做过项目的归纳总结

Y400AG:

主要内容点：安全补丁的合入，BUG的修改，PATH的合入。

收获：学会了gerrit，git，Linux命令如grep，adb等的使用，还有怎么编译代码，查看代码修改的能力

K101BG\_DCC:

主要内容点：SMR的制作，合入安全补丁，制作差分包

收获：学会了fota差分包工具的使用，学会了固定fingerprint，学会了后台配置FOTA，学会了怎么查看上版本的推送点。

犯错点：不清楚本地编译版本与服务器编译版本的区别，导致本地版本制作差分包推送到服务器

K101BG\_FIRST:

主要内容点：MR的制作，合入最新月份的安全补丁，partner\_gms,modem回退到SMR版本，以及其他私有仓库的修改合入

收获：进一步加深了对git的使用理解，学会了MR的制作

犯错点：回退时直接粘贴复制，没有意识到.git仓库的存在，导致复制后git status检查不出新文件的修改，没有把修改点推送的服务器，所以以后尽量使用对比工具合入。还有没有理解到gms.version 时间修改的含义，应该修改到以回退的SMR版本，GMS时间一致。

K101BG\_FIRST:

主要内容点：SMR的制作

收获：学会了base-os的固定，区分清楚了，fingerprint和base-os的区别。

犯错点：把base-os与fingerprint搞混，user版需要固定base-os，而userdebug才需要固定fingerprint。

1.更新代码方式：  repo sync -c --no-tags -r YB

部分同步： repo sync 仓库目录 如：repo sync build

-c 或者--current-branch表示只拉取当前分支代码，坑爹啊，我在init指定了分支，同步的时候，你却悄悄给我拉些没用的。

--no-tags 不拉取tags，tag虽然不大，但架不住多

--no-clone-bundle 不使用clone.bundle，clone.bundle是git bundle一样的打包文件，使用bundle文件可以做cdn下载的分流，cdn听上去不错，但是如果cdn到google的服务器，或者clone.bundle本来就占空间，不是很划算，所以不使用clone.bundle

-f 如果sync失败，继续同步（想想当年LZ写了一个while循环解决同步失败终止问题的）

--force-sync 如果文件目录有差异，强制覆盖掉

原文：https://blog.csdn.net/counsellor/article/details/86591081

2.截屏：shift + prtscn

3.编译脚本:

#!/bin/bash

cd android

./build/tinno\_script/tinno\_build.sh k510 userdebug clean

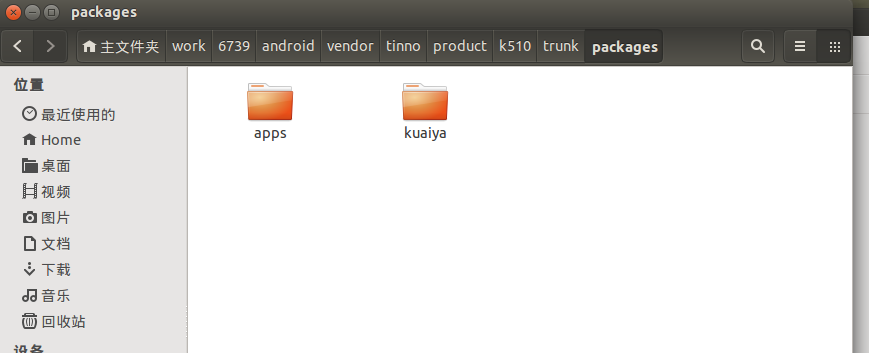
#./build/tinno\_script/tinno\_build.sh p160an userdebug clean myos

myos 是南京的apk仓库，不加这个参数的话就会使用本地的apk

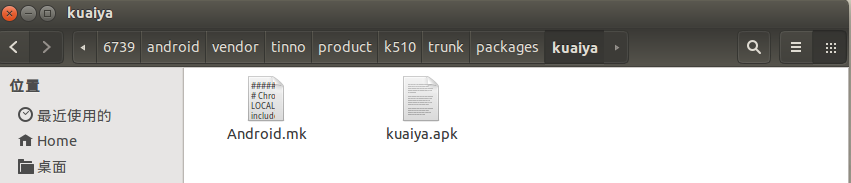
运行build.bash脚本命令：chmod a+x build.sh

bash build.sh

4.预装apk：

第一步：在 /home/android/work/6739/android/vendor/tinno/product/k510/trunk/packages下创建kuaiya文件夹

kuaiya文件夹内把apk拷贝进去，然后新建Android.mk



Android.mk内容如下

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

#Add kuaiya

include $(CLEAR\_VARS)

//生成应用文件夹名，文件名**(不能以数字开头，否则预置不进去)**

LOCAL\_MODULE := kuaiya

//设置在那个版本下编译，user, eng, tests, optional(全版本编译)

LOCAL\_MODULE\_TAGS := optional

//需要被编译模块

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

//安装的位置由LOCAL\_MODULE\_PATH指定，当LOCAL\_MODULE\_PATH没有设置时，系统将会根据LOCAL\_MODULE\_CLASS的值来判断安装的位置，LOCAL\_MODULE\_CLASS变量将通过/build/core/base\_rules.mk文件设置默认的安装目录，常用的值有:apk文件: APPS，so文件: SHARED\_LIBRARIES，bin文件: EXECUTABLES

其他文件: ETC

LOCAL\_MODULE\_CLASS := APPS

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

// 签名方式，如果是platform，代表签名成系统软件, 如果还是PRESIGND，表示用的是apk原本的签名

签名类型 testkey, media, platform, shared, 默认为testkey

LOCAL\_CERTIFICATE := PRESIGNED

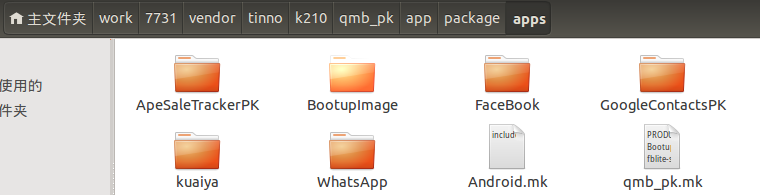
LOCAL\_DEX\_PREOPT := false

//应用生成路径

LOCAL\_MODULE\_PATH := $(TARGET\_OUT)/preloadapp

include $(BUILD\_PREBUILT)

第二步在qmb\_pk.mk文件里面的PRODUCT\_PACKAGES内添加kuaiya





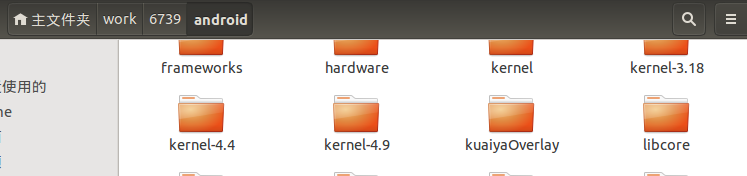
第三步编译环境: source build/envsetup.sh → lunch → mmm <要编译文件夹>

如：mmm vendor/tinno/product/k510/trunk/packages/kuaiya

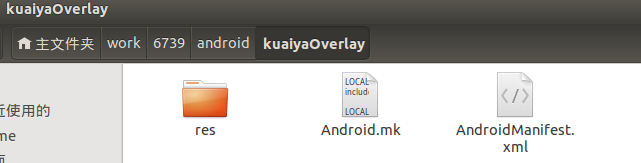
等待编译成功

5.源码覆盖方式：

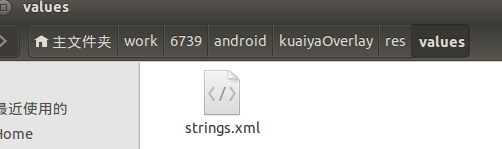
①：在项目的Android文件夹内新建覆盖文件夹



然后在文件内包含res文件夹，Android.mk ，AndroidManifest.xml



Res文件夹内创建要覆盖内容文件夹如：values文件夹内的strings.xml



Strings.xml内容如下：

<?xml version="1.0" encoding="utf-8"?>

<resources>

<string name="app\_name">kuaiya2</string>

</resources>

这里是覆盖app的名称

String.xml格式在apk反编译源码里面找

Android.mk内容如下：

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

include $(CLEAR\_VARS)

LOCAL\_RESOURCE\_DIR := $(LOCAL\_PATH)/res

LOCAL\_MODULE\_TAGS := optional

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

LOCAL\_SDK\_VERSION := current

LOCAL\_PACKAGE\_NAME := kuaiyaOverlay

LOCAL\_MODULE\_PATH := $(TARGET\_OUT\_VENDOR)/overlay

LOCAL\_CERTIFICATE := shared

LOCAL\_AAPT\_FLAGS := --auto-add-overlay

include $(BUILD\_PACKAGE)

AndroidManifest.xml内容如下:

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.dewmobile.kuaiya.apps.messagingoverlay">

<overlay

android:targetPackage="com.dewmobile.kuaiya.apps.messaging"

android:priority="1" android:isStatic="true"/>

</manifest>

高亮内容需要到apk反编译源码里面找对应的包名替换

②编译环境: source build/envsetup.sh → lunch → mmm <要编译文件夹>

如：mmm kuaiyaOverlay/

等待编译成功

6.ubuntu下使用evince README.pdf命令可以直接在命令行中打开pdf文件

7.在AndroidManifest.xml中没有注册layout文件，会造成页面闪退

8.Android系统所有可声明的权限。

访问 <http://developer.android.com/reference/android/Manifest.permission.html> 可以查看

9.自定义广播无法弹出提示问题

Android8.0加intent.setPackage(getPackageName());可以解决

10.Android WebView net::ERR\_CLEARTEXT\_NOT\_PERMITTED问题：

解决方法： 在Application中加入android:usesCleartextTraffic="true"

