密级状态：绝密（　　）　　秘密（　　）　　内部（　）　　公开（√）

RK3326_ANDROID9.0_SDK发布说明

（技术部，第二系统产品部）

文件状态：	当前版本：	V1.0
[ ] 正在修改	作　　者：	卞金晨
[√] 正式发布	完成日期：	2018-11-15
	审　　核：	黄祖芳/吴良清
	完成日期：	2018-11-15

福州瑞芯微电子股份有限公司

Fuzhou　Rockchips　Electronics　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
Beta_V0.0.1	卞金晨	2018.09.25	Beta 测试版	
V1.0	卞金晨	2018.11.15	发布正式版	

目 录

1 概述

本 SDK 是基于谷歌 Android9.0 64bit 系统，适配瑞芯微 RK3326 芯片的软件包，适用基于 RK3326 平台开发的产品。

2 主要支持功能

参数	模块名
数据通信	Wi-Fi、USB 以太网卡、USB、SDCARD
应用程序	Launcher3、APK 安装器、浏览器、计算器、日历、相机、闹钟、下载、电子邮件、资源管理器、GMS 应用、音乐、录音、设置、视频播放器

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。

RK3326_ANDROID9.0_SDK 下载地址如下：

```
repo init --repo-url=ssh://git@www.rockchip.com.cn:2222/repo-
release/tools/repo.git  -u ssh://git@www.rockchip.com.cn
:2222/Android_pie_stable/platform/rk3326/manifests.git-
m RK3326_Android_Pie_release.xml
```

如果需要包含 GMS 包的 SDK(需要开通权限)，使用如下地址：

```
repo init --repo-url=ssh://git@www.rockchip.com.cn:2222/repo-
release/tools/repo.git  -u ssh://git@www.rockchip.com.cn
:2222/Android_pie_stable/platform/rk3326/manifests.git-
m RK3326_Android_Pie_Express_release.xml
```

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 ssh:// git@www.rockchip.com.cn:2222/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 Rk3326_Android_Pie_release_20180925.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 RK3326

tar zxvf Rk3326_Android_Pie_release_20180925.tar.gz -C RK3326

cd RK3326

.repo/repo/repo sync -l

.repo/repo/repo sync
```

3.2 补充说明

Android9.0 SDK 已不再支持 UMS 功能，平台设备皆使用合并分区；

Android9.0 SDK 已支持全盘加密功能；

Android9.0 SDK 已支持 Verified boot 2.0 (avb)的功能。

4 SDK 编译说明

4.1 JDK 安装

Android9.0 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
```

4.2 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，其中 adb remount 前要先 disable-verity，执行 adb disable
-verity

4.3 代码编译

4.3.1 uboot 编译步骤

```
cd u-boot

make clean

make mrproper

./make.sh rk3326   适用于 RK3326 芯片

./make.sh px30   适用于 px30 芯片
```

编译完，会生成

trust.img

rk3326_loader_v1.10.112.bin 适用于 RK3326 芯片

px30_loader_v1.10.112.bin 适用于 px30 芯片（保证 u_boot 目录只有一个 bin）

uboot.img

三个文件。

4.3.2 kernel 编译步骤

1)、RK3326 平板样机配置与编译如下：

使用的 dts 是：arch/arm64/boot/dts/rockchip/rk3326-863-lp3-v10-avb.dts

```
cd kernel

make ARCH=arm64 rockchip_defconfig

make ARCH=arm64 rk3326-863-lp3-v10-avb.img  适用于 RK3326 芯片

make ARCH=arm64 px30-evb-ddr3-v10-avb.img 适用于 px30 芯片
```

4.3.3 Android 编译及固件生成步骤

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

You're building on Linux

Lunch menu… pick a combo:

    1. aosp_arm-eng
    2. aosp_arm64-eng
    3. aosp_mips-eng
    4. aosp_mips64-eng
    5. aosp_x86-eng
    6. aosp_x86_64-eng
    7. rk3326_evb-userdebug
    8. rk3326_evb-user
    9. rk3326_mid-userdebug
    10. rk3326_mid-user
    11. rk3326_m2g-userdebug
    12. rk3326_m2g-user
    13. rk3326_32bit-userdebug
    14. rk3326_32bit-user
```

如果是平板**:**

Android Pie Go 版本**,** 选择 rk3326_mid-userdebug**,** 输入对应序号 **9**。

Android Pie Normal 版本**,** 选择 rk3326_m2g-userdebug**,**输入对应序号 **11**。

Android Pie 32 位版本**,** 选择 rk3326_32bit-userdebug**,** 输入对应序号 **13**。

$ make -j4

注：Go 为 Google 专门针对低内存设备(1G DDR 及以下）进行优化的 Android 版本；Normal 为传统的 GMS 版本，适用于 2G DDR 及以上的设备.

完成编译后，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-rk3326_mid 目录。

