

密级状态：绝密() 秘密() 内部() 公开(√)

RK3229_ANDROID8.1-BOX-SDK_V1.0_ 201804233 发布说明

(第一系统产品部，技术部)

| | | |
|---------------------------------------|-------|------------|
| 文件状态： [] 正在修改 [√] 正式发布 | 当前版本： | V1.0 |
| | 作 者： | HuangJC |
| | 完成日期： | 2018-04-23 |
| | 审 核： | CW, ZXZ |
| | 完成日期： | 2018-04-23 |

福州瑞芯微电子股份有限公司

Fuzhou Rockchips Semiconductor Co., Ltd

(版本所有, 翻版必究)

版 本 历 史

| 版本号 | 作者 | 修改日期 | 修改说明 | 备注 |
|------|---------|------------|-----------|----|
| V0.1 | HuangJC | 2018.03.26 | Beta 版本发布 | |
| V1.0 | HuangJC | 2018.04.23 | 正式版本发布 | |
| | | | | |

目 录

| | |
|----------------------------|----|
| 版 本 历 史..... | 2 |
| 目 录..... | 3 |
| 1 概 述..... | 5 |
| 2 主要支持功能..... | 5 |
| 3 SDK 获取说明..... | 5 |
| 3.1 获取 SDK..... | 5 |
| 3.2 SDK 镜像..... | 6 |
| 4 软件开发指南..... | 6 |
| 4.1 开发指南..... | 6 |
| 5 SDK 编译说明..... | 7 |
| 5.1 JDK 安装..... | 7 |
| 5.2 编译模式..... | 7 |
| 5.3 代码编译..... | 8 |
| 5.3.1 uboot 编译步骤..... | 8 |
| 5.3.2 kernel 编译步骤..... | 8 |
| 5.3.3 Android 编译步骤..... | 8 |
| 5.3.4 固件打包..... | 8 |
| 5.3.5 自动编译脚本..... | 9 |
| 6 刷机说明..... | 9 |
| 附录 A 编译开发环境搭建..... | 11 |
| 附录 A-1 硬件要求..... | 11 |
| 附录 A-2 软件要求..... | 11 |
| 附录 A-3 设置 Linux 编译环境..... | 13 |
| 附录 A-4 设置 Mac OS 编译环境..... | 15 |
| 附录 B SSH 公钥操作说明..... | 18 |

| | | |
|--------|------------------------|----|
| 附录 B-1 | SSH 公钥生成..... | 18 |
| 附录 B-2 | 使用 key-chain 管理密钥..... | 19 |
| 附录 B-3 | 多台机器使用相同 ssh 公钥..... | 19 |
| 附录 B-4 | 一台机器切换不同 ssh 公钥..... | 20 |
| 附录 B-5 | 密钥权限管理..... | 21 |
| 附录 B-6 | Git 权限申请说明..... | 21 |

1 概述

本 SDK 是基于谷歌 Android8.1 32bit 系统，适配瑞芯微 RK3229 芯片的软件包，适用于 RK3229 Evb 开发板、RK3229 Box 产品及基于其上所有的开发产品。

2 主要支持功能

| 参数 | 模块名 |
|------|---|
| 数据通信 | WiFi、以太网卡、USB、SDCard |
| 应用程序 | RkTvLauncher、媒体中心、TvSetting、资源管理器、WiFi Display、浏览器、计算器、相机 |

3 SDK 获取说明

3.1 获取 SDK

SDK 通过瑞芯微代码服务器对外发布。其编译开发环境，参考[附录 A 编译开发环境搭建](#)。

客户向瑞芯微技术窗口申请 SDK，需同步提供 SSH 公钥进行服务器认证授权，获得授权后即可同步代码。关于瑞芯微代码服务器 SSH 公钥授权，请参考[附录 B SSH 公钥操作说明](#)。

RK3229_ANDROID8.1_BOX_SDK 下载命令如下：

```
repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u  
ssh://git@www.rockchip.com.cn/gerrit/rk/platform/manifest -b android-8.1 -m  
rk3229_box_oreo_release.xml
```

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 ssh://git@www.rockchip.com.cn/repo/rk/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，开发者可以通过这种方式，获得 SDK 代码的初始压缩包，该压缩包解压得到的源码，与通过 Repo

下载的源码是一致的。

以 rk3229_android8.1_box_v1.0_20180420.tgz 为例，拷贝到该初始化包后，通过如下命令可检出源码：

```
mkdir rk3229
tar xvf rk3229_android8.1_box_v1.0_20180420.tgz -C rk3229
cd rk3229
.repo/repo/repo sync -l
.repo/repo/repo sync -c --no-tags
```

