RV1106开发环境搭建指南

基于官方和板级供应商梳理。

---

1 开发环境

硬件平台：荣品RV1106开发板

软件平台：VMware虚拟机和Ubuntu18.04系统（推荐）

特别说明：不能放在虚拟机与 Windows 的共享文件夹中编译！！！

编译 SDK 环境搭建所依赖的软件包安装命令如下：

```
sudo apt-get update
sudo apt-get install repo git ssh make gcc gcc-multilib \
expect g++ gawk texinfo libssl-dev \
bison flex fakeroot cmake unzip gperf autoconf \
device-tree-compiler libncurses5-dev
```

注意要在虚拟机联网的情况下执行。

2 SDK初始化

在SDK目录下，运行./build.sh init选择板型：

```
./build.sh init
```

```
ls: cannot access 'BoardConfig*.mk': No such file or directory

You're building on Linux

Lunch menu...pick a combo:

------------------------------------------------------------

0. BoardConfig-rp-rv1106.mk

------------------------------------------------------------

Which would you like? [0]:
```

目前只有一种板型可以选择。

3 u-boot

u-boot的作用是引导内核，内核启动之后u-boot就没什么用了。u-boot直接使用SDK中的配置就可以，不用做任何修改。 编译u-boot：

```
./build.sh uboot
```

编译完成后，在./output/image目录下生成如下文件：

```
-rw-rw-r-- 1 book book 260544 Sep 26 22:05 download.bin
-rw-rw-r-- 1 book book 184320 Sep 26 22:05 idblock.img
-rw-rw-r-- 1 book book 262144 Sep 26 22:05 uboot.img
```

uboot.img：是u-boot镜像

idblock.img：用于加载引导uboot.img

download.bin：？

4 内核

内核源代码位于./sysdrv/source/kernel目录。

4.1 内核编译选项

默认的内核配置从./sysdrv/source/kernel/arch/arm/configs/目录下的rv1106_defconfig和rv1106-evb.config生成。要修改内核配置，可以修改这两个文件。

多数情况下，内核编译选项使用默认的配置就可以，不用修改。

4.2 设备树

DTS是设备树脚本，描写了系统的硬件配置，内核定制的主要工作就是修改DTS。

DTS目录在./sysdrv/source/kernel/arch/arm/boot/dts

RP-RV1106板使用的DTS文件是rp-rv1106.dts，包含了很多dtsi文件:

```
/dts-v1/;
#include "rv1106-board.dtsi"
#include "rp-rtc-hym8563.dtsi"
#include "rp-spi2can-mcp2515.dtsi"
#include "rp-sdcard-mmc1.dtsi"
#include "rp-emmc-mmc0-4bit.dtsi"
#include "rp-sound.dtsi"
#include "rp-wifi-sdio.dtsi"
#include "rp-bt-uart0.dtsi"
#include "rp-camera-gc2093x1.dtsi"
...
```

rp-rv1106.dts以及包含的dtsi文件很重要，要和设备的硬件配置保持一致，否则内核驱动不能正常工作。

4.3 内核编译

```
./build.sh kernel
```

编译完成后，在./output/image目录下生成boot.img

```
-rw-rw-r-- 1 book book 3330048 Sep 26 23:15 boot.img
```

编译完成后，在./output/out/sysdrv_out/board_uclibc_rv1106/目录下生成

```
rp-rv1106.dtb
vmlinux
```

rp-rv1106.dtb：这是 .dts 文件经过预处理和编译后生成的二进制文件。操作系统内核会读取这个文件来理解硬件的结构和属性。

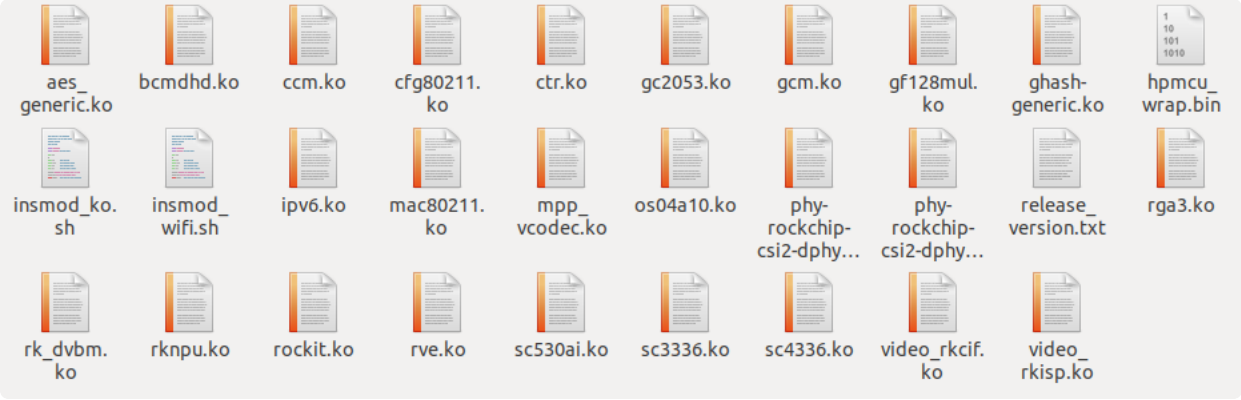
vmlinux：文件是包含了调试信息的未压缩的 Linux 内核可执行文件，它通常用于调试和分析。

