來源 “ <http://blog.xuite.net/lwchafter30/blog/373974237-Android+studio+1.5.1+NDK+JNI%E7%92%B0%E5%A2%83%E5%AE%89%E8%A3%9D%E8%88%87%E5%9F%B7%E8%A1%8C%E5%8E%9F%E7%90%86> ”

查了很多網站，包含Google原廠的NDK說明

好像都沒有很直接的於新版本例如:1.5的Android Studio上面說明如何設定NDK

近期整理了一下，用簡單且完整的方式解說給大家知道

NDK原理的主要六個步驟:

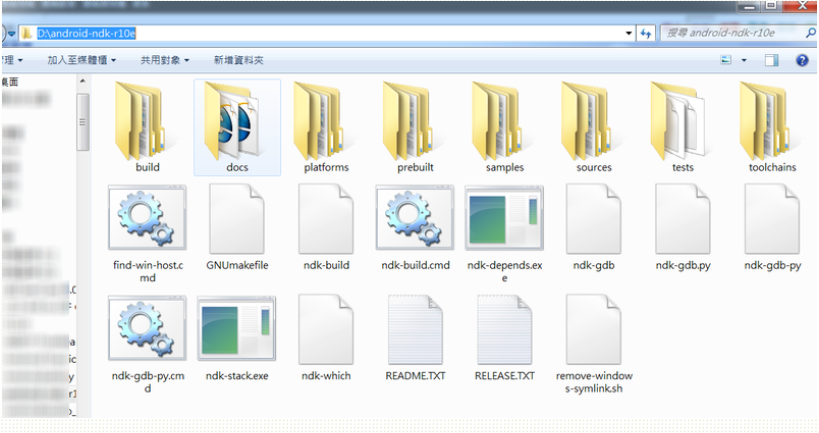
1. 編寫native 方法的 Java類別
2. 將 Java 源代碼編譯成 class 字節碼文件
3. 用 javah -jni 命令生成.h頭文件（javah 是 jdk 自帶的一個命令，-jni 參數表示將 class 中用native 聲明的函數生成 JNI 規則的函數）
4. 用本地代碼實現.h HEAD文件中的函數
5. 將本地代碼編譯成動態庫（Windows：\\*.dll，linux/unix：\\*.so，mac os x：\\*.jnilib）
6. 拷貝動態庫至 java.library.path 本地庫搜索目錄下，並運行 Java 程序

Android Studio環境下載與安裝

<http://developer.android.com/intl/zh-tw/ndk/downloads/index.html>

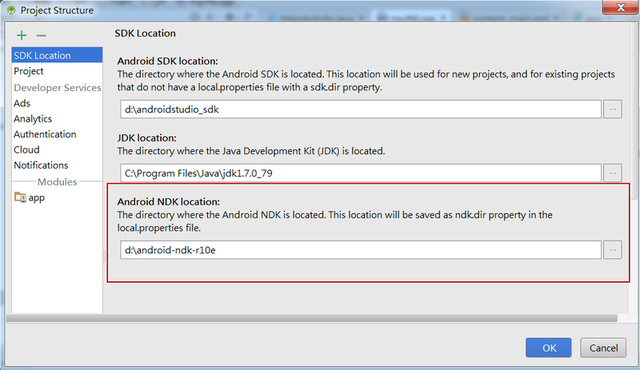
官方下載後，點兩下解壓到安裝檔的目錄下，譬如我是放在D:\android-ndk-r10e

解開後總容量大概3G



接下來到Android Studio中的 File->Project Structure

我就把NDK的位置設定到相對應的目錄就可以了

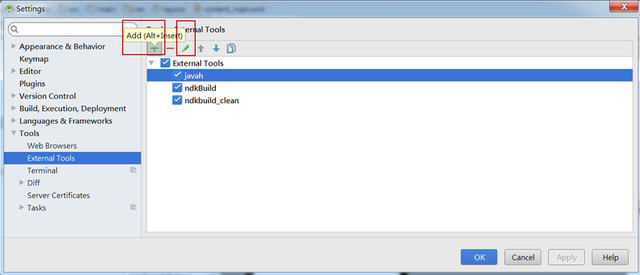
[](http://photo.xuite.net/_pic/lwchafter30/19521903/1112636443_l.jpg/redir)

-------以上完成基本的環境設定-------

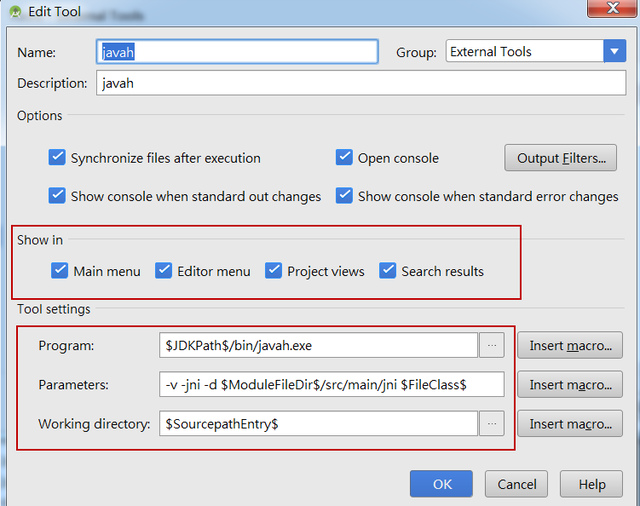
 接下來設定**External Tools**

 File->Settings 裡面 Tools-->External Tools

因為我已經透過 Add按鈕已經增加了三個會用到的指令，所以我是用鉛筆符號Edit做修改

[](http://photo.xuite.net/_pic/lwchafter30/19521903/1112637163_l.jpg/redir)

