一、关于NDK

Android NDK全称：Native Development Kit，即本地开发包。

1、NDK是一系列工具的集合。

NDK提供了一系列的工具，这些工具对开发者的帮助是巨大的。

它们能帮助开发者快速开发C（或C++）的动态库，并能自动将so库和java应用一起打包成apk。

NDK集成了交叉编译器，并提供了相应的mk文件隔离CPU、平台、ABI等差异，

开发人员只需要简单修改mk文件（指出“哪些文件需要编译”、“编译特性要求”等），就可以创建出so库。

NDK可以自动地将so和Java应用一起打包，极大地减轻了开发人员的打包工作。

2、NDK提供了一份稳定、功能有限的API头文件声明。

Google明确声明该API是稳定的，在后续所有版本中都稳定支持当前发布的API。

从该版本的NDK中看出，这些API支持的功能非常有限，包含有：

C标准库（libc）、标准数学库（libm）、压缩库（libz）、Log库（liblog）。

二、NDK实例的实现

对于Windows环境下NDK的开发，如果使用的NDK是r7之前的版本，必须要安装Cygwin才能使用NDK；

从r7开始，Google的提供了一个ndk-build.cmd的脚本，可以直接用它编译，而不需要使用Cygwin了。

只需要为Eclipse Android工程添加一个Builder，就能让Eclipse自动编译NDK。

本文是讲述NDK-r8e下的实现实例。

下面是在windows下为NDK配置自动编译的builder的过程（对于Linux，只需将ndk-build.cmd修改为ndk-build就可以了）。

（1）先下载安装NDK

下载地址：<http://developer.android.com/sdk/ndk/index.html>

下载后解压缩就可以用了。

（2）新建工程和jni文件夹

打开Eclipse，新建或导入一个Android工程（我的取名为NdkTest），在工程目录下新建jni文件夹；

该jni文件夹就用来保存NDK需要编译的文件代码等。

（3）新建并配置一个Builder：

（a）Project->Properties->Builders->New，新建一个Builder。

（b）在弹出的【Choose configuration type】对话框，选择【Program】，点击【OK】：

（c）在弹出的【Edit Configuration】对话框中，配置选项卡【Main】。

在“Name“中输入新builders的名称（我取名为Ndk\_Builder）。

在“Location”中输入nkd-build.cmd的路径。

（我的是E:\adt-bundle-windows-x86-20130522\ndk-r8e\ndk-build.cmd，

根据各自的ndk路径设置，也可以点击“Browser File System…”来选取这个路径）。

在“Working Diretcoty”中输入项目工程的路径${workspace\_loc:/project}

（我的是${workspace\_loc:/NdkTest}，也可以点击“Browse Workspace”来选取本工程目录）。

（d）【Edit Configuration】对话框中，配置选项卡【Refresh】。

勾选“Refresh resources upon completion”，

勾选“The entire workspace”，

勾选“Recuresively include sub-folders”。

（e）【Edit Configuration】对话框中，配置选项卡【Build options】。

勾选“After a “Clean””，

勾选“During manual builds”，

勾选“During auto builds”，

勾选“Specify working set of relevant resources”。

点击“Specify Resources…”

勾选本工程的“jni“目录，点击”finish“。

点击“OK“，完成配置。到这里Eclipse就能自动调用NDK编译jni目录下的C/C++代码了。

（4）新建JniClient.java

在工程中新建一个JniClient.java（为了调用C/C++代码），保存在MainActivity.java目录下；其内容如下：

[java] view plaincopy

package com.example.ndktest;

public class JniClient {

static public native String HelloWorld();

}

（5）生成JniClinet.class文件

用cmd命令定位到JniClient.java所在目录，输入“javac JniClient.java“后回车，生成JniClinet.class文件。

（如果是用的Eclipse建的工程，在bin\classes\com\example\ndktest目录下就已经有JniClinet.class文件了）。

（6）生成C++头文件

将JniClinet.class拷贝到bin\classes\com\example\ndktest目录，将cmd命令定位到bin\classes目录，

输入”javah com.example.ndktest.JniClient“后回车，在bin\classes目录下就生成了C++头文件了。

com\_example\_ndktest\_JniClient.h的文件内容如下：

[cpp] view plaincopy

/\* DO NOT EDIT THIS FILE - it is machine generated \*/

#include <jni.h>

/\* Header for class com\_example\_ndktest\_JniClient \*/

#ifndef \_Included\_com\_example\_ndktest\_JniClient

#define \_Included\_com\_example\_ndktest\_JniClient

#ifdef \_\_cplusplus

extern "C" {

#endif

/\*

\* Class: com\_example\_ndktest\_JniClient

\* Method: HelloWorld

\* Signature: ()Ljava/lang/String;

\*/

JNIEXPORT jstring JNICALL Java\_com\_example\_ndktest\_JniClient\_HelloWorld

(JNIEnv \*, jclass);

#ifdef \_\_cplusplus

}

#endif

#endif

（7）新建一个Android.mk文件

在jni目录下新建一个Android.mk文件，其内容如下（详细的语法以后再另外解释）：

[plain] view plaincopy

LOCAL\_PATH := $(call my-dir)

include $(CLEAR\_VARS)

LOCAL\_MODULE := NdkTest

LOCAL\_SRC\_FILES := com\_example\_ndktest\_JniClient.c

include $(BUILD\_SHARED\_LIBRARY)

（8）创建C++源文件

将com\_example\_ndktest\_JniClient.h拷贝到本工程的jni目录下；

然后新建一个com\_example\_ndktest\_JniClient.c文件，用来实现头文件中函数，其内容如下：

[cpp] view plaincopy

#include "com\_example\_ndktest\_JniClient.h"

#include <stdlib.h>

#include <stdio.h>

#ifdef \_\_cplusplus

extern "C"

{

#endif

JNIEXPORT jstring JNICALL Java\_com\_example\_ndktest\_JniClient\_HelloWorld

(JNIEnv \*env, jclass arg)

{

jstring str = (\*env)->NewStringUTF(env, "HelloWorld from JNI !");

return str;

}

#ifdef \_\_cplusplus

}

#endif

编辑并保存后，可以看到obj/local/armeabi目录下将自动生成libNdkTest.so库。

（9）用java调用c函数

修改MainActivity.java，完成对JniClient.java中函数的调用：

[java] view plaincopy

package com.example.ndktest;

import android.os.Bundle;

import android.app.Activity;

//remove for NdkTest

//import android.view.Menu;

//add for NdkTest

import android.widget.TextView;

public class MainActivity extends Activity {

//add for NdkTest

static {

System.loadLibrary("NdkTest");

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

//remove for NdkTest

//setContentView(R.layout.activity\_main);

//add for NdkTest

String str = JniClient.HelloWorld();

TextView tv = new TextView(this);

tv.setText(str);

setContentView(tv);

}

//remove for NdkTest

/\*@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.main, menu);

return true;

}\*/

}

（10）运行测试

运行，在模拟器中可以看到界面输出来自com\_example\_ndktest\_JniClient.c 文件中的“HelloWorld from JNI ! “。