4.4 外部驱动模块编译

内核有些驱动程序会编译成外部模块(ko文件)的形式。编译方法：

```
1   ./build.sh driver
```

编译完成后，在./output/out/sysdrv_out/kernel_drv_ko/目录下生成



ko文件放在开发板的目录：/oem/usr/ko/

```
aes_generic.ko  gc2053.ko          insmod_ko.sh   os04a10.ko                  rk_dvbm.ko  sc4336.ko
bcmdhd.ko       gcm.ko             insmod_wifi.sh phy-rockchip-csi2-dphy-hw.ko rknpu.ko    sc530ai.ko
ccm.ko          gf128mul.ko        ipv6.ko        phy-rockchip-csi2-dphy.ko   rockit.ko   video_rkcif.ko
cfg80211.ko     ghash-generic.ko   mac80211.ko    release_version.txt         rve.ko      video_rkisp.ko
ctr.ko          hpmcu_wrap.bin     mpp_vcodec.ko  rga3.ko                     sc3336.ko
```

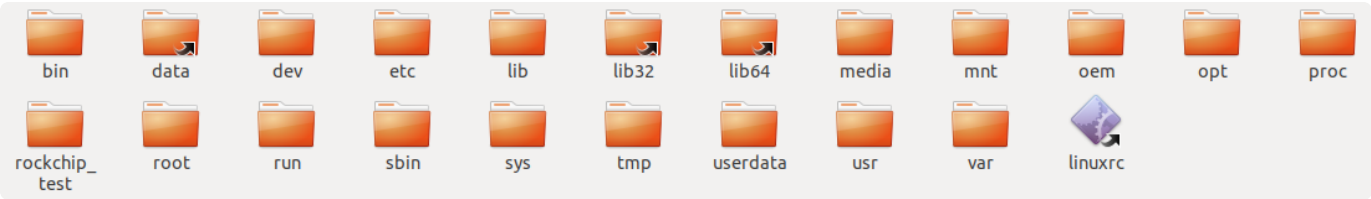
5 根文件系统

根文件系统是linux系统的运行入口。

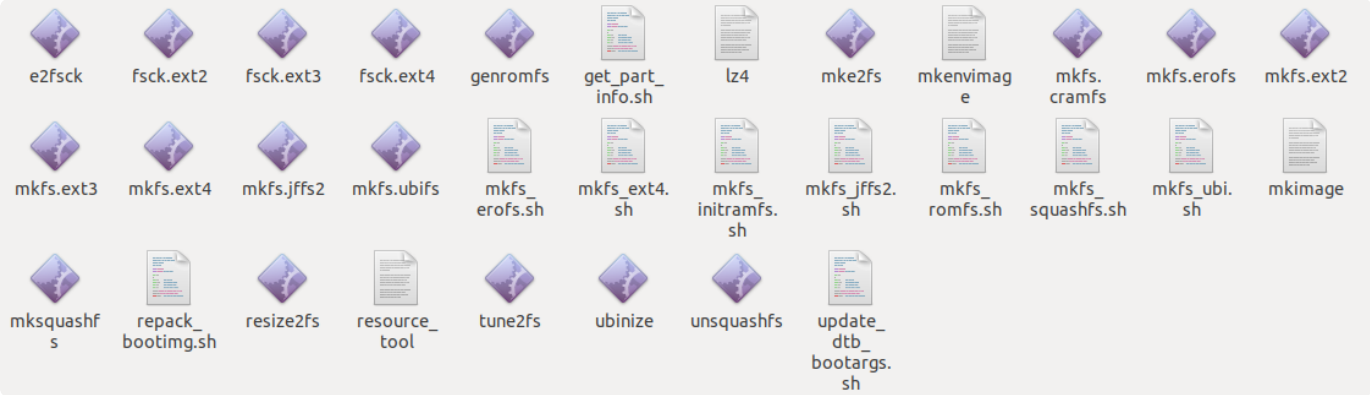
编译方法：

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

生成的文件系统在./output/out/sysdrv_out/rootfs_uclibc_rv1106.tar中，解压后得到：



在./output/out/sysdrv_out/pc/中生成，(?)



