

Android_Jni 操作指南

近日小弟闻听 android 可以调用 c/c++代码, 所以很是好奇, 就开始研究 Jni。在 android JNI 叫 NDK (Native Development Kit), 需要下载几个工具, 下面我给大家按步骤的说明 Jni 的使用方法:

一、准备工作

Cygwin <http://www.cygwin.com/setup.exe>

android-ndk-1.6_r1 http://dl.google.com/android/ndk/android-ndk-1.6_r1-windows.zip

CDT <http://dldx.csdn.net/fd.php?i=578218968285704&s=070a0643ccfb0a3aade962e99302e6ef>

Eclipse

二、安装 Cygwin

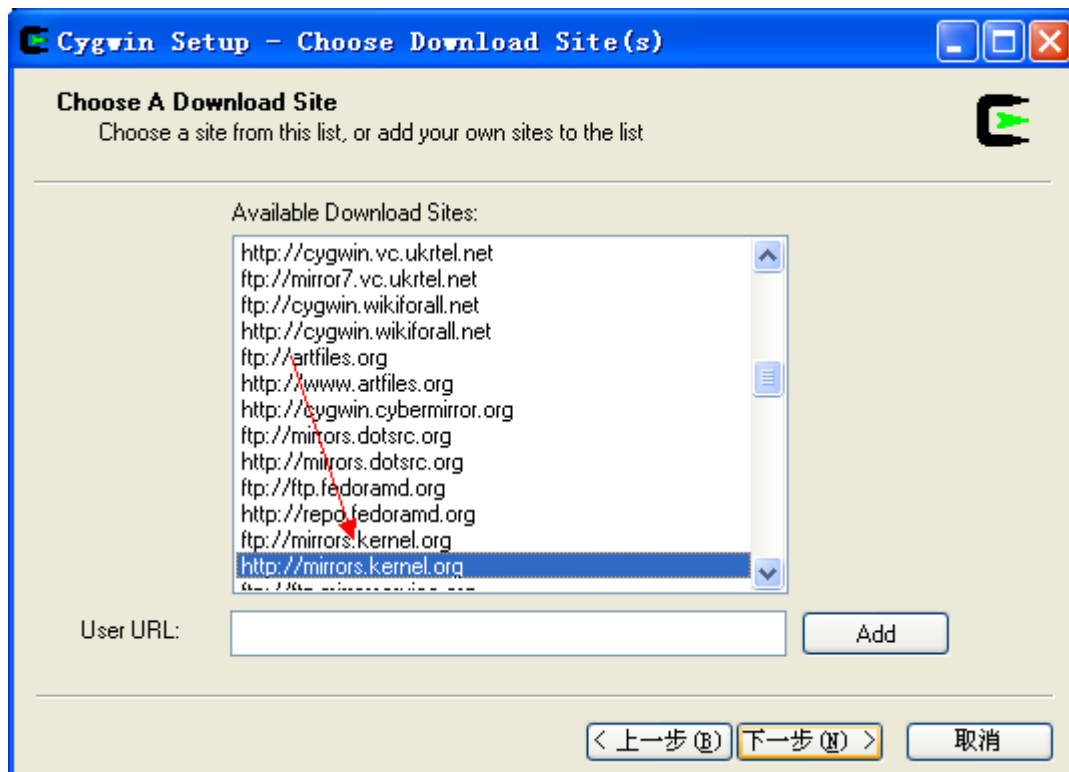
我们采用 Cygwin 去编译 C、C++代码

1. 从上面指定网址下载到 Cygwin

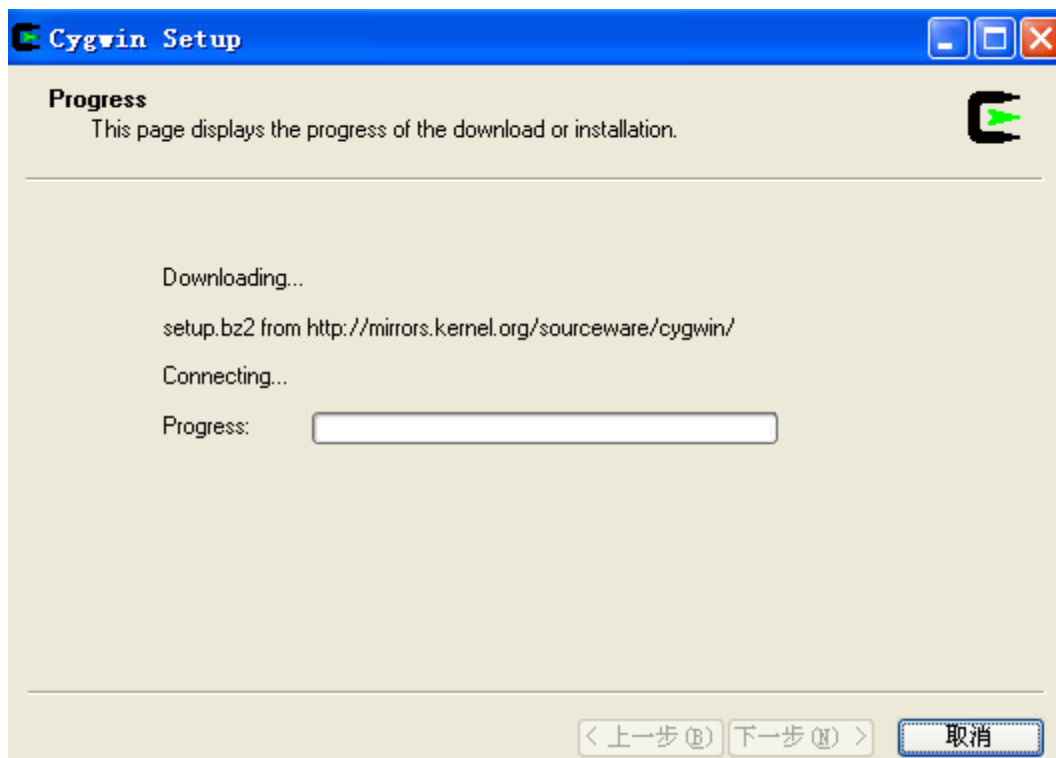


2. 点击安装 因为安装软件很容易 在这个我只介绍几个关键不步骤 (没介绍的就按照默认点下一步就可以了)。

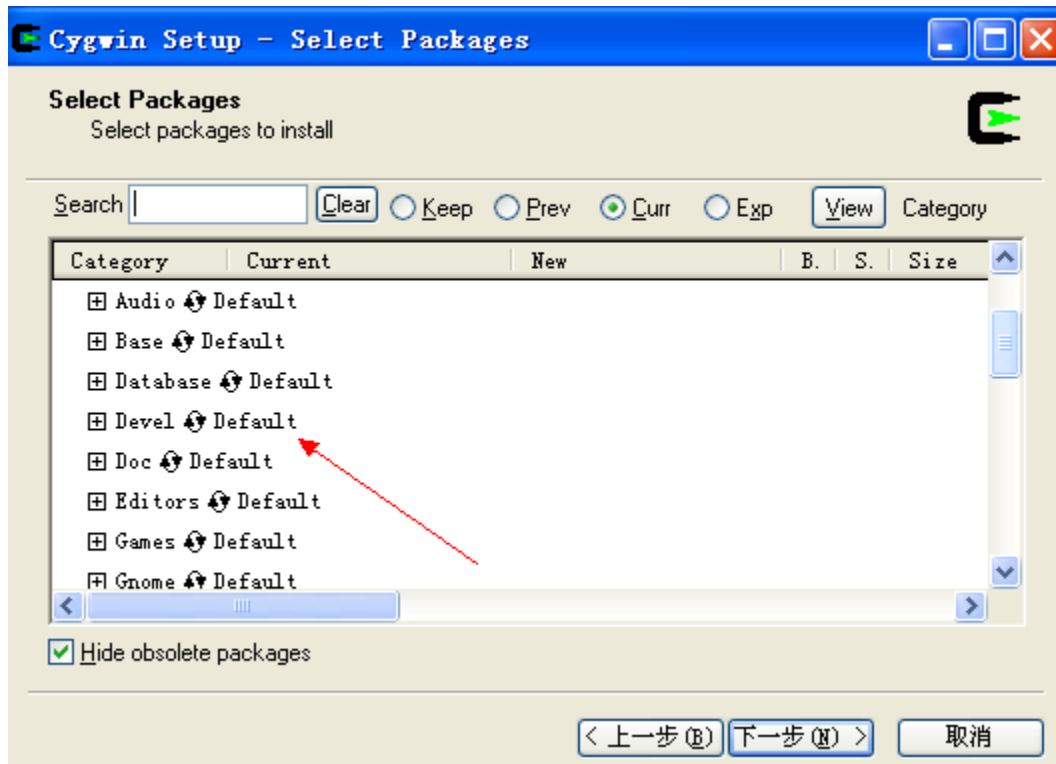




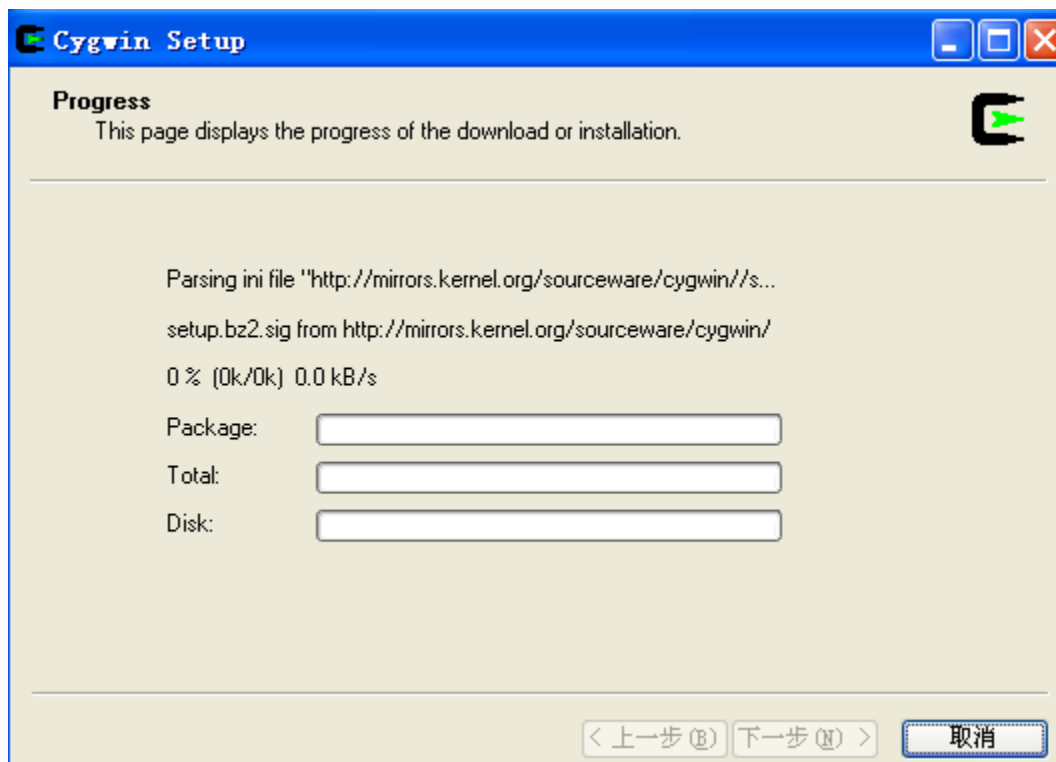
没有这个地址 可以 Add 进去



这一步需要很长时间（要耐心啊）



