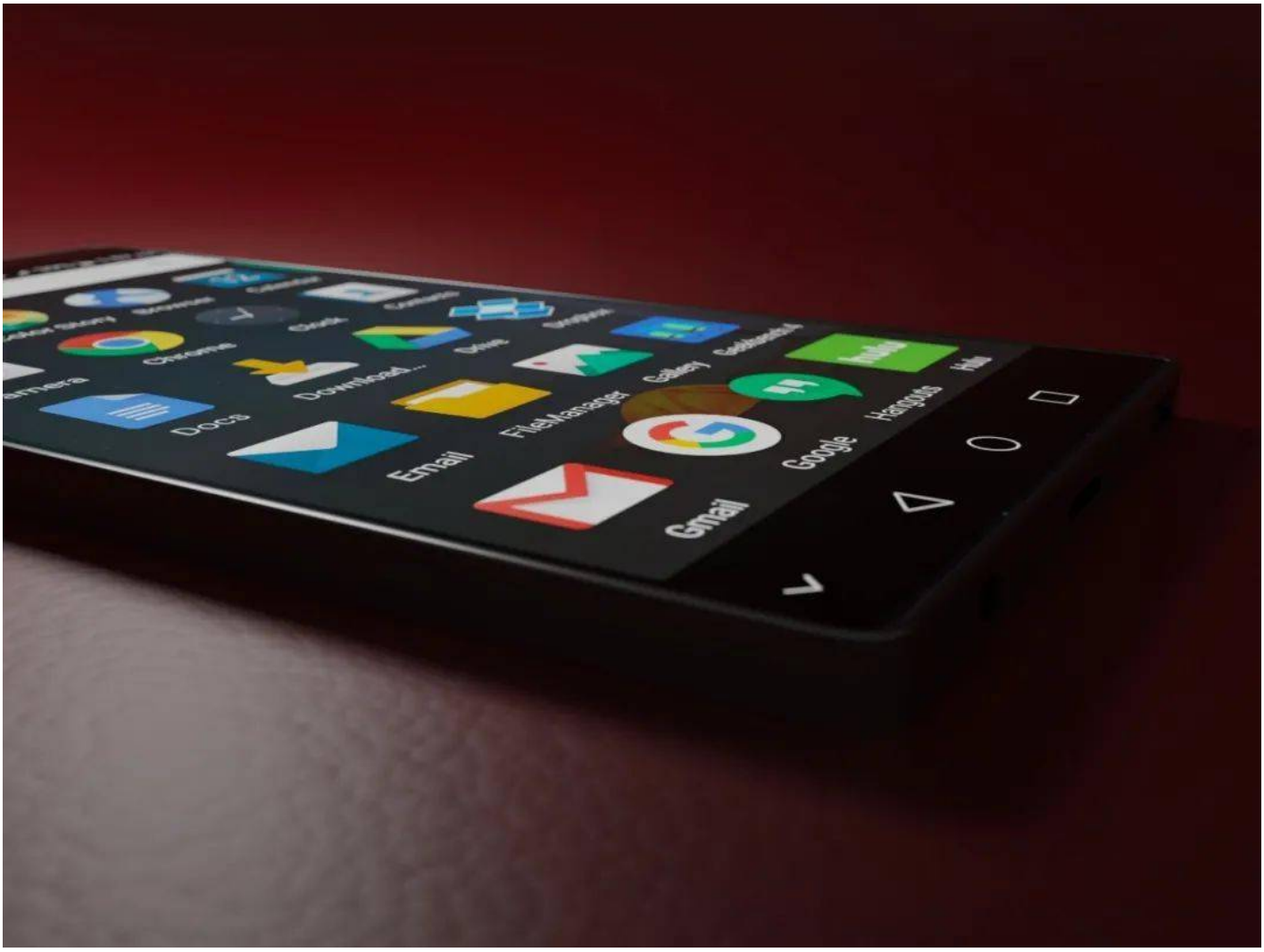
android8.1内核编译,Pixel 2 刷8.0.0/8.1.0 AOSP +4.4 Kernel

转载

weixin_39801714 于 2021-05-26 09:38:05 发布 249 收藏
文章标签： android8.1内核编译
原标题：Pixel 2 刷8.0.0/8.1.0 AOSP +4.4 Kernel



本文为看雪论优秀文章

看雪论坛作者ID：koozxcv

前面这些版本源码资源以及vendor资源大家应该刷过机的都知道怎么去找和下载，网上也有很多比较详细的教程，不赘述。 本文重点对整个编译脚本进行注解，以及解决一部分人遇到的触屏失灵问题。

