

密级状态： 绝密(        )    秘密(        )    内部资料(        )    公开(    ✓    )

# RK3399Pro Linux SDK 发布说明

(技术部，第三系统产品部)

<b>文件状态：</b> [    ] 草稿 [    ] 正在修改 [ ✓ ] 正式发布	<b>文件标识：</b>	RK-FB-CS-009
	<b>当前版本：</b>	1.1.3
	<b>作     者：</b>	Caesar Wang
	<b>完成日期：</b>	2019-10-23
	<b>审     核：</b>	Eddie Cai
	<b>审核日期：</b>	2019-10-23



## 文档修改记录

日期	修订版本	修订内容	修改人	核定人
2019-02-17	Beta_V0.01	初始 Beta 版本	Caesar Wang	Eddie Cai
2019-03-21	Beta_V0.02	1.修改 5.1.3 中 ./mkfirmware.sh 生成 image 的方法. 2.更改 8 章节中 rknn_demo 用例添加 debian 的说明 3.更改 8 章节中 SDK 固件升级到 v0.02	Caesar Wang	Eddie Cai
2019-06-06	V1.0.0	1. 正式发布版本 2. 添加 NPU 相关说明 3. 增加 Yocto 的编译说明 4. 增加 github 下载说明	Caesar Wang	CCH
2019-06-21	V1.0.1	5. 修改软件开发指南名字	Caesar Wang	CCH
2019-10-14	V1.1.2	6. 修改 Debian 编译说明	Caesar Wang	Eddie Cai
2019-10-23	V1.1.3	7. 支持 rk3399pro evb v13 编译	Caesar Wang	Eddie Cai

# 目 录

<b>1 概述 .....</b>	<b>4</b>
<b>2 主要支持功能 .....</b>	<b>4</b>
<b>3 SDK 获取说明 .....</b>	<b>4</b>
<b>4 软件开发指南 .....</b>	<b>5</b>
4.1 开发指南.....	5
4.2 NPU 开发工具.....	5
4.3 软件更新记录.....	6
<b>5 硬件开发指南 .....</b>	<b>6</b>
<b>6 RK3399Pro_Linux 工程目录介绍 .....</b>	<b>6</b>
<b>7 SDK 编译说明 .....</b>	<b>7</b>
7.1 NPU 编译说明.....	7
7.1.1 Uboot 编译.....	7
7.1.2 Kernel 编译步骤.....	8
7.1.3 Boot.img 以及 npu 固件生成步骤.....	8
7.1.4 全自动编译.....	8
7.2 RK3399pro 编译说明.....	9
7.2.1 Uboot 编译.....	9
7.2.2 Kernel 编译步骤.....	9
7.2.3 Recovery 编译步骤.....	9
7.2.4 Buildroot rootfs 及 app 编译.....	9
7.2.5 Debian rootfs 编译.....	10
7.2.6 Yocto rootfs 编译.....	10
7.2.7 全自动编译.....	11
7.2.8 固件的打包.....	12
<b>8 刷机说明 .....</b>	<b>13</b>
8.1 Windows 刷机说明.....	13
8.2 Linux 刷机说明 .....	14
8.3 系统分区说明.....	15
<b>9 RK3399Pro SDK 固件及简单 Demo 测试.....</b>	<b>15</b>
9.1 RK3399Pro SDK 固件.....	15
9.2 RKNN_DEMO 测试.....	15
<b>10 SSH 公钥操作说明 .....</b>	<b>17</b>
10.1 SSH 公钥生成 .....	17
10.2 使用 key-chain 管理密钥 .....	17
10.3 多台机器使用相同 SSH 公钥.....	18
10.4 一台机器切换不同 SSH 公钥.....	18
10.5 密钥权限管理.....	19
10.6 参考文档.....	20

## 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 © 2019 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](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)

## 1 概述

本 SDK 是基于 Linux 的 Buildroot、Yocto 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/Caffe 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或者从 Github 开源网站上获取。其编译开发环境，参考[第 7 节 SDK 编译说明](#)。

### 获取 SDK 方法一：从瑞芯微代码服务器获取源码

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\_release\_v1.0.0\_20190606.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_release_v1.0.0_20190606.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。

