Android系统架构概述

罗升阳

http://weibo.com/shengyangluo

http://blog.csdn.net/luoshengyang

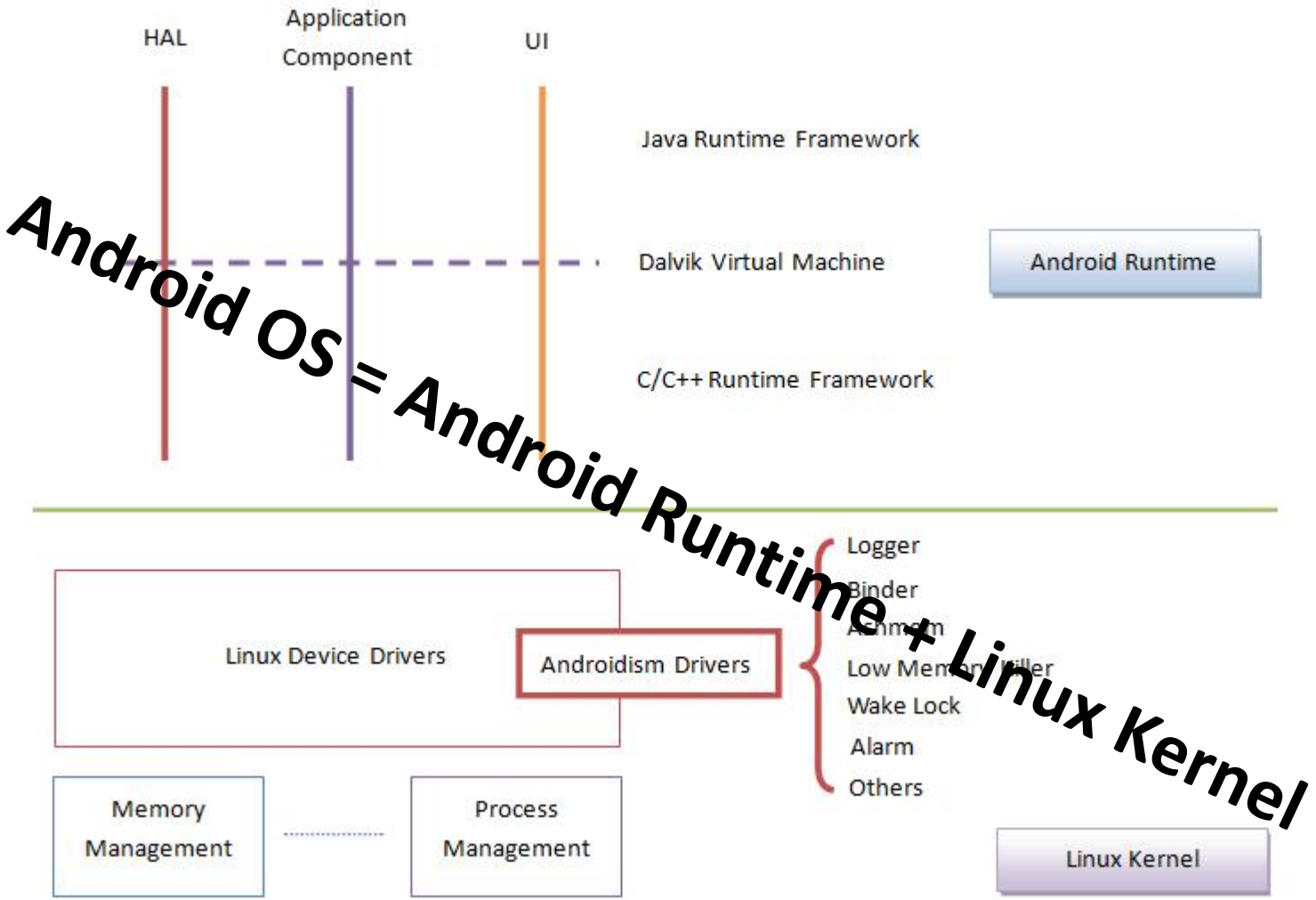
About Me

- 《老罗的Android之旅》博客作者
- 《Android系统源代码情景分析》书籍作者
- 博客：http://blog.csdn.net/Luoshengyang
- 微博：http://weibo.com/shengyangluo

Agenda

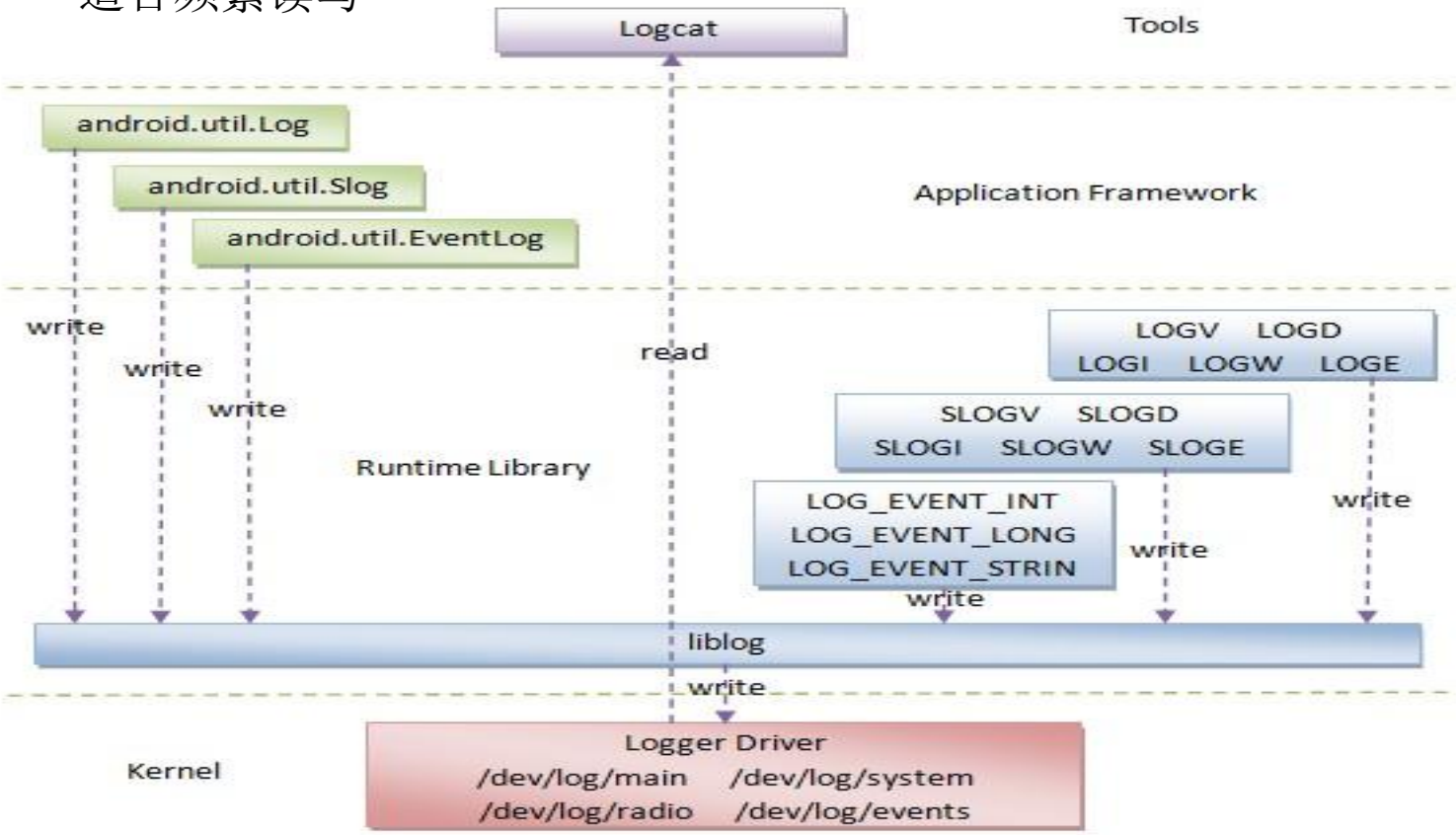
- Android系统整体架构
- Android专用驱动
- Android硬件抽象层
- Android应用程序组件
- Android应用程序框架
- Android用户界面架构
- Dalvik虚拟机