后续开发者可根据 FAE 窗口定期发布的更新说明，通过“.repo/repo/repo sync -c --no-tags”命令同步更新。

3.2 SDK 镜像

客户可以自己搭建 SDK 的 REPO 镜像服务器，来方便团队协同开发，镜像 SDK 的命令如下：

```
repo init --mirror --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo
-u ssh://git@www.rockchip.com.cn/gerrit/rk/platform/manifest -b android-8.1 -m
rk3229_box_oreo_release.xml
```

为方便客户快速获取，我们同时也会提供 mirror 的初始压缩包。解压后与上述命令下载的源码是一致的。

更多关于 REPO 镜像服务器搭建可以参考 RKDocs\common\RKTools manuals 目录下的《REPO 镜像服务器搭建和管理_V2.2_20131231.pdf》

4 软件开发指南

4.1 开发指南

RK3229 Box SDK Kernel 版本：Linux4.4，Android 版本：8.1.0，为帮助开发工程师更快上手熟悉 SDK 的开发调试工作，随 SDK 发布《RK3229 Android8.1-Box 软件开发指南 V1.01-20180418》。

可在 RKDocs\rk322x 目录下获取，并会不断完善更新。

5 SDK 编译说明

5.1 JDK 安装

Android8.1 系统编译依赖于 Java 8。编译之前需安装 OpenJDK。

安装命令如下。

```
sudo apt-get install openjdk-8-jdk
```

配置 Java 环境变量，例如，安装路径为/usr/lib/jvm/java-8-openjdk-amd64，可在终端执行如下命令配置环境变量。

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export PATH=$JAVA_HOME/bin:$PATH
export CLASSPATH=.:$JAVA_HOME/lib:$JAVA_HOME/lib/tools.jar
```

SDK 带有 Open JDK8 的配置脚本，在工程根目录下，命名为 javaenv.sh。

可直接执行以下命令，配置 JDK：

```
source javaenv.sh
```

5.2 编译模式

SDK 默认以 userdebug 模式编译。

使用 ADB 时，需要先执行 adb root 使 shell 获取 root 权限，进而执行其它像 adb remount、adb push 等操作。

注意：对 Android 8.1 系统，默认第一次执行 adb remount 会提示无效，可直接执行以下命令，解除 remount 分区校验：

```
adb disable-verity
reboot
```

下一次开机后即可正常操作 adb remount。

5.3 代码编译

5.3.1 uboot 编译步骤

```
make ARCHV=arm rk322x_box_defconfig  
make ARCHV=arm -j12
```

编译完，会生成 trust.img、rk322x_loader_vx.xx.xxx.bin、uboot.img 三个文件。

5.3.2 kernel 编译步骤

RK3229 BOX EVB 板配置与编译命令如下：

```
make ARCH=arm rockchip_defconfig  
make ARCH=arm rk3229-evb-android.img -j8
```

编译完成后，在 kernel 根目录生成 kernel.img，resource.img 两个镜像文件。

5.3.3 Android 编译步骤

客户按实际编译环境配置好 JDK 环境变量后，按照以下步骤配置完后，执行 make 即可。

```
$ source build/envsetup.sh  
$ lunch
```

选择 rk322x_box-userdebug

```
$ make -j4
```

5.3.4 固件打包

完成以上编译后，执行 SDK 根目录下的 mkimage.sh 脚本生成固件，所有烧写所需的镜像将都会拷贝于 rockdev/Image-rk322x_box 目录。

```
rockdev/Image-rk322x_box/  
├── baseparameter.img  
├── boot.img  
├── kernel.img  
├── MiniLoaderAll.bin  
├── misc.img  
├── oem.img  
└── parameter.txt
```



```
|—— pcba_small_misc.img
|—— pcba_whole_misc.img
|—— recovery.img
|—— resource.img
|—— system.img
|—— trust.img
|—— uboot.img
|—— vendor.img
```

5.3.5 自动编译脚本

为了提高编译的效率，降低人工编译可能出现的误操作，SDK 中集成了全自动化编译脚本，方便固件编译、备份。脚本的执行命令如下（在 SDK 工程根目录下）：

