系统产品部 2024 年 3 月 11 日培训资料
主题：AOSP 和 LineageOS 编译、刷机和部分模块讲解
议程：

主讲：韩玮
参加：系统产品部智 UI 产品开发组

AOSP 部分

环境要求

硬件

处理器：2.3.x 以上 Android 版本需要 64 位架构的处理器
内存：官方最低 16G，谷歌推荐 64G；实测虚拟机 32G 以下内存有概率崩溃，物理机没问题，虚拟机建议至少 40G 内存
硬盘：代码 250G，编译再加 150G，建议 500G 以上 SSD
网络：保证网络稳定，否则检出代码很容易中断

参考：虚拟机+i5 10700 + 32G 内存+SSD，检出代码约 4 小时，编译、链接、打包约 3.5 个小时

软件

Ubuntu 22.04 LTS x64
JDK AOSP 自带
Make AOSP 自带
Python 3
sudo apt install git-core gnupg flex bison build-essential zip curl zlib1g-dev libc6-dev-i386 libncurses5 lib32ncurses5-dev x11proto-core-dev libx11-dev lib32z1-dev libgl1-mesa-dev libxml2-utils xsltproc unzip fontconfig

以上都是针对 Android9.0 及以上版本的环境配置

下载代码

清华源：https://mirrors.tuna.tsinghua.edu.cn/help/AOSP/

科大源：https://mirrors.ustc.edu.cn/help/aosp.html
谷歌官方的方法不适合国内，参考
https://source.android.google.cn/docs/setup/download/downloading

方法一：

以下是推荐的国内下载代码方法

“

初始同步方法

第一次同步数据量特别大，如果网络不稳定，中间失败就要从头再来了。所以我们提供了打包的 AOSP 镜像，为一个 tar 包，大约 200G（单文件 200G，注意你的磁盘格式要支持）。这样你 就可以通过 HTTP(S) 的方式下载，该方法支持断点续传。

下载地址 https://mirrors.ustc.edu.cn/aosp-monthly/

请注意对比 checksum。

然后根据下文 ***替换已有的 AOSP 源代码的 remote*** 的方法更改同步地址。

解压后用命令 repo sync 就可以把代码都 checkout 出来。

Note: tar 包为定时从 https://mirrors.tuna.tsinghua.edu.cn/aosp-monthly/ 下载。

”

> ***替换已有的 AOSP 源代码的 remote***
> *如果你之前已经通过某种途径获得了 AOSP 的源码(或者你只是 init 这一步完成后)，你希望以后通过 TUNA 同步 AOSP 部分的代码，只需要修改 .repo/manifests.git/config，将*
> *url = https://android.googlesource.com/platform/manifest*
> *更改为*
> *url = https://mirrors.tuna.tsinghua.edu.cn/git/AOSP/platform/manifest*
> *或者可以不修改文件，而执行*
> *git config --global url.https://mirrors.tuna.tsinghua.edu.cn/git/AOSP/.insteadof https://android.googlesource.com*

方法二：

“

如果想用官方步骤+国内镜像地址的方法，将官方步骤中的 https://android.googlesource.com/ 全部使用 https://mirrors.tuna.tsinghua.edu.cn/git/AOSP/ 代替即可。由于使用 HTTPS 协议更安全，并且更便于我们灵活处理，所以强烈推荐使用 HTTPS 协议同步 AOSP 镜像。由于 AOSP 镜像造成 CPU/内存负载过重，我们限制了并发数量，因此建议：sync 的时候并发数不宜太高，否则会出现 503 错误，即-j 后面的数字不能太大，建议选择 4。请尽量选择流量较小时错峰同步。

”

分支选择
参考：https://source.android.google.cn/setup/start/build-numbers#source-code-tags-and-builds

build ID	标记	版本	支持的设备	安全补丁级别
TQ2A.230505.002.A1	android-13.0.0_r44	Android13	Pixel 4a (5G)、Pixel 5、Pixel 5a (5G)	2023-05-05
TQ2A.230505.002	android-13.0.0_r43	Android13	Pixel 4a、Pixel 4a (5G)、Pixel 5、Pixel 5a (5G)、Pixel 6、Pixel 6a、Pixel 6 Pro、Pixel 7、Pixel 7 Pro	2023-05-05

