Android源码解析

笔记本： Android
创建时间： 2015/12/15 14:55
标签： Java

Android源码解析

2015年12月15日
14:55

1，android安全机制

为了更好地保护系统私有数据的安全、完整及版权信息，Android 在系统的不同层面提供了多种安全机制，其中 Android 的主要安全机制包括 Java 混淆器、接入权限、数字证书、SSL、数据库安全、虚拟机（安全沙箱）、文件访问控制等。 Android 的安全框架如图 11-1 所示。

图 11-1　Android 的安全框架

1，接入权限

在 Android 中，接入权限分为 4 个等级，即 normal、dangerous、signature、signatureOr-System 等。权限的等级决定了进行安全保护的级别。其中 normal 权限不会给用户带来实质性的伤害，如调整背光等动作适用该权限；dangerous 权限可能会给用户带来潜在的伤害，如读取电话簿、联网等动作，系统在安装应用时发现应用需要该权限会提示用户；signature 权限要求具有同一签名的应用间才能相互访问；signatureOrSystem 权限主要被设备商使用。

框架层的接入权限定义在 frameworks\base\core\res\ AndroidManifest.xml 中。

创建接入权限：

创建接入权限非常简单，不过需要注意，和常规的理解不同，高级别权限并不兼容低级别权限，比如，对于 normal 接入权限，如果在两个拥有同样数字证书签名的应用间使用，则会导致失败。创建一个接入权限的方法如下：

```
<permission android:name="android.permission.GET_ACCOUNTS"
        android:permissionGroup="android.permission-group.ACCOUNTS"
        android:protectionLevel="normal"
        android:description="@string/permdesc_getAccounts"
        android:label="@string/permlab_getAccounts" />
```

创建一个接入权限组的方法如下：

```
<permission-group android:name="android.permission-group.STORAGE"
        android:label="@string/permgrouplab_storage"
        android:description="@string/permgroupdesc_storage" />
```

2，数字证书

在 Android 中，每个应用在发布时均必须拥有自己的数字证书签名，这个数字证书签名用于在应用的作者和应用程序之间建立信任关系。

事实上，数字证书签名一直存在，即使是在通过 SDK 进行开发的阶段，每次运行应程序时 SDK 均会自动生成一个用于调试模式的数字证书签名。只有在应用正式发布时，才会用到发布模式的数字证书签名。

数字证书签名机制有利于程序升级，除了判断包名外，只有当新版应用和旧版应用的数字签名相同时，Android 才会认为这两个程序是同一个应用的不同版本。另外

2，Android启动流程

1，系统启动流程

在 Android 中，在 BootLoader 加载系统映像后，会通过 system\core\rootdir\ 目录下的 init.rc 脚本进行初始化配置。在 init.rc 中可以配置系统时区、设置日志等级、设置全局环境变量、挂载文件系统、初始化网络配置、配置系统属性、启动守护进程等，具体的实现过程如图 14-1 所示。

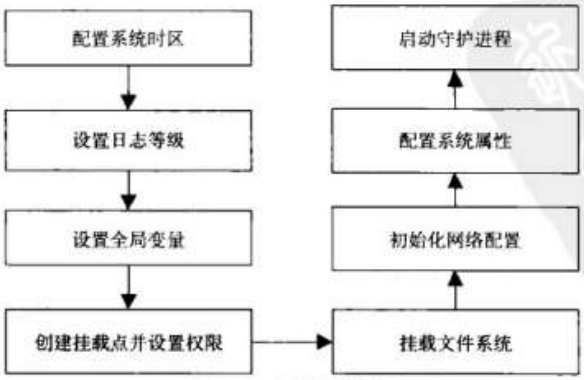

图 14-1　系统启动过程

2，APK应用启动流程

通过 WinRAR 查看 APK 包可以看到，APK 包大致包括 res、AndroidManifest.xml、classes.dex、resource.arsc、META-INF 和 libs 等几项。其中 res 文件夹包含的是资源文件，res 文件夹下的 XML 是已经编译过的，减小了存储空间。AndroidManifest.xml 为应用的配置文件，但是已经编译过。而 classes.dex 为应用的 DEX 字节码文件，是 CLASS 字节码去除冗余信息后的集合。resource.arsc 为资源文件，在 Android 2.3 前，采用的是 UTF-16 编码，在 Android 2.3 及以后版本中，采用的是 UTF-8 编码，这样的改变意味着在 Android 2.3 及以后版本中，基于英文应用的 APK 包会比原来小些，而基于中文应用的 APK 包会比原来大些。这是因为在 UTF-16 编码中，中文和英文均采用 2 字节的编码，而在 UTF-8 中，英文占用 1 字节，而中文要占用 3 字节。在 META-INF 中存放的是 Android 的数字签名证书。在 libs 中存放的通常是 JAR 和原生代码生成的共享库文件。

