

密级状态：绝密() 秘密() 内部资料() 公开(☒)

RK3399PRO_LINUX_SDK_V0.01_20181208

(技术部, 第三系统产品部)

文件状态： [<input checked="" type="checkbox"/>] 草稿 [] 正式发布 [] 正在修改	当前版本：	V0.01
	作 者：	Caesar Wang
	完成日期：	2018-12-08
	审 核：	
	完成日期：	2018-12-08

福州瑞芯微电子股份有限公司

Fuzhou Rockchip Electronics Co., Ltd

(版本所有, 翻版必究)

文档修改记录

日期	修订版本	修订内容	修改人	核定人
2018-12-08	V0.01	初始版本	Caesar Wang	

目录

1 概述.....	5
2 主要支持功能.....	5
3 SDK 获取说明.....	5
4 软件开发指南.....	7
4.1 开发指南.....	7
5 SDK 编译说明.....	8
5.1 Uboot 编译.....	8
5.2 Kernel 编译步骤.....	9
5.3 Recovery 编译步骤.....	9
5.4 Buildroot rootfs 及 app 编译.....	9
5.5 Debian rootfs 编译.....	10
5.5.1 Building base debian system by ubuntu-build-service from linaro.....	10
5.5.2 Building rk-debian rootfs.....	10
5.5.3 Creating the ext4 image(linaro-rootfs.img).....	11
5.6 全自动编译.....	11
5.7 固件的打包.....	12
6 刷机说明.....	13
6.1 Windows 刷机说明.....	13
6.2 Linux 刷机说明.....	14
6.3 系统分区说明.....	15
7 Secure CRT 的参数设置.....	16
8 RK3399pro_Linux 工程目录介绍.....	17
9 RK3399Pro SDK 固件.....	18
10 SSH 公钥操作说明.....	19
10.1 SSH 公钥生成.....	19
10.2 使用 key-chain 管理密钥.....	19
10.3 多台机器使用相同 SSH 公钥.....	20
10.4 一台机器切换不同 SSH 公钥.....	21
10.5 密钥权限管理.....	22
10.6 Git 权限申请说明.....	22

免责声明

本文档按“现状”提供，福州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，由其各自拥有者所有。

版权所有 © 2018 福州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。

福州瑞芯微电子股份有限公司

Fuzhou Rockchip 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

客户服务电话：+86-591-83991906

客户服务传真：+86-591-83951833

客户服务邮箱：service@rock-chips.com

1 概述

本 SDK 是基于 Linux 的 Buildroot 和 Debian 9 系统，内核基于 kernel 4.4，适用于 RK3399pro evb 以及基于其上所有 linux 产品开发。

本 SDK 支持 NPU tensorflow/Caff 模型、VPU 硬解码、GPU 3D、Wayland 显示、QT 等功能。具体功能调试和接口说明，请阅读工程目录 docs/下文档。

2 主要支持功能

功能	模块名
数据通信	Wi-Fi、以太网卡、USB、SDCARD
应用程序	图库、设置、视频、音频、视频播放

3 SDK 获取说明

SDK 通过瑞芯微代码服务器对外发布上获取。其编译开发环境，参考[第 5 节 SDK 编译说明](#)。

获取 RK3399PRO Linux 软件包，需要有一个帐户访问 Rockchip 提供的源代码仓库。客户向瑞芯微技术窗口申请 SDK，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[第 10 节 SSH 公钥操作说明](#)。

RK3399PRO_LINUX_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/linux/rk/platform/manifests -b linux -  
m rk3399pro_linux_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 rk3399pro_linux_sdk_v0.01_20181208.tgz 为例，拷贝到该初始化包后，通过如下命令可检出源码：

```
mkdir rk3399pro
tar xvf rk3399pro_linux_sdk_v0.01_20181208.tgz -C rk3399pro
cd rk3399pro
.repo/repo/repo sync -l
.repo/repo/repo sync
```

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”命令同步更新。

4 软件开发指南

4.1 开发指南

RK3399Pro Linux SDK Kernel 版本是 Linux4.4, Rootfs 分别是 buidroot(2018.02-rc3)和 debian9, 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《Rockchip Linux 软件开发指南》.

