一、概述

AOSP(Android Open Source Project) 是 Google 开放的 Android 开源项目，中文官网。

AOSP 通俗来讲就是一个 Android 系统源码项目，通过它可以定制 Android 操作系统，国内手机厂商都是在此基础上开发的定制系统。因为墙的缘故，如果无法连接谷歌服务器获取 AOSP 源码，可以从 清华大学镜像站 或者 中科大镜像 获取。

本篇文章以清华大学镜像站为例。

二、配置要求

- 使用 Linux 系统（本文中使用 Ubuntu 20.04）
- 电脑内存16G以上（官方要求），实测 8G 以上可以成功
- 如果要检出代码，至少需要 250 GB 可用磁盘空间；如果要进行构建，则还需要 150 GB。如果要进行多次构建，则需要更多空间。

> 自 2021 年 6 月 22 日起，不再支持在 Windows 或 MacOS 上进行构建。

三、环境准备

3.1 GIT

因为源码是用Git管理的，所以首先需要安装 Git。

在终端中输入：

```
sudo apt-get install git
```

设置 git 账户和邮箱

```
git config --global user.email "xxx@xx.com"
git config --global user.name "xxx"
```

3.2 CURL

安装 CURL

```
sudo apt-get install curl
```

3.3 Repo

Android 源码包含数百个 Git 库，光是下载这么多的 Git 库就是一项繁重的任务，所以 Google 开发了 Repo，它是用于管理 Android 版本库的一个工具，使用了 Python 对 Git 进行了一定的封装，简化了对多个 Git 版本库的管理。

首先创建一个 bin 目录，并加入到 PATH 中。后续步骤会用到。

```
mkdir ~/bin
PATH=~/bin:$PATH
```

安装 Repo

> sudo apt-get install repo

下载repo并设置权限：

> curl https://mirrors.tuna.tsinghua.edu.cn/git/git-repo > ~/bin/repo
> chmod a+x ~/bin/repo

安装完成后，通过以下命令可以验证是否安装成功：

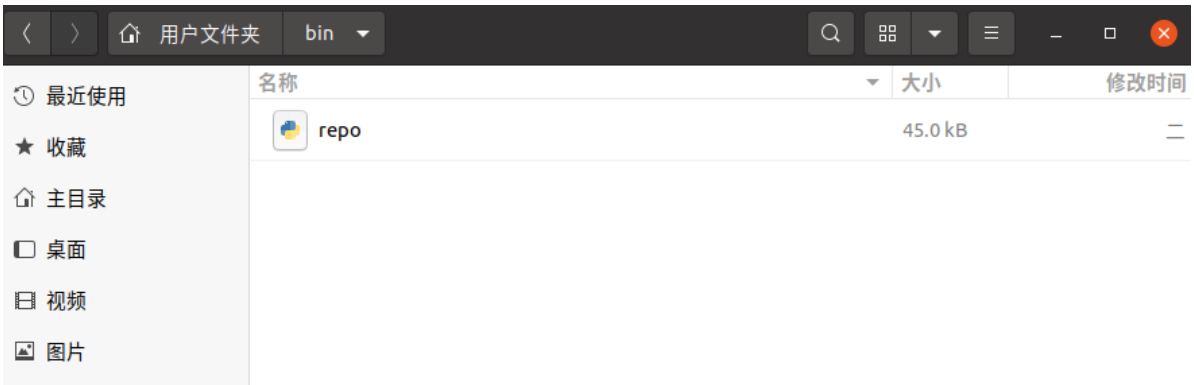
> repo version

如果安装成功，会出现以下信息：

```
chensen@ubuntu:~$ repo version
<repo not installed>
repo launcher version 2.17
       (from /home/chensen/bin/repo)
git 2.25.1
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0]
OS Linux 5.11.0-44-generic (#48~20.04.2-Ubuntu SMP Tue Dec 14 15:36:44 UTC 2021)
CPU x86_64 (x86_64)
Bug reports: https://bugs.chromium.org/p/gerrit/issues/entry?template=Repo+tool+
issue
```