<https://blog.csdn.net/weixin_34362875/article/details/91442507>

11.项目流程

SPM VPM 软件工程师

1. 跟SPM沟通项目配置 ， 新建提货单和 TSDS 编译配置

2. 本地提货单下拉代码 （如果跟随主干的分支走， 则不需要合并原型）

3. 客户需求导入jira

4. 找spm 和 modem 硬件工程师沟通 参数配置， 还有新建对应订单的 仓库

5. 按照jira 上面的问题号， 修改需求

6. 需求本地验证OK 之后提交 代码

7. 需求改完之后， 进入tsds 编译版本

8. 版本出完了之后 ，项目订单工程师下载验证，提供自检表 和 SVN版本路径给测试 进行全面测试验证。

9. 内部验证OK之后， VPM 把版本提交给客户验证。

10. 客户确认版本， 内部要进行一轮GMS认证， 等到内部认证全部PASS之后， 提交给代理， 最后拿到google的 approve, 最后内部DCC

SMR

1. 主要是谷歌的安全patch

2. GMS VERSION 不能修改，base\_os 属性值 你必须和你 DCC版本的fingerprint 一致

12.动态注册广播

private DynamicBroadcast dynamicBroadcast;

private String BROADCAST\_ACTION = "android.intent.action.BROADCAST\_ACTION";

dynamicBroadcast = new DynamicBroadcast();

IntentFilter intentFilter = new IntentFilter(BROADCAST\_ACTION);

registerReceiver(dynamicBroadcast, intentFilter);

public class DynamicBroadcast extends BroadcastReceiver {

@Override

public void onReceive(Context context, Intent intent){

//todo

}

}

13.更改开机动画、开机铃声

1. zip -r -0 bootanimation.zip ./

第一步 制作动画压缩包

2. adb root

3. adb remount

4. adb push bootanimation.zip /system/media/

5. adb reboot 看效果

14.跳过开机引导：\*89#

15.反编译软件使用

./apktool.sh <反编译pk>

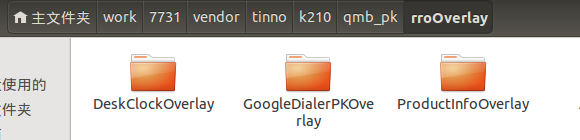
16.更改开机，关机logo：

在vendor/tinno/k210/qmb\_pk/logo 文件夹下覆盖开机，关机logo就行

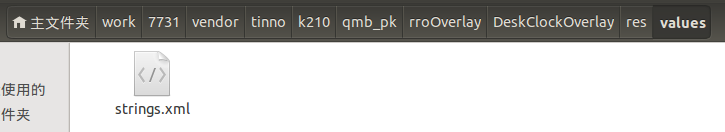
注意：logo文件必须要用256色也即位深为8位的bmp格式文件

17.更改内置应用字符串

第一步在vendor/tinno/k210/qmb\_pk/rroOverlay文件夹下新建要覆盖应用文件夹如DeskClockOverlay



接着按覆盖应用DeskClock中String.xml目录路径新建文件夹



第二步：在原应用String.xml中找到要覆盖字符串位置，然后在自己的String.xml中覆盖

如：把Start week on 该为 Start myweek on

String.xml内容如下：

<?xml version="1.0" encoding="utf-8"?>

<resources

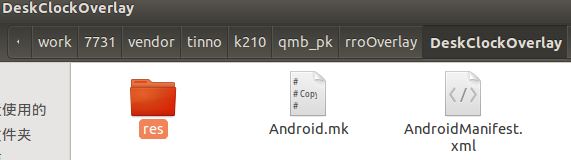
xmlns:xliff="urn:oasis:names:tc:xliff:document:1.2"

xmlns:tools="http://schemas.android.com/tools">

<string name="week\_start\_title">Start myweek on</string>

</resources>

第三步:在DeskClockOverlay目录下新建Android.xml、AndroidManifest.xml两个文件



1. Android.xml修改LOCAL\_PACKAGE\_NAME为覆盖文件夹名如DeskClockOverlay。（Android.xml原始内容格式可以在原覆盖应用内复制，AndroidManifest.xml同理一样）

Android.xml内容如下：

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

include $(CLEAR\_VARS)

LOCAL\_PACKAGE\_NAME := DeskClockOverlay

LOCAL\_MODULE\_TAGS := optional

LOCAL\_CERTIFICATE := platform

LOCAL\_SRC\_FILES := $(call all-java-files-under, res)

LOCAL\_SDK\_VERSION := current

LOCAL\_IS\_RUNTIME\_RESOURCE\_OVERLAY := true

LOCAL\_MODULE\_PATH := $(TARGET\_OUT\_VENDOR)/overlay

LOCAL\_AAPT\_FLAGS := --auto-add-overlay

include $(BUILD\_PACKAGE)

1. AndroidManifest.xml修改对应的包名，报名在原应用AndroidManifest内找

AndroidManifest.xml内容如下：

<?xml version="1.0" encoding="utf-8"?>

<!-- Copyright (C) 2017 Google Inc. All Rights Reserved. -->

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.android.deskclock.overlay">

<overlay android:targetPackage="com.android.deskclock" android:priority="1" android:isStatic="true" />

</manifest>

（如果系统应用没有Android.xml、AndroidManifest.xml可以不需要，如Settings）

最后用mmm 部分编译，再增量编译

18.文件夹内查找指定内容、文件名：

grep -rni “关键字”

**文件夹内查找文件名：**

find -iname “关键字”

locate “关键字”

19.安装apk命令 adb install -r <apk文件>

20.Settings内置选项在一级菜单activity方式

1. 在AndroidManifest.xml 添加如下代码

<activity android:name=".HardKey"

android:label="Mrlove"

android:icon="@drawable/ic\_home\_wikofeatures"

android:taskAffinity="">

<intent-filter android:priority="1">

<action android:name="com.android.settings.HARDKEY" />

<action android:name="android.settings.HARDKEY" />

<category android:name="android.intent.category.DEFAULT" />

</intent-filter>

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.DEFAULT" />

<category android:name="android.intent.category.VOICE\_LAUNCH" />

<category android:name="com.android.settings.SHORTCUT" />

</intent-filter>

<!--在category中的物理位置(按优先级排序时用到，并不是指第9个位置，数越大优先级越大越靠前)-->

<intent-filter android:priority="9">

<action android:name="com.android.settings.action.SETTINGS" />

</intent-filter>

<!--此项在主setting位置(ia.homepage)-->

<meta-data android:name="com.android.settings.category"

android:value="com.android.settings.category.ia.homepage" />

<meta-data android:name="com.android.settings.ACTIVITY\_ACTION"

android:value="com.android.settings.HardKey" />

<meta-data android:name="com.android.settings.PRIMARY\_PROFILE\_CONTROLLED"

android:value="true" />

</activity>

1. 实现HardKey这个activity代码如下

package com.android.settings;

import android.app.Activity;

import android.os.Bundle;

public class HardKey extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.hardkey);

}

}

1. 创建hardkey.xml代码如下

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="horizontal"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<TextView

android:id="@+id/textView2"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_gravity="center"

android:text="hello world" />

</LinearLayout>

21.Settings内置选项在一级菜单fragment方式

1. 在AndroidManifest.xml 添加如下代码

<!-- 这样写，然后在Settings中声明成Settings的内部类，从而不需要单独再写FlashSettingsActivity的实现类 -->

<activity android:name="Settings$FlashSettingActivity"

android:label="MrloveFragment"

android:icon="@drawable/ic\_home\_wikofeatures"

android:taskAffinity="">

<intent-filter android:priority="1">

<action android:name="com.android.settings.FLASH\_SETTING" />

<action android:name="android.settings.FLASH\_SETTING" />

<category android:name="android.intent.category.DEFAULT" />

</intent-filter>

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.DEFAULT" />

<category android:name="android.intent.category.VOICE\_LAUNCH" />

<category android:name="com.android.settings.SHORTCUT" />

</intent-filter>

<intent-filter android:priority="9">

<action android:name="com.android.settings.action.SETTINGS" />

</intent-filter>

<meta-data android:name="com.android.settings.category"

android:value="com.android.settings.category.ia.homepage" />

<!-- FRAGMENT\_CLASS 表明value指定的类是fragment -->

<meta-data android:name="com.android.settings.FRAGMENT\_CLASS"

android:value="com.android.settings.FlashSetting" /><!--FlashSettings实现类 -->

<meta-data android:name="com.android.settings.PRIMARY\_PROFILE\_CONTROLLED"

android:value="true" />

</activity>

②自己实现FlashSetting 这个fragment代码如下：

package com.android.settings;

import android.os.Bundle;

import android.preference.PreferenceFragment;

public class FlashSetting extends PreferenceFragment {

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

addPreferencesFromResource(R.xml.right\_fragment);

}

}

③在xml文件夹实现right\_fragment.xml

<?xml version="1.0" encoding="utf-8"?>

<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">

<PreferenceCategory

android:key="set\_local"

android:title="我的位置" />

<CheckBoxPreference

android:defaultValue="true"

android:key="apply\_wireless"

android:summary="使用无线网络在应用程序（比如Google地图）中查看位置"

android:title="使用无线网络"></CheckBoxPreference>

</PreferenceScreen>

④在Settings.java中添加

public static class FlashSettingActivity extends SettingsActivity { /\* empty \*/ }

⑤在package com.android.settings.core.gateway;的SettingsGateway.java文件中添加如下代码

一，引入包

import com.android.settings.FlashSetting;

二， 在public static final String[] ENTRY\_FRAGMENTS方法中添加：

FlashSetting.class.getName(),

三，在public static final String[] SETTINGS\_FOR\_RESTRICTED方法中添加：

Settings.FlashSettingActivity.class.getName(),

<https://www.jianshu.com/p/2e644c08646d>

<https://blog.csdn.net/fsc_fantexi/article/details/90668122>

22.以窗口的形式打开当前目录：nautilus $PWD

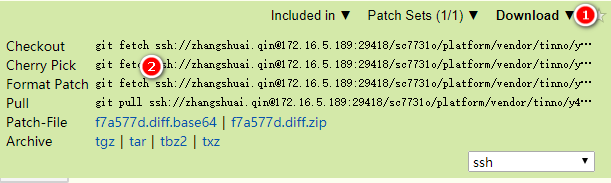
[https://blog.csdn.net/chenwen](https://blog.csdn.net/chenwenxin/article/details/44174563)[xin/article/details/44174563](https://blog.csdn.net/chenwenxin/article/details/44174563)