可在 docs/目录下获取, 并会不断完善更新.

5 SDK 编译说明

Ubuntu 16.04 系统:

编译 **Buildroot** 环境搭建所依赖的软件包安装命令如下:

```
sudo apt-get install repo git-core gitk git-gui gcc-arm-linux-gnueabi-hf u-boot-tools device-tree-compiler gcc-aarch64-linux-gnu mtools parted libudev-dev libusb-1.0-0-dev python-linaro-image-tools linaro-image-tools autoconf autotools-dev libsigsegv2 m4 intltool libdrm-dev curl sed make binutils build-essential gcc g++ bash patch gzip bzip2 perl tar cpio python unzip rsync file bc wget libncurses5 libqt4-dev libglib2.0-dev libgtk2.0-dev libglade2-dev cvs git mercurial rsync openssh-client subversion asciidoc w3m dlatex graphviz python-matplotlib libc6:i386 libssl-dev texinfo liblz4-tool genext2fs
```

编译 **Debian** 环境搭建所依赖的软件包安装命令如下:

```
sudo apt-get install repo git-core gitk git-gui gcc-arm-linux-gnueabi-hf u-boot-tools device-tree-compiler gcc-aarch64-linux-gnu mtools parted libudev-dev libusb-1.0-0-dev python-linaro-image-tools linaro-image-tools gcc-4.8-multilib-arm-linux-gnueabi-hf gcc-arm-linux-gnueabi-hf libssl-dev gcc-aarch64-linux-gnu g++ conf autotools-dev libsigsegv2 m4 intltool libdrm-dev curl sed make binutils build-essential gcc g++ bash patch gzip bzip2 perl tar cpio python unzip rsync file bc wget libncurses5 libqt4-dev libglib2.0-dev libgtk2.0-dev libglade2-dev cvs git mercurial rsync openssh-client subversion asciidoc w3m dlatex graphviz python-matplotlib libc6:i386 libssl-dev texinfo liblz4-tool genext2fs
```

Ubuntu 17.04 或更新的系统:

除了上面外还需如下依赖包:

```
sudo apt-get install lib32gcc-7-dev g++-7 libstdc++-7-dev
```

(不需要安装 gcc-4.8-multilib-arm-linux-gnueabi-hf)

5.1 Uboot 编译

进入工程 u-boot 目录下执行 make.sh 来获取 rk3399pro_loader_v1.15.115.bin trust.img uboot.img:

```
rk3399pro evb 开发板: ./make.sh evb-rk3399
```

编译后生成文件在 u-boot 目录下:

```
u-boot/
├── rk3399pro_loader_v1.15.115.bin
└── trust.img
```



```
└─ uboot.img
```

5.2 Kernel 编译步骤

进入工程目录根目录执行以下命令自动完成 kernel 的编译及打包：

rk3399pro evb 开发板：

```
cd kernel
```

```
make ARCH=arm64 rockchip_linux_defconfig
```

```
make ARCH=arm64 rk3399pro-evb-v10-linux.img -j12
```

编译后在 kernel 目录生成 boot.img,这个 boot.img 就是包含 kernel 的 Image 和 DTB.

5.3 Recovery 编译步骤

进入工程目录根目录执行以下命令自动完成 Recovery 的编译及打包：

rk3399pro evb 开发板：

```
./build.sh recovery
```

编译后在 Buildroot 目录/output/rockchip_rk3399_recovery/images 生成 recovery.img,

5.4 Buildroot rootfs 及 app 编译

进入工程目录根目录执行以下命令自动完成 Rootfs 的编译及打包：

RK3399 Evb 开发板：

```
./build.sh rootfs
```

编译后在 Buildroot 目录/output/images 下生成 rootfs.ext4.

