Security Class: Top-Secret (　　)　　　　Secret (　　)　　　　Internal (　　)
Public (　√　)

PX30 Linux SDK Release Note

（Technical Department, Dept.Ⅲ）

	Document ID：	RK-FB-CS-005
Status： [　]draft [　] Modifying [√] Released	Version：	1.3.0
	Author：	Ziyuan Xu
	Date：	2019-02-24
	Reviewer：	Eddie Cai
	Date：	2020-03-02

福州瑞芯微电子股份有限公司
Fuzhou Rockchip Electronics Co.,Ltd.

Revision History

Date	Version No.	Revision Description	Author	Reviewer
2019-04-25	V1.1.0	Initial version	Ziyuan Xu	Eddie Cai
2019-09-17	V1.~~1.1~~2.0	1. Update Linux_SDK_V1.2.0 description 2. Update application compile instruction 3. Add the instruction to get source code from github	Ziyuan Xu	Eddie Cai
2020-02-24	V1.3.0	1. Improve description for ROS 2. Add description for Debian10	Ziyuan Xu	Eddie Cai

Contents

Warranty Disclaimer

This document is provided according to "current situation" and Fuzhou Rockchip Electronics Co., Ltd. ("the company", the same below) is not responsible for providing any express or implied statement or warranty of accuracy, reliability, completeness, marketability, specific purpose or non-infringement of any statement, information and content of this document. This document is intended as a guide only.

Due to product version upgrades or other reasons, this document may be updated or modified from time to time without notice.

Brand Statement

Rockchip, "瑞芯微", "瑞芯", and other Rockchip trademarks are trademarks of Fuzhou Rockchip electronics Co., Ltd., and are owned by Fuzhou Rockchip electronics Co., Ltd.

All other trademarks or registered trademarks mentioned in this document are owned by their respective owners.

Copyright © 2019 Fuzhou Rockchip Electronics Co., Ltd.

Beyond reasonable use range, any unit or individual shall not extract or copy part or all of the content of this document, and shall not spread in any form without the written permission.

Fuzhou Rockchip Electronics Co., Ltd.
Address: No.18 Building, A District, No.89 Software Boulevard, FuZhou, FuJian, PRC
Website: www.rock-chips.com
Customer service Tel.：+86-4007-700-590

Customer service Fax：+86-591-83951833
Customer service e-Mail: fae@rock-chips.com

1 Overview

This SDK is based on Buildroot system, with the kernel 4.4, is suitable for PX30 EVB and all Linux product development on it.
This SDK supports MIPI camera, Music, GPU and other functions. For detailed function debugging and interface description, please read documents in the project directory docs/.

2 Main Functions

Functions	Module Names
Data Communication	Wi-Fi, BT, Camera-MIPI, SDCARD, Ethernet
Application	Music, system settings, gallery, camera, video

3 How to Obtain the SDK

The SDK is released by Rockchip server. Please refer to Chapter 6 SDK Compilation Instruction to build a development environment.
First method to obtain SDK: get source code from Rockchip code server:
In order to obtain PX30 Linux SDK, customers need an account to access the source code repository provided by Rockchip, Please provide SSH public key for server authentication and authorization when apply for SDK from Rockchip technical window. About Rockchip server SSH public key authorization, please refer to Chapter 10 SSH Public Key Operation Instruction.
PX30 Linux SDK download command is as follows：

```
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
px30_linux_release.xml
```

Repo, a tool built on Python script by Google to help manage git repositories, is mainly used to download and manage software repository of projects. The download address is as follows:

```
git clone ssh://git@www.rockchip.com.cn/repo/rk/tools/repo
```

For quick access to the SDK source code, Rockchip Technical Window usually provides corresponding version of SDK initial compression package. In this way, developers can obtain SDK source code through decompressing the initial compression package, which is the same as the one downloaded by repo.
Take px30_linux_sdk_release_v1.3.0_20200224.tgz as an example. After copying initialization package, you can get source code by running the following command:

```
mkdir px30
tar zxvf [x30_linux_sdk_release_v1.3.0_20200224.tgz -C px30
cd px30
.repo/repo/repo sync -l
.repo/repo/repo sync
```