```
source build.sh
```

该脚本会自动配置 JDK 环境，编译 U-Boot、kernel 和 Android，然后生成固件并打包生成 update.img。

另外脚本会将编译生成固件拷贝至 **IMAGE** 目录下。每次编译都会新建目录保存，自动备份调试开发过程的固件版本，并存放固件版本的各类信息。

6 刷机说明

刷机说明详见 RKDocs\common\RKTools manuals 目录下《Android 开发工具手册.pdf》。

SDK 提供烧写工具，如下图所示。编译生成相应的固件后，进入烧写模式，即可进行刷机。对于已烧过其它固件的机器，可以选择重新烧录固件，或是选择低格设备，擦除 **idb**，然后进行刷机。



注：烧写前，需安装最新的的 USB 驱动，驱动详见

RKTools/windows/
└── DriverAssitant_v4.5

注意，RK3229 ANDROID 8.1 最新版本使用了新的 spare 脚本来减少 system.img 的大小，需要使用 v2.51 版本以上的工具进行烧写，否则会出现无法启动 Android 的情况。

附录 A 编译开发环境搭建

本章节介绍了如何设置本地工作环境来编译 Android 源文件。您需要使用 Linux 或 Mac OS。目前不支持在 Windows 环境下进行编译。

Android 8.1 建议软硬件配置：

- 操作系统：64 位 Ubuntu 14.04 及以上
- 硬盘空间：最小 150GB
- Python 版本：2.7.6 及以上
- JDK 版本：[OpenJDK 8](#)

注意：

从 Android (2.3.x) Gingerbread 版本开始，编译环境都要求为 64 位操作系统。

References : <https://source.android.com/setup/initializing>

附录 A-1 硬件要求

您的开发工作站必须达到或超出以下软硬件要求：

- 如果是 Gingerbread (2.3.x) 及更高版本（包括 master 分支），需要使用 64 位环境。如果是较低版本，则可以在 32 位系统中进行编译。
- 如果是校验代码，至少需要 100GB 可用磁盘空间；如果要进行编译，则还需要 150GB。如果要进行多次编译或使用 ccache，则需要更多空间。
- 如果您在虚拟机中运行 Linux，则至少需要 16GB 的 RAM/交换空间。

附录 A-2 软件要求

[Android 开源项目 \(AOSP\)](#) master 分支历来都是在 Ubuntu Long Term Support (LTS) 版本中进行开发和测试，但您也可以使用其他 Ubuntu 分发版本。要查看建议使用的版本，请参阅下面的列表。

附录 A-2-1 操作系统和 JDK

如果您要针对 AOSP master 分支进行开发，请使用下列操作系统之一：Ubuntu 14.04 (Trusty)/Mac OS v10.10 (Yosemite) 或更高版本（具有 Xcode 4.5.2 和命令行工具）。

附录 A-2-2 主要软件包

- python.org 中提供的 Python 2.6 - 2.7
- gnu.org 中提供的 GNU Make 3.81 - 3.82
- git-scm.com 中提供的 Git 1.7 或更高版本

附录 A-2-3 操作系统

Android 通常是在 GNU/Linux 或 Mac OS 操作系统中进行编译。您也可以使用虚拟机在不支持的系统（例如 Windows）中编译 Android。

➤ GNU/Linux

- Android 6.0 (Marshmallow) - AOSP master: Ubuntu 14.04 (Trusty)
- Android 2.3.x (Gingerbread) - Android 5.x (Lollipop): Ubuntu 12.04 (Precise)
- Android 1.5 (Cupcake) - Android 2.2.x (Froyo): Ubuntu 10.04 (Lucid)

➤ Mac OS (Intel/x86)

- Android 6.0 (Marshmallow) - AOSP master: Mac OS v10.10 (Yosemite) 或更高版本，具有 Xcode 4.5.2 和命令行工具
- Android 5.x (Lollipop): Mac OS v10.8 (Mountain Lion)，具有 Xcode 4.5.2 和命令行工具
- Android 4.1.x-4.3.x (Jelly Bean) - Android 4.4.x (KitKat): Mac OS v10.6 (Snow Leopard) 或 Mac OS X v10.7 (Lion)，以及 Xcode 4.2 (Apple 的开发者工具)

- Android 1.5 (Cupcake) - Android 4.0.x (Ice Cream Sandwich): Mac OS v10.5 (Leopard) 或 Mac OS X v10.6 (Snow Leopard), 以及 Mac OS X v10.5 SDK

注意: 请考虑在 GNU/Linux (而不是其他操作系统) 上进行编译。Android 编译系统通常使用编译设备上运行的 ART 来预编译系统 dex 文件。由于 ART 只能在 Linux 上运行, 因此编译系统会在非 Linux 操作系统上跳过这个预编译步骤, 从而导致 Android 编译的性能下降。

附录 A-2-4 JDK

- Android 7.0 (Nougat) - Android 8.0 (O) : Ubuntu - [OpenJDK 8](#) ; Mac OS - [jdk 8u45 或更高版本](#)
- Android 5.x (Lollipop) - Android 6.0 (Marshmallow) : Ubuntu - [OpenJDK 7](#) ; Mac OS - [jdk-7u71-macosx-x64.dmg](#)
- Android 2.3.x (Gingerbread) - Android 4.4.x (KitKat) : Ubuntu - [Java JDK 6](#) ; Mac OS - [Java JDK 6](#)
- Android 1.5 (Cupcake) - Android 2.2.x (Froyo) : Ubuntu - [Java JDK 5](#)

附录 A-3 设置 Linux 编译环境

以下说明适用于所有分支 (包括 `master`)。

我们会定期在最近推出的一些 Ubuntu LTS (14.04) 版本中对 Android 编译过程进行内部测试, 但大多数 Ubuntu 分发版本都应该有所需的编译工具。欢迎向我们报告在其他分发版本中的测试结果 (无论结果是成功还是失败)。

如果是 Gingerbread (2.3.x) 及更高版本 (包括 `master` 分支), 需要使用 64 位环境。如果是较低版本, 则可以在 32 位系统中进行编译。

附录 A-3-1 安装 JDK

在 Ubuntu 上, 请使用 [OpenJDK](#)。要了解确切的版本, 请参阅 [JDK](#) 要求; 要了解相关说

明，请参阅以下各个部分。

➤ 如果 **Ubuntu >= 15.04**

请运行以下命令：

