# 培训文档

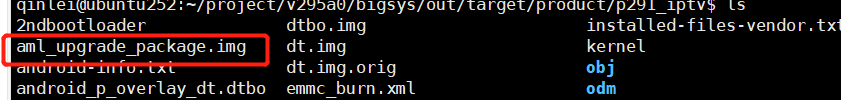
# 技能相关

## 大小系统目录结构以及编译的产物

### 大系统编译出来的文件目录：out/target/product/p291\_iptv/

p291\_iptv是amlogic平台的，其他平台有变化

aml\_upgrade\_package.img 是大系统编译出来的镜像，没有小系统的定制，（也称为基线版本）



小系统目录结构：

apk：是一个单独的仓库，存放西游记全家桶，VDI产品所需apk,由apk同事负责更新(王桂东、黄俊杰)

Makefile:本地版本编译文件

platform/build: 小系统编译所需脚本和文件

本地编译脚本：make.sh文件

版本上传脚本：svn\_git\_commit.sh，本地编译不会调用，ci会

on-project目录：小系统项目定制的存放目录

on-project/build: 小系统定制型号、属性等文件目录

build\_id.mk 项目型号、芯片型号、软件版本号在这修改

build\_prop.sh 添加或者定制安卓属性

ci\_amlogic905l3a\_emmc\_base.sh CI服务器编译调用的脚本

project\_priv.sh 在out/target/target-2015880044/下创建pub目录下所有存在的文件夹，并把根目录的apk、pub下的所有文件复制过去，准备进行编译

on-project/pub目录：目录结构同release仓库一致 platform/release

platform/release:是大系统编译出来的最终文件

## 大系统编译

### 完整编译 根目录执行命令

/build/make/make.sh 是本地编译调用的脚本

/build/ci-build.sh CI编译调用的脚本

### 模块编译

针对哪里有修改，就编译哪里，Android.mk 或者Android.bp文件的目录都可以进行模块编译

## 小系统编译流程以及相关编译文件

Makefile->调用make.mk->调用make.sh->调用project\_priv.sh

## 差分包编译

什么是差分包：就是和指定版本的差异包

怎么编译：修改platform/on-project/build/build\_id.mk

BUILD\_NUMBER:当前版本号

PRE\_NUM:就是要从那个版本还是计算差异

PRE\_BUILD\_NUMBER\_ARRAY：差分包版本号数组，写几个就会编译几个差分包

注意事项：差分包编译依赖platform/out\_user/target/下面的版本target包，make编译的时候如果make clean ,就会编不出来差分包(本地)

小系统CI编译没有 make clean 选项，默认保留target文件，可以编出差分包

## 大小系统文件配合过程

大系统CI编译会把编译出来的最终文件上传到release仓库，小系统在release目录拉取一下代码，就可以更新到大系统的修改

## 验证大系统某个修改

#### framework修改

out/target/product/p291\_iptv/system/framework一整个目录压缩，然后删除掉板卡里的framework目录，解压替换，重启