将箭头指向的 Devel 点成 install 然后下一步



这一个窗口需要经过几个阶段 大家耐心等待 我装的时候花了快一天啊 中间反反复复错一几次，如果出现找不到的情况换个镜像地址，我试了两个终于成功啦。

最后直接下一步 点击完成 Cygwin 就算安装完成



桌面上会出现这样一个图标

点击进去 随便试验几个命令看安装成功没 如和下面一样就表明安装成功

```
Administrator@VUICKI ~
$ gcc -v
Using built-in specs.
Target: i686-pc-cygwin
Configured with: /gnu/gcc/releases/respins/4.3.4-3a/gcc4-4.3.4-3/src/gcc-4.3.4/c
onfigure --srcdir=/gnu/gcc/releases/respins/4.3.4-3a/gcc4-4.3.4-3/src/gcc-4.3.4
--prefix=/usr --exec-prefix=/usr --bindir=/usr/bin --sbindir=/usr/sbin --libexec
dir=/usr/lib --datadir=/usr/share --localstatedir=/var --sysconfdir=/etc --infod
ir=/usr/share/info --mandir=/usr/share/man --with-gmp=/usr --with-mpfr=/usr --enable-b
ootstrap --enable-version-specific-runtime-libs --with-slibdir=/usr/bin --libexe
cdir=/usr/lib --enable-static --enable-shared --enable-shared-libgcc --disable-_
_cxa_atexit --with-gnu-ld --with-gnu-as --with-dwarf2 --disable-sjlj-exceptions
--enable-languages=ada,c,c++,fortran,java,objc,obj-c++ --disable-symvers --enabl
e-libjava --program-suffix=-4 --enable-libgomp --enable-libssp --enable-libada -
--enable-threads=posix --with-arch=i686 --with-tune=generic --enable-libgcj-subli
bs CC=gcc-4 CXX=g++-4 CC_FOR_TARGET=gcc-4 CXX_FOR_TARGET=g++-4 GNATMAKE_FOR_TARG
ET=gnatmake GNATBIND_FOR_TARGET=gnatbind --with-ecj-jar=/usr/share/java/ecj.jar
Thread model: posix
gcc version 4.3.4 20090804 (release) 1 (GCC)

Administrator@VUICKI ~
$ make -v
GNU Make 3.81
```

