我想很多初学者或许跟我一样，看完 Android 源码下载相关的文章以后，就开始兴致勃勃地去下载 Android 源码了，但是下载完了源码后，有没有像我一样产生如下几个困惑呢？

（1） Android 版本有哪些分支可用？每个分支的 TAG 是什么？

（2） Android 源码下载完了怎么没有看到 Linux 内核代码？Android 源码对应 Linux 内核是否可以从 kernel.org 官网去下载？Android 对标准的 Linux 内核做了哪些修改？

（3） Android 源码分支与 Linux 版本分支的对应关系是什么样的？高版本的 Android 源码能否使用低版本的 Linux 内核？或者低版本 Android 源码能否使用高版本的 Linux 内核？

（4） 开发板厂商提供的 Android 源码与 AOSP 官网下载的 Android 源码是什么关系呢？

作为我的《Android 内核开发》系列文章的第三篇，本文主要来解答一下上面提到的几个问题。

1. Android 版本有哪些分支可用？每个分支的 TAG 是什么？

Android 官网详细地介绍了当前 Android 的各个版本名称、Version、对应的 API Level、Branch TAG、以及 Supported devices，该链接地址如下：

http://source.android.com/source/build-numbers.html

由于官网被墙，这里我也给出了一份保存下来的离线 html 文档，你可以下载下来用浏览器打开查看，地址如下：

https://github.com/Jhuster/AOSP/tree/master/documents

当然，想查看当前可用的 Android 源码分支和版本，也可以在下载好的 Android 源码根目录下执行如下命令：

```
$ git --git-dir .repo/manifests/.git/ branch -a

或者

$ cd .repo/manifests
$ git branch -a | cut -d / -f 3
```

得到的结果示例如下（只截取了部分）：

那么，如果你想切换到其他 Android 分支，只需要重新执行 repo init 和 repo sync 即可，示例如下：

```
$ repo init -b android-4.2.2_r1
$ repo sync
```

2. Android 源码与 Linux 内核代码的关系？

文章开头提到的第二个问题主要涉及到 Android 源码与 Linux 内核代码的关系，我们首先要了解一个重要的概念：

Android 并没有使用标准的 Linux 内核，而是做了很多的修改。

Android 对标准的 Linux 内核代码做了大量的剪裁和优化，并且添加了许多特有的代码，主要包括：自定义 UI 系统，采用 Bionic Libc 库代替 glibc 库，添加 Gold-Fish 平台，编写专有的驱动程序，如 Binder、Logger、PowerManager 等等。

由 于版权分歧等原因，这些修改并没有 merge 到 Linux 主分支中去，因此，我们不能直接从 Linux Kernel 的官网（kernel.org）去下载适用于 Android 源码的 Linux 内核代码，而是要到 Google 官网提供的 kernel 网址去下 载经过修改后的 Linux 内核代码。

Google 提供了多个版本的 Linux Kernel，分别对应不同的设备或者厂商版本，你可以通过 git clone 命令来完成下载，如下所示：

Depending on which kernel you want,

```
$ git clone https://android.googlesource.com/kernel/common.git
$ git clone https://android.googlesource.com/kernel/x86_64.git
$ git clone https://android.googlesource.com/kernel/exynos.git
$ git clone https://android.googlesource.com/kernel/goldfish.git
$ git clone https://android.googlesource.com/kernel/msm.git
$ git clone https://android.googlesource.com/kernel/omap.git
$ git clone https://android.googlesource.com/kernel/samsung.git
$ git clone https://android.googlesource.com/kernel/tegra.git
```

- The goldfish project contains the kernel sources for the emulated platforms.
- The msm project has the sources for ADP1, ADP2, Nexus One, Nexus 4, Nexus 5, Nexus 6, and can be used as a starting point for work on Qualcomm MSM chipsets.
- The omap project is used for PandaBoard and Galaxy Nexus, and can be used as a starting point for work on TI OMAP chipsets.
- The samsung project is used for Nexus S, and can be used as a starting point for work on Samsung Hummingbird chipsets.
- The tegra project is for Xoom, Nexus 7, Nexus 9, and can be used as a starting point for work on NVIDIA Tegra chipsets.
- The exynos project has the kernel sources for Nexus 10, and can be used as a starting point for work on Samsung Exynos chipsets.
- The x86_64 project has the kernel sources for Nexus Player, and can be used as a starting point for work on Intel x86_64 chipsets.

如果你只是使用 Android 模拟器来跑 Android 内核，那么就可以选择 goldfish 版本作为你的 Linux Kernel，如果你手头有上述内核支持的设备，那么，你就下载对应的 kernel 即可。

那么，如果你手头的设备或者开发板不在上述支持的列表中怎么办呢？

放 心，一般而言，Android 开发板的厂商会在上述 Linux Kernel 的基础上针对自己的开发板修改适配出一套可用的 Linux Kernel 的，你只需要到厂商的官网或者论坛查找对应的代码即可。对于自己设计开发的板子，则需要下载相近的 kernel 代码（如 CPU 型号相同），然 后针对性的修改和移植即可。

3. Android 分支与 Linux 分支的关系

Android 版本迭代更新的过程中，Linux Kernel 也在不断的迭代更新，因此，往往新的 Android 版本会使用较新的 Linux 内核分支，具体的分支对应关系我在 Google 官网上没有找 到，但是在维基百科和 stackoverflow 上找到了一份表格，如下所示：

Android Version	API Level	Linux Kernel in AOSP
1.5 Cupcake	3	2.6.27
1.6 Donut	4	2.6.29
2.0/1 Eclair	5-7	2.6.29
2.2.x Froyo	8	2.6.32
2.3.x Gingerbread	9, 10	2.6.35
3.x.x Honeycomb	11-13	2.6.36
4.0.x Ice Cream San	14, 15	3.0.1
4.1.x Jelly Bean	16	3.0.31
4.2.x Jelly Bean	17	3.4.0
4.3 Jelly Bean	18	3.4.39

维基百科讲的更加详细，具体介绍了每一个 Android 分支到底有哪些修改，地址如下：

http://en.wikipedia.org/wiki/Android_version_history

一 般情况下，不同的 Android 分支最好能使用对应的 Linux 内核分支版本，这样才能保证系统的正常编译通过和运行无误，但是，也可以详细了解某个 Android 版本对应的 Linux 内核到底做了哪些修改，并且把这些修改移植到其他版本的 Linux 内核上，也是可以实现低版本的 Android 运行在 高版本的 Linux 内核上的，当然，反过来难度比较大，因为一般高版本的 Android 内核会用到高版本的 Linux 内核特性，移植起来会麻烦很多。

4. 厂商提供的 Android 版本与 AOSP 的关系