是可用 AOSP TAG

9873343	android-security-13.0.0_r5	Android13		2022-08-05
9682190	android-security-13.0.0_r4	Android13		2022-08-05

不是可用 AOSP TAG，是补丁 TAG
选择完 REPO 方式和 TAG 后，执行以下命令进行代码同步
repo init -u https://mirrors.tuna.tsinghua.edu.cn/git/AOSP/platform/manifest -b android-14.0.0_r1
repo sync

构建系统(Soong 编译系统)

参考：https://source.android.google.cn/docs/setup/build

Make(.mk)和 Soong(.bp)比较
以下是 Make 配置与 Soong 在 Soong 配置（Blueprint 或 .bp）文件中完成相同操作的比较。

Make 示例

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)
LOCAL_MODULE := libxmlrpc++
LOCAL_MODULE_HOST_OS := linux

LOCAL_RTTI_FLAG := -frtti
LOCAL_CPPFLAGS := -Wall -Werror -fexceptions
LOCAL_EXPORT_C_INCLUDES := $(LOCAL_PATH)/src

LOCAL_SRC_FILES := $(call \
     all-cpp-files-under,src)
include $(BUILD_SHARED_LIBRARY)
```

Soong 示例

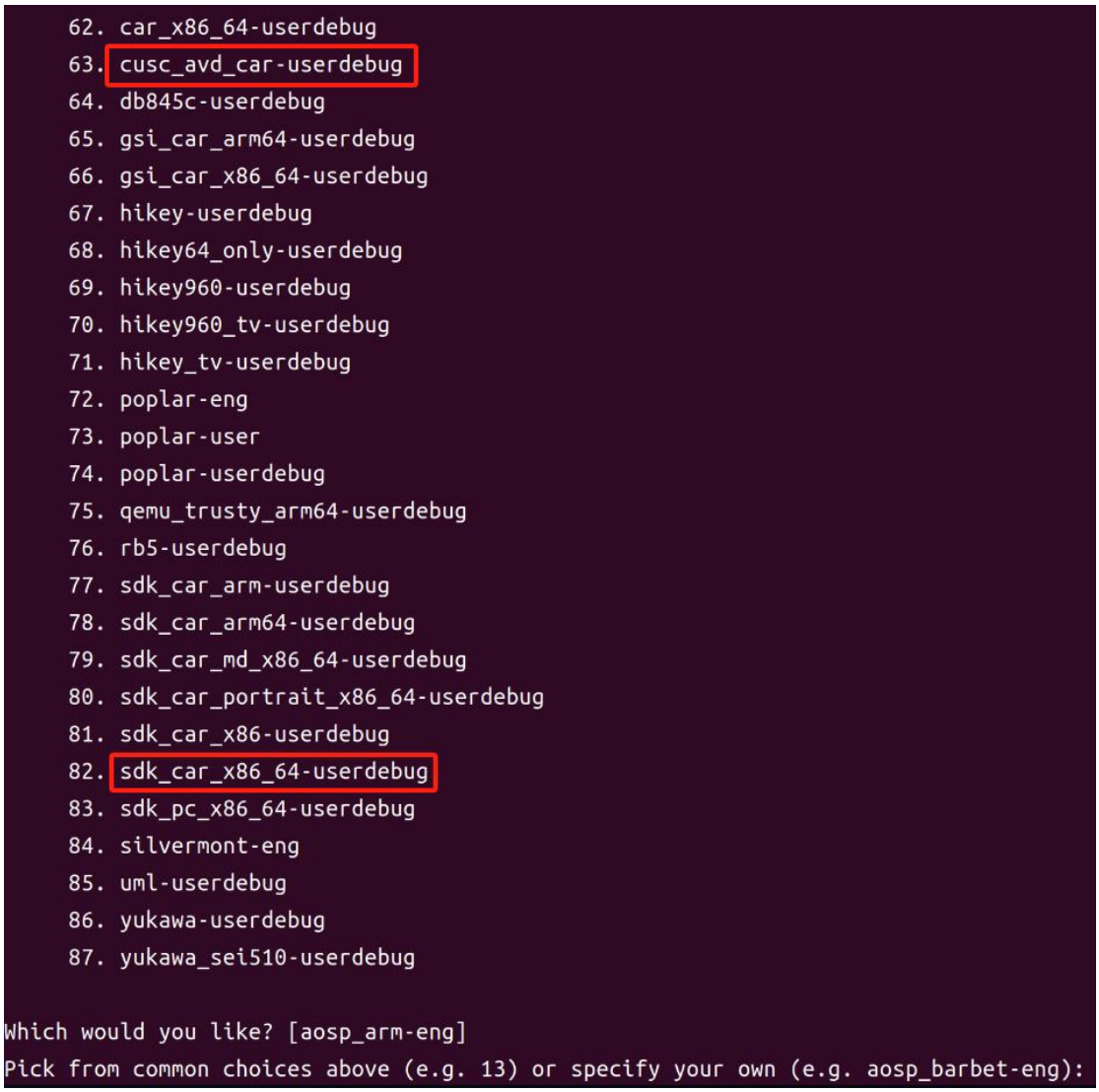
```
cc_library_shared {
     name: "libxmlrpc++",

