Android 平台怎样使用第三方动态库

1、检查所需文件是否齐全

使用第三方动态库，应该至少有 2 个文件，一个是动态库（.so），另一个是包含动态库 API 声明的头文件(.h)
例：
Add.c（实际上如果使用第三方库时我们是没有源码的，这里为了方便举例）：

```
#include <stdio.h>

int Add(int x, int y)
{
    return x+y;
}
```

Add.h:

```
#ifndef _ADD_H_
#define _ADD_H_

int Add(int x, int y);

#endif
```

使用如下 Makefile(其中红色路径需要按环境修改)：

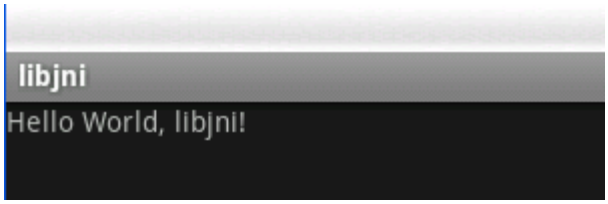
```
ANDROID_PATH = /home/rock/SV8860_SDK_v0.92/Source/skydroid1.6
TOOLCHAIN_PATH = $(ANDROID_PATH)/platform/prebuilt/linux-x86/toolchain/arm-eabi-4.2.1/bin

CFLAGS = -I$(ANDROID_PATH)/platform/development/ndk/build/platforms/android-4/arch-arm/usr/include
LDFLAGS = -L$(ANDROID_PATH)/platform/development/ndk/build/platforms/android-4/arch-arm/usr/lib

CC = $(TOOLCHAIN_PATH)/arm-eabi-gcc
LD = $(TOOLCHAIN_PATH)/arm-eabi-ld

all:
        $(CC) -O2 -fPIC -c $(CFLAGS) $(LDFLAGS) Add.c -o libadd.o
        $(LD) -o libadd.so libadd.o -shared
clean:
        rm -rf libadd.o
        rm -rf libadd.so
```

编译成动态库 libadd.so，用来模拟第三方动态库

2、封装原动态库

原动态库文件不包含 jni 接口需要的信息，所以我们需要对其进行封装，所以我们的需求是：将 libadd.so 里面的 API 封装成带 jni 接口的动态库 libaddjni.so，步骤如下：

A) 新建一个 Android 工程：com.android.libjni，其默认生成的 APK 运行结果是：



B) 在这个工程中新建一个 java 类，其中包含封装后的 API 函数，生成 jni 格式的头文件（.h）

LibJavaHeader.java：

```
package com.android.libjni;

public class LibJavaHeader {
    public native int Add(int x, int y);
}
```

在命令行中进入该工程的 src 目录

运行如下命令

> javac com\android\libjni\LibJavaHeader.java

> javah com.android.libjni.LibJavaHeader

此时在 src 目录下生成了一个名为 com_android_libjni_LibJavaHeader.h 的头文件，其内容如下：

com_android_libjni_LibJavaHeader.h:

```
/* DO NOT EDIT THIS FILE - it is machine generated */
#include <jni.h>
/* Header for class com_android_libjni_LibJavaHeader */

#ifndef _Included_com_android_libjni_LibJavaHeader
#define _Included_com_android_libjni_LibJavaHeader
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:     com_android_libjni_LibJavaHeader
 * Method:    Add
 * Signature: (II)I
 */
JNIEXPORT jint JNICALL Java_com_android_libjni_LibJavaHeader_Add
```

```
  (JNIEnv *, jobject, jint, jint);

#ifdef __cplusplus
}
#endif
#endif
```

C) 在前面建立的 Android 工程的 libjni.java 中添加测试 java 代码如下：

```
package com.android.libjni;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class libjni extends Activity {
    /** Called when the activity is first created. */
    @Override

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        int iResult = new LibJavaHeader().Add(5, 6);
        TextView tv = new TextView(this);
        tv.setText(Integer.toString(iResult));
        setContentView(tv);
    }

    static
    {
        System.loadLibrary("addjni");
    }
}
```

3、编写库的封装函数 libaddjni.c

根据前面生成的 com_android_libjni_LibJavaHeader.h 文件，编写 libaddjni.c，用来生成 libaddjni.so

libaddjni.c:

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

#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:     com_android_libjni_LibJavaHeader
```

```
 * Method:    Add
 * Signature: (II)I
 */
JNIEXPORT jint JNICALL Java_com_android_libjni_LibJavaHeader_Add
  (JNIEnv *pJE, jobject jo, jint jiX, jint jiY)
{
        return Add(jiX, jiY);
}

#ifdef __cplusplus
}
#endif
```

在 Android 源码的 platform/external 新建一个文件夹 libaddjni，将 libaddjni.c、Add.h 放入该目录，创建 Android.mk，内容如下：

```
LOCAL_PATH:= $(call my-dir)
include $(CLEAR_VARS)

LOCAL_PRELINK_MODULE := false

LOCAL_SRC_FILES := libaddjni.c

LOCAL_SHARED_LIBRARIES := libadd

LOCAL_C_INCLUDES := $(JNI_H_INCLUDE)

LOCAL_MODULE := libaddjni
include $(BUILD_SHARED_LIBRARY)
```

然后将 libadd.so 拷贝到 out/target/product/generic/obj/lib 和 out/target/product/generic/system/lib 目录中，在 platform 目录下运行 make libaddjni 编译之后会产生一个 out/target/product/generic/obj/lib/libaddjni.so

4、开始测试

#adb push libaddjni.so /system/lib
#adb push libadd.so /system/lib
然后在 eclipse 运行 android 工程 libjni，可以看到结果是：