Android系统整体架构



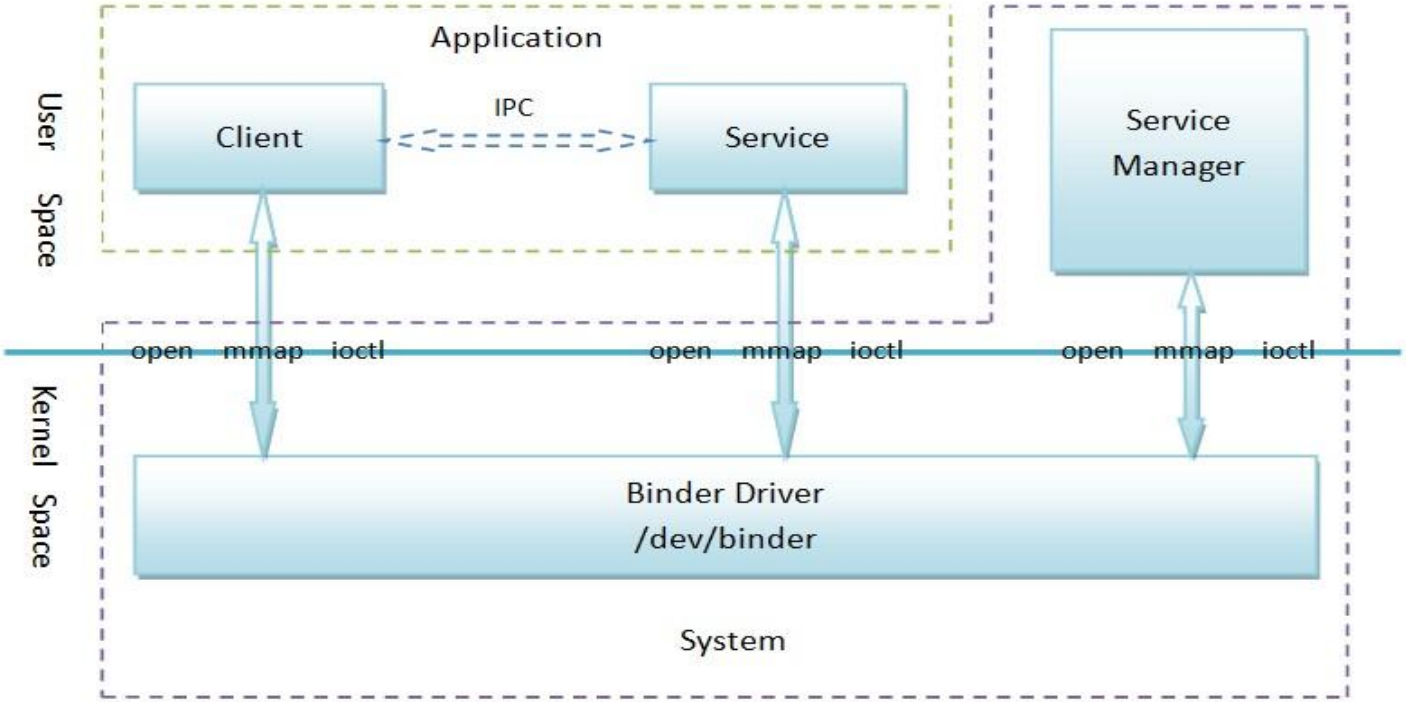
Android专用驱动

- Logger
  - 完全内存操作
  - 适合频繁读写



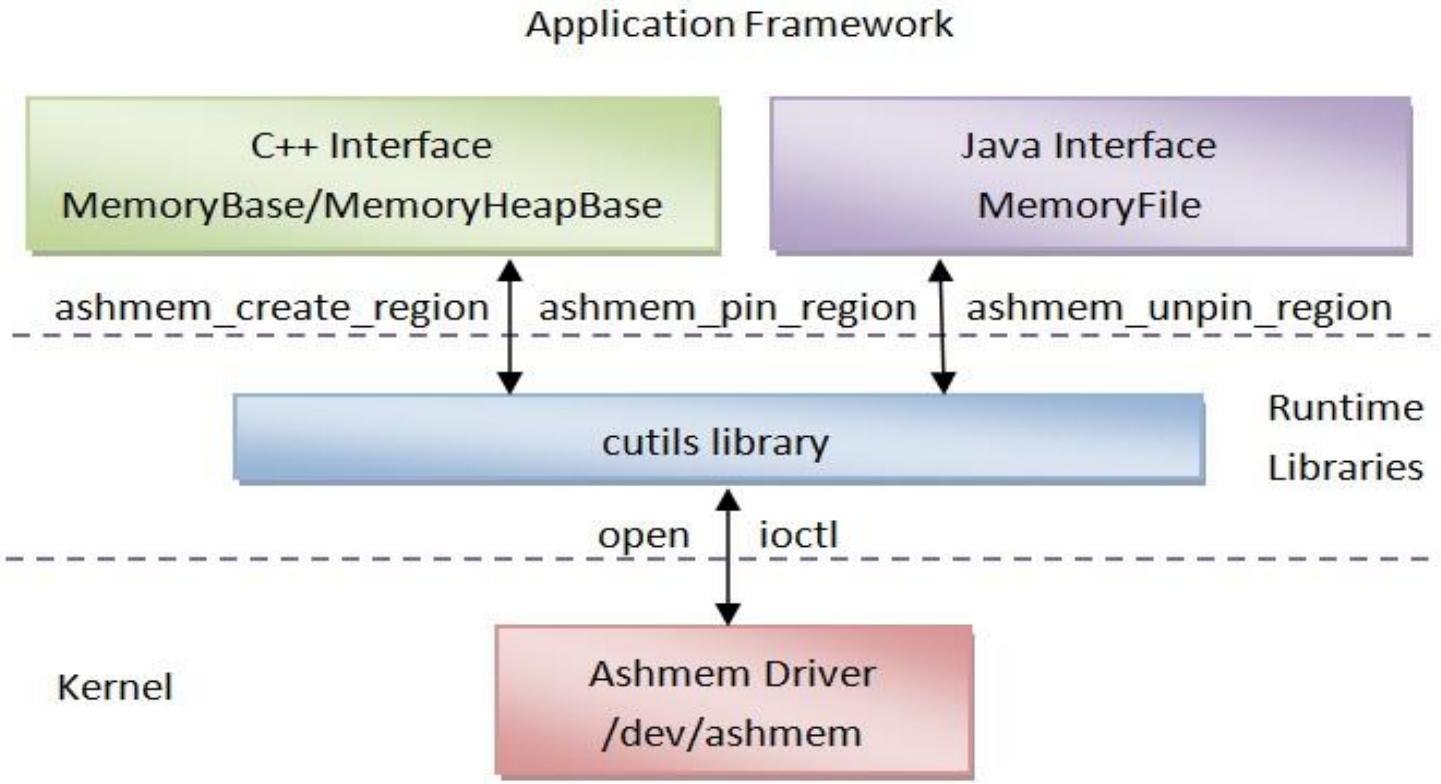
Android专用驱动(续)

- Binder
  - Client/Server模型
  - 进程间一次数据拷贝
  - 进程内直接调用



Android专用驱动(续)

