



# 计算机安全与维护

## 第八章 智能手机系统的安全与维 护基础



# 本章内容提要

- Android操作系统介绍
- Android的系统文件夹结构
- AndroidManifest.xml文件的解析
- Android系统的安全与权限
- Android系统的用户管理
- Android系统的软件安全
- Android应用程序下载源及安全机制



# 当前主流的智能手机操作系统

symbian  
OS



Windows  
Mobile

BlackBerry OS



MeeGo™



# Android手机操作系统介绍

- Android是一种基于Linux的自由及开放源代码的操作系统，主要使用于便携设备，如智能手机和平板电脑
- Android操作系统最初由Andy Rubin开发，2005年由Google收购注资，并组建开放手机联盟开发改良，逐渐扩展到平板电脑及其他领域
- 2008年10月第一部Android智能手机发布
- 2011年第一季度，Android在全球的市场份额首次超过塞班系统，跃居全球第一



# Android手机操作系统版本演变



ANDROID



**Cupcake**  
Android 1.5



**Donut**  
Android 1.6



**Eclair**  
Android 2.0/2.1



**Froyo**  
Android 2.2



**Gingerbread**  
Android 2.3



**Honeycomb**  
Android 3.0



**Ice Cream Sandwich**  
Android 4.0



**Jelly Bean**  
Android 4.1  
& Android 4.2



# Android手机操作系统版本演变

- Android在正式发行之前，最开始拥有两个内部测试版本，并且以著名的机器人名称来对其进行命名，它们分别是：阿童木（Android Beta），发条机器人（Android 1.0）
- Android 1.1：2008年9月发布的Android第一版
- 后来由于涉及到版权问题，谷歌将其命名规则变更为用甜点作为它们系统版本的代号的命名方法
- 甜点命名法开始于Android 1.5发布的时候



# Android手机操作系统版本演变

- Android 1.5 Cupcake (纸杯蛋糕) : 2009年4月30日发布
- Android 1.6 Donut (甜甜圈) : 2009年9月15日发布
- Android 2.0/2.0.1/2.1 Eclair (松饼) : 2009年10月26日发布
- Android 2.2/2.2.1 Froyo (冻酸奶) : 2010年5月20日发布



# Android手机操作系统版本演变

- Android 2.3.x Gingerbread (姜饼) : 2010年12月7日发布
- Android 3.0 Honeycomb (蜂巢) : 2011年2月2日发布
- Android 4.0 Ice Cream Sandwich (冰激凌三明治) : 2011年10月19日在香港发布
- Android 4.1 Jelly Bean (果冻豆) : 2012年6月28日发布



# Android操作系统的优劣势

## • 开放性

- 开发的平台允许任何移动终端厂商加入到Android联盟中来
- 显著的开放性可以使其拥有更多的开发者，随着用户和应用的日益丰富，一个崭新的平台也将很快走向成熟
- 开发性对于Android的发展而言，有利于积累人气。



# Android操作系统的优劣势

- 挣脱运营商的束缚

- 手机应用往往受到运营商制约，使用什么功能接入什么网络，几乎都受到运营商的控制
- 随着EDGE、HSDPA这些2G至3G移动网络的逐步过渡和提升，手机随意接入网络已不是运营商口中的笑谈
- 互联网巨头Google推动的Android终端天生就有网络特色，将让用户离互联网更近



# Android操作系统的优劣势

- 丰富的硬件选择
  - 由于Android的开放性，众多的厂商会推出千奇百怪，功能特色各具的多种产品
  - 功能上的差异和特色，却不会影响到数据同步、甚至软件的兼容
- 不受任何限制的开发商
  - Android平台提供给第三方开发商一个十分宽泛、自由的环境，不会受到各种条条框框的阻扰



# Android操作系统的优 势

- 无缝结合的Google应用

—在互联网的Google已经走过10年度历史，从搜索巨人到全面的互联网渗透，Google服务如地图、邮件、搜索等已经成为连接用户和互联网的重要纽带，而Android平台手机将无缝结合这些优秀的Google服务



# 本章内容提要

- Android操作系统介绍
- Android的系统文件夹结构
- AndroidManifest.xml文件的解析
- Android系统的安全与权限
- Android系统的用户管理
- Android系统的软件安全
- Android应用程序下载源及安全机制