备注：

若需要编译单个模块或者第三方应用，需对交叉编译环境进行配置。

交叉编译工具位于 buildroot/output/rockchip_rk3399/host/usr 目录下，需要将工具的 bin/目录和

aarch64-buildroot-linux-gnu/bin/目录设为环境变量，在顶层目录执行自动配置环境变量的脚本（只对当前控制台有效）：

```
source envsetup.sh
```

输入命令查看：

```
aarch64-linux-gcc --version
```

此时会打印出以下 log 即标志为配置成功：

```
aarch64-linux-gcc.br_real (Buildroot 2018.02-rc3-00218-gddd64f1) 6.4.0
```

5.5 Debian rootfs 编译

先进入 rootfs/目录

```
cd rootfs/
```

下面的编译和 debian 固件生成请参考当前目录的 readme.md

5.5.1 Building base debian system by ubuntu-build-service from linaro

```
sudo apt-get install binfmt-support qemu-user-static live-build
```

```
sudo dpkg -i ubuntu-build-service/packages/*
```

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

编译 32 位的 debian:

```
RELEASE=stretch TARGET=desktop ARCH=armhf ./mk-base-debian.sh
```

或编译 64 位的 debian:

```
RELEASE=stretch TARGET=desktop ARCH=arm64 ./mk-base-debian.sh
```

编译完成会在 rootfs/生成: linaro-stretch-alip-xxxxx-1.tar.gz (xxxxx 表示生成时间戳)。

FAQ:

上面编译如果遇到如下问题情况:

noexec or nodev issue /usr/share/debootstrap/functions: line 1450:/rootfs/ubuntu-build-service/stretch-desktop-armhf/chroot/test-dev-null: Permission denied E: Cannot install into target '/home/foxluo/work3/rockchip/rk_linux/rk3399_linux/rootfs/ubuntu-build-service/stretch-desktop-armhf/chroot' mounted with noexec or nodev

解决办法: mount -o remount,exec,dev xxx (xxx is the mount place), then rebuild it.

另外如果还有遇到其他编译异常, 先排除使用的编译系统是 ext2/ext4 的系统类型.

5.5.2 Building rk-debian rootfs

编译 32 位的 debian:

```
VERSION=debug ARCH=armhf ./mk-rootfs-stretch.sh
```

(开发阶段推荐使用后面带 debug)。

编译 64 位的 debian:

```
VERSION=debug ARCH=arm64 ./mk-rootfs-stretch-arm64.sh
```

(开发阶段推荐使用后面带 debug)。

5.5.3 Creating the ext4 image(linaro-rootfs.img)

```
./mk-image.sh
```

此时会生成 linaro-rootfs.img.

5.6 全自动编译

上面 Kernel/Uboot/Recovery/Rootfs 各个部分的编译，进入工程目录根目录执行以下命令自动完成所有的编译：`./build.sh`

具体参数使用情况，可 `help` 查询，比如如下：

```
Rk3399$ ./build.sh --help
Can't found build config, please check again

====USAGE: build.sh modules====
uboot      -build uboot
kernel     -build kernel
rootfs     -build default rootfs, currently build buildroot as default
buildroot  -build buildroot rootfs
yocto      -build yocto rootfs, currently build ros as default
ros        -build ros rootfs
debian     -build debian rootfs
pcba       -build pcba
recovery   -build recovery
all        -build uboot, kernel, rootfs, recovery image
....
default    -build all modules
```

每个板子的板级配置需要在 `/device/rockchip/rockchip/Boardconfig.mk` 进行相关配置。
比如:rk3399pro evb 主要配置如下：

```
# Target arch
export RK_ARCH=arm64
# Uboot defconfig
export RK_UBOOT_DEFCONFIG=evb-rk3399
# Kernel defconfig
export RK_KERNEL_DEFCONFIG=rockchip_linux_defconfig
# Kernel dts
export RK_KERNEL_DTS=rk3399pro-evb-v10-linux
# boot image type
export RK_BOOT_IMG=boot.img
# kernel image path
export RK_KERNEL_IMG=kernel/arch/arm64/boot/Image
# parameter for GPT table
export RK_PARAMETER=parameter-buildroot.txt
# Buildroot config
export RK_CFG_BUILDROOT=rockchip_rk3399
# Recovery config
export RK_CFG_RECOVERY=rockchip_rk3399_recovery
# ramboot config
export RK_CFG_RAMBOOT=
# Pcba config
export RK_CFG_PCBA=rockchip_rk3399_pcba
# Build image
```

5.7 固件的打包

上面 Kernel/Uboot/Recovery/Rootfs 各个部分的编译后，进入工程目录根目录执行以下命令自动完成所有固件打包到 rockdev 目录下：

Buildroot 固件生成: `./mkfirmware.sh`

Debian 固件生成:

`./build.sh BoardConfig_debian.mk`

`./mkfirmware` 可以生成 debian 的固件.