### 某个bin文件或者库修改

直接推送到板卡上替换验证

### 不明确修改之后编译出来的是什么

用本地大系统编译出来的最新的产物，替换掉release目录下的目录，编译本地版本验证

### 文件复制脚本编写

去大系统/build/ci-build.sh里面抄复制流程然后改改

把这个函数复制下来Copy\_File\_To\_Release()，看这个函数里面需要什么变量还有函数，都复制下来，修改成自己本地的路径就行了

## 验证其他修改

### 脚本文件：

比如现在板卡中 /vendor/bin/test.sh 脚本，原来的作用是输出1，现在要输出2，直接在串口在板卡中修改该脚本，保存重启即可验证

### rc文件

也是直接在板卡中修改rc文件，保存后重启查看

### apk文件

覆盖安装，重启看效果

## 版本号、项目型号修改

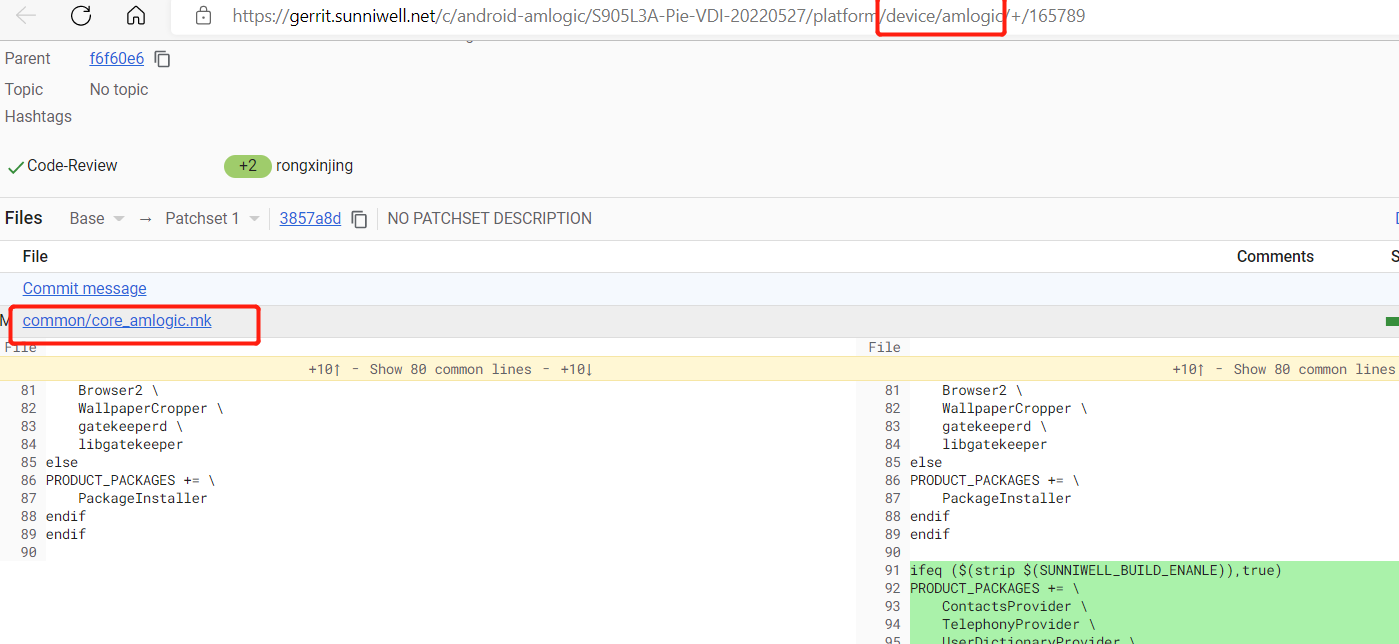
小系统中 platform/on-project/build/build\_id.mk中修改

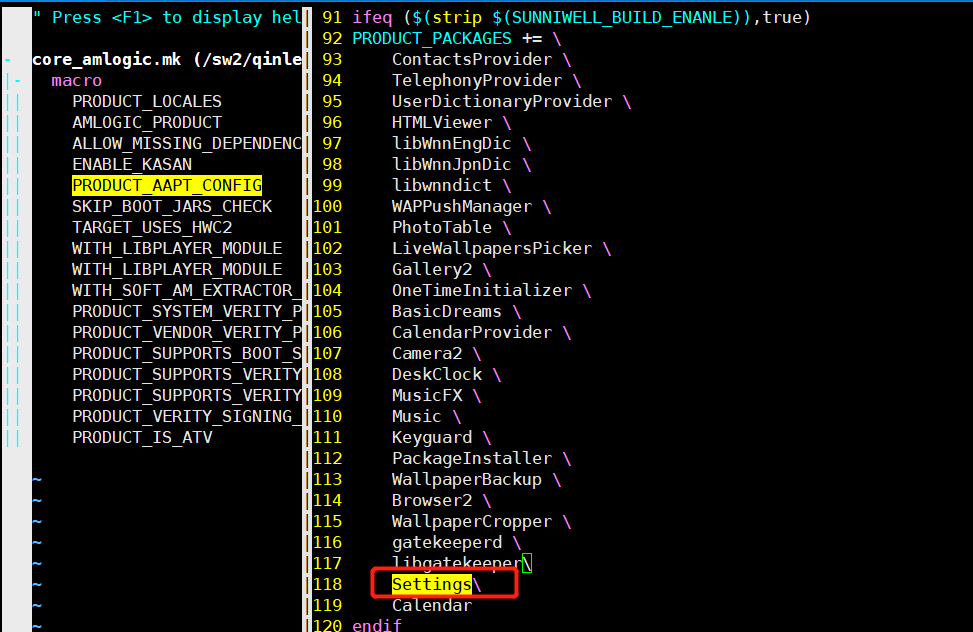
## 文件合入(举例安兔兔)

就是在 platform/on-project/pub/下面的 system 或者vendor 目录下，在app目录里创建一个文件夹，mkdir Antutu, 然后把安兔兔.apk 放到小系统 的 Antutu文件夹下，编译的时候脚本会对apk进行处理。

## 添加SDK应用

参考gerrit上提交记录/device/amlogic/common/core\_amlogic.mk就是完整的修改路径，缺什么应用，就在后面加上





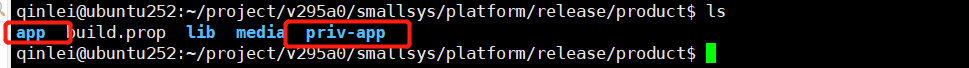
/packages/app下是sdk app源码，把需要添加的app应用的文件夹名加到上述的core\_amlogic.mk文件中，重新编译大系统就加上了

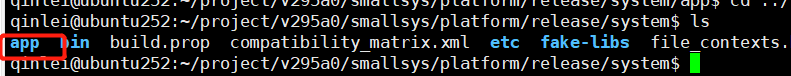
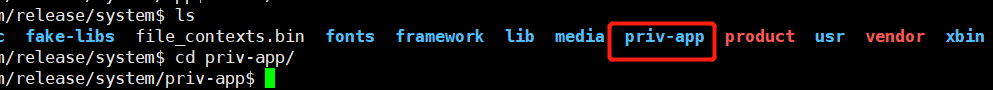
## 裁剪SDK应用