在./output/out/sysdrv_out/board_uclibc_rv1106/目录下生成

libthread_db.so.1

libthread_db-1.0.31.so

6 应用层软件

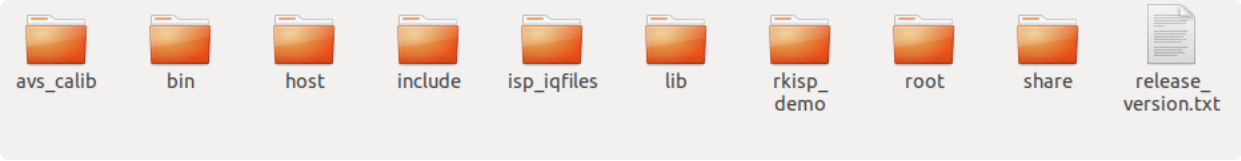
应用层软件包含media开发库和示例程序。

6.1 media开发库

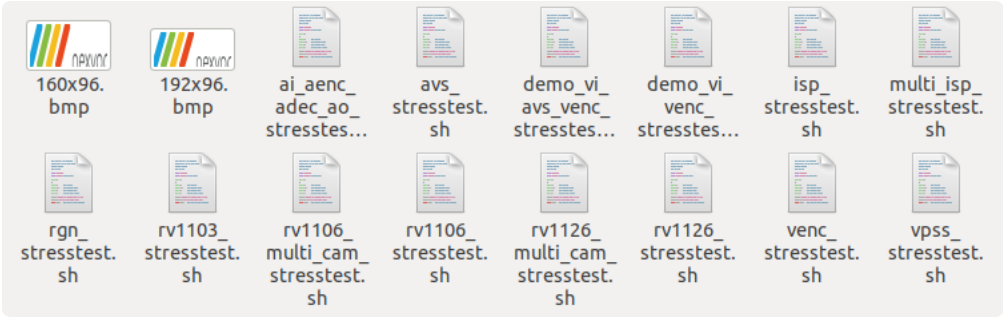
编译方法：

```
1   ./build.sh media
```

编译完成后，在./output/out/media_out/目录下生成



在./output/out/userdata/目录下生成



media开发库放在开发板的目录：/oem/usr/lib/

```
face_mask_classify.data   libfcgi.so          libpcre.so.1          librknnmrt.so
face_quality_v2.data      libfcgi++.so        librga.so             librkrawstream.so
libaec_bf_process.so      libfcgi.so.0        librkaiq.so           librksysutils.so
libcgicc.so               libiconv.so         librkaudio_common.so  librockchip_mpp.so
libcgicc.so.3             libiconv.so.2       librkaudio_detect.so  librockchip_mpp.so.0
libdrm_rockchip.so        libiconv.so.2.6.1   librkaudio.so         librockchip_mpp.so.1
libdrm_rockchip.so.1      libivs.so           librkAVS_genLut.so    librockit_full.so
libdrm_rockchip.so.1.0.0  libkms.so           librkAVS_genStitch.so librockit.so
libdrm.so                 libkms.so.1         librkdemuxer.so       librockit_tiny.so
libdrm.so.2               libkms.so.1.0.0     librkfsmk.so          librockiva.so
libdrm.so.2.4.0           libpcre.so          librkmuxer.so         librve.so
```