三、安装 NDK

1. 下载 android-ndk-1.6_r1 （我使用的是这个版本的）
2. 将 android-ndk-1.6_r1 解压 具体位置你可以自己定 我是放在了 E:\android 底下
3. 打开 Cygwin 进入到 android-ndk-1.6_r1 的根目录下 具体指令如下

```
cd cygdrive/e/android/android-ndk-1.6_r1/
```

4. 输入 build/host-setup.sh 如果出现下面的信息表明安装成功

```
Administrator@f06a14528e0c4ae /cygdrive/d/androidsdktools/android-ndk-1.5_r1
$ build/host-setup.sh
Detecting host toolchain.

CC      : compiler check ok (gcc)
LD      : linker check ok (gcc)
CXX     : C++ compiler check ok (g++)
Generate : out/host/config.mk
Toolchain : Checking for arm-eabi-4.2.1 prebuilt binaries

Host setup complete. Please read docs/OVERVIEW.TXT if you don't know what to do.
```

这里需要说明一下 如果出现让你 cd NDKROOT ... 你指需要执行

export NDKROOT=E:\android\android-ndk-1.6_r1 也就是说是你 NDK 的路径

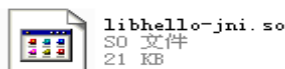
5. NDK 安装完成之后就让我们先编译一下 NDK sample 里面给的例子吧

首先进入到 ndk 根目录 然后 make APP=hello-jni -B

```
Administrator@UICKI /cygdrive/e/android/android-ndk-1.6_r1
$ make APP=hello-jni -B
Android NDK: Building for application 'hello-jni'
Compile thumb      : hello-jni <= apps/hello-jni/project/jni/hello-jni.c
SharedLibrary      : libhello-jni.so
Install            : libhello-jni.so => apps/hello-jni/project/libs/armeabi
```

如果出现上图的信息就表明编译成功

编译完成后会生成一个.so 共享库文件



四. 编写第一个 NDK 程序

由于是操作文档，所以在这里只是给大家简要的介绍一下概念

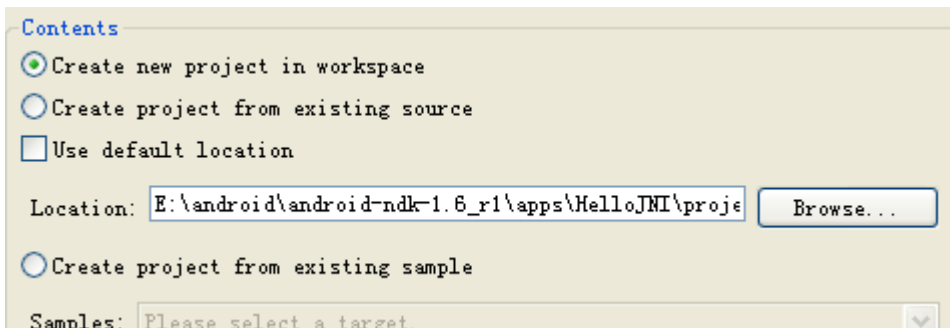
Android NDK 开发是使用 JNI 对本地的方法或者库来将 Java 程序和 Native 程序结合起来。JNI 明确分开了 Java 和本地代码（C/C++）的执行，结构上很清晰。

