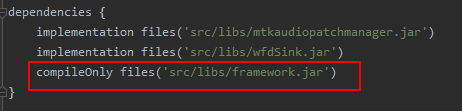
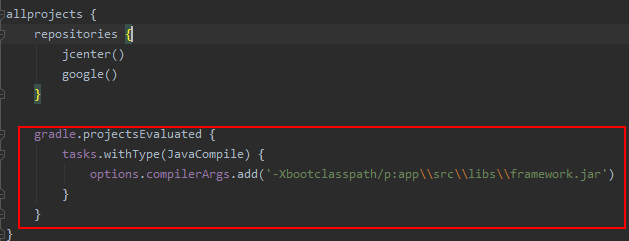
使用@hide内部类：

找到编译出来的Y:\1000\_9950\_First\_Test\release\out\target\common\obj\JAVA\_LIBRARIES\framework\_intermediates\ classes(framework).jar

在AndroiStudio中Add As Library，如下图



在工程的builid.gradle中添加以下内容，



即：

gradle.projectsEvaluated {

tasks.withType(JavaCompile) {

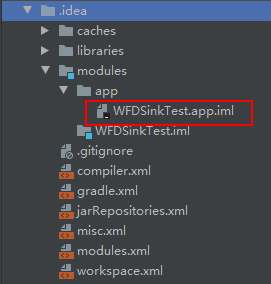
options.compilerArgs.add('-Xbootclasspath/p:app\\src\\libs\\framework.jar')

}

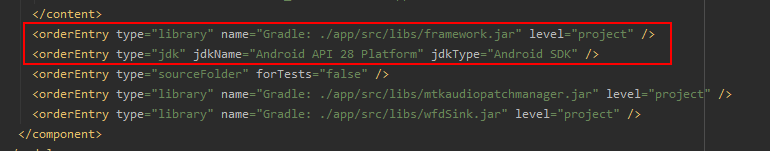
}

即可编译通过，不过idea依旧报错和不能代码提示。

找到以下目录中中的自动编译生成的”\*.app.iml”，



把引入的framework.jar包的优先级提高到SDK前面，即可去除Idea错误提示和使用自动补全功能。



完。

android:stateListAnimator="@null"

可去掉Button自带的阴影效果。

编译出可用的jar包，Mk文件中添加一下内容：

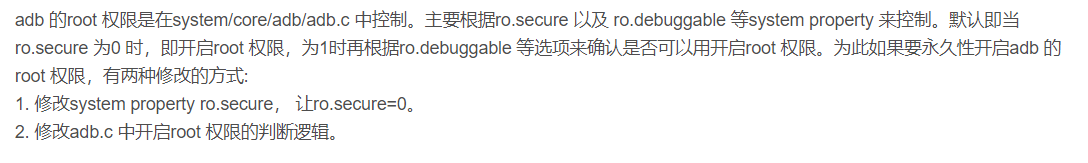
LOCAL\_JACK\_ENABLED := disabled

LOCAL\_DEX\_PREOPT := false

ro.secure

ro.debuggable

ro.adb.secure 是否需要弹出adb授权窗口



Jira密码123456



通过类名找jar包：

find -type f -name "classes.jar" -print0|xargs -0 -l unzip -l|grep -E ".jar$|MtkLog.class" |grep -B 1 "MtkLog.class"

或

find ./ -type f -name "classes.jar" -print0|xargs -0 -L1 -I {} unzip -l -v {}|grep -i -E '\.jar$|ControlCommand'|grep -B1 "ControlCommand"

打印当前处于栈顶的activity名：

dumpsys activity | grep ACTIVITY \r\n

找包名所在目录：

find -type f -iname "AndroidManifest.xml" |xargs grep "packageName"

Android.mk文件的LOCAL\_STATIC\_ANDROID\_LIBRARIES变量可直接导入android的support或androidx库，具体可应用的库名可在frameworks/support/jetifier/jetifier/source-transformer/rewriteMake.py