6 刷机说明

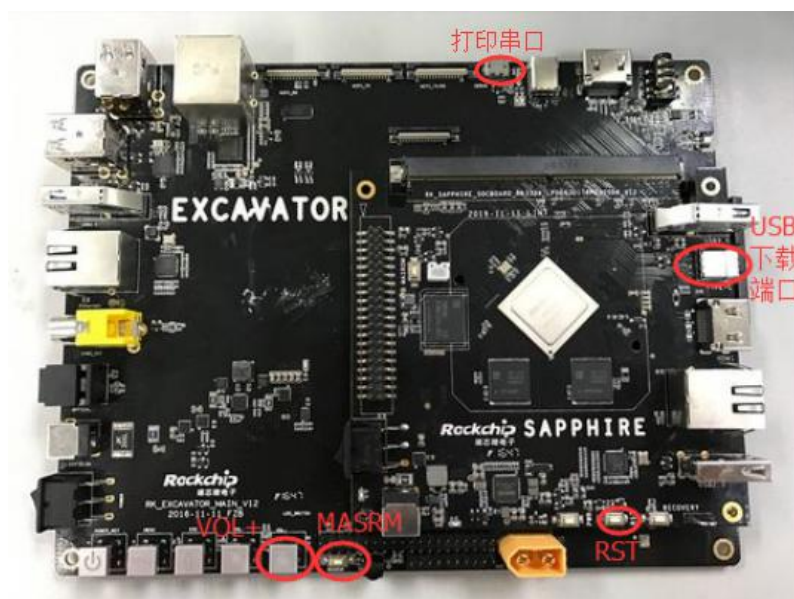


图 1 RK3399 挖掘机

6.1 Windows 刷机说明

SDK 提供 Windows 烧写工具(工具版本需要 V2.55 或以上), 工具位于工程根目录:

tools/

└─ windows/AndroidTool

如下图, 编译生成相应的固件后, 设备烧写需要进入 MASKROM 烧写模式, 连接好 usb 下载线后, 按住按键“MSROM”不放并按下复位键“RST”后松手, 就能进入 MASKROM 模式, 加载编译生成固件的相应路径后, 点击“执行”进行烧写, 也可以按“recovery”按键不放并按下复位键“RST”后松手进入 loader 模式进行烧写, 下面是 MASKROM 模式的 分区偏移及烧写文件。(Note: Window PC 需要在管理员权限运行工具才可执行)