NDK 程序需要一下步骤

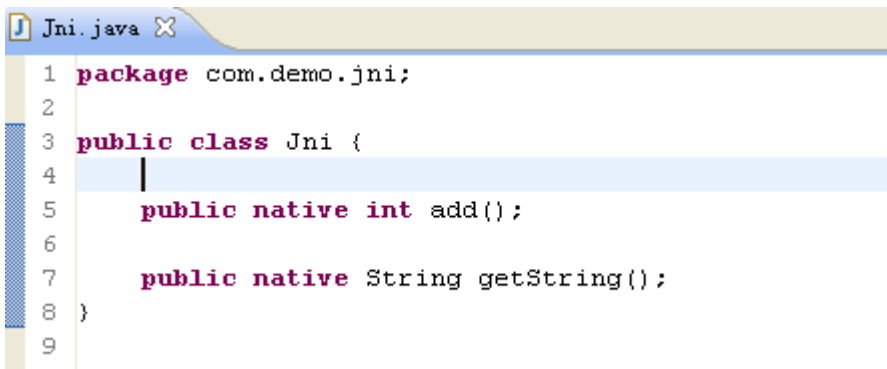
- (1) JNI 接口的设计
- (2) 使用 C/C++实现本地方法
- (3) 生成动态链接库.so 文件
- (4) 将动态链接库复制到 Java 工程，运行 Java 工程即可。

1. JNI 接口的设计

- (1) android-ndk-1.6_r1 下有一个 apps 专门放 NDK 工程，所以我们将工程也放在这个文件夹下，具体目录为 android-ndk-1.6_r1\apps\firstJNI\project
其中 firstJNI 为工程的文件夹 project 放 Java 工程和本地代码
- (2) 创建 Android 工程，指定目录到 android-ndk-1.6_r1\apps\NewJNI\project



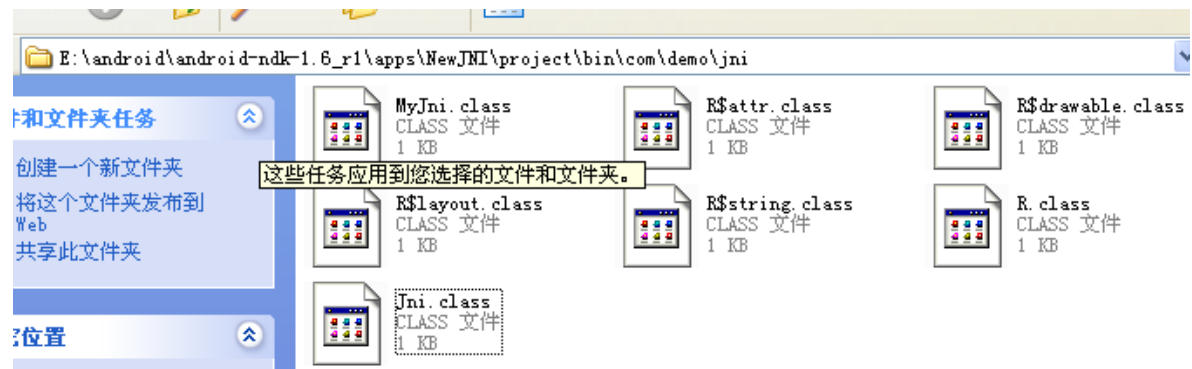
(3) 写 Java 类 Jni



- (4) 编译 Jni 将 Jni.java 文件放到工程的 bin 目录下 然后 cmd 打开 doc 进入到该工程的 bin 目录下 输入 “javac Jni.java” 生成 Jni.class 文件

```
E:\android\android-ndk-1.6_r1\apps\NewJNI\project\bin>javac Jni.java
E:\android\android-ndk-1.6_r1\apps\NewJNI\project\bin>
```

(5) 复制上一步生成的 Jni.class 文件到下面的目录下 覆盖以前的 Jni.class



(6) 进入到工程的 bin 目录下 输入 “javah -jni com.demo.jni.Jni”

```
E:\android\android-ndk-1.6_r1\apps\NewJNI\project\bin>javah -jni com.demo.jni.Jni
```

此时会在当前目录下生成 com_demo_jni_Jni.h 文件 如下图所示

```
E:\android\android-ndk-1.6_r1\apps\NewJNI\project\bin 的目录
2010-09-24 15:09 <DIR> .
2010-09-24 15:09 <DIR> ..
2010-09-24 15:05 2,096 classes.dex
2010-09-24 15:05 <DIR> com
2010-09-24 15:09 611 com_demo_jni_Jni.h
2010-09-24 15:05 7,175 HelloWorldJNI.apk
2010-09-24 15:05 4,454 resources.ap_
4 个文件 14,336 字节
3 个目录 33,013,477,376 可用字节
E:\android\android-ndk-1.6_r1\apps\NewJNI\project\bin>
```