### 方法一：在sdk编译流程中删除，比如说要裁剪settings应用，就把core\_amlogic.mk中的Settings删掉

### 方法二：在小系统中裁剪

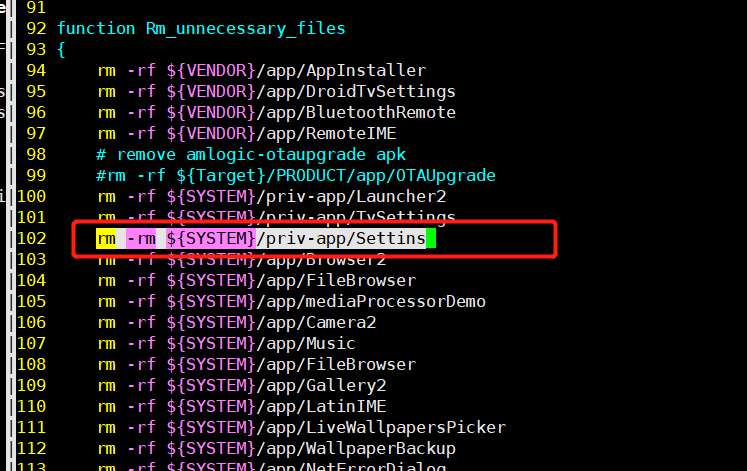
以下五个目录会存在app,裁剪之前要确认 裁剪目标在哪个文件夹下





确定目标在哪里之后，到/platform/on-project/build/中 修改 project\_priv.sh，删除对应的app



也可以裁剪app以外的文件夹

### 注意事项：需裁剪的应用不确定在哪个目录怎么处理

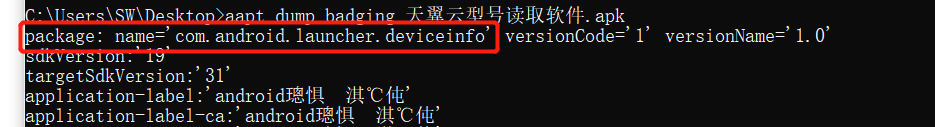
先打开应用，使用dumpsys window windows |grep "Current" 命令取获取当前窗口的包名类名，

然后去响应的文件夹中找,再到 project\_priv.sh中处理

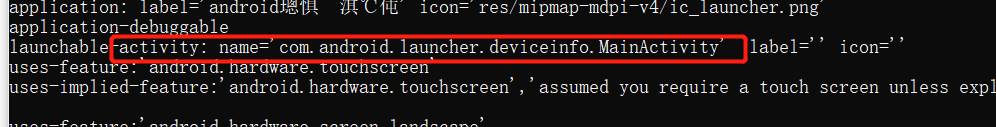
查看应用包名类名：windows下工具 aapt,和adb一样配置环境

去到apk所在路径中 输入aapt dump badging xxx.apk

包名



类名



## 安装应用

### 打开安装属性

一般是禁止用户自行安装第三方软件,如果要安装需要设置属性

setprop sys.swtms.install 1 放开安装权限

### 串口安装

把apk放到办卡中, 输入 pm install -r xxx.apk 进行安装

### adb 安装

连接上adb之后，adb窗口输入 adb install xxx.apk

### U盘安装

进入文件管理器，找到apk对应位置，双击安装就好

## 属性添加

### 三个属性分区

system：安卓版本号、蓝牙名称…

vendor：一般用于存放用户自定义属性，以后叫预置属性，基本都放这个分区

recovery 三个分区

预置属性文件： /platform/on-project/build/build\_prop.sh

set\_prop "ro.build.devicemodel" "Z81E"

### sys

掉电不保存属性，setprop sys.test.demo 1设置成1之后，重启系统，getprop就没有该属性了

### persist

掉电可保存属性, setprop persist.test.demo 1 ，重启之后getprop persist.test.demo 还是1

### ro

read only ，只读属性 ，不能通过 setprop去修改属性值，平时要调试的话，一般是在对应分区下的 build.prop修改验证

/system/build.prop

/vendor/build.prop

### adb控制属性

默认打开adb -> setprop persist.sys.usb.config adb/none 打开/关闭

## 数据表数据添加

/platform/on-project/pub/system/etc/下



### Swdefaultparam.txt

朝歌默认参数表，里面分三种表 system,secure,global，一般是apk回去读里面的东西，

恢复出厂后会重置数据，复原。

### Swupgradeparam.txt

升级参数，如果需要更新参数，则version要+1，编译升级包



## swRootService/swRootServiceTest

朝歌自写的一个功能模块

/vendor/sunniwell/system/core/swrootservice/client 目录对应swRootServiceTest 的源码

/vendor/sunniwell/system/core/swrootservice/server 目录对应swRootService 源码

使用

swRootServiceTest –-help 查看支持的功能，具体源码在

/vendor/sunniwell/system/core/swrootservice/client/rootService\_test.cpp

调用流程

这些文件调用流程，rootService\_test.cpp->SWRootServiceClient.cpp->SWRootService.cpp->swled.c