# Android的系统文件夹结构

- 根目录下的文件夹结构 (Android版本4.0.3)

acct	11 3月 13 08:46:00	rwxr-xr-x
cache	24 9月 12 22:54:00	rwxrwx---
config	11 3月 13 08:46:00	r-x-----
custom	01 1月 70 08:00:00	rwxr-xr-x
d	11 3月 13 08:46:00	rwxrwxrwx -> debug
data	26 12月 12 16:56:00	rwxrwx--x
default.prop	01 1月 70 08:00:00	rw-r--r-- 116 Bytes
dev	11 3月 13 08:46:00	rwxr-xr-x
etc	11 3月 13 08:46:00	rwxrwxrwx -> etc
ext_sdcard	11 3月 13 08:46:00	rwxrwxrwx -> ext_sdcard

proc	01 1月 70 08:00:00	r-xr-xr-x
root	05 7月 12 01:10:00	rwx-----
sbin	01 1月 70 08:00:00	rwxr-x---
sdcard	11 3月 13 08:46:00	rwxrwxrwx -> sdcard
sys	11 3月 13 08:46:00	rwxr-xr-x
system	11 12月 12 15:51:00	rwxr-xr-x
ueventd.goldfish.rc	01 1月 70 08:00:00	rw-r--r-- 272 Bytes
ueventd.omap4blazeboard.rc	01 1月 70 08:00:00	rw-r--r-- 448 Bytes
ueventd.rc	01 1月 70 08:00:00	rw-r--r-- 3.74K
vendor	11 3月 13 08:46:00	rwxrwxrwx -> vendor



# Android的系统文件夹结构

- cache: 缓存临时文件夹
- sdcard: SD卡中的FAT32文件系统挂载目录
- system: 放置系统中的大部分内容，后面详细介绍
- sys: 用于挂载sysfs文件系统，sysfs文件系统用来表示设备的结构，将设备的层次结构反映到用户空间中



# Android的系统文件夹结构

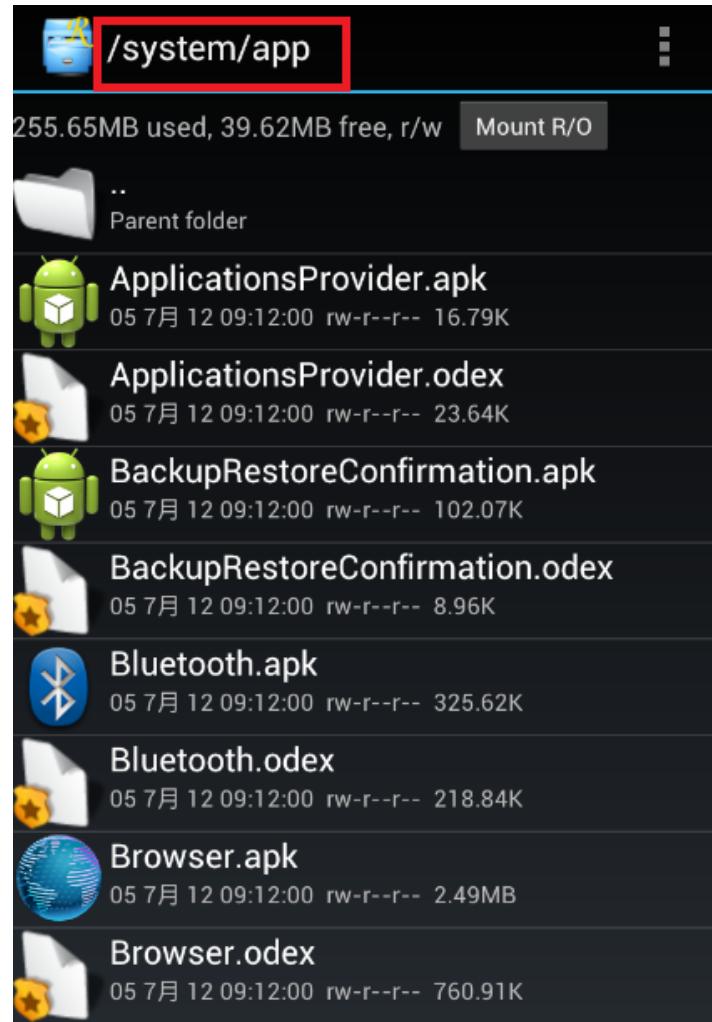
- sbin: 只放了一个用于调试的adb程序
- proc: 存放多种文件提供的系统信息，可以在整个系统范围的上下文中使用
- data: 存放用户安装的软件以及各种数据，后面会详细介绍
- dev: 设备节点文件的存放地



