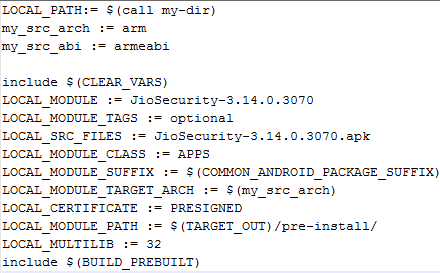
预置及更新APP方法总结

预置和更新app，操作的目录都是apk目录，针对不同的项目，apk文件夹所在的位置有所不同，理论上apk文件夹可以建立在项目内任意目录，编译系统会自动搜索并根据其内Android.mk (编译脚本) 来进行编译。

编译系统采用的是递归搜索,在搜索到父文件目录的Android.mk脚本后递归便被终止。因此一般可以将需要预置的apk文件夹放到一个总文件夹内，并在该文件夹根目录另外写一个Android.mk (管理脚本) ，以便对所有预置apk进行管理。

安卓6.0以下的项目，一般都在packages\apks下，而安卓6.0及以上的，一般都在vendor\apk下。

1. **预置APP时Android.mk文件部分选项含义**

****

**LOCAL\_PATH:= $(call my-dir)**：Android.mk文件必须以定义LOCAL\_PATH为开始。它用于在开发tree中查找源文件。指定Android.mk当前目录

**include $(CLEAR\_VARS)**：CLEAR\_VARS 变量由Build System提供。并指向一个指定的GNU Makefile，由它负责清理很多LOCAL\_xxx.例如：LOCAL\_MODULE, LOCAL\_SRC\_FILES, LOCAL\_STATIC\_LIBRARIES等等。但不清理LOCAL\_PATH.因为所有的编译控制文件由同一个GNU Make解析和执行，其变量是全局的。所以清理后才能避免相互影响。

**LOCAL\_PACKAGE\_NAME:**必须定义，**package的名字，这个名字在脚本中将标识app**或package

**LOCAL\_MODULE**：必须定义，=$(LOCAL\_PACKAGE\_NAME),表示Android.mk中的每一个模块。名字必须唯一且不包含空格，指示当前编译出来的模块名

**LOCAL\_MODULE\_TAGS** :制定该模块在哪个版本下需要编译

user: 指该模块只在user版本下才编译

eng: 指该模块只在eng版本下才编译

tests: 指该模块只在tests版本下才编译

optional:指该模块在所有版本下都编译

**LOCAL\_SRC\_FILES** :表示编译出此模块需要的源文件，可以有多个

**LOCAL\_MODULE\_CLASS** :将用于决定编译时的中间文件存放的位置，如果不指定，不会放到系统中，之后放在最后的obj目录下的对应目录中，例如out\target\common\obj

LOCAL\_MODULE\_CLASS := ETC 放于etc目录

LOCAL\_MODULE\_CLASS := EXECUTABLES 放于bin

LOCAL\_MODULE\_CLASS := SHARED\_LIBRARIES 放在lib下

LOCAL\_MODULE\_CLASS := APPS 放在APPS下

**LOCAL\_MODULE\_SUFFIX** :不用定义，module的后缀，.apk、.so、.jar

**TARGET\_ARCH** :目标CPU类型,如arm

**LOCAL\_CERTIFICATE** :签名认证，默认为testkey

Platform：用系统的签名，通过这种方式只能使apk的权限升级到system级别

PRESIGNED：使用APP本身的签名

**LOCAL\_MODULE\_PATH** :表示模块编译结果将要存放的目录

**LOCAL\_MULTILIB** :指定编译目标为 32位 或 64位

**include $(BUILD\_PREBUILT)**：拷贝预编译的文件(比如用include prebuild.mk编译的)到系统目录

**另外还有如：**

**include $(BUILD\_SHARED\_LIBRARY)** ：BUILD\_SHARED\_LIBRARY是Build System提供的一个变量，指向一个GNU Makefile Script。它负责收集自从上次调用 include $(CLEAR\_VARS) 后的所有LOCAL\_XXX信息。并决定编译为什么。