```
rockdev/Image-rk3326_mid
├── boot.img
├── kernel.img
├── MiniLoaderAll.bin
├── vbmeta.img
├── dtbo.img
├── 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
```

得到所有镜像文件后，为了方便烧写及量产，通常可手动将这些单独的镜像通过脚本打包成为 update.img。

4.4 刷机说明

SDK 提供烧写工具，如下图所示，版本为 2.59。编译生成相应的固件后，进入 loader 模式，

即可进行刷机。对于已烧过其它固件的机器，请选择低格设备，擦除 idb，然后进行刷机。
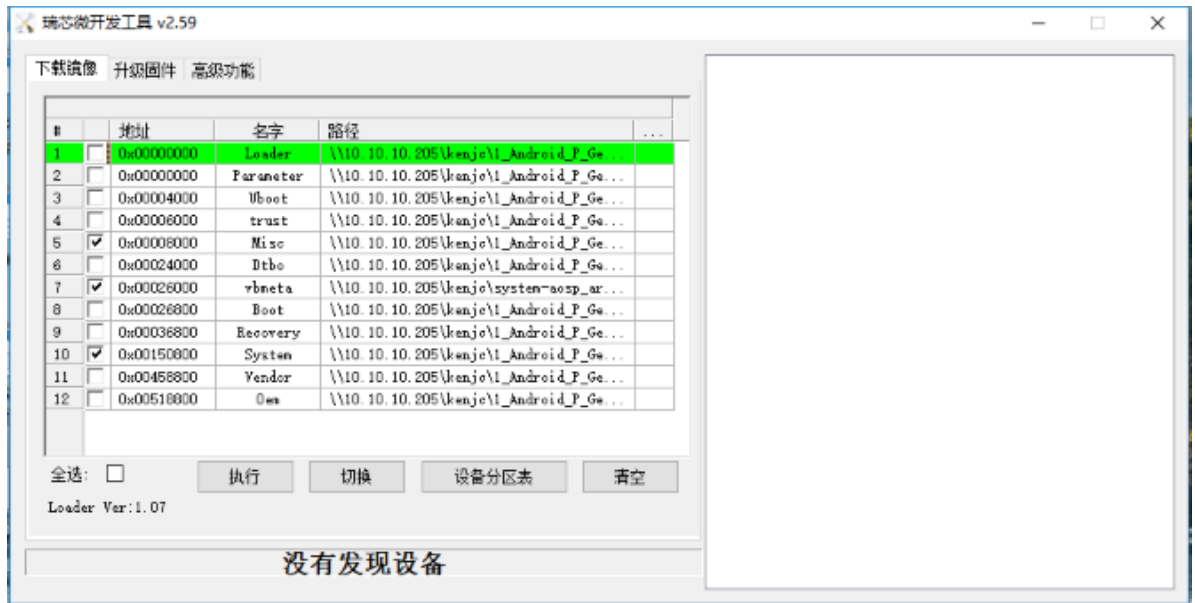
注：烧写工具必须使用 2.59 及以上版本的工具，量产工具使用 1.6 及以上版本；linux 下的烧写工具使用 1.34 及以上版本。

附录 **A** 编译开发环境搭建

1. **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.

2. **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.

3. **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.

4. 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:

sudo apt-get update

sudo apt-get install openjdk-8-jdk

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:

   openjdk-8-jre-headless_8u45-b14-1_amd64.deb with SHA256 0f5aba8db39088283b51e00054813063173a4d8809f70033976f83e214ab56c0

   openjdk-8-jre_8u45-b14-1_amd64.deb with SHA256 9ef76c4562d39432b69baf6c18f199707c5c56a5b4566847df908b7d74e15849

openjdk-8-jdk_8u45-b14-1_amd64.deb with SHA256

6e47215cf6205aa829e6a0a64985075bd29d1f428a4006a80c9db371c2fc3c4c

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

sha256sum {downloaded.deb file}

3. Install the packages:

sudo apt-get update

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

sudo dpkg -i {downloaded.deb file}

To fix missing dependencies:

sudo apt-get -f install

Update the default Java version - optional

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

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

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.

```
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
```

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.

5. 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. <username> must be replaced by the actual username of the user who is authorized to access the phones over USB.