# Android的系统文件夹结构

- **/system/app**

- 这个里面主要存放的是常规下载的应用程序，可以看到都是以APK格式结尾的文件
- 在这个文件夹下的程序为系统默认的组件

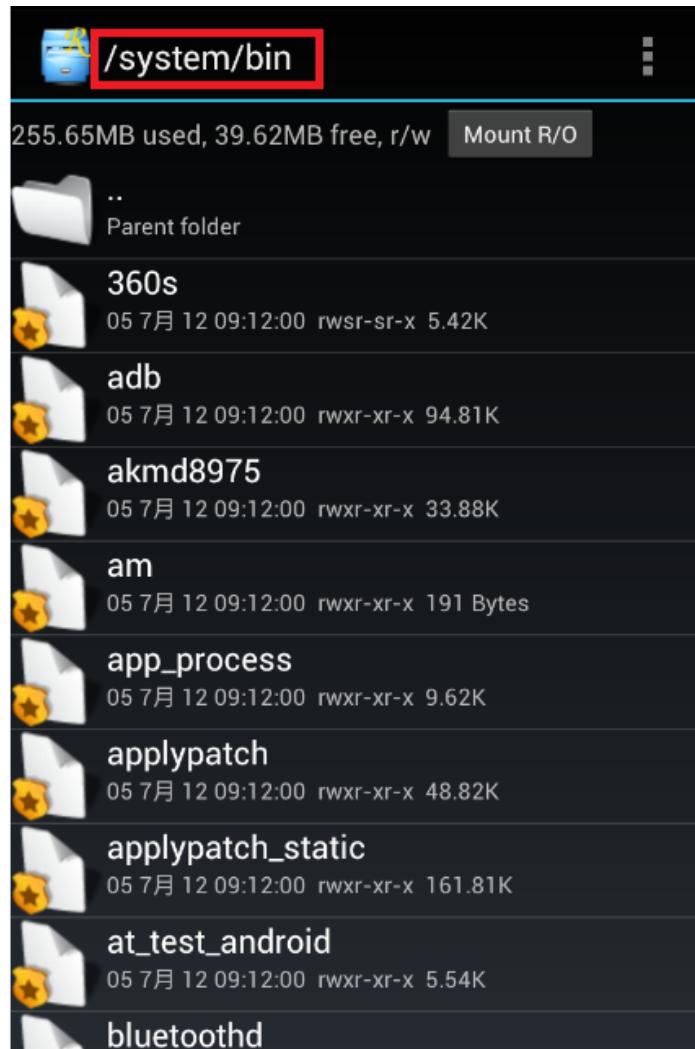




# Android的系统文件夹结构

- **/system/bin**

—这个目录下的文件都是系统的本地程序，从bin文件夹名称可以看出是binary二进制的程序，里面主要是Linux系统自带的组件(命令)