     rtti: true,
     cppflags: [
          "-Wall",
          "-Werror",
          "-fexceptions",
     ],
     export_include_dirs: ["src"],
     srcs: ["src/**/*.cpp"],

     target: {
          darwin: {
               enabled: false,
          },
     },
}
```

构建 Android

参考：https://source.android.google.cn/docs/setup/build/building
设置环境(通过脚本导入命令)
. build/envsetup.sh
或
source build/envsetup.sh
通过 hmm 查看导入的命令
选择目标
lunch aosp_arm-eng
或
lunch



通过菜单进行选择，菜单中的名称后缀解释如下：

构建类型	使用情况
user	权限受限；适用于生产环境
userdebug	与“user”类似，但具有 root 权限和调试功能；是进行调试时的首选编译类型
eng	具有额外调试工具的开发配置

可以选择的硬件设备参考：
https://source.android.google.cn/docs/setup/build/running?hl=zh-cn#selecting-device-build

编译代码

本部分是一个简短摘要，用于确保设置已完成。

使用 `m` 构建所有内容。`m` 可以使用 `-jN` 参数处理并行任务。如果您没有提供 `-j` 参数，构建系统会自动选择您认为最适合您系统的并行任务计数。

```
$ m
```

如上所述，您可以通过在 `m` 命令行中列出相应名称来构建特定模块，而不是构建完整的设备映像。此外，`m` 还针对各种特殊目的提供了一些伪目标。以下是一些示例：

- `droid` - `m droid` 是正常 build。此目标在此处，因为默认目标需要名称。
- `all` - `m all` 会构建 `m droid` 构建的所有内容，加上不包含 `droid` 标记的所有内容。构建服务器会运行此命令，以确保包含在树中且包含 `Android.mk` 文件的所有元素都会构建。
- `m` - 从树的顶部运行构建系统。这很有用，因为您可以在子目录中运行 `make`。如果您设置了 `TOP` 环境变量，它便会使用此变量。如果您未设置此变量，它便会从当前目录中查找相应的树，以尝试找到树的顶层。您可以通过运行不包含参数的 `m` 来构建整个源代码树，也可以通过指定相应名称来构建特定目标。
- `mma` - 构建当前目录中的所有模块及其依赖项。
- `mmma` - 构建提供的目录中的所有模块及其依赖项。
- `croot` - `cd` 到树顶部。
- `clean` - `m clean` 会删除此配置的所有输出和中间文件。此内容与 `rm -rf out/` 相同。

运行 `m help` 即可查看 `m` 提供的其他伪目标。

模拟器启动

emulator &

build 指纹

如需跟踪和报告与特定 Android build 相关的问题，请一定要了解 build 指纹。build 指纹是能让人看懂的唯一字符串，其中包含向每个 build 发出的制造商信息。有关详情，请参阅 Android 兼容性定义文档 (CDD) 中 Build 参数部分内的“FINGERPRINT”说明。
https://source.android.google.cn/docs/compatibility/14/android-14-cdd#322_build_parameters

build 指纹表示特定的 Android 实现和修订版本。此唯一键可让应用开发者和其他人报告与特定固件版本相关的问题。如需了解 Android 问题报告流程，请参阅报告 bug。

build 指纹封装了所有 Android 实现详情：

- API：Android 和原生，以及软 API 行为
- 核心 API 和部分系统界面行为
- CDD 中定义的兼容性和安全性要求
- 应用所采用的产品规范和 uses-feature 设置，用于定位符合预期要求的设备

- 硬件和软件组件的实现

如需查看完整详情，请参阅 CDD。如需有关打造全新 Android 设备的说明，请参阅添加新设备。

编译内核

设备内核+通用内核
参考：https://source.android.google.cn/docs/setup/build/building-kernels
repo init -u https://android.googlesource.com/kernel/manifest -b BRANCH
以下是 BRANCH 可选的分支名称(最右边)

设备	AOSP 树中的二进制文件路径	Repo 分支
Pixel 4 (flame) Pixel 4 XL (coral)	device/google/coral-kernel	android-msm-coral-4.14-android13
Pixel 3a (sargo) Pixel 3a XL (bonito)	device/google/bonito-kernel	android-msm-bonito-4.9-android12L