Developers can update via ".repo/repo/repo sync" command according to update instructions that are regularly released by FAE window.
Second method to obtain SDK: get source code from Github open source website:

Download repo tools

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

Make an px30 linux work directory:

mkdir px30

Enter the px30 linux work directory

cd px30/

Initialize repo repository:

../repo/repo init --repo-url=https://github.com/rockchip-linux/repo -u https://github.com/rockchip-linux/manifests -b master -m px30_linux_release.xml

Synchronize the whole project:

../repo/repo sync

4 Software Development Guide

PX30 Linux SDK Kernel version is Linux4.4, rootfs are Buidlroot (2018.02-rc3), to help engineers get familiar with SDK development and debugging more quickly, "Rockchip_Developer_Guide_Linux_Software_EN" is released with the SDK. It can be obtained in the "docs/" directory and will be continuously updated.

5 PX30 Linux Project Directory Introduction

There are buildroot, app, kernel, u-boot, device, docs, external and other directories in the project directory. Each directory or its sub-directories will correspond to a git project, and the commit should be done in the respective directory.
1) buildroot: customize buildroot root file system.
2) app: store some apps like test applications.
3) external: related libraries, including audio, video and so on.
4) kernel: kernel source code.
5) device/rockchip/px30: store some scripts and prepared files for compiling and packaging firmware.
6) docs: store project help files.
7) prebuilts: store cross-compilation toolchain.
8) rkbin: store firmware and tools.
9) rockdev: store compiled output firmware.
10) tools: store some commonly used tools.
11) u-boot: uboot code.

6 SDK Compiling Instruction

Ubuntu 16.04 system：
Please install software packages with below commands to setup Buildroot compiling environment:

sudo apt-get install repo git-core gitk git-gui gcc-arm-linux-gnueabihf 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 dblatex graphviz python-matplotlib libc6:i386 libssl-dev texinfo genext2fs coreutils

Ubuntu 17.04/18.04 system:
In addition to the above, the following dependencies is needed:
apt-get install liblz4-tool lib32gcc-7-dev g++-7 libstdc++-7-dev coreutils

6.1 U-Boot Compiling

Enter project u-boot directory and execute "./make.sh evb-px30" to get:
── u-boot
　├──px30_loader_v1.13.115.bin

```
├── trust.img
└── uboot.img
```

Besides, you can compile (./build.sh uboot) using the build.sh script of the project root directory. If the configuration was changed by 'make menuconfig', had better to save the configuration before compile again.
Note: close bl32, in other words, the trust firmware does not contain Secure OS (there is no TEE related services)

```
export TRUST_PACK_IGNORE_BL32=--ignore-bl32
```

If the Secure OS function is required, you can skip the above environment variable setting, just compile directly.

6.2 Kernel Compiling

Enter project root directory and execute the following command to automatically compile and package kernel:

PX30 EVB：

```
cd kernel
make ARCH=arm64 px30_linux_defconfig
make ARCH=arm64 px30-evb-ddr3-v10-linux.img -j12
```

Besides, you can compile (./build.sh kernel) with the build.sh script of the project root directory.
After compiling, will generate the boot.img in the kernel directory, which includes kernel image, DTB, and logo resources. This version of SDK supports compressed kernel startup, and the corresponding kernel firmware is the zboot.img.

6.3 Recovery Compiling

Enter project root directory and execute the following command to automatically complete compiling and packaging of recovery.

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

The recovery.img is generated in Buildroot directory "/output/rockchip_px30_recovery/images" after compiling.

6.4 Rootfs System

Enter project root directory and execute the following commands to automatically complete compiling and packaging of rootfs.

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

After compiling, the rootfs.img is generated in the rockdev directory.