### 获取 SDK 方法二：从 Github 开源网站获取源码

下载 repo 工具

```
git clone https://github.com/rockchip-linux/repo.git
```

建立 rk3399pro linux 工作目录

```

mkdir rk3399pro_linux
进入 rk3399pro linux 工作目录
cd rk3399pro_linux/
初始化 repo 仓库
../repo/repo init --repo-url=https://github.com/rockchip-linux/repo
-u
https://github.com/rockchip-linux/manifests -b master -m
rk3399pro_linux_release.xml
同步下载整个工程:
../repo/repo sync

```

## 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 Buidlroot(2018.02-rc3)、Yocto(thud 2.6) 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。

### 4.2 NPU 开发工具

本 SDK NPU 开发工具如下：

#### **RKNN\_DEMO(MobileNet SSD):**

RKNN 的 Demo 请参考目录 external/rknn\_demo/， 相关操作说明详见工程目录 docs/SoC platform related/RK3399PRO/ Rockchip\_Developer\_Guide\_Linux\_RKNN\_DEMO\_CN。

#### **RKNN-TOOLKIT:**

开发工具在 external/rknn-toolkit 目录下，具体使用说明请参考文档 docs/Develop reference documents/NPU/

- RK3399Pro\_Linux&Android\_RKNN\_API\_V0.9.6\_20190514.pdf
- RKNN-Toolkit FAQ.pdf
- RKNN-Toolkit 快速上手指南\_V1.0.0.pdf
- RKNN-Toolkit 使用指南\_V1.0.0.pdf
- 基于 RKNN 的深度神经网络模型设计建议.pdf

#### **RKNN-DRIVER:**

RKNN DRIVER 开发内容在工程目录 external/rknpu 下。

#### **RKNPUTools:**

RKNN API 的开发使用在工程目录 external/NRKNPUTool 下。

#### **NPU 软件启动说明:**

RK3399PRO 的 NPU 软件启动说明，请参考工程目录 docs/SoC platform related/RK3399PRO/ Rockchip\_RK3399Pro\_Instruction\_Linux\_NPU\_CN.pdf。

## 4.3 软件更新记录

软件发布版本升级通过工程 xml 进行查看，具体方法如下：

```
.repo/manifests$ ls -l -h rk3399pro_linux_release.xml
```

软件发布版本升级更新内容通过工程文本可以查看，具体方法如下：

```
.repo/manifests$ cat rk3399pro_linux_v0.01/RK3399PRO_Release_Note.txt
```

或者参考工程目录

docs/SoC platform related/RK3399PRO/RK3399PRO\_Linux\_SDK\_Release\_Note.pdf

## 5 硬件开发指南

硬件相关开发可以参考用户使用指南，在工程目录 docs/SoC platform related/RK3399PRO/Rockchip\_RK3399Pro\_User\_Guide\_Hardware\_xx.pdf

## 6 RK3399Pro\_Linux 工程目录介绍

进工程目录下有 buildroot、debian、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 Camera/Video/Music 等一些应用程序。
- 2) buildroot: 定制 buildroot 根文件系统。
- 3) debian: Debian 根文件系统。
- 4) device/rockchip: 存放一些编译和打包固件的脚本和预备文件。
- 5) docs: 存放工程帮助文件。
- 6) external: 相关库，包括音频、视频、网络等。
- 7) kernel: kernel 代码。
- 8) npu: 存放 npu 代码。
- 9) prebuilts: 存放交叉编译工具链。
- 10) recovery: 存放 recovery 工程文件。
- 11) rkbin: 存放固件和工具。
- 12) rockdev: 存放编译输出固件。
- 13) tools: 存放一些常用工具。
- 14) u-boot: uboot 代码。
- 15) yocto: yocto 根文件系统。

## 7 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
bihf u-boot-tools device-tree-compiler gcc-aarch64-linux-gnu mtools
parted libudev-dev libusb-1.0-0-dev python-linaro-image-tools lina
ro-image-tools autoconf autotools-dev libsigsegv2 m4 intltool libdr
m-dev curl sed make binutils build-essential gcc g++ bash patch gzi
p bzip2 perl tar cpio python unzip rsync file bc wget libncurses5 l
ibqt4-dev libglib2.0-dev libgtk2.0-dev libglade2-dev cvs git mercur
ial rsync openssh-client subversion asciidoc w3m dlatex graphviz p
ython-matplotlib libc6:i386 libssl-dev texinfo liblz4-tool genext2f
s
```

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
bihf u-boot-tools device-tree-compiler gcc-aarch64-linux-gnu mtools
parted libudev-dev libusb-1.0-0-dev python-linaro-image-tools lina
ro-image-tools gcc-4.8-multilib-arm-linux-gnueabi gcc-arm-linux-g
nueabi libssl-dev gcc-aarch64-linux-gnu g++ conf autotools-dev lib
sigsegv2 m4 intltool libdrm-dev curl sed make binutils build-essent
ial gcc g++ bash patch gzip bzip2 perl tar cpio python unzip rsync
file bc wget libncurses5 libqt4-dev libglib2.0-dev libgtk2.0-dev li
bglade2-dev cvs git mercurial rsync openssh-client subversion asciid
oc w3m dlatex graphviz python-matplotlib libc6:i386 libssl-dev te
xinfo liblz4-tool genext2fs
```

