**一、  系统设定默认语言规则**

1. 系统预置语言，即在 makefile 文件中定义的语言；

–  位置：mediatek/config/${Project}/ProjectConfig.mk，默认第一个是系统默认语言

1. 刷机过后，开机重启，如果未插卡，系统语言为预置的语言；
2. 插入SIM卡过后，系统语言根据 SIM 卡来改变， 默认会把刷机过后第一次插的 SIM 卡的语言设置为默认语言；(如果sim的mcc可以在内置的mcc表中找到)
3. 如果用户没有手动在 Settings 里面设置语言，以后系统语言一直是默认语言（第一次插的 SIM 卡的语言），即使是插入其它国家的卡，系统语言也不会变；
4. 如果用户在 Settings 里面手动设置了语言，以后系统语言会为设置后的语言，无论插入什么卡，系统语言不会变，一直为用户手动设置后的语言。

**二、  修改默认语言**

1、  取消sim卡语言自适应

public static void updateMccMncConfiguration(Context context, String mccmnc)

MccTable.java(frameworks\base\telephony\java\com\android\internal\telephony)

if (mcc != 0) {

                setTimezoneFromMccIfNeeded(context, mcc);

**setLocaleFromMccIfNeeded(context, mcc);**

                setWifiCountryCodeFromMcc(context, mcc);

            }

每次插卡开机，当sim准备好之后，系统会走updateMccMncConfiguration这个函数，其调用**这个setLocaleFromMccIfNeeded函数去根据mcc更新系统语言，可以设置是否调用这个函数来决定是否根据sim切换语言。**

2、  要求每次均可以根据sim卡切换语言

public static void setSystemLocale(Context context, String language, String country)

……

config.userSetLocale = false;

**config.****simSetLocale = true;**

……

    am.updateConfiguration(config);

第一次开机插入sim卡，如果根据mcc找到相应语言，系统会把当前sim卡语言信息保存起来，以后系统无论插入任何语言的sim卡，系统语言均不会在根据sim在进行切换，其中系统是根据**simSetLocale这个参数来决定是否要保存这个信息的，如果在上面把其设为false，则可以实现每次插卡均可自适应(除手动设置语言外)。**

**Android L预置apk**

, 如何将带源码的 APK 预置进系统？

2, 如何将无源码的APK预置进系统？

3, 如何预置APK使得用户可以卸载，恢复出厂设置时不能恢复？

4, 如何预置APK使得用户可以卸载，并且恢复出厂设置时能够恢复？

[SOLUTION]

**一、如何将带源码的APK预置进系统？**

1)     在 packages/apps 下面以需要预置的 APK的 名字创建一个新文件夹，以预置一个名为Test的APK为例

2)     将 Test APK的Source code 拷贝到 Test 文件夹下，删除 /bin 和 /gen 目录

3)     在 Test 目录下创建一个名为 Android.mk的文件，内容如下：

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

include $(CLEAR\_VARS)

LOCAL\_MODULE\_TAGS := optional

LOCAL\_SRC\_FILES := $(call all-subdir-java-files)

LOCAL\_PACKAGE\_NAME := Test

include $(BUILD\_PACKAGE)

4)     打开文件 device\mediatek\common\device.mk

将 Test 添加到 PRODUCT\_PACKAGES 里面。

PRODUCT\_PACKAGES += Test

5)     重新 build 整个工程

**二、如何将无源码的 APK 预置进系统？**

1)     在 packages/apps 下面以需要预置的 APK 名字创建文件夹，以预置一个名为Test的APK为例

2)     将 Test.apk 放到 packages/apps/Test 下面

3)     在  packages/apps/Test 下面创建文件 Android.mk，文件内容如下：

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

include $(CLEAR\_VARS)

# Module name should match apk name to be installed

LOCAL\_MODULE := Test

LOCAL\_MODULE\_TAGS := optional

LOCAL\_SRC\_FILES := $(LOCAL\_MODULE).apk

LOCAL\_MODULE\_CLASS := APPS

LOCAL\_MODULE\_SUFFIX := $(COMMON\_ANDROID\_PACKAGE\_SUFFIX)

LOCAL\_PREBUILT\_JNI\_LIBS:= \

**@**lib/armeabi/libtest.so  \

**@**lib/armeabi/libtest2.so

LOCAL\_CERTIFICATE := PRESIGNED

include $(BUILD\_PREBUILT)

若无so，删除LOCAL\_PREBUILT\_JNI\_LIBS

若有so，使用LOCAL\_PREBUILT\_JNI\_LIBS列出所有so的路径，不要忘记使用@。@标识符会将apk中的so抽离出来build进apk同级目录下的lib文件夹中