中查看，不需要自己提供jar包。

Android9的如下：

target\_map = """android-slices-builders,androidx.slice\_slice-builders

android-slices-core,androidx.slice\_slice-core

android-slices-view,androidx.slice\_slice-view

android-support-animatedvectordrawable,androidx.vectordrawable\_vectordrawable-animated

android-support-annotations,androidx.annotation\_annotation

android-support-asynclayoutinflater,androidx.asynclayoutinflater\_asynclayoutinflater

android-support-car,androidx.car\_car

android-support-collections,androidx.collection\_collection

android-support-compat,androidx.core\_core

android-support-constraint-layout,androidx-constraintlayout\_constraintlayout

android-support-constraint-layout-solver,androidx-constraintlayout\_constraintlayout-solver

android-support-contentpaging,androidx.contentpager\_contentpager

android-support-coordinatorlayout,androidx.coordinatorlayout\_coordinatorlayout

android-support-core-ui,androidx.legacy\_legacy-support-core-ui

android-support-core-utils,androidx.legacy\_legacy-support-core-utils

android-support-cursoradapter,androidx.cursoradapter\_cursoradapter

android-support-customtabs,androidx.browser\_browser

android-support-customview,androidx.customview\_customview

android-support-documentfile,androidx.documentfile\_documentfile

android-support-drawerlayout,androidx.drawerlayout\_drawerlayout

android-support-dynamic-animation,androidx.dynamicanimation\_dynamicanimation

android-support-emoji,androidx.emoji\_emoji

android-support-emoji-appcompat,androidx.emoji\_emoji-appcompat

android-support-emoji-bundled,androidx.emoji\_emoji-bundled

android-support-exifinterface,androidx.exifinterface\_exifinterface

android-support-fragment,androidx.fragment\_fragment

android-support-heifwriter,androidx.heifwriter\_heifwriter

android-support-interpolator,androidx.interpolator\_interpolator

android-support-loader,androidx.loader\_loader

android-support-localbroadcastmanager,androidx.localbroadcastmanager\_localbroadcastmanager

android-support-media-compat,androidx.media\_media

android-support-percent,androidx.percentlayout\_percentlayout

android-support-print,androidx.print\_print

android-support-recommendation,androidx.recommendation\_recommendation

android-support-recyclerview-selection,androidx.recyclerview\_recyclerview-selection

android-support-slidingpanelayout,androidx.slidingpanelayout\_slidingpanelayout

android-support-swiperefreshlayout,androidx.swiperefreshlayout\_swiperefreshlayout