### Ubuntu 17.04 或更高版本系统:

除了上述外还需如下依赖包:

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

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

**注意:** RK3399Pro 每次上电启动起来会加载 NPU 固件。默认 NPU 固件都是预编好放到 rootfs 的 /usr/share/npu\_fw 目录下, NPU 固件烧写以及启动方式请参考文档 docs/Soc Platform related/RK3399PRO/ Rockchip\_RK3399Pro\_Instruction\_Linux\_NPU\_CN。

下面分别对 NPU 和 RK3399pro 固件编译方法进行介绍:

## 7.1 NPU 编译说明

### 7.1.1 Uboot 编译

进入工程 npu/u-boot 目录下执行 make.sh 来获取 rknpn\_lion\_loader\_v1.03.103.bin trust.img uboot.img:

rk3399pro-npu:

```
./make.sh rknpn-lion
```

编译后生成文件在 u-boot 目录下:

```
u-boot/
├── rknpn_lion_loader_v1.03.103.bin
├── trust.img
└── uboot.img
```



### 7.1.2 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 v10/v11/v12 开发板：

```
cd kernel
make ARCH=arm64 rk3399pro_npu_defconfig
make ARCH=arm64 rk3399pro-npu-evb-v10.img -j12
```

rk3399pro evb v13 开发板：

```
cd kernel
make ARCH=arm64 rk3399pro_npu_pcie_defconfig
make ARCH=arm64 rk3399pro-npu-evb-v10-multi-cam.img -j12
```

### 7.1.3 Boot.img 以及 npu 固件生成步骤

进入工程 npu 目录执行以下命令自动完成 boot.img 的打包：

rk3399pro evb v10/v11/v12 开发板：

```
cd npu
./build.sh ramboot
./mkfirmware.sh rockchip_rk3399pro-npu
```

rk3399pro evb v13 开发板：

```
cd npu/device/rockchip
ln -sf rk3399pro-npu-multi-cam/BoardConfig.mk .BoardConfig.mk
cd / && cd npu
./build.sh ramboot
./mkfirmware.sh rockchip_rk3399pro-npu-multi-cam
```

### 7.1.4 全自动编译

上述 Kernel/Uboot/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：

rk3399pro evb v10/v11/v12 开发板：

```
cd npu/device/rockchip
ln -sf rk3399pro-npu/BoardConfig.mk .BoardConfig.mk
cd /
./build.sh uboot
./build.sh kernel
./build.sh ramboot
./mkfirmware.sh rockchip_rk3399pro-npu
```

rk3399pro evb v13 开发板：

```
cd npu/device/rockchip
ln -sf rk3399pro-npu-multi-cam/BoardConfig.mk .BoardConfig.mk
cd /
./build.sh uboot
./build.sh kernel
./build.sh ramboot rockchip_rk3399pro-npu-multi-cam
```

在 rockdev 目录下生成 boot.img, uboot.img, trust.img, MiniLoaderAll.bin

注意 rockdev 下生成 npu 固件需要放到 rootfs 指定位置/usr/share/npu\_fw，或手动放到 rootfs 中。

## 7.2 RK3399pro 编译说明

### 7.2.1 Uboot 编译

进入工程 u-boot 目录下执行 make.sh 来获取 rk3399pro\_loader\_v1.23.115.bin trust.img uboot.img:

rk3399pro evb 板子:

```
./make.sh rk3399pro
```

编译后生成文件在 u-boot 目录下:

```
u-boot/
├── rk3399pro_loader_v1.23.115.bin
├── trust.img
└── uboot.img
```