打开代码如下:

这个是由系统自动生成的文件

```

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

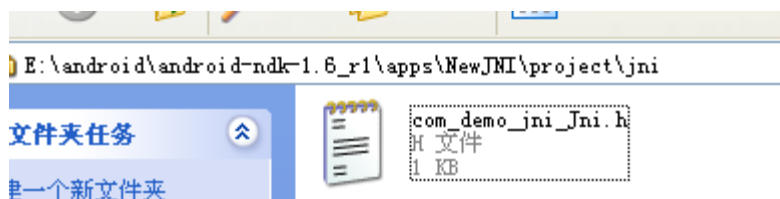
#ifndef _Included_com_demo_jni_Jni
#define _Included_com_demo_jni_Jni
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:      com_demo_jni_Jni
 * Method:     add
 * Signature:  ()I
 */
JNIEXPORT jint JNICALL Java_com_demo_jni_Jni_add
    (JNIEnv *, jobject);

/*
 * Class:      com_demo_jni_Jni
 * Method:     getString
 * Signature:  ()Ljava/lang/String;
 */
JNIEXPORT jstring JNICALL Java_com_demo_jni_Jni_getString
    (JNIEnv *, jobject);

#ifdef __cplusplus
}
#endif
#endif

```

- (7) 在工程的根目录下新建一个 jni 文件夹 将生成的 com_demo_jni_Jni.h 复制到该文件夹底下。该 jni 文件夹专门用来存储 C\C++ 文件。



以上完成了 NDK 开发中的 JNI 接口设计，下面我们使用 C\C++ 来完成对这些本地方法的实现。

安装 CDT 是可选的 只为了方便

CDT 就是一个插件 按照上面给的地址下载好后解压 会出现两个文件夹 你只需要将这两个文件夹里面的东西分别复制到 Eclipse 里的相应目录即可 然后重启你的 Eclipse 就可以在 Eclipse 写 C、C++ 代码了。

- (8) 我不介绍怎么用 Eclipse 创建 C 工程 大家自也可以用记事本写 我现在介绍正题 在我们刚才新建的 jni 文件夹底下创建一个 com_demo_jni_Jni.c 文件，用来实现 Jni.java 里面声明但为实现的原生方法。实现代码如下：

```

#include <stdio.h>
#include <stdlib.h>
#include "com_demo_jni_Jni.h"

int add(){

    int x,y;
    x=100;
    y=100;
    return x+y;
}

/*
 * Class:      com_demo_jni_Jni
 * Method:     add
 * Signature:  ()I
 */
JNIEXPORT jint JNICALL Java_com_demo_jni_Jni_add
(JNIEnv *env, jobject thiz){
    return add();
}

/*
 * Class:      com_demo_jni_Jni
 * Method:     getString
 * Signature:  ()Ljava/lang/String;
 */
JNIEXPORT jstring JNICALL Java_com_demo_jni_Jni_getString
(JNIEnv *env, jobject thiz){

    (*env)->NewStringUTF(env,"Hello world!!!! Jni");
}

```

有关 C 和 C++的语法在这里就不介绍了

我们已经完成了 C\C++代码的书写

- (9) jni 通过调用动态链接库（windows 下是 dll ， linux 下是 so）来实现 C\C++方法的调用，
因此现在我们就需要将上面写的 C 代码编译成 SO 文件

编译需要两个文件 Android.mk 和 Application.mk 下面我们介绍一下这两个文件的实现方式

Android.mk

存放目录为：如下图 Android.mk 文件和你的 C 文件在同一个目录下



让我们来看看 Android.mk 文件的内容吧


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

include $(CLEAR_VARS)

