

密级状态：绝密( ) 秘密( ) 内部( ) 公开(√)

# Rockchip Android 13.0 SDK开发指南

<div>文件状态: <input type="checkbox"/> 草稿 <input checked="" type="checkbox"/> 正式发布 <input type="checkbox"/> 正在修改</div>	文件标识:	RK-KF-YF-745
	当前版本:	V1.1.0
	作者:	吴良清
	完成日期:	2022-12-10
	审核:	金华君
	审核日期:	2022-12-10

## 免责声明

本文档按“现状”提供，瑞芯微电子股份有限公司（“本公司”，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

## 商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标，归本公司所有。

本文档可能提及的其他所有注册商标或商标，由其各自所有者所有。

## 版权所有 © 2022 瑞芯微电子股份有限公司

超越合理使用范畴，非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

瑞芯微电子股份有限公司

Rockchips Electronics Co., Ltd.

地址：福建省福州市铜盘路软件园A区18号

网址：[www.rock-chips.com](http://www.rock-chips.com)

客户服务电话：+86-4007-700-590

客户服务传真：+86-591-83951833

客户服务邮箱：[fae@rock-chips.com](mailto:fae@rock-chips.com)

## Rockchip Android 13.0 SDK支持芯片

芯片平台	是否支持	SDK版本
RK3588	支持	RKR1
RK3588S	支持	RKR1
RK3566	支持	RKR3
RK3568	支持	RKR3

## 历史版本

版本号	作者	修改日期	修改说明	备注
V0.0.1	吴良清	2022-10-15	发布RK3588/RK3588S RKR1版本SDK	仅供硬件调试
V1.1.0	吴良清	2022-12-10	发布RK3566/RK3568 RKR3版本SDK	

文档问题反馈：[wlq@rock-chips.com](mailto:wlq@rock-chips.com)

# Rockchip Android 13.0 SDK代码下载编译

## 代码下载

### 下载地址

```
repo init --repo-url https://gerrit.rock-chips.com:8443/repo-release/tools/repo
-u https://gerrit.rock-chips.com:8443/Android_T/manifests.git -m Android13.xml
```

如果申请的Express的权限，那么下载地址如下：

```
repo init --repo-url https://gerrit.rock-chips.com:8443/repo-release/tools/repo
-u https://gerrit.rock-chips.com:8443/Android_T/manifests.git -m
Android13_Express.xml
```

### 服务器镜像下载

```
repo init --repo-url https://gerrit.rock-chips.com:8443/repo-release/tools/repo
-u https://gerrit.rock-chips.com:8443/Android_T/manifests.git -m Android13.xml -
-mirror
```

如果申请的Express的权限，那么下载地址如下：

```
repo init --repo-url https://gerrit.rock-chips.com:8443/repo-release/tools/repo
-u https://gerrit.rock-chips.com:8443/Android_T/manifests.git -m
Android13_Express.xml --mirror
```

注，repo是google用Python脚本写的调用git的一个脚本，主要是用来下载、管理Android项目的软件仓库，其下载地址如下：

```
git clone https://gerrit.rock-chips.com:8443/repo-release/tools/repo
```

为方便客户快速获取SDK源码，瑞芯微技术窗口通常会提供对应版本的SDK初始压缩包。以Rockchip\_Android13.0\_SDK\_RELEASE.tar.gz.\*`为例，拷贝到该初始化包后，通过如下命令可检出源码：

```
mkdir Rockchip_Android13.0_SDK_RELEASE
cat Rockchip_Android13.0_SDK_RELEASE.tar.gz* | tar -zx -C
Rockchip_Android13.0_SDK_RELEASE
cd Rockchip_Android13.0_SDK_RELEASE
.repo/repo/repo sync -l
.repo/repo/repo sync -c
```

## 搭建自己的repo代码服务器

### 环境

安装 `openssh-server` 用于远程登录, `git` 用于管理工程, `keychain` 用于公私钥管理工具

```
sudo apt-get install openssh-server git keychain
```

### gitolite搭建

#### 服务器端操作

(以服务器地址: 10.10.10.206为例进行说明)

1. 创建git账户:

```
sudo adduser --system --shell /bin/bash --group git
sudo passwd git
```

2. 以“git”账户登录服务器
3. 确保“`~/.ssh/authorized_keys`”为空或者不存在
4. 拷贝服务器管理员的公钥到“`~/YourName.pub`”
5. 下载gitolite源码

```
git clone https://github.com/sitaramc/gitolite.git
```

6. 在git用户目录下创建bin目录

```
mkdir -p ~/bin
```

7. 执行下列命令安装gitolite, 不同版本安装方法不同, 请参考源码中的文档:

```
gitolite/install -to ~/bin
```

8. 设置管理员

```
~/bin/gitolite setup -pk YourName.pub
```

#### 客户端操作

1. 克隆服务器的gitolite管理仓库:

```
git clone ssh://git@10.10.10.206/gitolite-admin.git
```

2. 添加用户公钥到gitolite目录下

```
cp username.pub keydir/username.pub
```

### 3. 添加管理员用户

```
vi conf/gitolite.conf
@admin = admin1 admin2 admin3
repo gitolite-admin
RW+    =    @admin
```

## repo镜像搭建

### 服务器端操作

1. 用git账号登入服务器
2. 在根目录下下载repo工具

```
git clone https://gerrit.rock-chips.com:8443/repo-release/tools/repo
```

### 3. 新建RK\_Android13\_mirror目录

### 4. 进入 RK\_Android13\_mirror目录

### 5. 下载RK Android13 SDK镜像

### 6. 创建仓库组权限

### 客户端操作

1. 将服务器端的android\_t.conf拷贝到客户端的·gitolite-admin/conf/·下
2. 添加组权限

### 3. 新建自己的manifests仓库

## 客户端操作

1. 在客户端下载manifests\_xxx仓库  
在其他客户端电脑上下载manifests\_xxx.git仓库

### 2. 在客户端下载原始manifests仓库

3. 提交manifest.xml文件到新建的manifest\_xxx仓库中  
将原始manifests下面的文件拷贝到manifests\_xxx内

查看拷贝文件

本地提交

push到远程分支

4. 创建自己的代码下载链接  
在根目录下下载repo工具

按以上步骤操作后，自己的代码下载链接如下

其中：

//10.10.10.206 是你的服务器端地址

通过以上步骤就可以完成自己的repo服务器搭建了，可以把自己的代码服务器链接分享给同事们一起工作了。

## 代码管理

通过以上步骤搭建代码服务器后大部分代码仓库都使用RK默认的分支，如果有仓库需要修改自己的代码，可以参考下面的步骤进行操作。

### 切换自己的代码分支

1. 进入需要修改的代码仓库，以kernel目录为例进行说明

2. 切换一个本地分支

3. push xxx\_branch分支到远程服务器

其中 rk29 是remote 可以直接tab键自动补全

4. 进入.repo/manifests目录修改manifest里面指定的分支  
进入.repo/manifests目录通过grep kernel可以找到kernel仓库对应的manifest的位置