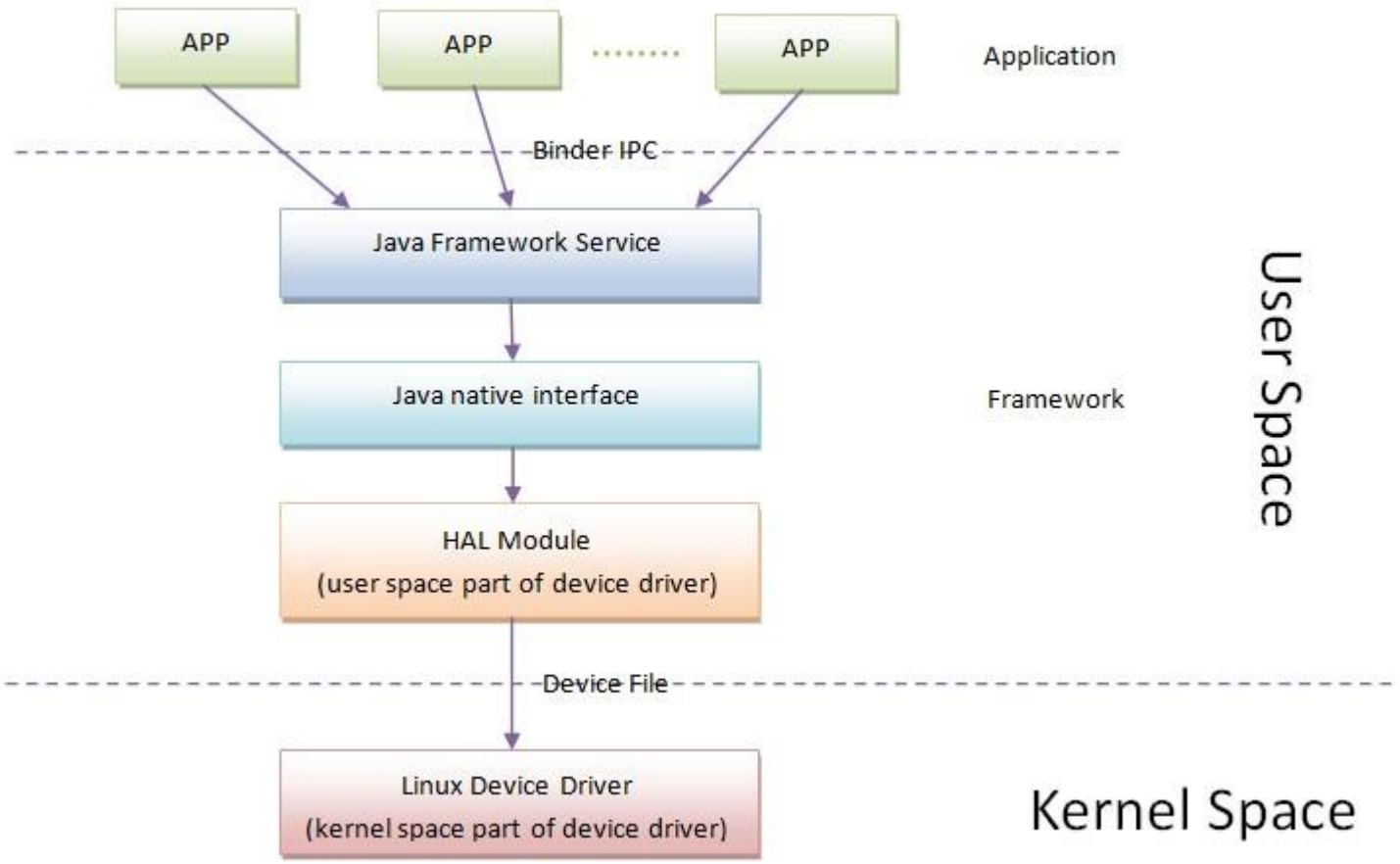
- Ashmem
  - 使用文件描述符描述
  - 通过Binder在进程间传递



Android硬件抽象层HAL

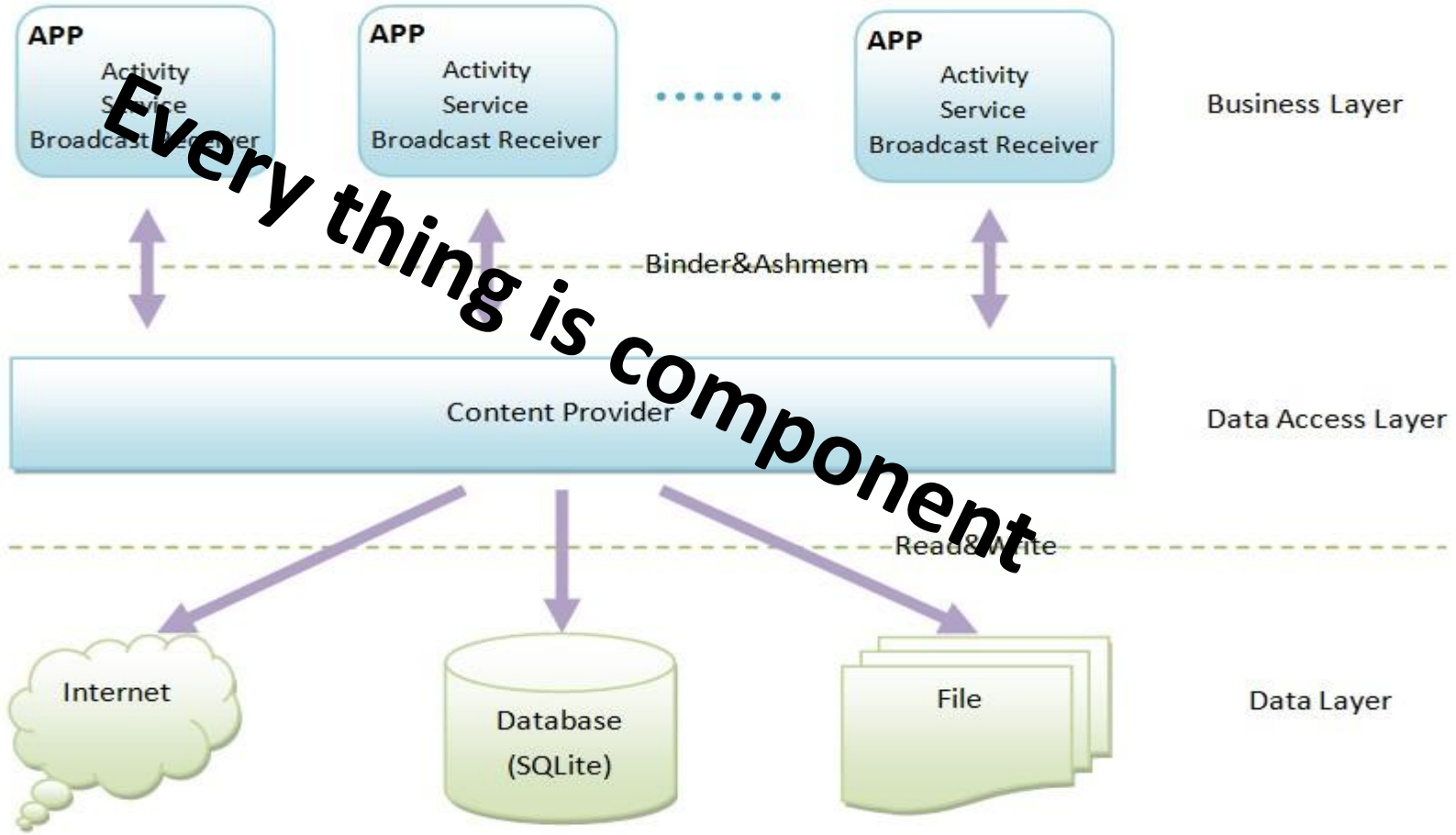
- 设备驱动分为内核空间和用户空间两部分
  - 保护厂商利益（出发点）
  - 内核空间主要负责硬件访问逻辑（GPL）
  - 用户空间主要负责参数和访问流程控制（Apache License）
- 用户空间部分设备驱动即为HAL Module
  - HAL Module通过设备文件访问内核空间部分设备驱动
- 系统服务通过HAL Module对硬件进行管理
  - 系统服务通过JNI访问HAL Module
- 应用程序通过系统服务对硬件进行访问
  - 应用程序通过Binder IPC访问系统服务