LOCAL_MODULE := NewJNI
LOCAL_SRC_FILES := com_demo_jni_Jni.c

include $(BUILD_SHARED_LIBRARY)
```

LOCAL_PATH := \$(call my-dir)

LOCAL_PATH 表示此时位于工程目录的根目录中 (call my-dir)返回当前目录的地址

LOCAL_MODULE := NewJNI

LOCAL_MODULE 用来区分 Android.mk 中的每一个模块。文件名必须为一 不能有空格 这个是你以后库文件生成的文件名 编译器会自动给你加上前缀 lib 和后缀.so.

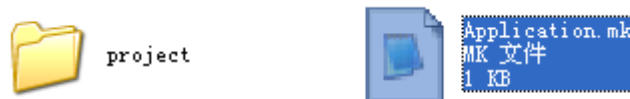
LOCAL_SRC_FILES := com_demo_jni_Jni.c

include \$(BUILD_SHARED_LIBRARY)

BUILD_SHARED_LIBRARY 的意思就是建立一个动态共享库

Application.mk

Application.mk 的存放位置在如下图 和工程是在一级上



让我们看看里面的内容

```
APP_PROJECT_PATH := $(call my-dir)/project
APP_MODULES := NewJNI
```

具体意思不解释 这个 APP_MODULES 和 Android.mk 里面的是一致的

(10) 编译 C\C++代码

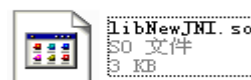
现在我们去编译这些代码，让其生成.SO 文件让 Java 来调用

首先启动 Cygwin，进入 NDK 目录，输入 make APP=NewJNI 出现如下画面表示编译成功

```
ava_com_demo_jni_Jni_getString' was here
make: *** [out/apps/NewJNI//objs/NewJNI/com_demo_jni_Jni.o] Error 1

Administrator@UICKI /cygdrive/e/android/android-ndk-1.6_r1
$ make APP=NewJNI
Android NDK: Building for application 'NewJNI'
Compile thumb : NewJNI <= apps/NewJNI/project/jni/com_demo_jni_Jni.c
SharedLibrary : libNewJNI.so
Install : libNewJNI.so => apps/NewJNI/project/libs/armeabi

Administrator@UICKI /cygdrive/e/android/android-ndk-1.6_r1
$
```



此时会在工程目录下 “lib\armeabi” 目录中出现

文件

通过上面的步骤我们已经完成了编译 C\C++代码并生成库文件

(11) 最后在我们的 Android 工程中使用

在 Activity 类中加入如下代码

```
package com.demo.jni;

import android.app.Activity;

public class MyJni extends Activity {

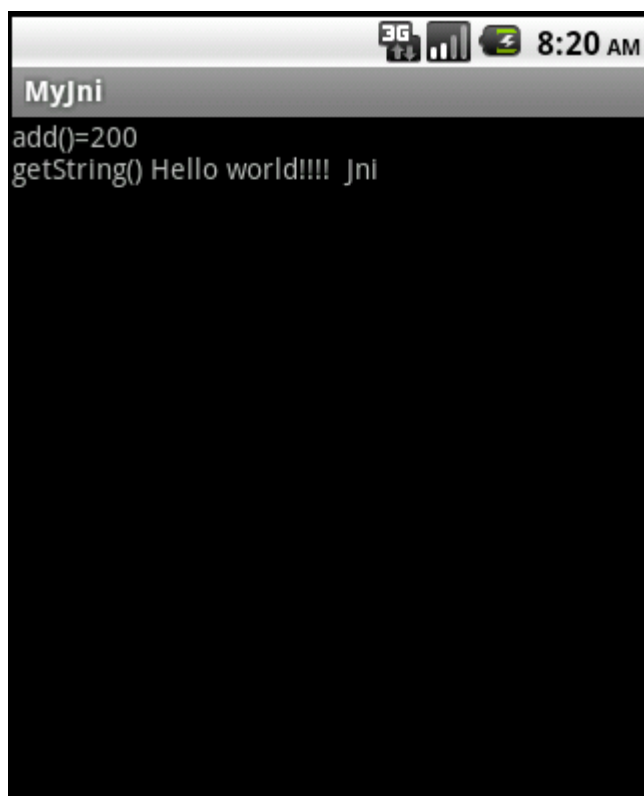
    private TextView textView;
    static{
        System.loadLibrary("NewJNI");
    }
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Jni jni = new Jni();
        int result = jni.add();

        textView = (TextView) findViewById(R.id.tv);
        textView.setText("add()="+result+"\ngetString() = "+jni.getString());
    }
}
```

其中 System.loadLibrary("NewJNI");表示装在动态库 libNewJNLso

运行该工程 效果图如下，成功的调用了 libNewJNLso 中的 add()和 getString()方法。



自此我们完成了一个简单 NDK 的开发。

注：如有纰漏的地方请大家邮件通知我 piziyuyu@163.com 非常感谢