Android boot.img 结构

Android 的boot.img 包括 boot header，kernel， ramdisk

首先来看看Makefile是如何产生我们的boot.img的：

boot镜像不是普通意义上的文件系统，而是一种特殊的Android定制格式，由boot header，压缩的内核，ramdisk以及second stage loader（可选）组成，可以从mkbootimg.h文件中看到。

boot,img文件跳过4k的文件头之后，包括两个 gz包，一个是boot.img-kernel.gz：Linux内核，一个是boot.img-ramdisk.cpio.gz

大概的组成结构如下

\*

\*\* +-----------------+

\*\* | boot header | 1 page

\*\* +-----------------+

\*\* | kernel | n pages

\*\* +-----------------+

\*\* | ramdisk | m pages

\*\* +-----------------+

\*\* | second stage | o pages

\*\* +-----------------+

boot header为包括命令行参数等等,地址为000-----0xFFF

ramdisk为 1F8B0800000000开头

kernel为 0000A0E1 重复8遍开头

关于boot header这个数据结构我们需要重点注意，在这里我们关注其中几个比较重要的值，这些值定义在boot/boardconfig.h里面，不同的芯片对应vendor下不同的boardconfig，在这里我们的值分别是（分别是kernel/ramdis/tags载入ram的物理地址）：

#define PHYSICAL\_DRAM\_BASE 0x00200000

#define KERNEL\_ADDR (PHYSICAL\_DRAM\_BASE + 0x00008000)

#define RAMDISK\_ADDR (PHYSICAL\_DRAM\_BASE + 0x01000000)

#define TAGS\_ADDR (PHYSICAL\_DRAM\_BASE + 0x00000100)

#define NEWTAGS\_ADDR (PHYSICAL\_DRAM\_BASE + 0x00004000)

上面这些值分别和我们开篇时候提到的那几个名词相对应，比如kernel\_addr就是ZTEXTADDR，RAMDISK\_ADDR就是INITRD\_PHYS,而TAGS\_ADDR就是PARAMS\_PHYS。bootloader会从boot.img的分区中将kernel和ramdisk分别读入RAM上面定义的地址中，然后就会跳到ZTEXTADDR开始执行。

ramdisk映像是一个最基础的小型文件系统，它包括了初始化系统所需要的全部核心文件，例如:初始化init进程以及init.rc（可以用于设置很多系统的参数）等文件。以下是一个典型的ramdisk中包含的文件列表：

./init.trout.rc

./default.prop

./proc

./dev

./init.rc

./init

./sys

./init.goldfish.rc

./sbin

./sbin/adbd

./system

./data

如果要分离可以用winhex将boot。img打开

找到0000A0E1 到1F8B0800000000的前面的数据块保持为ramdisk.img

找到1F8B0800000000到文件尾部的数据块保持为kernel

out/host/linux-x86/bin/mkbootimg --kernel out/target/product/msm7630\_surf/kernel --ramdisk out/target/product/msm7630\_surf/ramdisk.img --cmdline "console=ttyMSM1,115200n8 androidboot.hardware=qcom" --base 0x00200000 --pagesize 4096 --output out/target/product/msm7630\_surf/boot.img

根据上面的命令我们可以首先看看mkbootimg 这个工具的源文件：system/core/mkbootimg.c。看完之后我们就能很清晰地看到boot.img的内部构造，它是由boot header /kernel /ramdisk /second stage构成的，其中前3项是必须的，最后一项是可选的。

header + padding + kernel + padding + ramdisk + padding + ...

4 \* 2, magic，固定为"ANDROID!"

4 \* 1, kernel长度，小端unsigned

4 \* 1, kernel地址，应为base + 0x00008000 (base为0x200000)

4 \* 1, ramdisk长度，小端unsigned

4 \* 1, ramdisk地址，应为base + 0x01000000

4 \* 1, second stage长度，小端unsigned，为0

4 \* 1, second stage地址，应为base + 0x00f00000

4 \* 1, tags地址，应为base + 0x00000100

4 \* 1, page大小，小端unsigned, 为2048或者4096

4 \* 2, 未使用，固定为0x00

4 \* 4, 板子名字，一般为空

4 \* 128, 内核命令参数，为mem=211M console=ttyMSM2,115200n8 androidboot.hardware=qcom console=ttyUSBCONSOLE0 androidboot.console=ttyUSBCONSOLE0

4 \* 8, id, 为sha之类，实际写0x00就可

padding, 以上header为608字节，把这部分补齐到page\_size \* 2大小

kernel\_size, kernel内容

padding，把kernel\_size补齐到page\_size \* 2

ramdisk\_size, ramdisk内容

padding, 把ramdisk补齐到page\_size \* 2

second\_size, second内容，一般为0

padding, 补齐second\_sise为page\_size，一般为0

配合 boot.img 来看会比较好理解.

由此可知 boot\_img\_hdr 中各成员值为：

TAGS\_ADDR 如上 target/<your-platform>/rules.mk 所定义的 : 0x40200100, 所以 boot\_linux(), 就是传入TAGS\_ADDR,

然后将资料写入 tag, tag 的结构如下所示.

然后进入到 kernel 的入口函数: entry(0, machtype, tags)