

Appendix E: Java Native Interface Linking Error

The steps to implement the Java native methods on the C/C++ side are the following:

1. Declare a method in a Java class as native.

```
private native byte[] JUSBReadControl(String devicePath, byte type, byte request,
    short value, short index, short length);
```

2. Create the JNI header file with javah.

```
javah -jni usb.windows.JUSB
```

3. Include the header file in the C/C++ file that implements this native function.

4. Build the DLL

Usually we assume that the DLL function name corresponds to the function name we supplied in the JNI header file. In fact the compiler creates the function name according to the JNI header file.

Unfortunately, we encountered a problem that a Java native method got a fatal linking error while running the Java main program. We did step 1 to 4 as usual but the main application still claimed that no native method of that name exists in the DLL. The reason of such a failure is in mangling the function name by the Visual C++ Compiler. In other words: the compiler creates a new name for the JNI native function which does not correspond to the origin header file and therefore cannot be found by the Java native interface. To eliminate this malfunction of the compiler we used the **dumpbin** [7] command to get the names of the DLL supported functions. According to the dump function names we are able to check if a name mangling has happened or not. Look at the source of *JUSBReadControl* method in the *JUSB* class for further information.

DUMPBIN command

The **dumpbin** command can be used as follows:

```
dumpbin /EXPORTS /LINKERMEMBER C:\WINDOWS\SYSTEM32\jusb.dll
```

Table 56: Dumpbin command to see the export function of a DLL

The output of that command is shown in **Table 57**.

Output of DumpBin

```
Dump of file C:\WINDOWS\SYSTEM32\jusb.dll
Section contains the following exports for jusb.dll

1  0 000010AF _Java_usb_windows_DeviceImpl_closeHandle@12
2  1 00001050 _Java_usb_windows_DeviceImpl_getAttachedDeviceType@16
3  2 00001069 _Java_usb_windows_DeviceImpl_getConfigurationDescriptor@16
...
11 A 0000110E _Java_usb_windows_JUSB_JUSBReadControl@32
12 B 00001005 _Java_usb_windows_JUSB_doInterruptTransfer@20
13 C 00001104 _Java_usb_windows_JUSB_getConfigurationBuffer@16
14 D 000010F5 _Java_usb_windows_JUSB_getDevicePath@12
15 E 00001064 _Java_usb_windows_USB_getRootHubName@12
16 F 0000106E _Java_usb_windows_Windows_getHostControllerDevicePath@12
17 10 0000100F _Java_usb_windows_Windows_getHostControllerName@12
```

Table 57: Output of dumpbin command

A mangled function name has the following structure:

```
_Java_usb_Windows_JUSB_readControl@@YGPAV_jbyteArray@@PAUJNIEEnv_@
@PAV_jobject@@PAV_jstring@@PAV1@@@Z
```

Mangled function name

Table 58: A mangled function name by the compiler