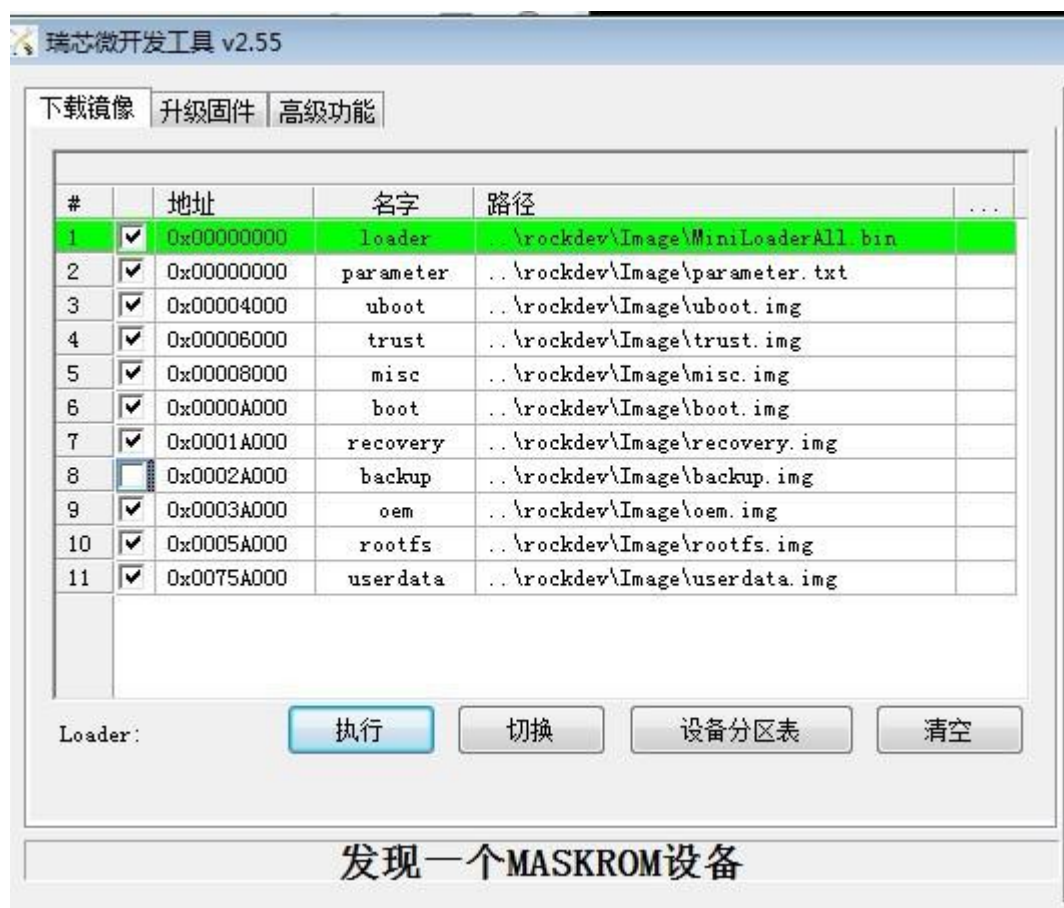


图 2 烧写工具 AndroidTool.exe

注：烧写前，需安装最新 USB 驱动，驱动详见：

tools/windows/DriverAssitant_v4.6.zip

6.2 Linux 刷机说明

Linux 下的烧写工具位于 tools/linux 目录下(Linux_Upgrade_Tool 工具版本需要 V1.33 或以上)，请确认你的板子连接到 maskrom/loader rockusb. 比如编译生成的固件在 rockdev 目录下，升级命令如下：

```
sudo ./upgrade_tool ul rockdev/MiniLoaderAll.bin
```

```
sudo ./upgrade_tool di -p rockdev/parameter.txt
```

```
sudo ./upgrade_tool di -u rockdev/uboot.img
```

```
sudo ./upgrade_tool di -t rockdev/trust.img
```

```
sudo ./upgrade_tool di -misc rockdev/misc.img
```

```
sudo ./upgrade_tool di -b rockdev/boot.img
```

```
sudo ./upgrade_tool di -recovery rockdev/recovery.img
```

```
sudo ./upgrade_tool di -oem rockdev/oem.img
```

```
sudo ./upgrade_tool di -rootfs rocdev/rootfs.img
```

```
sudo ./upgrade_tool di -userdata rockdev/userdata.img
```

```
sudo ./upgrade_tool rd
```

或在根目录，机器在 **maskrom** 状态运行如下升级：

```
./rkflash.sh
```

6.3 系统分区说明

默认分区说明 (下面是挖掘机分区参考):

Number	Start (sector)	End (sector)	Size	Code	Name
1	16384	24575	4096K	0700	uboot
2	24576	32767	4096K	0700	trust
3	32768	40959	4096K	0700	misc
4	40960	106495	32.0M	0700	boot
5	106496	172031	32.0M	0700	recovery
6	172032	237567	32.0M	0700	backup
7	237568	368639	64.0M	0700	oem
8	368640	3514367	3584M	0700	rootfs
9	7708672	15269854	3691M	0700	userdata

uboot 分区: 烧写 uboot 编译出来的 uboot.img.

