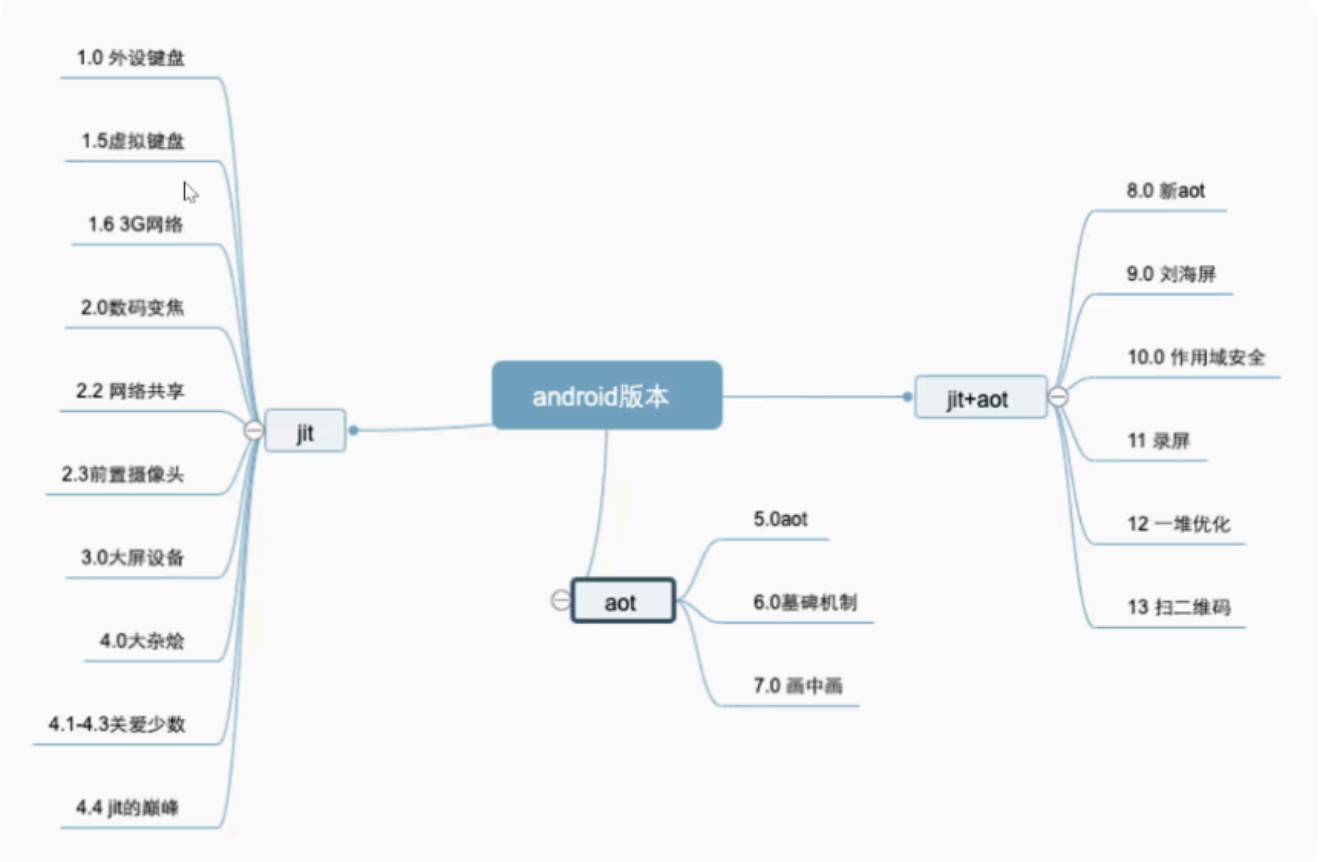
2025.6.9

Android1.0-15.0发展历史



Android5.0：引入AOT编译。

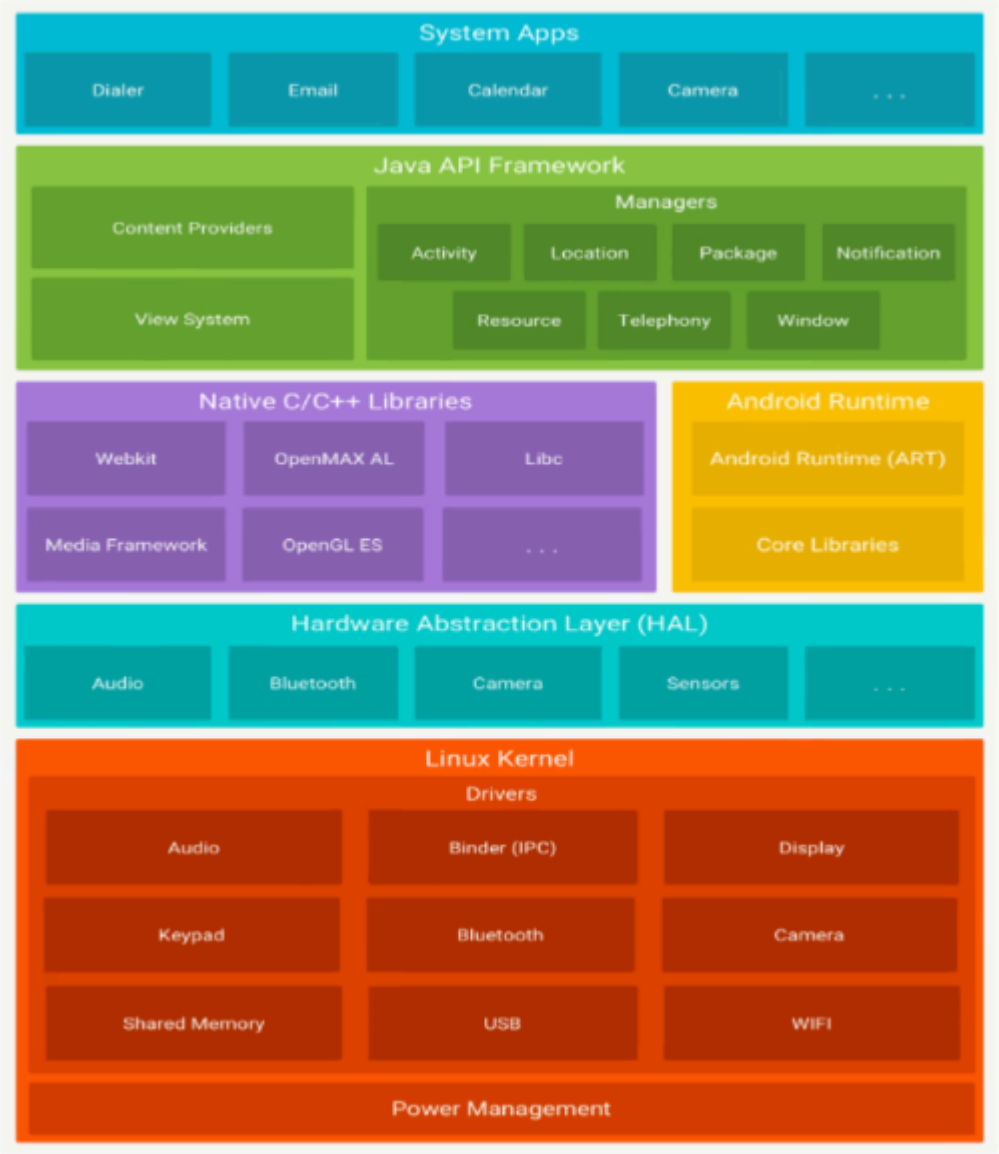
优点：安装时被编译为原生机器码，运行时直接执行；更低的运行时功耗；

缺点：安装时间变长；存储空间占用更大

Android6.0：引入墓碑机制：后台运行不再占用内存，电池续航能力提升30%。

Android架构介绍

共分五层架构：Kernel层、硬件抽象层、art C++层、Java framework层、应用层



kernel层

Linux 内核

Android Runtime (ART) 依靠 Linux 内核来执行底层功能，例如线程和低层内存管理。

使用 Linux 内核可让 Android 利用主要安全功能，并且允许设备制造商为内核开发硬件驱动程序。

硬件抽象层

抽象硬件能力，驱动硬件，兼容适配不同厂商或型号的硬件设备
为更高级别的 Java API 框架提供设备硬件功能使用。HAL 包含多个库模块
例如相机或蓝牙，wifi模块