Android硬件抽象层HAL(续)



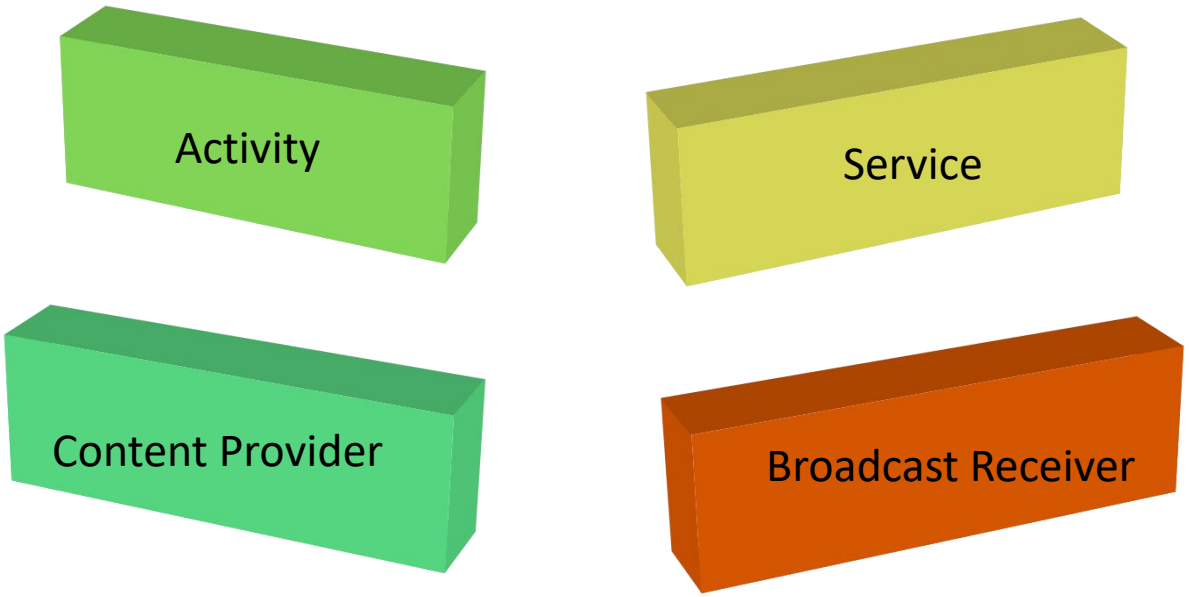
Android应用程序组件

- Android应用程序的一般架构



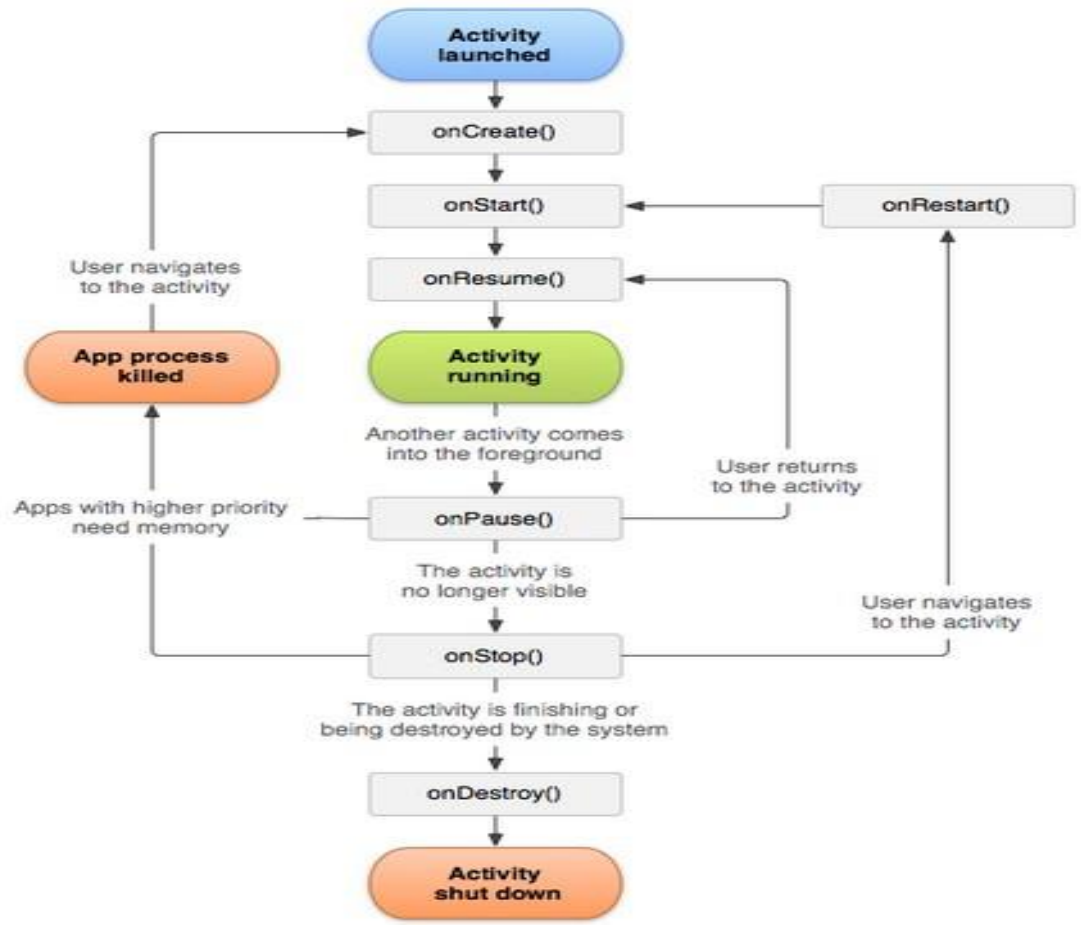
Android应用程序组件(续)

- 四大组件(砖头)
  - Activity -- UI、交互
  - Service -- 后台计算
  - Broadcast Receiver -- 广播
  - Content Provider -- 数据