```
sudo apt-get updatesudo apt-get install openjdk-8-jdk
```

➤ 如果是 **Ubuntu LTS 14.04**

目前没有适用于 Ubuntu 14.04 的受支持 OpenJDK 8 程序包。[Ubuntu 15.04 OpenJDK 8](#) 软件包能够在 Ubuntu 14.04 中顺利使用。我们发现，按照以下说明操作时，更高的程序包版本（例如适合 15.10、16.04 的版本）在 Ubuntu 14.04 中无法正常工作。

1. 从 old-releases.ubuntu.com 下载适用于 64 位架构的 .deb 软件包：

- [openjdk-8-jre-headless_8u45-b14-1_amd64.deb](#)

(SHA256 :

0f5aba8db39088283b51e00054813063173a4d8809f70033976f83e214ab56c0)

- [openjdk-8-jre_8u45-b14-1_amd64.deb](#)

(SHA256 :

9ef76c4562d39432b69baf6c18f199707c5c56a5b4566847df908b7d74e15849)

- [openjdk-8-jdk_8u45-b14-1_amd64.deb](#)

(SHA256 :

6e47215cf6205aa829e6a0a64985075bd29d1f428a4006a80c9db371c2fc3c4c)

2. （可选）对照随以上每个程序包列出的 SHA256 字符串，确认已下载文件的校验和。例如，

使用 sha256sum 工具：

```
sha256sum {downloaded.deb file}
```

3. 安装程序包：

```
sudo apt-get update
```

为下载的每个 .deb 文件运行 dpkg。运行过程中可能会因缺少依赖项而出现错误：

```
sudo dpkg -i {downloaded.deb file}
```

解决缺少依赖项的问题：

```
sudo apt-get -f install
```

➤ 更新默认的 **Java** 版本 - 可选

（可选）对于以上 Ubuntu 版本，您可以通过运行以下命令来更新默认的 Java 版本：

```
sudo update-alternatives --config javasudo update-alternatives --config javac
```

在编译过程中，如果您遇到 Java 版本错误，请按照[错误的 Java 版本](#)部分中的说明设置其路径。

附录 A-3-2 安装所需的程序包 (Ubuntu 14.04)

您将需要 64 位版本的 Ubuntu。建议您使用 Ubuntu 14.04。

```
sudo apt-get install git-core gnupg flex bison gperf build-essential zip curl  
zlib1g-dev gcc-multilib g++-multilib libc6-dev-i386 lib32ncurses5-dev  
x11proto-core-dev libx11-dev lib32z-dev ccache libgl1-mesa-dev libxml2-utils  
xsltproc unzip
```

注意：要使用 SELinux 工具进行政策分析，您还需要安装 **python-networkx** 软件包。

如果您使用 LDAP 并且希望运行 ART 主机测试，则还需要安装 **libnss-sss:i386** 软件包。

附录 A-3-3 安装所需的程序包 (Ubuntu 12.04)

您可以使用 Ubuntu 12.04 来编译较低版本的 Android。master 或最近推出的一些版本不支持 Ubuntu 12.04。