art C++层

原生 C/C++ 库

- 为android核心组件提供类库编译支持（例如 ART 和 HAL）。
- 为 用 Android NDK 访问某些原生平台库 提供能力支持。

比如：
流媒体处理 （media ）
opengGL es等
webkit
三方lib库： scan扫码， 安全加密 等

Java framework层

Java API 框架

以 Java 语言编写的 API 使用 Android OS 的整个功能集（native c/c++ lib 和art）。这些 API 形成创建 Android 应用所需的构建块。

应用层

app层

- 为系统应用 或者 普通应用提供能力服务
- Android 随附一套用于电子邮件、短信、日历、浏览器，电话，短信等系统应用等
- 美团，支付宝，微信等

Android启动流程

1Loader层：激活Kernel。

当电源按下时引导芯片代码从预定义的地方（固化在ROM）开始执行。加载引导程序BootLoader到RAM中，然后执行。

2.引导程序BootLoader

引导程序BootLoader是在Android操作系统开始运行前的一个小程序，它的主要作用是把系统OS拉起来并运行。

3.Linux内核启动

当内核启动时，设置缓存、被保护存储器、计划列表、加载驱动。在内核完成系统设置后，它首先在系统文件中寻找init.rc文件，并启动init进程。

4.init进程启动

（1）创建和挂载启动所需的文件目录。

（2）初始化和启动属性服务。

（3）解析init.rc配置文件并启动Zygote进程

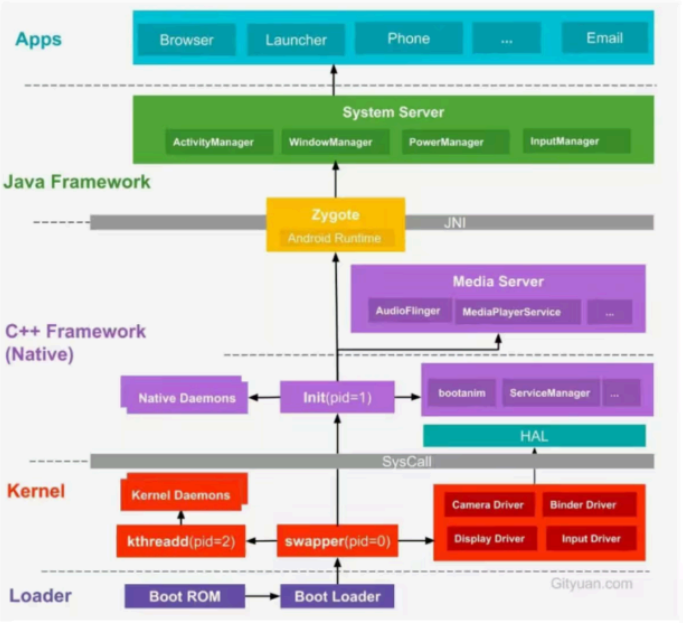
启动Kernel的swapper进程(pid=0)：该进程又称为idle进程 用于进程管理，内存管理，加载Display,Camera Driver, Binder Driver等相关工作