1.Javah設定

[](http://photo.xuite.net/_pic/lwchafter30/19521903/1112637072_l.jpg/redir)

以下提供複製直接貼上去 注意 這些路徑都是**右上左下的斜線**

**快速複製區**

$JDKPath$/bin/javah.exe

-v -jni -d $ModuleFileDir$/src/main/jni $FileClass$

$SourcepathEntry$

**簡單解說:**

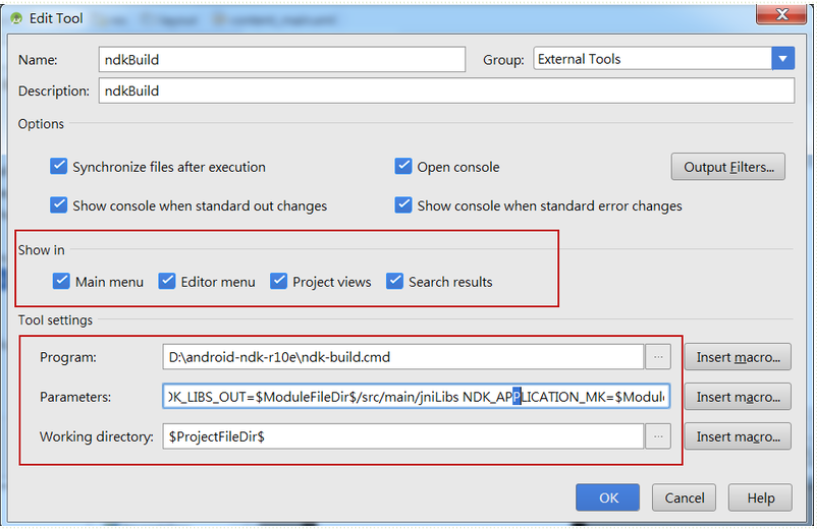
$JDKPath$是環境變數，該行會自動取得你電腦裡面**JDK的javah檔案路徑**

**參數為的是命令提示字元 下 javah OOOXXX的OOOXXX的內容**

主旨就是路徑在Android 專案的src\main\jni下的資源來執行javah的動作

如果有疑問，可以按後面的按鈕 "Insert macro"來查看目前電腦裡環境變數相對應產生的結果

2.NDKBuild設定



**快速複製區**

D:\android-ndk-r10e\ndk-build.cmd

NDK\_PROJECT\_PATH=$ModuleFileDir$/build/intermediates/ndk NDK\_LIBS\_OUT=$ModuleFileDir$/src/main/jniLibs NDK\_APPLICATION\_MK=$ModuleFileDir$/src/main/jni/Application.mk APP\_BUILD\_SCRIPT=$ModuleFileDir$/src/main/jni/Android.mk V=1

$ProjectFileDir$

**簡單解說:**

第一行就是指定NDK目錄裡的build.cmd檔案

第二第三行是同一個欄位，因為太長自動分行

重點是 NDK產生的結果會於src/main/jniLibs裡面XX.so的一些檔案

其他的路徑是因人而異，不過

src/main/jni/Application.mk

src/main/jni/Android.mk

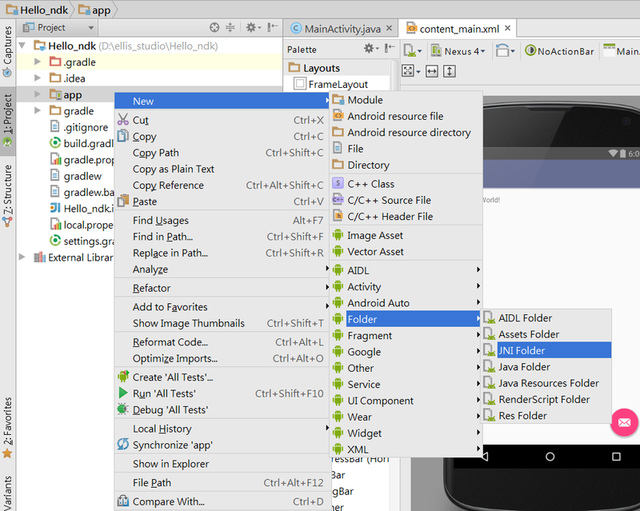
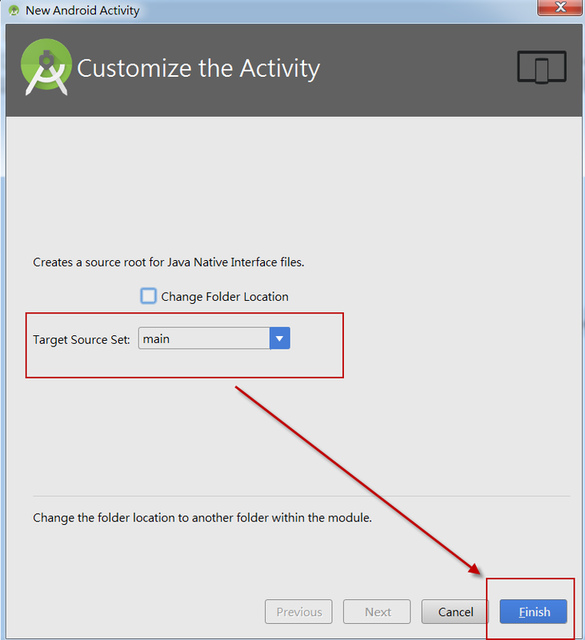
這兩個必要檔案等一下因為實作關係一定會放在這個路徑裡面，所以就這樣定。