6.5 Re-compile the Application

Enter project root directory and execute the following commands to re-compile the specified application, eg qcamrea:

```
source envsetup.sh rockchip_px30_64
make qcamera-rebuild
```

6.6 Fully Automatic Compiling

After compiling various parts of kernel/u-boot/recovery/rootfs above, enter root directory of project directory and execute the following commands to automatically complete all compiling:

```
./build.sh
```

Detailed parameter usage, you can use help to search, for example:

```
Usage: build.sh [OPTIONS]
Available options:
BoardConfig*.mk    -switch to specified board config
uboot              -build uboot
kernel             -build kernel
modules            -build kernel modules
rootfs             -build default rootfs, currently build buildroot as default
buildroot          -build buildroot rootfs
ramboot            -build ramboot image
multi-npu_boot     -build boot image for multi-npu board
yocto              -build yocto rootfs
debian             -build debian rootfs
pcba               -build pcba
recovery           -build recovery
all                -build uboot, kernel, rootfs, recovery image
cleanall           -clean uboot, kernel, rootfs, recovery
firmware           -pack all the image we need to boot up system
updateimg          -pack update image
otapackage         -pack ab update otapackage image
save               -save images, patches, commands used to debug
allsave            -build all & firmware & updateimg & save

Default option is 'allsave'.
```

Take PX30 as an example：

The board level configuration of each board needs to be configured in device/rockchip/px30/BoardConfig.mk.

The main configuration of PX30 evb is as follows:

```
# Target arch
export RK_ARCH=arm64
# Uboot defconfig
export RK_UBOOT_DEFCONFIG=evb-px30
# Trust choose ignore bl32, including --ignore-bl32
export TRUST_PACK_IGNORE_BL32=
# Kernel defconfig
export RK_KERNEL_DEFCONFIG=px30_linux_defconfig
# Kernel dts
export RK_KERNEL_DTS=px30-evb-ddr3-v10-linux
# boot image type
export RK_BOOT_IMG=boot.img
# kernel image path
export RK_KERNEL_IMG=kernel/arch/arm64/boot/Image
export RK_KERNEL_ZIMG=kernel/arch/arm64/boot/Image.lz4
# parameter for GPT table
export RK_PARAMETER=parameter-buildroot.txt
# Buildroot config
export RK_CFG_BUILDROOT=rockchip_px30_64
# Recovery config
export RK_CFG_RECOVERY=rockchip_px30_recovery
# ramboot config
export RK_CFG_RAMBOOT=
# Pcba config
export RK_CFG_PCBA=rockchip_px30_pcba
# Build jobs
export RK_JOBS=12
# target chip
export RK_TARGET_PRODUCT=px30
# Set rootfs type, including ext2 ext4 squashfs
export RK_ROOTFS_TYPE=ext4
# rootfs image path
export RK_ROOTFS_IMG=rockdev/rootfs.${RK_ROOTFS_TYPE}
# Set oem partition type, including ext2 squashfs
export RK_OEM_FS_TYPE=ext2
# Set userdata partition type, including ext2, fat
export RK_USERDATA_FS_TYPE=ext2
# Set flash type. support <emmc, nand, spi_nand, spi_nor>
export RK_STORAGE_TYPE=emmc
#OEM config
export RK_OEM_DIR=oem_normal
#userdata config
export RK_USERDATA_DIR=userdata_normal
#misc image
export RK_MISC=wipe_all-misc.img
#choose enable distro module
export RK_DISTRO_MODULE=
```

6.8 Robot Configuration and Compiling

For Robot developers, we provide the deleted BoardConfig board-level configuration for Robot. In device/rockchip/px30/BoardConfig_robot64.mk configuration: buildroot deletes QT, App and other UI display related

configuration, which greatly reduces the firmware size and suitable for robot developers with no screen and small capacity products.