```
sudo apt-get install git gnupg flex bison gperf build-essential zip curl libc6-dev  
libncurses5-dev:i386 x11proto-core-dev libx11-dev:i386 libreadline6-dev:i386  
libgl1-mesa-glx:i386 libgl1-mesa-dev g++-multilib mingw32 tofrodos  
python-markdown libxml2-utils xsltproc zlib1g-dev:i386sudo ln -s  
/usr/lib/i386-linux-gnu/mesa/libGL.so.1 /usr/lib/i386-linux-gnu/libGL.so
```

附录 A-4 设置 Mac OS 编译环境

在默认安装过程中，Mac OS 会在一个保留大小写但不区分大小写的文件系统中运行。Git 并不支持此类文件系统，而且此类文件系统会导致某些 Git 命令（例如 **git status**）的行为出现异常。因此，我们建议您始终在区分大小写的文件系统中对 AOSP 源文件进行操作。使用下文中介

绍的磁盘映像可以非常轻松地做到这一点。

有了适当的文件系统，在新型 Mac OS 环境中编译 **master** 分支就会变得非常简单。要编译较低版本的分支，则需要一些额外的工具和 SDK。

附录 A-4-1 创建区分大小写的磁盘映像

您可以使用磁盘映像在现有的 Mac OS 环境中创建区分大小写的文件系统。要创建磁盘映像，请启动磁盘工具，然后选择“新建映像”。完成编译至少需要 25GB 空间；更大的空间能够更好地满足未来的需求。使用稀疏映像有助于节省空间，而且以后可以随着需求的增加进行扩展。请务必选择“Case sensitive, Journaled”存储卷格式。

您也可以通过 shell 使用以下命令创建磁盘映像：

```
hdiutil create -type SPARSE -fs 'Case-sensitive Journaled HFS+' -size 40g  
~/android.dmg
```

这将创建一个 **.dmg**（也可能是 **.dmg.sparseimage**）文件，该文件在装载后可用作具有 Android 开发所需格式的存储卷。

如果您以后需要更大的存储卷，还可以使用以下命令来调整稀疏映像的大小：

```
hdiutil resize -size <new-size-you-want>g ~/android.dmg.sparseimage
```

对于存储在主目录下的名为 **android.dmg** 的磁盘映像，您可以向 **~/.bash_profile** 中添加辅助函数：

- 要在执行 **mountAndroid** 时装载磁盘映像，请运行以下命令：

```
# mount the android file image  
mountAndroid() { hdiutil attach ~/android.dmg -mountpoint  
/Volumes/android; }
```

注意：如果系统创建的是 **.dmg.sparseimage** 文件，请将 **~/android.dmg** 替换为 **~/android.dmg.sparseimage**。

- 要在执行 **umountAndroid** 时卸载磁盘映像，请运行以下命令：

```
# unmount the android file image  
umountAndroid() { hdiutil detach /Volumes/android; }
```

装载 **android** 存储卷后，您将在其中开展所有工作。您可以像对待外接式存储盘一样将其弹

出（卸载）。

附录 A-4-2 安装 JDK

要查看要在开发各种 Android 版本时使用的 Java 版本，请参阅上述软件要求。

➤ 安装所需的程序包

1. 使用以下命令安装 Xcode 命令行工具：

```
xcode-select --install
```

对于较低版本的 Mac OS (10.8 或更低版本)，您需要通过 [Apple 开发者网站](#) 安装 Xcode。

如果您尚未注册成为 Apple 开发者，则需要创建一个 Apple ID 才能下载。

2. 通过 [macports.org](#) 安装 MacPorts。

注意：请确保在路径中 `/opt/local/bin` 显示在 `/usr/bin` 之前。否则，请将以下内容添加到 `~/ .bash_profile` 文件中：

```
export PATH=/opt/local/bin:$PATH
```

注意：如果主目录中没有 `.bash_profile` 文件，请创建一个。

3. 通过 MacPorts 获取 Make、Git 和 GPG 程序包：

```
POSIXLY_CORRECT=1 sudo port install gmake libsdl git gnupg
```

如果您使用 Mac OS X v10.4，还需要安装 bison：

```
POSIXLY_CORRECT=1 sudo port install bison
```

➤ 设置文件描述符数量上限

在 Mac OS 中，可同时打开的文件描述符的默认数量上限太低，在高度并行的编译流程中，可能会超出此上限。

要提高此上限，请将下列行添加到 `~/ .bash_profile` 中：

