Ubuntu 10.04安裝android NDK

1、下載NDK包，下載地址：(Only suitable for ARM)

<http://www.ideasandroid.com/android/sdk/android-ndk-r4-linux-x86.zip>

[http://dl.google.com/android/ndk/android-ndk-r4b-linux-x86.zip](http://www.ideasandroid.com/android/sdk/android-ndk-r4-linux-x86.zip)2、解壓，我們這裡將解壓到/home/price/android/目錄下。

解壓後目錄結構為：/home/ price/android/android-ndk-r4

3、配置path路徑(此步也可以省略，只是以後命令要加上路徑/home/myleft/android-ndk-r4b/ndk-build)：

命令列執行~/.bashrc，在檔末尾加入如下內容(包含Android SDK)：

export ANDROID\_SDK=~/android/android-sdk

export ANDROID\_NDK=~/android/android-ndk-r4b

export PATH=$PATH:$ANDROID\_SDK/platform-tools:$ANDROID\_NDK

保存後，重啟機器。至此 android NDK安裝完畢。

4、產生JNI的header檔：

首先用Java產生JNI的函式介面，如命名為*CalculatorJni*，此時會有*CalculatorJni.class*的二進位檔在以下路徑：

*price@price:~/Document/Calculator/bin/classes/com/example/calculator$ ls*

*CalculatorJni.class …*

*price@price:~/Document/Calculator/bin/classes$ javah -classpath ./ com.example.calculator.CalculatorJni*

*price@price:~/Document/Calculator/bin/classes$ ls*

*com\_example\_calculator\_CalculatorJni.h*

5、編譯sample工程：

執行命令 *cd /home/price/android-ndk-r6b/samples/hello-jni* 進入示例專案根目錄，然後執行如下命令：

*ndk-build*

您將看到系統會編譯出*libhello-jni.so*至此示例項目編譯完成，可將hello-jni項目導入ADT進行進一步的測試。

接下來就創建一個專案來測試一下，步驟如下：

(1) 使用Eclipse創建一個Android項目，名字為“*HelloNDKJNI*”，Build Target設置為“Android 2.2”，Application Name設置為“HelloNdkJni”，Package Name設置為“com.gcx.ndkjni”，Create Activity設置為“.HelloNdkJni”，Min SDK Version設置為“8”；

Caution: If the source code has already exist but eclipse fails to open it (The reason why it is required to use eclipse is to check if all of the parameters in this project are correct so that it can be compiled successfully by the NDK in Android FS). Try to modify this project as below:

1. copy *.project*, *.classpath* and *default.properties* files from other projects which eclipse can open successfully.

2. Create a folder named “*jni*” to store the native code. The *LOCAL\_MODULE* variable of *Android.mk* in this folder should be identical to the *LOCAL\_JNI\_SHARED\_LIBRARIES* of *Android.mk* which exists in the path the *AndroidManifest.xml* file does.

(2) 接下來創建C語言庫，在Eclipse的Package Explore裡面的*HelloNDKJNI*專案下創建目錄“*jni*”，並在該目錄下創建兩檔“Android.mk”和“hello-ndk-jni.c”, Android.mk檔內容如下：

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

*include $(CLEAR\_VARS)*

*LOCAL\_MODULE := hello-ndk-jni*

*LOCAL\_SRC\_FILES := hello-ndk-jni.c*

*include $(BUILD\_SHARED\_LIBRARY)*

hello-ndk-jni.c檔內容如下：

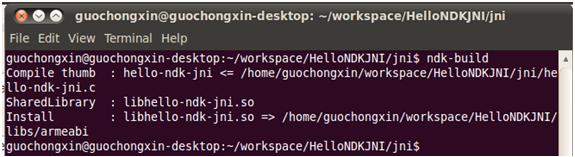
*#include <string.h>*

*#include <jni.h>*

*jstring Java\_com\_gcx\_ndkjni\_HelloNdkJni\_stringFromNDKJNI( JNIEnv\* env,jobject thiz )*

*{ return (\*env)->NewStringUTF(env, "Hello from NDK JNI !"); }*

(3-1) 進入到創建的*HelloNDKJNI*專案中的*jni*目錄，執行命令*ndk-build*，此時會在專案中生成*libs*和*obj*目錄，並在裡面生成相應的檔，運行結果如下圖所示：此方式僅用於ARM

[](http://photo.blog.sina.com.cn/showpic.html#blogid=3e4774e30100mjug&url=http://s15.sinaimg.cn/orignal/3e4774e3g928f53d45c2e)

(3-2) For x86, You can get the *SimpleJNI* code in ***Android/development/samples/SimpleJNI*.**

[Compile]

01. First, you should download all Android code

02. In Android Folder, type *make SimpleJNI*.

Caution 1: Do not let eclipse compile this project since the *R.java* is generated after compilation. Then when the NDK in android FS tries to compile this project, error occurs! At this moment, just delete the *R.java* file first and use make command again.