5. 提交修改的manifest到远程分支

提交manifests仓库后，其他同事就可以同步到你们自己的分支的kernel代码了。

## 代码修改提交

按上面步骤切换完分支后就可以在自己分支上提交自己的修改了，提交直接push到xxx\_branch分支上面。

## 同步RK的代码

1. 同步RK代码需要在服务器端进行sync操作

2. 客户端合并RK对manifests的修改

- 下载RK原始manifests仓库

使用对比工具对比manifests（RK原始）和manifests\_xxx(自己的)，将RK修改的差异部分合并到自己的仓库中（主要修改tag，增加删除仓库等）。

- 对比确认后将修改push到manifests\_xxx上。

这步也可以确认自己修改了哪些仓库，在下一步中将进行修改仓库的合并。

3. 有自己切分支的目录需要手动把RK的修改merge到自己的分支上面  
以kernel为例：

- 查看当前指向的远程分支

可以看到当前指向的是：`remotes/m/master` -> `rk29/xxx_branch`

- 创建本地分支（从自己的远程分支上切）

- 确认当前RK发布的最新TAG

可以看到当前最新的Android13的tag是 `android-13.0-mid-rkr1`

- 合并 `android-13.0-mid-rkr1` 到本地分支

查看是否有冲突，如果有冲突先解决冲突，没有冲突在执行下一步

- push合并完的代码到远程分支

- 其他切分的目录都按这个方式进行合并提交即可

## kernel代码路径说明

RK3588 Android13只支持5.10 版本的kernel， kernel源码在工程中kernel目录下，

## 代码编译

### 一键编译命令

### 各个平台编译命令汇总

Soc	类型	参考机型	Android	一键编译	kernel编译	uboot编译
RK3588S	平板	样机	build/envsetup.sh;lunch rk3588s_t-userdebug	./build.sh -AUCKu -d rk3588s-tablet-v10	make ARCH=arm64 rockchip_defconfig android-13.config pcie_wifi.config;make ARCH=arm64 rk3588s-tablet-v10.img -j24	./make.sh rk3588
RK3588S	开发板	RK3588S-EVB1	build/envsetup.sh;lunch rk3588_t-userdebug	./build.sh -AUCKu -d rk3588s-evb1-lp4x-v10	make ARCH=arm64 rockchip_defconfig android-13.config pcie_wifi.config;make ARCH=arm64 rk3588s-evb1-lp4x-v10.img -j24	./make.sh rk3588
RK3588	开发板	RK3588 EVB1	build/envsetup.sh;lunch rk3588_t-userdebug	./build.sh -AUCKu	make ARCH=arm64 rockchip_defconfig android-13.config pcie_wifi.config;make ARCH=arm64 rk3588-evb1-lp4-v10.img -j24	./make.sh rk3588
RK3566	平板	样机	build/envsetup.sh;lunch rk3566_tgo-userdebug	./build.sh -AUCKu	make ARCH=arm64 rockchip_defconfig android-13.config rk356x.config;make ARCH=arm64 rk3566-rk817-tablet.img -j24	./make.sh rk3566
RK3566	开发板	EVB2-LP4X-V10	build/envsetup.sh;lunch rk3566_t-userdebug	./build.sh -AUCKu	make ARCH=arm64 rockchip_defconfig android-13.config rk356x.config;make ARCH=arm64 rk3566-evb2-lp4x-v10.img -j24	./make.sh rk3566
RK3568	开发板	EVB1-DDR4-V10	build/envsetup.sh;lunch rk3568_t-userdebug	./build.sh -AUCKu	make ARCH=arm64 rockchip_defconfig android-13.config rk356x.config;make ARCH=arm64 rk3568-evb1-ddr4-v10.img -j24	./make.sh rk3568

### 其他编译说明

#### Android13.0不能直接烧写kernel.img和resource.img

Android13.0的kernel.img和resource.img包含在boot.img中，需要使用build.sh -AK 命令来编译kernel。编译后烧写rockdev下面的boot.img。也可以使用如下方法单独编译kernel。这个过程会重新编译Android，所以编译时间会比较长，建议用下面单独编译kernel的方式的编译。



单独编译kernel生成boot.img

编译的原理：在kernel目录下将编译生成的 kernel.img 和 resource.img 替换到旧的 boot.img 中。  
以 RK3588 样机为例，编译时替换对应的boot.img及dts：  
其中 BOOT\_IMG=../rockdev/Image-rk3588\_t/boot.img 这里指定的是旧的boot.img的路径，命令如下：

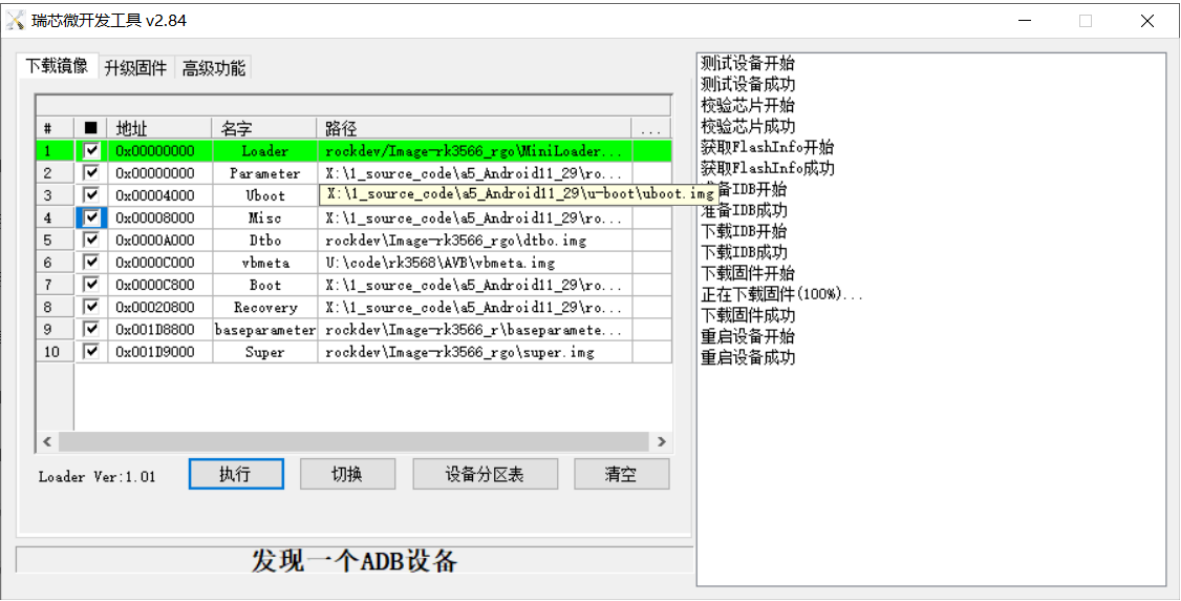
导clang到环境

编译后可以直接烧写kernel-5.10目录下的boot.img到机器的boot位置，烧写时**请先加载分区表**（parameter.txt），以免烧写位置错误。