23.通过gerrit 打补丁

一、直接搜索ID或者commit信息，然后cherry-pick到自己项目上 并add 一个 Reviewers



二、如果失败就下载 cherry-pick 包 到需要打补丁的目录



手动解决冲突后

1.git add

2.git cherry pick continue

3.git push tinno HEAD: refs/for/地址 //提交到git服务器待审核

然后在gerrit add 一个 Reviewers

如果手动解决冲突有不可修改文件，直接添加不可修改内容

1.git add

2.git commit -m <信息>

3.git push tinno HEAD: refs/for/地址 //提交到git服务器待审核

三、如果没有找到ID或者commit信息

git branch -a //查看所有分支

git checkout -b <补丁所在分支> //在本地切换到补丁所在分支

gitk. //搜索commit信息 找到要修改的内容

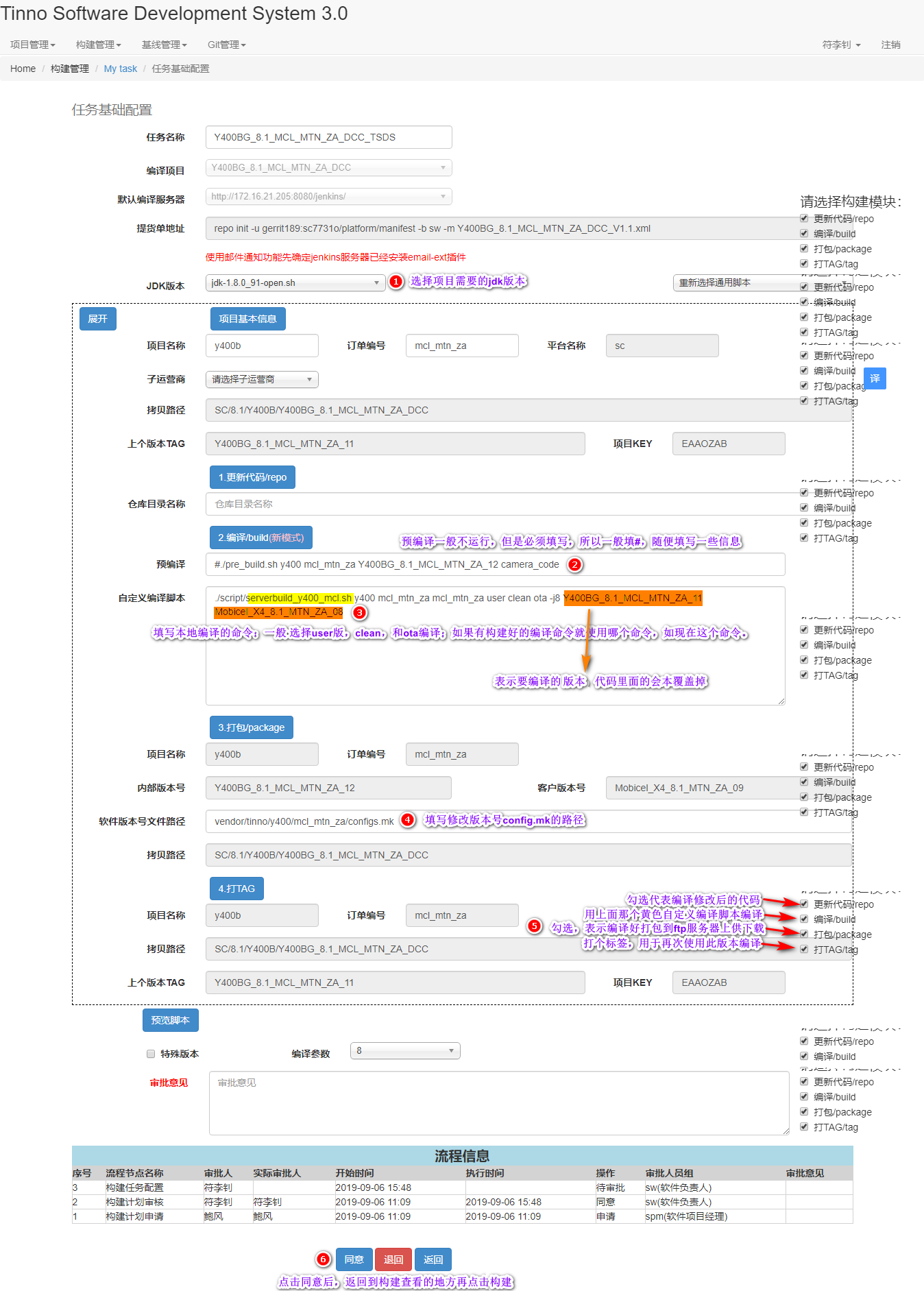
git checkout -b <当前分支> //切换回当前分支

找到修改目录手动修改内容

git add

git commit -m <信息>

git push tinno HEAD: refs/for/地址

24.如何使用TSDS构建项目

25.2018以前旧项目debug 软件编译前修改点（用于gms的debug）

25.1 fingerprint等属性值的固定方法

1、fingerprint属性与用于同次gms的user版本要一致，除了ro.build.type字段值不同

如：user版：Mobicel/X4/X4:8.1.0/OPM2.171019.012/1568094378:user/release-keys

则debug版： Mobicel/X4/X4:8.1.0/OPM2.171019.012/1568094378:userdebug/release-keys

2、incremental属性值修改---与user软件的值要一致，且与fingerprint中incremental字段值一致

基于上一次举例，要设置 ro.build.version.incremental=1568094378

3、tag属性值要与fingerprint中tag字段一致

基于上一次举例，要设置 ro.build.tags=release-keys

具体操作

①先在对应订单目录的buildinfo.sh查找fingerprint属性值的组成方式

②从buildinfo.sh和configs.mk文件中查找对应的宏赋值的地方，如果找不到去device目录grep搜索，没有结果则从build/make/core/Makefile文件中查找本订单fingerprint赋值的位置

如在订单目录(Y400AG\_MTN/vendor/tinno/y400/mcl\_mtn\_za)执行

grep -rniw fingerprint

buildinfo.sh:74: echo "ro.build.fingerprint=$BUILD\_FINGERPRINT"

在device目录执行

grep -rniw BUILD\_FINGERPRINT

无结果

在build/make/core/Makefile目录执行

grep -rniw BUILD\_FINGERPRINT

Makefile:238: BUILD\_FINGERPRINT := Mobicel/X4/X4:$(PLATFORM\_VERSION)/

$(BUILD\_ID)/$(BF\_BUILD\_NUMBER):$(TARGET\_BUILD\_VARIANT)/$(BUILD\_VERSION\_TA

GS)

在订单目录(Y400AG\_MTN/vendor/tinno/y400/mcl\_mtn\_za)执行

grep -rniw ro.build.version.incremental

buildinfo.sh:8:echo "ro.build.version.incremental=$BUILD\_NUMBER"

在build/make/core/Makefile目录执行

grep -rniw BUILD\_NUMBER

Makefile:202: BF\_BUILD\_NUMBER := $(BUILD\_NUMBER)

(可以看到红色的BF\_BUILD\_NUMBER := $(BUILD\_NUMBER) = ro.build.version.incremental

而上文第二点要修改的ro.build.version.incremental=1568094378其实修改BUILD\_NUMBER

就可以了，但是 BUILD\_NUMBER 是个变化的值需要固定，可以直接在手机里面查看具体的值)

连接手机执行

adb shell getprop ro.build.version.incremental

1568104533

在config.mk中修改即可

在订单目录(Y400AG\_MTN/vendor/tinno/y400/mcl\_mtn\_za)执行

grep -rniw ro.build.tags

buildinfo.sh:34:echo "ro.build.tags=$BUILD\_VERSION\_TAGS"

连接手机执行

adb shell getprop | grep ro.build.tags

[ro.build.tags]: [release-keys]

在buildinfo.sh，Makefile修改即可

修改的具体位置:

①/home/android/work/Y400AG\_MTN/vendor/tinno/y400/mcl\_mtn\_za 下

configs.mk

+ifeq ($(TARGET\_BUILD\_VARIANT), userdebug)

+BUILD\_NUMBER := 1568104533 //具体值通过

+endif adb shell getprop

buildinfo.sh ro.build.version.incremental

-echo "ro.build.tags=$BUILD\_VERSION\_TAGS" 查看手机的值

+echo "ro.build.tags=release-keys"

②/home/android/work/Y400AG\_MTN/build/make/core下

Makefile

（如果订单仓库有build\_fingerprint.mk文件，就在里面固定release-keys）

- BUILD\_FINGERPRINT := Mobicel/V4/V4:$(PLATFORM\_VERSION)/$(BUILD\_ID)/

$(BF\_BUILD\_NUMBER):$(TARGET\_BUILD\_VARIANT)/$(BUILD\_VERSION\_TAGS)

+BUILD\_FINGERPRINT := Mobicel/V4/V4:$(PLATFORM\_VERSION)/$(BUILD\_ID)/

$(BF\_BUILD\_NUMBER):$(TARGET\_BUILD\_VARIANT)/release-keys

特殊：

①检查Makefile中build\_desc的$(BUILD\_VERSION\_TAGS)是否固定为release-keys。

②每次都要检查一下BUILD\_FINGERPRINT中属性值是否和修改固定的一致

如：BF\_BUILD\_NUMBER **:≠** $(BUILD\_NUMBER)，那么①就要添加BF\_BUILD\_NUMBER := 1568104533

示例：http://172.16.5.189/#/c/110757/

补充：

旧版本修改方式:(李进总结)