Caution 2: If you run a clean on the project it should regenerate all the generated java files, namely R. In Eclipse, under the Project menu, is an option build automatically. That would help you build the *R.java* file everytime modifications are made. The Clean... option is also there under Project.

Caution3: The *LOCAL\_PACKAGE\_NAME* variable in *Android.mk* should be same as *SimpleJNI*.

03. *SimpleJNI* is the package name, it will help to compile out application and library.

04. When compiled completely, you will get the file - *simplejni.apk*

[Run]

01. First get the the file - simplejni.apk

02. Start adb

03. Go to *$(Android SDK)/platform-tools/*, and copy APK file to this path.

04. #*adb install simplejni.apk* (install your application)

4) 刷新Eclipse中的Package Explore中的*HelloNDKJNI*項目，此時*obj*和*libs*目錄也添加進去了，在*obj/armeab*i分支下也多了*libhello-ndk-jni.so*檔，*hello-ndk-jni*這個名是根據2-2)步中的*Android.mk*檔中的*LOCAL\_MODULE*決定的，接下來修改*src/com.gcx.ndkjni*分支下的*HelloNdkJni.java*檔，最後的檔內容如下：

*package com.gcx.ndkjni;*

*import android.app.Activity;*

*import android.os.Bundle;*

*import android.widget.TextView;*

*public class HelloNdkJni extends Activity {*

*@Override*

*public void onCreate(Bundle savedInstanceState) {*

*super.onCreate(savedInstanceState);*

*TextView tv =new TextView(this);*

*tv.setText(stringFromNDKJNI());*

*setContentView(tv);*

*}*

*public native String stringFromNDKJNI();*

*static{ System.loadLibrary("hello-ndk-jni"); }*

*}*

## 在使用Eclipse編譯時如果出現[Android make apk error:This attribute must be localized.](http://jjnnykimo.pixnet.net/blog/post/31996155)的錯誤訊息，主要是因為使用的xml檔案中有沒有考慮到多國語系的問題。如下：因為在TextView中屬性text是設定成固定的內容"this will make error"所造成的 a.xml <TextView android:id="@+id/myTextView1" android:text="t**his will make error**" android:layout\_width="fill\_parent"></TextView> 必須改為利用在res/values/string.xml中定義顯示的文字內容"this will ok"後，再將原text改為@string/str\_info即可

res/values/string.xml  
<string name="**str\_info**">**this will ok**</string>  
a.xml  
<TextView android:id="@+id/myTextView1" android:text="@string/**str\_info**" android:layout\_width="fill\_parent"></TextView>

(5) 接下來就可以在模擬器運行程式了，運行前關於模擬器的配置及Run Configurations可以參考

通過這一步驟可生成Android開發者所熟悉的apk檔：TestHelloLib.apk。

(6) 使用emulator及安裝APK之方法

1. 若emulator的名稱為*EmuGingerbread*. 進入$(Android SDK)/tools，並執行#*./emulator –avd EmuGingerbread*

2. Install APK as described before.

集成測試*TestHelloLib.apk*和*libhello.so*。先運行*emulator*並將*TestHelloLib.apk*和*libhello.so*上傳至emulator中。注意要將*libhello.so*上傳到emulator的*/system/lib*目錄，由於該目錄是只讀的，上傳之前先要執行adb remount(To avoid read-only file system)：

|  |
| --- |
| *# adb remount*  *# adb push out/target/product/generic/system/lib/libhello.so /system/lib*  *# adb install TestHelloLib.apk* |

