Create .so Files Using NDK

(Source: Internet)

- 1. Install ndk-build in window (search and download from the internet)
- 2. Set "path" environment in window. Suppose

"C:\Users\PCuser\AppData\Local\Android\Sdk\ndk-bundle" is where we install ndk-builder. In window, Control Panel > System > Advanced system settings > Advanced > Environment Variables > System variables > Path > Edit... In the variable value, add ";" and also add "C:\Users\PCuser\AppData\Local\Android\Sdk\ndk-bundle". Then, click OK.

3. Create a folder with whatever name (for example D:\hello-jni). In the folder "hello-jni", create a folder named exactly "jni". In the folder "jni", add three files, including Android.mk, Application.mk, and hello-jni.c. The content of these files is as follows.

.....

In the file "Android.mk"

LOCAL_PATH := \$(call my-dir)

include \$(CLEAR_VARS)

LOCAL_MODULE := hello-jni

LOCAL_SRC_FILES := hello-jni.c

include \$(BUILD_SHARED_LIBRARY)

include \$(BUILD_EXECUTABLE)

Note that LOCAL_MODULE is to set the name of the .so files

In the file "Application.mk"

APP_CFLAGS += -Wno-error=format-security

APP_ABI := all

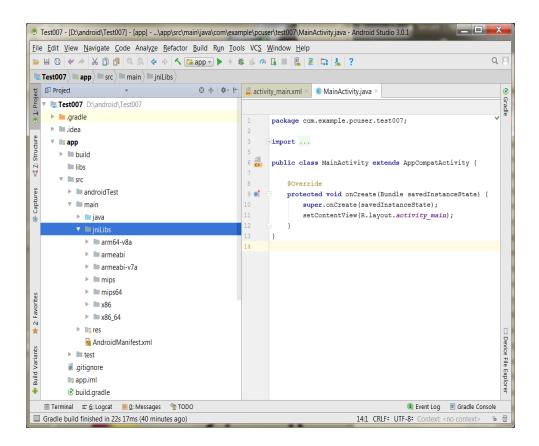
APP ABI := armeabi armeabi-v7a x86

```
In the file hello-jni.c
#include <string.h>
#include <ini.h>
JNIEXPORT jstring JNICALL
Java_com_example_pcuser_test007_MainActivity_getStringFromJNI(JNIEnv
*env, jobject thisObj) {
 return (*env)->NewStringUTF(env, "Hello from native code!");
}
jstring
Java_com_example_pcuser_test007_MainActivity_getJniString(JNIEnv* env,
jobject thiz){
  return (*env)->NewStringUTF(env, "Hello from JNI! Saurabh");
}
JNIEXPORT jint JNICALL
Java_com_example_pcuser_test007_MainActivity_getIntSqure(JNIEnv* env,
jobject obj,jint value) {
  return value * value;
}
JNIEXPORT jboolean JNICALL
Java_com_example_pcuser_test007_MainActivity_getBooleanMethod(JNIEn
v* env,jobject obj, jboolean unsignedChar) {
  return !unsignedChar;
}
```

4. Open cmd in window and change to the folder "jni" using the command > cd /d D:\android\hello-jni\jni

5. Run ndk-build by the following command D:\android\hello-jni\jni> ndk-build

- 6. The .so files is output in the folder "hello-jni\libs"
- 7. To use .so files, create a new project in Android Studio. In "Application name", type "Test007" and in "Company domain", type "pcuser.example.com". Then, click "Next" until it is done.
- 8. In the folder "/src/main/", create a folder called "jniLibs". Put the .so files in this folder as the following figure.



9. In the file "build.gradle" add the following lines.

```
₱ Test007 - [D:\android\Test007] - app - Android Studio 3.0.1

\underline{\text{File}} \ \ \underline{\text{Edit}} \ \ \underline{\text{View}} \ \ \underline{\text{N}} \text{avigate} \ \ \underline{\text{C}} \text{ode} \ \ \text{Analy} \underline{\text{ze}} \ \ \underline{\text{R}} \text{efactor} \ \ \underline{\text{B}} \text{uild} \ \ R\underline{\text{un}} \ \ \underline{\text{T}} \text{ools} \ \ \text{VCS} \ \ \underline{\text{W}} \text{indow} \ \ \underline{\text{H}} \text{elp}
늘 발 형 (ゼ ㅎ ) ¼ jù jǔ | 역 및 (숙 ㅎ ) < ┗app ▼ ▶ + 非 ☆ ル O 및 ■ | 및 | ⑧ | 다 및 및 | ?
⊕ ‡ | ♣ l t activity_main.xml × © MainActivity.java × ⊙ app ×
               ▶ ■ armeabi
                                                          android{}
               ▶ ■ armeabi-v7a
                                                          android {
               ▶ ■ mips
                                                                compileSdkVersion 26
               ▶ III mips64
                                                                defaultConfig {
               ▶ 1 x86
                                                                    applicationId "com.example.pcuser.test007"
               ► 1 x86_64
                                                                    minSdkVersion 15
                                                                    targetSdkVersion 26
             ► li≡res
                AndroidManifest.xml
                                                                    versionName "1.0"
           ▶ ■ test
                                                                    testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner
          aitignore ...
           lmi.qqs 📰
         build.gradle
           proguard-rules.pro
     ▶ ■ build
     ▶ □ gradle
        aitianore
                                                                    release {
                                                                         minifyEnabled false
        build.gradle
                                                                         proguardFiles getDefaultProguardFile('proguard-android.txt'), 'progu
       gradle.properties
        gradlew
        gradlew.bat
        local.properties
                                                           dependencies {
       © settings.gradle
                                                               implementation fileTree(dir: 'libs', include: ['*.jar'])
                                                               Test007.iml
     III External Libraries
   Gradle build finished in 1s 558ms (moments ago)
                                                                                             66 chars, 2 line breaks 14:1 CRLF= UTF-8= Co
```

10. Modify the file "MainActivity.java" as follows.

```
package com.example.pcuser.test007;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;

public class MainActivity extends AppCompatActivity {
    static {
        System.loadLibrary("hello-jni");
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Log.d("Hello", getStringFromJNI());
    }
    public native String getStringFromJNI();
}
```

11. The result shows as follows.