先在对应订单目录的buildinfo.sh查找fingerprint属性值的组成方式

，从本文件和configs.mk文件中查找对应的宏，如果找不到去device目录grep搜索

没有结果则从build/make/core/Makefile文件中查找本订单fingerprint赋值的位置，对

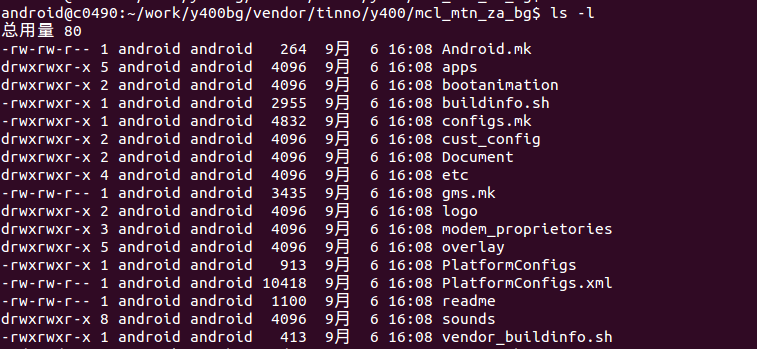
BUILD\_NUMBER、TAGERT\_BUILD\_VARIANT和release-keys 进行修改，如果是取宏则在订单目录修改，

release-keys可直接在buildinfo强改，并且/build/make/core/Makefile文件对应fingerprint也要修改。

注：如果TARGET\_BUILD\_VARIANT值是编译脚本中定义为直接从输入编译命令取值则不用修改。

26.订单目录简介

配置文件：作用、路径



Android.mk： include all subdir makefile

apps/： 预置客户APP

bootanimation： 开关机动画资源

buildinfo.sh： 项目相关定义的ro属性配置文档

configs.mk： 控制device的宏定义，等订单项目配置makefile文件

etc： 手机网络频段信息设置等相关信息

gms.mk： gms相关配置信息makefile文件

logo： 手机开机第一帧图片和电池电量低图片

overlay： 相关资源替换目录

sound： 手机铃声、通知、闹钟等音频文件配置信息

platformConfigs： 手机系统属性值配置文件

vendor/tinno/component

├── common

│   └── apn\_common

├── config

│   ├── apps\_config.mk //apk打开关闭配置

│   ├── build\_config.mk //编译配置

│   ├── device\_config.mk //项目配置，共用的

│   └── sprdcp\_config.mk //sprd编译前复制配置

├── modem

│   ├── k101\_trunk\_modem //modem命令，不同的目录不同的订单，用MODEM\_DIR := k101\_trunk\_modem 来定义，

│   └── modem.mk //modem的复制命令

├── opensource //开源代码

├── requirment //订单需求

├── tinnoapps //apps

│   └── ProductInfo

└── tool

vendor

分单项目目录补充

reorder\_config：翻单定制文件夹

27.Google规范暗码讲解

暗码：各常用暗码，及所进页面的作用（apk），常用测试或验证情景（Google规范暗码：\*#\*#暗码#\*#\*）

0661 ：查看设备信息（LCD物料信息、TP物料信息、设备有的sensors、camera物料信息等）

86436 ：查看版本号（客户版本号，内部版本号，基线版本号等）

8 ：开机下的工厂测试（或称ftm测试，整机测试）

Ps ：关机下ftm测试--power+音量下键

84666364：工程模式--ylog，log level switch等...

\*#06# ：手机国际识别码（IMEI）

1. jack编译报错的问题

错误信息：

ERROR: Communication error with Jack server (52) make

解决方法：

工程根目录内执行以下三句

export JACK\_JAR=./out/host/linux-x86/framework/jack.jar

./out/host/linux-x86/bin/jack-admin stop-server

./out/host/linux-x86/bin/jack-admin start-server

<http://www.mamicode.com/info-detail-2162318.html>

1. 出现Try increasing heap size with java option '-Xmx'错误解决方案

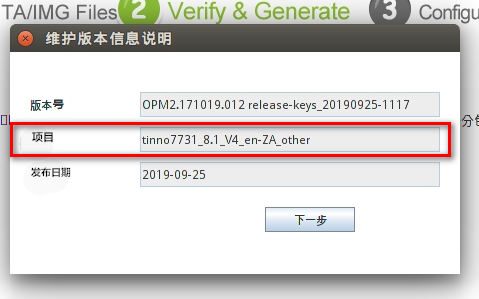
出现这个错误是由于电脑内存不足，在命令行分别执行以下三条语句，然后继续编译

export JACK\_SERVER\_VM\_ARGUMENTS="-Dfile.encoding=UTF-8 -XX:+TieredCompilation -Xmx4g"

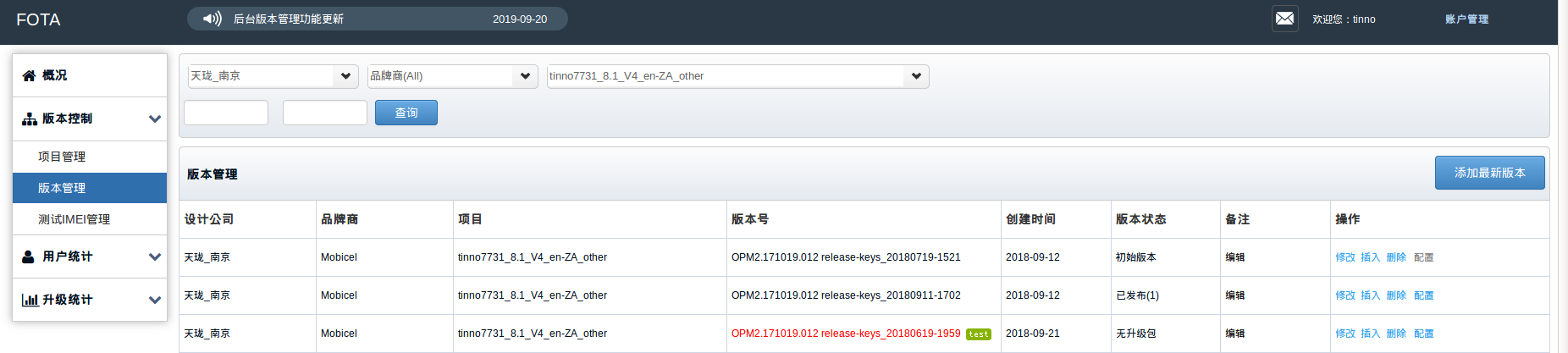
./prebuilts/sdk/tools/jack-admin kill-server