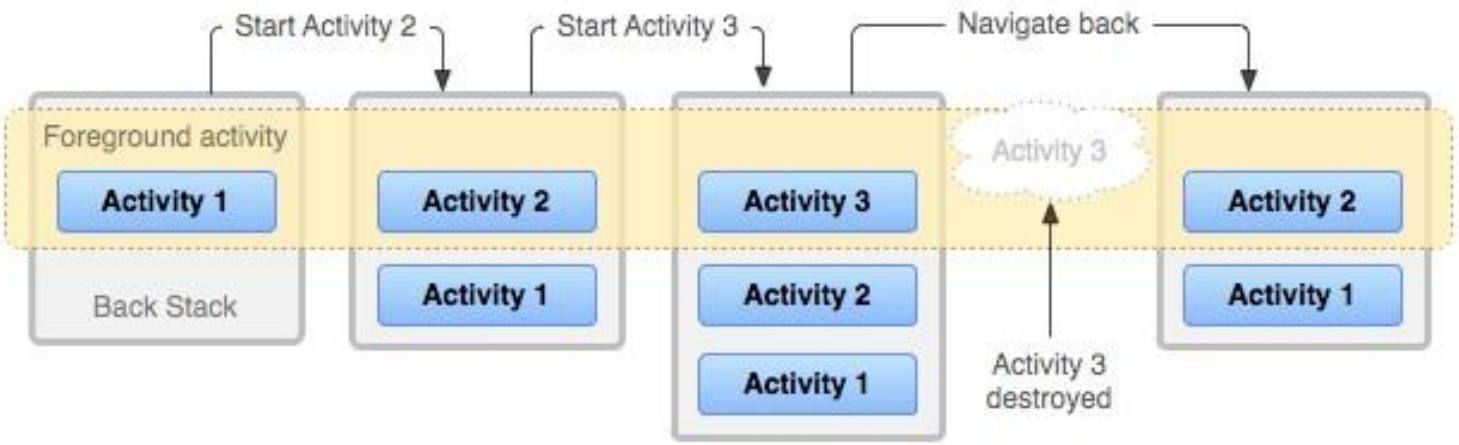
Android应用程序组件(续)

- Activity生命周期
  - 由ActivityManagerService管理



Android应用程序组件(续)

- Activity堆栈
  - 由ActivityManagerService维护

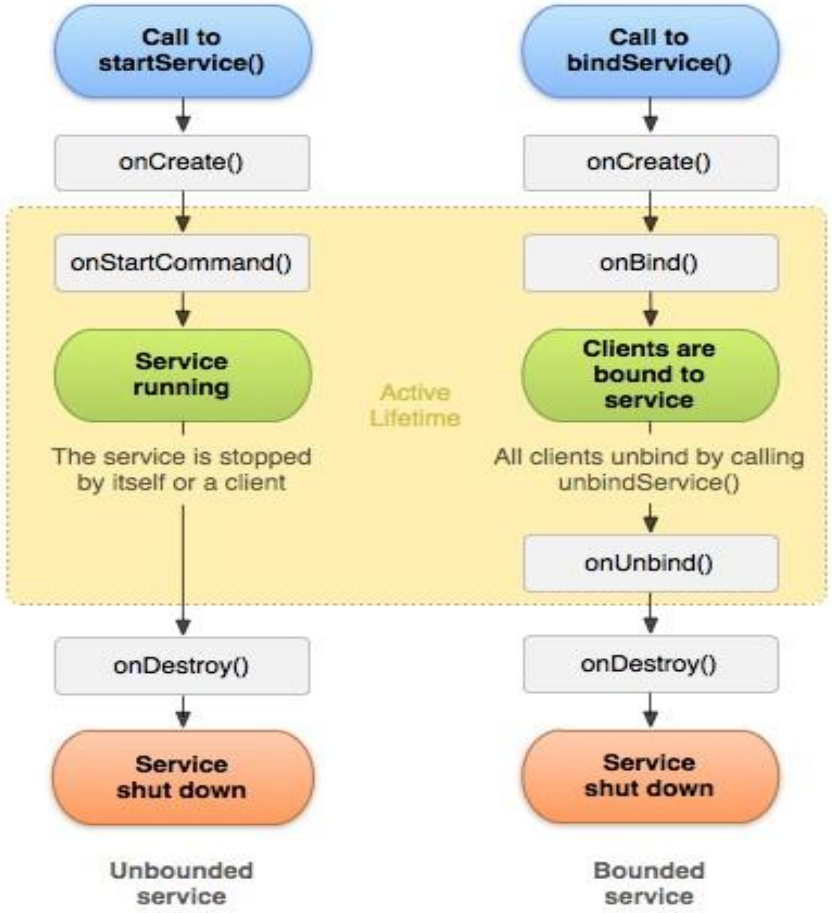


Android应用程序组件(续)

- Activity在堆栈中以Task的形式聚集在一起
  - Task由一系列相关的Activity组成，描述用户完成某一个操作所需要的Activity
  - 当我们从Launcher上点击一个应用图标的时候，就启动一个Task
  - Task是用Android多任务的一种体现
  - http://developer.android.com/guide/components/tasks-and-back-stack.html

Android应用程序组件(续)

- Service
  - Unbounded service
  - Bounded service

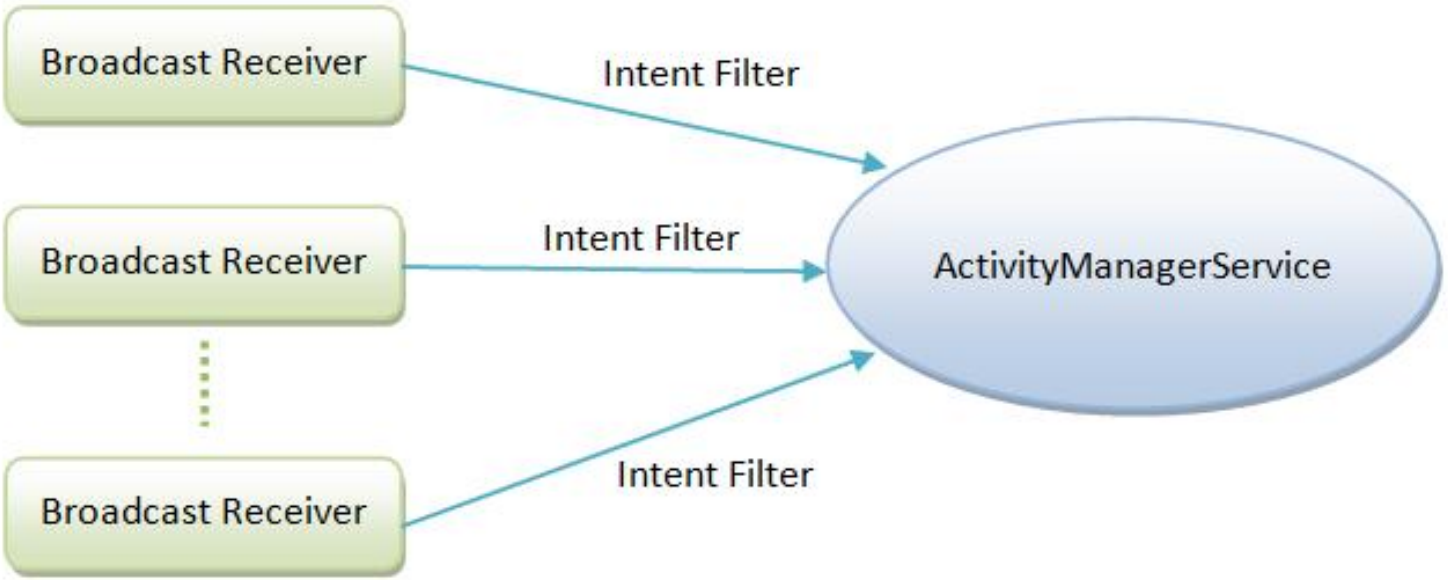


Android应用程序组件(续)

- Broadcast Receiver
  - 注册
    - 静态 -- AndroidManifest.xml
    - 动态 -- Context.registerReceiver
  - 广播
    - 无序 -- Context.sendBroadcast
    - 有序 -- Context.sendOrderedBroadcast