1. AOSP对应版本：android-8.0.0_r34

2. kernel对应版本：remotes/origin/android-msm-wahoo-4.4-oreo-dr1

3. image+bootloader版本：walleye-opd3.170816.023-factory-f269631e

4. vendor：Pixel 2 binaries for Android 8.0.0 (OPD3.170816.023)5. 刷机脚本：

#!/bin/bash

#内核源码目录

exportKERNEL_PATH=/home/koozxcv/vp/Kernel/kernel/msm

export GCC_PATH=/home/koozxcv/vp/Kernel/AOSP_SOURCE/android-8.0.0_r34/prebuilts/gcc/linux-x86/aarch64/aarch64-linux-android-4.9/bin/aarch64-linux-android-

#编译工具链

exportGCC_PATH=/opt/android-ndk-r15b/toolchains/aarch64-linux-android-4.9/prebuilt/linux-x86_64/bin/aarch64-linux-android-

#AOSP源码目录

exportAOSP_PATH=/home/koozxcv/vp/Kernel/AOSP_SOURCE/android-8.0.0_r34

#生成的目标文件目录

exportTARGET_PATH= $AOSP_PATH/out/target/product/walleye

#

exportDEVICE_PATH= $AOSP_PATH/device/google/wahoo-kernel

exportKERNEL_OUT_PATH= $KERNEL_PATH/arch/arm64/boot

cd$KERNEL_PATH

#删除旧的vmlinux.o和vmlinux，当然是如果存在的情况下

#在修改了kernel源码后，KERNEL_PATH下的vmlinux.o和vmlinux文件就会发生#改变，而 $KERNEL_PATH/arch/arm64/boot下的 Image Image.gz #Image.gz-dtb都是根据vmlinux拷贝或者压缩而来的，

#具体细节查看$KERNEL_PATH/arch/arm64/boot/Makefile，当vmlinux发生##改变的时候，会接连引起Image Image.gz Image.gz-dtb、Image-dtb的重新生成。

rm vmlinux.o

rm vmlinux

sleep 2

#######交叉编译内核########

#交叉编译-指定编译目标平台架构是arm64

exportARCH=arm64

#交叉编译-指定交叉编译工具链位置

exportexportCROSS_COMPILE= $GCC_PATH

#交叉编译-指定wahoo_defconfig配置文件

make -j$(nproc --all) wahoo_defconfig

make -j$(nproc --all)

#关于Image.lz4-dtb下面会做简单介绍，设备树在内核中是个比较重要的概念

cp $KERNEL_OUT_PATH/Image.lz4-dtb $DEVICE_PATH/

#编译完dtbo.img后拷贝到源码下的/out/target/product/walleye目录

cp $KERNEL_OUT_PATH/dtbo.img $TARGET_PATH/

cd$TARGET_PATH

rm boot.img

rm kernel

#########编译AOSP################

cd$AOSP_PATH

source$AOSP_PATH/build/envsetup.sh

lunch 29

make -j$(nproc --all) dtc mktimg

#当拷贝dtbo.img和Image.lz4-dtb到目标目录后，这里才可以正常编译bootimage

make -B bootimage和dtbo.img

cd$TARGET_PATH

#####刷入编译好的boot.img和dtbo.img#####

adb reboot bootloader

echo"waiting device"

sleep 8

fastboot flash boot boot.img

fastboot flash dtbo_a dtbo.img

fastboot reboot

6. 刷机脚本重的一些关键知识

Image.lz4-dtb

Image.lz4-dtb是内核的设备树描述文件，通过此文件编译内核时不会编译冗余的代码，所以要编译某个版本的内核时，应该先将生产的设备树文件拷贝到内核编译前查询的目录：

cp$KERNEL_OUT_PATH/Image.lz4-dtb $DEVICE_PATH/

Image.lz4-dtb是Linus Torvalds在2011年3月17日的ARM Linux邮件列表宣称“this whole ARM thing is a f*cking pain in the ass”，引发ARM Linux社区的地震，随后ARM社区进行了一系列的重大修正。

在过去的ARM Linux中，arch/arm/plat-xxx和arch/arm/mach-xxx中充斥着大量的垃圾代码，相当多数的代码只是在描述板级细节，而这些板级细节对于内核来讲，不过是垃圾，如板上的platform设备、resource、i2c_board_info、spi_board_info以及各种硬件的platform_data。读者有兴趣可以统计下常见的s3c2410、s3c6410等板级目录，代码量在数万行。

社区必须改变这种局面，于是PowerPC等其他体系架构下已经使用的Flattened Device Tree(FDT)进入ARM社区的视野。Device Tree是一种描述硬件的数据结构，它起源于 OpenFirmware (OF)。在Linux 2.6中，ARM架构的板极硬件细节过多地被硬编码在arch/arm/plat-xxx和arch/arm/mach-xxx，采用Device Tree后，许多硬件的细节可以直接透过它传递给Linux，而不再需要在kernel中进行大量的冗余编码。

可能出现的问题(我自己遇到的问题，坑的一头雾水那种)