### 7.2.2 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 v10 开发板:

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

rk3399pro evb v11/v12 开发板:

```
cd kernel
make ARCH=arm64 rockchip_linux_defconfig
make ARCH=arm64 rk3399pro-evb-v11-linux.img -j12
```

rk3399pro evb v13 开发板:

```
cd kernel
make ARCH=arm64 rockchip_linux_defconfig
make ARCH=arm64 rk3399pro-evb-v13-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。

### 7.2.3 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\_rk3399pro\_recovery/images 生成 recovery.img。

### 7.2.4 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:

rk3399pro evb 开发板:

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

编译后在 Buildroot 目录 output/rockchip\_rk3399pro/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\_rk3399pro/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
```

## 7.2.5 Debian rootfs 编译

先进入 debian/目录

```
cd debian/ && ./build.sh debian
```

或者后续的编译和 debian 固件生成请参考当前目录 debian/readme.md。

### 7.2.5.1 Building base debian system

```
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 debian/生成: linaro-stretch-alip-xxxxx-1.tar.gz (xxxxx 表示生成时间戳)。

FAQ:

上述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/fo
xluo/work3/rockchip/rk_linux/rk3399_linux/rootfs/ubuntu-build-servi
ce/stretch-desktop-armhf/chroot' mounted with noexec or nodev
```

解决方法:

```
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。

### 7.2.5.2 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.sh
```

(开发阶段推荐使用后面带 debug)。

### 7.2.5.3 Creating the ext4 image(linaro-rootfs.img)

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

此时会生成 linaro-rootfs.img。

## 7.2.6 Yocto rootfs 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:

RK3399PRO EVB 开发板: ./build.sh yocto

编译后在 yocto 目录 build/lastest 下生成 rootfs.img。

#### FAQ:

上面编译如果遇到如下问题情况:

```
Please use a locale setting which supports UTF-8 (such as LANG=en_US.UTF-8).
```

```
Python can't change the filesystem locale after loading so we need a UTF-8
```

```
when Python starts or things won't work.
```

解决办法:

```
locale-gen en_US.UTF-8
```

```
export LANG=en_US.UTF-8 LANGUAGE=en_US.en LC_ALL=en_US.UTF-8
```

或者参考 <https://webkul.com/blog/setup-locale-python3>

编译后生成的 image 在 yocto/build/lastest/rootfs.img

默认用户名登录是 root

Yocto 更多信息请参考 [Rockchip Wiki](#)

## 7.2.7 全自动编译

上述 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 all
```

默认是 **buildroot**, 可以通过设置环境变量 **RK\_ROOTFS\_SYSTEM** 指定 **rootfs**。

比如需要 **Yocto** 可以通过以下命令进行生成:

```
$export RK_ROOTFS_SYSTEM=yocto
```

```
$/build.sh all
```

具体参数使用情况, 可 **help** 查询, 比如:

```
rk3399pro$ ./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/rk3399pro/Boardconfig.mk 进行相关配置。

rk3399pro evb 主要配置如下:

```
# Target arch
export RK_ARCH=arm64
# Uboot defconfig
export RK_UBOOT_DEFCONFIG=rk3399pro
# Kernel defconfig
export RK_KERNEL_DEFCONFIG=rockchip_linux_defconfig
# Kernel dts
export RK_KERNEL_DTS=rk3399pro-evb-v11-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_rk3399pro
# Recovery config
export RK_CFG_RECOVERY=rockchip_rk3399pro_recovery
# ramboot config
```

### 7.2.8 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：

**固件生成：**

```
./mkfirmware.sh
```

## 8 刷机说明

目前 rk3399pro evb 有 v10/v11/v12/v13 版本，绿色板子是 v10 版本，黑色板子是 v11/v12/v13 版本。板子功能位置是一样，下面以 rk3399pro evb v12 板子做介绍，如下图说明。