Pixel 3 (blueline) Pixel 3 XL (crosshatch)	device/google/crosshatch-kernel	android-msm-crosshatch-4.9-android12
Pixel 2 (walleye) Pixel 2 XL (taimen)	device/google/wahoo-kernel	android-msm-wahoo-4.4-android10-qpr3
Pixel (sailfish) Pixel XL (marlin)	device/google/marlin-kernel	android-msm-marlin-3.18-pie-qpr2
Hikey960	device/linaro/hikey-kernel	hikey-linaro-android-4.14 hikey-linaro-android-4.19 common-android12-5.4 common-android13-5.10
Beagle x15	device/ti/beagle_x15-kernel	omap-beagle-x15-android-4.14 omap-beagle-x15-android-4.19
Android 通用内核	N/A	common-android-4.4 common-android-4.9 common-android-4.14 common-android-4.19 common-android-4.19-stable common-android11-5.4 common-android12-5.4 common-android12-5.10 common-android13-5.10 common-android13-5.15 common-android14-5.15 common-android14-6.1 common-android-mainline

以下是构建方法

Android 13 之前
build/build.sh
Android 13 及以后
tools/bazel build //common:kernel_aarch64_dist
tools/bazel run //common:kernel_aarch64_dist -- --dist_dir=$DIST_DIR

以下是一些 Tips：

1. Android 11 引入了 GKI，用于将内核拆分为由 Google 维护的内核映像和由供应商维护的模块，二者分别单独构建。
2. 在 Android 12 中，Cuttlefish 和 Goldfish 融合，因此它们共享同一个内核：virtual_device。
3. Android 13 引入了使用 Bazel (Kleaf) 构建内核的功能，以取代 build.sh。

如需详细了解如何使用 Bazel 构建 Android 内核，请参阅：Kleaf - 使用 Bazel 构建 Android 内核。https://android.googlesource.com/kernel/build/+/refs/heads/master/kleaf/README.md

对设备和内核的 Kleaf 支持

下表列出了对各个设备内核的 Kleaf 支持。对于未列出的设备，请与设备制造商联系。

设备	Repo 分支	Kleaf 支持	build/build.sh 支持
Android 通用内核 db845c 虚拟设备（x86_64、arm64） 虚拟设备（i686、arm） Rockpi4	common-android-4.4 common-android-4.9 common-android-4.14 common-android-4.19 common-android-4.19-stable common-android11-5.4 common-android12-5.4 common-android12-5.10	✖	✅
Android 通用内核	common-android13-5.10 common-android13-5.15	✅	✅（官方）[1]
Android 通用内核	common-android14-5.15 common-android14-6.1 common-android-mainline	✅	✖

虚拟设备（x86_64、arm64）	common-android13-5.10 common-android13-5.15	✅（官方）[1]	⚠（未维护）[2]
虚拟设备（x86_64、arm64）	common-android14-5.15 common-android14-6.1 common-android-mainline	✅	❌
虚拟设备（i686、arm）	common-android13-5.10 common-android13-5.15	❌	✅
虚拟设备（i686、arm）	common-android14-5.15 common-android14-6.1 common-android-mainline	✅	❌

有关 Android 通用内核喝 GKI 解释，参考：
https://source.android.google.cn/docs/core/architecture/kernel/android-common