- 如果报告的 repo launcher version 编号为 2.15 或更高，则表明版本号正确，安装无误。
- (from /usr/bin/repo) 表明是通过软件包进行安装的。
- (from /home/<>/bin/repo) 表明是手动安装的

打开刚刚创建的bin目录下，可以看到repo文件：



##3.4 安装 openjdk-8

> sudo apt-get install openjdk-8-jdk

安装 Android Studio

Android Studio 下载地址

将下载的文件解压缩到相应位置。例如 /usr/local/ 中（用于用户个人资料）或者 /opt/ 中（用于共享用户）。

解压完成后，启动 Android Studio。打开一个终端，转到 android-studio/bin/ 目录，执行命令 ./studio.sh

四、下载源码

4.1 建立工作目录

在本地创建一个文件夹，用于存放源码

> mkdir aosp
> cd aosp

4.2 初始化仓库

repo 的运行过程中会尝试访问官方的 git 源更新自己，如果想使用 tuna 的镜像源进行更新，可以将如下内容复制到你的~/.bashrc里：

> export REPO_URL='https://mirrors.tuna.tsinghua.edu.cn/git/git-repo/'

设置身份，添加自己的邮箱和姓名：

> git config --global user.email "xxx@xx.com"
> git config --global user.name "name"

初始化仓库：

> repo init -u https://aosp.tuna.tsinghua.edu.cn/platform/manifest

```
chensen@ubuntu:~/Project/AOSP$ repo init -u https://mirrors.tuna.tsinghua.edu.cn/git/AOSP/platform
/manifest
Downloading Repo source from https://mirrors.tuna.tsinghua.edu.cn/git/git-repo/
remote: Enumerating objects: 7116, done.
remote: Counting objects: 100% (7116/7116), done.
remote: Compressing objects: 100% (3730/3730), done.
remote: Total 7116 (delta 4580), reused 5467 (delta 3312)
接收对象中: 100% (7116/7116), 3.13 MiB | 867.00 KiB/s, 完成.
处理 delta 中: 100% (4580/4580), 完成.
Downloading manifest from https://mirrors.tuna.tsinghua.edu.cn/git/AOSP/platform/manifest
remote: Enumerating objects: 93926, done.
remote: Counting objects: 100% (93926/93926), done.
remote: Compressing objects: 100% (39198/39198), done.
remote: Total 93926 (delta 36011), reused 91379 (delta 34542)
接收对象中: 100% (93926/93926), 22.67 MiB | 734.00 KiB/s, 完成.
处理 delta 中: 100% (36011/36011), 完成.

Your identity is: smashing <smashing_chen@163.com>
If you want to change this, please re-run 'repo init' with --config-name

Testing colorized output (for 'repo diff', 'repo status'):
  black    red      green    yellow   blue     magenta  cyan     white
  bold     dim      ul       reverse
Enable color display in this user account (y/N)? y

repo has been initialized in /home/chensen/Project/AOSP
```

初始化并指定Android版本

> repo init -u https://aosp.tuna.tsinghua.edu.cn/platform/manifest -b android-9.0.0_r10

初始化完成后，目录下会有一个隐藏文件夹.repo, 切换 到.repo/manifests目录，执行 git branch -a 可以看到所有分支

repo init -u https://mirrors.tuna.tsinghua.edu.cn/git/AOSP/platform/manifest -b +分支名 便可以下载对应安卓版本的源码,本文使用的P版本，如果下载最新版本代码可忽略

```
chensen@ubuntu:~/Project/AOSP$ repo init -u https://mirrors.tuna.tsinghua.edu.cn/git/AOSP/platform
/manifest -b android-9.0.0_r10
Downloading Repo source from https://mirrors.tuna.tsinghua.edu.cn/git/git-repo/
remote: Enumerating objects: 7116, done.
remote: Counting objects: 100% (7116/7116), done.
remote: Compressing objects: 100% (3730/3730), done.
remote: Total 7116 (delta 4581), reused 5467 (delta 3312)
接收对象中: 100% (7116/7116), 3.13 MiB | 1.04 MiB/s, 完成.
处理 delta 中: 100% (4581/4581), 完成.
Downloading manifest from https://mirrors.tuna.tsinghua.edu.cn/git/AOSP/platform/manifest
remote: Enumerating objects: 93926, done.
remote: Counting objects: 100% (93926/93926), done.
remote: Compressing objects: 100% (39198/39198), done.
remote: Total 93926 (delta 36009), reused 91379 (delta 34542)
接收对象中: 100% (93926/93926), 22.71 MiB | 1018.00 KiB/s, 完成.
处理 delta 中: 100% (36009/36009), 完成.

Your identity is: smashing <smashing_chen@163.com>
If you want to change this, please re-run 'repo init' with --config-name

repo has been initialized in /home/chensen/Project/AOSP
```