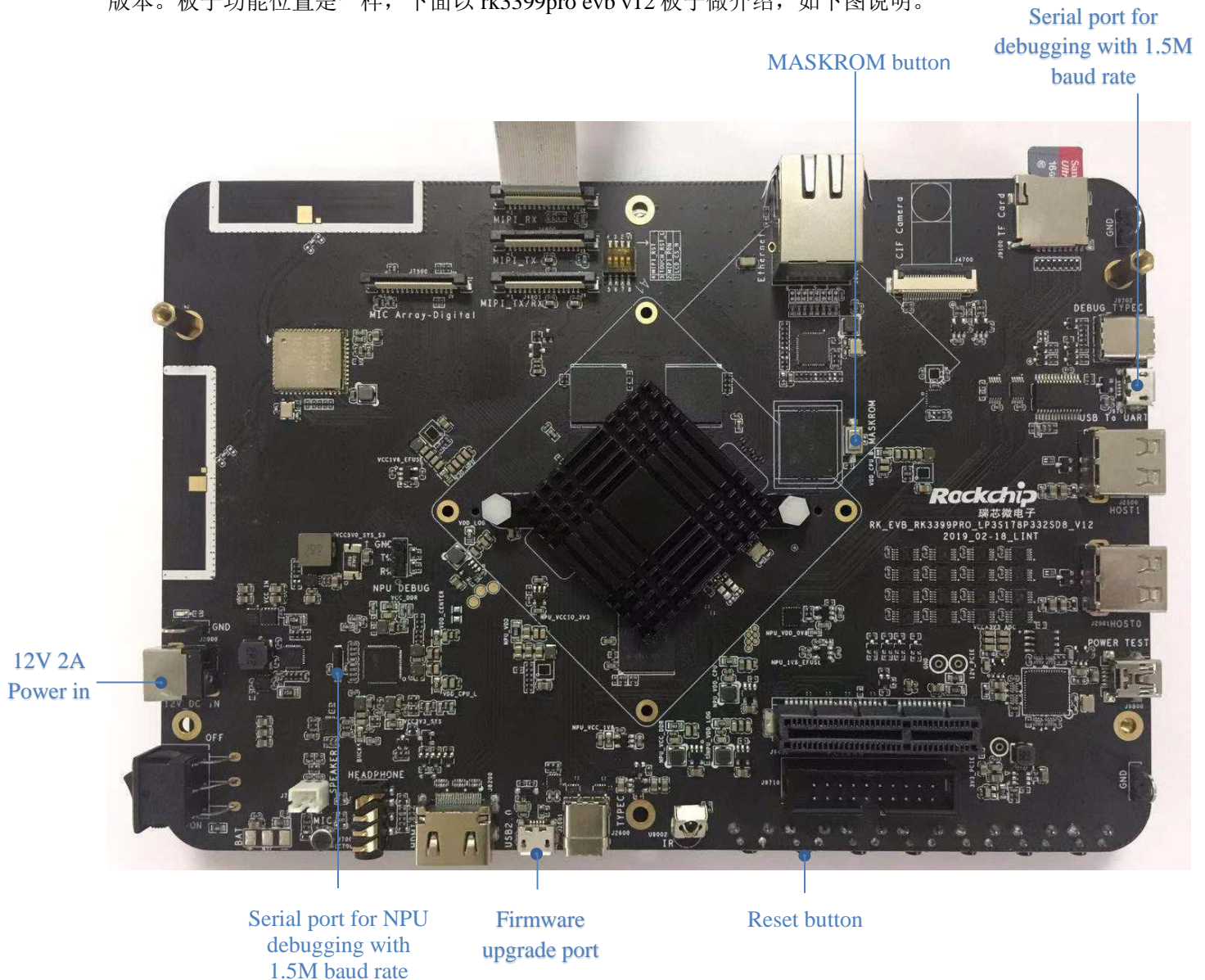


图 1 RK3399PRO EVB

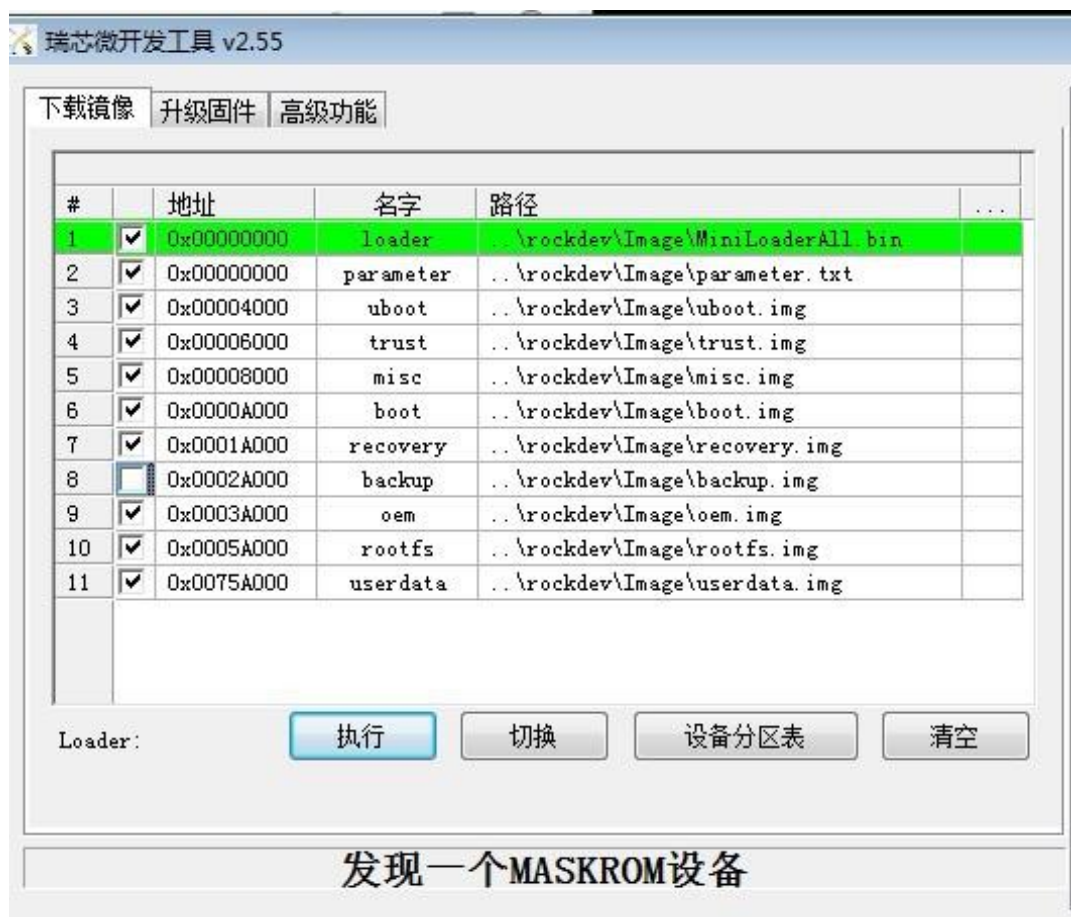
### 8.1 Windows 刷机说明

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(aka BootROM) 烧写模式，连接好 usb 下载线后，按住按键“MASROM”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。(注意：**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\_v4.8.zip

## 8.2 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 rockdev/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
```

## 8.3 系统分区说明

默认分区说明 (下面是 RK3399PRO EVB 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	303104	96.0M	0700	recovery
6	303104	368639	32.0M	0700	backup
7	368640	499711	64.0M	0700	oem
8	499712	25696863	1024M	0700	rootfs
9	2596864	30535646	13.3G	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, Yocto 或 Debian 编出来的 rootfs.img。

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。

## 9 RK3399Pro SDK 固件及简单 Demo 测试

### 9.1 RK3399Pro SDK 固件

RK3399PRO\_LINUX\_SDK\_V1.0.0\_20190606 固件下载链接如下  
(包含 Debian/Buildroot/Yocto 的固件)

V10(绿色)板子:

Debian: <https://eyun.baidu.com/s/3dGfYj5Z>

Buildroot: <https://eyun.baidu.com/s/3dbu9mI>

Yocto: <https://eyun.baidu.com/s/3ggdmnTL>

V11/v12(黑色)板子:

Debian: <https://eyun.baidu.com/s/3smEf8xn>

Buildroot: <https://eyun.baidu.com/s/3i6lVFW5>

Yocto: <https://eyun.baidu.com/s/3gheOyD9>

### 9.2 RKNN\_DEMO 测试

首先插入 usb camera,

然后在 Buildroot 系统中运行 **rknn\_demo** 或 Debian 系统中运行 **test\_rknn\_demo.sh**

具体参考工程文档 docs/Soc Platform Related/RK3399PRO/



## Rockchip\_Developer\_Guide\_Linux\_RKNN\_DEMO\_CN.pdf

。在 Buildroot 中运行结果如下：

