はじめに

64bit Java から 32bit でコンパイルした Native は実行できないで注意。

```
Can't load IA 32-bit .dll on a AMD 64-bit platform
```

のようなエラーが出る。

共有ライブラリの検索パスについて

OS	JNI	JNA
windows	PATH 環境変数 java.library.path	PATH 環境変数 jna.library.path
Linux	LD_LIBRARY_PATH 環境変数 java.library.path	LD_LIBRARY_PATH 環境変数 jna.library.path

JNI

Java から Native を呼ぶためのもの。
手順は以下のとおり。

1.Java プログラムの作成とコンパイル
2.Java からヘッダファイル作成
3.C プログラムの作成とコンパイル
4. 実行

Java プログラムの作成とコンパイル

■ HelloWorldJNI.java

■コンパイル

```
javac HelloWorldJNI.java
```

Java からヘッダファイル作成

■ヘッダファイル作成

```
javah HelloWorldJNI
```

で HelloWorldJNI.h が作成される。

C プログラムの作成とコンパイル

■ C プログラムの作成

HelloWorldJNI.h のプロトタイプ宣言を確認して C プログラムを書く

■コンパイル (mingw の例)

gnupack（mingw）を使った場合

```
gcc -shared HelloWorldJNI.c -Id:/java/include -Id:/java/include/win32 -Wl,--enable-auto-import
-Wl,--add-stdcall-alias -o HelloWorldJNI.dll
```

みたいな感じ。JDK の include と include/win32 に incllude path を通す。

```
-Wl,--add-stdcall-alias
```

を付けないと、実行時に関数が見つけられずエラーになるので注意。

また、gnupack でも mingw の場合は

```
/d/java/include
```

のような形式ではなく

```
d:/java/include
```

のようなパスの書き方をする。

■コンパイル (Linux の gcc の例)

```
gcc -fPIC -shared HelloWorldJNI.c -I /usr/lib/jvm/java-1.7.0-openjdk-1.7.0.25.x86_64/include -I
/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.25.x86_64/include/linux/ -o libHelloWorldJNI.so
```

Linux の場合、共有ライブラリは lib****.so のように lib が先頭に着くので注意。

実行

■ Windows の場合

Windows の場合は共有ライブラリは

```
PATH 環境変数
java.library.path オプション
```

から検索される。以下は実行例

```
java HelloWorldJNI
java -Djava.library.path=d:/temp
```

■ Linux の場合

Linux の場合は共有ライブラリは

```
LD_LIBRARY_PATH 環境変数
java.library.path オプション
```

から検索される。以下は実行例

```
java HelloWorldJNI
java -Djava.library.path=. HelloWorldJNI
```

JNA

JNI より簡単にネイティブライブラリを使う。

例１

例２

自分で DLL を作成し、JNA で呼び出す

■ hello.c

■コンパイル

Windows の場合

```
gcc -shared -o hello.dll hello.c
```

Linux の場合

```
gcc -fPIC -shared -o libhello.so hello.c
```

※ライブラリ名に「lib」をつける必要があるので注意

■ test.java

■実行

Windows の場合
Linux の場合は LD_LIBRARY_PATH 環境変数で示すパスに so ファイルを置く必要がある。

```
export LD_LIBRARY_PATH=.:$LD_LIBRARY_PATH
```

など。

Windows の場合
Windows の場合は共有ライブラリは

```
PATH 環境変数
jna.library.path オプション
```

から検索される。以下は実行例

```
java test
java -Djna.library.path=d:/temp
```

Linux の場合
Linux の場合は共有ライブラリは

```
LD_LIBRARY_PATH 環境変数
jna.library.path オプション
```

から検索される。以下は実行例

```
java test
java -Djna.library.path=. test
```