4.3 同步源码

> repo sync

```
chensen@ubuntu:~/Project/AOSP$ repo sync
Fetching:  0% (0/668) warming up

Fetching:  1% (8/668) device/generic/goldfishb'  % Total    % Received % Xferd  Average Speed   Time    Time     Time
  Current\n                                    Dload  Upload   Total   Spent    Left  Speed\n\r  0     0    0     0    0     0      0      0 --:--:-- --:--:-
- --:--:--     0\r  0     0    0     0    0     0      0      0 --:--:--  0:00:01 --:--:--     0\r  0     0    0
0    0     0      0      0 --:--:--  0:00:02 --:--:--     0\r  0     0    0     0    0     0      0      0 --:--:--
0:00:03 --:--:--     0\r  0     0    0     0    0     0      0      0 --:--:--  0:00:04 --:--:--     0\r  0     0
0     0    0     0      0      0 --:--:--  0:00:05 --:--:--     0\r  0     0    0     0    0     0      0      0 --:-
-:--  0:00:06 --:--:--     0\r  0     0    0     0    0     0      0      0 --:--:--  0:00:07 --:--:--     0\r  0
 0    0     0    0     0      0      0 --:--:--  0:00:08 --:--:--     0\r  0     0    0     0    0     0      0
0 --:--:--  0:00:09 --:--:--     0\r  0     0    0     0    0     0      0      0 --:--:--  0:00:10 --:--:--     0\r
 0     0    0     0    0     0      0      0 --:--:--  0:00:11 --:--:--     0\r  0     0    0     0    0     0      0
      0 --:--:--  0:00:12 --:--:--     0\r  0     0    0     0    0     0      0      0 --:--:--  0:00:13 --:--:--
  0\r  0     0    0     0    0     0      0      0 --:--:--  0:00:14 --:--:--     0\r  0     0    0     0    0     0
     0      0 --:--:--  0:00:15 --:--:--     0\r  0     0    0     0    0     0      0      0 --:--:--  0:00:16 --:--
:--     0\r  0     0    0     0    0     0      0      0 --:--:--  0:00:17 --:--:--     0\r  0     0    0     0    0
    0      0      0 --:--:--  0:00:18 --:--:--     0\r  0     0    0     0    0     0      0      0 --:--:--  0:00:19
 --:--:--     0curl: (6) Could not resolve host: mirrors.tuna.tsinghua.edu.cn\n'
Fetching: 29% (200/668) platform/external/libpcap
```