PX30 Linux Robot SDK download command is as follows：

```
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
px30_robot_release.xml
```

If you had download PX30 Linux SDK, you can use the below command in SDK directory to switch Robot SDK:

```
.repo/repo/repo sync -m px30_robot_release.xml
```

Moreover, the Robot SDK support ROS(Robot Operating System), and it provides indigo and kinetic version. If you wanna apply ROS system belong belong on buildroot, please enables some configuration in defconfig like below. It's a recommendatory way to apply indigo ROS for PX30.

```
    --- a/configs/rockchip_px30_64_defconfig
    +++ b/configs/rockchip_px30_64_defconfig
    @@ -1,4 +1,5 @@
     #include "rk3326_arm64.config"
    +#include "ros_indigo.config"
```

6.9 Firmware Package

Type the 'build.sh' to compile fully, and the script will generate all images to IMAGE/ directory, which is include all partitions images.

6.10 Debian10 Compiling

Enter project root directory and execute the following commands to automatically complete compiling Debian10.

```
    ./build.sh distro
```

More detailed introduction for Debian10, please refer to docs/Linux/ApplicationNote/Rockchip_Debian10_Developer_Guide_CN.pdf.

Note: The current compilation of Debian10 QT also depends on the compilation of Buildroot qmake, so please compile Buildroot before compiling Debian10.
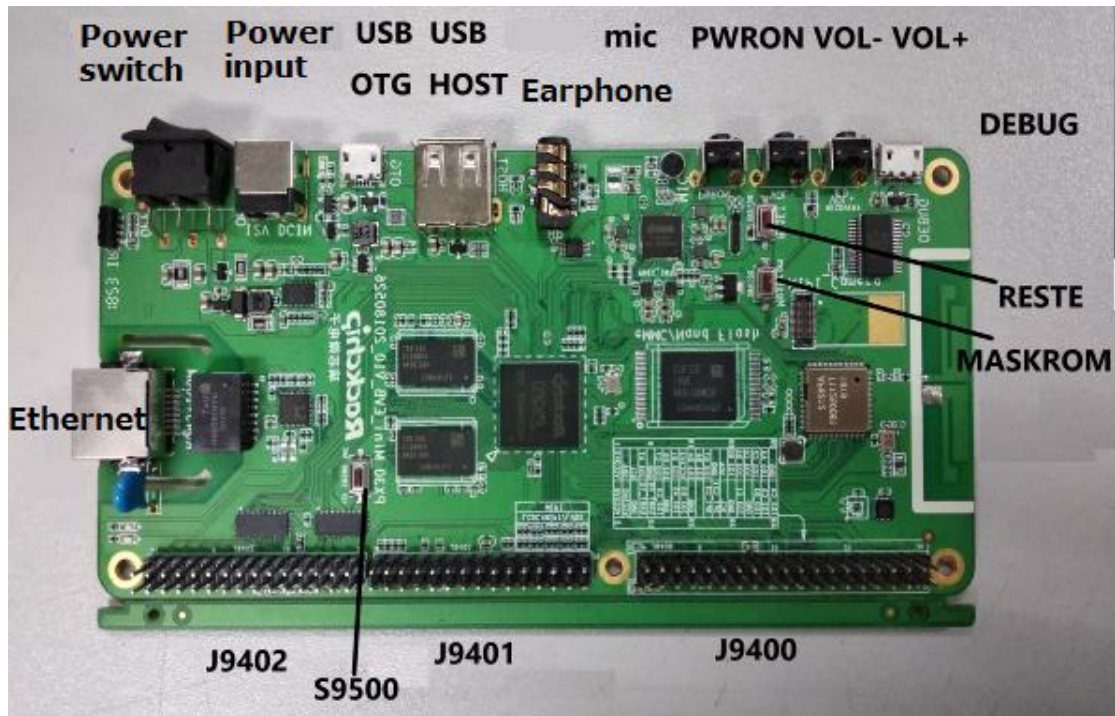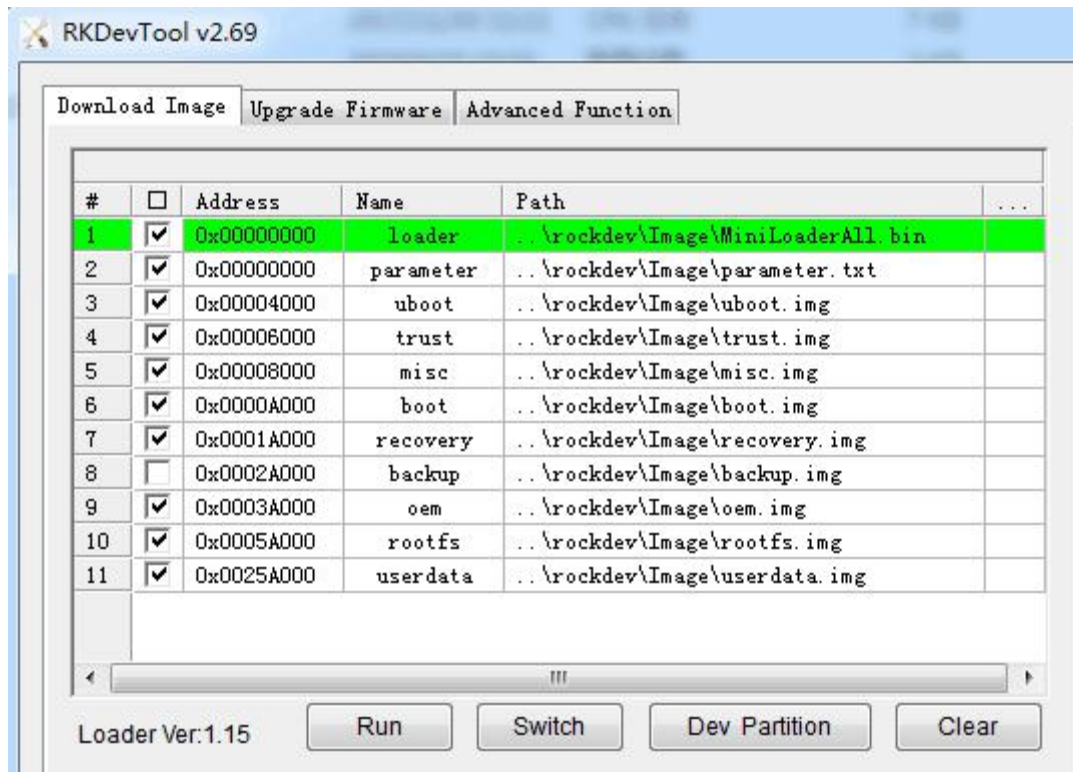
7 Upgrade Instruction

