gcj で遊んでみる

gcj については↓を読んでみる。
http://gcc.gnu.org/java/
http://www.shudo.net/article/Fedora-Core-Expert-200507-GCJ/
http://www.asahi-net.or.jp/~wg5k-ickw/html/online/gcj-3.2.1/gcj-ja.html

で、パッケージとか Jar とか動的ロードとかの扱いを試してみた。
gcc は、FreeBSD 6.2-RELEASE 上でコンパイルした gcc-4.2.1 を使用
テストコードは、最後に示しているものを利用。

■一度にコンパイル

```
gcj --main=test.Test test/*.java
```

■中間ファイルを作ってコンパイル

```
gcj -c test/Hoge.java
gcj -c test/Test.java
gcj --main=test.Test *.o
```

パッケージのルートが "." でない場合 (たとえば src/ とか) は

```
gcj -Isrc -c src/test/Test.java
gcj -Isrc -c src/test/Hoge.java
```

■ Class ファイルからオブジェクトコードを作る

```
javac test/*.java
gcj -c test/Test.class
gcj -c test/Hoge.class
gcj -c test/Test\$TestInner.class
gcj --main=test.Test *.o
```

内部クラスのオブジェクトコードを作るのも忘れないように
外部のパッケージを使っている場合も、その jar やフォルダを -I で指定する。

■ jar からオブジェクトコードを作る

```
javac test/*.java
jar cvf test.jar test/*.class
gcj -c test.jar
gcj --main=test.Test test.o
```

jar ファイルはそのままオブジェクトコードになる

■ shared object にする

```
gcj -o libHoge.so -shared test/Hoge.java
gcj --main=test.Test test/Test.java -L. -lHoge
```

jar ファイルを shared object にすることもできる

```
jar cvf test.jar test/*.class
gcj -o libTest.so -shared test.jar
```

■動的ロード (1)

```
javac test/Hoge.java
javac test/Test.java
gcj --main=Loader Loader.java
```

ロードするクラスへ正しくアクセスできないとだめ。
java や gij と違って起動時に classpath を設定できるわけではないので、
外部の jar を利用したいなら、
パッケージの jar ごと shared object にして動的リンクするか、
自分で管理する必要がある？

■動的ロード (2)

```
gcj -o libHoge.so -shared test/Hoge.java
javac test/Test.java
gcj --main=Loader Loader.java -L. -lHoge
```

実行時に、libHoge.so が LD_LIBRARYPATH で参照できる必要がある

■動的ロードの優先順位

```
gcj -o libHoge.so -shared test/Hoge.java
javac test/Test.java
```

とした後で、
Hoge.java のコンストラクタで、クラス名を表示する前に

```
System.out.println("Modified");
```

と表示させるように変更して

```
javac test/Hoge.java
```

とクラスを作る。で、次の二つを実行

リンクしない

```
gcj --main=Loader Loader.java
```

実行結果は、

```
miyo@vmouton:% ./a.out
modified
test.Hoge
test.Test
test.Test$TestInner
```

```
modified
test.Hoge
```

Hoge を変更しているので、当然。

libHoge.so をリンクする

```
gcj --main=Loader Loader.java -L. -lHoge
```

実行結果は、

```
miyo@vmouton:% ./a.out
test.Hoge
test.Test
test.Test$TestInner
test.Hoge
```

つまり Loader はもちろん、Test の方でも
コンパイル時に動的リンクした libHoge.so が優先される。
何か設定とかプログラムで変更できるのかな？

■ 1.5 のコード

gcj では、1.5 のコード、ジェネリクスや、
イテレーション構文、オートボキシングのあるコードはコンパイルできません。
が、一度 javac を使ってクラスファイルに落とすことで、gcj に通せます。

```
import java.util.*;

public class Test5{
  public static void main(String[] args){
    ArrayList<Integer> list = new ArrayList<Integer>();
    for(int i = 0; i < 10; i++){
      list.add(i);
    }
    for(Integer s: list){
      System.out.println(s);
    }
  }
}
```

なら、

```
javac Test5.java
gcj --main=Test5 Test5.class
```

とか

■テストコード

test/Hoge.java

```
package test;
public class Hoge{
  public Hoge(){
    //System.out.println("modified");
```

```
    System.out.println(this.getClass().getName());
  }
}
```

test/Test.java

```
public class Test{
  public Test(){
    System.out.println(this.getClass().getName());
    new TestInner();
    new Hoge();
  }
  class TestInner{
    public TestInner(){
      System.out.println(this.getClass().getName());
    }
  }
  public static void main(String[] args){
    new Test();
  }
}
```

Loader.java

```
public class Loader{

  public static void main(String[] args) throws Exception{
   ClassLoader.getSystemClassLoader().loadClass("test.Hoge").newInstance();
   ClassLoader.getSystemClassLoader().loadClass("test.Test").newInstance();
  }
}
```