trust 分区: 烧写 uboot 编译出来的 trust.img.

misc 分区: 烧写 misc.img。给 recovery 使用.

boot 分区: 烧写 kernel 编译出来的 boot.img.

recovery 分区: 烧写 recovery.img.

backup 分区: 预留，暂时没有用。后续跟 android 一样作为 recovery 的 backup 使用

oem 分区: 给厂家使用，存放厂家的 app 或数据。只读。代替原来音箱的 data 分区。挂载在 /oem 目录.

rootfs 分区: 存放 buildroot 或者 debian 编出来的 rootfs.img,只读.

userdata 分区:存放 app 临时生成的文件或者是给最终用户使用。可读写，挂载在 /userdata 目录下.

7 Secure CRT 的参数设置

利用 Secure CRT 软件打印调试信息 log，需要对串口参数进行设置，具体设置细节如下图：

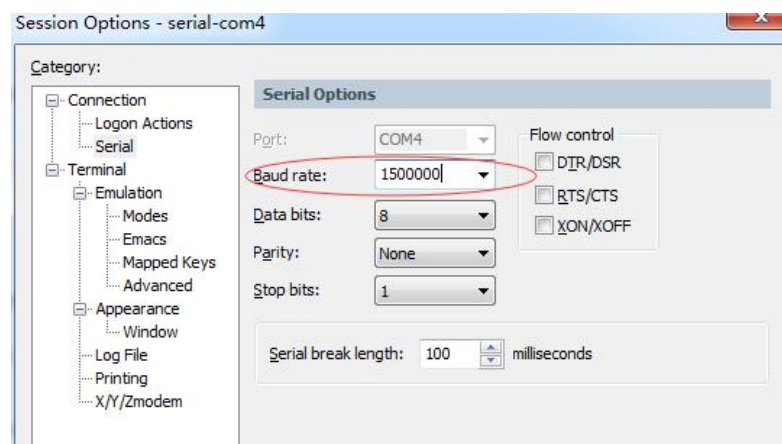


图 3 Secure CRT 参数设置

8 RK3399pro_Linux 工程目录介绍

进工程目录下有 buildroot、buildroot、recovery、app、kernel、u-boot、device、docs、external 等目录。每个目录或其子目录会对应一个 git 工程，提交需要在各自的目录下进行。

- 1) app: 存放上层应用 app，主要是 Carmachin 和一些测试应用程序。
- 2) buildroot: 定制根文件系统。
- 3) device/rockchip: 存放一些编译和打包固件的脚本和预备文件。
- 4) docs: 存放工程帮助文件。
- 5) external: 相关库，包括音频、视频、网络等。
- 6) kernel: kernel 代码。
- 7) prebuilts: 存放交叉编译工具链。
- 8) recovery: 存放 recovery 工程文件。
- 9) rkbin: 存放固件和工具。
- 10) rockdev: 存放编译输出固件
- 11) rootfs: Debian 根文件系统
- 12) tools: 存放一些常用工具。
- 13) u-boot: uboot 代码。
- 14) npu: 存放 npu 代码。