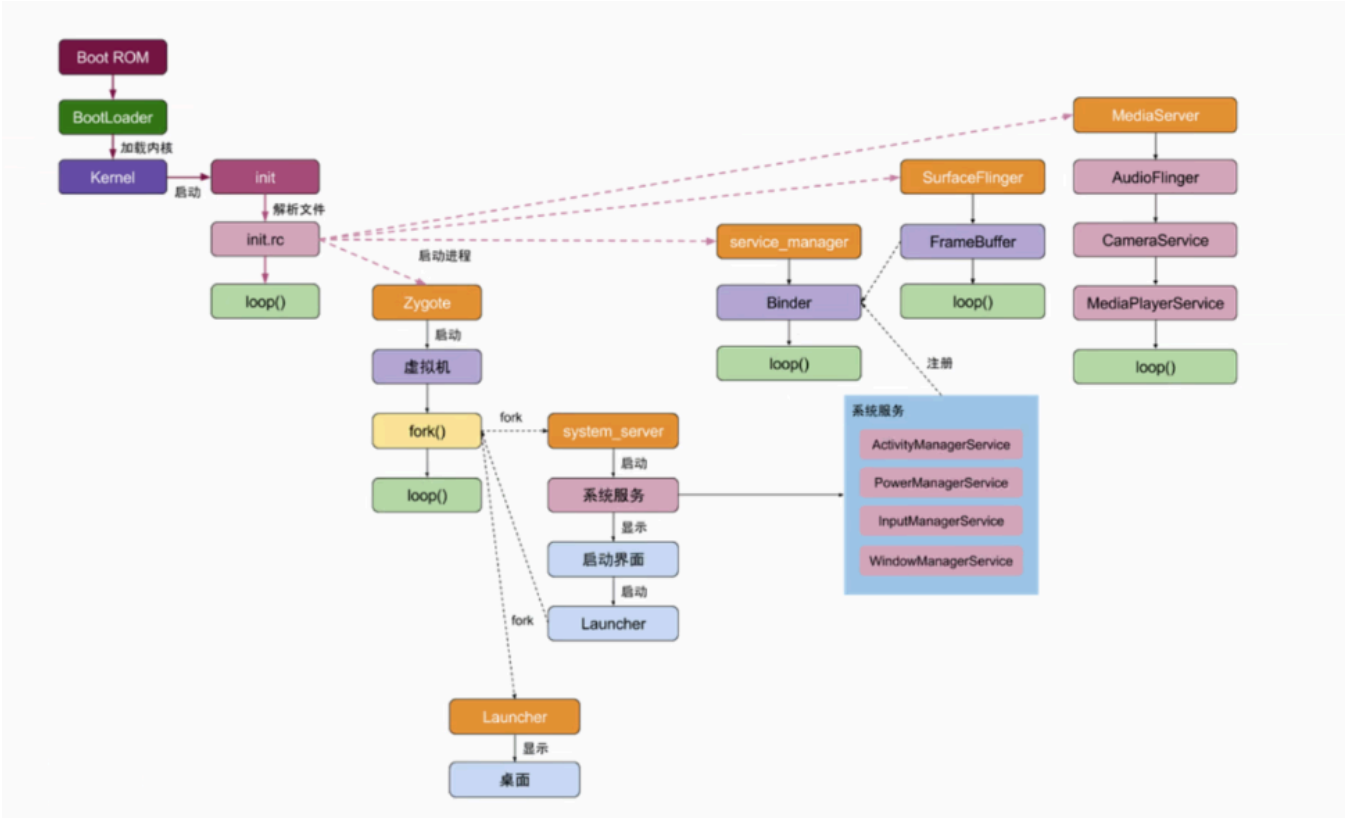
init进程会孵化出ueventd、logd、installd、adbd、lmkd等用户守护进程