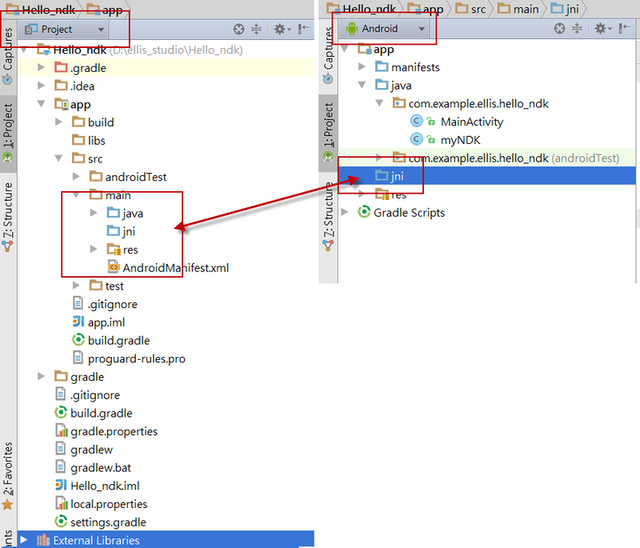
NOTES:  
Use "ndk-build --help" to display the command-line options.  
Use "ndk-build V=1" to display the build messages.  
Use "ndk-build -B" to perform a force re-built.

3.開始針對程式碼實作

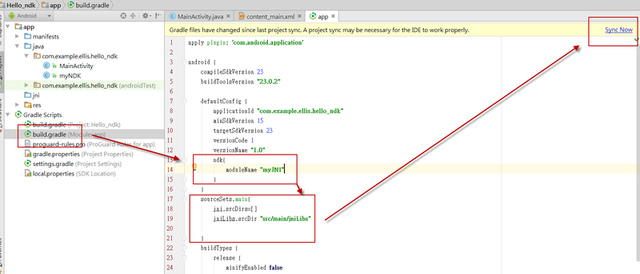
(一)設定JNI相關資源

產生JNI資料夾於src\main\jni

[](http://photo.xuite.net/_pic/lwchafter30/19521903/1112640756_l.jpg/redir)  
[](http://photo.xuite.net/_pic/lwchafter30/19521903/1112641049_l.jpg/redir)

如果成功 可以在兩種視野介面下，找到相對的JNI檔案位置，且必須是藍色的資料夾顏色  
[](http://photo.xuite.net/_pic/lwchafter30/19521903/1112641163_l.jpg/redir)

 (二)針對app的Gradle做整體設定

[](http://photo.xuite.net/_pic/lwchafter30/19521903/1112641284_l.jpg/redir)

**快速複製區**

ndk{  
 moduleName **"myJNI"**}

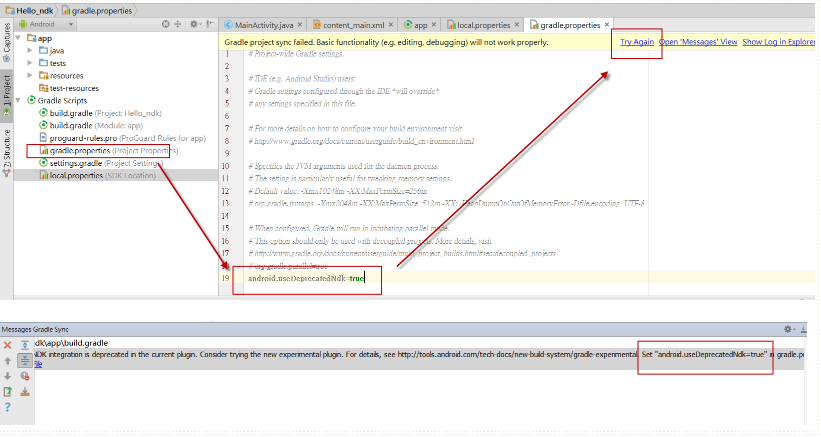
sourceSets.main{  
 jni.srcDirs=[]  
 jniLibs.srcDir **"src/main/jniLibs"**}

解說:  
加入ndk路徑，也針對等一下NDK編譯書來的路徑指定出來  
按下Sync 會產生編譯錯，系統會建議你加入另一個NDK相關的指令，馬上到下一步修正去.....