编译出来的可执行文件 swRootService 和 swRootServiceTest 在这

bigsys/out/target/product/xxxx/vendor/bin

RootService 和 swRootServiceTest 如果源码有修改， 运行Test报错

CANNOT LINK EXECUTABLE "swRootServiceTest": cannot locate symbol "\_ZN7android19SWRootServiceClient8startCmdEPcS1\_i" referenced by "/vendor/bin/swRootServiceTest

则要把 编译出来的 vendor/lib/libswrootservice.so 放到 vendor/lib 下

## 小系统文件定制

### 为什么要定制

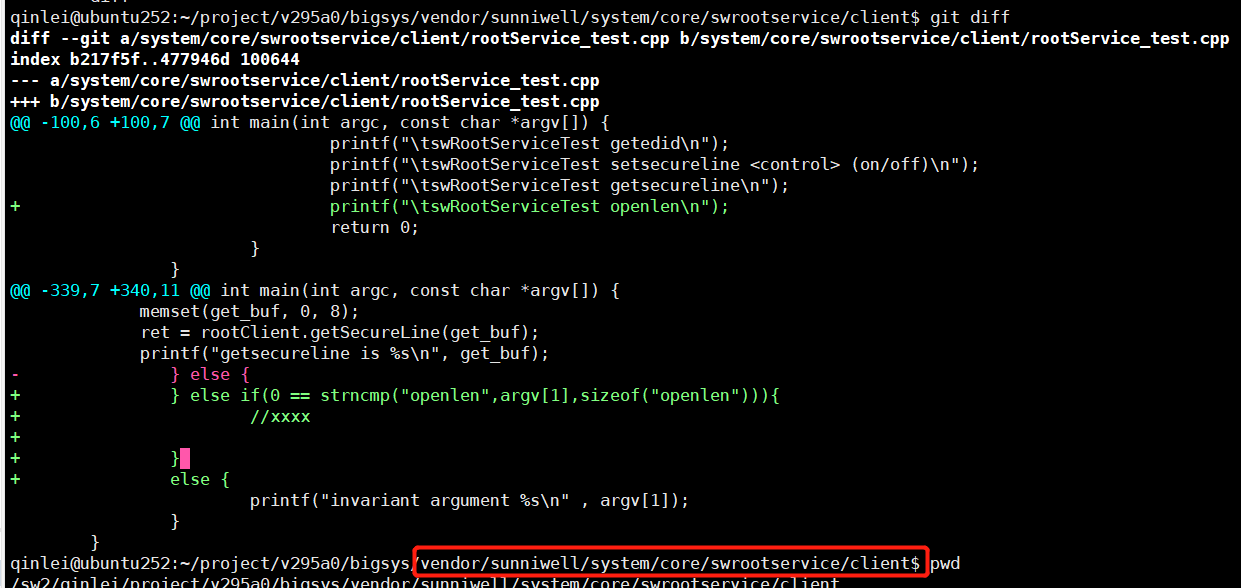
多个项目可能会使用同一套sdk,sdk上的修改不适用于所有项目的时候，就需要定制

### 定制流程以及注意事项

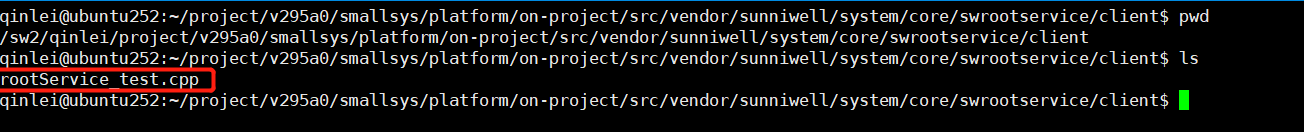
首先在大系统修改，加入需要的功能代码，然后编译出需要的目标文件，看编译出来的文件生成在哪个目录，比如swRootServiceTest文件，编译生成在 大系统的 vendor/bin 目录中，然后放到小系统 pub下对应大系统的目录中, swRootServiceTest 是放到 小系统pub/vendor/bin下 ，

然后在小系统on-project/src 文件夹下 (没有就自己创建)，创建和大系统修改文件路径一样的文件夹（从根目录开始），然后把修改后的源码放到该文件夹中

sdk修改

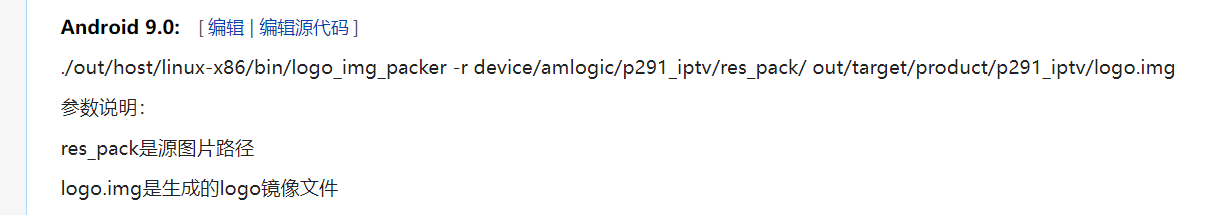


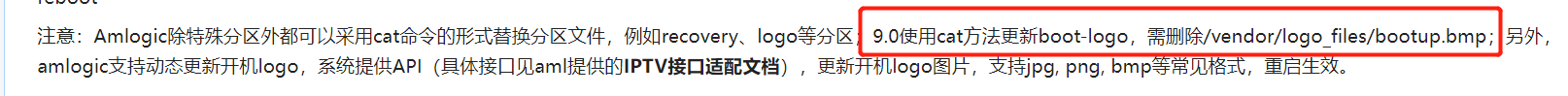
**小系统on-project/src文件夹：**



### 开机logo定制

#### 制作





./out/host/linux-x86/bin/logo\_img\_packer -r device/amlogic/p291\_iptv/res\_pack/ out/target/product/p291\_iptv/logo.img