init进程还启动 servicemanager(binder服务管家)、 bootanim(开机动画)等重要服务

init进程孵化出Zygote进程

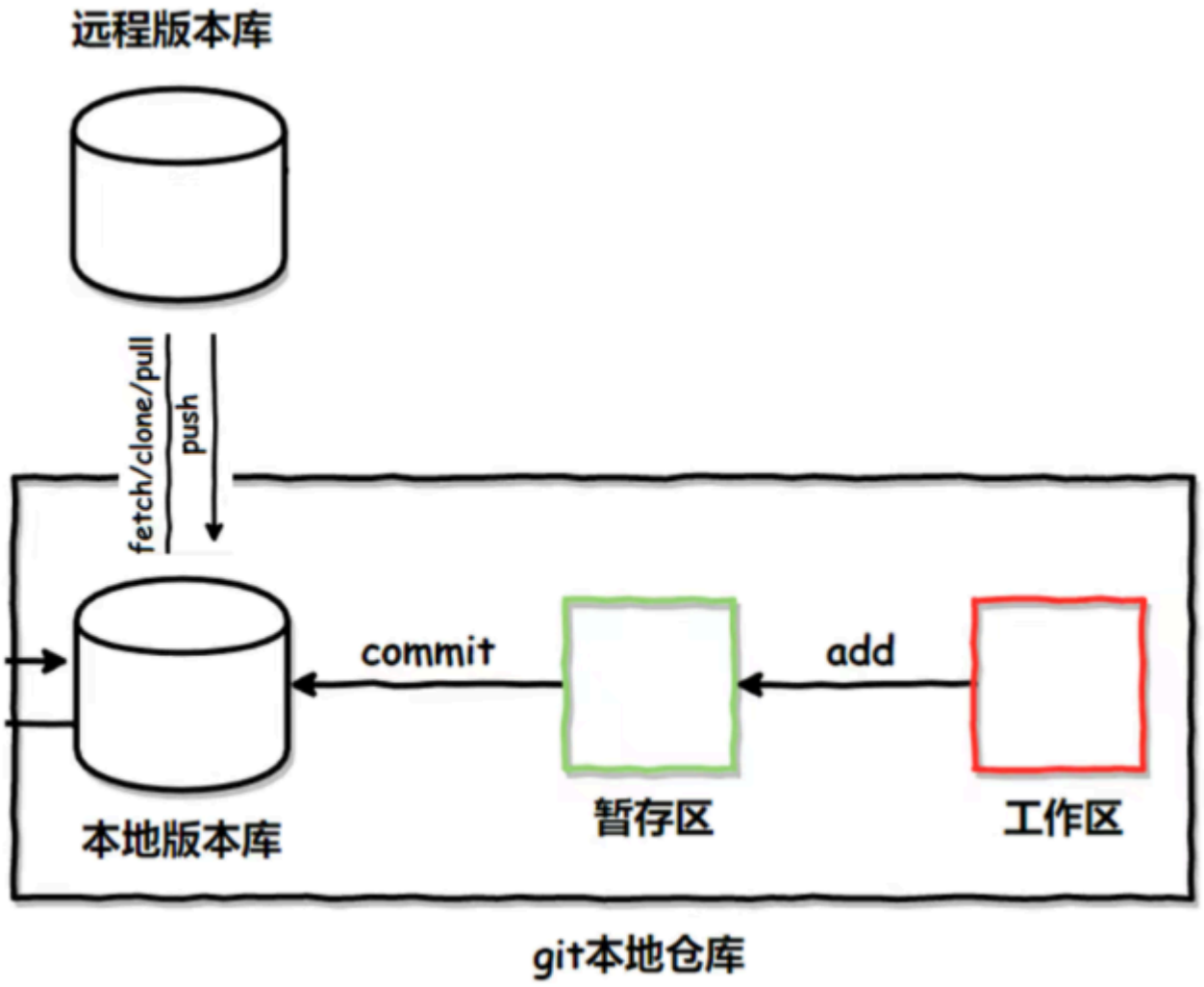
Zygote进程孵化各种manager 和 Launcher




Boot ROM
BootLoader
加载内核
Kernel
启动
init
解析文件
init.rc
loop()
启动进程
Zygote
启动
虚拟机
fork()
loop()
fork
system_server
启动
系统服务
显示
启动界面
启动
Launcher
fork
Launcher
显示
桌面
MediaServer
SurfaceFlinger
AudioFlinger
service_manager
FrameBuffer
CameraService
Binder
loop()
MediaPlayerService
loop()
注册
loop()
系统服务
ActivityManagerService
PowerManagerService
InputManagerService
WindowManagerService