```
[root@rk3399pro:/]# rknn demo
librga:RGA_GET_VERSION:3.02,3.020000
ctx=0x2e834c20,ctx->rgaFd=3
Rga built version:version:+2017-09-28 10:12:42
Success build
size = 12582988, g_bo.size = 13271040
size = 12582988, cur_bo->size = 13271040
size = 12582988, cur_bo->size = 13271040
...
get device /dev/video10
read model:/usr/share/rknn_demo/mobilenet_ssd.rknn, len:32002449
spec = local:transfer_proxy
D RKNNAPI: =====
D RKNNAPI: RKNN VERSION:
D RKNNAPI:   API: 0.9.6 (47a27f6 build: 2019-06-02 22:29:50)
D RKNNAPI:   DRV: 0.9.7 (d65a37f build: 2019-05-31 14:23:01)
D RKNNAPI: =====
Using UVC media node
```

最终在屏幕显示效果如下：



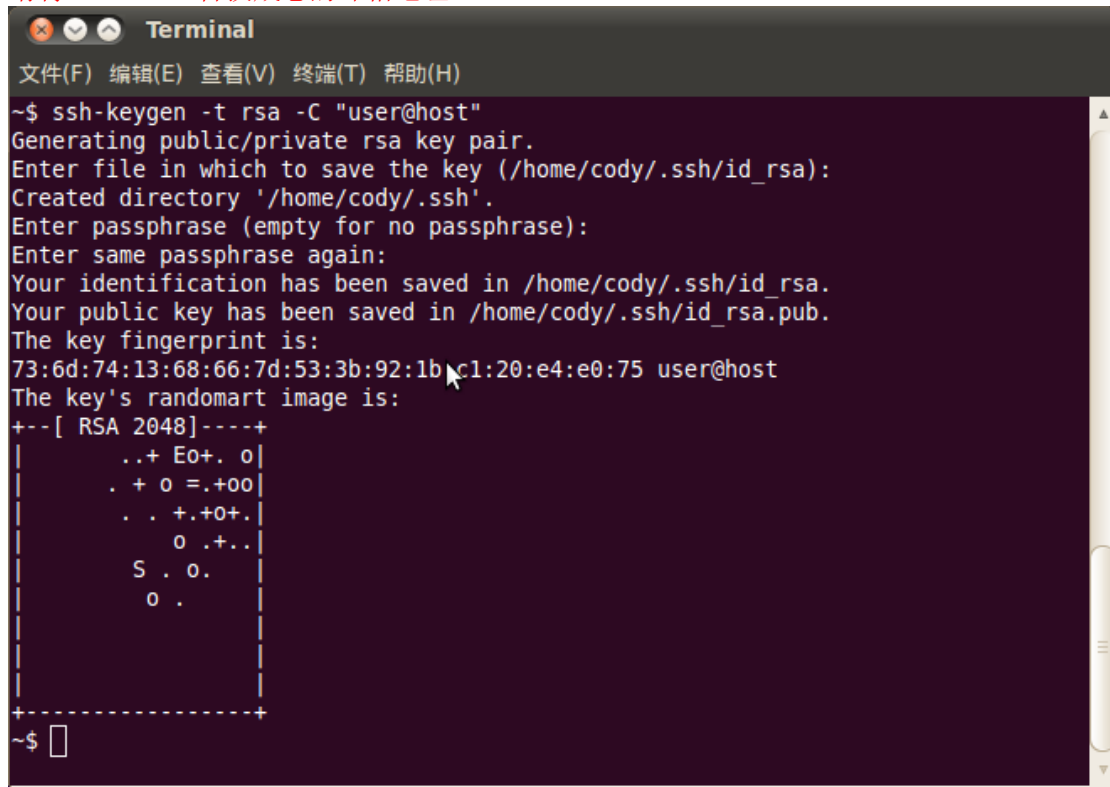
## 10 SSH 公钥操作说明

### 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。



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。

### 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
```