然后等待下载完毕：

```
正在更新文件： 100% (8122/8122), 完成.external/tinyxml2正在更新文件：  66% (5361/8122)
Checking out: 68% (459/668) platform/packages/apps/CarrierConfig正在更新文件： 100% (30060/30060), 完成.
正在更新文件： 100% (4277/4277), 完成.packages/apps/MusicFX正在更新文件：  52% (2249/4277)
正在更新文件： 100% (2871/2871), 完成.
正在更新文件： 100% (3541/3541), 完成.frameworks/hardware/interfaces正在更新文件： 100% (3541/3541)
正在更新文件： 100% (1699/1699), 完成.packages/providers/CalendarProvider正在更新文件：  75% (243/323)
正在更新文件： 100% (4427/4427), 完成.
正在更新文件： 100% (323/323), 完成.
正在更新文件： 100% (891/891), 完成.
正在更新文件： 100% (18572/18572), 完成.
正在更新文件： 100% (7584/7584), 完成.system/ca-certificates正在更新文件：  99% (7509/7584)
正在更新文件： 100% (7584/7584), 完成.test/vti/alert正在更新文件：  69% (103/149)
正在更新文件： 100% (149/149), 完成.
正在更新文件： 100% (17/17), 完成.
正在更新文件： 100% (15073/15073), 完成.ols/tradefederation正在更新文件：  70% (10552/15073)
正在更新文件： 100% (33748/33748), 完成.
正在更新文件： 100% (683/683), 完成.
正在更新文件： 100% (763/763), 完成.
正在更新文件： 100% (12188/12188), 完成.
正在更新文件： 100% (16166/16166), 完成.
正在更新文件： 100% (37036/37036), 完成.
Checking out: 95% (636/668) platform/prebuilts/gcc/darwin-x86/aarch64/aarch64-linux-android-4.9正在更新文件：  28% (62
正在更新文件： 100% (2218/2218), 完成.
正在更新文件： 100% (8081/8081), 完成.
正在更新文件： 100% (73/73), 完成.
正在更新文件： 100% (4084/4084), 完成.
正在更新文件： 100% (10157/10157), 完成.
正在更新文件： 100% (1150/1150), 完成.
Checking out: 100% (668/668), done in 2m5.672s
repo sync has finished successfully.
```

源码下载完成后，看到下面的信息可以说明下载成功。整个源码的大小为36.5 g

如果没有指定版本，如何知道下载好的 AOSP 是什么版本?

找到build/make/core/version_defaults.mk文件打开，搜索PLATFORM_SDK_VERSION，找到了PLATFORM_SDK_VERSION := 28，从 SDK 版本可以知道 AOSP 版本是 9.0

```
ifndef PLATFORM_SDK_VERSION
  # This is the canonical definition of the SDK version, which defines
  # the set of APIs and functionality available in the platform.  It
  # is a single integer that increases monotonically as updates to
  # the SDK are released.  It should only be incremented when the APIs for
  # the new release are frozen (so that developers don't write apps against
  # intermediate builds).  During development, this number remains at the
  # SDK version the branch is based on and PLATFORM_VERSION_CODENAME holds
  # the code-name of the new development work.

  # When you change PLATFORM_SDK_VERSION please ensure you also update the
  # corresponding methods for isAtLeast* in the following java file:
  # frameworks/support/compat/gingerbread/android/support/v4/os/BuildCompat.java

  # When you increment the PLATFORM_SDK_VERSION please ensure you also
  # clear out the following text file of all older PLATFORM_VERSION's:
  # cts/tests/tests/os/assets/platform_versions.txt
  PLATFORM_SDK_VERSION := 28
endif
```

五、源码编译

5.1 整编

整编，顾名思义就是编译整个 Android 源码，最终 out 目录会生成几个重要的镜像文件，其中有 system.img、userdata.img、ramdisk.img 等，这些是可以刷机的。

5.1.1 安装编译环境依赖

```
sudo apt-get install -y libx11-dev:i386 libreadline6-dev:i386 libgl1-mesa-dev
g++-multilib
sudo apt-get install -y git flex bison gperf build-essential libncurses5-
dev:i386
sudo apt-get install -y tofrodos python-markdown libxml2-utils xsltproc zlib1g-
dev:i386
sudo apt-get install -y dpkg-dev libsdl1.2-dev libesd0-dev
sudo apt-get install -y git-core gnupg zip curl zlib1g-dev gcc-multilib
sudo apt-get install -y libc6-dev-i386 x11proto-core-dev libx11-dev
sudo apt-get install -y unzip m4 lib32z-dev ccache libssl-dev
sudo apt-get install -y lib32ncurses5-dev
```

这时可能无法安装 libesd0-dev，执行以下操作：

> sudo vim /etc/apt/sources.list

然后添加：

> deb http://kr.archive.ubuntu.com/ubuntu/ xenial main universe
> deb-src http://kr.archive.ubuntu.com/ubuntu/ xenial main universe