Git分布式源代码工具使用

git整体逻辑



git常用命令

git init 初始化仓库

git clone ssh/http 从远端仓库拉取代码

git add . 将本地的修改全部提交到暂存区

git commit -m "" 提交到本地仓库，-m进行备注信息

git push 远端仓库名 将本地仓库提交到远端仓库

git status 查看当前分支状态

git checkout 还原某个文件

git reset 还原某次版本提交

git branch 查看当前分支 switch切换分支

git的目录结构

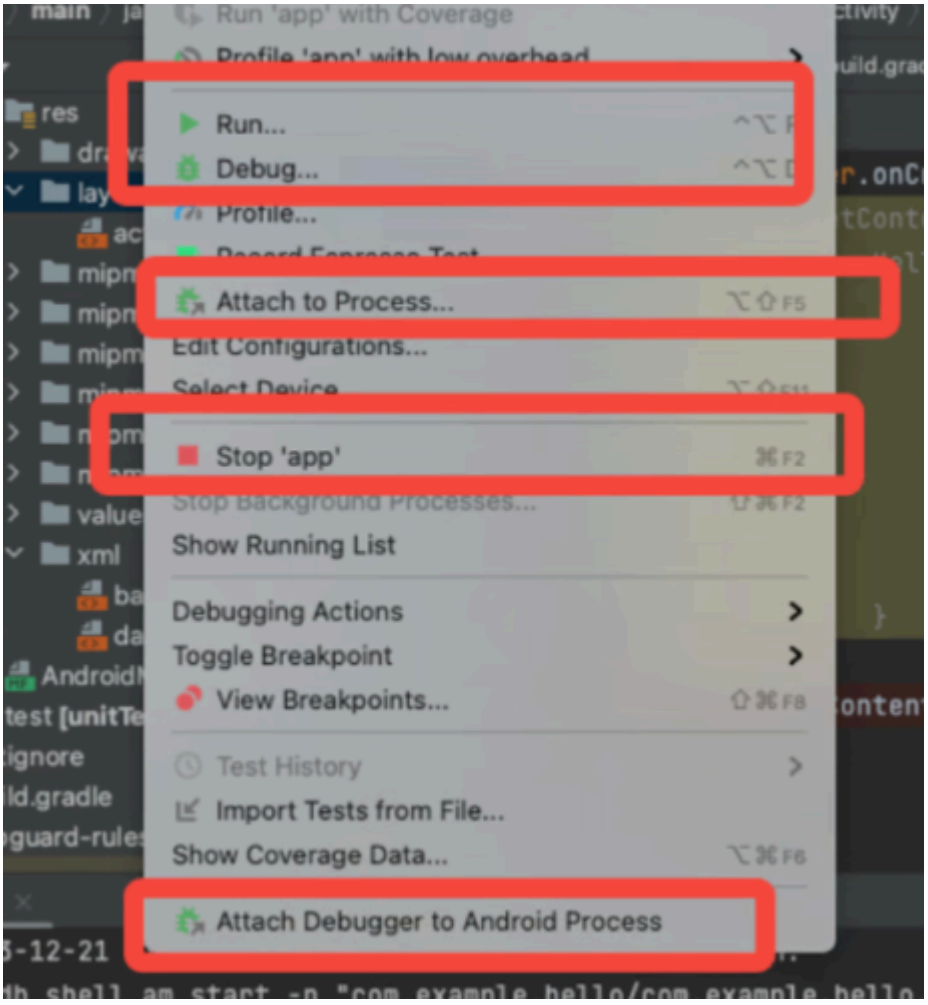
特点:树状结构包含哈希值

运行环境搭建

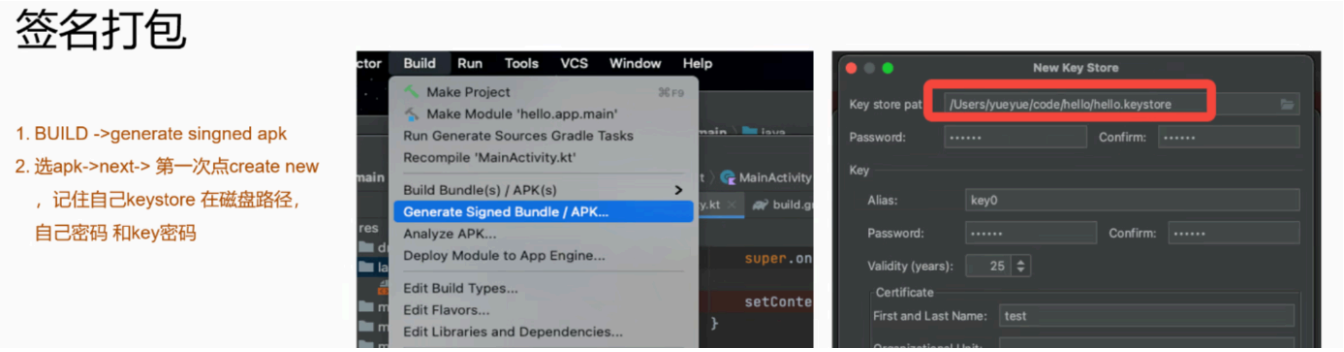
SDK、JDK_HOME、Android studio等安装搭建

Android运行、调试、编译、打包

debug、run：



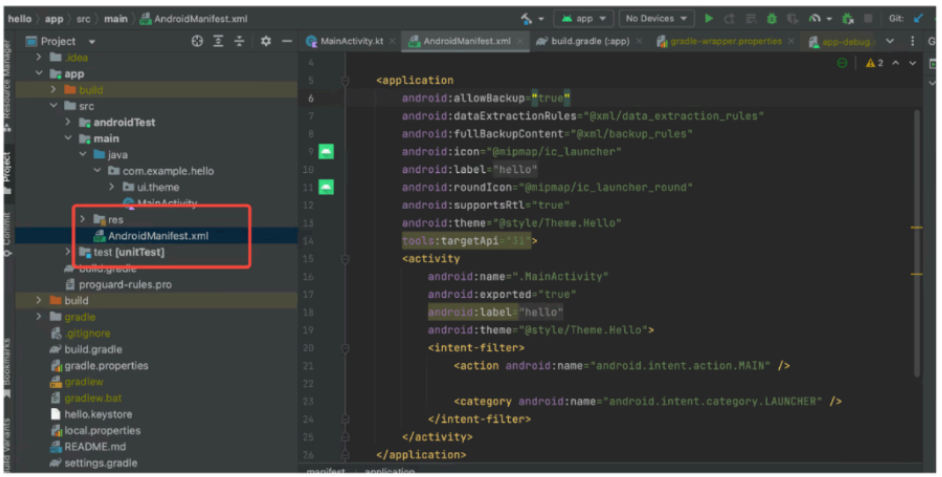
