Android.bp快速入门

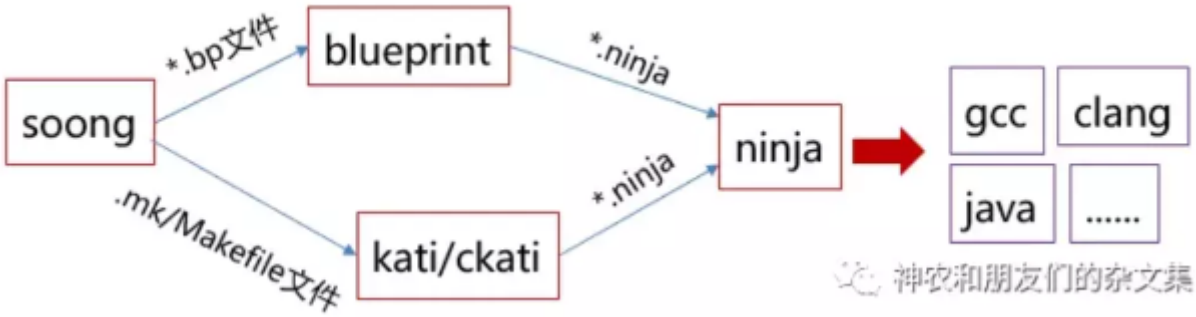
xiangzhihong
5.2k • 3

发布于 2022-05-16
英语

一、Soong 编译系统

在 Android 7.0 发布之前，Android 仅使用 GNU Make 描述和执行其构建规则。在Android系统级编译中，Make 构建系统得到了广泛的支持和使用，但它有一些缺点：编译缓慢、容易出错、无法扩展且难以测试，而Soong 构建系统正好提供了 Android build 所需的灵活性。

Soong 构建系统是在 Android 7.0 (Nougat) 中引入的，旨在取代 Make。它利用 Kati GNU Make 克隆工具和 Ninja 构建系统组件来加速 Android 的构建，Soong的编译流程图如下。



Soong编译系统下，原本打算输入是.bp文件，输出是.ninja文件，但是由于系统中的.mk文件还没有被完全消除掉，因此提供kati和ckati工具将.mk/Makefile文件转换为.ninja文件。.ninja成为编译系统的汇编文件，是不需要人为去修改的，相当于配置文件来调用gcc、java、clang等编译器去编译。

Soong编译系统的设计思想是消除.mk文件中的if/else等逻辑，使.bp文件只是一个简单的编译逻辑，这些复杂的选择配置逻辑应该放在更高层，比如使用更好调试的Python来编写。

二、Android.bp

Android.bp的出现就是为了替换Android.mk文件。bp跟mk文件不同，它是纯粹的配置，没有分支、循环等流程控制，不能做算数逻辑运算。如果需要控制逻辑，那么可以使用Android.mk或者Go语言进行编写。

2.1 示例

Android.bp 文件中的模块以模块类型开头，然后是一组格式属性：name: value，在一点上Android.bp的语法结构与JSON的语法结构相似，都是以键值对的形式编写，比如。

```
android_app {
    name: "Provision",
    srcs: ["**/*.java"],
    platform_apis: true,
    product_specific: true,
    certificate: "platform",
}
```

每个模块都必须具有 name 属性，并且相应值在所有 name 文件中必须是唯一的，仅有两个例外情况是命名空间和预构建模块中的 Android.bp 属性值，这两个值可能会重复。
srcs 属性以字符串列表的形式指定用于构建模块的源文件。可以使用模块引用语法 ":<module-name>" 来引用生成源文件的其他模块的输出，如 genrule 或 filegroup。

```
android_app{}
```

上面代码的意思是，该模块用于构建一个apk。如果要给模块指定一个名字，可以使用下面的方式。

```
name: "Provision",
```

如果指定了 android_app 模块类型（在代码块的开头），就需要设定该模块的name 。此设置会为模块命名，生成的 APK 将与模块名称相同，不过带有 .apk 后缀。例如，在本例中，生成的 APK 将命名为 Provision.apk

```
 srcs: ["**/*.java"],
```

上面代码的srcs用于指定当前的模块编译的源码位置，*表示通配符 。

```
platform_apis: true,
```

设置该标记后会使用sdk的hide的api來编译。编译的APK中使用了系统级API，必须设定该值。和Android.mk中的LOCAL_PRIVATE_PLATFORM_APIS的作用相同 。

```
product_specific: true,
```

设置该标记后，生成的apk会被安装在Android系统的product分区，和Android.mk中LOCAL_PRODUCT_MODULE作用相同。

certificate 用于指定APK的签名方式。如果不指定，默认使用testkey签名。与LOCAL_CERTIFICATE作用相同。 Android中共有四中签名方式：

- testkey：普通apk，默认使用该签名。
- platform：该apk完成一些系统的核心功能。经过对系统中存在的文件夹的访问测试，这种方式编译出来的apk所在进程的UID为system。
- shared：该apk需要和home/contacts进程共享数据。
- media：该apk是media/download系统中的一环。

