使用 AndroidStudio 进行 NDK 开发（一）

原文链接：https://dailyios.com/article/Using_Android_Studio_for_NDK_development.html

在 AndroidStudio 中进行 NDK 开发比起以往的 Eclipse 要方便的多，下面来介绍下如何使用 AndroidStudio 这个 IDE 工具实现 NDK 相关开发工作。 —— 由 DailyiOS 分享

在AndroidStudio中进行NDK开发比起以往的Eclipse要方便的多，下面来介绍下如何使用AndroidStudio这个IDE工具实现NDK相关开发工作。

1. 准备工作

在实际写代码之前，首先我们还是需要做一些准备工作：

1. 下载NDK开发包：Android官方下载页面
2. 配置系统环境变量

下载好NDK开发包之后，直接解压到任意目录，然后需要配置一下系统环境变量，之所以要配置环境变量，是为了方便使用命令ndk-build脚本进行NDK编译。配置参考如下：

```
# 在.bash_profile中配置如下代码
export ANDROID_NDK=/Users/liangqi/android-ndk
export PATH=$ANDROID_NDK:$PATH

# 然后执行如下代码，更新配置文件
source .bash_profile
```

其实编译 C/ C++ 代码不一定在AndroidStudio中，如果配置好环境变量，直接使用进入项目中的jni目录执行ndk-build命令即可在当前目录下生成一个libs的目录，里面存放了不同 平台的.so包，当然运行这个命令的前提是，这个目录下至少得有一个Android.mk文件，如果需要指定具体的编译平台，那么还需要添加一个Application.mk文件，当然，如果命令行让你头疼，那么你可以采用gradle的方式来解决这些问题，接下来我们将分别介绍这些使用方式。

2. 项目配置

使用AndroidStudio开发前我们也要做点额外工作，我们需要在项目根目录下local.properties中添加编译NDK的路径：

```
ndk.dir=/Users/liangqi/android-ndk
```

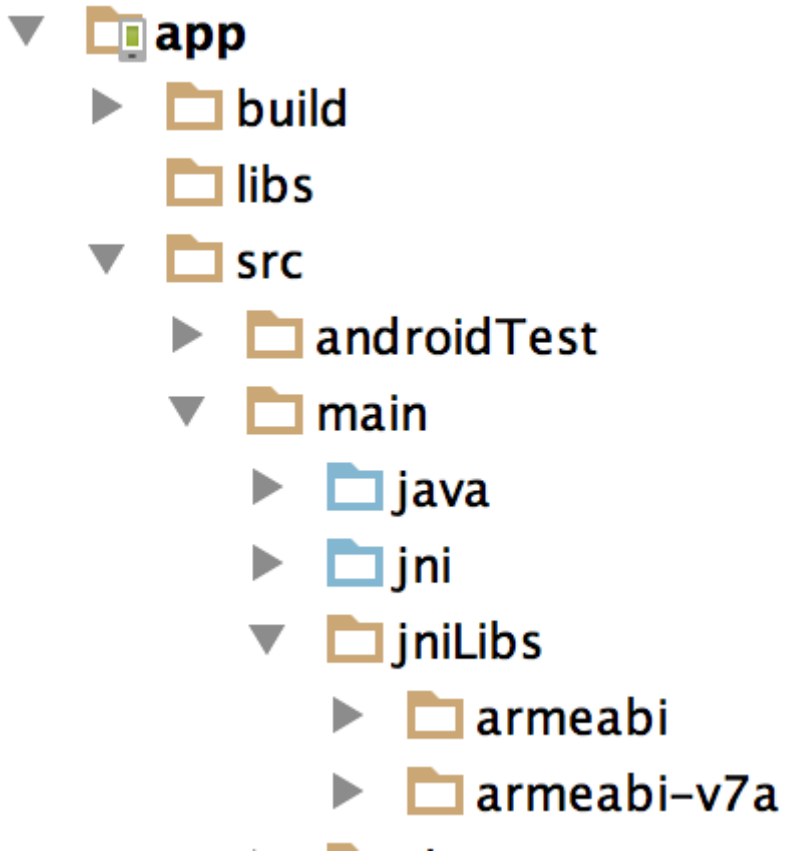
如果这个文件不存在，你可以手动生成一个，然后再添加上述内容即可。完成这个步骤之后，我们就可以正式开始着手NDK相关的开发工作了。之所以要配置这个目录，目的是让我们开发的项目在使用gradle编译时能够找到`NDK`相关编译路径

那么，接下来的工作也分为两种情况：

1. 没有（C/C++）源码，别人已经提供好相应的.so文件，不需要编译代码
2. 拥有（C/C++）源码，需要自己编译.so文件

2.1 已有 .so 文件，不需要编译源码

这类情况是最简单的，.so文件以及被其他人员编译好，或者是第三方库来提供的，那么我们只需要把相应.so文件放到AndroidStudio目录src/main/jniLibs/下即可，当然，肯定需要按CPU架构分不同的子目录，例如arm，如下：



jniLibs是AndroidStudio默认提供的ndk目录，用来存放已经编译好的.so文件，当然你也可以放在任意自定义目录下，例如src/main/libs，然后在build.gradle中指定相应的资源目录位置即可：

```
android {
    sourceSets.main {
```

```
            // 你的.so库的实际路径
            jniLibs.srcDir 'src/main/libs'
        }
    }
```