然后执行：

> sudo apt update

再次安装

> sudo apt install libesd0-dev

5.1.2 配置编译环境

在终端中输入以下命令：

> source build/envsetup.sh

5.1.3 选择编译目标

在终端中输入以下命令：

> lunch

```
chensen@ubuntu:~/Project/AOSP$ lunch

You're building on Linux

Lunch menu... pick a combo:
     1. aosp_arm-eng
     2. aosp_arm64-eng
     3. aosp_mips-eng
     4. aosp_mips64-eng
     5. aosp_x86-eng
     6. aosp_x86_64-eng
     7. aosp_car_arm-userdebug
     8. aosp_car_arm64-userdebug
     9. aosp_car_x86-userdebug
     10. aosp_car_x86_64-userdebug
     11. mini_emulator_arm64-userdebug
     12. m_e_arm-userdebug
     13. m_e_mips64-eng
     14. m_e_mips-userdebug
     15. mini_emulator_x86_64-userdebug
     16. mini_emulator_x86-userdebug
     17. uml-userdebug
     18. aosp_cf_x86_auto-userdebug
     19. aosp_cf_x86_phone-userdebug
     20. aosp_cf_x86_tablet-userdebug
     21. aosp_cf_x86_tablet_3g-userdebug
     22. aosp_cf_x86_tv-userdebug
     23. aosp_cf_x86_wear-userdebug
     24. aosp_cf_x86_64_auto-userdebug
     25. aosp_cf_x86_64_phone-userdebug
     26. aosp_cf_x86_64_tablet-userdebug
     27. aosp_cf_x86_64_tablet_3g-userdebug
     28. aosp_cf_x86_64_tv-userdebug
     29. aosp_cf_x86_64_wear-userdebug
     30. cf_x86_auto-userdebug
     31. cf_x86_phone-userdebug
     32. cf_x86_tablet-userdebug
     33. cf_x86_tablet_3g-userdebug
     34. cf_x86_tv-userdebug
     35. cf_x86_wear-userdebug
     36. cf_x86_64_auto-userdebug
     37. cf_x86_64_phone-userdebug
     38. cf_x86_64_tablet-userdebug
     39. cf_x86_64_tablet_3g-userdebug
     40. cf_x86_64_tv-userdebug
     41. cf_x86_64_wear-userdebug
     42. aosp_marlin-userdebug
     43. aosp_marlin_svelte-userdebug
     44. aosp_sailfish-userdebug
     45. aosp_walleye-userdebug
     46. aosp_walleye_test-userdebug
```

```
     45. aosp_walleye-userdebug
     46. aosp_walleye_test-userdebug
     47. aosp_taimen-userdebug
     48. hikey-userdebug
     49. hikey64_only-userdebug
     50. hikey960-userdebug

Which would you like? [aosp_arm-eng] 10

============================================
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=9
TARGET_PRODUCT=aosp_car_x86_64
TARGET_BUILD_VARIANT=userdebug
TARGET_BUILD_TYPE=release
TARGET_ARCH=x86_64
TARGET_ARCH_VARIANT=x86_64
TARGET_2ND_ARCH=x86
TARGET_2ND_ARCH_VARIANT=x86_64
HOST_ARCH=x86_64
HOST_2ND_ARCH=x86
HOST_OS=linux
HOST_OS_EXTRA=Linux-5.11.0-43-generic-x86_64-Ubuntu-20.04.3-LTS
HOST_CROSS_OS=windows
HOST_CROSS_ARCH=x86
HOST_CROSS_2ND_ARCH=x86_64
HOST_BUILD_TYPE=release
BUILD_ID=PPR2.181005.003
OUT_DIR=out
============================================
chensen@ubuntu:~/Project/AOSP$
```

编译目标格式说明

编译目标的格式：BUILD-BUILDTYPE，比如上面的 poplar-eng 的 BUILD 是 poplar，BUILDTYPE 是 eng.
什么是 BUILD

BUILD 指的是特定功能的组合的特定名称，即表示编译出的镜像可以运行在什么环境。其中 aosp(Android Open Source Project)代表 Android 开源项目；arm 表示系统是运行在 arm 架构的处理器上，arm64 则是指 64 位 arm 架构处理器，x86 则表示 x86 架构的处理器，更多设备代码和编译目标参考官方文档。
什么是 BUILDTYPE