Figure 7-1 PX30 EVB

7.1 Windows Upgrade Instruction

SDK provides windows upgrade tool (this tool should be V2.55 or later version) which is located in project root directory:

```
tools/
├── windows/AndroidTool
```

As shown below, after compiling the corresponding firmware, device needs to enter MASKROM mode for upgrade. After connecting USB cable, long press the button "MSROM" and press reset button "RST" at the same time and then release, device will enter MASKROM Mode. Then you should load the paths of the corresponding images and click "Run" to start upgrade. You can also press the "recovery" button and press reset button "RST" then release to enter loader mode to upgrade. Partition offset and upgrade files of MASKROM Mode are shown as follows (Windows PC needs to run the tool as an administrator):

Figure 7-2 The upgrade tool **AndroidTool.exe**

Note：Before upgrade, please install the latest USB driver, which is in the below directory:
tools/USB driver/DriverAssitant_v4.8

7.2 Linux Upgrade Instruction

The Linux upgrade tool (Linux_Upgrade_Tool should be v1.33 or later versions) is located in "tools/linux" directory. Please make sure your board is connected to maskrom/loader rockusb, if the compiled firmware is in rockdev directory, the upgrade commands are as below:
```
./upgrade_tool ul rockdev/MiniLoaderAll.bin
./upgrade_tool di -p rockdev/parameter.txt
./upgrade_tool di -uboot rockdev/uboot.img
./upgrade_tool di -trust rockdev/trust.img
./upgrade_tool di -misc rockdev/misc.img
./upgrade_tool di -boot rockdev/boot.img
./upgrade_tool di -recovery rockdev/recovery.img
./upgrade_tool di -oem rockdev/oem.img
./upgrade_tool di -rootfs rockdev/rootfs.img
./upgrade_tool di -userdata rockdev/userdata.img
./upgrade_tool rd
```
Or in root directory, run the following command on your machine to upgrade in maskrom state:
```
./rkflash.sh
```