```
# adb protocol on passion (Rockchip products)
SUBSYSTEM=="usb", ATTR{idVendor}=="2207",
ATTR{idProduct}=="0010", MODE="0600", OWNER="<username>"
```

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.
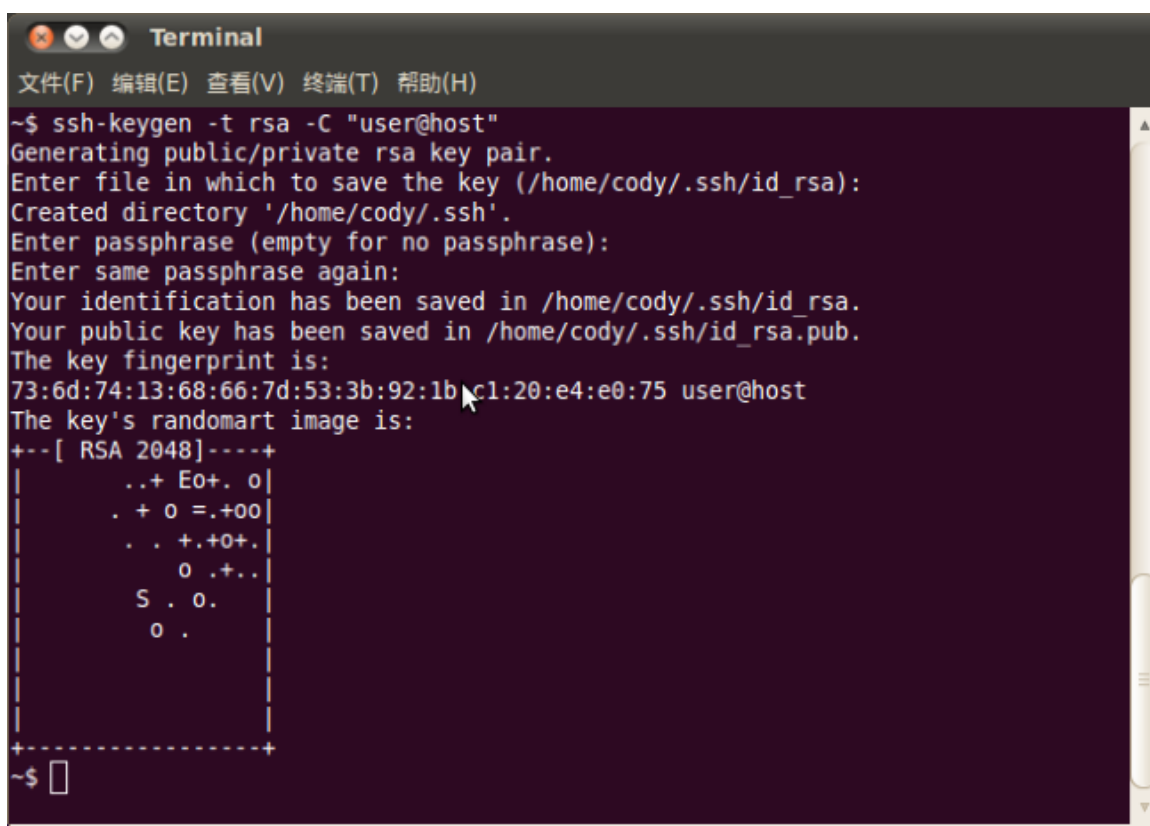
References : http://source.android.com/source/initializing.html

附录 B SSH 公钥操作说明

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。

命令运行完成会在你的目录下生成 key 文件。

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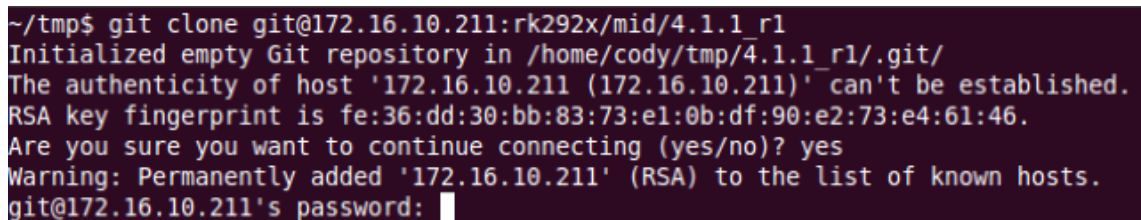
