NAME

dlopen – open a shared object

SYNOPSIS

cc [*flag* ...] *file* ... **–ldl** [*library* ...]

#include <dlfcn.h>

void * dlopen(const char * *pathname***, int** *mode***);**

MT-LEVEL

MT-Safe

DESCRIPTION

dlopen() is one of a family of routines that give the user direct access to the dynamic linking facilities. (see *Linker and Libraries Guide*). These routines are made available via the library loaded when the option **-ldl** is passed to the link-editor.

These routines are available to dynamically linked processes ONLY.

dlopen() makes a shared object available to a running process. **dlopen()** returns to the process a "handle" which the process may use on subsequent calls to **dlsym()** and **dlclose()**. The value of this *handle* should not be interpreted in any way by the process. *pathname* is the path name of the object to be opened. A path name containing an embedded **'/'** is interpreted as an absolute path or relative to the current directory, otherwise the set of search paths currently in effect by the run-time linker will be used to locate the specified file (see *NOTES* section below).

If the value of *pathname* is 0, **dlopen()** makes the symbols contained in the original **a.out**, any objects loaded at program startup with the **a.out**, and any objects that were loaded using **dlopen()** together with the **RTLD_GLOBAL** flag, available through **dlsym()**.

When a shared object is brought into the address space of a process, it may contain references to symbols whose addresses are not known until the object is loaded. These references must be relocated before the symbols can be accessed. The *mode* parameter governs when these relocations take place and may have the following values:

RTLD_LAZY Only references to data symbols are relocated when the object is first loaded. References to functions are not relocated until a given function is invoked for the first time. This *mode* should improve performance, since a process may not reference all of the functions in any given shared object. This behavior mimics the normal loading of shared object dependencies by a dynamic executable during process initialization.

RTLD_NOW All necessary relocations are performed when the object is first loaded. This may waste some processing, if relocations are performed for functions that are never referenced. This behavior may be useful for applications that need to know as soon as an object is loaded that all symbols referenced during execution will be available.

Any object loaded by **dlopen()** that requires relocations against global symbols can reference the symbols in the **a.out**, any objects loaded at program startup, from the object itself, and from any dependencies the object references. By default, the relocations of an object loaded by one **dlopen( )** invocation may not reference symbols from objects loaded by a different **dlopen( )** invocation. However, the *mode* parameter may also be **ored** with the following value to effect the scope of symbol availability:

RTLD_GLOBAL The objects symbols are made available for the relocation processing of any other object. In addition, symbol lookup using **dlopen(0,** *mode***)** and an associated **dlsym( )**, allows objects loaded with this *mode* to be searched.

RETURN VALUES

If *pathname* cannot be found, cannot be opened for reading, is not a shared or relocatable object, or if an error occurs during the process of loading *pathname* or relocating its symbolic references, **dlopen( )** will return **NULL**. More detailed diagnostic information will be available through **dlerror( )**.

SEE ALSO

ld(1), **dladdr**(3X), **dlclose**(3X), **dlerror**(3X), **dlsym**(3X)
Linker and Libraries Guide

NOTES

If other shared objects were link edited with *pathname* when *pathname* was built (i.e., the *pathname* has dependencies on other shared objects), those objects will automatically be loaded by **dlopen( )**. The directory search path used to find both *pathname* and the other *needed* objects may be affected by setting the environment variable **LD_LIBRARY_PATH** (which is analyzed once at process startup), or from a run-path setting within the application or the shared object from which the **dlopen( )** originated. These search rules will only be applied to pathnames that do not contain an embedded **'/'**. Objects whose names resolve to the same absolute or relative path name may be opened any number of times using **dlopen( )**, however, the object referenced will only be loaded once into the address space of the current process.

Some symbols defined in dynamic executables or shared objects may not be available to the runtime linker. The symbol table created by **ld** for use by the runtime linker might contain only a subset of the symbols defined in the object.