

Announcements

- Program #5
 - is due Thursday
- Reading
 - <http://java.sun.com/docs/books/tutorial/native1.1/>
- Pickup old assignments in my office
 - Midterms (including re-grades)
 - Programming assignments

Java Native Interface

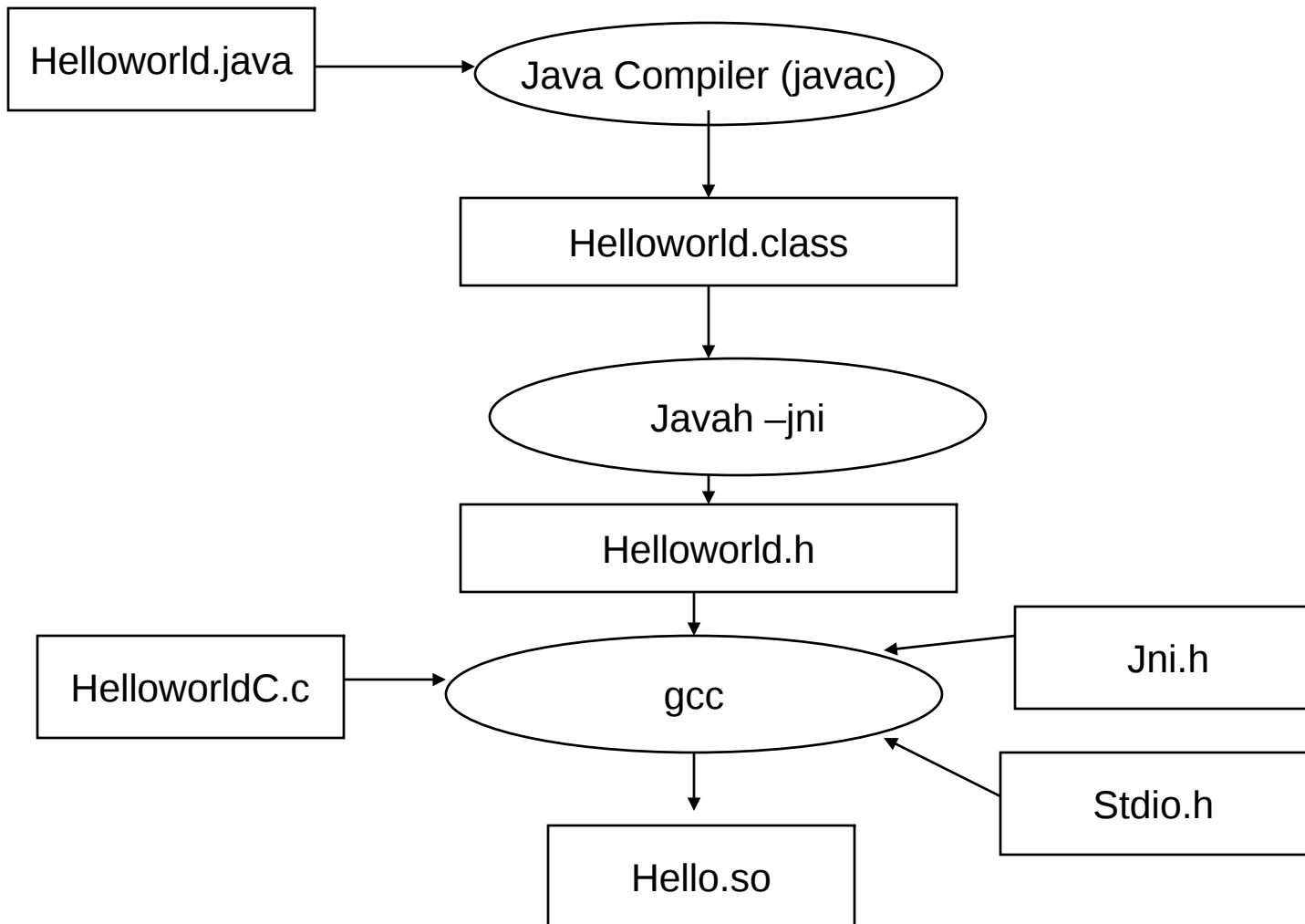
- Goals

- Sometimes need to access things in lower level language
- Provide full access to Java Objects in other languages