eval `keychain --eval ~/.ssh/id_rsa`

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 failture 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
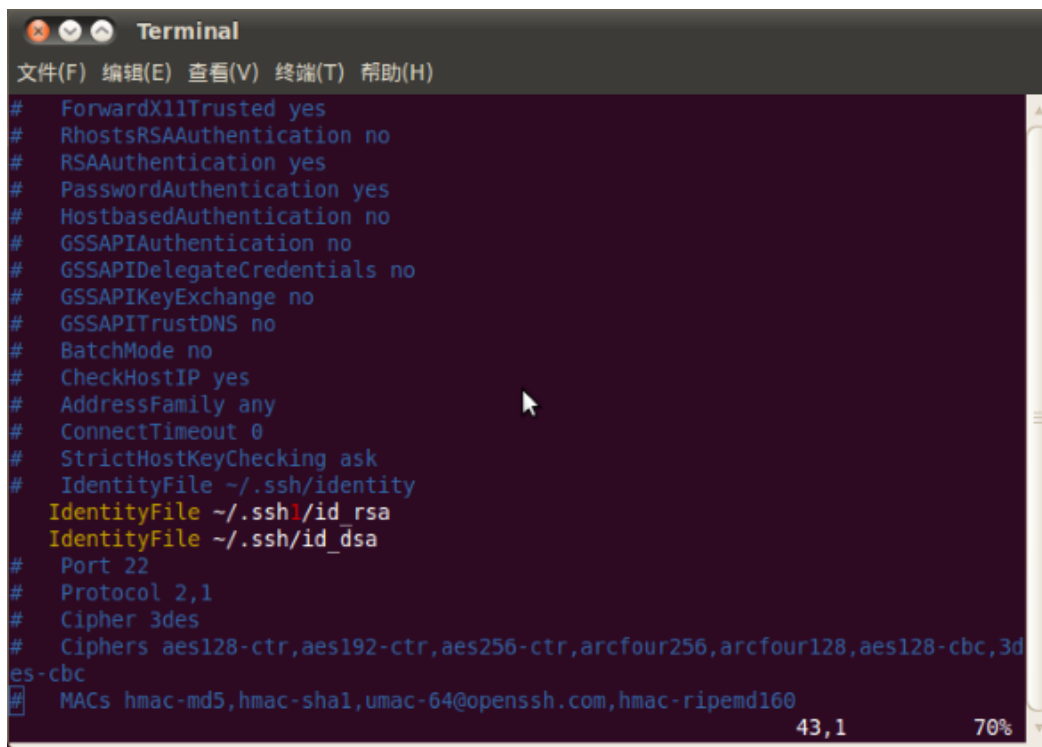
可以参考 ssh_config 文档配置 ssh。

~$ man ssh_config
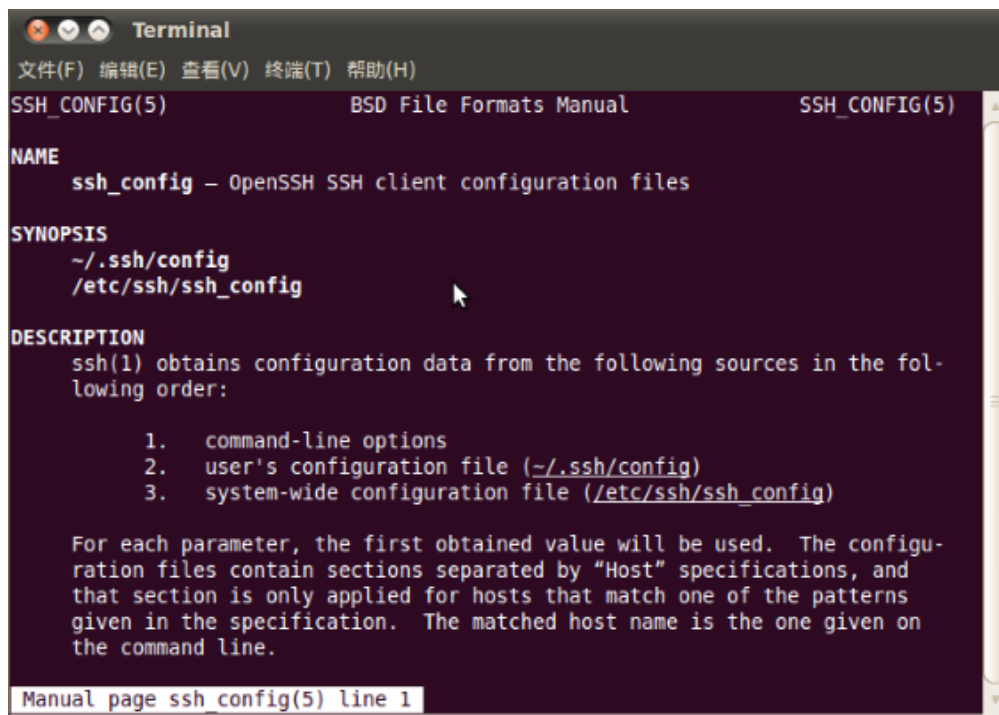
通过如下命令，配置当前用户的 ssh 配置。

~$ 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。

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。