固件烧写

固件烧写工具

Android13的USB驱动DriverAssitant需要更新到V5.1.1版本，可以参考下面的工具章节进行更新。  
Windows烧写工具：（工具是时刻更新，请及时同步更新）



在下文工具说明章节有详细说明

固件说明

完整编译后会生成如下文件：

工具烧写如下文件即可：

也可以直接烧写 update.img

固件说明

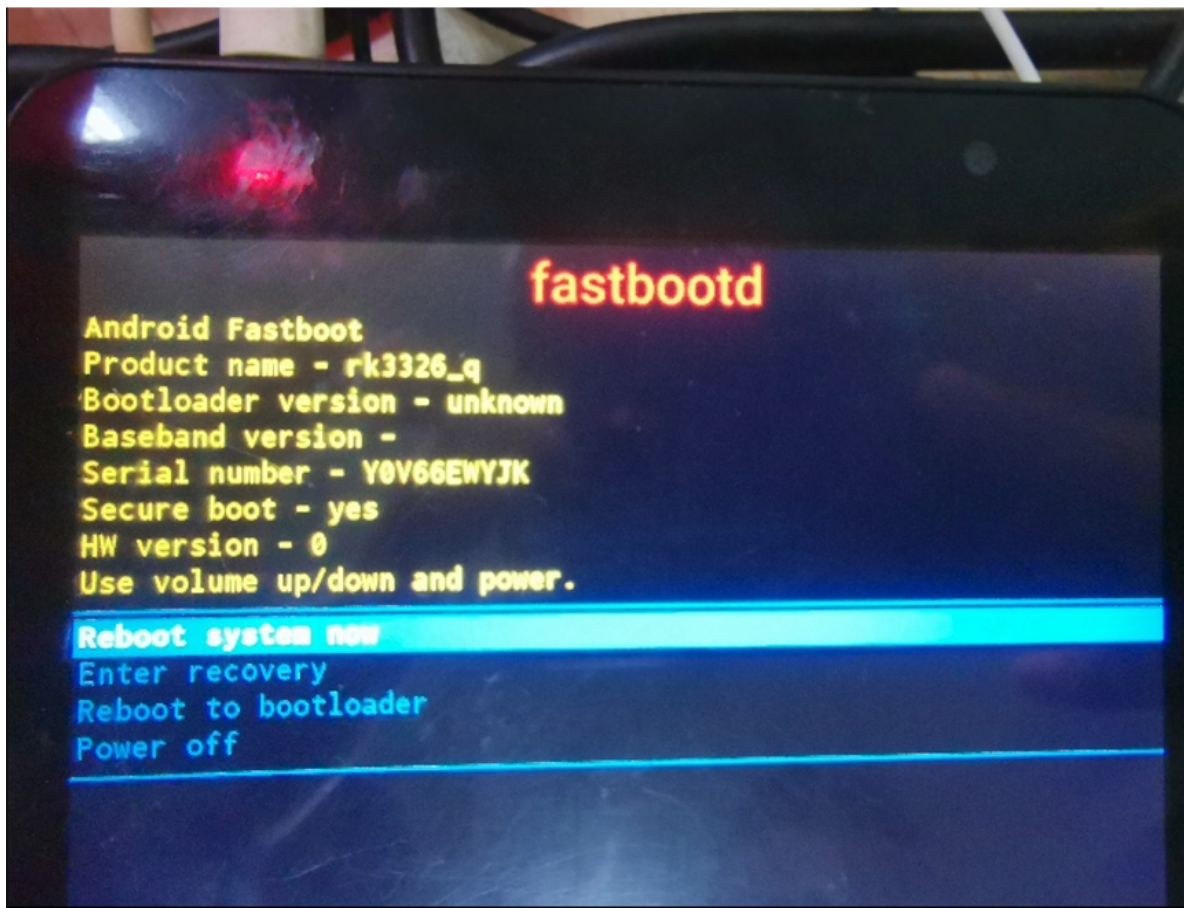
固件	说明
boot.img	包含ramdis、kernel、dtb
boot-debug.img	与boot.img的差别是user固件可以烧写这个boot.img进行root权限操作
dtbo.img	Device Tree Overlays 参考下面的dtbo章节说明
config.cfg	烧写工具的配置文件，可以直接导入烧写工具显示需要烧写的选项
MiniLoaderAll.bin	包含一级loader
misc.img	包含recovery-wipe开机标识信息，烧写后会进行recovery
parameter.txt	包含分区信息
固件	说明
-----	-----
pcba_small_misc.img	包含pcba开机标识信息，烧写后会进入简易版pcba模式
pcba_whole_misc.img	包含pcba开机标识信息，烧写后会进入完整版pcba模式
recovery.img	包含recovery-ramdis、kernel、dtb
super.img	包含odm、product、vendor、system、system_ext分区内容
trust.img	包含BL31、BL32 RK3566/RK3568没有生成这个固件，不需要烧写
uboot.img	包含uboot固件
vbmata.img	包含avb校验信息，用于AVB校验
update.img	包含以上需要烧写的img文件，可以用于工具直接烧写整个固件包

## fastboot烧写动态分区

R的新设备支持动态分区，已经移除了system/vendor/odm/product/system\_ext分区，请烧写super.img，单独烧写system/vendor/odm等（可以在out下面找到对应img文件）可以用fastbootd，要求adb和fastboot版本均为最新，SDK提供了编译好的工具包：

- 使用命令烧写动态分区：

**注：**进入fastbootd模式后，屏幕上会显示相关设备信息，如图所示：



注：非动态分区使用fastboot，请进入bootloader：

烧写GSI的方法：

- 确认机器解锁后，进入fastbootd，只需要烧写GSI中的system.img及固件中的misc.img，烧写后会进入recovery进行恢复出厂设置。下面附上整个烧写流程：

1. 重启至bootloader，未解锁->机器解锁：

2. 恢复出厂设置，重启至fastbootd：

3. 开始烧写GSI

- 注：也可以使用DSU(Dynamic System Updates)烧写GSI，目前Rockchip平台已经默认支持DSU。由于该功能需要消耗大量内存，不建议1G DDR及以下的设备使用，有关DSU的说明和使用，请参考Android官网：

<https://source.android.com/devices/tech/ota/dynamic-system-updates>

- 注1：VTS测试时，需要同时烧写编译出的boot-debug.img到boot分区；
- 注2：CTS-ON-GSI测试时则不需要烧boot-debug.img；
- 注3：测试时请使用Google官方发布的，带有-signed结尾的GSI镜像；