BUILDTYPE 则指的是编译类型，通常有三种:

```
user：用来正式发布到市场的版本，权限受限，如没有 root 权限，不能 dedug 等。
userdebug：在user版本的基础上开放了 root 权限和 debug 权限。
eng：代表 engineer，开发工程师的版本，拥有最大的权限(root等)，具有额外调试工具的开发配置。
```

如果没有谷歌手机设备，可以选择 arm 或者 x86，我选择了 aosp_x86-eng，编译完后运行模拟器看看，因此这里选择序号 26。

5.1.4 开始编译

通过 make 指令进行代码编译：

```
make -j8
```

其中 -jN 参数表示处理并行任务，通常使用的任务数 N 介于编译时所用计算机上硬件线程数的 1-2 倍之间。

查看计算机上的核心数：

```
cat /proc/cpuinfo | grep processor
processor : 0
processor : 1
processor : 2
processor : 3
```

可看到创建的虚拟机 CPU 核心共有 4 个，那么要实现最快的编译速度，可以使用介于 make -j4 到 make -j8 之间的命令。

最终会在 out/target/product/generic/目录生成了三个重要的镜像文件： system.img、userdata.img、ramdisk.img。

- system.img：系统镜像，里面包含了Android系统主要的目录和文件，通过init.c进行解析并mount挂载到/system目录下。
- userdata.img：用户镜像，是Android系统中存放用户数据的，通过init.c进行解析并mount挂载到/data目录下。
- ramdisk.img：根文件系统镜像，包含一些启动Android系统的重要文件，比如init.rc。

5.2 源码单编

比如我们要编译系统的Settings应用模块，
在AOSP根目录执行：

```
source build/envsetup.sh
lunch 10
```

进入Settings的目录：

```
cd packages/apps/Settings
```

mm编译当前目录下的模块，不编译依赖模块。

```
mm
```

编译成功后会有提示生成文件的存放路径。

除了Settings.odex文件，还会在out/target/product/generic_x86/system/priv-app/Settings目录下生成Settings.apk。

此外还有以下命令可以进行单编：

- mmm：编译指定目录下的模块，不编译它所依赖的其它模块。
- mma：编译当前目录下的模块及其依赖项。
- mmma：编译指定路径下所有模块，并且包含依赖。

如果你修改了源码，想查看生成的APK文件，有两种方式：

- 通过adb push或者adb install 来安装APK。

- 使用make snod命令，重新生成 system.img，运行模拟器查看。

六、运行模拟器

在编译完成之后,就可以通过以下命令运行Android虚拟机了，命令如下:

```
source build/envsetup.sh
lunch 10
emulator
```