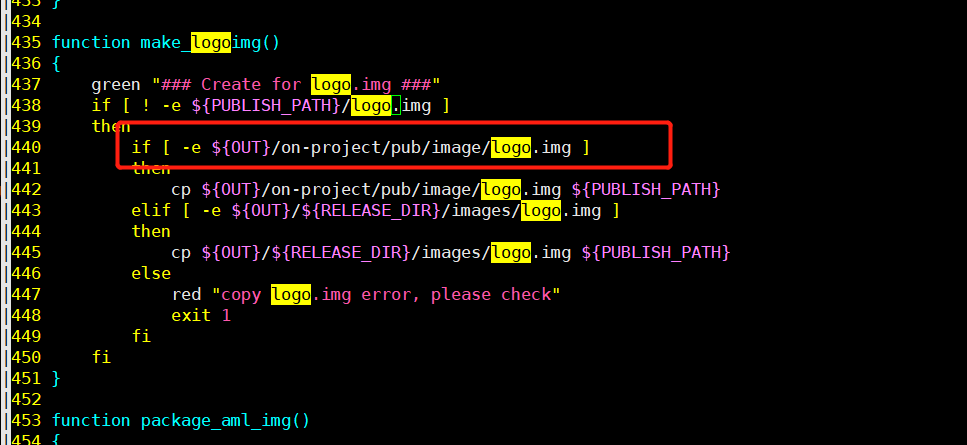
#### 快速验证



#### 合入小系统

验证无误之后，要放到小系统作为定制，怎么看放到小系统哪个路径呢

去到 platform/build/make.sh 搜索 logo



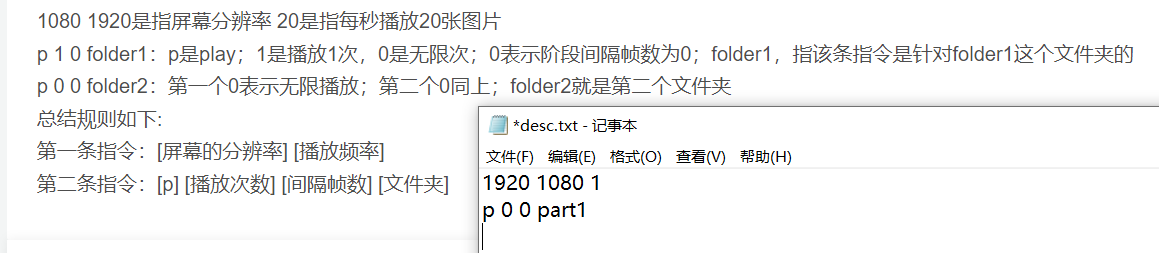
就会得到 logo.img定制时所放的路径，不存在就自己创建，然后把修改后的 logo.img放到该路径中，编译之后若发现定制未生效，就要去查看脚本中哪里把它覆盖掉了

### 开机动画定制

linux下制作 用命令 zip -r -0 bootanimation.zip ./

制作Android开机动画文件时，文件bootanimation.zip压缩模式需为存储模式。如果不是存储模式，开机界面将会出现黑屏的情况，无法正常播放。

#### 动画制作



#### 快速验证

把板卡中 /system/media/bootanimation.zip 替换掉 重启

#### 合入小系统

路径在 pub/system/meida/bootanimation.zip

### 其他文件定制(举例swRootServiceTest)

具体看16点

## 默认壁纸修改

大系统中壁纸路径在 framework/base/core/res/res/下



该文件夹编译出来的文件是 /system/framework/framework-res.apk

定制就把源码放到 op-project/src下对应的目录下，定制文件放到 pub的对应目录下

## Init.amlogic.rc文件

安卓init进程会去加载 .rc结尾的文件

修改权限、创建文件夹、拉起服务、配置触发器

定制路径是小系统 pub/vendor/etc/init/hw/

**sdk中/device/amlogic/common/products/iptv/init.amlogic.system.g12a.rc**

**编译完之后 会复制到大系统 out/target/product/xxx/vendor/etc/init/hw/init.amlgic.rc**

**定制的话 就把大系统编译生成的 rc文件放到 pub/vendor/etc/init/hw/下**

## Patch补丁生成和使用

git diff > ~/xxx.patch 生成补丁 xxx.diff 一样的

去对应目录 patch -p1 < xxx.patch 打补丁

## Uboot编译以及快速替换

### 环境变量配置

export PATH=/opt/gcc-linaro-aarch64-none-elf-4.8-2013.11\_linux/bin:${PATH}

### 编译

[Amlogic android-SDK说明 - Sunniwiki (sunniwell.net)](https://wiki.sunniwell.net/index.php/Amlogic_android-SDK%E8%AF%B4%E6%98%8E#.E4.BA.94.E3.80.81Uboot.E7.BC.96.E8.AF.91)