android-support-textclassifier,androidx.textclassifier\_textclassifier

android-support-transition,androidx.transition\_transition

android-support-tv-provider,androidx.tvprovider\_tvprovider

android-support-v13,androidx.legacy\_legacy-support-v13

android-support-v14-preference,androidx.legacy\_legacy-preference-v14

android-support-v17-leanback,androidx.leanback\_leanback

android-support-v17-preference-leanback,androidx.leanback\_leanback-preference

android-support-v4,androidx.legacy\_legacy-support-v4

android-support-v7-appcompat,androidx.appcompat\_appcompat

android-support-v7-cardview,androidx.cardview\_cardview

android-support-v7-gridlayout,androidx.gridlayout\_gridlayout

android-support-v7-mediarouter,androidx.mediarouter\_mediarouter

android-support-v7-palette,androidx.palette\_palette

android-support-v7-preference,androidx.preference\_preference

android-support-v7-recyclerview,androidx.recyclerview\_recyclerview

android-support-vectordrawable,androidx.vectordrawable\_vectordrawable

android-support-viewpager,androidx.viewpager\_viewpager

android-support-wear,androidx.wear\_wear

android-support-webkit,androidx.webkit\_webkit

android-arch-core-common,androidx.arch.core\_core-common

android-arch-core-runtime,androidx.arch.core\_core-runtime

android-arch-lifecycle-common,androidx.lifecycle\_lifecycle-common

android-arch-lifecycle-common-java8,androidx.lifecycle\_lifecycle-common-java8

android-arch-lifecycle-extensions,androidx.lifecycle\_lifecycle-extensions

android-arch-lifecycle-livedata,androidx.lifecycle\_lifecycle-livedata

android-arch-lifecycle-livedata-core,androidx.lifecycle\_lifecycle-livedata-core

android-arch-lifecycle-runtime,androidx.lifecycle\_lifecycle-runtime

android-arch-lifecycle-viewmodel,androidx.lifecycle\_lifecycle-viewmodel

android-arch-paging-common,androidx.paging\_paging-common

android-arch-paging-runtime,androidx.paging\_paging-runtime

android-arch-persistence-db,androidx.sqlite\_sqlite

android-arch-persistence-db-framework,androidx.sqlite\_sqlite-framework

android-arch-room-common,androidx.room\_room-common

android-arch-room-migration,androidx.room\_room-migration

android-arch-room-runtime,androidx.room\_room-runtime

android-arch-room-testing,androidx.room\_room-testing

$(ANDROID\_SUPPORT\_DESIGN\_TARGETS),com.google.android.material\_material"""

目录：

release\prebuilts\sdk

release\prebuilts\sdk\current

中有相关支持库的编译文件

## 1. 模块

mk 语法允许将 Source 打包成一个模块，模块又分为：

* 动态库：可以被 install/copy 到应用程序包（apk）
* 静态库：可以被链接入动态库

一个 mk 中能定义一个或者多个模块，也可以将同一份 Source 加入到多个模块中。

## 2. 示例解析

看一个简单的例子：

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

include $(CLEAR\_VARS)

LOCAL\_MODULE := hello-jni

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

include $(BUILD\_SHARED\_LIBRARY)

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

mk 文件须由 LOCAL\_PATH 开始，用于在开发 tree 中查找源文件，宏 my-dir 则由 Build System 提供，返回包含 Android.mk 的目录路径。

* ****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。可以理解为，在每次 mk 编译之前，需要执行 CLEAR\_VARS 的 mk 指令集，用于清理、重置环境，这个清理动作是必须的，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\_MODULE := hello-jni****

模块的名称，必须定义，名字要唯一不能带空格。Build System 会自动添加适当的前缀和后缀。例如，foo 要产生动态库，则生成 libfoo.so。 但请注意：如果模块名被定为：libfoo，则生成libfoo.so，不再加前缀。

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

定义需要编译打包的 C/C++ 源码。不必列出头文件。

* ****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，并决定编译成什么类型的库。

- 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*

## 3. mk 其他变量

* ****LOCAL\_MODULE\_TAGS****

LOCAL\_MODULE\_TAGS := optional  
指定在什么类型的版本下编译：

**user**/**debug**/eng: 指定该模块只在 **user** 版本下才编译（makePRODUCT-XXX-**user**）

optional: 在所有版本下都会编译（貌似在tests下不会，这点待确认）

* ****LOCAL\_PRIVATE\_PLATFORM\_APIS****  
  LOCAL\_PRIVATE\_PLATFORM\_APIS := true

设置后，会使用 sdk 的 hide 的 api 來编译

* ****LOCAL\_SDK\_VERSION****

LOCAL\_SDK\_VERSION := current  
若是在 Android.mk 中添加该选项，则编译时会忽略源码隐藏的API，故在使用源码的 hide api 后会导致编译失败  
API 的分类：

- internal api

翻译为内部API，理解为供sdk内部使用的API。这类接口最初打算就是不对外公开的- hide api

在源码中看到使用@hide 标记的方法或类，就是hide的。这类接口本意是要公开，但是当前阶段仍然不稳定或未开发完成。所以暂时不推荐开发者调用。但可以使用- 普通api

第三方app也可以使用的api

* ****LOCAL\_CERTIFICATE****

LOCAL\_CERTIFICATE := platform  
指定用的是什么签名，如上用的是 platform 签名。

* ****LOCAL\_USE\_AAPT2****

LOCAL\_USE\_AAPT2 := true  
aapt 是Android Asset Packaging Tool的缩写，是编译和打包资源的工具。而aapt2是在aapt上做了优化。