如果是在编译完后运行虚拟机，由于之前已经执行过source和lunch命令了，可以直接运行：

```
emulator
```

emulator 还有很多参数，可以用 emulator -help 查看，参数如下：

```
-sysdir <dir> 为模拟器在<dir>目录中搜索系统硬盘镜像
-system <file> 为模拟器从<file>文件中读取初始化系统镜像
-datadir <dir> 设置用户数据写入的目录
-kernel <file> 为模拟器设置使用指定的模拟器内核
-ramdisk <file> 设置内存RAM 镜像文件(默认为<system>/ramdisk.img)
-image <file> 废弃，使用-system <file> 替代
-init-data <file> 设置初始化数据镜像(默认为<system>/userdata.img)
-initdata <file> 和"-init-data <file>"使用方法一致
-data <file> 设置数据镜像(默认为<datadir>/userdata-qemu.img)
-partition-size <size> system/data 分区容量大小(MB)
-cache <file> 设置模拟器缓存分区镜像(默认为零时文件)
-no-cache 禁用缓存分区
-nocache 与"-no-cache"使用方法相同
-sdcard <file> 指定模拟器SDCard 镜像文件(默认为<system>/sdcard.img)
-wipe-data 清除并重置用户数据镜像(从initdata 拷贝)
-avd <name> 指定模拟器使用Android 虚拟设备
-skindir <dir> 设置模拟器皮肤在<dir>目录中搜索皮肤(默认为<system>/skins 目录)
-skin <name> 选择使用给定的皮肤
-no-skin 不适用任何模拟器皮肤
-noskin 使用方法与"-no-skin"相同
-memory <size> 物理RAM 内存大小(MB)
-netspeed <speed> 设置最大网络下载、上传速度
-netdelay <delay> 网络时延模拟
-netfast 禁用网络形态
-tarce <name> 代码配置可用
-show-kernel 显示内核信息
-shell 在当前终端中使用根Shell 命令
-no-jni Dalvik 运行时禁用JNI 检测
-nojni 使用方法与"-no-jni"相同
-logcat <tag> 输出给定tag 的Logcat 信息

-no-audio 禁用音频支持
-noaudio 与"-no-audio"用法相同
-audio <backend> 使用指定的音频backend
-audio-in <backend> 使用指定的输入音频backend
-audoi-out <backend> 使用指定的输出音频backend
-raw-keys 禁用Unicode 键盘翻转图
-radio 重定向无线模式接口到个性化设备
-port <port> 设置控制台使用的TCP 端口
```

```
-ports <consoleport>,<adbport> 设置控制台使用的TCP 端口和ADB 调试桥使用的TCP 端口
-onion <image> 在屏幕上层使用覆盖PNG 图片
-onion-alpha <%age> 指定上层皮肤半透明度
-onion-rotation 0|1|2|3 指定上层皮肤旋转
-scale <scale> 调节模拟器窗口尺寸(三种：1.0-3.0、dpi、auto)
-dpi-device <dpi> 设置设备的resolution (dpi 单位) (默认165)
-http-proxy <proxy> 通过一个HTTP 或HTTPS 代理来创建TCP 连接
-timezone <timezone> 使用给定的时区，而不是主机默认的
-dns-server <server> 在模拟系统上使用给定的DNS 服务
-cpu-delay <cpudelay> 调节CUP 模拟
-no-boot-anim 禁用动画来快速启动
-no-window 禁用图形化窗口显示
-version 显示模拟器版本号
-report-console <socket> 向远程socket 报告控制台端口
-gps <device> 重定向GPS 导航到个性化设备
-keyset <name> 指定按键设置文件名
-shell-serial <device> 根shell 的个性化设备
-old-system 支持旧版本(pre 1.4)系统镜像
-tcpdump <file> 把网络数据包捕获到文件中
-bootchart <timeout> bootcharting 可用
-qemu args.... 向qemu 传递参数
-qemu -h 显示qemu 帮助
-verbose 和"-debug-init"相同
-debug <tags> 可用、禁用调试信息
-debug-<tag> 使指定的调试信息可用
-debug-no-<tag> 禁用指定的调试信息
-help 打印出该帮助文档
-help-<option> 打印出指定option 的帮助文档
-help-disk-images 关于硬盘镜像帮助
-help-keys 支持按钮捆绑(手机快捷键)
-help-debug-tags 显示出-debug <tag>命令中的tag 可选值
-help-char-devices 个性化设备说明
-help-environment 环境变量
-help-keyset-file 指定按键绑定设置文件
-help-virtula-device 虚拟设备管理
```

七、 Android Studio 导入系统源码

7.1 生成AS的项目配置文件

如果你整编过源码，查看out/host/linux-x86/framework/idegen.jar是否存在，如果不存在，进入源码根目录执行如下的命令：

```
source build/envsetup.sh
lunch [选择整编时选择的参数或者数字]
mmm development/tools/idegen/
```

如果没整编过源码，可以直接执行如下命令单编idegen模块：

```
source build/ensetup.sh
make idegen
```

如果出现报错：

```
Command 'make' not found, but can be installed with:
```

则执行以下命令:

```
sudo apt install make
sudo apt install make-guile
```

命令安装 make 成功后，再次运行mmm development/tools/idegen/。

idegen模块编译成功后，会在 out/host/linux-x86/framework目录下生成idegen.jar，执行如下命令：

```
sudo development/tools/idegen/idegen.sh
```

这时会在源码根目录生成android.iml 和 android.ipr （Android Studio 的工程配置文件），这两个文件一般是只读模式，这里建议改成可读可写，否则，在更改一些项目配置的时候可能会出现无法保存的情况。

```
sudo chmod 777 android.iml
sudo chmod 777 android.ipr
```