./prebuilts/sdk/tools/jack-admin start-server

<https://blog.csdn.net/csdn_zyp2015/article/details/56277765>

1. ota包上传服务器失败后，进入后台手动上传图解
2. 进入后台后，在项目管理中，搜索当前项目名，项目名获取方式如下：

②搜索页面如下：

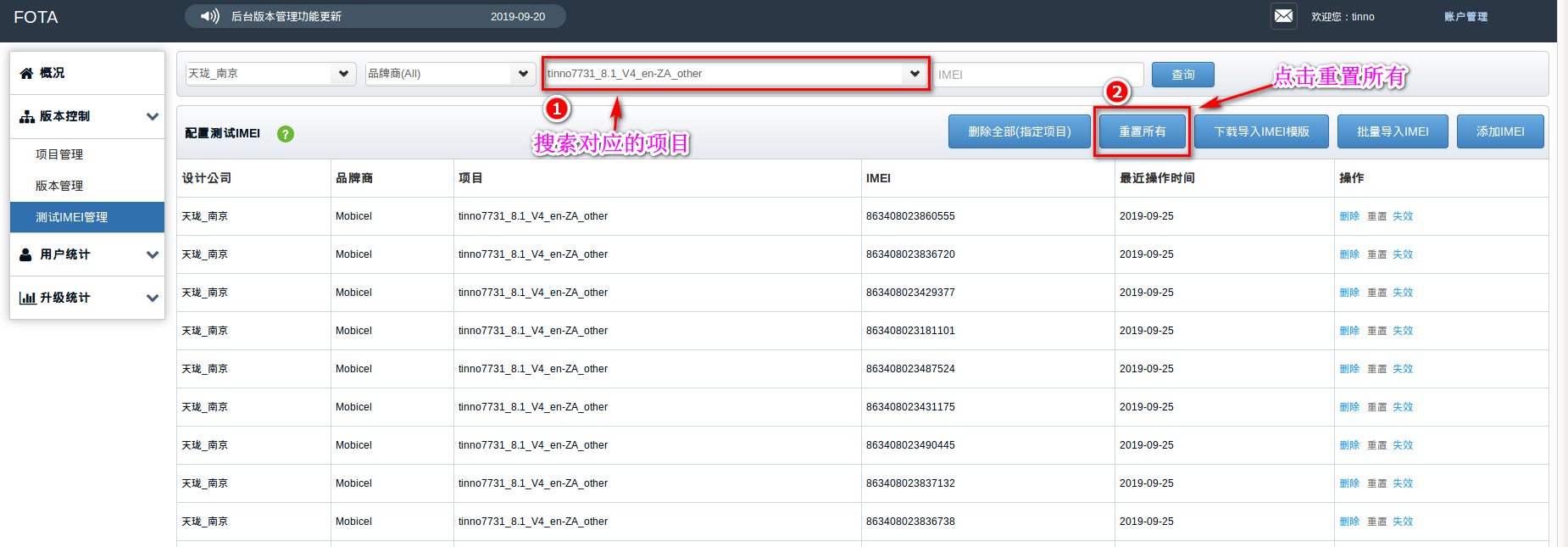


1. 看发布日期选择正确的项目，点击配置按钮



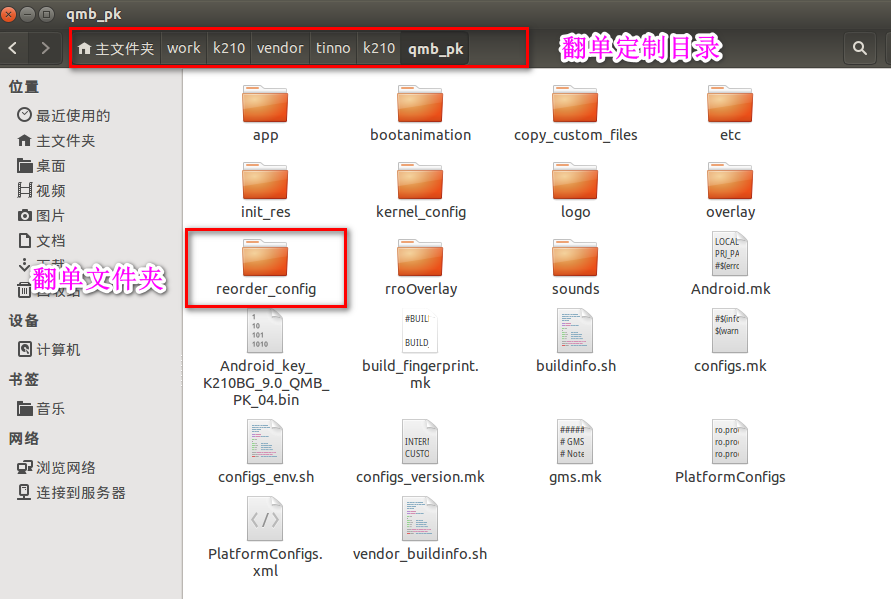
1. 进入配置页面后，按照图示操作,上传update.zip时要选择手机上个版本停留更新的源版本位置，不要无脑的选中最后一个（补充：如果手里没有手机，就在上版本源代码输出的out/target/product/k101/system下grep搜索fota，可以看到ro.fota.version=OPM2.171019.012 release-keys\_20191009-0955就是上个版本的最新升级停留位置）

  
⑤点击测试IMEI管理，在搜索栏搜索对应项目，再点击重置所有

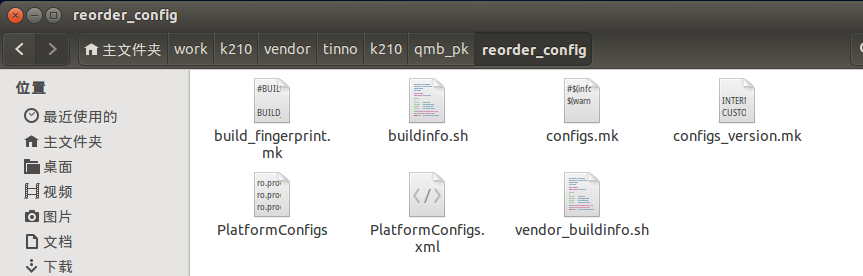


31.翻单项目如何初始化

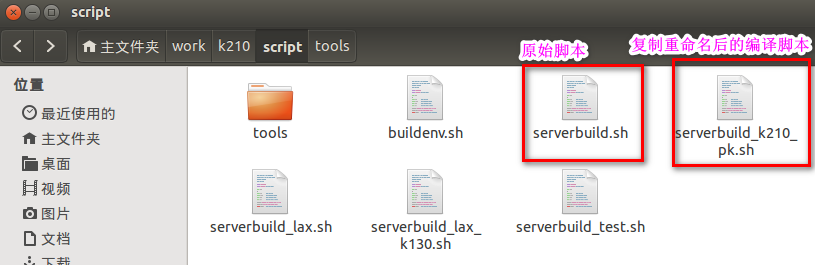
①首先在要进行翻单的定制化项目下新建reorder\_config文件夹，如下图所示：



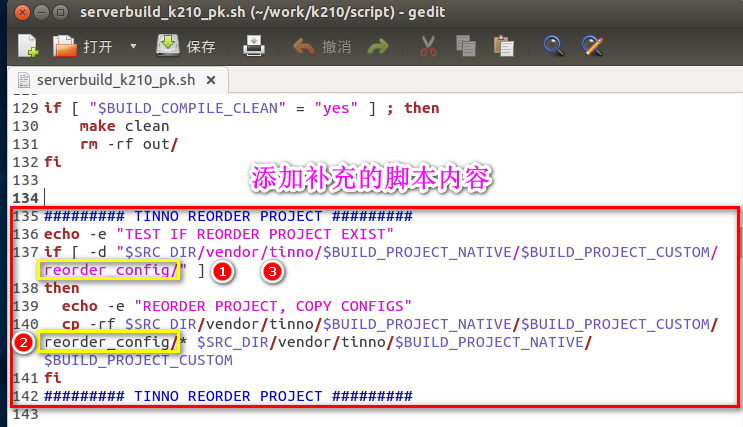
②文件夹内复制qmb\_pk翻单项目里面需要覆盖的文件，一般是如下图所示这些文件：

（各文件作用参考：[26.订单目录简介](#订单目录简介)）

1. 修改编译脚本，首先复制当前项目编译脚本进行定制化修改，如下图所示



④erverbuild\_k210\_pk.sh文件内修改内容可以参考其他同基线翻单项目修改，注意脚本中①②两处修改为自己翻单文件夹，修改内容如下:



代码原文：

######### TINNO REORDER PROJECT #########

echo -e "TEST IF REORDER PROJECT EXIST"

if [ -d "$SRC\_DIR/vendor/tinno/$BUILD\_PROJECT\_NATIVE/$BUILD\_PROJECT\_CUSTOM/reorder\_config/" ]

then

echo -e "REORDER PROJECT, COPY CONFIGS"

cp -rf $SRC\_DIR/vendor/tinno/$BUILD\_PROJECT\_NATIVE/$BUILD\_PROJECT\_CUSTOM/reorder\_config/\* $SRC\_DIR/vendor/tinno/$BUILD\_PROJECT\_NATIVE/$BUILD\_PROJECT\_CUSTOM

fi

######### TINNO REORDER PROJECT #########

添加代码含义：

把翻单文件夹的内容覆盖定制项目下相同内容，从而达到编译后的项目属于订单项目，注意，当运行翻单脚本后，原始项目文件会被覆盖掉，注意git checkout . 当前项目，查看原始项目数据

32.验证翻单项目GMS，security安全补丁，以及修改点部分操作

首先还原翻单定制化脚本覆盖掉的内容

android@Precision-T1700-c0508:~/work/K210/vendor/tinno/k210/qmb\_pk$ git checkout .

android@Precision-T1700-c0508:~/work/K210/vendor/tinno/k210/qmb\_pk$ git status .

连接手机查看修改点

android@Precision-T1700-c0508:~/work/K210/vendor/tinno/k210/qmb\_pk$ adb shell getprop |grep 2020

查看手机gms更新时间是否为最新时间

android@Precision-T1700-c0508:~/work/K210/vendor/tinno/k210/qmb\_pk$ adb shell getprop |grep gms

查看项目中的gms时间

android@Precision-T1700-c0508:~/work/K210/vendor/tinno/k210/trunk$ ls

bootanimation build\_fingerprint.mk buildinfo.sh configs\_env.sh configs.mk configs\_version.mk etc gms.mk kernel\_config logo overlay PlatformConfigs sounds vendor\_buildinfo.sh

android@Precision-T1700-c0508:~/work/K210/vendor/tinno/k210/trunk$ cat gms.mk |grep version

ro.com.google.gmsversion=9\_201902.go

查看git提交有没有最新修改

android@Precision-T1700-c0508:~/work/K210/vendor/tinno/k210/trunk$ gitk gms.mk &

[1] 6007

查看项目代码中security安全补丁时间

android@Precision-T1700-c0508:~/work/K210/vendor/tinno/k210/trunk$ cat gms.mk |grep security

[1]+ 已完成 gitk gms.mk

查看手机中security安全补丁时间

android@Precision-T1700-c0508:~/work/K210/vendor/tinno/k210/trunk$ adb shell getprop |grep security

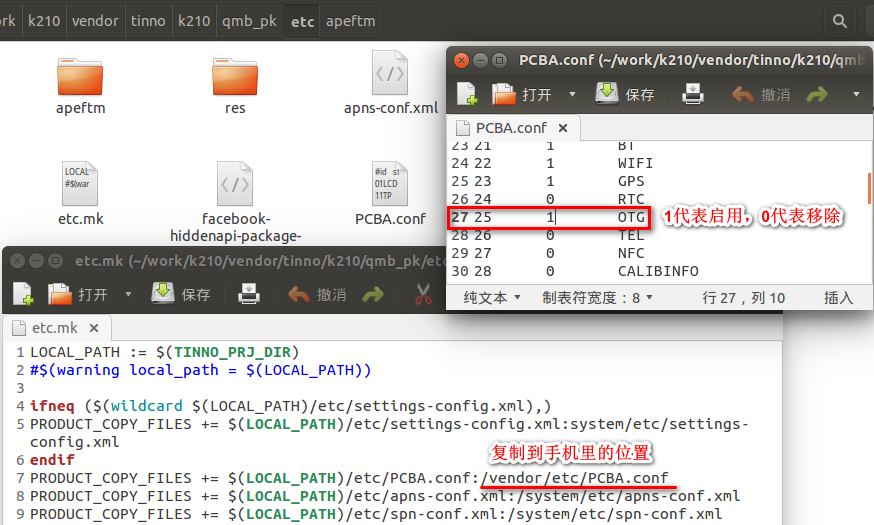
33. 修复FTM apk（开机）和FTM（关机）没有OTG测试项

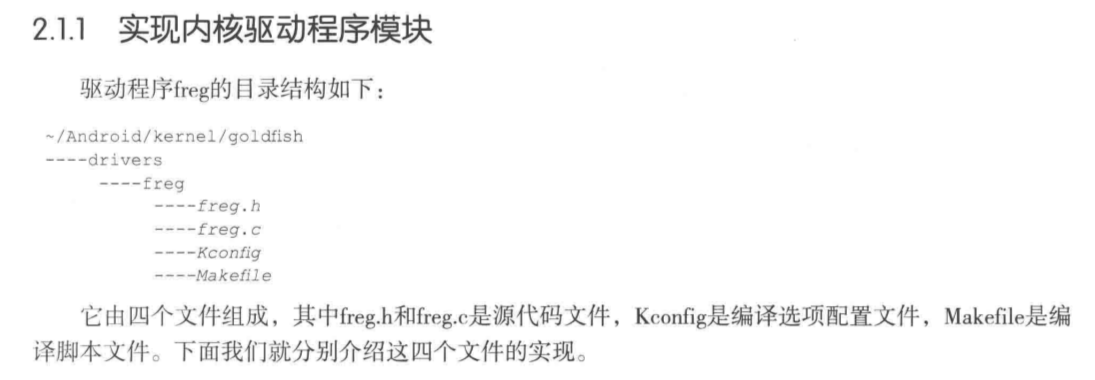
开机模式下：按\*#\*#8#\*#\*查看是否有OTG选项

关机模式下：按音量下键和电源键查看

①修复开机模式：打开vendor/tinno/k210/qmb\_pk/etc/apeftm，将ftm-config.xml中如图所示OTG值改为1，将ftm-config.xml push到apeftm.mk中所示位置，就可以本地验证是否修改成功