7.3 System Partition Instruction

Default partition (below is PX30 evb reference partition):

Number	Start (sector)	End (sector)	Size	Code	Name
1	16384	24575	4096K	700	uboot
2	24576	32767	4096K	700	trust
3	32768	40959	4096K	700	misc
4	40960	106495	32.0M	700	boot
5	106496	172031	32.0M	700	recovery
6	172032	237567	32.0M	700	backup
7	237568	368639	64.0M	700	oem
8	368640	3514367	1536M	700	rootfs
9	3514368	30535646	12.8G	700	userdata

uboot partition: update uboot.img compiled by uboot.

trust partition: update trust.img compiled by uboot.

misc partition: update misc.img for recovery.

boot partition: update boot.img compiled by kernel.

recovery partition: update recovery.img compiled by recovery.

backup partition: reserved.
oem partition: used by manufacturer to store their app or data. Mounted in /oem directory
rootfs partition: store rootfs.img compiled by buildroot or debian.
userdata partition: store files temporarily generated by app or for users. Read and write, mounted in /userdata directory.

8 Debugging Port Parameter Setting

The Linux SDK uses the UART2 as debug console, the parameter should be baudrate 1500000, data bits 8, stop bit 1, none parity bit, flow control is off.

9 Download Firmware

The firmware download link of PX30_LINUX_SDK_V1.3.0_20200224

PX30_Minievb_V11：
Buildroot:
Baidu YunPan
Google Drive

Debian10：
Baidu YunPan
Google Drive

Robot:
Baidu YunPan
Google Drive
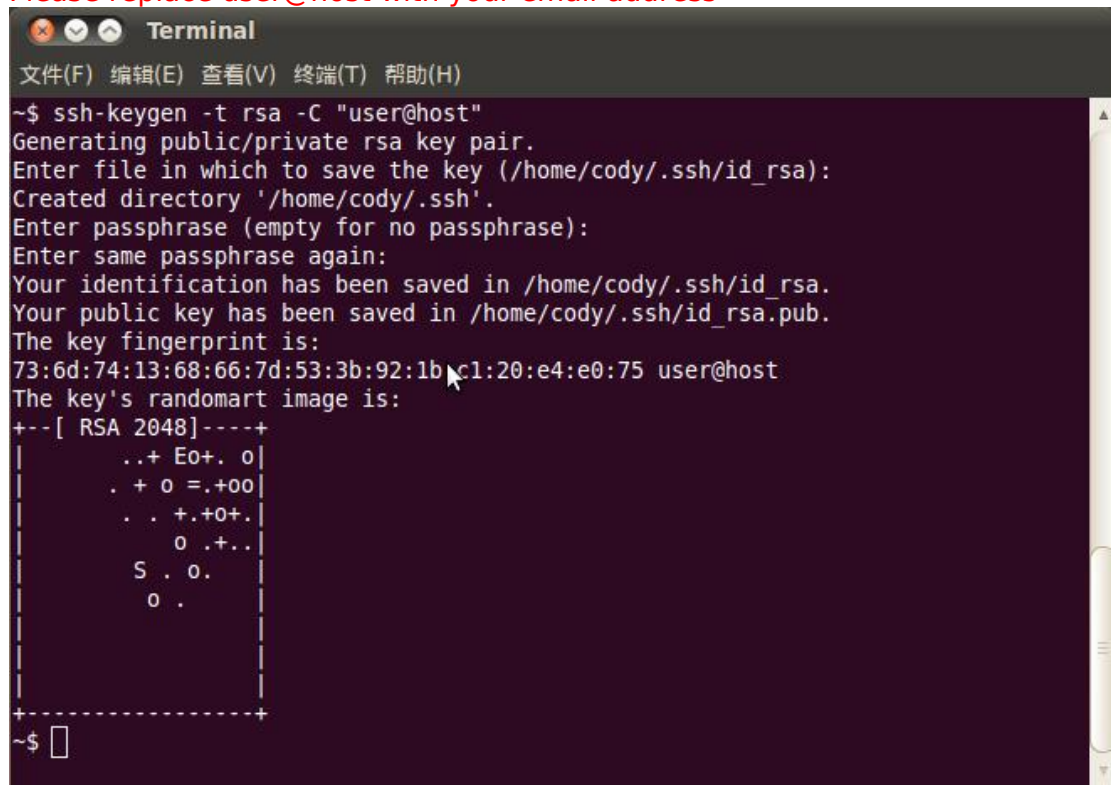
10 SSH Public Key Operation Instruction

