简述

- 了解Android源代码重要目录结构；Android启动流程；repo、重要命令的含义和作用

记录

Android源代码结构

packages目录

packages目录	描述
apps	核心应用程序
experimental	第三方应用程序
inputmethods	输入法目录
providers	内容提供者目录
screensavers	屏幕保护
services	通信服务

framework目录

/frameworks/base目录	描述
api	定义API
core	核心库
docs	文档
include	头文件
libs	库
media	多媒体相关库
nfc-extras	NFC相关
opengl	2D/3D 图形API
sax	XML解析器
telephony	电话通讯管理
tests	测试相关
wifi	wifi无线网络
cmds	重要命令：am、app_proce等
data	字体和声音等数据文件

/frameworks/base目录	描述
graphics	图形图像相关
keystore	和数据签名证书相关
location	地理位置相关库
native	本地库
obex	蓝牙传输
packages	设置、TTS、VPN程序
services	系统服务
test-runner	测试工具相关
tools	工具

C/C++程序库部分

- 系统运行库层（Native)中的 C/C++程序库的类型繁多，功能强大，C/C++程序库并不完全在一个目录中
-

目录位置	描述
bionic/	Google开发的系统C库，以BSD许可形式开源。
/frameworks/av/media	系统媒体库
/frameworks/native/opengl	第三方图形渲染库
/frameworks/native/services/surfaceflinger	图形显示库，主要负责图形的渲染、叠加和绘制等功能
external/sqlite	轻量型关系数据库SQLite的C++实现

Android系统启动流程

1. 启动电源以及系统启动
    - 当电源按下时引导芯片代码开始从预定义的地方（固化在ROM）开始执行。加载引导程序Bootloader到RAM，然后执行。
2. 引导程序Bootloader
    - 引导程序是在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
    - 内核启动时，设置缓存、被保护存储器、计划列表，加载驱动。当内核完成系统设置，它首先在系统文件中寻找”init”文件，然后启动root进程或者系统的第一个进程。
4. init进程启动
    - init进程是Android系统中用户空间的第一个进程，作为第一个进程，它被赋予了很多极其重要的工作职责，比如创建zygote(孵化器)和属性服务等。
    - init进程是由多个源文件共同组成的，这些文件位于源码目录system/core/init。

- 作用
  - 创建一些文件夹并挂载设备
  - 初始化和启动属性服务
  - 解析init.rc配置文件并启动zygote进程

5. Zygote进程启动
   - 在Android系统中，DVM(Dalvik虚拟机)、应用程序进程以及运行系统的关键服务的SystemServer进程都是由Zygote进程来创建的，我们也将它称为孵化器。它通过fork(复制进程)的形式来创建应用程序进程和SystemServer进程，由于Zygote进程在启动时会创建DVM，因此通过fork而创建的应用程序进程和SystemServer进程可以在内部获取一个DVM的实例拷贝。
   - 创建JavaVM并为JavaVM注册JNI，创建服务端Socket，启动SystemServer进程。
   - 作用
     1. 创建AppRuntime并调用其start方法，启动Zygote进程。
     2. 创建DVM并为DVM注册JNI.
     3. 通过JNI调用ZygoteInit的main函数进入Zygote的Java框架层。
     4. 通过registerZygoteSocket函数创建服务端Socket，并通过runSelectLoop函数等待ActivityManagerService的请求来创建新的应用程序进程。
     5. 启动SystemServer进程。
6. SystemServer进程启动
   - 作用
     1. 启动Binder线程池，这样就可以与其他进程进行通信。
     2. 创建SystemServiceManager用于对系统的服务进行创建、启动和生命周期管理。
     3. 启动各种系统服务。

引导服务	**作用**
Installer	系统安装apk时的一个服务类，启动完成Installer服务之后才能启动其他的系统服务
ActivityManagerService	负责四大组件的启动、切换、调度。
PowerManagerService	计算系统中和Power相关的计算，然后决策系统应该如何反应
LightsService	管理和显示背光LED
DisplayManagerService	用来管理所有显示设备
UserManagerService	多用户模式管理
SensorService	为系统提供各种感应器服务
PackageManagerService	用来对apk进行安装、解析、删除、卸载等等操作
核心服务	
BatteryService	管理电池相关的服务

引导服务	作用
UsageStatsService	收集用户使用每一个APP的频率、使用时常
WebViewUpdateService	WebView更新服务
其他服务	
CameraService	摄像头相关服务
AlarmManagerService	全局定时器管理服务
InputManagerService	管理输入事件
WindowManagerService	窗口管理服务
VrManagerService	VR模式管理服务
BluetoothService	蓝牙管理服务
NotificationManagerService	通知管理服务
DeviceStorageMonitorService	存储相关管理服务
LocationManagerService	定位管理服务
AudioService	音频相关管理服务

7. Launcher启动
    - 被SystemServer进程启动的ActivityManagerService会启动Launcher，Launcher启动后会将已安装应用的快捷图标显示到界面上。
        - 应用程序Launcher在启动过程中会请求PackageManagerService返回系统中已经安装的应用程序的信息，并将这些信息封装成一个快捷图标列表显示在系统屏幕上，这样用户可以通过点击这些快捷图标来启动相应的应用程序。

repo

- Repo 是一个建立在 Git 之上的工具。Repo 帮助管理许多 Git 存储库，上传到修订控制系统，并使部分开发工作流程自动化。Repo 并不是要取代 Git，只是为了让使用 Git 更容易。repo 命令是一个可执行的 Python 脚本，您可以将其放在路径中的任何位置。
- 使用 Repo 需遵循的格式
    - ```
      repo command options
      ```
    - 可选元素显示在方括号 [ ] 中。
- 帮助
    - 所有命令的摘要
        - ```
          repo help
          ```
    - 某个命令的详细信息
        - ```
          repo help command
          ```
          ```

    - 仅查看可用选项的列表
        - ```
          repo command --help
          ```

- repo init
    - `-u`：指定从中检索清单代码库的网址。常见清单位于 `https://android.googlesource.com/platform/manifest`。
    - `-m`：选择代码库中的清单文件。如果未选择清单名称，则默认为 `default.xml`。
    - `-b`：指定修订版本，即特定的 manifest-branch。
- repo sync
    - `-c`：只从服务端获取当前分支。
    - `-f`：即使某个项目同步失败，也继续同步其他项目。
    - `-j <num>`：设定并发数。默认 4 个并发。
    - 可以查看 `.repo/manifest.xml` 根据 `path` 拉取一部分代码
          ```