(三)修正gradle.properties的錯誤

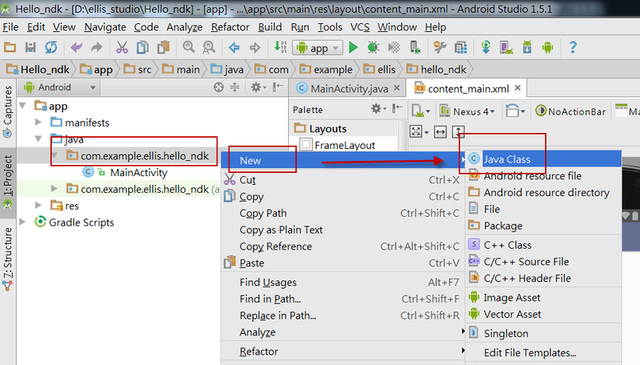
補上以下這行，錯誤就可以修正完畢

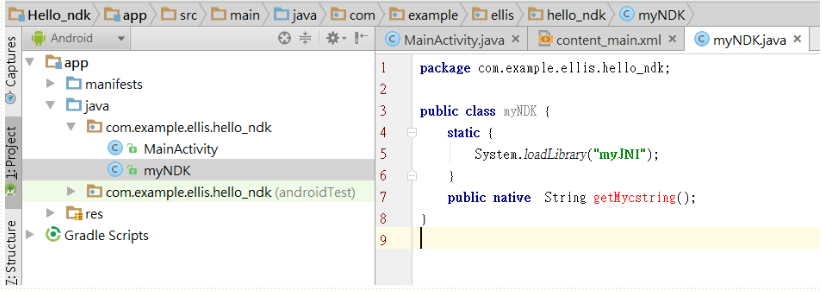
**android.useDeprecatedNdk**=**true**



(四)產生JNI檔案

接著於程式碼中　新增Java Class檔案  檔名我是使用myNDK

[](http://photo.xuite.net/_pic/lwchafter30/19521903/1112637779_l.jpg/redir)



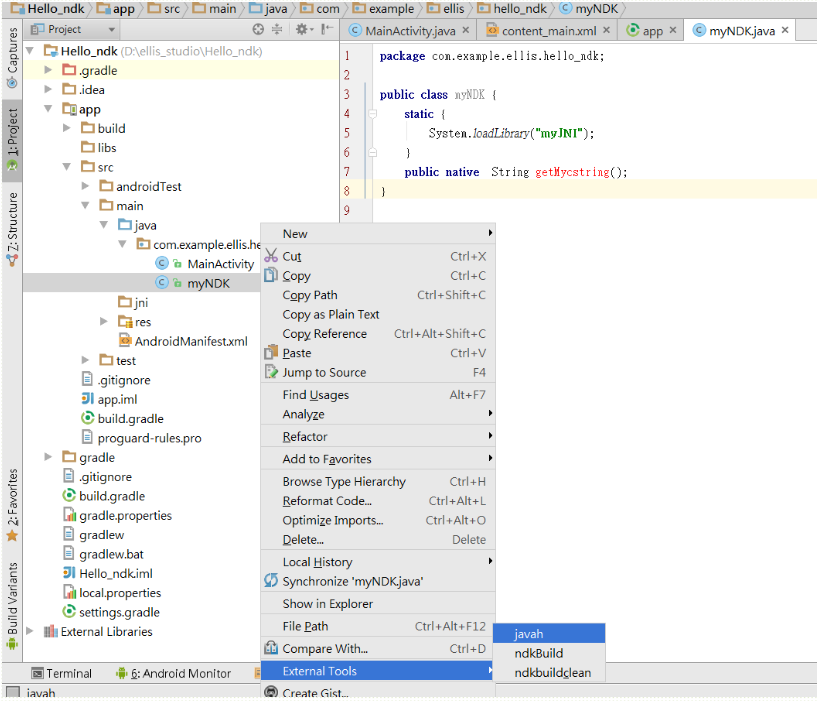
**快速複製區:**

**static** {  
 System.*loadLibrary*(**"myJNI"**);  
}  
**public native** String getMycstring();

解說:  
有產生一個相對應的C語法資源，檔名是myJNI，裡面有可以呼叫的方法叫做getMycstring()

(五)產生Header檔案(必須由JavaH自動產生比較簡單，盡量不要自己手動建置一個新檔案來產生)

於上一個步驟JNI檔案上按下 右鍵--->external tools-->javah(我的命名)

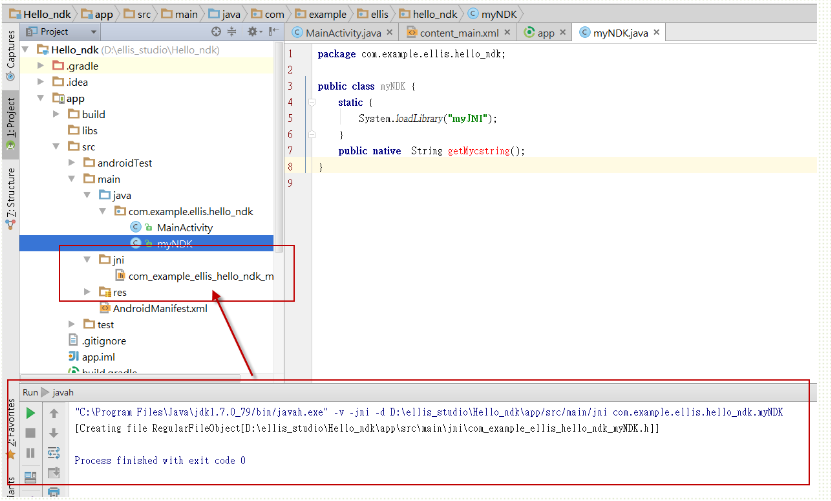


接著Android Studio內的Terminal會被開啟，執行產生C語言HEAD檔案的步驟，

成功的話會向下圖一樣，就會於jni資料夾中產生一個根據我的Project Name產生的一個HEAD檔

建議檔名不要去動他....