10.1 SSH Public Key Generation

Use the following command to generate:：
ssh-keygen -t rsa -C "user@host"
Please replace user@host with your email address

A public key file will be generated in your directory after running the command

```
~$ 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
```

Please keep the generated private key file id_rsa and password properly, and email the public key id_rsa.pub to fae@rock-chips.com, and CC corresponding sales to open the SDK download permission.

10.2 Use Key-chain to Manage Keys

It is recommended to use a simple tool keychain to manage keys.
The detailed usage is as follows:
1. Install keychain package:
$sudo aptitude install keychain
2. Configure the key:
$vim ~/.bashrc
Add the following line:
eval `keychain --eval ~/.ssh/id_rsa`
id_rsa is the private key file name.
After the above configuration, log in to console again and you will be prompted to enter password. Just enter password used to generate the key. If there is no password, you can skip it.
In addition, try not to use sudo or root users unless you know how to handle, otherwise it will lead to permission and key management confusion.

10.3 Multiple Machines Use the Same SSH Public Key

If the same SSH public key needs to be used in different machines, you can copy ssh private key file id_rsa to "~/.ssh/id_rsa" of the machines you want to use.
The following prompt will appear when using a wrong private key, please be careful to replace it with the correct private key.

```
~/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:
```

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

```
~$ 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
```

Adding ssh private key may result in the following error.
Agent admitted failture to sign using the key
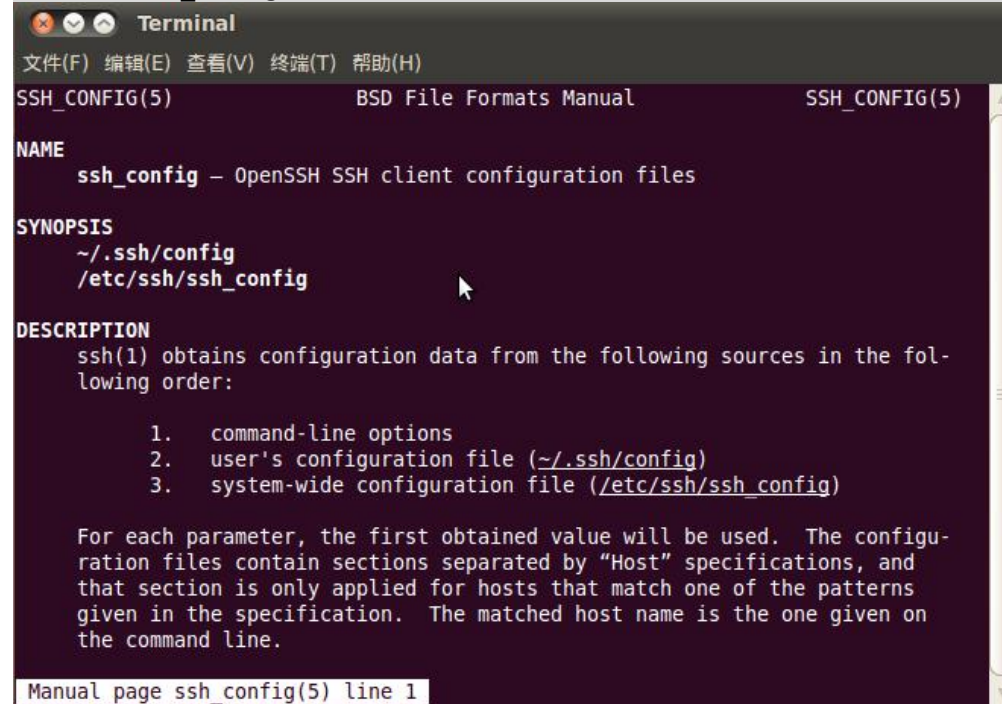Enter the following command in console to solve