```
ulimit -S -n 1024
```

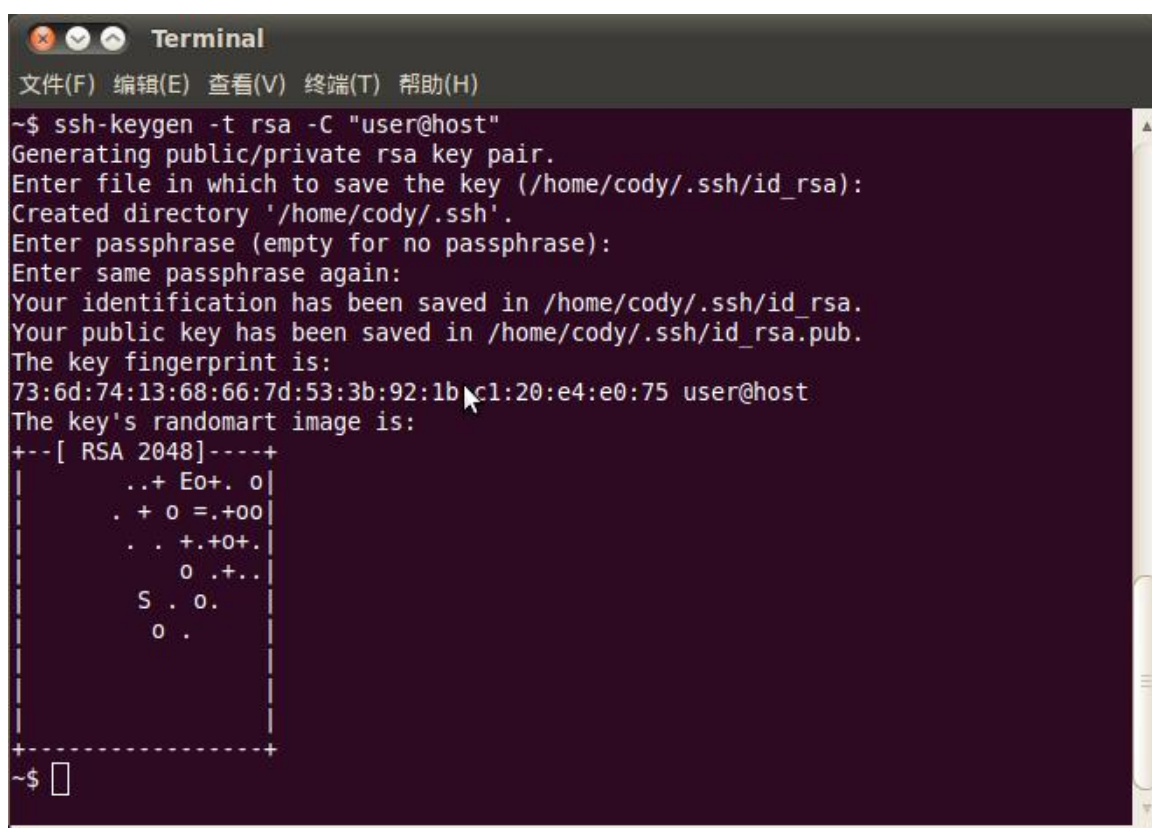
附录 B SSH 公钥操作说明

附录 B-1 SSH 公钥生成

使用如下命令生成：

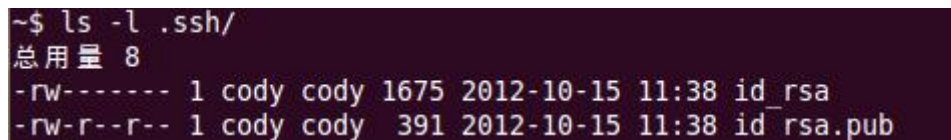
```
ssh-keygen -t rsa -C "user@host"
```

请将 **user@host** 替换成您的邮箱地址。



```
Terminal
文件(F) 编辑(E) 查看(V) 终端(T) 帮助(H)
~$ ssh-keygen -t rsa -C "user@host"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/cody/.ssh/id_rsa):
Created directory '/home/cody/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/cody/.ssh/id_rsa.
Your public key has been saved in /home/cody/.ssh/id_rsa.pub.
The key fingerprint is:
73:6d:74:13:68:66:7d:53:3b:92:1b:c1:20:e4:e0:75 user@host
The key's randomart image is:
+--[ RSA 2048 ]-----+
|      .+ Eo+. o      |
|    . + o =.+oo      |
|   . . +.+o+.       |
|      o .+. .       |
|     S . o.         |
|      o .           |
+-----+
~$
```

命令运行完成会在你的目录下生成 **key** 文件。