7.2 配置AS的项目配置文件

由于要将所有源码导入AS会导致第一次加载很慢，可以在android.iml中修改excludeFolder配置，将不需要看的源码排除掉。等源码项目加载完成后，还可以通过AS对Exclude的Module进行调整。如果你的电脑的性能很好，可以不用进行配置。
在android.iml中搜索excludeFolder，在下面加入这些配置。

```
<excludeFolder url="file://$MODULE_DIR$/bionic" />
<excludeFolder url="file://$MODULE_DIR$/bootable" />
<excludeFolder url="file://$MODULE_DIR$/build" />
<excludeFolder url="file://$MODULE_DIR$/cts" />
<excludeFolder url="file://$MODULE_DIR$/dalvik" />
<excludeFolder url="file://$MODULE_DIR$/developers" />
<excludeFolder url="file://$MODULE_DIR$/development" />
<excludeFolder url="file://$MODULE_DIR$/device" />
<excludeFolder url="file://$MODULE_DIR$/docs" />
<excludeFolder url="file://$MODULE_DIR$/external" />
<excludeFolder url="file://$MODULE_DIR$/hardware" />
<excludeFolder url="file://$MODULE_DIR$/kernel" />
<excludeFolder url="file://$MODULE_DIR$/out" />
<excludeFolder url="file://$MODULE_DIR$/pdk" />
<excludeFolder url="file://$MODULE_DIR$/platform_testing" />
<excludeFolder url="file://$MODULE_DIR$/prebuilts" />
<excludeFolder url="file://$MODULE_DIR$/sdk" />
<excludeFolder url="file://$MODULE_DIR$/system" />
<excludeFolder url="file://$MODULE_DIR$/test" />
```

```
<excludeFolder url="file://$MODULE_DIR$/toolchain" />
<excludeFolder url="file://$MODULE_DIR$/tools" />
<excludeFolder url="file://$MODULE_DIR$/.repo" />
```

7.3 导入系统源代码到AS中

在AS安装目录的bin目录下，打开studio64.vmoptions文件，根据自己电脑的实际情况进行设置，这里修改为如下数值：

```
-Xms1024m
-Xmx1024m
```

如果你是在VirtualBox中下载的系统源码，那么将VirtualBox中的系统源码拷贝到共享文件夹中，这样源码就会自动到Windows或者Mac上，如果你不知道如何设置VirtualBox共享文件夹，可以查看Android AOSP基础（一）VirtualBox 安装 Ubuntu这篇文章。
通过AS的Open an existing Android Studio project选项选择android.ipr 就可以导入源码，这里我用了大概7分钟就导入完毕。导入后工程目录切换为Project选项就可以查看源码

遇到的错误

报错一

```
repo init -u https://aosp.tuna.tsinghua.edu.cn/platform/manifest
Downloading Repo source from https://gerrit.googlesource.com/git-repo
fatal: Cannot get https://gerrit.googlesource.com/git-repo/clone.bundle
fatal: error [Errno 111] Connection refused
fatal: cloning the git-repo repository failed, will remove '.repo/repo'
```

由于repo 每次执行的时候都会去更新自己，由于下载地址被墙，导致无法更新而不能继续执行
解决方法：
为repo 设置国内镜像更新地址 --repo-url=https://gerrit-googlesource.lug.ustc.edu.cn/git-repo

》 sudo repo init -u https://mirrors.tuna.tsinghua.edu.cn/git/lineageOS/LineageOS/android.git -b cm-14.1 --repo-url=https://gerrit-googlesource.lug.ustc.edu.cn/git-repo

报错二

执行编译的时候 报错，

> error while loading shared libraries: libtinfo.so.5: cannot open shared object file: No such file or directory

打开虚拟化引擎

报错三

运行模拟器时报错：

```
emulator: ERROR: No initial system image for this configuration!
```

http://liuwangshu.cn/framework/aosp/2-download-aosp.html

https://zhuanlan.zhihu.com/p/68918808

https://blog.csdn.net/ding1145536113/article/details/112060072