Android 9.0 S905L3/L3A(L3/L3A 共SDK，但uboot代码不共用)

1.L3-uboot编译

cd uboot

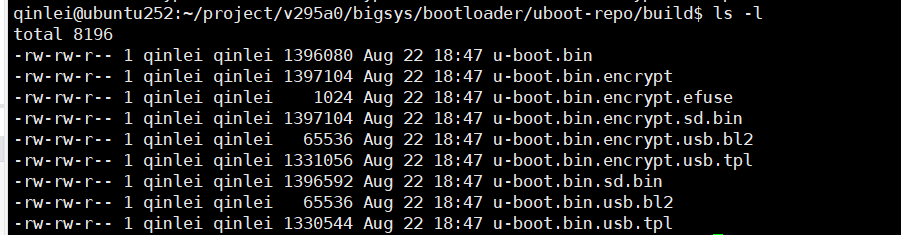
./mk gxl\_p211\_v1

2.L3A-uboot编译

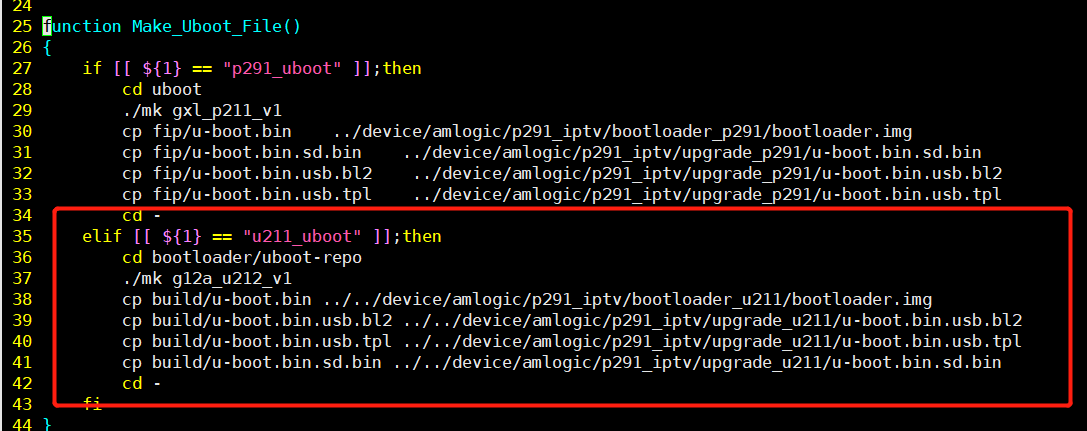
cd bootloader/uboot-repo

./mk g12a\_u212\_v1

编译完成之后会在 build目录下生成几个文件



看到 大系统根目录中的 make.sh S905L3A走的是u211\_boot



### 替换验证

对着上面的图片，把编译出来的uboot文件复制到小系统 platform/release/loader目录中，然后编译本地版本验证(四个文件都有修改就编版本)