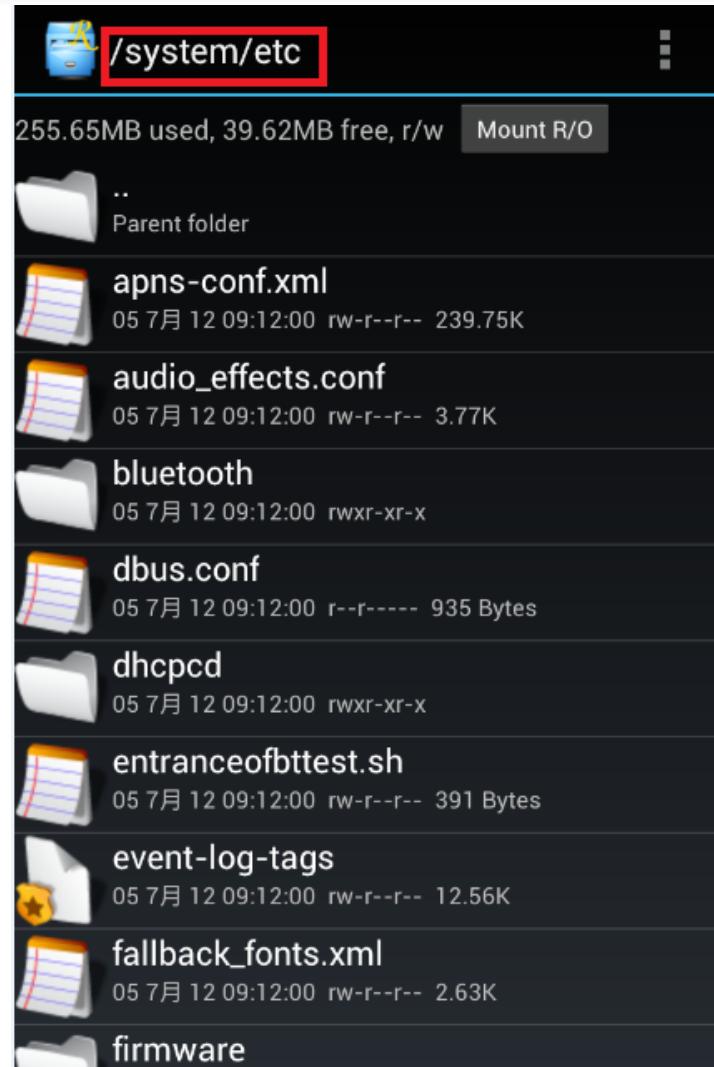




# Android的系统文件夹结构

- **/system/etc**

—从文件夹名称来看保存的都是系统的配置文件，比如APN接入点设置等核心配置

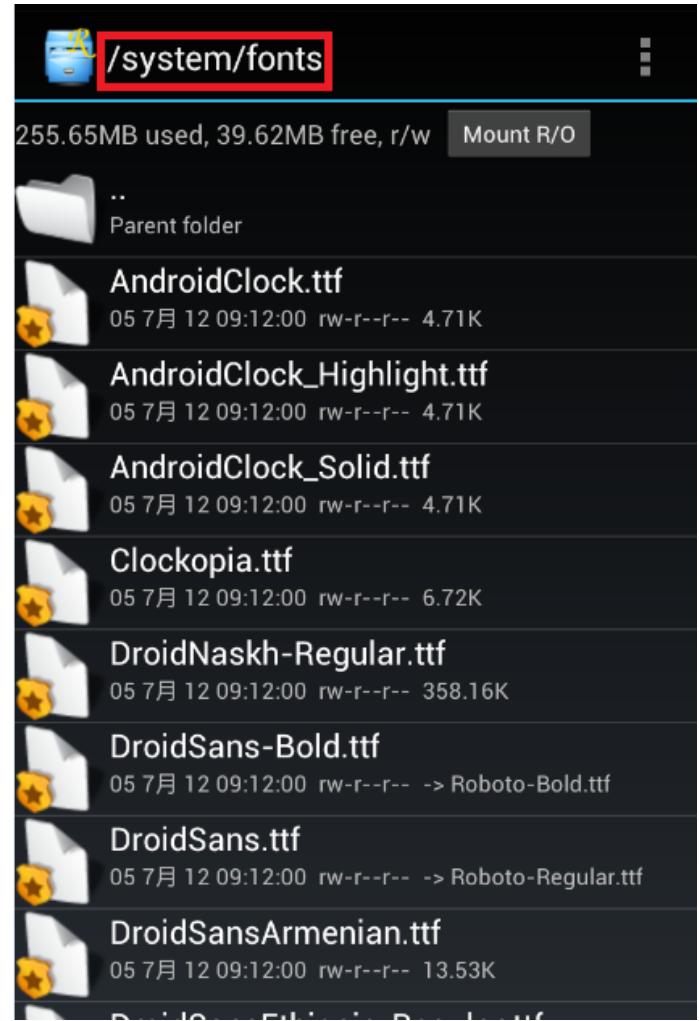




# Android的系统文件夹结构

- **/system/fonts**

—字体文件夹，除了标准字体和粗体、斜体外可以看到文件体积最大的可能是中文字库，或一些unicode字库