在导入.so文件完成之后，那么你可以在相应的java类文件中，加载这个静态库，一般来说，.so文件如果由第三方提供，他在提供.so文件的同时也会提供相应的java调用类文件，或者按之前双方定好的规则自己创建相应类文件，并生成相应的方法，之所以要约定好只因为，NDK下的 C/C++ 函数和Java桥接的函数命名是有约束的，规则如下：

```
Java_PackageName_ClassName_MethodName
```

双方必须按这个规则来实现或者调用此函数，否则不会成功，例如，我们现在有一个函数：String stringFromJNI() 的java函数 ，它在com.example.hellojni.HelloJni这个文件下，这个函数用来返回一个字符串，功能由底层C来实现，那么相应的C语言jni开发文件中就必须按上述规则命名一个Java_com_example_hellojni_HelloJni_stringFromJNI(JNIEnv* env, jobject thiz)的函数，并返回一个字符串结果：

```
#include
#include

jstring Java_com_example_hellojni_HelloJni_stringFromJNI( JNIEnv* env, job
{
    return (*env)->NewStringUTF(env, "Hello from JNI !  Compiled with ABI
}
```

同样对应的java文件也必须：

1. 文件必须在com.example.hellojni包名下
2. 类文件名必须是HelloJni
3. 方法名必须是stringFromJNI

```
package com.example.hellojni;

class HelloJni {
     public static native String stringFromJNI();

     static {

         System.loadLibrary("hellojni");
```

```
    }
}
```

2.2 有源码，需要编译 .so 文件

如果有C/C++源码，没有.so文件，那么我们就得手动把源码文件编译成.so文

2. 使用gradle脚本自动实现编译

AndroidStudio默认的源码存放目录是:

```
src/main/jni
```

如果你没发现此目录，那么你可以手动创建一个，把所有的C/C++源码放在此文件下，当然并非必须要放在此目录下，你可以自定义目录，然后在build.gradle中做一个资源路径指定即可：

```
// build.gradle
android{
    sourceSets.main {
        // 你的源码目录
        jni.srcDir 'src/main/otherDir'
    }
}
```

2.2.1 手工执行命令经行编译

在使用手工编译（C/C++）文件之前，我们要回到文章开头部分，我们需要配置好系统环境变量，这样我们才能在系统环境下执行 ndk 相关编译命令，如果您的环境变量还没有配置，那么可以参考下文章开头部分，如果已经做好这部分工作，那么咱们继续。

接下来，我们还要创建如下两个文件：

1. Android.mk
2. Applicatoin.mk (非必要)

2.2.1.1 创建Android.mk

Android.mk文件用来指定源码编译的配置信息，例如工作目录，编译模块的名称，参与编译的文件等，大致内容如下：

```
LOCAL_PATH      :=  $(call my-dir)
include             $(CLEAR_VARS)
LOCAL_MODULE    :=  hello_jni
LOCAL_SRC_FILES :=  hello_jni.c
include             $(BUILD_SHARED_LIBRARY)
```

- LOCAL_PATH：设置工作目录，而my-dir则会返回Android.mk文件所在的目

LOCAL_PATH）。
- LOCAL_MODULE：用来设置模块的名称。
- LOCAL_SRC_FILES：用来指定参与模块编译的C/C++源文件名。
- BUILD_SHARED_LIBRARY：作用是指定生成的静态库或者共享库在运行时依赖的共享库模块列表。

2.2.1.2 创建Application.mk

这个文件用来配置编译平台相关内容，我们最常用的估计只是 `APP_ABI` 字段，它用来指定我们需要基于哪些CPU架构的.so文件，当然你可以配置多个平台：

```
APP_ABI := armeabi armeabi-v7a x86 mips
```

如果不创建Application.mk文件，那么手动编译的.so文件只有armeabi平台一个版本，其他平台的不会被编译。

假设我们配置好了Android.mk文件，那么接下来我们就可以执行如下命令来生成.so文件了，我们假设开发 `NDK` 的目录为默认目录：

```
cd src/main/jni/
ndk-build
```

如果顺利，那么你将会看到，在src/main/目录下会多了一个libs目录，这是NDK使用命令编译.so文件的生成的默认目录，而AndroidSutdio默认加载NDK的目录是jniLibs，那么你有两种解决方式：

1. 配置build.gradle资源目录，参见文章2.1小节
2. 使用 ndk-build NDK_LIBS_OUT=../jniLibs 指定具体的输出目录

当你得到了.so文件，那么接下来就是在java文件中调用执行即可，如果想了解更多ndk-build命令内容，可参见：Android ndk-build 使用文档

2.2.2 使用gradle脚本

当然该机器做的事我们还是尽量让机器来做，因此，接下来我打算使用build.gradle来添加一些配置，让Gradle自动帮我完成编译工作，这简直就是爽歪歪啦！

使用gradle,你再也不用手动添加Android.mk和Application.mk文件，一切在build.gradle文件中就都能搞定，在这里我们直接贴出build.gradle中ndk相关的


```
android.ndk {
    // 模块名称
    moduleName = "hello-jni"

    // 指定编译平台，更多平台信息 参见https://developer.android.com/ndk/g
    abiFilters "armeabi", "armeabi-v7a"
    /*
     * Other ndk flags configurable here are
     * cppFlags.add("-fno-rtti")
     * cppFlags.add("-fno-exceptions")
     * ldLibs.addAll(["android", "log"])
     * stl       = "system"
     */
}
```

使用gradle的好处是，自动编译生成apk文件，并且把相关的.so文件打包到apk安装包中，一劳永逸。

Android Studio

了解 Android Studio Live Templates , 加快开发的 “咒语”

自定义 View 强势来袭，用自定义 View 实现歌词显示控件下篇之自定义 LyricView 的实现