Lenguajes de Programación 2

Librerías

Introducción

- Una librería es un programa cuyos elementos pueden ser utilizados por otros programas.
- La forma de crear y utilizar una librería dependen del lenguaje de programación o del sistema operativo.

Librerías en C/C++

- Dos tipos
  - Estáticas: Enlazadas con los OBJ's para forman un nuevo programa.
  - Dinámicas: Enlazadas con el programa en tiempo de ejecución.

Librerías en C/C++

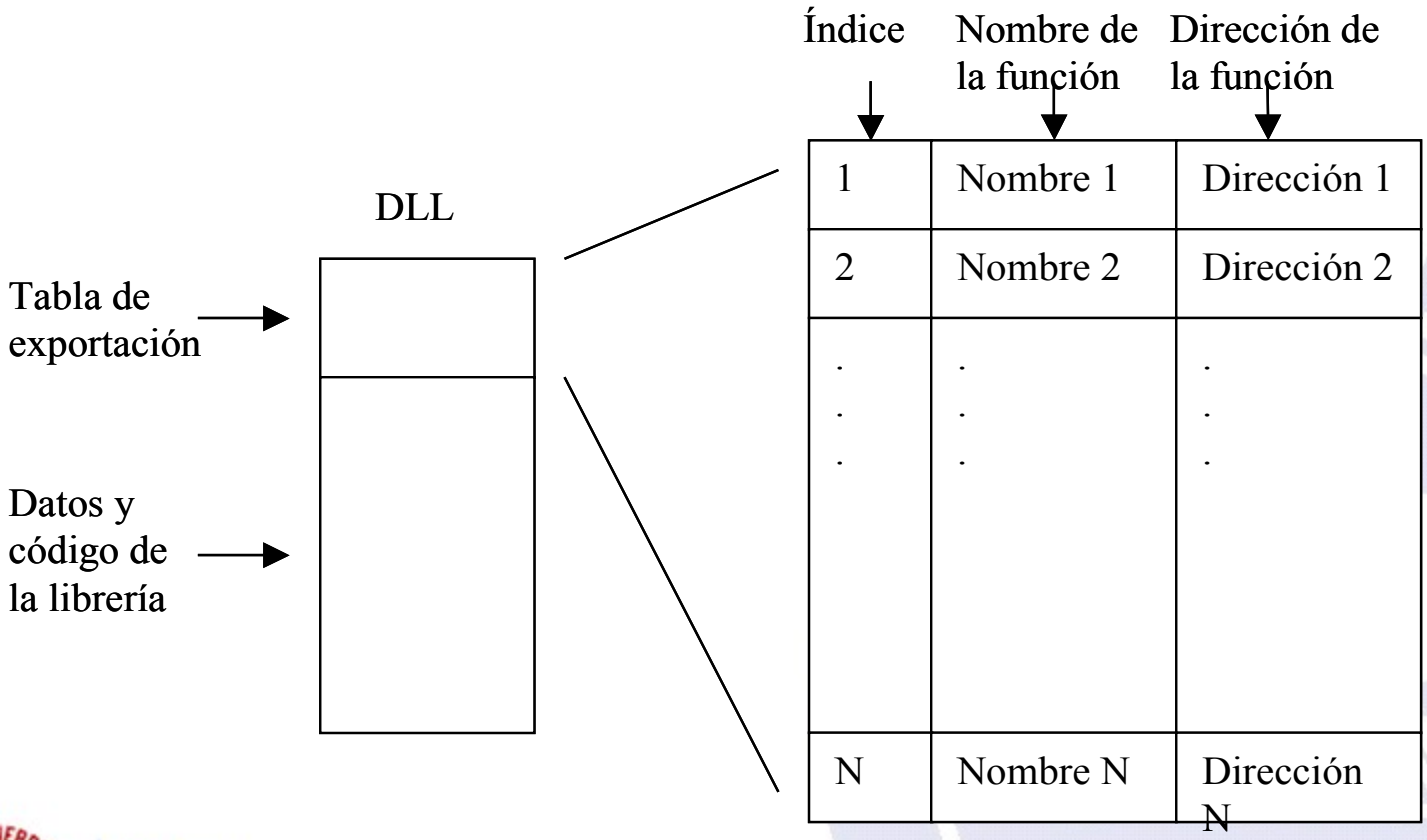
- Librerías Estáticas
  - Forman parte del programa final
  - Para utilizarlas hay que agregarlas al proceso de enlace. Si se utiliza un IDE, se agrega junto con los archivos fuente.

Librerías en C/C++

- Librerías Dinámicas
  - NO forman parte del programa final.
  - Son ejecutadas por los hilos de los procesos que las utilizan.
  - Exponen un conjunto de funciones exportadas.
  - La llamada a una función exportada es mapea al espacio de direc. del hilo llamador.

Librerías en C/C++

- Librerías Dinámicas, Estructura



Librerías en C/C++

- Librerías Dinámicas, Creación
  - En un IDE, crear un proyecto especificando el tipo adecuado para un DLL.
  - Agregar los archivos
    - *.H: Prototipos de las funciones exportadas.
    - *.C o *.CPP: Implementación.
  - Indicar las funciones exportables usando
    - __declspec (dllexport)

    ó
    - *.DEF: Declaración de las funciones a exportar.
  - Compilar.

