1V－05　Javaネイティブメソッドを自動生成する手法

望月　泰行　　木野　茂徳　　下間　芳樹
三菱電機（株）情報技術総合研究所

1. はじめに

Java 言語はさまざまな分野で利用されはじめているが、他のプログラミング言語により記述されているライブラリも多く存在する。そのため、Java 言語でシステムシステム開発を行なう際には、Java プログラムから既存のプログラミング言語の API にアクセスするネイティブメソッドが必要性となる。本稿では、既存のプログラミング言語としてC言語をとりあげ、Cライブラリの関数に対し透過的なアクセスを可能にする Java ネイティブメソッドの考え方をもとに、Java ネイティブメソッドの実装を自動生成する手法を検討した。

2. C関数に対する透過的なアクセス

C言語の関数に対して透過的にアクセスするJavaネイティブメソッドについて説明する。これは、与えられたC関数と等価なJavaメソッドを実装するということである。例えば、Cライブラリにおいて次の関数が定義されているとする。

```
int getData(COND *cond, DATA *data);
```

ここで、CONDとDATAはともに構造体の別名であり、intを返す関数getDataは条件式condでデータを検索し、検索結果をdataに格納する。このC関数に対して透過的にアクセスするJavaネイティブメソッドは次にように定義される。

```
native int getData(COND cond, DATA data);
```

ここで、CONDとDATAはJavaのクラスであり、C関数の引数の構造体と等価なデータ構造を持つ。

3. ネイティブメソッドの実装

C言語の関数に対して透過的にアクセスする Javaネイティブメソッドの実装は次の４つの部分から構成される。

- 引数のJavaデータからCデータへの変換
- Cライブラリの関数実行
- 引数のCデータからJavaデータへの変換

ネイティブメソッドの実装において重要なのは、エラー通知のポリシーとメモリ管理のポリシーである。特にエラー通知に関しては、Cライブラリの関数実行中に発生するエラーはJava側に透過的に通知し、それ以外のエラーはJavaのException を使って通知するべきである。

4. データ変換の実装

C言語のデータ構造として、ここでは次の２つを扱う。第１に、(a)primitiveな型とその配列：同じビット長のJavaのprimitiveな型に対応させる。例えば、C言語のchar型はJavaのbyte型に対応する。次に、(b)構造体とその配列：構造体ごとにJavaのクラスを定義し、構造体のフィールドと同じフィールドをJavaクラスに設ける。

構造体のフィールドは上記のいずれかの型であるので、どのようなデータの変換ルーチンも、再帰的に記述することによって、最終的にはprimitiveな型のデータ変換に帰着する。

5. ネイティブメソッドの自動生成

上記の実装手法によれば、次のような制限を設けることにより、C言語の関数とデータ構造の宣言からネイティブメソッドのプログラムを自動生成することが可能となる。

- C関数の引数の入出力条件を記述する
- 共用体データは自動生成の対象としない
- C配列の要素数の表現方法を記述する

6．おわりに

Cライブラリの関数に対して透過的にアクセスするJavaネイティブメソッドの実装手法について検討し、ネイティブメソッドを自動生成するための条件について述べた。

Automated Implementation Technique of Java Native Method
MOCHIZUKI Yasuyuki, KINO Shigenori, SHIMOTSUMA Yoshiki
Informatioon Technology R&D Center, Mitsubishi Electric Corporation
5-1-1 Ofuna, Kamakura, Kanagawa 2478501, Japan