# Android

## 系统架构

应用程序、应用程序框架、核心类库、Linux内核



### 应用程序

### 应用程序框架

所有的应用程序都由一系列的服务和系统组成，包括：

1. **一个可扩展的视图 Views**
2. **内容管理器 Content Providers**

使得应用程序可以访问另一个应用程序的数据，或者共享它们自己的数据

1. **资源管理器 Resource Manager**

非代码资源的访问，如本地字符串、图形和layout files

1. **通知管理器 Notification Manager**

使得应用程序可以在状态栏中显示客户通知信息

1. **活动类管理器 Activity Manager**

管理应用程序生命周期并提供常用的导航回退功能

### Android程序库

1. 系统C库

从BSD继承来的标准C系统函数库，为基于embedded Linux的设备定制

1. 媒体库

基于PacketVideo OpenCORE。支持录放，并且可以录制许多流行的音频视频格式以及静态映像文件

1. Surface Manager

对显示子系统的管理，并且为多个应用程序提供2D和3D图层的无缝融合

1. LibWebCore

一个最新的Web浏览器引擎，用来支持Android浏览器和一个可嵌入的Web视图

1. SGL

内置的2D图形引擎

1. 3D libraries

基于OpenGL的实现，可以使用硬件3D加速

1. FreeType

位图和向量字体显示

1. SQLite

对所有应用程序可用、功能强劲的轻型数据库引擎

### Android运行库

包括一个核心库，提供了Java编程语言核心库的大多数功能。

每一个Android应用程序都在它自己的进程中进行，都拥有一个独立的Dalvik虚拟机。Dalvik是针对同时高效地运行多个VM实现的，执行.dex的Dalvik可执行文件，该类文件针对最小内存使用做了优化。该虚拟机是**基于寄存器**的，所有的类都经Java汇编器编译，然后通过SDK中的DX工具转化为.dex格式由虚拟机执行。

Dalvik虚拟机依赖于Linux的一些功能，如线程机制和底层内存管理机制。

### Linux内核

Android的核心系统服务依赖于Linux内核，如安全性、内存管理、进程管理、网络协议栈和驱动模型，Linux内核也作为硬件和软件栈之间的硬件抽象层。

## 四大组件

### Activity

一般代表手机的一屏。

使用Intent来实现Activity之间的切换。

### Service

是没有界面的长生命周期的代码。

开启Service的两种方式：

1. **Context.startService()**

→onCreate()

→onStart()

//需要显示调用stopService进行关闭

1. **Context.bindService()**

→onCreate()

→onBind() 会返回IBind接口实例，允许客户端回调Service的方法

//与调用者共存亡

### BroadcastReceiver

Broadcast是一种广泛运用的在应用程序之间传输信息的机制。BroadcastReceiver是对Broadcast进行过滤接收并相应的一类组件。

注册方式：

1. **在AndroidManifest.xml中进行注册**

无论应用程序开关，都能接收到广播出来的对象

1. **代码中注册**

不执行unregisterReceiver()，与静态注册效果一致

### ContentProvider

第三方应用数据的访问方案。应用想对外提供的数据，可以通过派生Content Provider类，封装成一枚Content Provider，使用一个uri作为独立的标识，形如：content://com.XX