接下來在模擬器選單中可以看到已經安裝的*TestHelloLib*程式，運行即可。由於JNI介面printHello()並沒有作介面上的改動，要驗證其效果需要用Android的logcat工具來查看。運行*adb logcat*可以找到下面的log片斷：

|  |
| --- |
| I/ActivityManager(   48): Starting activity: Intent { action=android.intent.action.MAIN categories={android.intent.category.LAUNCHER} flags=0x10200000 comp={com.test/com.test.TestHelloLib} }  I/ActivityManager(   48): Start proc com.test for activity com.test/.TestHelloLib: pid=174 uid=10024 gids={}  D/dalvikvm(  174): Trying to load lib /system/lib/libhello.so 0x43481c58  D/dalvikvm(  174): Added shared lib /system/lib/libhello.so 0x43481c58  D/dalvikvm(  174): No JNI\_OnLoad found in /system/lib/libhello.so 0x43481c58  D/dalvikvm(  174): +++ not scanning '/system/lib/libwebcore.so' for 'printHello' (wrong CL)  D/dalvikvm(  174): +++ not scanning '/system/lib/libmedia\_jni.so' for 'printHello' (wrong CL)  **D/TestLib (  174): Hello LIB!**  I/ActivityManager(   48): Displayed activity com.test/.TestHelloLib: 806 ms |

這裡包含了呼叫*printHello()*的log資訊，其中"**D/TestLib (  174): Hello LIB!"** 就是printHello()所列印的訊息。至此成功完成Android JNI的實例驗證。

## [Android 開發筆記 - 使用 Android NDK (Native Development Kit)](http://changyy.pixnet.net/blog/post/29437517)

在此使用Ubuntu 10.04 64-bit 環境。首先，從[Android NDK | Android Developers](http://developer.android.com/sdk/ndk/index.html" \t "_blank)下載Android NDK (我在Ubuntu 所以下載 android-ndk-r5c-linux-x86.tar.bz2)並挑選一個地方解壓縮(此例為 ~/android-ndk-r5c)，除此之外要把 Android SDK更新至最新版，如果已經安裝好Android開發環境，更新方式透過 Eclipse -> [Window]-> [Android SDK and AVD Manager] -> [Installed packages] -> [Update All...]

接著要進行NDK環境設置，可查看android-ndk-r5c/docs/INSTALL.html，可以得知Android NDK至少需使用Android 1.5系統，而編譯環境至少是GNU Make 3.8.1，除此之外還需要Nawk或GNU Awk工具，而常見的awk則不適用。

在 Ubuntu 10.04 64-bit desktop 環境上：

$ awk -W version  
GNU Awk 3.1.6  
Copyright (C) 1989, 1991-2007 Free Software Foundation.  
~$ make -version  
GNU Make 3.81  
Copyright (C) 2006 Free Software Foundation, Inc

透過 Eclipse 建立一個 android 2.3.3 project，例如 MyNDK：

Project name: MyNDK  
Build Target: Android 2.3.3/Android Open Source Project/2.3.3/10  
Application name: MyNDK  
Package name: com.example.ndk  
Create Activity: MyNDK  
Min SDK Version: 3

接著在 Eclipse 左邊 Project Explorer 建立jni和libs兩個目錄，位置擺在剛剛的Project 裡(點一下MyNDK，接著按右鍵選 New -> Folder輸入jni，之後在建立libs目錄)：

MyNDK/jni  
MyNDK/libs

在 MyNDK/jni 裡頭新增兩個檔案，分別是 JNI 程式碼和 Android.mk，結構如下：

MyNDK/jni  
MyNDK/jni/Android.mk  
MyNDK/jni/my-jni.c

其中 Android.mk 和 \*.c 可以從android-ndk-r5c/samples/hello-jni/jni 裡頭尋得範例，簡單筆記：

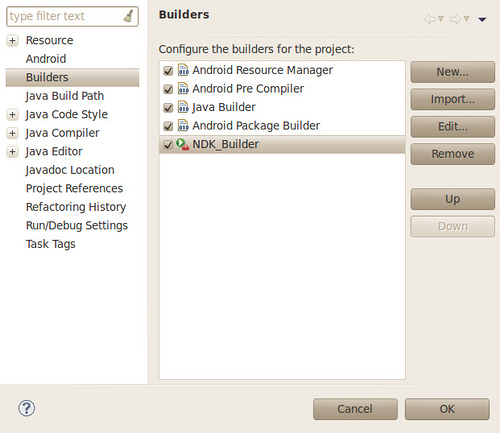
MyNDK/jni/Android.mk (參考 android-ndk-r5c/samples/hello-jni/jni/Android.mk)

LOCAL\_PATH := $(call my-dir)  
include $(CLEAR\_VARS)  
LOCAL\_MODULE    := my-jni  
LOCAL\_SRC\_FILES := my-jni.c  
include $(BUILD\_SHARED\_LIBRARY)

MyNDK/jni/my-jni.c (參考 android-ndk-r5c/samples/hello-jni/jni/hello-jni.c)