签名打包：



application介绍

Application介绍

Application是用来维护全局应用状态的基类.
可以创建子类并且在AndroidManifest.xml文件中通过name属性来标记Application入口类



- **创建继承自Application类的子类**

```
public class MyApp extends Application{

    @Override
    public void onCreate() {
        super.onCreate();
        // do something
    }
}
```

- **配置自定义 Application**

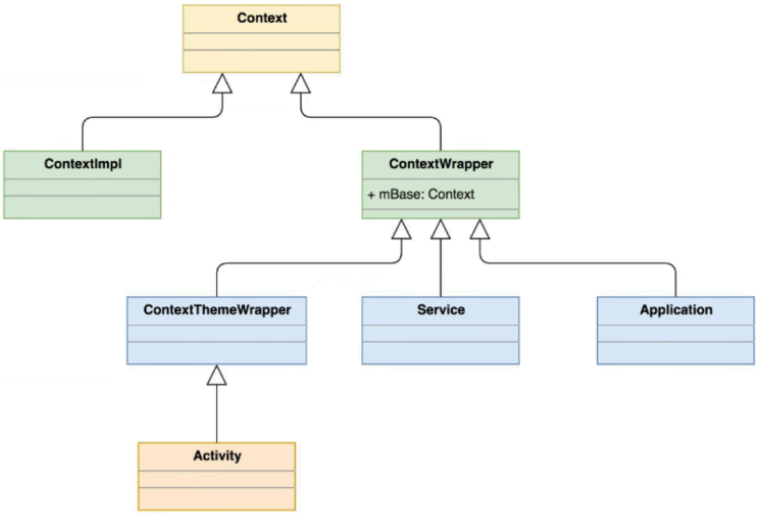
在 AndroidManifest.xml 文件中声明 Application

```
<application
        android:name=".MyApp"
</application>
```