- Keys Ideas

- Need to identify signatures of native methods
- Need to incorporate native code into program

Process of Compiling Using JNI



Using JNI Functions

- Declaring Functions
 - Declare like a normal method, but add “native” keyword
 - `public native void displayHelloWorld();`
- Loading the Code
 - Use `loadLibrary` call
 - Making it static forces it to happen automatically
 - `System.loadLibrary("hello")`
- Invoke Native Method
 - `new HelloWorld().displayHelloWorld();`

Full Example of JNI

- Java Code

```
class HelloWorld {  
    public native void displayHelloWorld();  
    static { System.loadLibrary("hello"); }  
    public static void main(String[] args) {  
        new HelloWorld().displayHelloWorld();  
    }  
}
```

- C Function Signature

- Constructing the function name
 - Java_ + Full Classname + _ + Method Name

```
JNIEXPORT void JNICALL Java_HelloWorld_displayHelloWorld  
(JNIEnv *, jobject);
```

C Code for JNI Call

```
#include <jni.h>
```

```
#include "HelloWorld.h"
```

```
#include <stdio.h>
```

```
JNIEXPORT void JNICALL
```

```
    Java_HelloWorld_displayHelloWorld(JNIEnv *env, jobject obj)
```

```
{
```

```
    printf("Hello world!\n");
```

```
    return;
```

```
}
```

Accessing Java Data from C

- Mapping Types

Java Type	Native Type	Size (bits)
Boolean	Jboolean	8, unsigned
Byte	Jbyte	8
Char	Jchar	16
Short	Jshort	16
Int	Jint	32
Long	Jlong	64

Access Java Arrays

- Need to call special functions to get data and size
- Example:

```
JNIEXPORT jfloat JNICALL
```

```
Java_FloatArray_sumArray(JNIEnv *env, jobject obj, jfloatArray arr) {
```

```
    jfloat *body, sum = 0;
```

```
    jsize l, len;
```

```
    len = (*env)->GetArrayLength(env, arr);
```

```
    body = (*env)->GetFloatArrayElements(env, arr, 0);
```

```
    for (i=0; i<len; i++) {
```

```
        sum += body[i];
```

```
    }
```

```
    (*env)->ReleaseFloatArrayElements(env, arr, body, 0);
```

```
    return sum;
```

```
}
```

Calling Java Methods from C

- Steps

- Get Object handle
- Get MethodID – need to know signature
 - (argument-types)return-type
- Call method

- Example

```
JNIEXPORT void JNICALL Java_Callbacks_nativeMethod(
    JNIEnv *env, jobject obj, jint depth) {
    jclass cls = (*env)->GetObjectClass(env, obj);
    jmethodID mid = (*env)->GetMethodID(env, cls, "callback", "(I)V");
    if (mid == 0) { return; }
    printf("In C, depth = %d, about to enter Java\n", depth);
    (*env)->CallVoidMethod(env, obj, mid, depth);
    printf("In C, depth = %d, back from Java\n", depth);
}
```

Accessing Object Fields

- Getting FieldID

- Need to get a unique id for the field
- Must specify type information
- `fid = (*env)->GetFieldID(env, cls, "s", "Ljava/lang/String;");`
 - CIs is of type jclass
 - S is field name
- Can make one call to GetFieldID
 - Really computes offset for item

- Getting the data

- Pass the fid, env, and object
- `str = (*env)->GetObjectField(env, obj, fid);`

My Research

- Instrumentation

- Modifying Running Applications
- Uses
 - Measurement
 - Testing
 - Simulation

- Self Tuning Software

- Automatic selection of algorithms and parameters at runtime
- Allowing software to be tuned while running

- Understanding Parallel Programmers Effort

- What takes time in developing programs?
- How to improve programmer efficiency