#include <string.h>  
#include <jni.h>  
jstring Java\_com\_example\_ndk\_MyNDK\_stringFromJNI( JNIEnv\* env, jobject thiz )  
{return (\*env)->NewStringUTF(env, "Hello from My JNI !");}

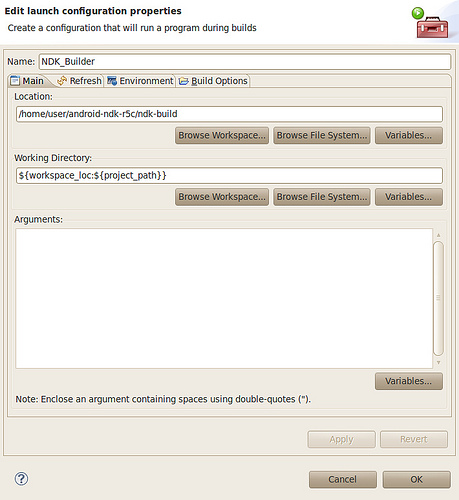
接著要透過 android ndk去編譯 my-jni.c，除了在 Command line 可以切換到 MyNDK 目錄，用 android-ndk-r5c/ndk-build進行編譯外，還可以透過 Eclipse 設定好，讓你每次自動編譯好：

[](http://www.flickr.com/photos/frank921/5911636574/)

在 Eclipse左邊Project Explorer點選 MyNDK，選右鍵 -> Properties -> Builders -> New -> Program

**在 Main 分頁：**

Name: NDK\_Builder  
Location(ndb-build位置): /home/user/android-ndk-r5c/ndk-build  
Working Directory:(project位置): ${workspace\_loc:${project\_path}}

[](http://www.flickr.com/photos/frank921/5911077007/)

**在 Refresh 分頁**：

將 Refresh resources upon completion 打勾，並且在 Specific resources選擇MyNDK/libs目錄

在 Build Options 分頁：

勾選 Allocate Console(necessary for input)、Launch in backgroud、After a "Clean"、During manual builds、During auto builds 和 Specify working set of relevant resources

在 Specific resources 選擇 MyNDK/jni

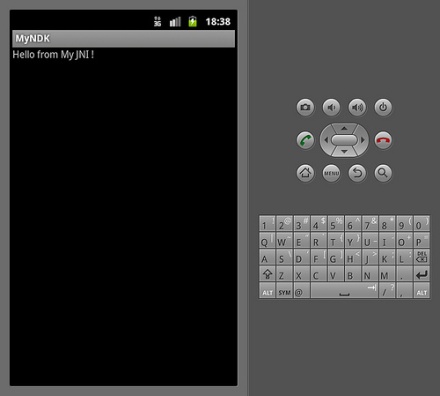
只要編譯或清除時，自動會編譯 jni 的部份，可以使用 Build all(快速鍵Ctrl+B) 試試，可以看到 Console 的輸出：

Compile thumb  : my-jni <= my-jni.c  
SharedLibrary  : libmy-jni.so  
Install : libmy-jni.so => libs/armeabi/libmy-jni.so

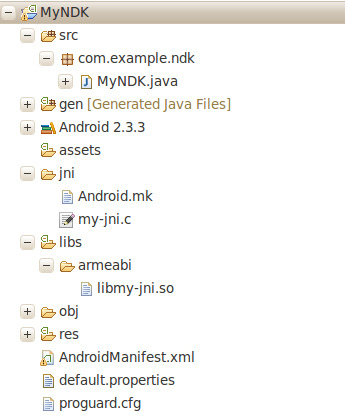
接著，才正式要寫 Java 程式部份，切換到 MyNDK.java：

package com.example.ndk;  
import android.app.Activity;  
import android.os.Bundle;  
import android.widget.TextView;  
public class MyNDK extends Activity {  
    /\*\* Called when the activity is first created. \*/  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        //setContentView(R.layout.main);  
        TextView  tv = new TextView(this);  
        tv.setText( stringFromJNI() );  
        setContentView(tv);  
    }  
    public native String  stringFromJNI();  
    static {System.loadLibrary("my-jni");}  
}

其中特別的是要宣告 jni 使用到的函數，這跟 my-jni.c 裡頭的宣告有關，以及 System.loadLibrary 的部份，跟 jni/Android.mk 設定的編譯結果也有關係，如果都不相符，執行時會出現 exception。最後跑起來的成果：

[](http://www.flickr.com/photos/frank921/5911076917/)

整個 Project 結構：

[](http://www.flickr.com/photos/frank921/5911636534/)