## 使用DTBO功能

Android 10.0及以上支持Device Tree Overlays功能，开发过程体现在需要烧写dtbo.img，用于多个产品间的兼容等。

修改方法：

1. 找到(或指定)模板文件：

例如：

2. 添加或修改需要的节点：

例如：

**注：使用dtbo时一定要确保dts中存在alias，否则无法成功overlay**

## 修改fstab文件

1. 找到(或指定)模板文件：

例如：

2. 修改：添加分区挂载、修改swap\_zram参数，修改data分区格式等

## 修改parameter.txt

Android 13添加了生成parameter.txt的工具，支持根据配置参数编译出parameter.txt。如果没有配置模板文件，则会寻找添加修改好的parameter.txt文件。

1. 找到(或指定)模板文件：

例如：

2. 修改配置分区大小(例如)：

3. 不使用parameter生成工具：

添加一个parameter.txt文件到你的device目录下即可：

例如：device/rockchip/rk3326/rk3326\_q/parameter.txt

4. 仅使用工具生成parameter.txt(例如)：

**注：如果需要大版本OTA升级，请直接使用之前版本的parameter.txt**

5. 新加一个分区

以新建baseparameter分区为例进行说明：

- 在产品的BoardConfig.mk中定义：BOARD\_WITH\_SPECIAL\_PARTITIONS  
like: BOARD\_WITH\_SPECIAL\_PARTITIONS := baseparameter:1M,logo:16M

- 在RebuildParameter.mk中添加BOARD\_WITH\_SPECIAL\_PARTITIONS

## Android常用配置

---

### 新建产品lunch

以RK3588平台新建rk3588s\_t产品为例，分以下步骤：

1) 修改device/rockchip/rk3588/AndroidProducts.mk增加rk3588s\_t的lunch

2) 在device/rockchip/rk3588目录下新建rk3588s\_t目录

参考device/rockchip/rk3588下已有的rk3588\_t产品目录新建，可以先直接拷贝rk3588\_t为rk3588s\_t，然后将rk3588s\_t目录下的所有 rk3588\_t 字符改为 rk3588s\_t

## Kernel dts说明

---

### 新建产品dts

产品新建dts可以根据下表的配置选择对应的dts作为参考。

Soc	PMIC	开发板类型	机型	DTS
RK3588	双PMIC: RK806 * 2	开发板	RK3588 EVB1	rk3588-evb1-lp4-v10
RK3588	单PMIC: RK806+RK860	开发板	RK3588硬件参考设计对应的软件配置	rk3588-evb7-lp4-v10
RK3588S	双PMIC: RK806 * 2	开发板	RK3588S EVB1	rk3588s-evb1-lp4x-v10
RK3588S	单PMIC: RK806+RK860	平板	RK3588S平板硬件设计参考图对应的软件配置	rk3588s-tablet-v11
RK3588S	双PMIC: RK806 * 2	平板	样机	rk3588s-tablet-v10
RK3588	单PMIC: RK860	box 开发板	RK3588_NVR_DEMO1_LP4X	rk3588-nvr-demo-v10-android
RK3566	RK817	平板	样机	rk3566-rk817-tablet
RK3566	RK809	开发板	RK3566 EVB2	rk3566-evb2-lp4x-v10
RK3566	分立	开发板	RK3566 BOX DEMO	rk3566-box-demo-v10
RK3568	RK809	开发板	RK3568 EVB1	rk3568-evb1-ddr4-v10

## 文档说明

### 外设支持列表

DDR/EMMC/NAND FLASH/WIFI/3G/CAMERA的支持列表实时更新在redmine上，链接如下：

### Android文档

### Android\_SELinux(Sepolicy)开发指南

### Wi-Fi文档

### 3G/4G模块说明文档

**Kernel文档**

**DDR相关文档**

**Audio模块文档**

**CRU模块文档**

**GMAC模块文档**

**PCie模块文档**

**I2C模块文档**

**PIN-Ctrl GPIO模块文档**

**SPI模块文档**

**Sensor模块文档**

**IO-Domain模块文档**

**Leds模块文档**

**Thermal温控模块文档**

**PMIC电源管理模块文档**

## MCU模块文档

## 功耗与休眠模块文档

## UART模块文档

## DVFS CPU/GPU/DDR变频相关文档

## EMMC/SDMMC/SDIO模块文档

## PWM模块文档

## USB模块文档

## HDMI-IN功能文档

## 安全模块文档

## uboot介绍文档

## Trust介绍文档

## Camera文档

## Camera IQ Tool文档

## 工具文档



## PCBA开发使用文档

## 显示屏驱动调试指南

## HDMI调试指南

## 图像显示DRM Hardware Composer（HWC）问题分析排查

## DRM显示开发指南

## RGA相关问题分析排查

## 图形显示框架常见问题分析

包括frameworks、GPU.Gralloc、GUI、HWComposer、HWUI、RGA

## rknn-toolkit2开发SDK和文档

## 工具使用

### StressTest

设备上使用Stresstest 工具，对待测设备的各项功能进行压力测试，确保整个系统运行的稳定性。SDK通过打开计算器应用，输入“83991906=”暗码，可启动StressTest应用，进行各功能压力测试。

Stresstest 测试工具测试的内容主要包括：

#### 模块相关

- Camera 压力测试：包括Camera 打开关闭，Camera 拍照以及Camera 切换。
- Bluetooth 压力测试：包括Bluetooth 打开关闭。
- Wi-Fi 压力测试：包括Wi-Fi 打开关闭，（ping 测试以及iperf 测试待加入）。