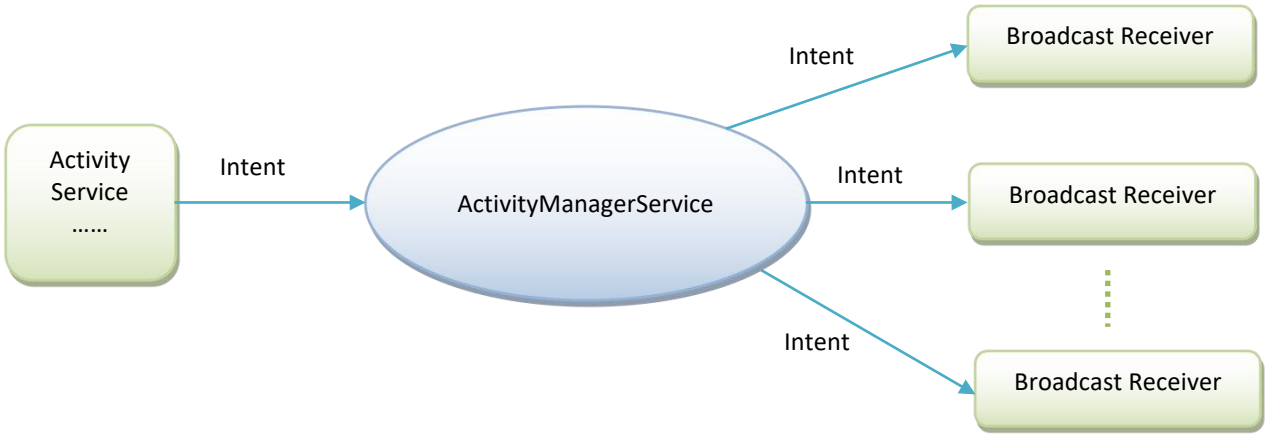
Android应用程序组件(续)

- 注册广播



Android应用程序组件(续)

- 发送广播

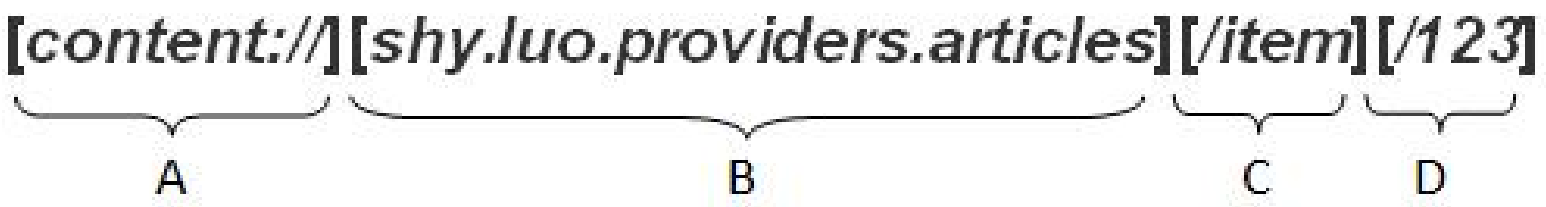


Android应用程序组件(续)

- Content Provider
  - 通过URI来描述
  - 数据访问接口
  - 数据更新机制

Android应用程序组件(续)

- Content Provider的URI结构
  - A -- Scheme
  - B -- Authority
  - C -- Resource Path
  - D -- Resource ID



Android应用程序组件(续)

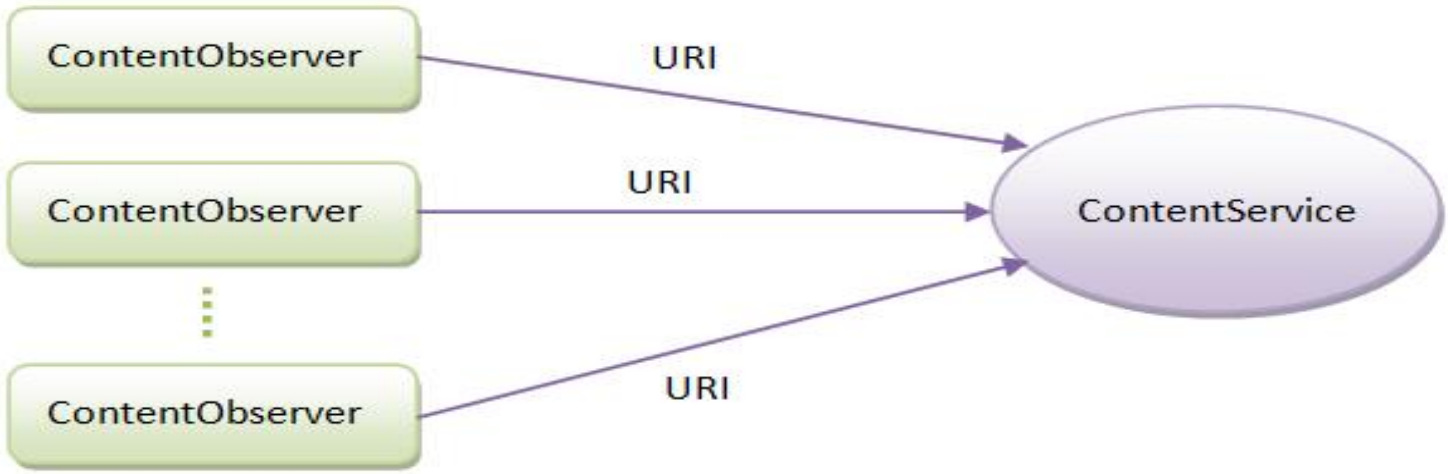
- Content Provider数据访问接口
  - Insert
  - Update
  - Delete
  - Query
  - Call -- Hidden

Android应用程序组件(续)

- Content Provider数据更新机制
  - 注册内容观察者 -- ContentResolver.ContentObserver
  - 发送数据更新通知 -- ContentResolver.notifyChange

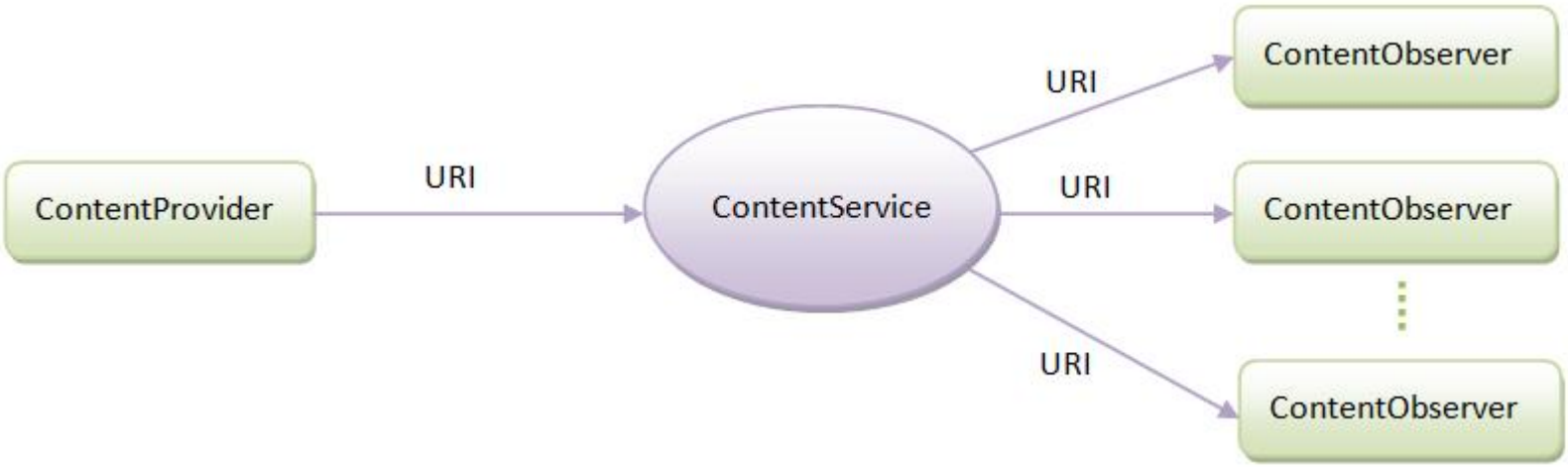
Android应用程序组件(续)