②修复关机模式：打开vendor/tinno/k210/qmb\_pk/etc，将PCBA.conf中如图所示OTG值改为1,将PCBA.coonf push

到etc.mk中所示位置，就可以本地验证是否修改成功

Android源码阅读笔记

Android UI体验之全屏沉浸式透明状态栏效果

<https://www.cnblogs.com/xieqing/p/6515888.html>

sts需要刷userdebug版本固定fingerprint

gms，vts是一起跑的，cts，gts是一起跑的，verifier是去vts目录下取apk安装手动测试的

我司的项目是不支持ab升级（不一定），是32位的，所以选择镜像文件是：

system-aosp\_arm\_a-4632655.zip

用刷机工具单选systen，选择解压的image，进行刷机

K101BG\_8.1\_MCL\_MTN\_ZA\_FIRST，MR版本工作流程：

①首先如果要求回退到smr版本就回退到smr版本

②所有首单私人仓库的修改点都要在MR版本中修改

③关于vender仓库下partner\_gms文件夹内的东西可以，以没有回退的版本为准，回退后少的文件要添加上，保证跑gms不会出错。

④要求打的补丁一定要打完，还有当月的安全补丁也要补上

⑤还有着重检查首单修改过的文件夹：devices，external，frameworks，package，system，这些文件夹的改动

⑥关于modem的回退，直接以回退版本为准

**注意：**回退版本时，SPM给出的需求及回退点不一定完全正确，检查问题时要全方面的考虑。

验证缩小问题点，关于差分包过大，可能导致的原因：gms没有回退到位，需求不完整，可能这版软件的有些apk需要增加或者减少。

附修改点原文:

vendor

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/vendor/partner\_gms

回退GMS到之前SMR的版本

vendor/tinno/component/modem

回退modem到之前SMR的版本

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

vendor/tinno/项目私有仓库

1、对比合并需求修改

2、注意自动旋转屏幕是否关闭

3、如果使用了google，记得更换为 photos

frameworks

b4fa2fc1993984753796448eb95490d888e33917

<REQ><YEAAOA-500><将Google assist 设置为default assist app>

GTS 定位PATCH：

vendor/tinno/component/modem

<BUG><BABOZA-360><K101:GTS GtsLocationTestCases模块中3个fail项>

vendor

<BUG><BABOZA-360><K101:GTS GtsLocationTestCases模块中3个fail项>

vendor仓库下，GMS相关修改：

1、注意更新GooglePackageInstaller版本

<BUG><KECAPWKA-254><K130AG\_9.0\_WIK\_TH:CLONE - WIKO OM Common REQ CVE-2019-2114(A-123700348)>

2、合并：

<BUG><KECAPWKA-254><K130AG\_9.0\_WIK\_TH:CLONE - WIKO OM Common REQ GTS 7.0\_r1 GtsNmgiarcTestCases com.google.android.comms.DialerTests#testDialerInstalledCorrectly>

3、合并：

<REQ><KBBAOBRA-2><GtsNmgiarcTestCases fail com.google.android.comms.MessagesTests#testMessagesVersionCheck>

4、合并

<REQ><KBABGBDA-127><update GMSCORE to lastest>

34.grep搜索限制命令 --exclude=\*.xml

35.Android App多语言开发 github案例

<https://github.com/Mrlove133481/AndroidMultiLanguage>

36. ProductInfo应用分析

**ProductInfo结构：**



**①AndroidManifest.xml**

* 为应用的 Java 软件包命名。软件包名称充当应用的唯一标识符。
* 描述应用的各个组件，包括构成应用的 Activity、服务、广播接收器和内容提供程序。它还为实现每个组件的类命名并发布其功能，例如它们可以处理的 [Intent](https://link.jianshu.com?t=https%3A%2F%2Fdeveloper.android.com%2Freference%2Fandroid%2Fcontent%2FIntent.html%3Fhl%3Dzh-cn" \t "_blank) 消息。这些声明向 Android 系统告知有关组件以及可以启动这些组件的条件的信息。
* 确定托管应用组件的进程。
* 声明应用必须具备哪些权限才能访问 API 中受保护的部分并与其他应用交互。还声明其他应用与该应用组件交互所需具备的权限
* 声明应用所需的最低 Android API 级别
* 列出应用必须链接到的库

这个应用比较特殊，因为没有声明<category android:name="android.intent.category.LAUNCHER" />，所以安装完成后没有图标和默认的启动界面

此应用是通过声明的广播接收者：.SecretCode监听特定的action进行启动特定的activity界面

**②BroadcastReceiver实现类**

实现当接收到广播发送者发送的信息时，响应具体的代码逻辑，这里是通过判断分析广播发送者发送的不同暗码信息，响应不同的activity。

**③activity实现类**

四个activity实现具体的打开展示界面，这里的activity主要通过不同的暗码，加载不同的布局文件，然后在代码中从外部不同的设备信息文件中，抓取需要的设备信息，填充到相应的布局块中进行显示。

**④资源文件**

首先就是xml文件夹下面的各种页面的布局文件，这些文件定义了最终的呈现布局，然后通过加载values文件夹里面的string.xml，填充布局文件定义的不同初始化字段。

**⑤Android.mk**

Android不同模块的编译文件，在此文件中义了不同的编译脚本信息，指定以何种方式编译模块。

**⑥其他**

诸如图标、图片文件夹drawable、activity布局文件夹layout等

**ProductInfo启动运行过程分析**

因为此应用没有图标和默认的启动界面，是通过广播接收者，监听特定的广播进而启动不同的activity。

**①首先/packages/apps/Dialer/java/com/android/dialer/app/SpecialCharSequenceMgr.java**

static boolean handleSecretCode(Context context, String input) {

// Secret codes are accessed by dialing \*#\*#<code>#\*#\*

int len = input.length();

if (len <= 8 || !input.startsWith("\*#\*#") || !input.endsWith("#\*#\*")) {

return false;

}

String secretCode = input.substring(4, len - 4);

TelephonyManagerCompat.handleSecretCode(context, secretCode);

return true;

}

handleSecretCode方法对输入的暗码进行解析，然后执行TelephonyManagerCompat.handleSecretCode方法。

**②/packages/apps/Dialer/java/com/android/contacts/common/compat/TelephonyManagerCompat.java**

private static final String SECRET\_CODE\_ACTION = "android.provider.Telephony.SECRET\_CODE";

public static void handleSecretCode(Context context, String secretCode) {

// Must use system service on O+ to avoid using broadcasts, which are not allowed on O+.

if (BuildCompat.isAtLeastO()) {

if (!TelecomUtil.isDefaultDialer(context)) {

LogUtil.e(

"TelephonyManagerCompat.handleSecretCode",

"not default dialer, cannot send special code");

return;

}

context.getSystemService(TelephonyManager.class).sendDialerSpecialCode(secretCode);

} else {

// System service call is not supported pre-O, so must use a broadcast for N-.

Intent intent =

new Intent(SECRET\_CODE\_ACTION, Uri.parse("android\_secret\_code://" + secretCode));

context.sendBroadcast(intent);

}

}

此方法通过发送一条SECRET\_CODE\_ACTION，并携带Uri数据形式的广播，把消息传播出去。

**③AndroidManifest.xml中注册的广播接收者.SecretCode响应监听到的SECRET\_CODE\_ACTION这条广播。**

<receiver android:name=".SecretCode">

<intent-filter>

<action android:name="android.provider.Telephony.SECRET\_CODE"/>

<data android:scheme="android\_secret\_code" android:host="74657799" />

。。。。。。

</intent-filter>

</receiver>

广播接收者SecretCode接收到SECRET\_CODE\_ACTION这条广播，并对定义的data数据相匹配，如果匹配成功则进入SecretCode这个实现类中执行相应的的方法。

**④广播接收者SecretCode.java实现类**

@Override

public void onReceive(Context context, Intent intent) {

if (intent.getAction().equals(SECRET\_CODE\_ACTION)) {

final String SECRET\_CODE\_ACTION = "android\_secret\_code://";

Uri uri = intent.getData();

String str = null;

if (uri != null) {

str = uri.toString().substring(SECRET\_CODE\_ACTION.length(), uri.toString().length());}

if("74657799".equals(str)) {

try { Intent i = new Intent("action.com.tnmb.productinfo.VersionInfo");

i.putExtra("edit\_input", "\*#\*#" + str + "#\*#\*");

i.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK); // must

context.startActivity(i);

} catch (Exception e) {android.util.Log.d("InfoProject", "uri = " + uri.toString() + "; str = " + str);}}