#### 非模块相关

- 飞行模式开关测试
- 休眠唤醒拷机测试
- 视频拷机测试
- 重启拷机测试
- 恢复出厂设置拷机测试
- Arm 变频测试

- Gpu 变频测试
- DDR 变频测试

## PCBA测试工具

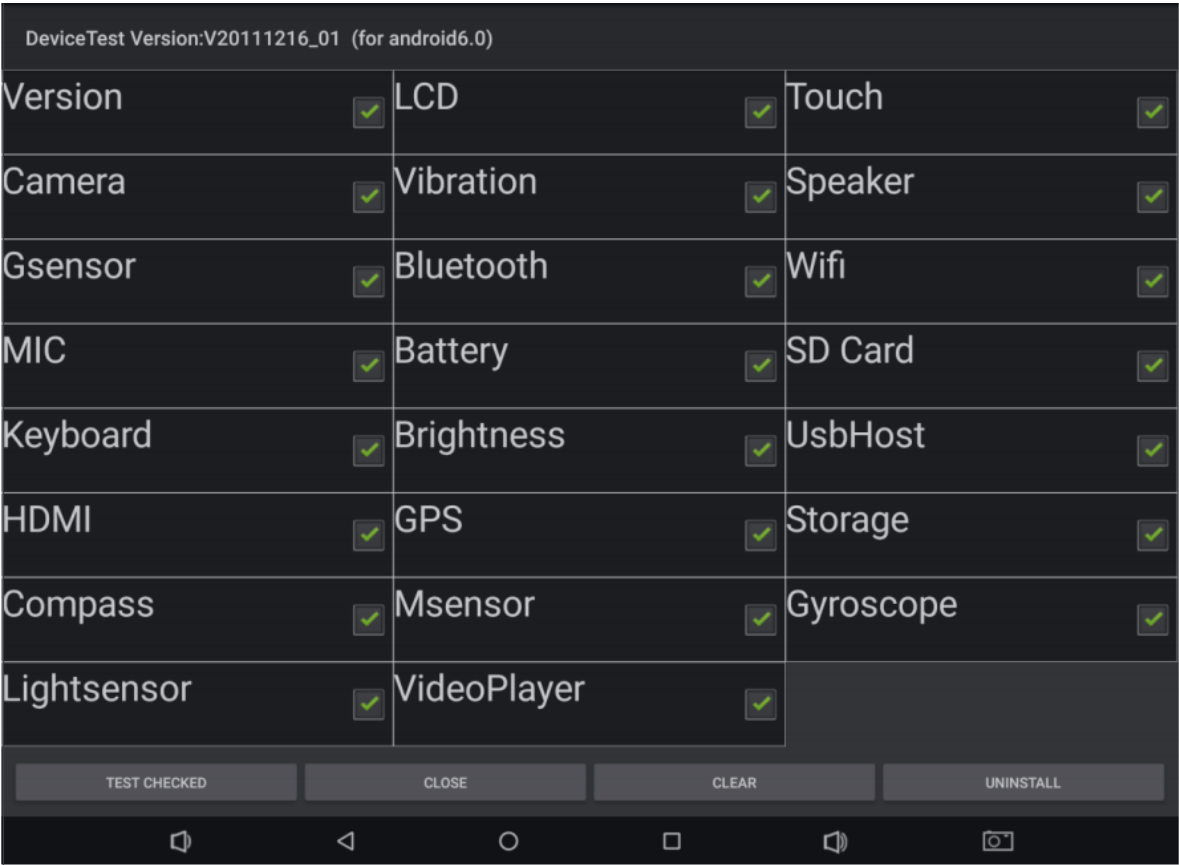
PCBA 测试工具用于帮助在量产的过程中快速地甄别产品功能的好坏，提高生产效率。目前包括屏幕（LCD）、无线（Wi-Fi）、蓝牙（bluetooth）、DDR/EMMC 存储、SD 卡（sdcard）、USB HOST、按键（KEY），喇叭耳机（Codec）测试项目。

这些测试项目包括自动测试项和手动测试项，无线网络、DDR/EMMC、以太网为自动测试项，按键、SD卡、USB HOST、Codec、为手动测试项目。

具体PCBA功能配置及使用说明，请参考：

## DeviceTest

DeviceTest 用于工厂整机测试，主要测试装成整机以后外围器件是否正常。SDK 通过打开计算器，输入暗码“000.=”进入 DeviceTest，如下所示：



在产线可以根据这个界面进行对应外设的测试，测试时点击“TEST CHECKED”对所测项目逐项进行测试，测试如果成功点击 pass，失败点击 failed，最终结果会显示在界面上，如下图所示，红色为 failed 项，其余为通过项，工厂可根据测试结果进行相应的维修。另外，如果客户需要对该工具进行定制，请联系 FAE 窗口申请对应的源码。

## USB驱动

Rockchip USB驱动安装包，包括ADB、固件烧写驱动

## 开发烧写工具

### Windows版本

RKTools/windows/AndroidTool/AndroidTool\_Release\_v2.90.zip，工具版本会时刻更新，请及时同步更新



## Linux版本

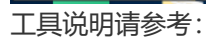
RKTools/linux/Linux\_Upgrade\_Tool/Linux\_Upgrade\_Tool\_v1.56.zip

## SD升级启动制作工具

用于制作SD卡升级、SD卡启动、SD卡PCBA测试

## 写号工具

以管理员权限打开软件



用于测试DDR的硬件连接，排查虚焊等硬件问题

用于efuse的烧写，适用于RK3288W/RK3368/RK3399平台

## 用于固件的efuse/otp签名

用于工厂批量烧写固件

用于修改update.img固件

## userdata分区数据预置工具

用于制作Userdata分区预置数据包的工具

## Camera IQ Tool

用于调试ISP图像效果

## 系统调试

### ADB工具

#### 概述

ADB (Android Debug Bridge) 是 Android SDK里的一个工具，用这个工具可以操作管理 Android 模拟器或真实的 Android 设备。主要功能有：

- 运行设备的 shell (命令行)
  - 管理模拟器或设备的端口映射
  - 计算机和设备之间上传/下载文件
  - 将本地 apk软件安装至模拟器或 Android 设备
- ADB 是一个“客户端 - 服务器端”程序，其中客户端主要是指PC，服务器端是Android 设备的实体机器或者虚拟机。根据PC连接Android设备的方式不同，ADB 可以分为两类：
- 网络 ADB：主机通过有线/无线网络（同一局域网）连接到STB设备
  - USB ADB：主机通过 USB 线连接到STB设备

#### USB adb使用说明

USB adb 使用有以下限制：

- 只支持 USB OTG 口
- 不支持多个客户端同时使用（如 cmd 窗口，eclipse等）
- 只支持主机连接一个设备，不支持连接多个设备

连接步骤如下：

- 1、Android设备已经运行 Android 系统，设置->开发者选项->已连接到计算机 打开，usb调试开关打开。
- 2、PC主机只通过 USB 线连接到机器 USB otg 口，然后电脑通过如下命令与Android设备相连。

3、测试是否连接成功，运行“adb devices”命令，如果显示机器的序列号，表示连接成功。

### ADB常用命令详解

(1) 查看设备情况

查看连接到计算机的 Android 设备或者模拟器：

返回的结果为连接至开发机的 Android 设备的序列号或是IP和端口号 (Port) 、 状态。

## (2) 安装apk

将指定的 apk 文件安装到设备上：

示例如下：

重新安装应用：

## (3) 卸载apk

完全卸载：

示例如下：

## (4) 使用 rm移除 apk 文件：

示例如下：

示例说明：移除“system/app”目录下的“WishTV.apk”文件。

## (5) 进入设备和模拟器的shell

进入设备或模拟器的 shell 环境：

## (6) 从电脑上传文件到设备

用 push 命令可以把本机电脑上的任意文件或者文件夹上传到设备。本地路径一般指本机电脑；远程路径一般指 adb 连接的单板设备。

adb push <本地路径> <远程路径>

示例如下：

示例说明：将本地“WishTV.apk”文件上传到 Android 系统的“system/app”目录下。

## (7) 从设备下载文件到电脑

pull 命令可以把设备上的文件或者文件夹下载到本机电脑中。

示例如下：

示例说明：将 Android 系统“system/app”目录下的文件或文件夹下载到本地“F:\”目录下。

## (8) 查看 bug报告

需要查看系统生成的所有错误消息报告，可以运行 adb bugreport指令来实现，该指令会将 Android 系统的dumpsys、dumpstate 与 logcat 信息都显示出来。

