Apéndice B
Tecnología JNI

Apéndice B

Tecnología JNI (Java Native Interface)

La plataforma Java es relativamente nueva, lo que significa que algunas veces podríamos necesitar integrar programas escritos en Java con servicios, programas o APIs existentes escritos en lenguajes distintos a Java. La plataforma Java proporciona el Interfaz Nativo Java (JNI) para ayudarnos con este tipo de integración.

El JNI define una convención de nombres y llamadas para que la Máquina Virtual Java pueda localizar e invocar a los métodos nativos. De hecho, JNI está construido dentro de la máquina virtual Java, por lo que ésta puede llamar a sistemas locales para realizar entrada / salida, gráficos, trabajos de red y operaciones de threads sobre el host del sistema operativo[JNI 02].

Para poder utilizar JNI bajo linux se deben de llevar acabo los siguientes pasos:

1. **Compilar el Programa**

Para compilar el programa sólo ejecutamos el comando del compilador javac como lo haríamos normalmente:

javac Conexion.java

Luego, necesitamos generar una librería con la declaración del método nativo y la implementación del método nativo para llamar a funciones declaradas.

2. **Generar el Fichero de Cabecera**

Para generar una librería, ejecutamos el comando javah sobre la clase Conexion En este caso, el fichero de cabecera generado se llama Conexion.h. Proporciona una

firma de método que debemos utilizar cuando implementemos la funciones nativas declaradas.

javah -jni Conexión

3. Firma del Método

La librería Conexion.h define el interfase para mapear el método en lenguaje Java a la función nativa C. Utiliza una firma de método para mapear los argumentos y valor de retorno del método mappedfile.transferir.java al método nativo transferir de la librería nativelib. Aquí está la firma del método nativo transferir:

```
/*
 * Class:     Conexion
 * Method:    transferir
 * Signature: (Ljava/lang/String;)[B
 */
JNIEXPORT jbyteArray JNICALL Java_Conexion_transferir
(JNIEnv *, jobject, jstring);
```

Los parámetros de la firma de la función son los siguientes:

JNIEnv *: Un puntero al entorno JNI. Este puntero es un manejador del thread actual en la máquina virtual Java y contiene mapeos y otra información útil.

jobject: Una referencia a un método que llama a este código nativo. Si el método llamante es estático, este parámetro podría ser del tipo jclass en lugar de jobject.

jstring: El parámetro suministrado al método nativo. En este ejemplo, es el nombre del fichero a leer.

4. Implementar el Método Nativo

En el archivo de código nativo C , la definición de transferir es una copia de la declaración C contenida en el fichero Conexion.h. La definición es seguida por la implementación del método nativo. JNI proporciona mapeo por defecto tanto para C como para C++.

5. **Compilar la Librería Dinámica o de Objetos Compartidos**

La librería necesita ser compilada como una librería dinámica o de objetos compartidos para que pueda ser cargada durante la ejecución. Las librerías o archivos estáticos son compiladas dentro de un ejecutable y no pueden ser cargadas en tiempo de ejecución. La librería dinámica para el ejemplo transferir se compila de esta forma:

```
gcc -o libnativelib.so -shared -Wl,-soname,libnative.so -I/export/home/jdk1.2/include -I/export/home/jdk1.2/include/linux Conexion.c -static –lc
```

6. **Ejecutar el Ejemplo**

Para ejecutar el ejemplo, la máquina virtual Java necesita poder encontrar la librería nativa. Para hacer esto, configuramos el path de librerías al path actual de esta forma:

```
LD_LIBRARY_PATH=`pwd`
Export LD_LIBRARY_PATH
```

Con el path de librerías especificado de forma apropiada a nuestra plataforma, llamamos al programa como lo haríamos normalmente:

```
java Conexion
```