9 RK3399Pro SDK 固件

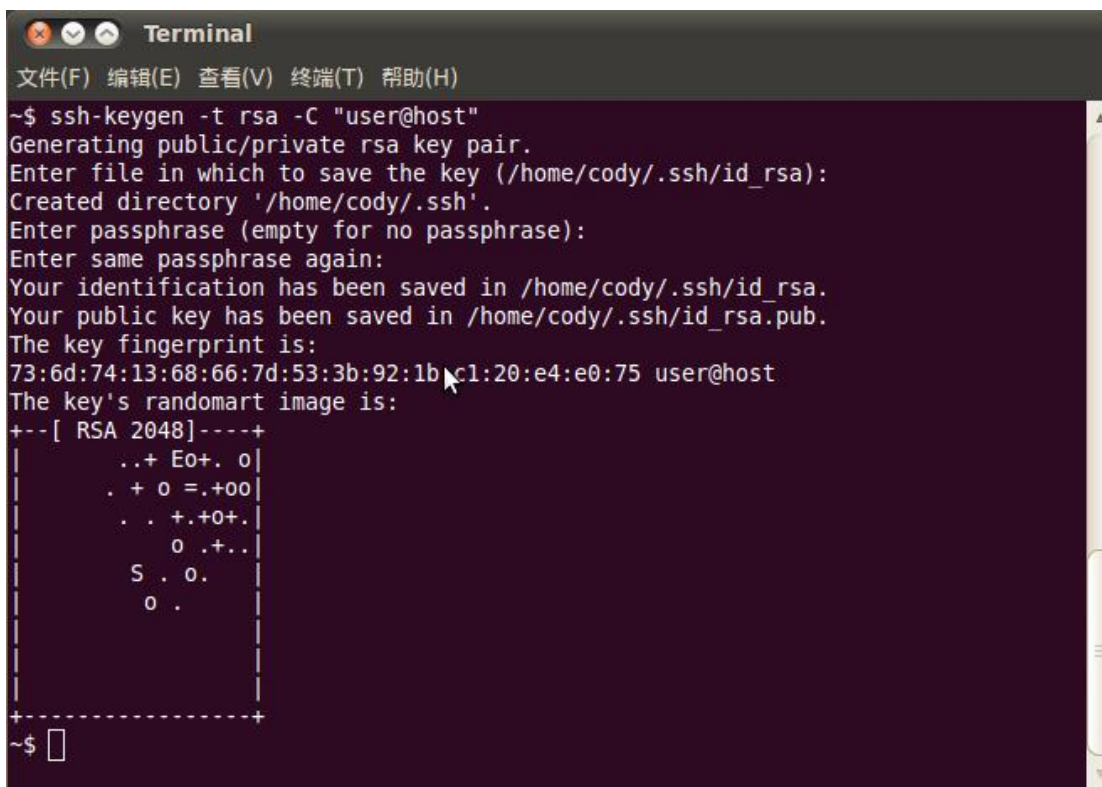
10 SSH 公钥操作说明

10.1 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+. 0      |
|      . + 0 =.+00     |
|      . . +.+0+.     |
|      0 .+.         |
|      S . 0.         |
|      0 .            |
|                      |
+-----+
~$
```

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。

10.2 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。

10.3 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
```