### (9) 查看设备的系统信息

在 adb shell下查看设备系统信息的具体命令。

## Logcat工具

Android 日志系统提供了记录和查看系统调试信息的功能。日志都是从各种软件和一些系统的缓冲区中记录下来的，缓冲区可以通过 Logcat 来查看和使用。Logcat 是调试程序用的最多的功能。该功能主要是通过打印日志来显示程序的运行情况。由于要打印的日志量非常大，需要对其进行过滤等操作。

### Logcat命令使用

用 logcat 命令来查看系统日志缓冲区的内容：  
基本格式：

示例如下：

## 常用的日志过滤方式

控制日志输出的几种方式：

- 控制日志输出优先级  
示例如下：

示例说明：显示优先级为 warning 或更高的日志信息。

- 控制日志标签和输出优先级  
示例如下：

示例说明：支持所有的日志信息，除了那些标签为“ActivityManager”和优先级为“Info”以上的、标签为“MyApp”和优先级为“Debug”以上的。

- 只输出特定标签的日志  
示例如下：

或者

示例说明：只输出标签为 WishTV的日志。

- 只输出指定优先级和标签的日志  
示例如下：

示例说明：只输出优先级为 I，标签为 WishTV的日志。

## Procrank工具

Procrank 是 Android 自带的一款调试工具，运行在设备侧的 shell 环境下，用来输出进程的内存快照，便于有效的观察进程的内存占用情况。

包括如下内存信息：

- VSS: Virtual Set Size 虚拟耗用内存大小（包含共享库占用的内存）
- RSS: Resident Set Size 实际使用物理内存大小（包含共享库占用的内存）
- PSS: Proportional Set Size 实际使用的物理内存大小（比例分配共享库占用的内存）
- USS: Unique Set Size 进程独自占用的物理内存大小（不包含共享库占用的内存）

注意：

- USS 大小代表只属于本进程正在使用的内存大小，进程被杀死后会被完整回收；
- VSS/RSS 包含了共享库使用的内存，对查看单一进程内存状态没有参考价值；
- PSS 是按照比例将共享内存分割后，某单一进程对共享内存区的占用情况。

## 使用procrank

执行procrank前需要先让终端获取到root权限

su

命令格式：

常用指令说明：

- v: 按照 VSS 排序
- r: 按照 RSS 排序
- p: 按照 PSS 排序
- u: 按照 USS 排序
- R: 转换为递增[递减]方式排序
- w: 只显示 working set 的统计计数
- W: 重置 working set 的统计计数
- h: 帮助

示例：

输出内存快照：

按照 VSS 降序排列输出内存快照：

默认procrank输出是通过PSS排序。

## 检索指定内容信息

查看指定进程的内存占用状态，命令格式如下：

其中 cmdline 表示需要查找的应用程序名，PID 表示需要查找的应用进程。

输出 systemUI 进程的内存占用状态：

或者：

## 跟踪进程内存状态



通过跟踪内存的占用状态，进而分析进程中是否存在内存泄露场景。使用编写脚本的方式，连续输出进程的内存快照，通过对比 USS 段，可以了解到此进程是否有内存泄露。

示例：输出进程名为 com.android.systemui 的应用内存占用状态，查看是否有泄露：

1、编写脚本 test.sh

2、通过 adb 工具连接到设备后，运行此脚本：./test.sh

## Dumpsys工具

Dumpsys 工具是 Android系统中自带的一款调试工具，运行在设备侧的 shell 环境下，提供系统中正在运行的服务状态信息功能。正在运行的服务是指 Android binder机制中的服务端进程。

dumpsys 输出打印的条件：

- 1、只能打印已经加载到 ServiceManager中的服务；
- 2、如果服务端代码中的 dump 函数没有被实现，则没有信息输出。

## 使用Dumpsys

- 查看Dumpsys帮助  
作用：输出dumpsys帮助信息。

- 查看Dumpsys包含服务列表  
作用：输出dumpsys所有可打印服务信息，开发者可以关注需要调试服务的名称。

- 输出指定服务的信息  
作用：输出指定的服务的 dump 信息。  
格式：dumpsys [servicename]  
示例：输出服务 SurfaceFlinger的信息，可执行命令：

- 输出指定服务和应有进程的信息  
作用：输出指定服务指定应用进程信息。  
格式：dumpsys [servicename] [应用名]  
示例：输出服务名为 meminfo，进程名为 com.android.systemui 的内存信息，执行命令：

注意：服务名称是大小写敏感的，并且必须输入完整服务名称。

## Last log 开启

- 在dts文件里面添加下面两个节点

- 使用方法：

## FIQ模式

当设备死机或者卡住的时候可以在串口输入fiq命令查看系统的状态，具体命令如下：

## 常见问题

### 当前kernel和u-boot版本？

### 如何获取当前SDK对应的RK release版本

Rockchip Android13.0 SDK包括AOSP原始代码和RK修改的代码两部分，其中RK修改的仓库包含在 `.repo/manifests/include` 目录下面的xml中，AOSP默认的仓库在 `.repo/manifests/default.xml`。

版本确认：

- RK修改部分

说明RK的版本是android-13.0-mid-rkr1

- AOSP部分

说明AOSP的版本是android-13.0.0\_r6

当需要提供版本信息的时候提供以上两个版本信息即可。

单个仓库可以直接通过如下命令获取tag信息

RK的版本是以android-13.0-mid-rkrxx的格式递增的，所以当前的最新tag是android-13.0-mid-rkr1

### 如何确认本地SDK已经完整更新RK发布的SDK状态

RK发布SDK版本时会在.repo/manifests/commit/目录下对应提交该版本的commit信息，客户可以通过对比这个commit信息来确认是否有完整更新SDK，具体操作如下：

- 按“如何获取当前SDK对应的RK release版本”的说明先确认SDK的RK版本，下面以RK版本是RKR1为例进行说明；
- 用如下命令保存本地的commit信息

- 通过比较.repo/manifests/commit/commit\_release\_rkr1.xml和release\_manifest\_rkr1\_local.xml，即可确认SDK代码是否更新完整，其中.repo/manifests/commit/commit\_release\_rkr1.xml为RK版本RKR6发布的commit信息。