BUILD\_STATIC\_LIBRARY：编译为静态库。

BUILD\_SHARED\_LIBRARY ：编译为动态库

BUILD\_EXECUTABLE：编译为Native C可执行程序

**LOCAL\_MODULE\_TARGET\_ARCH** :可选值arm/arm x86/arm64,指定目标lib库的类型

**LOCAL\_PRIVILEGED\_MODULE := true** ：表示将app装入system/priv-app,如果没有这个设置，模块会装入system/app,priv-app里能获得系统权限

1. **如何指定最后的目标安装路径**

在Android.mk文件中，指定最后的目标安装路径，用LOCAL\_MODULE\_PATH和LOCAL\_UNSTRIPPED\_PATH来指定。不同的文件系统路径用以下的宏进行选择：

**TARGET\_ROOT\_OUT**：表示根文件系统out/target/product/xxxxx/root。

**TARGET\_OUT**：表示system文件系统out/target/product/xxxx/system。

**TARGET\_OUT\_DATA**：表示data文件系统out/target/product/xxxx/data。

**TARGET\_OUT\_SHARED\_LIBRARIES**：表示out/target/product/xxxx/system/lib

**TARGET\_OUT\_APPS**：表示out/target/product/xxxx/system/app

**ANDROID\_PRODUCT\_OUT**：out/target/product/xxxx/

**TARGET\_OUT\_JAVA\_LIBRARIES**：out/target/product/xxxx/system/framework

1. **如何指定apk预置路径**

**A）** 默认预置apk到system/app/目录（普通系统apk，不可卸载），如前文Android.mk脚本编写之后即可；

**B）**预置apk到system/priv-app/目录（系统核心apk，不可卸载），在前文Android.mk脚本中添加并配置变量：

LOCAL\_PRIVILEGED\_MODULE := true

**C）**预置apk到data/app/目录并且卸载后不需要再会恢复，在前文Android.mk脚本中配置变量：

LOCAL\_MODULE\_PATH := $(TARGET\_OUT\_DATA\_APPS)

**D）**预置apk到data/app/目录并且卸载后恢复出厂可以恢复，在前文Android.mk中配置变量：

LOCAL\_MODULE\_PATH := $(TARGET\_OUT)/vendor/operator/app

1. **对于包含lib库文件的APK预置的处理**

**情况一：**对于预置到data/app/目录下的apk,包括可恢复和不可恢复（即上C和 D）, 一般lib库文件可以不用手动添加，apk在首次运行时，会自动将自身的lib库抽取安装到自身的根目录；

**情况二：**对于预置到system/app/ 和 system/priv-app 目录下的apk（即上一段 A和 B）,因为在android系统中，system分区是不允许应用执行写操作的，因此需要在Android.mk脚本中进行配置，手动添加lib库文件到编译环境,以便lib库文件在编译之后拷贝到对应编译后的apk目录下,否则apk执行时会因找不到lib库而报错；

1. **如何添加lib库文件**
2. **：**不从apk中解压lib库而直接添加

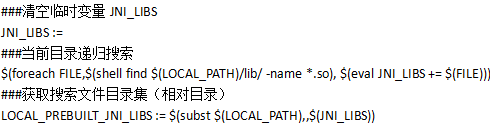
使用LOCAL\_PREBUILT\_JNI\_LIBS属性在Android.mk中添加并配置变量（注意路径对应），@标识符会将apk中的so抽离出来，拷贝到对应编译后的apk目录，如：

截图20160914093453

1. **：**手动解压lib文件到当前apk的编译目录并添加

先解压当前apk内的lib文件夹到当前apk编译目录，同方法一在Android.mk中添加并配置变量（注意路径对应）

若当前apk包含的lib库文件数量比较多时，上述代码可以通过修改为如下代码进行优化，优化的思路是用递归搜索来替代手工对lib库文件进行添加：



若apk支持不同cpu类型的so，针对so的部分的处理:



即将和TARGET\_ARCH对应的so抽离出来

**注**：LOCAL\_PREBUILT\_JNI\_LIBS方法用户提取第三方应用的lib方法

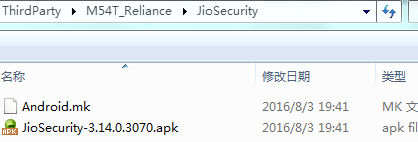
LOCAL\_STATIC\_JAVA\_LIBRARIES则加载加载静态库的方式

LOCAL\_PREBUILT\_JNI\_LIBS在5.0及以上才可以使用，5.0以下的情况，需要将.so文件抽取出来，单独建立文件夹和Android.mk文件进行编译

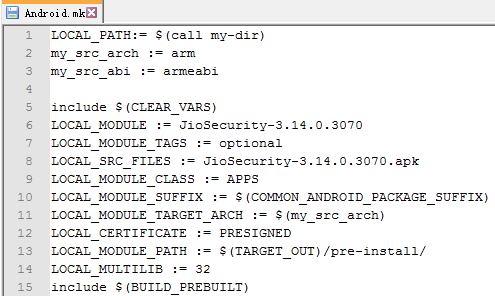
**六、各种预置情况分析**

**1、普通app**

1. 进入apk\ThirdParty\目标项目名下，新建一个文件夹
2. 在apk\ThirdParty\目标项目名下，拷贝其它目录中的Android.mk文件到新建的文件夹下
3. 将需要预置的app拷贝到自己新建的目录下
4. 打开拷贝的Android.mk文件，修改其中的LOCAL\_MODULE := 需要预置的app文件名，LOCAL\_SRC\_FILES := 需要预置的APP文件名.apk，例如对于JioSecurity，文件夹目录如下



Android.mk文件的内容为：



E、进入到apk\products，打开对应项目的.mk问件，如对于M54T\_Reliance，则打开JSR\_apks\_M54T\_Reliance.mk，注意此处打开的是文件名包含了apks的.mk文件

F、在该文件中填加刚刚上面加入的LOCAL\_MODULE指定的行的内容，例如对于JioSecurity，则添加下面一行：JioSecurity-3.14.0.3070 \

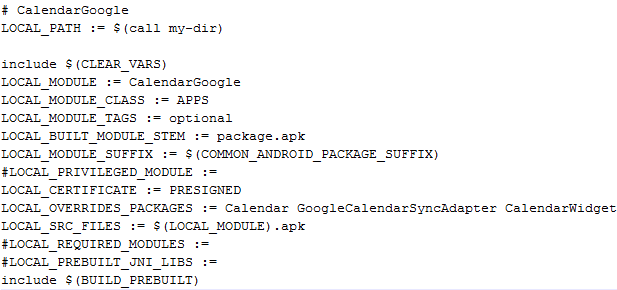
**2、Google app**

1. 进入到apks\GMS\apps目录下，新建一个文件夹

B、拷贝其它目录中的Android.mk文件到新建的文件夹下

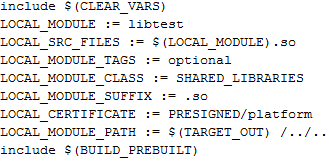
C、将需要预置的app拷贝到自己新建的目录下

D、打开拷贝的Android.mk文件，修改其中的LOCAL\_MODULE := 需要预置的app文件名，例如对于CalendarGoogle，文件夹目录下，其Android.mk文件内容为

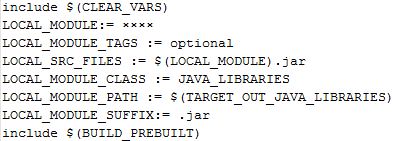


E、正对某些app，还需要显示的加入其库文件，不同的项目对于库文件的处理不同，例如对于M54T\_Reliance，在当前文件夹下的Android.mk文件中显示的指出需要哪些库文件，而在E43\_Ewis中，则需要将.so文件放到该目录下，并修改apks\GMS\lib文件夹下，并修改Android.mk文件

1. **预置库文件**

****

1. **预置jar文件**

****