX. This implementation of the API first appeared in FreeBSD 7.0.

AUTHORS

The GElf API was implemented by Joseph Koshy <*jkoshy@FreeBSD.org*>.