### uboot和kernel阶段logo图片替换

uboot和kernel阶段的logo分别为开机显示的第一张和第二张logo图片，可以根据产品需求进行修改替换。

uboot logo源文件： `kernel/logo.bmp`

kernel logo源文件： `kernel/logo_kernel.bmp`

如果需要更换某一张，只需用同名的bmp替换掉，重新编译内核即可，编译后的文件在boot.img中。  
说明：Logo图片大小目前只支持到8M以内大小的bmp格式图片，支持8、16、24、32位的bmp。

## 关机充电和低电预充

关机充电和低电预充可以在dts中配置，具体如下：

其中：

rockchip,uboot-charge-on：uboot关机充电，与android关机充电互斥

rockchip,android-charge-on：android关机充电，与uboot关机充电互斥

rockchip,uboot-low-power-voltage：配置低电预充到开机的电压，可以根据实际需求进行配置

rockchip,screen-on-voltage：配置低电预充到亮屏的电压，可以根据实际需求进行配置

## Uboot阶段充电图片打包和替换

充电图片路径，可以直接替换同名文件，格式要求与原文件一样。

如果打开uboot充电，但是没有显示充电图片，可能是图片没有打包到resource.img中，可以按如下命令打包

执行以上命令后uboot充电图片会打包到kernel目录的resource.img中，此时需要再将resource.img打包到boot.img中，可以在android根目录执行./mkiamge.sh，然后烧写rockdev/下面的boot.img即可。

## HDMI IN配置

hdmi in功能SDK默认是关闭的，如需打开，按以下操作：

## RM310 4G配置

4G功能SDK默认是关闭的，如需打开，按以下操作：

## WIFI休眠策略配置

wifi默认休眠策略是休眠一直保持连接，如需休眠断开，按以下操作：

## Recovery旋转配置

支持Recovery旋转0/90/180/270度，默认不旋转（即旋转0度），旋转配置说明如下：

## Android Surface旋转

Android系统显示旋转，可以修改如下配置，配置参数为0/90/180/270



## OTP和efuse说明

OTP支持芯片

- RK3326
- PX30
- RK3566
- RK3568
- RK3588

EFUSE支持芯片

- RK3288
- RK3368
- RK3399

固件签名和otp/efuse烧写参考文档

## 代码中如何判断设备的OTP/EFUSE是否已经烧写

OTP/EFUSE的状态会通过kernel的cmdline进行传递，cmdline中的fuse.programmed用来标识OTP/EFUSE状态，具体如下：

- "fuse.programmed=1": 软件固件包已经进行了secure-boot签名，硬件设备的efuse/otp已经被烧写。
- "fuse.programmed=0": 软件固件包已经进行了secure-boot签名，硬件设备的efuse/otp没有被烧写。
- cmdline中没有fuse.programmed: 软件固件包没有进行secure-boot签名（Miniloader不传递），或者Miniloader太旧没有支持传递。

## 开关selinux

如下修改，false为关闭，true为打开

## 开机弹出“Android系统出现问题”警告

出现警告框的原因有两种：

1. 固件不匹配，system/boot/vendor三个fingerprint不一致，不是同一套固件。
2. 机器打开支持了IO调试功能的config，编译时，使用文档前面所说的内核编译命令即可关闭。
3. 对于需要使用IO调试功能的项目，可以直接不管上述两种原因，直接合入frameworks/base下的patch去掉弹窗：

## 如何打开设置中以太网的设置项

系统设置中默认没有以太网设置的选项，如果项目中需要以太网可以按如下配置打开：

## 关于AVB和security boot的操作

AVB和security boot的操作参考文档

## IO命令无法使用

IO命令需要依赖DEVMEM，而DEVMEM默认是关闭的，所以导致IO默认无法使用，如果调试需要使用IO命令可以按如下修改：

如果是GO的产品则需要修改：

删除掉下面这行：

如果要编译Android，则还需要修改如下代码

## SN号的命令规则

SN号必须以字母开头，长度14个字节以内。

## Kernel编译报LZ4的错误

Kernel编译的时候报如下错误：

```
SORTEX    vmlinux
SYSMAP    System.map
OBJCOPY   arch/arm/boot/Image
Kernel:   arch/arm/boot/Image is ready
SHIPPED   arch/arm/boot/compressed/hyp-stub.S
SHIPPED   arch/arm/boot/compressed/fdt_rw.c
SHIPPED   arch/arm/boot/compressed/fdt.h
SHIPPED   arch/arm/boot/compressed/libfdt.h
SHIPPED   arch/arm/boot/compressed/libfdt_internal.h
SHIPPED   arch/arm/boot/compressed/fdt_ro.c
SHIPPED   arch/arm/boot/compressed/fdt_wip.c
SHIPPED   arch/arm/boot/compressed/fdt.c
SHIPPED   arch/arm/boot/compressed/libfuncs.S
SHIPPED   arch/arm/boot/compressed/ashldi3.S
SHIPPED   arch/arm/boot/compressed/bswapsdi2.S
LDS       arch/arm/boot/compressed/vmlinux.lds
AS        arch/arm/boot/compressed/head.o
LZ4       arch/arm/boot/compressed/piggy_data
Incorrect parameters
Usage :
    lz4 [arg] [input] [output]

input  : a filename
         with no FILE, or when FILE is - or stdin, read standard input
Arguments :
    -1      : Fast compression (default)
    -9      : High compression
    -d      : decompression (default for .lz4 extension)
    -z      : force compression
    -f      : overwrite output without prompting
    -h/-H   : display help/long help and exit
arch/arm/boot/compressed/Makefile:191: recipe for target 'arch/arm/boot/compressed/piggy_data' failed
make[2]: *** [arch/arm/boot/compressed/piggy_data] Error 1
arch/arm/boot/Makefile:71: recipe for target 'arch/arm/boot/compressed/vmlinux' failed
make[1]: *** [arch/arm/boot/compressed/vmlinux] Error 2
arch/arm/Makefile:351: recipe for target 'zImage' failed
make: *** [zImage] Error 2
```

问题原因：

系统自带的lz4版本太低，要求1.8.3及以上版本

解决方法：

直接拷贝android编译出来的lz4覆盖系统的lz4

## Android Samba功能

## NFS启动

参考文档及补丁：

## 多屏异显异触

参考文档

## 多屏异声

参考文档

## 附录 A 编译开发环境搭建 Compiling and development environment setup

### Initializing a Build Environment

This section describes how to set up your local work environment to build the Android source files. You must use Linux or Mac OS; building under Windows is not currently supported.

For an overview of the entire code-review and code-update process, see Life of a Patch.

Note: All commands in this site are preceded by a dollar sign (\$) to differentiate them from output or entries within files. You may use the Click to copy feature at the top right of each command box to copy all lines without the dollar signs or triple-click each line to copy it individually without the dollar sign.