功能和启动内核

每个 Android 平台版本都支持启动基于三个 Linux 内核版本中任意一个的新设备。如下表所示，Android 11 的启动内核为 android-4.14-stable 、android-4.19-stable 和 android11-5.4 。由于更新平台版本时通常不需要升级内核，因此缺少平台版本最新功能的内核仍可用于启动设备。因此，即使将平台版本升级到 Android 11 后，专为 Android 10 设计的内核（例如 android-4.19-q ）也可以在设备上使用。从 Android 12 开始，功能内核将少于启动内核，以限制功能内核的数量必须支持的稳定 KMI。

Android平台发布	启动内核	特征内核
安卓 14 (2023)	android14-6.1 android14-5.15 android13-5.15 android13-5.10 android12-5.10 [1]	android14-6.1 android14-5.15
安卓 13 (2022)	android13-5.15 android13-5.10 android12-5.10 [1] android12-5.4 [1] android11-5.4 [1]	android13-5.15 android13-5.10
安卓 12 (2021)	android12-5.10 android12-5.4 android11-5.4 [1] android-4.19-stable	android12-5.10 android12-5.4
安卓 11 (2020)	android11-5.4 android-4.19-stable android-4.14-stable	android11-5.4 android-4.19-stable android-4.14-stable
安卓 10 (2019)	android-4.19-q android-4.14-q android-4.9-q	android-4.19-q android-4.14-q android-4.9-q

[1]如果相关 BSP 已针对平台版本进行更新，则可能适用其他限制。更一般而言，内核的Android 版本号必须高于或等于目标 FCM 版本。有关详细信息，请参阅供应商接口对象 - 匹配内核分支。

有关 Android 操作系统核心主题，包括架构、内核、HAL 层等开发文档，可以参考：
https://source.android.google.cn/docs/core

有关 Android automotive 的主题，可以参考：
https://source.android.google.cn/docs/automotive
Android 14 Automotive 远程唤醒工作流示例
https://source.android.google.cn/docs/automotive/remote_access

Build LineageOS for Xiaomi Mi MIX 2S

LineageOS 介绍

LineageOS（也称 Lineage OS、Lineage OS Android Distribution）是一个面向智能手机和平板电脑的自由、免费、开放源代码的 Android 系统分支。它是深受欢迎的定制 ROM CyanogenMod 的继任者。它在 2016 年 12 月 Cyanogen 公司突然宣布停止开发并关闭项目基础设施后复刻而生。LineageOS 于 2016 年 12 月 24 日正式启动，其源代码存放于 GitHub。

由于 Cyanogen 公司保留了使用 Cyanogen 名称的权利，因此项目复刻后更名为 LineageOS。

目前 LineageOS 最新版本是 21，基于 AOSP 14. 参考 https://www.lineageos.org/Changelog-28/

LineageOS 的思路是，内核使用手机厂商或者 SoC 供应商提供的开源内核版本，或支持的二进制版本，系统源码使用 AOSP，应用层自行开发的模式。所以源代码分为三个部分：

1. AOSP 源码，谷歌 OpenSource 托管
2. LineageOS APP 源码，Github 托管
3. Kernel 源码或二进制模组，Github 托管，或使用脚本从手机端直接拉取

基于这个模式，LineageOS 支持的设备是有限的，谷歌手机因为内核开源，所以全系列支持，其它手机厂根据不同机型支持的情况也不同，参见：https://wiki.lineageos.org/devices/

本次以小米 Mix 2s(代号 Polaris)为例讲解
参考：https://wiki.lineageos.org/devices/polaris/build/

环境要求

1. Mix 2s 手机一台(可解锁)
2. 编译软硬件环境和 AOSP 要求差不多，按照之前 AOSP 的要求准备即可
3. 因为涉及到编译版本的刷机操作，所以需要准备 platform-tools 环境，可以安装 Android Studio 并配置 SDK，也可以只下载 platform-tools，地址 https://dl.google.com/android/repository/platform-tools-latest-linux.zip，下载后配置环境变量：