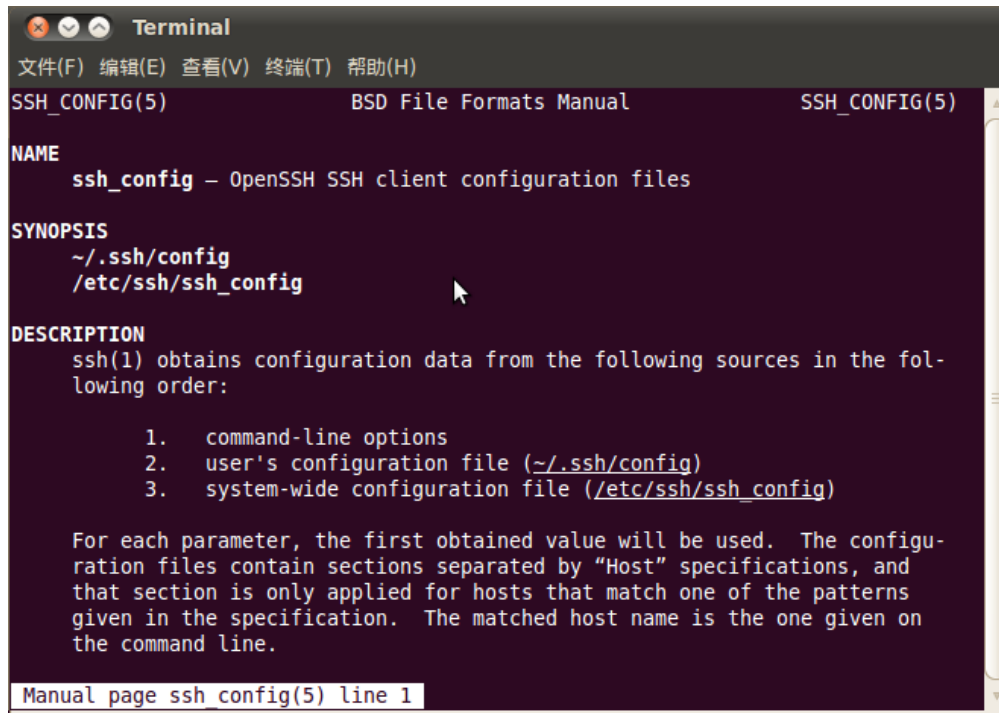
在 `console` 输入如下命令即可解决。

```
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`。

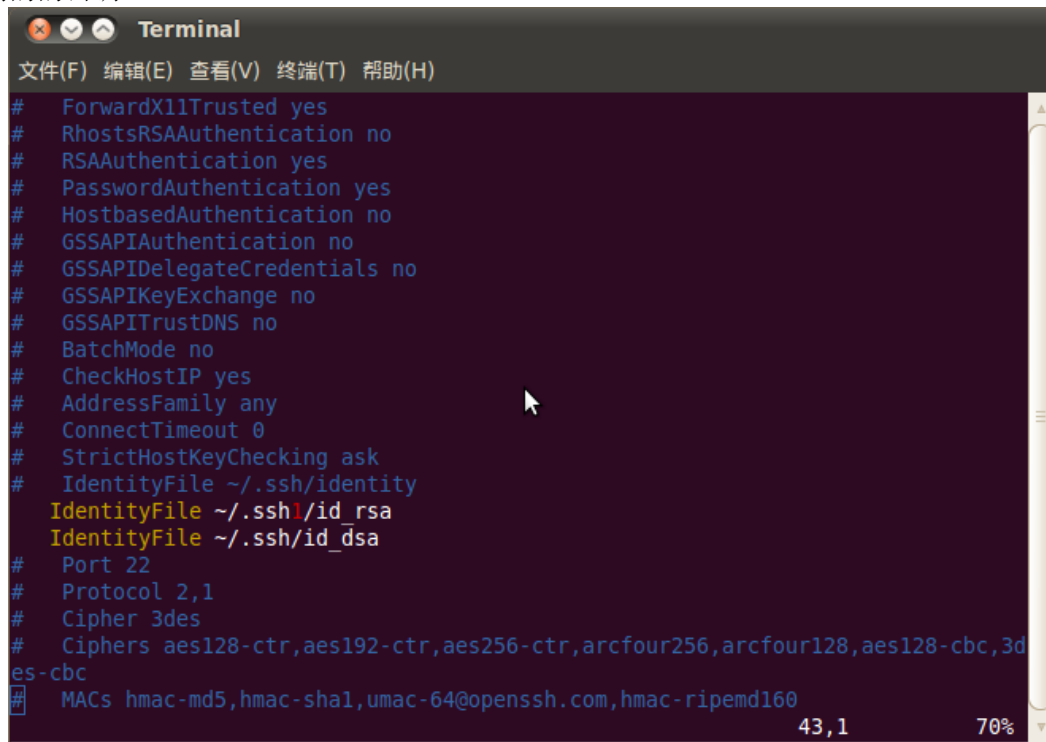
```
~$ man ssh_config
```



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“~/.ssh/id\_rsa”作为认证私钥。通过这种方法，可以切换不同的密钥。



## 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 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 参考文档

更多详细说明，可参考文档 `sdk/docs/RKTools manuals/Rockchip SDK Kit 申请指南 V1.6-201905.pdf`。