Librerías en C/C++

- Librerías Dinámicas, Ejemplo
  - Archivo cabecera: MiLibreria.H

```
#include <windows.h>
#ifdef __cplusplus
  extern "C" {
#endif

void Saludame(char * szNombre);

#ifdef __cplusplus
  }
#endif
```

Librerías en C/C++

- Librerías Dinámicas, Ejemplo
  - Archivo implementación: MiLibreria.cpp

```
BOOL WINAPI DllMain ( HANDLE hModule, DWORD dwReason, LPVOID
  lpReserved ) {
  switch ( dwReason ) {
    case DLL_PROCESS_ATTACH:
         break;
    case DLL_THREAD_ATTACH:
         break;
    case DLL_THREAD_DETACH:
         break;
    case DLL_PROCESS_DETACH:
         break;
  }
  return TRUE;
}
```

Librerías en C/C++

- Librerías Dinámicas, Ejemplo
  - Archivo implementación: MiLibreria.cpp

```
#include "MiLibreria.h"
BOOL WINAPI DllMain ( ... ) {
  // Aqui va lo indicado en la anterior diapositiva
}
void Saludame(char* szNombre) {
  printf("Hola %s \n", szNombre);
}
```

Librerías en C/C++

- Librerías Dinámicas, Ejemplo
  - Indicar funciones exportable
    - Archivo implementación: MiLibreria.def

```
LIBRARY SALUDAMEDLL
DESCRIPTION "Implementación de una DLL"
EXPORTS
 Saludame @1
```

    - Usando

```
#include <windows.h>
#ifdef __cplusplus extern "C" {
#endif
__declspec (dllexport) void Saludame(char * szNombre);
#ifdef __cplusplus }
#endif
```

Librerías en C/C++

- Librerías Dinámicas, Utilización
  - Modo implícito
    - Se utiliza el archivo LIB generado al compilar la DLL, en la configuración correspondiente al IDE.
    - Se incluye el archivo cabecera de la DLL en los archivos de implementación que llamen a alguna de las funciones exportadas.

Librerías en C/C++

- Librerías Dinámicas, Utilización
  - Modo implícito, ejemplo

```
#include "MiLibreria.h"

int main( ... ) {
    Saludame("JUAN");
    return 0;
}
```

Librerías en C/C++

- Librerías Dinámicas, Utilización
  - Modo explícito:
    - Se utilizan las funciones del API de Windows
      - LoadLibrary
      - FreeLibrary
      - GetProcAddress
    - Se utilizan punteros a función adecuador para cada función exportada que se desee utilizar.

Librerías en C/C++

- Librerías Dinámicas, Utilización
  - Modo explícito, Ejemplo

```
#include <windows.h>
typedef void (* PFUNC) (char *);
int main( ... ) {
    HINSTANCE hDll = LoadLibrary("MyDll.dll");
   if ( hDll != NULL ) {
      PFUNC pfnSaludo = (PFUNC)GetProcAddress(hDll, "Saludame");
      if( pfnSaludo != NULL )
            pfnSaludo("María");
      FreeLibrary( hDll );
    }
    return 0;
}
```

Librerías en C/C++

- Librerías Dinámicas, Mecanismo de búsqueda
  - En el directorio donde se encuentra el ejecutable de la aplicación.
  - En el directorio de trabajo actual
  - En el directorio System. Si es NT o Windows 2000, en el directorio System32.
  - En el directorio de Windows.
  - En la lista de directorios de la variable PATH.

Librerías en Java

- Definiciones
  - Están estrechamente relacionadas a los directorios.
  - Una librería es un directorio con un conjunto de archivos CLASS que forman parte de la librería.
  - Las librerías reciben el nombre de paquetes.
  - Para encontrar una librería, tanto el compilador como el intérprete utilizan la variable de entorno CLASSPATH

Librerías en Java

- Creación
  - Para indicar que las clases implementadas en un archivo .JAVA pertenecen a un paquete, se coloca como primera línea del archivo la instrucción:

    package NombreDelPaquete;

  - Adicionalmente, los archivos CLASS generados deberán de copiarse a una carpeta con el nombre del paquete.

Librerías en Java

- Ejemplo
  - Deseamos crear dos clases A y B que formen parte de un nuevo paquete MiPaquete. El archivo .JAVA ´podría ser:

```
// Archivo: MisClases.java
package MiPaquete;
public class A {}
public class B {}
```

Librerías en Java

- Ejemplo
  - Creamos un programa que instancie las clases A y B. Para esto debemos de crear un directorio MiPaquete en el mismo directorio donde se compilará y ejecutará este programa.

```
// Archivo: ProbandoMiPaquete.java
import MiPaquete;
class ProbandoMiPaquete {
  public static void main(String[] args) {
        A objA = new A();
        B objB = new B();
  }
}
```

Librerías en Java

- Archivos JAR
  - Comprimen uno o más paquetes conservando la información acerca de los directorios (paquetes) donde se encuentran cada archivo .CLASS.
  - Pueden ser utilizados en CLASSPATH como lugares de búsqueda de archivos .CLASS.
  - Constituyen una mejor forma distribuir un conjunto de archivos .CLASS, como los que forman una aplicación, dado que se tiene la opción de comprimir estos.

Librerías en Java

- Archivos JAR, Creación
  - Se utiliza el programa "jar.exe", instalado por el JDK.
  - Ejemplo 1: Creación del *Paq1.jar*

```
jar  cvf  Paq1.jar  Foo.class  Bar.class
```

  - Ejemplo 2: Utilización de un archivo de manifiesto *manifiesto.mf* y empaquetado de todos los .CLASS en el directorio *Foo*

```
jar cvfm Paq2.jar manifiesto.mf -C foo/ .
```