## NAME

**ld-elf.so.1**, **ld.so**, **rtld** - run-time link-editor

## DESCRIPTION

The **ld-elf.so.1** utility is a self-contained shared object providing run-time support for loading and link-editing shared objects into a process' address space. It is also commonly known as the dynamic linker. It uses the data structures contained within dynamically linked programs to determine which shared libraries are needed and loads them using the mmap(2) system call.

After all shared libraries have been successfully loaded, **ld-elf.so.1** proceeds to resolve external references from both the main program and all objects loaded. A mechanism is provided for initialization routines to be called on a per-object basis, giving a shared object an opportunity to perform any extra set-up before execution of the program proper begins. This is useful for C++ libraries that contain static constructors.

When resolving dependencies for the loaded objects, **ld-elf.so.1** translates dynamic token strings in rpath and soname. If the **-z origin** option of the static linker was set when linking the binary, the token expansion is performed at the object load time, see ld(1). The following strings are recognized now:

| | |
|---|---|
| *$ORIGIN* | Translated to the full path of the loaded object. |
| *$OSNAME* | Translated to the name of the operating system implementation. |
| *$OSREL* | Translated to the release level of the operating system. |
| *$PLATFORM* | Translated to the machine hardware platform. |
| *$LIB* | Translated to the system library path component on the platform. It is *lib* for native binaries, and typically *lib32* for compat32 binaries. Other translations might exist for other ABIs supported on the platform. |

The **ld-elf.so.1** utility itself is loaded by the kernel together with any dynamically-linked program that is to be executed. The kernel transfers control to the dynamic linker. After the dynamic linker has finished loading, relocating, and initializing the program and its required shared objects, it transfers control to the entry point of the program. The following search order is used to locate required shared objects:

1. DT_RPATH of the referencing object unless that object also contains a DT_RUNPATH tag
2. DT_RPATH of the program unless the referencing object contains a DT_RUNPATH tag
3. Path indicated by LD_LIBRARY_PATH environment variable

4. DT_RUNPATH of the referencing object
5. Hints file produced by the ldconfig(8) utility
6. The */lib* and */usr/lib* directories, unless the referencing object was linked using the "**-z** *nodefaultlib*" option

The **ld-elf.so.1** utility recognizes a number of environment variables that can be used to modify its behaviour. On 64-bit architectures, the linker for 32-bit objects recognizes all the environment variables listed below, but is being prefixed with LD_32_, for example: LD_32_TRACE_LOADED_OBJECTS. If the activated image is setuid or setgid, the variables are ignored.

LD_DUMP_REL_POST
: If set, **ld-elf.so.1** will print a table containing all relocations after symbol binding and relocation.

LD_DUMP_REL_PRE
: If set, **ld-elf.so.1** will print a table containing all relocations before symbol binding and relocation.

LD_DYNAMIC_WEAK
: If set, use the ELF standard-compliant symbol lookup behavior: resolve to the first found symbol definition.

  By default, FreeBSD provides the non-standard symbol lookup behavior: when a weak symbol definition is found, remember the definition and keep searching in the remaining shared objects for a non-weak definition. If found, the non-weak definition is preferred, otherwise the remembered weak definition is returned.

  Symbols exported by dynamic linker itself (see dlfcn(3)) are always resolved using FreeBSD rules regardless of the presence of the variable. This variable is unset for set-user-ID and set-group-ID programs.

LD_LIBMAP
: A library replacement list in the same format as libmap.conf(5). For convenience, the characters ‘=’ and ‘,’ can be used instead of a space and a newline. This variable is parsed after libmap.conf(5), and will override its entries. This variable is unset for set-user-ID and set-group-ID programs.

LD_LIBMAP_DISABLE
: If set, disables the use of libmap.conf(5) and LD_LIBMAP. This variable is unset for set-user-ID and set-group-ID programs.

LD_ELF_HINTS_PATH
: This variable will override the default location of "hints" file. This variable is unset for set-user-ID and set-group-ID programs.

LD_LIBRARY_PATH
: A colon separated list of directories, overriding the default search path for shared libraries. This variable is unset for set-user-ID and set-group-ID programs.

LD_LIBRARY_PATH_RPATH
: If the variable is specified and has a value starting with any of ’y’, ’Y’ or ’1’ symbols, the path specified by LD_LIBRARY_PATH variable is allowed to override the path from DT_RPATH for binaries which does not contain DT_RUNPATH tag. For such binaries, when the variable LD_LIBRARY_PATH_RPATH is set, "**-z** *nodefaultlib*" link-time option is ignored as well.

LD_PRELOAD
: A list of shared libraries, separated by colons and/or white space, to be linked in before any other shared libraries. If the directory is not specified then the directories specified by LD_LIBRARY_PATH will be searched first followed by the set of built-in standard directories. This variable is unset for set-user-ID and set-group-ID programs.

LD_PRELOAD_FDS
: A colon separated list of file descriptor numbers for libraries. This is intended for preloading libraries in which we already have a file descriptor. This may optimize the process of loading libraries because we do not have to look for them in directories. It may also be useful in a capability base system where we do not have access to global namespaces such as the filesystem.

LD_LIBRARY_PATH_FDS
: A colon separated list of file descriptor numbers for library directories. This is intended for use within capsicum(4) sandboxes, when global namespaces such as the filesystem are unavailable. It is consulted just after LD_LIBRARY_PATH. This variable is unset for set-user-ID and set-group-ID programs.

LD_BIND_NOT
: When set to a nonempty string, prevents modifications of the PLT slots when doing bindings. As result, each call of the PLT-resolved function is resolved. In combination with debug output, this provides complete account of all bind actions at runtime. This variable is unset for set-user-ID and set-group-ID programs.