- 注册Content Provider的内容观察者



Android应用程序组件(续)

- 发送Content Provider数据更新通知



Android应用程序框架

- 管理硬件
- 提供服务
- 组件管理
- 进程管理

Android应用程序框架(续)

- 按服务类型划分
  - Hardware Service
    - CameraService
    - LocationManagerService
    - LightsService
    - ……
  - Software Service
    - PackageManagerService
    - ActivityManagerService
    - WindowManagerService
    - ……

Android应用程序框架(续)

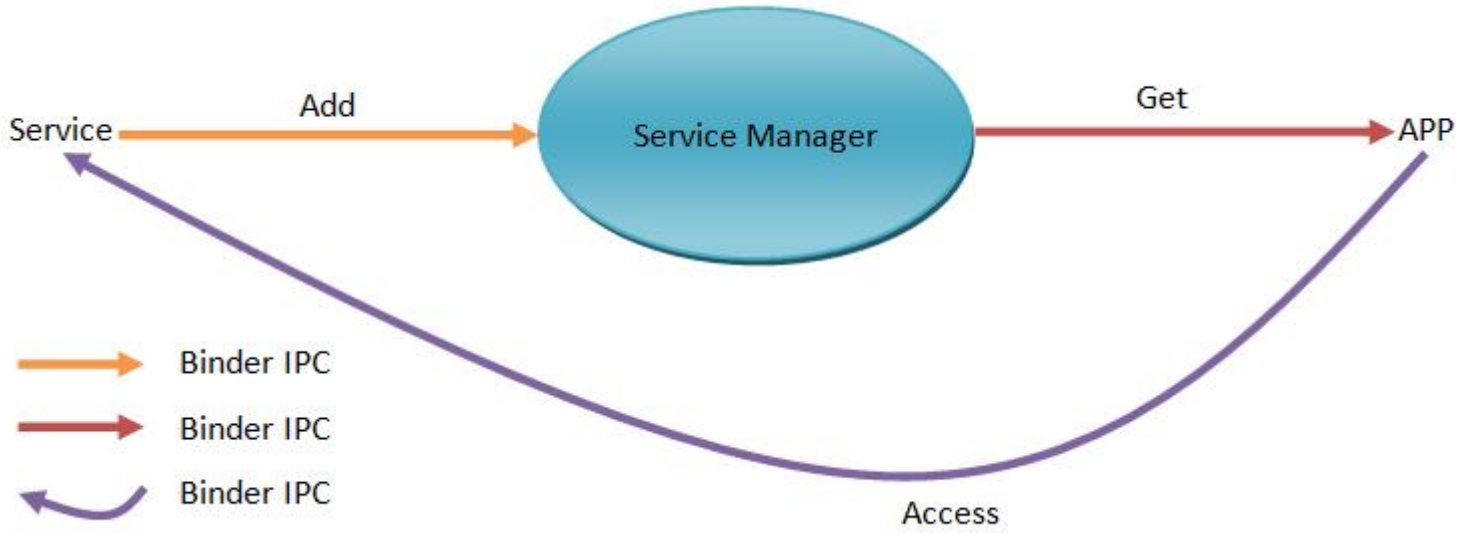
- 按开发语言划分
  - Java Runtime Framework
    - PackageManagerService
    - ActivityManagerService
    - WindowManagerService
    - ……
  - Native Runtime Framework
    - MediaPlayerService
    - SurfaceFlinger
    - AudioFlinger
    - ……

Android应用程序框架(续)

- 按进程划分
  - System Server Process
    - PackageManagerService
    - ActivityManagerService
    - WindowManagerService
    - ……
  - Independent Process
    - SurfaceFlinger
    - MediaPlayerService
    - ……

Android应用程序框架(续)

- 服务注册、获取和访问过程

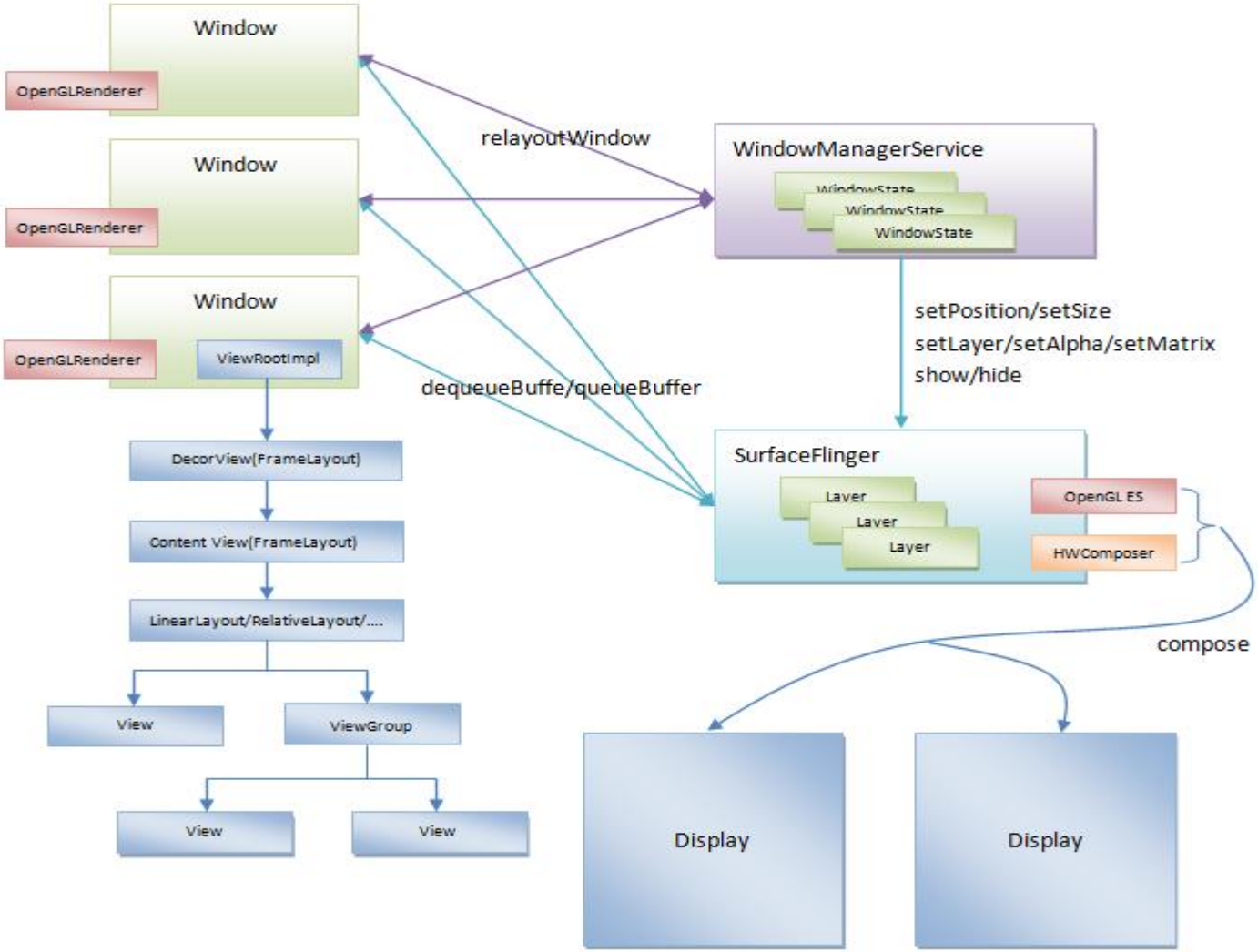


Android用户界面架构

- 窗口管理框架
  - Window
  - WindowManagerService
  - SurfaceFlinger
- 资源管理框架
  - AssetManager
  - Resources

