NAME

elf_begin - open an ELF file or ar(1) archive

LIBRARY

ELF Access Library (libelf, -lelf)

SYNOPSIS

#include <libelf.h>

*Elf **
elf_begin(*int fd*, *Elf_Cmd cmd*, *Elf *elf*);

DESCRIPTION

Function **elf_begin**() is used to open ELF files and ar(1) archives for further processing by other APIs in the elf(3) library. It is also used to access individual ELF members of an ar(1) archive in combination with the elf_next(3) and elf_rand(3) APIs.

Argument *fd* is an open file descriptor returned from an open(2) system call. Function **elf_begin**() uses argument *fd* for reading or writing depending on the value of argument *cmd*. Argument *elf* is primarily used for iterating through archives.

The argument *cmd* can have the following values:

ELF_C_NULL Causes **elf_begin**() to return NULL. Arguments *fd* and *elf* are ignored, and no additional error is signalled.

ELF_C_READ This value is to be when the application wishes to examine (but not modify) the contents of the file specified by the arguments *fd* and *elf*. It can be used for both ar(1) archives and for ELF objects.

If argument *elf* is NULL, the library will allocate a new ELF descriptor for the file being processed. The argument *fd* should have been opened for reading.

If argument *elf* is not NULL, and references a regular ELF file previously opened with **elf_begin**(), then the activation count for the descriptor referenced by argument *elf* is incremented. The value in argument *fd* should match that used to open the descriptor argument *elf*.

If argument *elf* is not NULL, and references a descriptor for an ar(1) archive opened earlier with **elf_begin**(), a descriptor for an element in the archive is returned as described in the section *Processing ar(1) archives* below. The value for argument *fd* should match that used to open the archive earlier.

If argument *elf* is not NULL, and references an ar(1) archive opened earlier with **elf_memory**(), then the value of the argument *fd* is ignored.

ELF_C_RDWR This command is used to prepare an ELF file for reading and writing. This command is not supported for ar(1) archives.

Argument *fd* should have been opened for reading and writing. If argument *elf* is NULL, the library will allocate a new ELF descriptor for the file being processed. If the argument *elf* is non-null, it should point to a descriptor previously allocated with **elf_begin**() with the same values for arguments *fd* and *cmd*; in this case the library will increment the activation count for descriptor *elf* and return the same descriptor.

Changes to the in-memory image of the ELF file may be written back to disk using the elf_update(3) function.

ELF_C_WRITE This command is used when the application wishes to create a new ELF file. Argument *fd* should have been opened for writing. Argument *elf* is ignored, and the previous contents of file referenced by argument *fd* are overwritten.

Processing ar(1) archives

An ar(1) archive may be opened in read mode (with argument *cmd* set to ELF_C_READ) using **elf_begin**() or **elf_memory**(). The returned ELF descriptor can be passed into to subsequent calls to **elf_begin**() to access individual members of the archive.

Random access within an opened archive is possible using the elf_next(3) and elf_rand(3) functions.

The symbol table of the archive may be retrieved using elf_getarsym(3).

RETURN VALUES

The function returns a pointer to a ELF descriptor if successful, or NULL if an error occurred.

EXAMPLES

To iterate through the members of an ar(1) archive, use:

```
Elf_Cmd c;
Elf *ar_e, *elf_e;
...
```

```
      c = ELF_C_READ;
      if ((ar_e = elf_begin(fd, c, (Elf *) 0)) == 0) {
              ... handle error in opening the archive ...
      }
      while ((elf_e = elf_begin(fd, c, ar_e)) != 0) {
              ... process member referenced by elf_e here ...
              c = elf_next(elf_e);
              elf_end(elf_e);
      }
```

To create a new ELF file, use:

```
      int fd;
      Elf *e;
      ...
      if ((fd = open("filename", O_RDWR|O_TRUNC|O_CREAT, 0666)) < 0) {
              ... handle the error from open(2) ...
      }
      if ((e = elf_begin(fd, ELF_C_WRITE, (Elf *) 0)) == 0) {
              ... handle the error from elf_begin() ...
      }
      ... create the ELF image using other elf(3) APIs ...
      elf_update(e, ELF_C_WRITE);
      elf_end(e);
```

ERRORS

Function **elf_begin**() can fail with the following errors:

[ELF_E_ARCHIVE] The archive denoted by argument *elf* could not be parsed.

[ELF_E_ARGUMENT]
The value in argument *cmd* was unrecognized.

[ELF_E_ARGUMENT]
A non-null value for argument *elf* was specified when *cmd* was set to ELF_C_RDWR.

[ELF_E_ARGUMENT]
The value of argument *fd* differs from the one the ELF descriptor *elf* was created with.

[ELF_E_ARGUMENT]	Argument *cmd* differs from the value specified when ELF descriptor *elf* was created.
[ELF_E_ARGUMENT]	An ar(1) archive was opened with *cmd* set to ELF_C_RDWR.
[ELF_E_ARGUMENT]	The file referenced by argument *fd* was empty.
[ELF_E_ARGUMENT]	The underlying file for argument *fd* was of an unsupported type.
[ELF_E_IO]	The file descriptor in argument *fd* was invalid.
[ELF_E_IO]	The file descriptor in argument *fd* could not be read or written to.
[ELF_E_RESOURCE]	An out of memory condition was encountered.
[ELF_E_SEQUENCE]	Function **elf_begin**() was called before a working version was established with elf_version(3).
[ELF_E_VERSION]	The ELF object referenced by argument *fd* was of an unsupported ELF version.

SEE ALSO

elf(3), elf_end(3), elf_errno(3), elf_memory(3), elf_next(3), elf_rand(3), elf_update(3), gelf(3)