。。。。。。

}

通过监听到的不同暗码，通过Intent隐式启动action.com.tnmb.productinfo.VersionInfo这个activity

**⑤VersionInfo.java，activity实现类**

@Override

public void onCreate(Bundle icicle) {

super.onCreate(icicle);

Bundle extras = getIntent().getExtras();

if(extras != null){

String input = extras.getString("edit\_input");}

if (input.startsWith("\*#\*#563412") && input.endsWith("#\*#\*")) {

Log.e(LOG\_TAG, "WikoVersionInfo:onCreate");

addPreferencesFromResource(R.xml.wiko\_de\_version\_info);

setValueSummary(KEY\_WIKO\_VERSION, "ro.tcustom.build.version"); }

。。。。。。

}

VersionInfo类通过Intent传递的不同暗码加载不同的布局文件如R.xml.wiko\_de\_version\_info，然后在系统中获取需要显示的设备信息，通过setValueSummary()方法设置到最终显示的布局页面上。

ProductInfo其他activity页面启动过程与上述大同小异。

模块添加编译过程

①./Android.mk

②tinno/component/config/apps\_config.mk:4:PRODUCT\_PACKAGES += ProductInfo

③device/sprd/pike2/k121/k121.mk:447:include vendor/tinno/component/config/apps\_config.mk

**ProductInfo优化与改进**

①代码冗余太多，对于没有使用的暗码和代码，应该及时删除，避免在分析代码的时候引起歧义混淆，还有就是暗码规范，不能随意使用特定系统暗码，如：06，查看IMEI号暗码。

②因为监听的是系统广播发送者发送的action，而系统发送的是无序广播，所以当外部广播接收者监听到相同字段的暗码的时候，会同时响应。所以最好自己实现暗码的监听，发送有序广播，在接收到广播后，阻断其传播，避免广播的混乱。

③暗码太多，在没有使用文档的帮助下，很难知道具体的暗码，并且对应的页面具体显示的是什么设备信息。所以可以设置一个统一入口界面，分门别类的建立各个子页面，一目了然，而不是输入一大堆记不住的暗码。

37.Linux下Shell命令的输出信息同时显示在屏幕和保存到日志文件中

#直接覆盖日志文件

ls -l | tee ./t.log

#将输出内容附加到日志文件

ls -l | tee -a ./t.log

<https://www.cnblogs.com/EasonJim/p/8440725.html>

38.更正历史提交用户名和邮箱

#!/bin/sh

#更改当前文件下git仓库提交的用户名和邮箱

echo "[user]

name = Mrlove133481

email = 1334819077@qq.com" >> ./.git/config

#更改提交中所有邮箱为OLD\_EMAIL的为新的用户名和新的邮箱

git filter-branch --env-filter '

OLD\_EMAIL="zhangshuai.qin@tinno.com"

CORRECT\_NAME="Mrlove133481"

CORRECT\_EMAIL="1334819077@qq.com"

if [ "$GIT\_COMMITTER\_EMAIL" = "$OLD\_EMAIL" ]

then

export GIT\_COMMITTER\_NAME="$CORRECT\_NAME"

export GIT\_COMMITTER\_EMAIL="$CORRECT\_EMAIL"

fi

if [ "$GIT\_AUTHOR\_EMAIL" = "$OLD\_EMAIL" ]

then

export GIT\_AUTHOR\_NAME="$CORRECT\_NAME"

export GIT\_AUTHOR\_EMAIL="$CORRECT\_EMAIL"

fi

' -f --tag-name-filter cat -- --branches --tags #-f为强行覆盖

#强行推送所有修改到主分支

git push origin master –force

<https://blog.csdn.net/hello5orld/article/details/51386218>

<https://help.github.com/en/github/using-git/changing-author-info>

39.远程拷贝

scp -r \* 172.19.160.70:文档/AndroidQ/

40.使用Android SDK 查看apk信息

命令程序位置

android\_sdk/build-tools/version/

命令示例

./aapt dump badging apk\_path

查看手机内相关应用版本号

adb shell pm dump com.google.android.setupwizard（包名） | grep "versionName"

<https://blog.csdn.net/ak47007tiger/article/details/90699134>

41.   k510cg mcl\_mtn\_za\_af的初始化，请参考

①

针对K510 南非衍生订单

    Myos:       vendor/tinno/myos/configs/k510/mcl\_mtn\_za\_ug/ （订单负责人初始化）

   RF文件:    vendor/tinno/hw/k510bg/rf/mcl\_mtn\_za\_ug/ （硬件射频初始化）

   Kernel 2个配置文件：  kernel-4.4/arch/arm/configs/k510\_mcl\_mtn\_za\_ug\_defconfig   （订单负责人初始化）

                  kernel-4.4/arch/arm/configs/k510\_mcl\_mtn\_za\_ug\_debug\_defconfig           （订单负责人初始化）

分区配置（仅南非订单）：android/device/tinno/k510/PartitionOverlay\_mcl\_mtn\_za\_ug.mk   （订单负责人初始化）

新建衍生项目后需要自行确认上述仓库是否初始化，camera最容易遗忘。

②

参照一个订单仓库如mcl\_mtn\_za\_ug，复制里面的所有内容

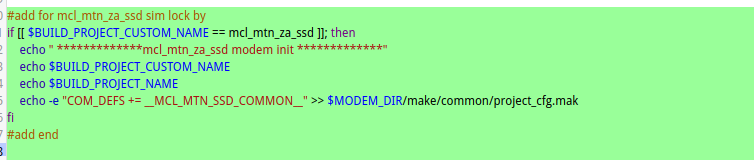
到自己的新仓库下。并且修改所有\_ug为\_af

注意看设备名称是否也要修改，如DANY\_RW,修改为DANY\_ZAM

42. 锁卡功能需求

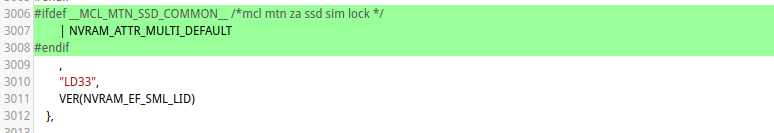
MTK平台

1、首先是修改build/tinno\_script/build\_modem.sh 在这个编译文件中根据订单来添加锁卡的需要的宏。



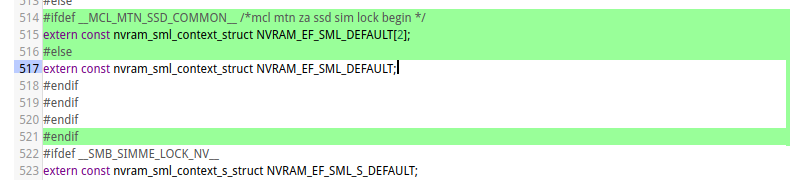
2、然后在modem/mcu/pcore/custom/service/nvram/l4\_nvram\_def.c文件中根据已经定义好的宏和项目是单卡还是多卡来修改。如果是多卡，则需要添加多卡的配置。并且在这个文件中添加对于SMS FDN的控制（如需要则添加控制）。



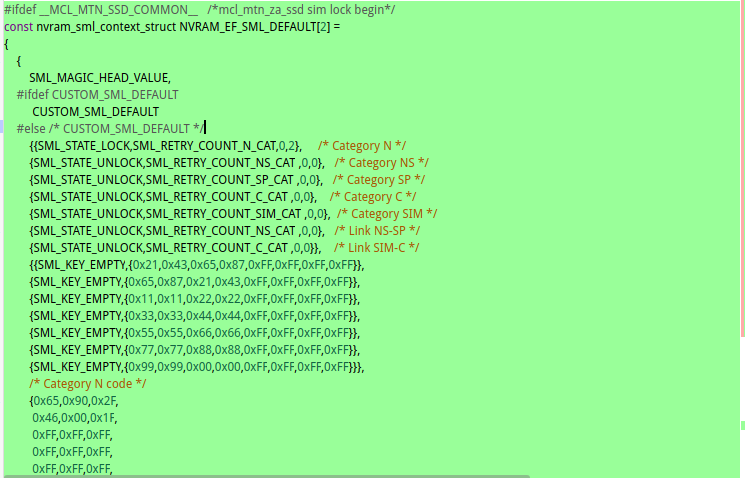


————————————————————————————————————————

modem/mcu/pcore/custom/service/nvram/custom\_nvram\_sec.h，在这个文件中添加需要锁卡的数量，如果是锁单卡，则不需要修改，如果是多卡，则需要添加判断，并且需改数组大小。



最后在modem/mcu/pcore/custom/service/nvram/custom\_nvram\_sec.c这个文件进行锁卡白名单以及锁卡依赖的添加。



如果是多卡项目，并且锁两张卡：

卡2依赖卡1，则这里要写两个。

卡2不依赖卡1，则这里只有一个是卡1的 并且之前的关于锁卡数量的地方也的进行更改为1.并且卡2的锁卡根据项目需求在代码中需改

3、根据项目来修改锁卡框的弹出以及多卡项目的过期卡判断 mediatek/proprietary/packages/apps/SystemUI/src/com/android/keyguard/KeyguardUpdateMonitor.java

frameworks/base/core/java/android/os/SystemProperties.java

实例：

k510 mcl\_mtn\_za\_ug项目：

1、卡2依赖卡1，并且卡2是非白名单的时候限制2G网络

（1）、首先是依赖，在modem中的三个文件中进行配置。

（2）、2G网络限制

framework/opt/telephony/src/java/com/android/internal/telephony/PhoneFactory.java

判断卡2不是白名单，就将网络限制为2G

mediatek/proprietary/packages/services/Telephony/src/com/android/phone/MobileNetworkSettings.java 这个文件将设置中的切换网络的按钮置为不能点击

k510 mcl\_mtn\_za\_ssd项目：

1、卡2依赖卡1，并且卡2是非白名单的时候卡2只能通话

（1）、首先是修改mediatek/proprietary/packages/apps/MtkSettings/src/com/android/settings/sim/SimDialogActivity.java这个文件，将设置中的SIM卡选项中的移动数据选项固定为卡1。