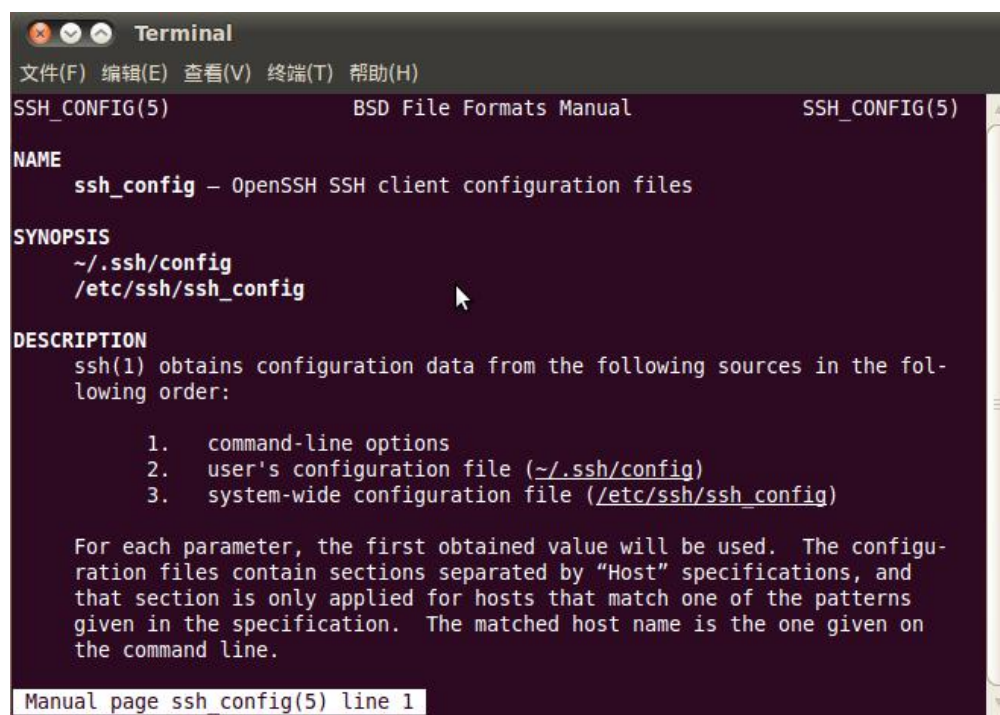
在 console 输入如下命令即可解决。

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

10.4 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
```

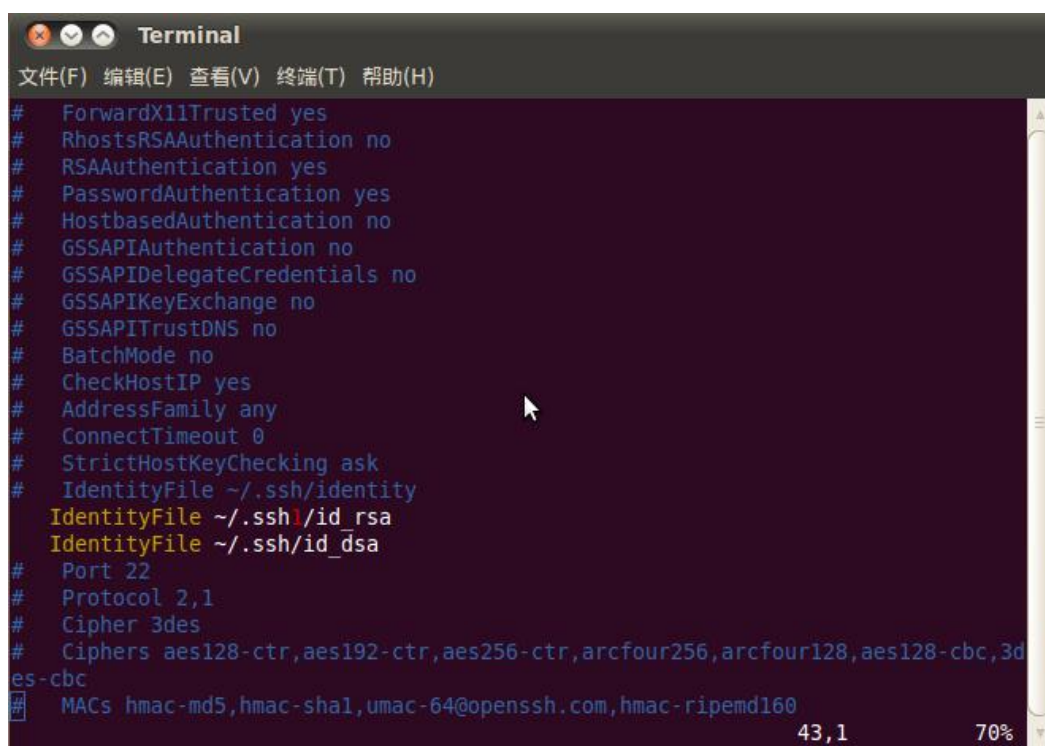


通过如下命令，配置当前用户的 SSH 配置。

```
~$ 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。

A terminal window titled "Terminal" with a menu bar containing "文件(F)", "编辑(E)", "查看(V)", "终端(T)", and "帮助(H)". The terminal displays a list of SSH configuration options, each preceded by a hash symbol (#). The options are: 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 ~/.ssh/id_rsa, IdentityFile ~/.ssh/id_dsa, Port 22, Protocol 2,1, Cipher 3des, Ciphers aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,aes128-cbc,3des-cbc, and MACs hmac-md5,hmac-sha1,umac-64@openssh.com,hmac-ripemd160. The terminal has a dark background and a scrollbar on the right side. The status bar at the bottom right shows "43,1" and "70%".

```
# 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 ~/.ssh/id_rsa
IdentityFile ~/.ssh/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
```

10.5 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。

10.6 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。