# Android的系统文件夹结构

- /system/framework

—framework主要是些核心的文件，从后缀名为jar可以看出是系统平台框架

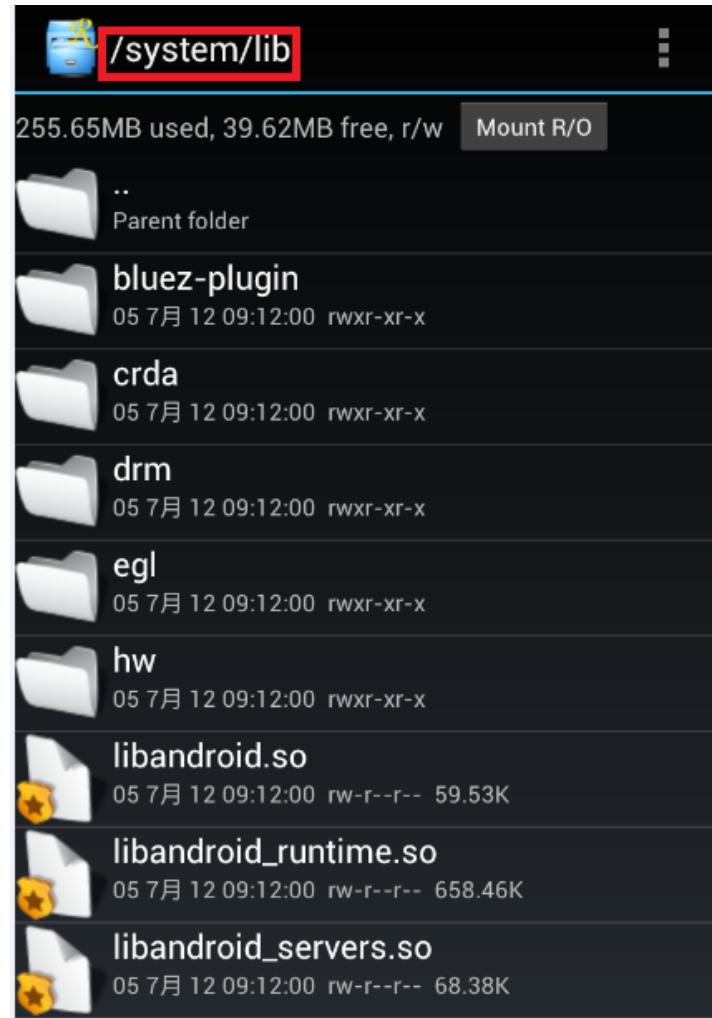




# Android的系统文件夹结构

- **/system/lib**

—lib目录中存放的主要是系统底层库，一些so文件，如平台运行时库

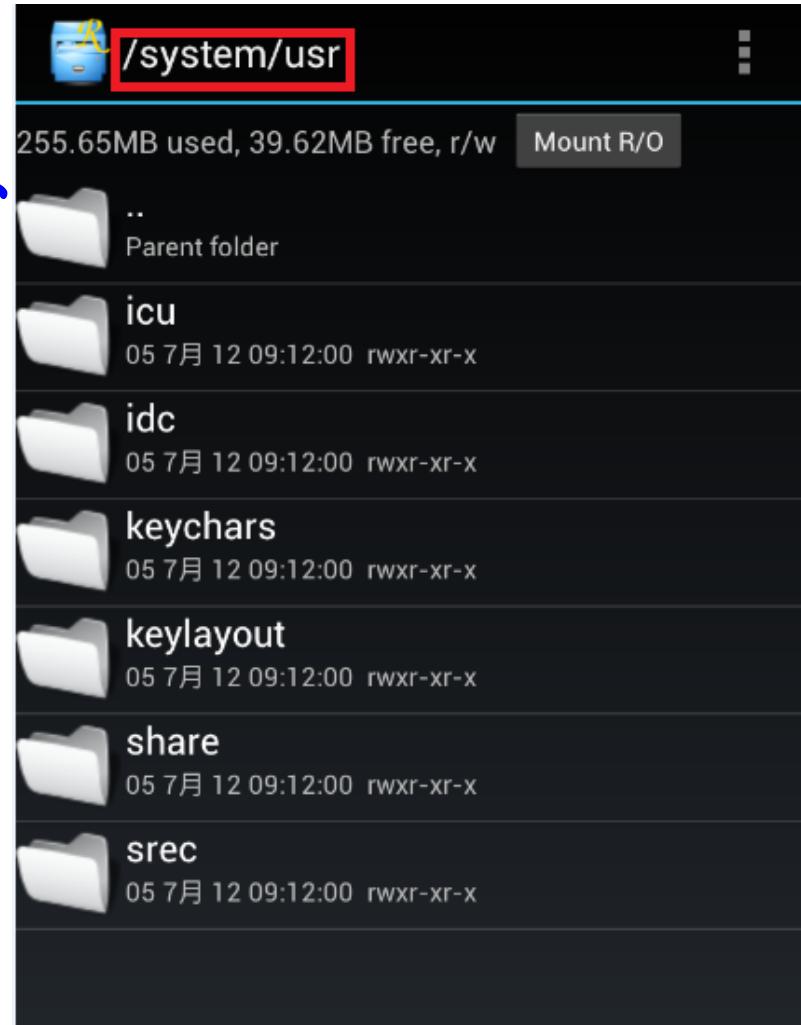