如果有錯誤與失敗，HEAD檔案不會產生，那就要檢查第二個步驟中，External tools 裡的javah內的路徑是不是符合你的電腦環境

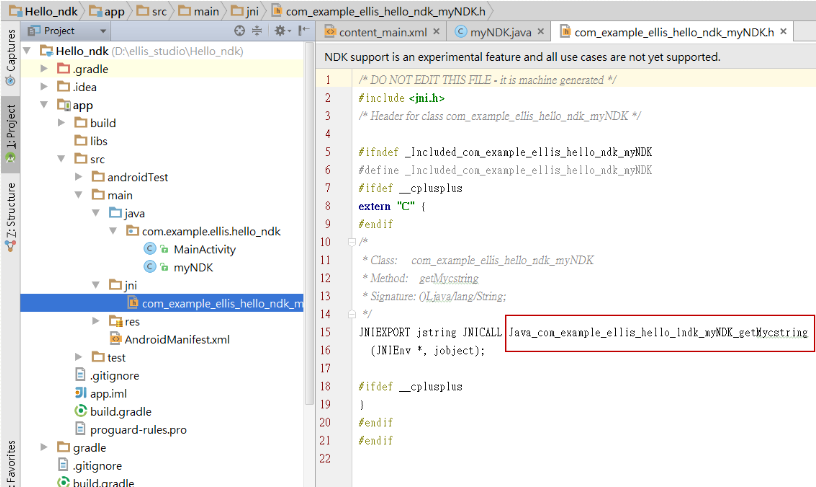


 產生完成後，檢查一下HEAD內容

自動產生的內容裡面，他會將你的project name，譬如我的範例是:**com.example.ellis.hello\_ndk**

供android java 呼叫的java class 檔名是myNDK，會呼叫到的C方法是**getMycstring**

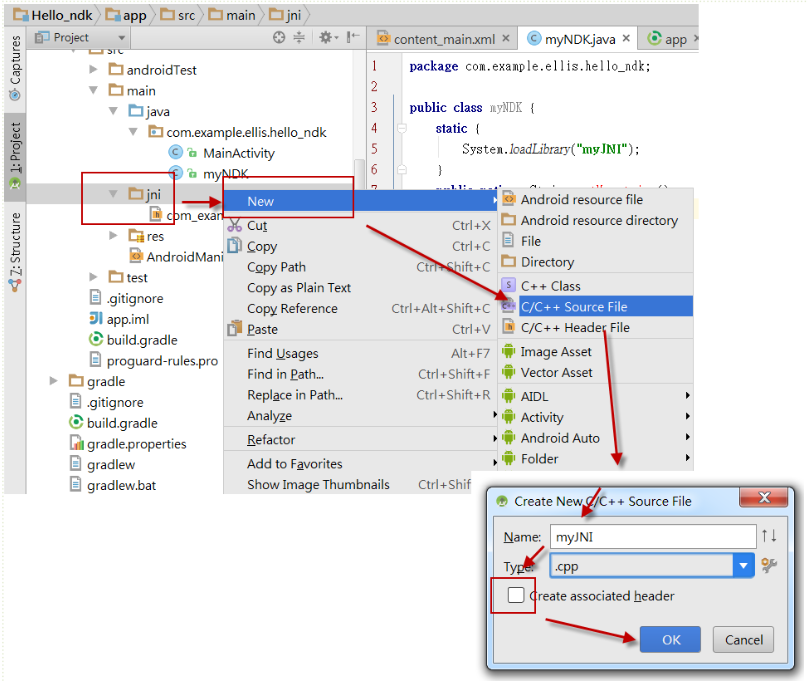
所以他會將JAVA的設定全部結合在這個HEAD中，所以我強烈建議由javah自動去產生