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

10.4 One Machine Switches Different SSH Public Keys

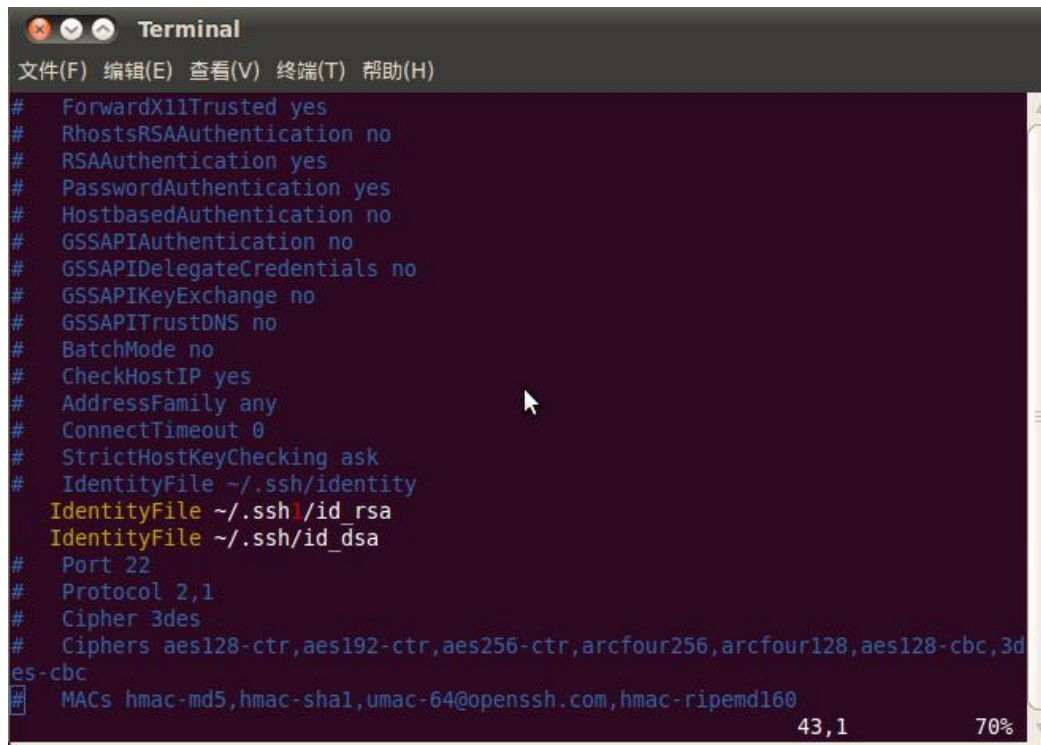You can configure SSH by referring to ssh_config document.

```
~$ man ssh_config
```


Run the following command to configure SSH configuration of current user.

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

As shown in the figure, ssh uses the file "~/.ssh1/id_rsa" of another directory as an authentication private key. In this way, different keys can be switched.

10.5 Key Authority Management

Server can monitor download times and IP information of a key in real time. If an abnormality is found, download permission of the corresponding key will be disabled.
Keep the private key file properly. Do not grant second authorization to third parties.

10.6 Git Access Application Instruction

Please email the public key file created according to above chapters to fae@rock-chips.com, to apply for SDK code download permission.