NOMBRE

dlopen: obtenga acceso a un archivo de objeto ejecutable

SINOPSIS

```
#include <dlfcn.h>

void *dlopen(const char *file, int mode);
```

DESCRIPCIÓN

dlopen() hace que un archivo de objeto ejecutable especificado por *archivo* esté disponible para el programa que llama. La clase de archivos elegibles para esta operación y la forma de su construcción están especificadas por la implementación, aunque normalmente dichos archivos son objetos ejecutables como bibliotecas compartidas, archivos o programas reubicables. Tenga en cuenta que algunas implementaciones permiten la construcción de dependencias entre dichos objetos que están incrustados en archivos. En tales casos, una operación *dlopen()* cargará dichas dependencias además del objeto al que hace referencia el *archivo* . Las implementaciones también pueden imponer restricciones específicas en la construcción de programas que pueden emplear *dlopen()* y sus servicios relacionados.

Un *dlopen()* exitoso devuelve un *identificador* que la persona que llama puede usar en llamadas posteriores a *dlsym()* y *dlclose()* . El valor de este *identificador* no debe ser interpretado de ninguna manera por la persona que llama.

El archivo se utiliza para construir una ruta de acceso al archivo de objeto. Si *el archivo* contiene un carácter de barra inclinada, el argumento del *archivo* se usa como el nombre de la ruta del archivo. De lo contrario, *el archivo* se usa de manera dependiente de la implementación para generar un nombre de ruta.

Si el valor del *archivo* es 0, *dlopen()* proporciona un *identificador* en un objeto de símbolo global. Este objeto proporciona acceso a los símbolos de un conjunto ordenado de objetos que consta del archivo de imagen del programa original, junto con cualquier objeto cargado al inicio del programa según lo especificado por ese archivo de imagen de proceso (por ejemplo, bibliotecas compartidas) y el conjunto de objetos cargados. usando una operación *dlopen()* junto con el indicador RTLD_GLOBAL. Como el último conjunto de objetos puede cambiar durante la ejecución, el conjunto identificado por *handle* también puede cambiar dinámicamente.

Solo se trae una única copia de un archivo de objeto al espacio de direcciones, incluso si se invoca *dlopen()* varias veces en referencia al archivo, e incluso si se usan diferentes nombres de ruta para hacer referencia al archivo.

El parámetro de *modo* describe cómo *operará dlopen()* en *el archivo* con respecto al procesamiento de las reubicaciones y el alcance de la visibilidad de los símbolos provistos dentro *del archivo* . Cuando un objeto se introduce en el espacio de direcciones de un proceso, puede contener referencias a símbolos cuyas direcciones no se conocen hasta que se carga el objeto. Estas referencias deben reubicarse antes de poder acceder a los símbolos. El parámetro de *modo* rige cuándo tienen lugar estas reubicaciones y puede tener los siguientes valores:

RTLD_LAZY

Las reubicaciones se realizan en un momento dependiente de la implementación, que va desde el momento de la llamada *dlopen()* hasta que se produce la primera referencia a un símbolo dado. La especificación de RTLD_LAZY debería mejorar el rendimiento en las implementaciones que admitan el enlace dinámico de símbolos, ya que es posible que un proceso no haga referencia a todas las funciones en un objeto dado. Y, para los sistemas que soportan la resolución dinámica de símbolos para la ejecución normal del proceso, este comportamiento imita el manejo normal de la ejecución del proceso.

RTLD_NOW
: Todas las reubicaciones necesarias se realizan cuando el objeto se carga por primera vez. Esto puede desperdiciar algo de procesamiento si se realizan reubicaciones para funciones a las que nunca se hace referencia. Este comportamiento puede ser útil para las aplicaciones que necesitan saber, tan pronto como se carga un objeto, que todos los símbolos a los que se hace referencia durante la ejecución estarán disponibles.

Cualquier objeto cargado por *dlopen()* que requiera reubicaciones contra símbolos globales puede hacer referencia a los símbolos en el archivo de imagen de proceso original, cualquier objeto cargado al inicio del programa, desde el objeto mismo así como cualquier otro objeto incluido en la misma invocación de *dlopen()* , y cualquier objeto que se haya cargado en cualquier invocación de *dlopen()* y que haya especificado el indicador RTLD_GLOBAL. Para determinar el alcance de la visibilidad de los símbolos cargados con una invocación *dlopen()* , el parámetro de *modo* debe ser bit a bit o con uno de los siguientes valores:

RTLD_GLOBAL
: Los símbolos del objeto están disponibles para el proceso de reubicación de cualquier otro objeto. Además, la búsqueda de símbolos usando *dlopen* (*0, modo*) y un *dlsym()* asociado permite buscar objetos cargados con este *modo* .

RTLD_LOCAL
: Los símbolos del objeto no están disponibles para el proceso de reubicación de ningún otro objeto.

Si no se especifican RTLD_GLOBAL ni RTLD_LOCAL, se aplicará un comportamiento predeterminado especificado por la implementación.

If a *file* is specified in multiple *dlopen()* invocations, *mode* is interpreted at each invocation. Note, however, that once RTLD_NOW has been specified all relocations will have been completed rendering further RTLD_NOW operations redundant and any further RTLD_LAZY operations irrelevant. Similarly note that once RTLD_GLOBAL has been specified the object will maintain the RTLD_GLOBAL status regardless of any previous or future specification of RTLD_LOCAL, so long as the object remains in the address space (see *dlclose()*).

Symbols introduced into a program through calls to *dlopen()* may be used in relocation activities. Symbols so introduced may duplicate symbols already defined by the program or previous *dlopen()* operations. To resolve the ambiguities such a situation might present, the resolution of a symbol reference to symbol definition is based on a symbol resolution order. Two such resolution orders are defined: *load* or *dependency* ordering. *Load* order establishes an ordering among symbol definitions, such that the definition first loaded (including definitions from the image file and any dependent objects loaded with it) has priority over objects added later (via *dlopen()*). *Load* ordering is used in relocation processing. *Dependency* ordering uses a breadth-first order starting with a given object, then all of its dependencies, then any dependents of those, iterating until all dependencies are satisfied. With the exception of the global symbol object obtained via a *dlopen()* operation on a *file* of 0, *dependency* ordering is used by the *dlsym()* function. *Load* ordering is used in *dlsym()* operations upon the global symbol object.

When an object is first made accessible via *dlopen()* it and its dependent objects are added in *dependency* order. Once all the objects are added, relocations are performed using *load* order. Note that if an object or its dependencies had been previously loaded, the *load* and *dependency* orders may yield different resolutions.

The symbols introduced by *dlopen()* operations, and available through *dlsym()* are at a minimum those which are exported as symbols of global scope by the object. Typically such symbols will be those that were specified in (for example) C source code as having *extern* linkage. The precise manner in which an implementation constructs the set of exported symbols for a *dlopen()* object is specified by that implementation.

RETURN VALUE

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

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

dlclose(), *dlerror()*, *dlsym()*.