(六)產生C/C++的原始碼檔

在jni資料夾上右鍵--> new --> c/c++ source file

產生的小視窗，記得把create associated header取消，只產生source檔案



產生之後，就是開始編寫C的內容

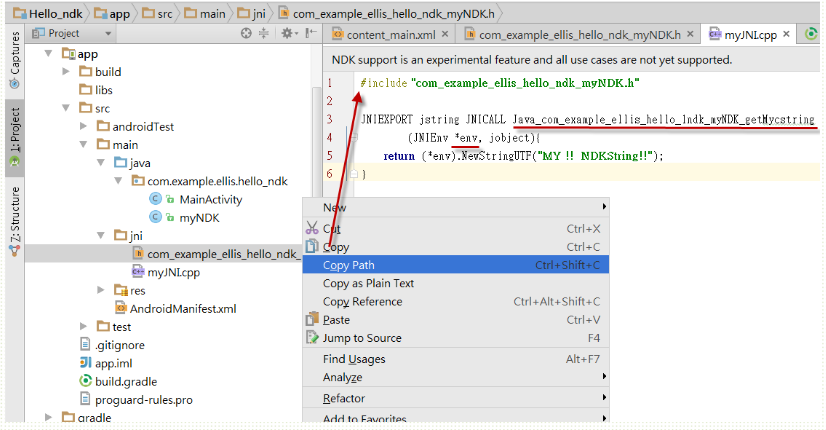
首先 先include剛剛自動產生的HEAD檔，因為檔名有長，可以用Copy Path的方式取得

之後方法的初始化，請打開剛剛的HEAD檔，複製其內容後貼來

VIP:記得，第一個運算子請把它改成 \*env

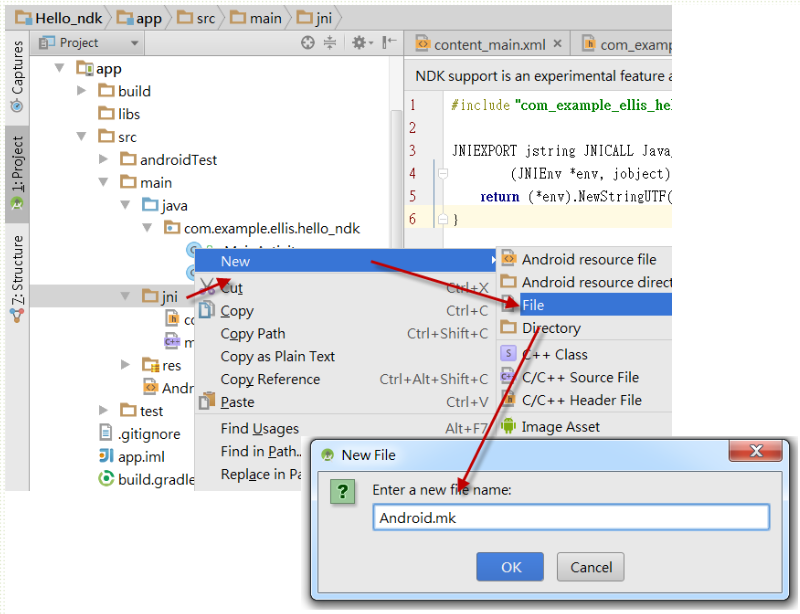
**快速複製區**

#include **"XXXX.h"**JNIEXPORT jstring JNICALL XXXXOOOO\_getMycstring  
 (JNIEnv \*env, jobject){  
 **return** (\*env).NewStringUTF(**"MY !! NDKString!!"**);  
}



(七)產生2個MK檔...快結束了

在jni資料夾上按下右鍵，新增兩個純檔案，第一個Android.mk，第二個是Application.mk



接著是兩個檔案裡面的內容 解說:

LOCAL\_MODULE := myJNI  
LOCAL\_SRC\_FILES := myJNI.cpp

第一個模組名稱，必須跟build.gradle裡面的moduleName相同

也跟JAVA程式裡面，myNDK class裡面 System.*loadLibrary*(**"myJNI"**)相同

SCR\_FILES的名稱則是真正C語言的原始碼名稱

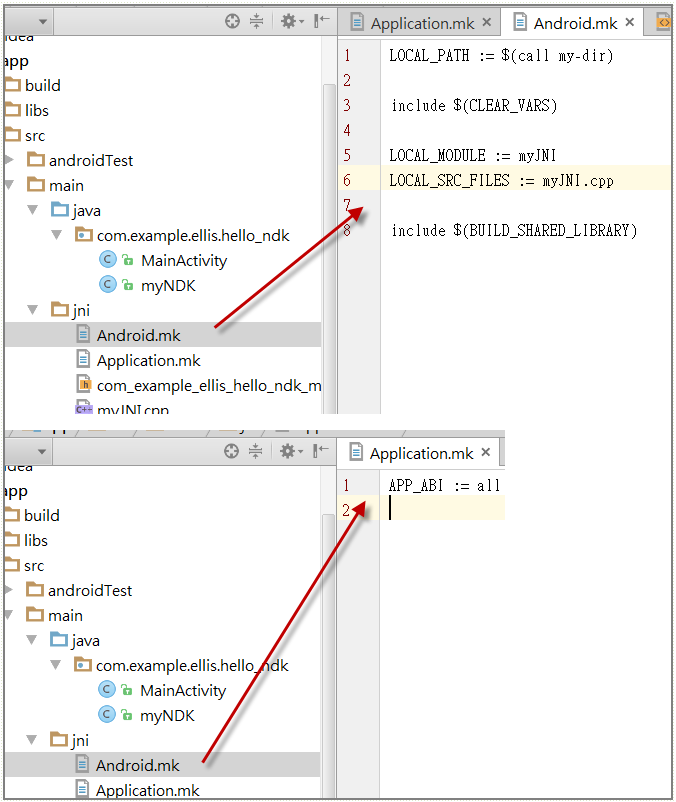
**快速複製區**

Android.mk的內容

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

Application.mk的內容

APP\_ABI := all

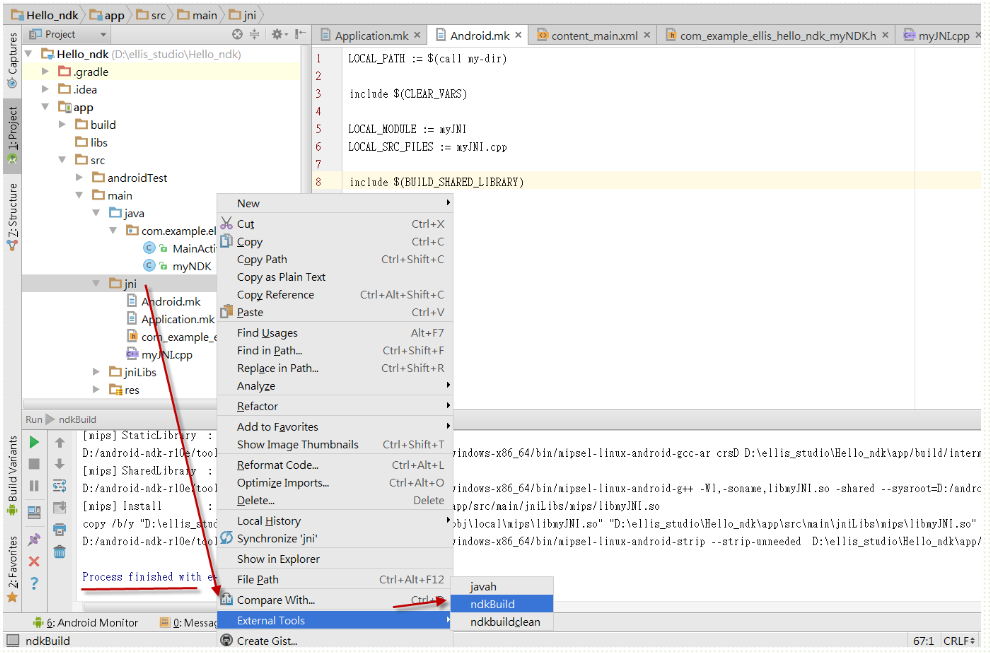
­

最後 就是真正執行一次C的編譯

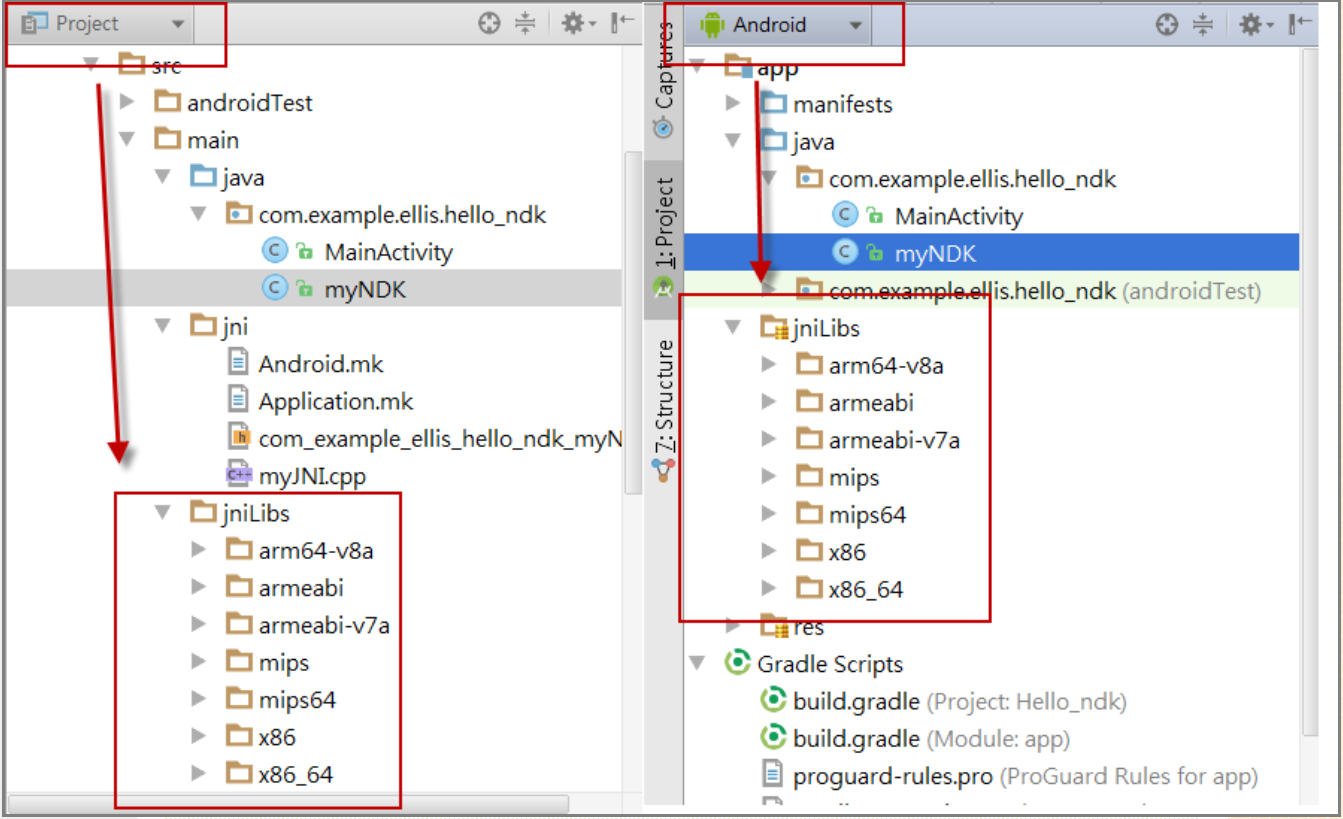
請到JNI資料夾按右鍵-->External Tools-->ndkbuild

如果成功，就會向下圖出現一堆編譯順序，之後會寫藍色的字，表示完成。

Process finished with exit code 0

­

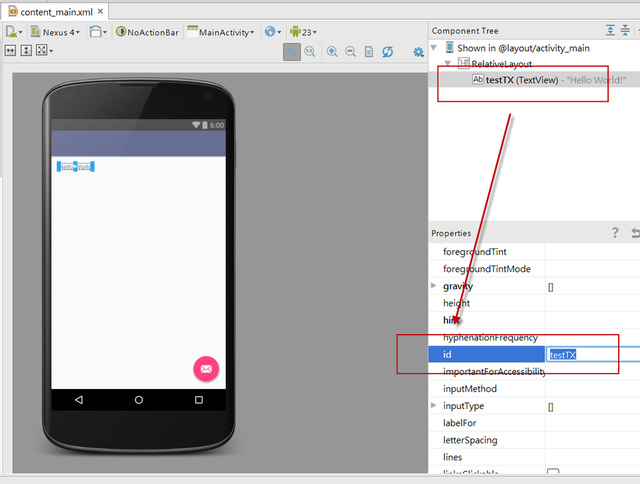
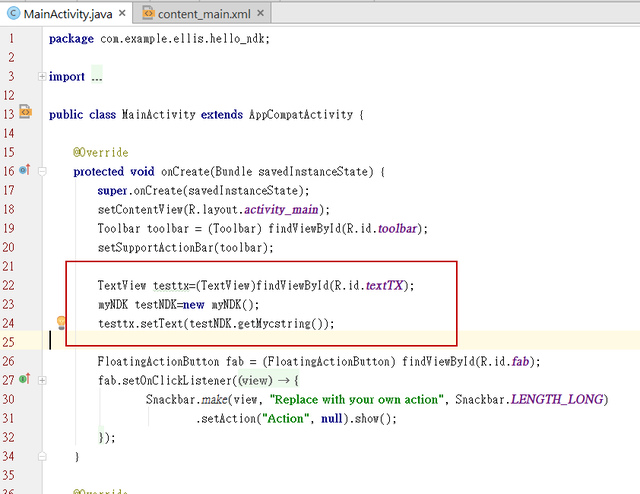
最後檢查一下編譯產生的XXX.so檔案，是不是有依照各種手機CPU類型編譯在不同的資料夾中



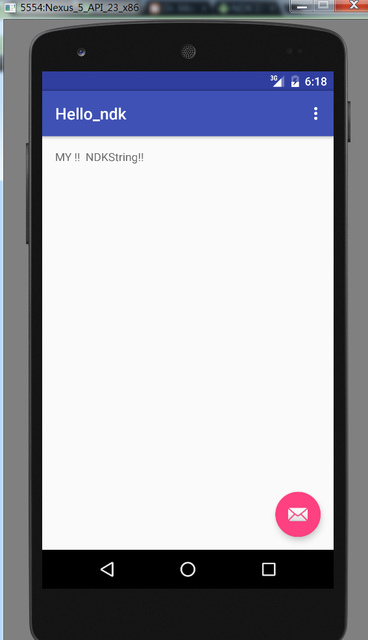
-----------------以上就是已經建構完成C語言與JAVA間的架接步驟---------------------

**接著是如何在主程式中被使用呢?**

TextView testtx=(TextView)findViewById(R.id.***textTX***);  
myNDK testNDK=**new** myNDK();  
testtx.setText(testNDK.getMycstring());  
  
**就完成呼叫的測試了...............**

[](http://photo.xuite.net/_pic/lwchafter30/19521903/1112652851_l.jpg/redir)  
[](http://photo.xuite.net/_pic/lwchafter30/19521903/1112653465_l.jpg/redir)

路已經鋪好了，要做什麼祕密的東西，或是特殊演算法，都可以用C來架接啦.......

[](http://photo.xuite.net/_pic/lwchafter30/19521903/1112653946_l.jpg/redir)

本文是參考以下幾個網頁並修正的結果

**以下這個寫得非常好，推薦給大家**

<http://wiki.jikexueyuan.com/project/jni-ndk-developer-guide/workflow.html>

-----其他參考------

<https://8085studio.wordpress.com/2015/04/25/android-studio-ndk-jni/>

<https://www.youtube.com/watch?v=RmPuwdxR1qs>

<http://kn-gloryo.github.io/Build_NDK_AndroidStudio_detail/>