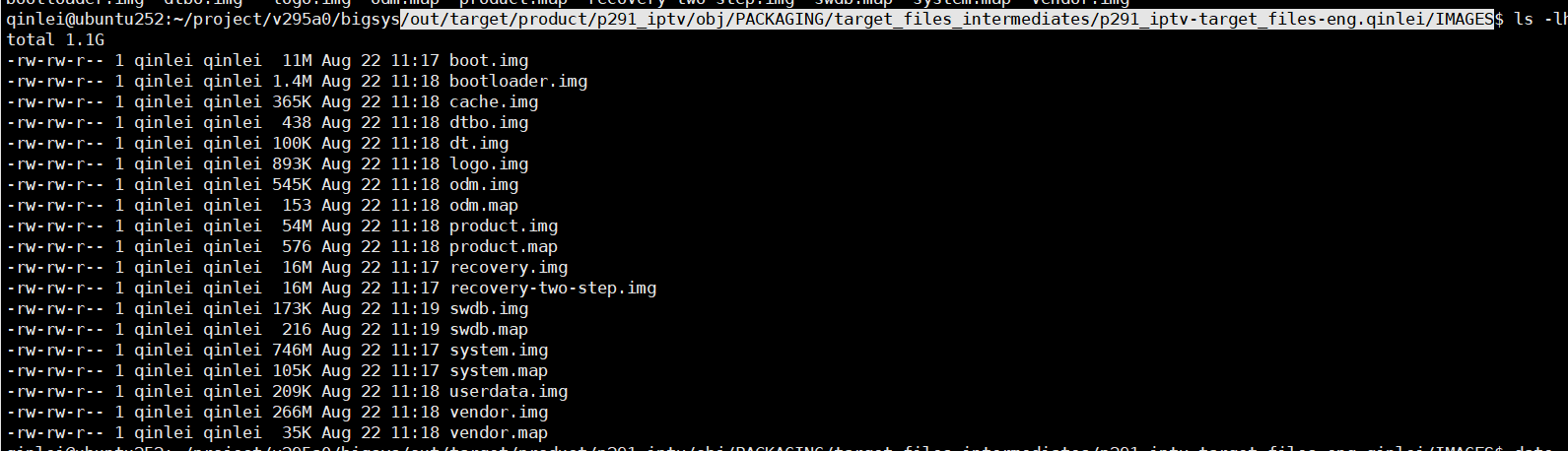
如果只修改到bootloader.img，就可以替换到盒子里验证

方法：

boot.img、dtb文件的替换wiki上也有，

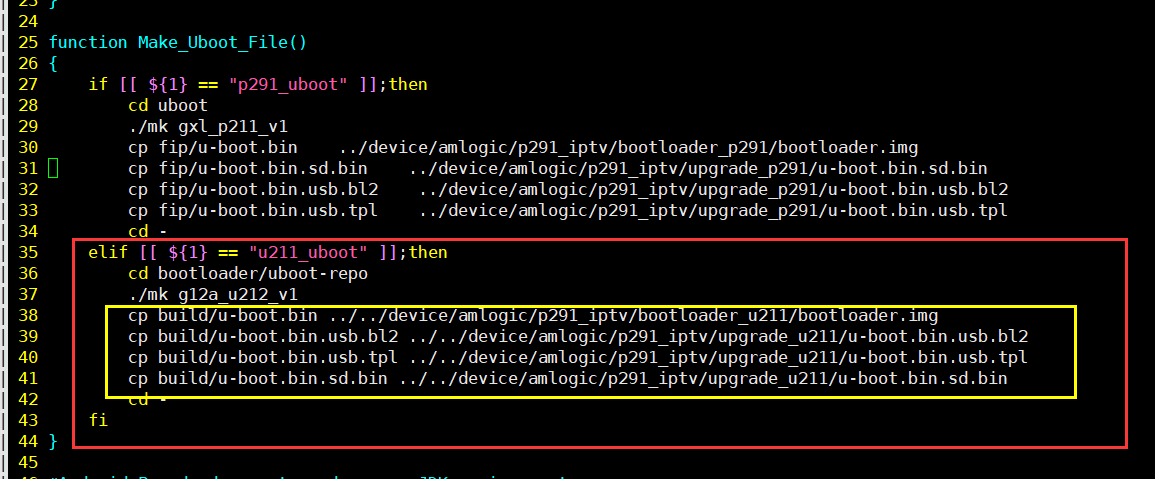
编译sdk之后，boot.img在sdk哪个目录？

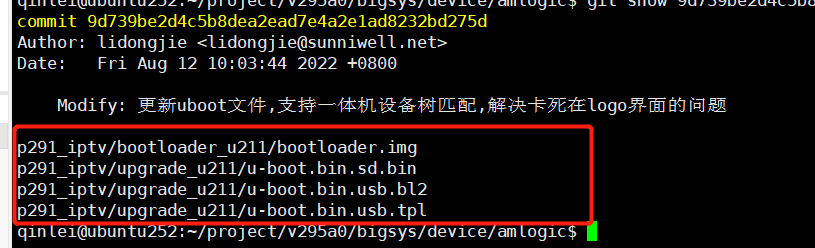
/out/target/product/p291\_iptv/obj/PACKAGING/target\_files\_intermediates/p291\_iptv-target\_files-eng.xxxx/IMAGES



### 提交修改到大系统

**注意：Amlogic SDK编译不会自动编译uboot，所以当uboot代码发生更新或修改时，需先编译uboot，然后将生成的u-boot.bin文件放置在/device/amlogic目录下做替换**（/device/amlogic目录下存放u-boot.bin文件的位置与SDK相关，位置不固定），上述编译操作在/build/make目录下的ci编译脚本和快速编译脚本make.sh均有体现

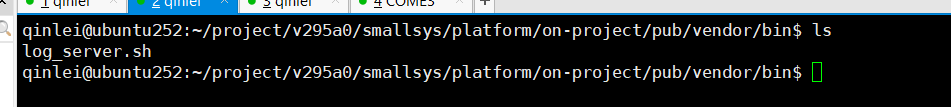




## 添加服务脚本

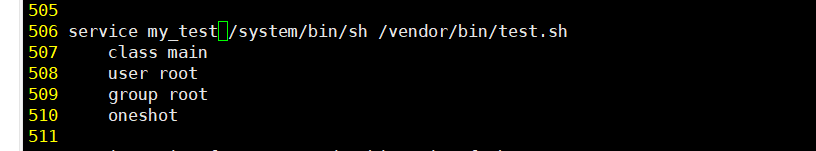
当前版本中有两个脚本，一个开机启动脚本，一个日志服务脚本(属性触发)