```
~$ ls -l .ssh/
总用量 8
-rw----- 1 cody cody 1675 2012-10-15 11:38 id_rsa
-rw-r--r-- 1 cody cody 391 2012-10-15 11:38 id_rsa.pub
```

请妥善保存生成的私钥文件 **id_rsa** 和密码，并将 **id_rsa.pub** 发邮件给 SDK 发布服务器的管理员。

附录 B-2 使用 key-chain 管理密钥

推荐您使用比较简易的工具 keychain 管理密钥。

具体使用方法如下：

1. 安装 keychain 软件包：

```
$sudo aptitude install keychain
```

2. 配置使用密钥：

```
$vim ~/.bashrc
```

增加下面这行：

```
eval `keychain --eval ~/.ssh/id_rsa`
```

其中，id_rsa 是私钥文件名称。

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。

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。

附录 B-3 多台机器使用相同 ssh 公钥

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。

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。

```
~/tmp$ git clone git@172.16.10.211:rk292x/mid/4.1.1_r1
Initialized empty Git repository in /home/cody/tmp/4.1.1_r1/.git/
The authenticity of host '172.16.10.211 (172.16.10.211)' can't be established.
RSA key fingerprint is fe:36:dd:30:bb:83:73:e1:0b:df:90:e2:73:e4:61:46.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.16.10.211' (RSA) to the list of known hosts.
git@172.16.10.211's password: 
```

添加正确的私钥后，就可以使用 git 克隆代码，如下图。

```
~$ cd tmp/
~/tmp$ git clone git@172.16.10.211:rk292x/mid/4.1.1_r1
Initialized empty Git repository in /home/cody/tmp/4.1.1_r1/.git/
The authenticity of host '172.16.10.211 (172.16.10.211)' can't be established.
RSA key fingerprint is fe:36:dd:30:bb:83:73:e1:0b:df:90:e2:73:e4:61:46.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.16.10.211' (RSA) to the list of known hosts.
remote: Counting objects: 237923, done.
remote: Compressing objects: 100% (168382/168382), done.
Receiving objects: 9% (21570/237923), 61.52 MiB | 11.14 MiB/s
```

添加 ssh 私钥可能出现如下提示错误。

Agent admitted failure to sign using the key

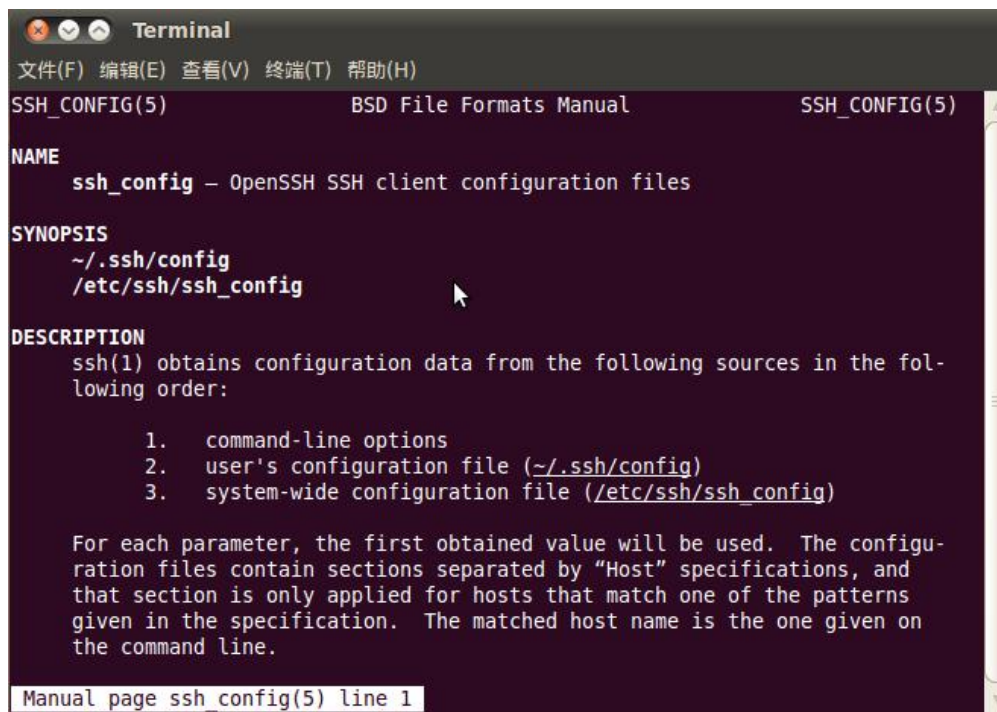
在 console 输入如下命令即可解决。

