ASR TP4
Bibliothèques dynamiques et interfaçage avec Java

JNI

JNI pour Java Native Interface, permet d'interfacer une bibliothèque écrite en C avec du code Java (`http://docs.oracle.com/javase/7/docs/technotes/guides/jni/spec/jniTOC.html`).

Le principe à suivre est le suivant :

1. définir le prototype des fonctions de la bibliothèque dans le programme Java, en les faisant précéder du mot-clé `native`,
2. générer le fichier `.class` correspondant,
3. appelé la commande `javah` sur le fichier `.class` afin de générer un fichier `.h` faisant correspondre les types C et Java,
4. écrire le fichier `.c` associé faisant appel à la bibliothèque.

Exemple, fichier `Main.java` :

```
1  public class Main {
2
3      // Prototype de la fonction native
4      public native static int quaranteDeux();
5      public native static int fact(int n);
6
7      static { // Chargement de la bibliothèque libtest.so
8          System.loadLibrary("test");
9      }
10
11     public static void main(String[] args) {
12         System.out.println(Main.quaranteDeux());
13         System.out.println(Main.fact(5));
14     }
15
16 }
```

Génération automatique à partir de la commande `javah Main` du fichier `Main.h` contenant les définitions C des fonctions natives :

```
1  /* DO NOT EDIT THIS FILE - it is machine generated */
2  #include <jni.h>
3  /* Header for class Main */
4
5  #ifndef _Included_Main
6  #define _Included_Main
7  #ifdef __cplusplus
8  extern "C" {
9  #endif
10 /*
```

```
11  * Class:     Main
12  * Method:    quaranteDeux
13  * Signature: ()I
14  */
15 JNIEXPORT jint JNICALL Java_Main_quaranteDeux
16   (JNIEnv *, jclass);
17
18 /*
19  * Class:     Main
20  * Method:    fact
21  * Signature: (I)I
22  */
23 JNIEXPORT jint JNICALL Java_Main_fact
24   (JNIEnv *, jclass, jint);
25
26 #ifdef __cplusplus
27 }
28 #endif
29 #endif
```

Création du fichier `Main.c` correspondant faisant la `glue` avec la bibliothèque :

```
1 #include "Main.h"
2 #include "test.h"
3
4 JNIEXPORT jint JNICALL Java_Main_quaranteDeux (JNIEnv * env, jclass obj) {
5   return quaranteDeux();
6 }
7
8 JNIEXPORT jint JNICALL Java_Main_fact(JNIEnv* env, jclass obj, jint n) {
9   return fact(n);
10 }
```

Fichier d'en-tête de la bibliothèque `test.h` :

```
1 #ifndef TEST_HPP
2 #define TEST_HPP
3
4 int quaranteDeux();
5 int fact( int n );
6
7 #endif
```

Code de la bibliothèque `test.c` :

```
1 #include "test.h"
2
3 int quaranteDeux()
4 {
5   return 42;
6 }
7
8 int fact(int n)
9 {
```

```
10    return n <= 1 ? 1 : n * fact( n-1 );
11  }
```

Il ne reste plus qu'à compiler les fichiers `Main.c` et `test.c` afin de générer la bibliothèque `libtest.so`, voir le cours donné en lien à la section 2.2.

Procéder de même pour utiliser votre bibliothèque `libfcts.so`.