若apk支持不同cpu类型的so，针对so的部分的处理:

Ifeq ($(TARGET\_ARCH),arm)

LOCAL\_PREBUILT\_JNI\_LIBS := \

@lib/armeabi-v7a/xxx.so\

@ lib/armeabi-v7a/xxxx.so

else ifeq ($(TARGET\_ARCH),x86)

LOCAL\_PREBUILT\_JNI\_LIBS := \

@lib/x86/xxx.so

else ifeq ($(TARGET\_ARCH),arm64)

LOCAL\_PREBUILT\_JNI\_LIBS := \

@lib/armeabi-v8a/xxx.so

…

即将和TARGET\_ARCH对应的so抽离出来

4)     打开文件 device\mediatek\common\device.mk

 将 Test 添加到 PRODUCT\_PACKAGES 里面。

PRODUCT\_PACKAGES += Test

5)     重新 build 整个工程

注：如果App使用System Level的permission，需要預置到/system/priv-app底下 (原在/system/app)。

修改Android.mk，增加LOCAL\_PRIVILEGED\_MODULE := true，以声明app需要放在/system/priv-app下。

**三、如何预置APK使得用户可以卸载，恢复出厂设置时不能恢复？**

1)     在 packages/apps 下面以需要预置的 APK 名字创建文件夹，以预置一个名为Test的APK为例

2)     将 Test.apk 放到 packages/apps/Test 下面

3)     在  packages/apps/Test 下面创建文件 Android.mk，文件内容如下：

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

include $(CLEAR\_VARS)

# Module name should match apk name to be installed

LOCAL\_MODULE := Test

LOCAL\_MODULE\_TAGS := optional

LOCAL\_SRC\_FILES := $(LOCAL\_MODULE).apk

LOCAL\_MODULE\_CLASS := APPS

LOCAL\_MODULE\_SUFFIX := $(COMMON\_ANDROID\_PACKAGE\_SUFFIX)

# LOCAL\_PRIVILEGED\_MODULE := true

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

LOCAL\_CERTIFICATE := PRESIGNED

include $(BUILD\_PREBUILT)

4)     打开文件 device\mediatek\common\device.mk

 将 Test 添加到 PRODUCT\_PACKAGES 里面。

PRODUCT\_PACKAGES += Test

5)     重新 build 整个工程

注意：这个比不能卸载的多了一句

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

**四、如何预置APK使得用户可以卸载，并且恢复出厂设置时能够恢复？**

1在 vendor\mediatek\proprietary\binary\3rd-party\free下面以需要预置的 APK 名字创建文件夹，以预置一个名为Test的APK为例

2 将Test.apk 放入vendor\mediatek\proprietary\binary\3rd-party\free\Test下面

3 在vendor\mediatek\proprietary\binary\3rd-party\free\Test 下面创建文件 Android.mk，文件内容如下

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

include $(CLEAR\_VARS)

# Module name should match apk name to be installed

LOCAL\_MODULE := Test

LOCAL\_MODULE\_TAGS := optional

LOCAL\_SRC\_FILES := $(LOCAL\_MODULE).apk

LOCAL\_MODULE\_CLASS := APPS

LOCAL\_MODULE\_SUFFIX := $(COMMON\_ANDROID\_PACKAGE\_SUFFIX)

LOCAL\_CERTIFICATE := PRESIGNED

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

include $(BUILD\_PREBUILT)

2 打开文件device\mediatek\common\device.mk

将 Test 添加到 PRODUCT\_PACKAGES 里面。

PRODUCT\_PACKAGES += Test

3 然后重新build整个工程

请注意：

若需要apk作为32bit的apk运行，则需要在Android.mk中定义

LOCAL\_MULTILIB :=32

Android里一个listview会对应一个数据数组，那我们只需在这个数组里按格式添加我们所需要的数据就行了，我们的目标是com.android.settings.LocalePicker.java，在该文件133行位置做如下修改：  
 mLocales = new Loc[finalSize+1];  
 for (int i = 0; i < finalSize ; i++) {  
 mLocales = preprocess;  
 }  
 Locale temp = new Locale("vi", "VN");  
 mLocales[finalSize] =new Loc(toTitleCase(temp.getDisplayLanguage(temp)), temp);  
 这步做完后setting的语言列表里就有了越南语了，但点击越南语后，界面没发现有什么变化，那是因为android系统没有对越南语写响应的国际化资源文件，你可以自己写应用测试