```
ssh-add ~/.ssh/id_rsa
```

附录 B-4 一台机器切换不同 ssh 公钥

可以参考 ssh_config 文档配置 ssh。

```
~$ man ssh_config
```

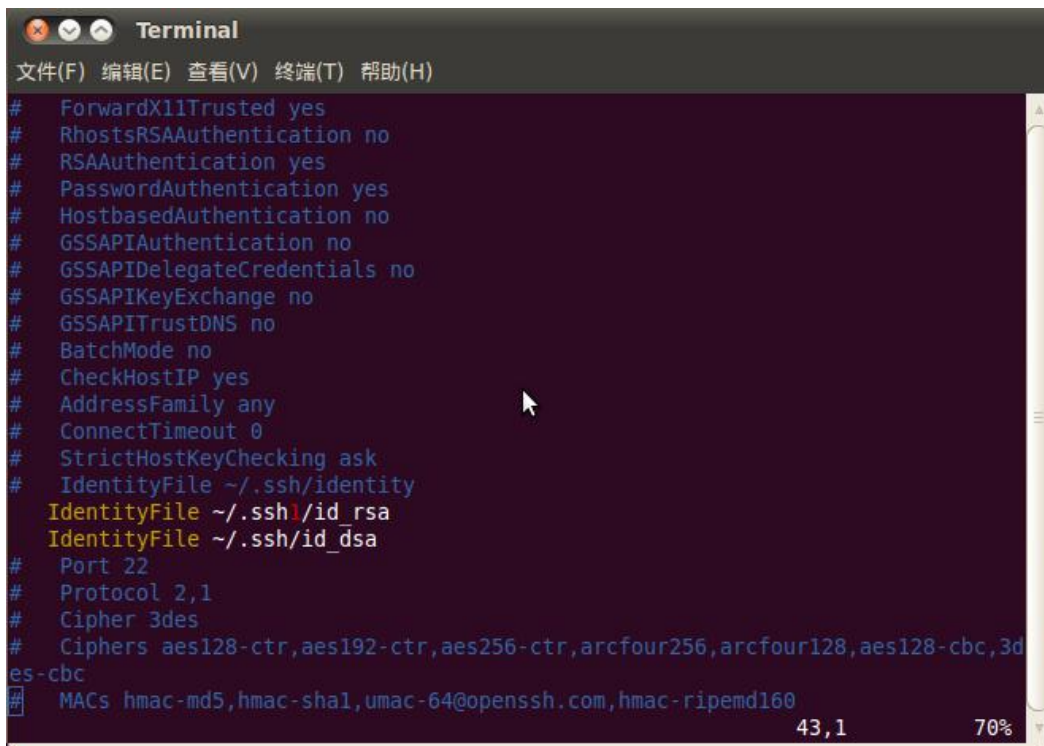


通过如下命令，配置当前用户的 ssh 配置。

```
~$ cp /etc/ssh/ssh_config ~/.ssh/config
```

```
~$ vi ~/.ssh/config
```

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，可以切换不同的的密钥。



```
Terminal
文件(F) 编辑(E) 查看(V) 终端(T) 帮助(H)
# ForwardX11Trusted yes
# RhostsRSAAuthentication no
# RSAAuthentication yes
# PasswordAuthentication yes
# HostbasedAuthentication no
# GSSAPIAuthentication no
# GSSAPIDelegateCredentials no
# GSSAPIKeyExchange no
# GSSAPITrustDNS no
# BatchMode no
# CheckHostIP yes
# AddressFamily any
# ConnectTimeout 0
# StrictHostKeyChecking ask
# IdentityFile ~/.ssh/identity
IdentityFile ~/.ssh1/id_rsa
IdentityFile ~/.ssh1/id_dsa
# Port 22
# Protocol 2,1
# Cipher 3des
# Ciphers aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,aes128-cbc,3des-cbc
# MACs hmac-md5,hmac-sha1,umac-64@openssh.com,hmac-ripemd160
43,1 70%
```

附录 B-5 密钥权限管理

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。

请妥善保管私钥文件。并不要二次授权与第三方使用。

附录 B-6 Git 权限申请说明

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