### Choosing a Branch

Some requirements for the build environment are determined by the version of the source code you plan to compile. For a full list of available branches, see Build Numbers. You can also choose to download and build the latest source code (called master), in which case you will simply omit the branch specification when you initialize the repository.

After you have selected a branch, follow the appropriate instructions below to set up your build environment.

### Setting up a Linux build environment

These instructions apply to all branches, including master.

The Android build is routinely tested in house on recent versions of Ubuntu LTS (14.04) and Debian testing. Most other distributions should have the required build tools available.

For Gingerbread (2.3.x) and newer versions, including the master branch, a 64-bit environment is required. Older versions can be compiled on 32-bit systems.

Note: See Requirements for the complete list of hardware and software requirements, then follow the detailed instructions for Ubuntu and Mac OS below.

# Installing the JDK

The master branch of Android in the Android Open Source Project (AOSP) comes with prebuilt versions of OpenJDK below prebuilts/jdk/ so no additional installation is required.

Older versions of Android require a separate installation of the JDK. On Ubuntu, use OpenJDK. See JDK Requirements for precise versions and the sections below for instructions.

## For Ubuntu >= 15.04

Run the following:

## For Ubuntu LTS 14.04

There are no available supported OpenJDK 8 packages for Ubuntu 14.04. The Ubuntu 15.04 OpenJDK 8 packages have been used successfully with Ubuntu 14.04. Newer package versions (e.g. those for 15.10, 16.04) were found not to work on 14.04 using the instructions below.

1. Download the .deb packages for 64-bit architecture from [old-releases.ubuntu.com](http://old-releases.ubuntu.com):

2. Optionally, confirm the checksums of the downloaded files against the SHA256 string listed with each package above. For example, with the sha256sum tool:

3. Install the packages:

Run dpkg for each of the .deb files you downloaded. It may produce errors due to missing dependencies:

To fix missing dependencies:

Update the default Java version - optional

Optionally, for the Ubuntu versions above update the default Java version by running:

Note: If, during a build, you encounter version errors for Java, see [Wrong Java version for likely causes and solutions](#).

## Installing required packages (Ubuntu 14.04)

You will need a 64-bit version of Ubuntu. Ubuntu 14.04 is recommended.

Note: To use SELinux tools for policy analysis, also install the python-networkx package. Note: If you are using LDAP and want to run ART host tests, also install the libnss-sss:i386 package.

# Configuring USB Access



Under GNU/Linux systems (and specifically under Ubuntu systems), regular users can't directly access USB devices by default. The system needs to be configured to allow such access. The recommended approach is to create a file `/etc/udev/rules.d/51-android.rules` (as the root user) and to copy the following lines in it. `must` be replaced by the actual username of the user who is authorized to access the phones over USB.

Those new rules take effect the next time a device is plugged in. It might therefore be necessary to unplug the device and plug it back into the computer. This is known to work on both Ubuntu Hardy Heron (8.04.x LTS) and Lucid Lynx (10.04.x LTS). Other versions of Ubuntu or other variants of GNU/Linux might require different configurations. References : <http://source.android.com/source/initializing.html>

## 附录 B SSH公钥操作说明 SSH public key operation instruction

### 附录 B-1 SSH公钥生成 SSH public key generation

使用如下命令生成：

Use the following command to generate:

请将`user@host`替换成您的邮箱地址。  
Please replace `user@host` with your email address.

命令运行完成会在你的目录下生成`key`文件。  
It will generate the key file in your directory after the command is executed successfully.

请妥善保管生成的私钥文件`id_rsa`和密码，并将`id_rsa.pub`发邮件给SDK发布服务器的管理员。  
Please keep carefully the generated private key file `id_rsa` and password, and send `id_rsa.pub` to SDK release server admin through email.

### 附录 B-2 使用key-chain管理密钥 Use key-chain to manage the key

推荐您使用比较简易的工具`keychain`管理密钥。  
Recommend you use the simple tool `keychain` to manage the key.  
具体使用方法如下：  
The detailed usage is as follows:

1. 安装`keychain`软件包：  
Install `keychain` software package:

2. 配置使用密钥：  
Configure to use the key:

增加下面这行：  
Add the following command:

其中，id\_rsa是私钥文件名称。

Among which, id\_rsa is the file name of the private key.

以上配置以后，重新登录控制台，会提示输入密码，只需输入生成密钥时使用的密码即可，若无密码可不输入。

Log in the console again after configuring as above, and it will prompt to input the password.

Only need to input the password used for generating the key if there is one.

另外，请尽量不要使用sudo或root用户，除非您知道如何处理，否则将导致权限以及密钥管理混乱。

Besides, please avoid using sudo or root user unless you know clearly how to deal with, otherwise it will cause the authority and key management problems.

## 附录 B-3 多台机器使用相同ssh公钥 Multiple devices use the same ssh public key

在不同机器使用，可以将你的ssh私钥文件id\_rsa拷贝到要使用的机器的“~/.ssh/id\_rsa”即可。

In order to use on different devices, you can copy ssh private key file id\_rsa to the target device “~/.ssh/id\_rsa”.

在使用错误的私钥会出现如下提示，请注意替换成正确的私钥。

Below hint will show up if using the wrong private key. Please replace with the correct private key.

添加正确的私钥后，就可以使用git 克隆代码，如下图。

After adding the correct private key, you can use git to clone code, shown as below picture:

添加ssh私钥可能出现如下提示错误。

Below error may occur when adding ssh private key:

在console输入如下命令即可解决。

Input the following command at console can fix it.

## 附录 B-4 一台机器切换不同ssh公钥 Switch different ssh public keys on one device

可以参考ssh\_config文档配置ssh。

You can refer to ssh\_config document to configure ssh.

通过如下命令，配置当前用户的ssh配置。

Use the following commands to configure ssh for current user.

如图，将ssh使用另一个目录的文件“~/.ssh1/id\_rsa”作为认证私钥。通过这种方法，可以切换不同的密钥。

As below picture, identify another directory ssh file “~/.ssh1/id\_rsa” as certificate private key. In this way, you can switch different keys.

## 附录 B-5 密钥权限管理 Key authority management

服务器可以实时监控某个key的下载次数、IP等信息，如果发现异常将禁用相应的key的下载权限。  
The server can real-time monitor for the specific key the download times, IP and other information. If any abnormal case is found, it will prohibit the download authority of the corresponding key.

请妥善保管私钥文件。并不要二次授权与第三方使用。

Please keep carefully the private key file. DO NOT re-authorize it to the third party.

## **附录 B-6 Git权限申请说明 Git authority application instruction**

参考上述章节，生成公钥文件，发邮件至 [fae@rock-chips.com](mailto:fae@rock-chips.com)，申请开通SDK代码下载权限。

Refer to above chapters, generate the public key file, and send email to [fae@rock-chips.com](mailto:fae@rock-chips.com) applying for SDK code download authority.