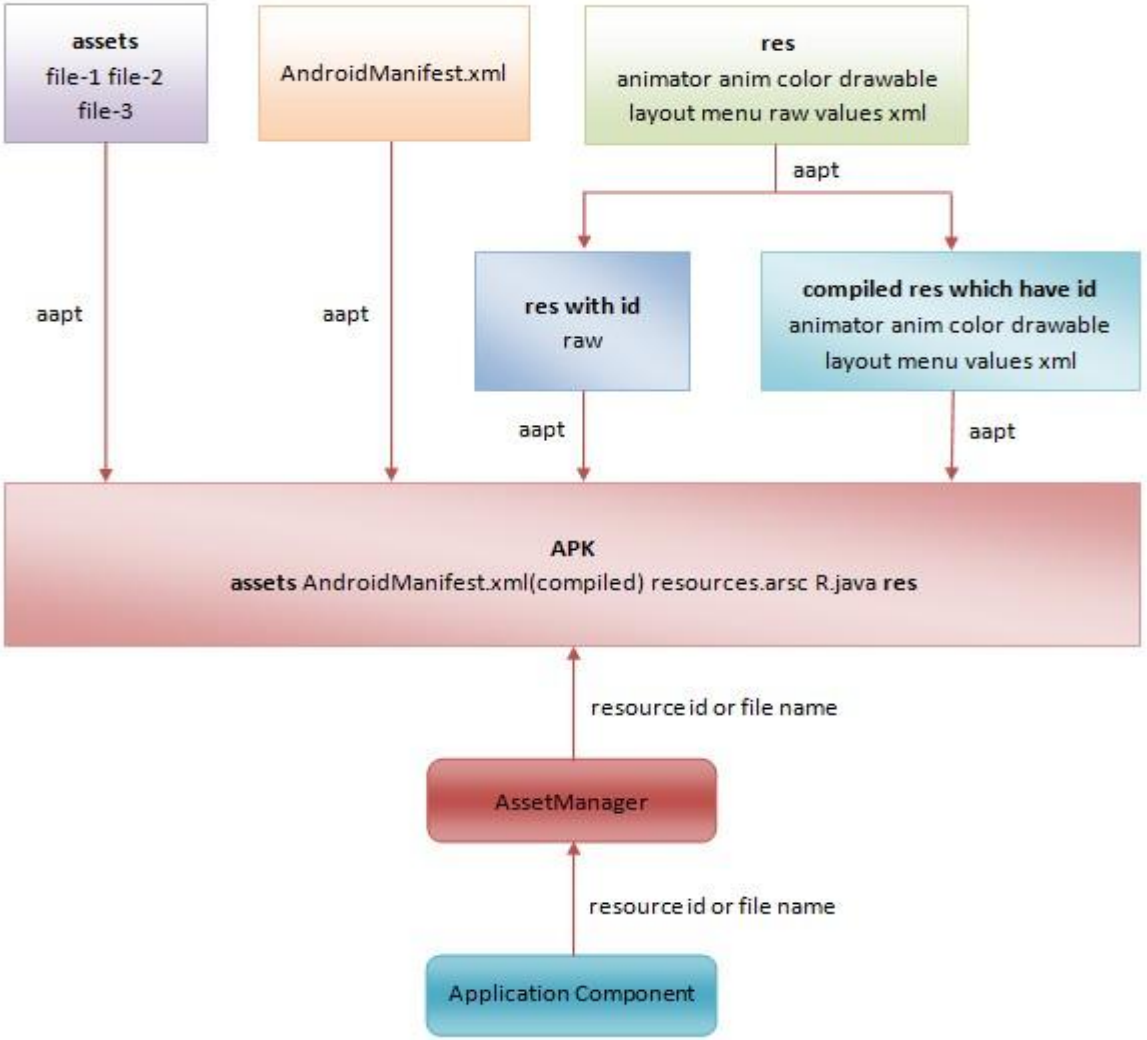
Android用户界面架构(续)

- 窗口管理框架



Android用户界面架构(续)

- 资源管理框架



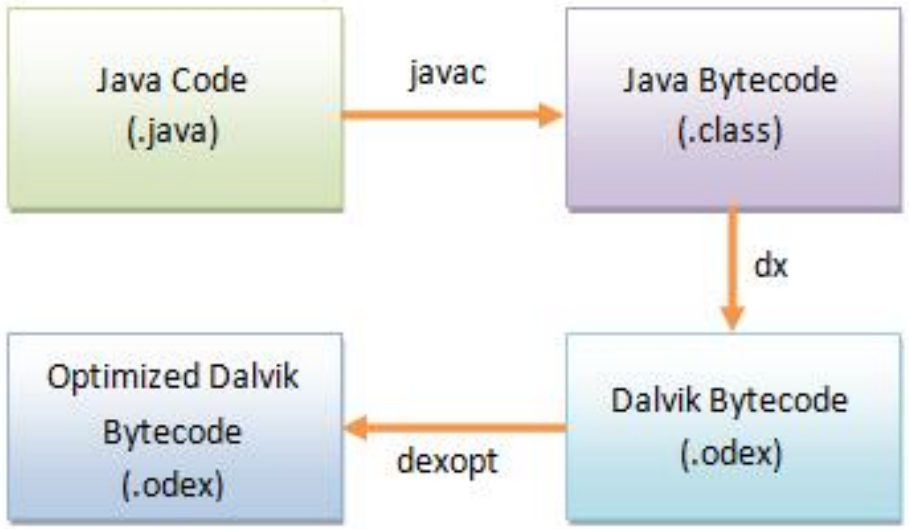
Dalvik虚拟机

- Java虚拟机与Dalvik虚拟机区别

	Java Virtual Machine	Dalvik Virtual Machine
Instruction Set	Java Bytecode (Stack Based)	Dalvik Bytecode (Register Based)
File Format	.class file (one file, one class)	.dex file (one file, many classes)

Dalvik虚拟机(续)

- Dex文件编译和优化



Dalvik虚拟机(续)

- 内存管理
  - Java Object Heap
    - 大小受限，16M/24M/32M/48M
  - Bitmap Memory(External Memroy)：
    - 大小计入Java Object Heap
  - Native Heap
    - 大小不受限

Dalvik虚拟机(续)

- 垃圾收集(GC)
  - Mark，使用RootSet标记对象引用
  - Sweep，回收没有被引用的对象
- GingerBread之前
  - Stop-the-word，也就是垃圾收集线程在执行的时候，其它的线程都停止
  - Full heap collection，也就是一次收集完全部的垃圾
  - 一次垃圾收集造成的程序中止时间通常都大于100ms
- GingerBread之后
  - Cocurrent，也就是大多数情况下，垃圾收集线程与其它线程是并发执行的
  - Partial collection，也就是一次可能只收集一部分垃圾
  - 一次垃圾收集造成的程序中止时间通常都小于5ms

Dalvik虚拟机(续)

- 即时编译(JIT)
  - 从2.2开始支持JIT，并且是可选的，编译时通过WITH_JIT宏进行控制
  - 基于执行路径(Executing Path)对热门的代码片断进行优化(Trace JIT)，传统的Java虚拟机以Method为单位进行优化(Method JIT)
  - 可以利用运行时信息进行激进优化，获得比静态编译语言更高的性能
  - 实现原理：http://blog.reverberate.org/2012/12/hello-jit-world-joy-of-simple-jits.html

Dalvik虚拟机(续)

- Java本地调用(JNI)
  - 实现Java与C/C++代码互调
  - 大部分Java接口的都是通过JNI调用C/C++接口实现的
  - 提供有NDK进行JNI开发
- 进程和线程管理
  - 与Linux进程和线程一一对应
  - 通过fork系统调用创建进程
  - 通过pthread库接口创建线程

Q&A

Thank You