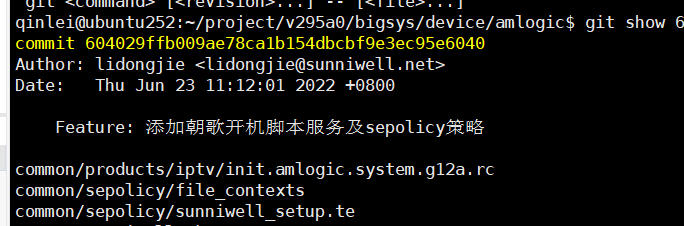
### 脚本作为参数配置

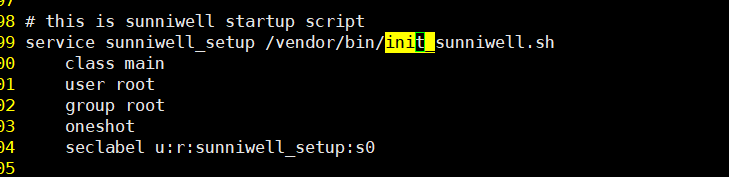
不用配置sdk权限，服务配置成 /system/bin/sh 脚本当成参数传递给 /system/bin/sh



### 脚本直接当成服务配置

需要在sdk中配置相关selinux，并在 init.amlgic.rc中添加服务



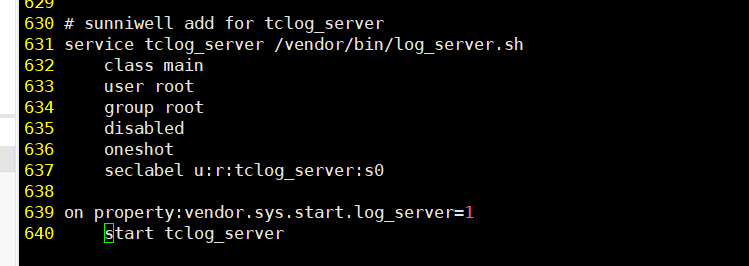


### 服务配置选项

## 安卓触发器

### 什么是触发器

### 触发器配置



当 vendor.sys.start.log\_server属性值设置成1的时候，就启动 tclog\_server

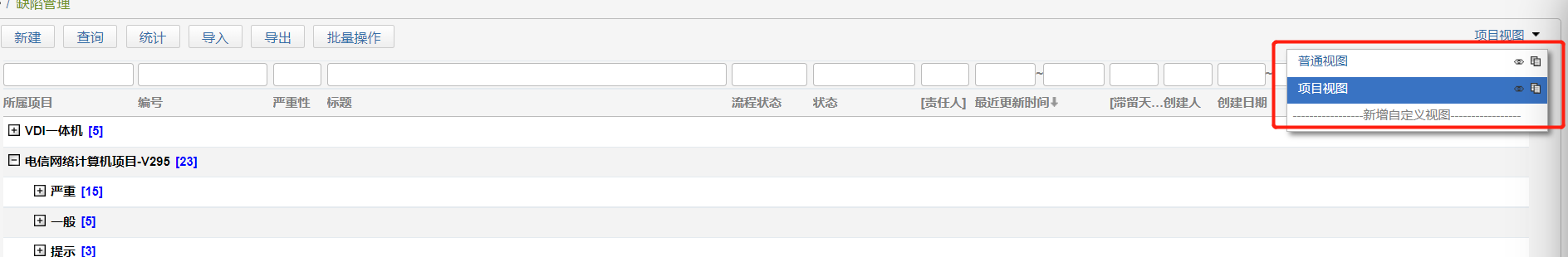
### 触发器启动脚本服务

# 服务器使用

## RDM相关操作

### 问题单分发





创建审核中的就是还没分发出去的



点开



填写好解决人，然后确定



### 问题单转发

点开要转发的问题单，

写好解决人之后，点击确定



### 问题单回归测试

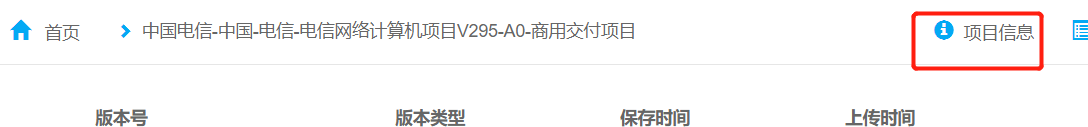
点开需要提交测试的问题，



填写好以下信息之后，确定



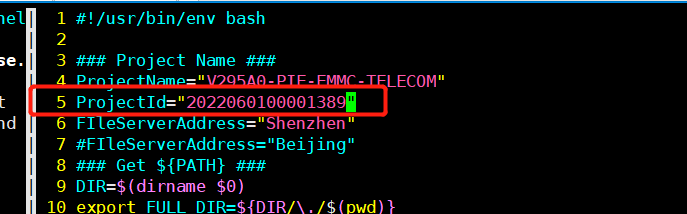
## release使用



可以看到 项目相关信息，



项目id，小系统的CI脚本会根据项目ID上传到对应的release、



添加项目成员





## 申请CI和CI相关配置以及注意事项

# 软件工具使用

## 文件对比工具使用

## 抓包工具使用

## Apk逆向助手工具使用

# 代码仓库相关操作

## 代码仓库申请

### 仓库命名规则

## 创建项目分支

### Git

### Gerrit

## 分支合并

## 仓库迁移

# 版本发布指导

版本发布流程 版本发布就是在ci上编译新版本，合入修改->大系统ci->小系统ci->自测试基础功能(重要)->写转测表->发转测邮件->RDM提交