NAME

uselib − load shared library

SYNOPSIS

#include <unistd.h>

[[deprecated]] int uselib(const char *****library*);**

DESCRIPTION

The system call **uselib**() serves to load a shared library to be used by the calling process. It is given a pathname. The address where to load is found in the library itself. The library can have any recognized binary format.

RETURN VALUE

On success, zero is returned. On error, −1 is returned, and *errno* is set to indicate the error.

ERRORS

In addition to all of the error codes returned by **open**(2) and **mmap**(2), the following may also be returned:

EACCES
The library specified by *library* does not have read or execute permission, or the caller does not have search permission for one of the directories in the path prefix. (See also **path_resolution**(7).)

ENFILE
The system-wide limit on the total number of open files has been reached.

ENOEXEC
The file specified by *library* is not an executable of a known type; for example, it does not have the correct magic numbers.

STANDARDS

uselib() is Linux-specific, and should not be used in programs intended to be portable.

NOTES

This obsolete system call is not supported by glibc. No declaration is provided in glibc headers, but, through a quirk of history, glibc before glibc 2.23 did export an ABI for this system call. Therefore, in order to employ this system call, it was sufficient to manually declare the interface in your code; alternatively, you could invoke the system call using **syscall**(2).

In ancient libc versions (before glibc 2.0), **uselib**() was used to load the shared libraries with names found in an array of names in the binary.

Since Linux 3.15, this system call is available only when the kernel is configured with the **CONFIG_USELIB** option.

SEE ALSO

ar(1), **gcc**(1), **ld**(1), **ldd**(1), **mmap**(2), **open**(2), **dlopen**(3), **capabilities**(7), **ld.so**(8)