如果每次更换内核版本的时候不能直接在msm下git checkoout 版本来切换，因为会有很多已经生成的文件在下一次编译的时候不会再进行编译，会出错，

一种比较保险的方式就是，git checkout .丢弃本地没有提交的所有修改，返回到最初版本。

此外，另一种最保险的方法是重新

git clonehttps: //android.googlesource.com/kernel/msm

然后checkout到干净的分支，再进行编译。优点是之前一些找不到原因的问题可能会在下次正常操作后不再出现，但是缺点是浪费时间。

建议

脚本中编译内核部分没有对编译内核失败进行报错，因此首次编译内核生成dtb文件的时候最好手动编译。或者单独写个脚本编译；

7. 触屏失灵原因及解决办法：

本文重点解决的大家比较关心的安装大多数网上方法刷完机后触屏失灵问题：

pixel 2的刷机和别的google系列的手机差别并不大，不过值得注意的是如果只替换boot.img，会导致有几个htc驱动的version magic和内核的version magic不一致，从而使得这几个驱动在boot时，不能被insmod,所以刷完之后就会有触摸屏幕没有反应的状况。 所以，在pixel 2刷机之前，首先要对内核做一些修改，去掉这个version magic的校验机制。

方法一、去除内核校验函数

1 .在walleye_defconfig里面，注释掉：

ONFIG_MODVERSIONS=y

CONFIG_MODULE_SRCVERSION_ALL=y

2 .在kernel/module.c文件里的check_modinfo函数，注释掉返回失败的选项：

staticintcheck_modinfo(structmodule *mod, structload_info *info, intfla

gs)

{

constchar*modmagic = get_modinfo(info, "vermagic");

interr;

if(flags & MODULE_INIT_IGNORE_VERMAGIC)

modmagic = NULL;

/* This is allowed: modprobe --force will invalidate it. */

if(!modmagic) {

err = try_to_force_load(mod, "bad vermagic");

if(err)

returnerr;

} elseif(!same_magic(modmagic, vermagic, info->index.vers)) {

pr_err("%s: version magic '%s' should be '%s'n",

mod->name, modmagic, vermagic);

//return -ENOEXEC; 【这里注释掉】

}

if(!get_modinfo(info, "intree"))

add_taint_module(mod, TAINT_OOT_MODULE, LOCKDEP_STILL_OK);

if(get_modinfo(info, "staging")) {

add_taint_module(mod, TAINT_CRAP, LOCKDEP_STILL_OK);

pr_warn("%s: module is from the staging directory, the quality "

"is unknown, you have been warned.n", mod->name);

}

/* Set up license info based on the info section */

set_license(mod, get_modinfo(info, "license"));

return0;

}

方法二、改掉自身编译的内核的version magic, 使得和原本userdebug 版本相匹配

1. 在include/generated/utsrelease.h 里将UTS_RELEASE替换成自己的userdebug版本的uts_release号(可通过uname -a查看)

defineUTS_RELEASE "4.4.56-g84c2982-dirty"

//替换后：

defineUTS_RELEASE "4.4.56-g594d847d09a1"

2. 在s/setlocalversion里，在scm_version函数里加上：

scm_version

{

localshort

short= false

++ printf"4.4.56-g594d847d09a1"

++ return

cd" $srctree"

iftest-e .scmversion; then

cat .scmversion

return

fi

3. 在module.c里，注释掉check_verion函数：

staticintcheck_version(Elf_Shdr *sechdrs,

unsignedintversindex,

constchar*symname,

struct module*mod,

constunsignedlong*crc,

conststruct module*crc_owner)

{

unsignedinti, num_versions;

structmodversion_info* versions;

++ return1;

/* Exporting module didn't supply crcs? OK, we're already tainted. */

if(!crc)

return1;

/* No versions at all? modprobe --force does this. */

if(versindex == 0)

returntry_to_force_load(mod, symname) == 0;

上述两种方法都可以解决我们要修改的的目标不是htc的那几个驱动,如果目标是那几个驱动怎么办呢？

最好的办法是能够自己解包然后打包vendor.img，然后把自己编译出来的htc驱动包进去，这是完美的方案，但是由于解包打包vendor.img没有现成的方案也就不这样做了(哈哈哈哈哈，喷我吧)。好在，我们还可以自己insmod:

这种方法就不要更改内核所有的签名版本config之类的，就要它不能insmod原来的驱动。然后我们把自己编译的驱动adb push 进去，再手动insmod,注意下顺序：

insmod synaptics_dsx_core_htc.ko

insmod synaptics_dsx_fw_update_htc.ko

insmod synaptics_dsx_rmi_dev_htc.ko

insmod htc_battery.ko

最后：

我自己在实践中发现pixel 2刷其他版本时也遇到类似的问题，同样的方式解决就行。nexus 6p不存在这样的问题。

看雪ID：koozxcv

*本文由看雪论坛 koozxcv 原创，转载请注明来自看雪社区。返回搜狐，查看更多

责任编辑：