```
unzip platform-tools-latest-linux.zip -d ~
```

在~/.profile 文件中添加以下代码：

```
# add Android SDK platform tools to path
if [ -d "$HOME/platform-tools" ] ; then
    PATH="$HOME/platform-tools:$PATH"
fi
```

运行 source ~/.profile 导入环境变量配置

4. 安装编译需要的库

sudo apt install bc bison build-essential ccache curl flex g++-multilib gcc-multilib git git-lfs gnupg gperf imagemagick lib32readline-dev lib32z1-dev libelf-dev liblz4-tool libsdl1.2-dev libssl-dev libxml2 libxml2-utils lzop pngcrush rsync schedtool squashfs-tools xsltproc zip zlib1g-dev
如果是低于 Ubuntu 23.10，还需要安装 lib32ncurses5-dev libncurses5 libncurses5-dev
5. Java 要求，如果是 LineageOS 18.1+版本，源码自带 JDK，无需自行安装
6. 如果是 LineageOS 17.1+，还需要安装 Python 3 (sudo apt install python-is-python3)

检出代码

7. 创建安装目录

```
mkdir -p ~/bin
mkdir -p ~/android/lineage
```

8. 下载 repo

```
curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
chmod a+x ~/bin/repo
```

在~/.profile 中添加以下代码

```
# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi
```

运行 source ~/.profile 导入环境变量配置

9. 配置 git

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
git lfs install
```

10. 配置编译缓存(可选)

```
export USE_CCACHE=1
export CCACHE_EXEC=/usr/bin/ccache
ccache -o compression=true
```

11. 初始化 LineageOS 源代码仓库(以 LineageOS 21 为例)

```
cd ~/android/lineage
repo init -u https://github.com/LineageOS/android.git -b lineage-21.0 --git-lfs
repo sync
```

注意：由于 LineageOS 的源代码包含 AOSP，所以在 repo init 之后，sync 之前做一点修改
进入到源码根目录/.repo/manifests，打开 default.xml，找到 remote name="aosp"节点，修改 fetch="https://mirrors.tuna.tsinghua.edu.cn/git/AOSP"，保存退出后再运行 repo sync.
经过漫长的等待，就会下载完源代码。一定要保证 github 访问稳定，否则会导致代码不完整甚至下载中断，可通过多次运行 repo sync 来保证代码完整性

12. 准备设备特殊代码

在源代码根目录运行以下命令

```
source build/envsetup.sh
breakfast polaris
```

这会下载 mix 2s 的特殊设备配置文件和内核
如果你的设备已经刷过 LineageOS，可以使用设备获取这些文件。
进入~/android/lineage/device/xiaomi/polaris 目录，接上手机并开启 adb 调试模式，运行

```
./extract-files.sh
```

脚本会从手机上直接获取所需的配置文件和内核。

编译代码

13. 编译
运行

```
croot
brunch polaris
```

等待编译结束，时间与 AOSP 的编译时间相当

刷机步骤

14. 刷机
进入到 out 目录，找到两个文件
recovery.img // 这是 LineageOS 的 recovery.
lineage-21.0-20240305-UNOFFICIAL-polaris.zip // 这是 lineage 的安装包
a) 解锁手机
b) 手机开启 adb 调试模式，adb -d reboot bootloader 进入到 bootloader。也可以通过同时按住音量键下+电源键进入。进入到 bootloader 后使用 fastboot devices 查看设备是否识别
c) 刷入编译好的 recovery：fastboot flash recovery recovery.img . 刷入后，不要直接重启进入到系统，而是按住音量上+电源键进入到 recovery，如果不是 LineageOS 的图标，则需要重复 b 步骤后再次刷入 recovery，这是因为 Android 目前使用 A/B 分区的方式保证升级的安全性，所以存在两个 recovery 分区，可能需要刷两次同时覆盖两个分区。
d) 进入到 LineageOS 后，三清，然后进入 apply update from adb, 进入到 sideload 线刷模式
e) 在电脑端输入 adb -d sideload lineage-21.0-20240305-UNOFFICIAL-polaris.zip ，等待 100% 后刷入成功，重启进入系统。系统刷完
f) 如果要刷入 GMS，则在 e 步骤刷完 zip 后继续刷入其它 GMS 包，然后再重启即可。

Android Input 子系统参考文章

Android Input 子系统-含实例源码
https://www.cnblogs.com/weiqifa/p/9604149.html

input 子系统——kernel 中 input 设备介绍
https://blog.csdn.net/u013604527/article/details/53432623/

android 物理按键
https://blog.csdn.net/xubin341719/article/details/7881735

Android UpdateEngine 子系统参考资料

OTA 更新概览
https://source.android.google.cn/docs/core/ota?hl=zh-cn

LineageOS Updater APP
https://github.com/LineageOS/android_packages_apps_Updater