2.2 常见模块类型

在Android.bp中，我们会基于模块类型来构建我们所需要的东西，常用的有以下几种类型：

- cc_library_shared：编译成动态库，类似于Android.mk中的BUILD_SHARED_LIBRARY。
- cc_binary：编译成可执行文件，类似于Android.mk中的BUILD_EXECUTABLE。
- name：编译出的模块的名称，类似于Android.mk中的LOCAL_MODULE。
- srcs：源文件，类似于Android.mk中的LOCAL_SRC_FILES。
- local_include_dirs：指定路径查找头文件，类似于Android.mk中的LOCAL_C_INCLUDES。
- shared_libs：编译所依赖的动态库，类似于Android.mk中的LOCAL_SHARED_LIBRARIES。
- static_libs：编译所依赖的静态库，类似于Android.mk中的LOCAL_STATIC_LIBRARIES。
- cflags：编译Flag，类似于Android.mk中的LOCAL_CFLAGS。

android_app

用于构建Android 应用程序安装包，是Android系统应用开发中常用的模块类型，与Android.mk中的BUILD_PACKAGE作用相同。

java_library

java_library用于将源代码构建并链接到设备的.jar文件中。默认情况下，java_library只有一个变量，它生成一个包含根据设备引导类路径编译的.class文件的.jar包。生成的jar不适合直接安装在设备上，通常会用作另一个模块的static_libs依赖项。

host的bootclasspath编译。

android_library

android_library将源代码与android资源文件一起构建并链接到设备的".jar"文件中。android_library有一个单独的变体，它生成一个包含根据device的bootclasspath编译的.class文件的.jar文件，以及一个包含使用aapt2编译的android资源的.package-res.apk文件。生成的apk文件不能直接安装在设备上，但可以用作android_app模块的static_libs依赖项。

2.3 设置变量

在bp中可以通过=号来设定一个全局变量。

```
src_path = ["**/*.java"]
android_app {
    name: "Provision",
    srcs: src_path,
    platform_apis: true,
    product_specific: true,
    certificate: "platform",
}
```

三、基础语法

3.1 数据类型

Android.bp中变量和属性是强类型，变量根据第一项赋值动态变化，属性由模块类型静态设置，支持的类型为：

- 布尔值（true或 false）
- 整数 （int）
- 字符串（"string"）
- 字符串列表 (["string1", "string2"])
- 映射 ({key1: "value1", key2: ["value2"]})

3.2 条件语句

Soong 不支持 Android.bp 文件中的条件语句。编译规则中如果需要处理条件语句，那么需要在

顶级属性。

例如，要支持特定的架构文件，可以使用以下命令：

```
cc_library {
  ...
  srcs: ["generic.cpp"],
  arch: {
    arm: {
      srcs: ["arm.cpp"],
    },
    x86: {
      srcs: ["x86.cpp"],
    },
  },
}
```

3.3 运算符

可以使用 + 运算符附加字符串、字符串列表和映射。可以使用 + 运算符对整数求和。附加映射会生成两个映射中键的并集，并附加在两个映射中都存在的所有键的值。

3.4 示例

Android.bp位于Android 10 : packages/apps/Car/Notification 目录下，参考示例如下：

```
    name: "CarNotificationLib",
    srcs: ["src/**/*.java"],
    resource_dirs: ["res"],
    manifest: "AndroidManifest-withoutActivity.xml",
    platform_apis: true,
    optimize: {
```

四、常见问题

4.1 引入第三方jar

在实际开发中，经常需要在app中引入第三方的jar。在Android.bp中，引入第三方的jar可以按下面的方式。

首先，在项目的根目录新建 libs文件夹，放入要导入的jar包，比如 CarServicelib.jar，然后在Android.bp中引入该jar。

```
java_import {
  name: "CarServicelib.jar",
  jars: ["libs/CarServicelib.jar"],
}
android_app {
    // 省去其它属性
    static_libs: [
        "CarServicelib.jar"
    ],
}
```

4.2 引入AndroidX

开发过程中，如果想要引入AndroidX的类库，可以参考下面的方式进行引入。

```
android_app {
    // 省去其它属性
    // 引入AndroidX库下的lib
    static_libs: [
        "androidx.cardview_cardview",
        "androidx.recyclerview_recyclerview",
        "androidx-constraintlayout_constraintlayout"
    ],
```

参考：

Soong 编译系统

Android 编译之android.bp

android

阅读 4k • 发布于 2022-05-16

赞 4 收藏 3 分享

xiangzhihong

著有《React Native移动开发实战》1，2、《Kotlin入门与实战》《Weex跨平台开发实战》、《Flutter跨平台...

5.2k 声望 **15.2k** 粉丝

关注作者

0 条评论

得票 最新

撰写评论 ...

提交评论

评论支持部分 Markdown 语法：`**粗体** _斜体_ [链接](http://example.com) `代码` - 列表 > 引用`。你还可以使用 @ 来通知其他用户。

4 3

jira-dev-tool插件安装失败的解决方法

最近，在运行【React + React Hook + TS 最佳实践仿 Jira 企业级项目】的时候，安装jira-dev-tool 插件出...

xiangzhihong 阅读 326

Android-Lifecycle超能解析-生命周期的那些事儿

版权声明：本文已授权微信公众号：Android必修课，转载请申明出处众所周知，Android凡是需要展示给用...

XBaron 赞 4 阅读 6.7k

布局大杀器—ConstraintLayout

Hi，大家好，看到标题后大家是不是一脸懵逼，这是啥？这小编搞事情？说好的六大布局咋又来个布局杀手...

下码看花 赞 1 阅读 3.9k

Android-博客及公众号推荐

首先强烈的推荐 stormzhang的博客，一直在关注他的博客和公众号，对我影响很大，不仅仅是Android学习...

秦子帅 赞 2 阅读 3.8k

六大布局之RelativeLayout

上一期我们给大家讲解了FrameLayout的使用，这一期我们为大家讲解一下RelativeLayout（相对布局）的...

下码看花 赞 1 阅读 5.9k

Android安全之Intent Scheme Url攻击

0X01 前言Intent scheme url是一种用于在web页面中启动终端app activity的特殊URL，在针对intent...

YAQ御安全 赞 1 阅读 9k

Android常用的网络框架

一、Android 常用的网络框架大多数应用程序基本都需要连接网络，发送一些数据给服务端，或者从服务端...

陛下陛下 赞 1 阅读 3.1k