**解包/打包 RK 固件**

**RK 固件格式**

RK 固件 release\_update.img 包含引导加载程序 loader.img 和实际的固件数据update.img:

release\_update.img

|- loader.img

`- update.img

update.img 是打包工具读取 package-file 索引文件从而将多个映像文件打包而成固件 。一个典型的 package-file 文件的内容为:

# NAME Relative path

package-file package-file

bootloader Image/MiniLoaderAll.bin

parameter Image/parameter.txt

trust Image/trust.img

uboot Image/uboot.img

misc Image/misc.img

resource Image/resource.img

kernel Image/kernel.img

boot Image/boot.img

recovery Image/recovery.img

system Image/system.img

backup RESERVED

#update-script update-script

#recover-script recover-script

* package-file：update.img 的索引文件，也包含在 update.img 中。
* Image/MiniLoaderAll.bin：通过 CPU ROM 代码加载的第一个 bootloader。
* Image/parameter.txt：参数文件，可以在其中设置内核启动参数和分区布局。
* Image/trust.img：Arm trust file (ATF) 映像，用于安全启动。
* Image/misc.img：misc 分区映像，用于控制 Android 的启动模式。
* Image/kernel.img：Android 内核映像。
* Image/resource.img：具有启动图片和内核设备树的资源映像。
* Image/boot.img：Android initramfs，一个在正常启动时加载的根文件系统，包含重要的初始化和服务描述。
* Image/recovery.img：Recovery 模式映像。
* Image/system.img：Android 系统分区映像。

解包是从 release\_update.img 中提取 update.img，然后解开里面的所有映像文件。 重新打包时，则是相反的过程。它将由 package-file 描述的所有映像文件合成到 update.img 中，该文件将与 bootloader 一起打包以创建最终的 release\_update.img。

**安装工具**

git clone https://github.com/TeeFirefly/rk2918\_tools.git

cd rk2918\_tools

make

sudo cp afptool img\_unpack img\_maker mkkrnlimg /usr/local/bin

**解包 RK 固件**

* 解包 release\_update.img:
* $ cd /path/to/your/firmware/dir
* $ img\_unpack Firefly-RK3399\_20161027.img img
* rom version: 6.0.1
* build time: 2016-10-27 14:58:18
* chip: 33333043
* checking md5sum....OK
* 解包 update.img:
* $ cd img
* $ afptool -unpack update.img update
* Check file...OK
* ------- UNPACK -------
* package-file 0x00000800 0x00000280
* Image/MiniLoaderAll.bin 0x00001000 0x0003E94E
* Image/parameter.txt 0x00040000 0x00000350
* Image/trust.img 0x00040800 0x00400000
* Image/uboot.img 0x00440800 0x00400000
* Image/misc.img 0x00840800 0x0000C000
* Image/resource.img 0x0084C800 0x0003FE00
* Image/kernel.img 0x0088C800 0x00F5D00C
* Image/boot.img 0x017EA000 0x0014AD24
* Image/recovery.img 0x01935000 0x013C0000
* Image/system.img 0x02CF5000 0x2622A000
* RESERVED 0x00000000 0x00000000
* UnPack OK!
* 在 update 目录检查目录树:
* $ cd update/
* $ tree
* .
* ├── Image
* │ ├── boot.img
* │ ├── kernel.img
* │ ├── MiniLoaderAll.bin
* │ ├── misc.img
* │ ├── parameter.txt
* │ ├── recovery.img
* │ ├── resource.img
* │ ├── system.img
* │ ├── trust.img
* │ └── uboot.img
* ├── package-file
* └── RESERVED
* 1 directory, 12 files

**打包 RK 固件**

首先, 确保在 parameter.txt 文件中的 system 分区足以容纳 system.img. 参考 [《Parameter 文件格式》](http://www.t-firefly.com/download/Firefly-RK3399/docs/Rockchip%20Parameter%20File%20Format%20Ver1.3.pdf) 了解分区布局。

例如，在 parameter.txt 的前缀为 “CMDLINE” 的行中，可以找到类似于以下内容的 system 分区的描述:

0x00200000@0x000B0000(system)

“@”之前的十六进制字符串是分区的大小（以扇区为单位，此处为 1 扇区= 512 字节），因此系统分区的大小为:

$ echo **$((** 0x00200000 \* 512 / 1024 / 1024 **))**M

1024M

创建 release\_update\_new.img:

*# 当前目录仍然是 update/，package-file 文件*

*# 以及所描述的所有映像文件均在此目录下。*

*# 将参数文件复制为 "parameter" 文件，afptool 默认使用此名。*

$ afptool -pack . ../update\_new.img

------ PACKAGE ------

Add file: ./package-file

Add file: ./Image/MiniLoaderAll.bin

Add file: ./Image/parameter.txt

Add file: ./Image/trust.img

Add file: ./Image/uboot.img

Add file: ./Image/misc.img

Add file: ./Image/resource.img

Add file: ./Image/kernel.img

Add file: ./Image/boot.img

Add file: ./Image/recovery.img

Add file: ./Image/system.img

Add file: ./RESERVED

Add CRC...

------ OK ------

Pack OK!

$ img\_maker -rk33 loader.img update\_new.img release\_update\_new.img

generate image...

append md5sum...

success!

**自定义**

**自定义 system.img**

system.img 是 ext4 文件系统格式的映像文件，可以直接挂载到系统进行修改:

sudo mkdir -p /mnt/system

sudo mount Image/system.img /mnt/system

cd /mnt/system

*# Modify the contents of the inside.*

*# Pay attention to the free space,*

*# You can not add too many APKs*

*# 结束时，需要卸载*

cd /

sudo umount /mnt/system

请注意，system.img 的可用空间几乎为 0，如果需要扩展映像文件，请相应地调整 parameter.txt 中的分区布局。

以下是如何将映像文件的大小增加 128MB 的示例。

扩展之前先确保 system.img 没有被系统挂载上:

mount | grep system

改变映像文件及其内在文件系统的大小:

dd **if**=/dev/zero bs=1M count=128 >> Image/system.img

*# Expand file system information*

e2fsck -f Image/system.img

resize2fs Image/system.img