# Android编译原理

## 引言

先看下面几条指令，相信编译过Android源码的人都再熟悉不过的。

source setenv.sh //初始化编译环境，包括后面的lunch和make指令

lunch //指定此次编译的目标设备以及编译类型

make -j12 //开始编译，默认为编译整个系统，其中-j12代表的是编译的job数量为12。

跟大家分享一下在研究上述几条指令含义的过程中，深入了解到的Android Build(编译)系统。

## 编译命令

准备好编译环境后，编译Android源码的第一步是 source build/envsetup.sh，其中source命令就是用于运行shell脚本命令，功能等价于”.”，因此该命令也等价于. build/envsetup.sh。在文件envsetup.sh声明了当前会话终端可用的命令，这里需要注意的是当前会话终端，也就意味着每次新打开一个终端都必须再一次执行这些指令。起初并不理解为什么新开的终端不能直接执行make指令，到这里总算明白了。

#### 2.1 代码编译

| **编译指令** | **解释** |
| --- | --- |
| m | 在源码树的根目录执行编译 |
| mm | 编译当前路径下所有模块，但不包含依赖 |
| mmm [module\_path] | 编译指定路径下所有模块，但不包含依赖 |
| mma | 编译当前路径下所有模块，且包含依赖 |
| mmma [module\_path] | 编译指定路径下所有模块，且包含依赖 |
| make [module\_name] | 无参数，则表示编译整个Android代码 |

下面列举部分模块的编译指令：

| **模块** | **make命令** | **mmm命令** |
| --- | --- | --- |
| init | make init | mmm system/core/init |
| zygote | make app\_process | mmm frameworks/base/cmds/app\_process |
| system\_server | make services | mmm frameworks/base/services |
| java framework | make framework | mmm frameworks/base  如果编译失败：需要再编译一次framework-res  [务必保证系统为debug版本，否则不生效的！！](https://www.zhihu.com/question/40425628) |
| framework资源 | make framework-res | mmm frameworks/base/core/res |
| jni framework | make libandroid\_runtime | mmm frameworks/base/core/jni |
| binder | make libbinder | mmm frameworks/native/libs/binder |

上述mmm命令同样适用于mm/mma/mmma，编译系统采用的是增量编译，只会编译发生变化的目标文件。当需要重新编译所有的相关模块，则需要编译命令后增加参数-B，比如make -B [module\_name]，或者 mm -B [module\_path]。

**Tips:**

* 对于m、mm、mmm、mma、mmma这些命令的实现都是通过make方式来完成的。
* mmm/mm编译的效率很高，而make/mma/mmma编译较缓慢；
* make/mma/mmma编译时会把所有的依赖模块一同编译，但mmm/mm不会;
* 建议：首次编译时采用make/mma/mmma编译；当依赖模块已经编译过的情况，则使用mmm/mm编译。

## DEBUG 技巧

貌似了

## 常见的坑

编译frameworks/base的mk不是递归编译关系，其子模块需要单独mmm，比如services模块，修改了pm之类，必须mmm frameworks/base/services

编译framwork，有些依赖不好处理

## REF

[理解Android编译命令](http://gityuan.com/2016/03/19/android-build/)