* ****LOCAL\_JNI\_SHARED\_LIBRARIES****

LOCAL\_JNI\_SHARED\_LIBRARIES := libbluetooth\_jni  
声明 mk 要使用到的共享 JNI 库的名称

* ****LOCAL\_JAVA\_LIBRARIES****

LOCAL\_JAVA\_LIBRARIES := javax.obex telephony-common services.net  
指定依赖的共享java类库，这个是编译时依赖，最终不会打包

* ****LOCAL\_STATIC\_JAVA\_LIBRARIES****

LOCAL\_STATIC\_JAVA\_LIBRARIES := \

com.android.vcard \

bluetooth.cc\

services.net \

libprotobuf-java-lite \

指定依赖的静态java类库，最终会打包到apk里面。引用多个的时候，可以像上面那样写。

* ****LOCAL\_STATIC\_ANDROID\_LIBRARIES****

LOCAL\_STATIC\_ANDROID\_LIBRARIES := \

$(ANDROID\_SUPPORT\_DESIGN\_TARGETS) \

android-support-transition \

android-support-v13 \

android-support-v7-appcompat \

android-support-v7-cardview \

android-support-v7-recyclerview \

android-support-v7-palette \

android-support-v4 \

声明要调用 android 的包，像 v7,v13 包

* ****LOCAL\_ANNOTATION\_PROCESSORS****

LOCAL\_ANNOTATION\_PROCESSORS := \

bt-androidx-annotation-nodeps \

bt-androidx-room-common-nodeps \

bt-androidx-room-compiler-nodeps \

bt-androidx-room-migration-nodeps \

bt-antlr4-nodeps \

bt-apache-commons-codec-nodeps \

bt-auto-common-nodeps \

bt-javapoet-nodeps \

bt-kotlin-metadata-nodeps \

bt-sqlite-jdbc-nodeps \

bt-jetbrain-nodeps \

guava-21.0 \

声明用到的注解

* ****LOCAL\_ANNOTATION\_PROCESSOR\_CLASSES****

LOCAL\_ANNOTATION\_PROCESSOR\_CLASSES := \

androidx.room.RoomProcessor

声明用到的注解器

* ****LOCAL\_PROTOC\_OPTIMIZE\_TYPE****

protocol 相关配置

* ****LOCAL\_REQUIRED\_MODULES****

LOCAL\_REQUIRED\_MODULES := SoundRecorder  
指定依赖的模块。一旦本模块被安装，通过此变量指定的模块也将被安装

* ****LOCAL\_PROGUARD\_ENABLED****

LOCAL\_PROGUARD\_ENABLED := disabled

LOCAL\_PROGUARD\_ENABLED := full obfuscation

混淆配置，默认为full obfuscation，全代码混淆，disabled不开启。

* ****LOCAL\_PROGUARD\_FLAG\_FILES****

LOCAL\_PROGUARD\_FLAG\_FILES := proguard.flags  
混淆规则配置文件

特别注意Android.mk文件中LOCAL\_USE\_AAPT2 := true 的使用，没有配置改选项话layout中使用app:xxx\_xxx属性会报错。



查看CPU架构：

getprop ro.product.cpu.abi

源码签名文件目录：

release/build/target/product/security/\*

生成androidStudio上可用的jks系统签名文件：

首先进入密钥对的目录

cd android\build\target\product\security

在本目录下生成platform.pem文件

openssl pkcs8 -inform DER -nocrypt -in platform.pk8 -out platform.pem

在本目录下生成platform.p12文件，并设置别名和密码，这里设置的别名和密码就是在Android Studio打包时选择的别名和密码

openssl pkcs12 -export -in platform.x509.pem -out platform.p12 -inkey platform.pem -password pass:android -name androiddebugkey

在本目录下生成platform.jks文件

keytool -importkeystore -deststorepass android -destkeystore ./platform.jks -srckeystore ./platform.p12 -srcstoretype PKCS12 -srcstorepass android

所以生成的签名文件为platform.jks，别名为androiddebugkey，密码为android。