注意　如果不希望应用在系统低内存时被系统销毁，需将 application 标签的 android:persistent 属性设置为 true。

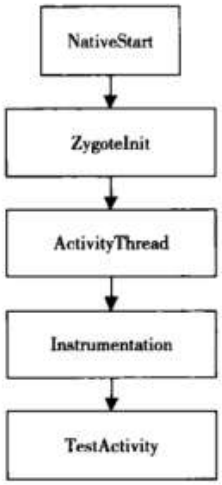

图 14-4　应用的启动过程

Android 进程分为 zygote 进程和普通进程。当启动一个应用时，Dalvik 虚拟机会先通过 NativeStart 接口初始化 JNI，为通过 zygote 进程创建新进程做好准备。原生代码的启动位于

3，Android系统管理

1，内存管理

1，引用对象

在创建进程时，Dalvik 虚拟机会为每个进程分配一定量的堆内存。占用内存较多的程序很容易引起 OutOfMemoryError 等异常。

从 JDK 1.2 开始，Java 对象的引用被分为强引用（HardReference）、弱引用（WeakReference）、软引用（SoftReference）和虚引用（PhantomReference）等 4 个级别。

强引用表示即使虚拟机内存“吃紧”抛出了 OutOfMemoryError 异常，该类型的对象也不会被回收；软引用表示在虚拟机内存“吃紧”抛出 OutOfMemoryError 异常前，该类型的对象就会被回收；弱引用更适合那些数量不多但体积较庞大的对象，弱引用对象最容易被回收；虚引用一般没有实际意义，仅观察 GC 的活动状态，对于测试比较实用，同时必须和引用队列（ReferenceQueue）一起使用。

创建引用队列：

```
ReferenceQueue<String> rq=new ReferenceQueue<String>();
```

创建一个弱引用：

```
WeakReference<String>  wf=new WeakReference<String>(str,rq);
```

2，垃圾回收策略

Android中，每个应用占据一个虚拟机，垃圾回收是基于应用的。在应用层，可以通过System.gc()调用垃圾回收器。Android原生代码中是通过计数机制进行垃圾回收，强引用计数为0时，对象将自动释放。

2，应用管理

需要注意，为了设计的方便，声明的功能可以包括硬件和软件两个方面，表 15-1 所示是 uses-feature 标签的详细说明。

表 15-1　uses-feature 标签的详细说明

功　能	说　明
android.hardware.audio.low_latency	低延迟音频设备
android.hardware.bluetooth	蓝牙
android.hardware.camera	摄像头
android.hardware.camera.autofocus	自动对焦
android.hardware.camera.flash	Flash 调用摄像头
android.hardware.camera.front	前置摄像头
android.hardware.location	定位
android.hardware.location.network	蜂窝网络定位
android.hardware.location.gps	GPS 定位
android.hardware.microphone	麦克风
android.hardware.nfc	NFC
android.hardware.sensor.accelerometer	加速度传感器
android.hardware.sensor.barometer	压力传感器
android.hardware.sensor.compass	罗盘传感器
android.hardware.sensor.gyroscope	陀螺仪传感器
android.hardware.sensor.light	光线传感器
android.hardware.sensor.proximity	距离传感器
android.hardware.telephony	电话
android.hardware.telephony.cdma	CDMA
android.hardware.telephony.gsm	GSM
android.hardware.touchscreen	触摸屏
android.hardware.touchscreen.multitouch	多点触摸
android.hardware.touchscreen.multitouch.distinct	两点触摸
android.hardware.touchscreen.multitouch.jazzhand	五点触摸
android.hardware.wifi	WiFi
android.software.live_wallpaper	动态壁纸
android.software.sip	SIP
android.software.sip.voip	网络电话