LD_BIND_NOW
: When set to a nonempty string, causes **ld-elf.so.1** to relocate all external function calls before starting execution of the program. Normally, function calls are bound lazily, at the first call of each function. LD_BIND_NOW

increases the start-up time of a program, but it avoids run-time surprises caused by unexpectedly undefined functions.

LD_TRACE_LOADED_OBJECTS
When set to a nonempty string, causes **ld-elf.so.1** to exit after loading the shared objects and printing a summary which includes the absolute pathnames of all objects, to standard output.

LD_TRACE_LOADED_OBJECTS_ALL
When set to a nonempty string, causes **ld-elf.so.1** to expand the summary to indicate which objects caused each object to be loaded.

LD_TRACE_LOADED_OBJECTS_FMT1

LD_TRACE_LOADED_OBJECTS_FMT2
When set, these variables are interpreted as format strings a la printf(3) to customize the trace output and are used by ldd(1)'s **-f** option and allows ldd(1) to be operated as a filter more conveniently. If the dependency name starts with string *lib*, LD_TRACE_LOADED_OBJECTS_FMT1 is used, otherwise LD_TRACE_LOADED_OBJECTS_FMT2 is used. The following conversions can be used:

| | |
|---|---|
| %a | The main program's name (also known as "__progname"). |
| %A | The value of the environment variable LD_TRACE_LOADED_OBJECTS_PROGNAME. Typically used to print both the names of programs and shared libraries being inspected using ldd(1). |
| %o | The library name. |
| %p | The full pathname as determined by **rtld**'s library search rules. |
| %x | The library's load address. |

Additionally, '\n' and '\t' are recognized and have their usual meaning.

LD_UTRACE
If set, **ld-elf.so.1** will log events such as the loading and unloading of shared objects via utrace(2).

| | |
|---|---|
| LD_LOADFLTR | If set, **ld-elf.so.1** will process the filtee dependencies of the loaded objects immediately, instead of postponing it until required. Normally, the filtees are opened at the time of the first symbol resolution from the filter object. |
| LD_SHOW_AUXV | If set, causes **ld-elf.so.1** to dump content of the aux vector to standard output, before passing control to any user code. |

## DIRECT EXECUTION MODE

**ld-elf.so.1** is typically used implicitly, loaded by the kernel as requested by the PT_INTERP program header of the executed binary. FreeBSD also supports a direct execution mode for the dynamic linker. In this mode, the user explicitly executes **ld-elf.so.1** and provides the path of the program to be linked and executed as an argument. This mode allows use of a non-standard dynamic linker for a program activation without changing the binary or without changing the installed dynamic linker. Execution options may be specified.

The syntax of the direct invocation is

*/libexec/ld-elf.so.1* [**-b** *exe*] [**-d**] [**-f** *fd*] [**-p**] [**-u**] [**-v**] [**--**] *image_path* [*image arguments*]

The options are:

**-b** *exe* Use the executable *exe* instead of *image_path* for activation. If this option is specified, *image_path* is only used to provide the *argv[0]* value to the program.

**-d** Turn off the emulation of the binary execute permission.

**-f** *fd* File descriptor *fd* references the binary to be activated by **ld-elf.so.1**. It must already be opened in the process when executing **ld-elf.so.1**. If this option is specified, *image_path* is only used to provide the *argv[0]* value to the program.

**-p** If the *image_path* argument specifies a name which does not contain a slash "/" character, **ld-elf.so.1** uses the search path provided by the environment variable PATH to find the binary to execute.

**-u** Ignore all LD_ environment variables that otherwise affect the dynamic linker behavior.

**-v** Display information about this run-time linker binary, then exit.

**--** Ends the **ld-elf.so.1** options. The argument following -- is interpreted as the path of the binary to execute.

In the direct execution mode, **ld-elf.so.1** emulates verification of the binary execute permission for the current user. This is done to avoid breaking user expectations in naively restricted execution environments. The verification only uses Unix DACs, ignores ACLs, and is naturally prone to race conditions. Environments which rely on such restrictions are weak and breakable on their own. It can be turned off with the **-d** option.

## VERSIONING

Newer **ld-elf.so.1** might provide some features or changes in runtime behavior that cannot be easily detected at runtime by checking of the normal exported symbols. Note that it is almost always wrong to verify __FreeBSD_version in userspace to detect features, either at compile or at run time, because either kernel, or libc, or environment variables could not match the running **ld-elf.so.1**.

To solve the problem, **ld-elf.so.1** exports some feature indicators in the FreeBSD private symbols namespace FBSDprivate_1.0. Symbols start with the _rtld_version prefix. Current list of defined symbols and corresponding features is:

_rtld_version__FreeBSD_version
: Symbol exports the value of the __FreeBSD_version definition as it was provided during the **ld-elf.so.1** build. The symbol is always present since the _rtld_version facility was introduced.

_rtld_version_laddr_offset
: The *l_addr* member of the *link_map* structure contains the load offset of the shared object. Before that, *l_addr* contained the base address of the library. See dlinfo(3).

  Also it indicates the presence of *l_refname* member of the structure.

_rtld_version_dlpi_tls_data
: The *dlpi_tls_data* member of the structure *dl_phdr_info* contains the address of the module TLS segment for the calling thread, and not the address of the initialization segment.

## FILES

| | |
|---|---|
| */var/run/ld-elf.so.hints* | Hints file. |
| */var/run/ld-elf32.so.hints* | Hints file for 32-bit binaries on 64-bit system. |
| */etc/libmap.conf* | The libmap configuration file. |
| */etc/libmap32.conf* | The libmap configuration file for 32-bit binaries on 64-bit system. |

## SEE ALSO

ld(1), ldd(1), dlinfo(3), capsicum(4), elf(5), libmap.conf(5), ldconfig(8)