（2）、framework/opt/telephony/src/java/com/android/internal/telephony/PhoneFactory.java这个文件，将卡2的数据模式置为2G

（3）、mediatek/proprietary/packages/services/Telephony/src/com/android/phone/MobileDataPreference.java修改这个文件将设置中的网络选项中的卡2的数据开启选项置灰并且将卡2的数据流量给关闭掉。

（4）、mediatek/proprietary/packages/services/Telephony/src/com/android/phone/MobileNetworkSettings.java修改这个文件将设置中的切换网络的按钮置灰，使其不能进行网络的选择

（5）、mediatek/proprietary/frameworks/opt/telephony/src/java/com/mediatek/internal/telephony/dataconnection/  
MtkDcTracker.java在这个文件中再次限制卡2的数据，使得卡2不能使用数据

（6）、mediatek/proprietary/packages/apps/SystemUI/src/com/android/systemui/qs/customize/TileAdapter.java修改这个文件移除下拉通知栏里面的快捷切换卡1卡2数据流量的按钮

展锐平台

1、展锐平台的锁卡配置基本上都是在对应的NV工程中配置的，包括锁MNC、MCC以及锁网、卡槽依赖、过期卡限制等等功能。

展锐平台一共有5中锁，这5种锁可以单独使用也可以组合使用。在组合使用的时候，modem检测的顺序如下：

SIM锁-->集团锁-->服务供应商锁-->网络子集锁-->网络锁

2、除了在NV工程中的配置之外，还会根据项目的需求在java代码中进行修改。

实例：

K110EG：

实例提交记录：[http://172.16.5.189/#/c/118726/](http://172.16.5.189/%23/c/118726/)

1、锁定卡1，不锁定卡2，卡1和卡2无关联并且卡2为非白名单时在android端限制为2G网络。

sprd/platform/frameworks/base/ex-interface/core/java/android/telephony/TelephonyManagerEx.java

vendor/sprd/platform/frameworks/base/ex-interface/telephony/java/android/telephony/CarrierConfigManagerEx.java

这两个文件中定义了判断卡是否满足锁卡规则的函数，并且给定了默认值为满足

vendor/sprd/plugins/packages/services/Telephony/addons/SimLockSupport/src/plugin/sprd/simlock/SimLockManagerPlugin.java 修改此文件去掉锁卡弹出框

vendor/sprd/platform/frameworks/opt/telephony/src/java/com/android/internal/telephony/uicc/policy/PrimaryCardController.java 修改此文件对卡2为非白名单时候的网络限制为2G

vendor/sprd/platform/packages/services/Telephony/src/com/android/phone/ExtendedNetworkSettings.java 修改此文件 去除网络类型选择项。

43. mtk做差分包

**①**编译软件,再编ota； make otapackage

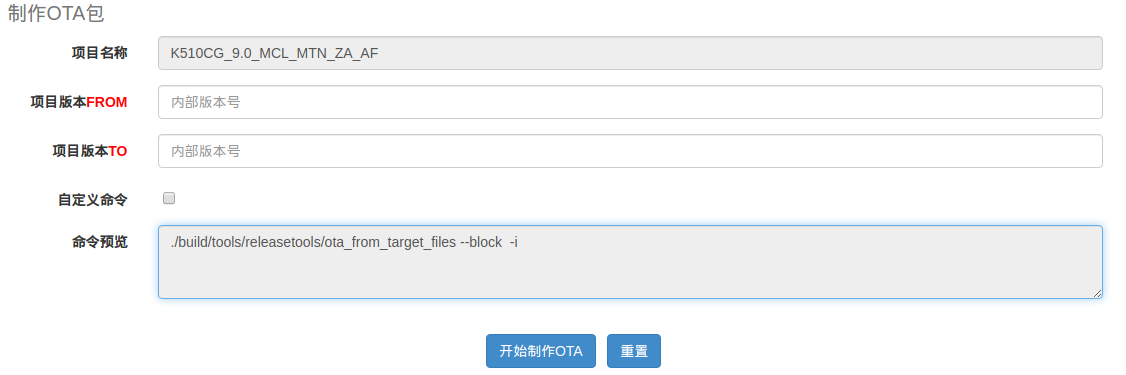
**补充：**如果编译过otapackage制作环境，则同Android版本的编译环境可以通用

**②**拷贝两个版本target包到项目根目录

**③**<编译命令> <target版本一.zip> <target版本二.zip> <升级包.zip>

**例如：**./build/tools/releasetools/ota\_from\_target\_files -s ./device/mediatek/build/releasetools/mt\_ota\_from\_target\_files --block -k ./device/mediatek/common/security/mt6580/releasekey -i V3760AN\_SYM\_BD\_BR1\_06\_TARGET.zip V3760AN\_SYM\_BD\_BR1\_07\_TARGET.zip V3760-update6\_7.zip

**编译命令可以在项目的ota制作界面取：**



44.公司常用地址





45.user版本获取root权限的命令

前提条件，用build/tinno\_script/tinno\_build.sh编译时需要带入debug参数

执行步骤：

echo tinno\_debug > /tmp/tinno\_debug

adb reboot bootloader

sudo fastboot flash para /tmp/tinno\_debug

sudo fastboot reboot

1. TSDS4.0系统开发：新增OTA制作功能



1. 预置APK

1.根据apk的名称在：vendor\tinno\common\packages\apps\下建立相应的文件夹，如：

vendor\tinno\common\packages\apps\ApeWeather；

把apk拷贝到该文件夹下，如：ApeWeather3\_GO\_V9.0.10.02.apk；

2.编写相应的Android.mk文件，如：

LOCAL\_PATH := $(my-dir)

include $(CLEAR\_VARS)

LOCAL\_MODULE := ApeWeather

LOCAL\_MODULE\_TAGS := optional

LOCAL\_SRC\_FILES := ApeWeather3\_GO\_V9.0.10.02.apk

LOCAL\_MODULE\_CLASS := APPS

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

LOCAL\_CERTIFICATE := PRESIGNED

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

include $(BUILD\_PREBUILT)

1)LOCAL\_CERTIFICATE 设置apk的签名，如果apk是自签名的，就用PRESIGNED,

如果是平台签名就用platform，这关系到apk的升级，不要设置错误；

2)LOCAL\_MODULE\_PATH 设置apk的目标路径，项目预制的apk，一般放在/product分区下，

其中：

TARGET\_OUT\_PRODUCT\_APPS 对应 /product/app;

TARGET\_OUT\_PRODUCT\_APPS\_PRIVILEGED 对应 /product/priv-app;

3)若应用要设为可卸载的，有两种方法，建议统一用方法二：

方法一是和之前的方法类似，将应用放在 /product/opertor/app下，

如：LOCAL\_MODULE\_PATH := $(TARGET\_OUT\_PRODUCT)/opertor/app;

方法二是在\vendor\tinno\common\res\txt\pms\_sysapp\_removable\_product\_list.txt中

添加apk的包名即可，如：com.ape.weathergo，在这里添加是整个平台都适用的，若项目自己特有的，可添加：

vendor/tinno/product/$(TINNO\_TARGET\_PROJECT)/$(ODM\_PROJECT\_NAME)/txt/pms\_sysapp\_removable\_product\_list.txt

文件，如：vendor/tinno/product/k510/trunk/txt/pms\_sysapp\_removable\_product\_list.txt，在其中添加包名，

注意，这个文件会覆盖平台的，所以建议拷贝平台的文件过来，在此基础上再修改；

3.在文件\vendor\tinno\common\configs\TinnoProjectConfig.mk中定义是否预制该apk的宏，如：

MYOS\_APEWEATHER = yes

这个宏对平台都是适用的，若在项目中，不需要预制该apk，则可在项目的设置文件如：

\vendor\tinno\product\k510\trunk\OdmProjectConfig.mk 设置该宏为no即可，如：

MYOS\_APEWEATHER = no

4.在\vendor\tinno\common\configs\tinno\_common.mk中设置是否预制该apk的脚本，如：

ifeq ($(strip $(MYOS\_APEWEATHER)), yes)

PRODUCT\_PACKAGES += ApeWeather

endif

48.订单配置MTK默认开关

订单配置的文件放在vendor/tinno/product/$(TINNO\_TARGET\_PROJECT)/$(ODM\_PROJECT\_NAME/OdmProjectConfig.mk,

目前代码已经可以直接配置该文件直接对MTK默认的开关的控制。

拿msensor做例子说明，比如，K510 wik\_fr订单需要去掉msensor，操作如下：

1.编辑vendor/tinno/product/k510/wik\_fr/OdmProjectConfig.mk文件，添加

CUSTOM\_KERNEL\_DEFCONFIG=yes

CUSTOM\_KERNEL\_MAGNETOMETER=no

CUSTOM\_KERNEL\_ORIENTATION\_SENSOR=no

其中CUSTOM\_KERNEL\_MAGNETOMETER和CUSTOM\_KERNEL\_ORIENTATION\_SENSOR是在device/tinno/k510/ProjectConfig.mk上有定义。

特别提醒：CUSTOM\_KERNEL\_DEFCONFIG=yes这个宏是该功能的开关，要定制的话，一定要设置为yes.

2.在kernel-4.4/arch/arm/configs/目录下添加

k510\_wik\_fr\_debug\_defconfig

k510\_wik\_fr\_defconfig

两个文件;

3.修改

k510\_wik\_fr\_debug\_defconfig

k510\_wik\_fr\_defconfig

# CONFIG\_CUSTOM\_KERNEL\_MAGNETOMETER is not set

PS：defconfig文件的去掉修改是# CONFIG\_CUSTOM\_KERNEL\_MAGNETOMETER is not set，而不是CONFIG\_CUSTOM\_KERNEL\_MAGNETOMETER=n,要不然会编译会报错的;

这样就完成了msensor去掉的操作，当然这个例子是涉及到驱动的需要2,3步，如果不是驱动修改，则执行第1步就可以了。