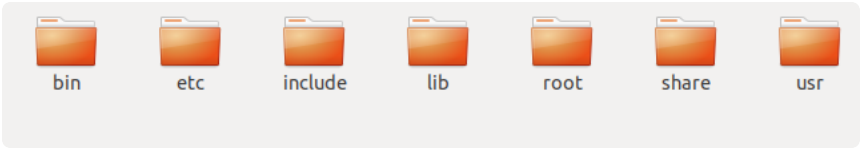
6.2 示例程序app

编译方法：

```
1   ./build.sh app
```

注：app依赖media

编译完成后，在./output/out/app_out/目录下生成



示例程序放在开发板的目录：/oem/usr/bin/

```
cgi-fcgi            rk_mpi_ao_test          sample_ai_aenc_adec_ao_stresstest
dumpsys             rk_mpi_avio_test        sample_avs
modetest            rk_mpi_avs_test         sample_avs_stresstest
mpi_enc_test        rk_mpi_dup_venc_test    sample_demo_dual_camera
mpp_info_test       rk_mpi_mb_test          sample_demo_multi_camera_eptz
rgaImDemo           rk_mpi_mmz_test         sample_demo_vi_avs_venc
rk_adc_test         rk_mpi_rgn_test         sample_demo_vi_avs_venc_stresstest
rkaiq_3A_server     rk_mpi_sys_test         sample_demo_vi_venc
rk_event_test       rk_mpi_tde_test         sample_demo_vi_venc_stresstest
rk_gpio_test        rk_mpi_venc_test        sample_isp_stresstest
rkipc               rk_mpi_vi_dup_test      sample_mulit_isp_stresstest
rkisp_demo          rk_mpi_vi_test          sample_multi_vi
rk_led_test         rk_mpi_vpss_test        sample_multi_vi_avs
RkLunch.sh          rk_pwm_test             sample_multi_vi_avs_osd_venc
RkLunch-stop.sh     rk_rve_sample_test      sample_rgn_stresstest
rk_mpi_adec_test    rk_system_test          sample_venc_stresstest
rk_mpi_aenc_test    rk_time_test            sample_vi
rk_mpi_af_test      rk_watchdog_test        sample_vi_vpss_osd_venc
rk_mpi_ai_test      sample_ai               sample_vpss_stresstest
rk_mpi_amix_test    sample_ai_aenc          simple_adec_ao
```

7 固件打包

运行以下命令进行固件打包：

```
./build.sh firmware
```

这个命令执行以下几项工作：

- 产生环境配置镜像文件env.img
- 把./output/out/rootfs_uclibc_rv1106打包成ext4镜像文件rootfs.img
- 把./output/out/oem(新目录)打包成ext4镜像文件oem.img
- 把./output/out/userdata打包成ext4镜像文件userdata.img
- 把所有镜像文件打包成一个固件镜像文件update.img

执行完固件打包命令后，在./output/image目录下新增文件如下：

env.img

oem.img

userdata.img

update.img

rootfs.img

sd_update.txt

tftp_update.txt

sd_update.txt：SD卡升级文件 ，

tftp_update.txt：tftp升级文件 。

参考文档《Rockchip_RV1106_RV1103_User_Guide_Linux_EVB_CN.pdf》第6、7页。

8 文件系统和工具

- 简化文件系统，单独提取出来放在fs目录下
- SDK常用操作，写成了一个脚本mkimg, 放在tool目录下
- mkimg中的IMG_OUT_PATH指定了镜像输出目录
- mkimg的使用方法：

```
cd tool
./mkimg uboot    #编译uboot
./mkimg boot     #编译boot
./mkimg root     #打包rootfs分区
./mkimg data     #打包oem分区和userdata分区
./mkimg upkg     #打包固件update.img
```

9 运行示例程序

9.1 挂载NFS

```
mount -t nfs -o vers=3,nolock 192.168.1.111:/home/nfs /nfs
```

注意：192.168.1.111为虚拟机IP地址，虚拟机需要开启NFS Server。

9.2 后台运行rkaiq_3A_Server

```
cd /oem/usr/bin
./rkaiq_3A_server  /oem/usr/share/iqfiles  &
```

瑞芯微提供了AIQ（自动图像质量）调整机制，和ISP配合，对摄像头图像进行自动调整，保证在各种光线环境下图像不失真。

rkaiq_3A_server是瑞芯微提供的摄像头图像质量自动调整程序，应该在启用摄像头之前后台运行这个程序，否则摄像头的图像质量会严重失真。

如果不启用rkaiq_3A_server，就需要在自己的程序里加入AIQ相关的代码。瑞芯微也提供了AIQ编程的API。

9.3 运行摄像头测试程序 simple_vi_venc

```
./simple_vi_venc -w 1280 -h 720 -e h264 -o test.h264 -c 1000
```

simple_vi_venc的功能是捕获摄像头画面，进行视频压缩编码，保存到文件中。

源代码在./media/samples/simple_test/simple_vi_venc.c，可以通过阅读这个源代码，了解RV1106平台上摄像头画面获取和视频编码的编程方法。

上述程序设置画面分辨率为1280*720，捕捉1000帧画面，编码格式为H.264，保存到test.h264文件中。

9.4 查看结果

先拷贝到虚拟机上：

```
cp  test.h64  /nfs
```

在虚拟机上甩ffplay播放：

```
ffplay   /home/nfs/test.h264
```