编译内核

本文详细介绍了为 Android 设备编译自定义内核 (/devices/architecture/kernel/)的流程。以下说明会逐步指导您如何选择正确的源代码，编译内核，以及将结果嵌入到根据 Android 开源项目 (AOSP) 编译的系统映像中。

您可以使用 Repo (/setup/develop/repo#init) 获取最新的内核源代码，并通过在源代码检出的根目录下运行 **`build/build.sh`** 来编译这些内核源代码，而无需更多配置。

注意：内核源代码检出的根目录包含 **`build/build.sh`**。Android 树仅包含预编译的内核二进制文件。内核树包含内核源代码和用于编译内核的所有工具，包括此脚本。

对于较旧的内核或下文未列出的内核，请参阅有关如何编译旧版内核 (/setup/build/building-kernels-deprecated)的说明。

下载源代码和编译工具

对于最新的内核，可以使用 **`repo`** (/setup/develop/repo#init) 下载源代码、工具链和编译脚本。一些内核（例如 Pixel 3 内核）需要从多个 Git 代码库获取源代码，而其他内核（如通用内核）只需要一个源代码。使用 **`repo`** 方法可确保源代码目录设置正确。

下载相应分支的源代码：

```
repo init -u https://android.googlesource.com/kernel/manifest -b BRANCH
```

```
repo sync
```

下表列出了可通过此方法获取的内核的 ***BRANCH*** 名称：

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-android10-c2f2
Pixel 3a (sargo) Pixel 3a XL (bonito)	device/google/bonito-kernel	android-msm-bonito-4.9-pie-b4s4

Pixel 3 (blueline) Pixel 3 XL (crosshatch)	device/google/crosshatch-kernel	android-msm-crosshatch-4.9-pie-qpr2
Pixel 2 (walleye) Pixel 2 XL (taimen)	device/google/wahoo-kernel	android-msm-wahoo-4.4-pie-qpr2
Pixel (sailfish) Pixel XL (marlin)	device/google/marlin-kernel	android-msm-marlin-3.18-pie-qpr2
Hikey/Hikey960	device/linaro/hikey-kernel	hikey-linaro-android-4.4 hikey-linaro-android-4.9 hikey-linaro-android-4.14 hikey-linaro-android-4.19
Beagle x15	device/ti/beagle_x15-kernel	omap-beagle-x15-android-4.14 omap-beagle-x15-android-4.19
Beagle x15	device/ti/beagle_x15-kernel	omap-beagle-x15-android-4.14 omap-beagle-x15-android-4.19
Android 通用内核	不适用	common-android-4.4 common-android-4.9 common-android-4.14 common-android-4.19 common-android-5.4 common-android-mainline

注意：您可以在一个 Repo 检出中切换不同的分支。通用内核清单（以及大多数其他清单）定义了要完全克隆（非浅克隆）的内核 Git 代码库，这使您能够在这些分支之间快速切换。切换到不同的分支类似于初始化分支；**-u** 参数是可选的。例如，要从现有的 Repo 检出切换到 **common-android-mainline**，请运行以下命令：

$ repo init -b common-android-mainline && repo sync。

编译内核

然后使用以下内容编译内核：

```
build/build.sh
```

注意：通用内核是通用的可自定义内核，因此不会定义默认配置。如需了解如何为通用内核指定编译配置，请参阅自定义内核编译 (#customize-build)。

内核二进制文件、模块和相应的映像位于 **out/*BRANCH*/dist** 目录下。

运行内核

您可以通过多种方式运行自定义编译的内核。下面介绍了几种适合各种开发场景的已知方法。

嵌入到 Android 映像编译中

将 **Image.lz4-dtb** 复制到 AOSP 树中相应的内核二进制位置，然后重新编译启动映像。

或者，您也可以在使用 **make bootimage**（或用于编译启动映像的任何其他 **make** 命令行）时定义 **TARGET_PREBUILT_KERNEL** 变量。所有设备均支持该变量，因为它是通过 **device/common/populate-new-device.sh** 进行设置的。例如：

```
export TARGET_PREBUILT_KERNEL=DIST_DIR/Image.lz4-dtb
```

使用 fastboot 刷新和启动内核

最新的设备具有引导加载程序扩展，可以简化生成和启动启动映像的过程。

要启动内核而不刷新，请运行以下命令：

```
adb reboot bootloader
fastboot boot Image.lz4-dtb
```

使用此方法时，内核实际上并未刷新，因此不会在重新启动时保留。

注意：内核名称因设备而异。要找到内核的正确文件名，请参阅 AOSP 树中的 **device/*VENDOR*/*NAME*-kernel**。

自定义内核编译

编译流程和结果可能会受环境变量的影响。它们中的大多数是可选的，并且每个内核分支都应该具有适当的默认配置。此处列出了最常用的变量。如需完整（且最新）的列表，请参阅 `build/build.sh`。

环境变量	说明	示例
`BUILD_CONFIG`	要从中初始化编译环境的编译配置文件。系统会相对于 Repo 根目录定义具体位置。默认为 `build.config`。必须为通用内核指定此变量。	`BUILD_CONFIG=common/build.config.cuttlefish.x86_64`
`OUT_DIR`	内核编译的基本输出目录。	`OUT_DIR=/path/to/my/out`
`DIST_DIR`	内核分发的基本输出目录。	`OUT_DIR=/path/to/my/dist`
`CC`	替换要使用的编译器。回退至 `build.config` 定义的默认编译器。	`CC=clang`
`SKIP_MRPROPER`	跳过 `make mrproper`	`SKIP_MRPROPER=1`
`SKIP_DEFCONFIG`	跳过 `make defconfig`	`SKIP_DEFCONFIG=1`

本地编译的自定义内核配置

如果您需要定期切换内核配置选项（例如，在开发某项功能时），或者需要设置一个用于开发用途的选项，可以通过维护编译配置的本地修改或副本来实现这种灵活性。

将 ***POST_DEFCONFIG_CMDS*** 变量设为一个在常规 `make defconfig` 步骤完成后立即接受评估的语句。由于 `build.config` 文件源于编译环境，因此 `build.config` 中定义的函数可以作为 post-defconfig 命令的一部分进行调用。

一个常见示例是在开发期间针对 crosshatch 内核停用链接时优化 (LTO)。虽然 LTO 对已发布的内核有益，但编译时产生的开销可能巨大。添加到本地 **build.config** 的以下代码段将在使用 **build/build.sh** 时永久停用 LTO。

```
POST_DEFCONFIG_CMDS="check_defconfig && update_debug_config"
function update_debug_config() {
    ${KERNEL_DIR}/scripts/config --file ${OUT_DIR}/.config \
         -d LTO \
         -d LTO_CLANG \
         -d CFI \
         -d CFI_PERMISSIVE \
         -d CFI_CLANG
    (cd ${OUT_DIR} && \
     make O=${OUT_DIR} $archsubarch CC=${CC} CROSS_COMPILE=${CROSS_COMPILI
}
```

确定内核版本

您可以通过多种方式确定要编译的正确版本。

AOSP 树中的内核版本

AOSP 树包含预编译的内核版本。大多数情况下，Git 日志会在提交消息中显示正确的版本：

```
cd $AOSP/device/VENDOR/NAME
git log --max-count=1
```

系统映像中的内核版本

要确定系统映像中使用的内核版本，请对内核文件运行以下命令：

```
file kernel
```

对于 `Image.lz4-dtb` 文件，请运行以下命令：

```
grep -a 'Linux version' Image.lz4-dtb
```


Last updated 2020-02-05.