context介绍

Context简介

Android 系统提供的抽象类 . 允许访问application的 resources and classes, 以及启动 activity，services，广播 接收 intents等



adb命令

Android 调试桥(adb)

Android 调试桥（`adb`）是一种功能多样的命令行工具，可让您与设备进行通信。`adb` 命令可用于执行各种设备操作，例如安装和调试应用。`adb` 提供对 Unix shell（可用来在设备上运行各种命令）的访问权限。它是一种客户端-服务器程序，包括以下三个组件：

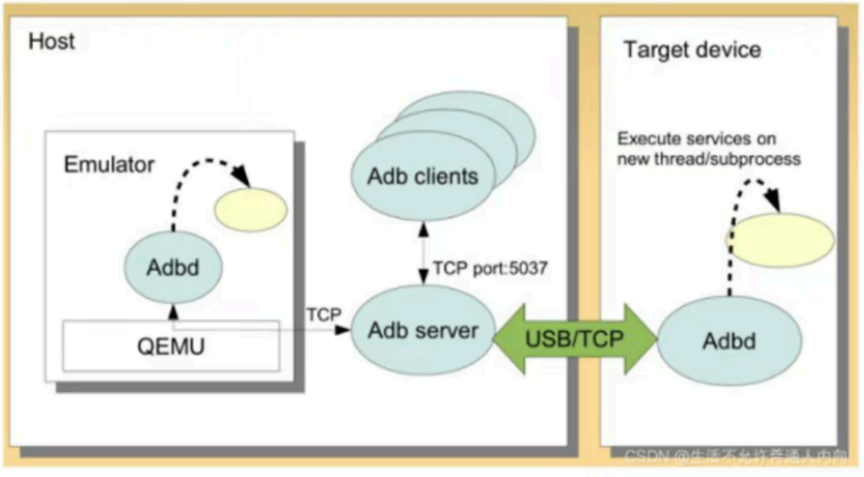
- **客户端**：用于发送命令。客户端在开发机器上运行。您可以通过发出 `adb` 命令从命令行终端调用客户端。
- **守护程序 (adbd)**：用于在设备上运行命令。守护程序在每个设备上作为后台进程运行。
- **服务器**：用于管理客户端与守护程序之间的通信。服务器在开发机器上作为后台进程运行。

`adb` 包含在 Android SDK 平台工具软件包中。您可以使用 SDK 管理器下载此软件包，该管理器会将其安装在 `android_sdk/platform-tools/` 下。如果您需要独立的 Android SDK 平台工具软件包，请点击此处进行下载。

如需了解如何通过 `adb` 连接设备以供使用，包括如何使用 Connection Assistant 来排查常见问题，请参阅在硬件设备上运行应用。

Adb原理及启动流程

1. 检查手机是否处于开发者模式
2. shell 启动adb客户端进程
3. 启动 adb服务进程并连接
4. adb服务进程与设备的adbd进程建立连接



adb命令：

:查看系统参数值

adb shell getprop system_prams

::设置系统参数值

adb shell setprop system_prams params_value

::查看手机型号

adb shell getprop ro.product.model

::重启

adb reboot

::查看日志

adb logcat > log.txt

adb logcat | findstr "matched_string"

::安装/卸载

adb install xxx.apk

adb uninstall package_name