# Android的系统文件夹结构

- **/system/usr**

—用户文件夹，包含共享、  
键盘布局、时间区域文  
件等

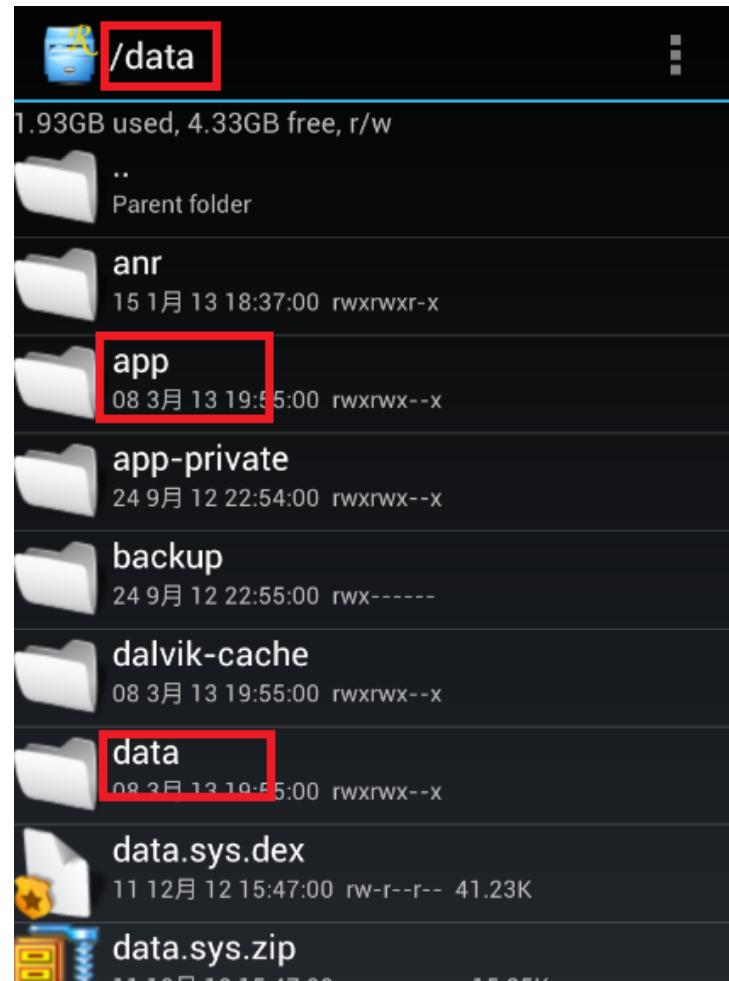




# Android的系统文件夹结构

- **data文件夹**

—存放的是用户的软件信息（非自带rom安装的软件）





# Android的系统文件夹结构

- /data/app  
—存放用户安装的软件

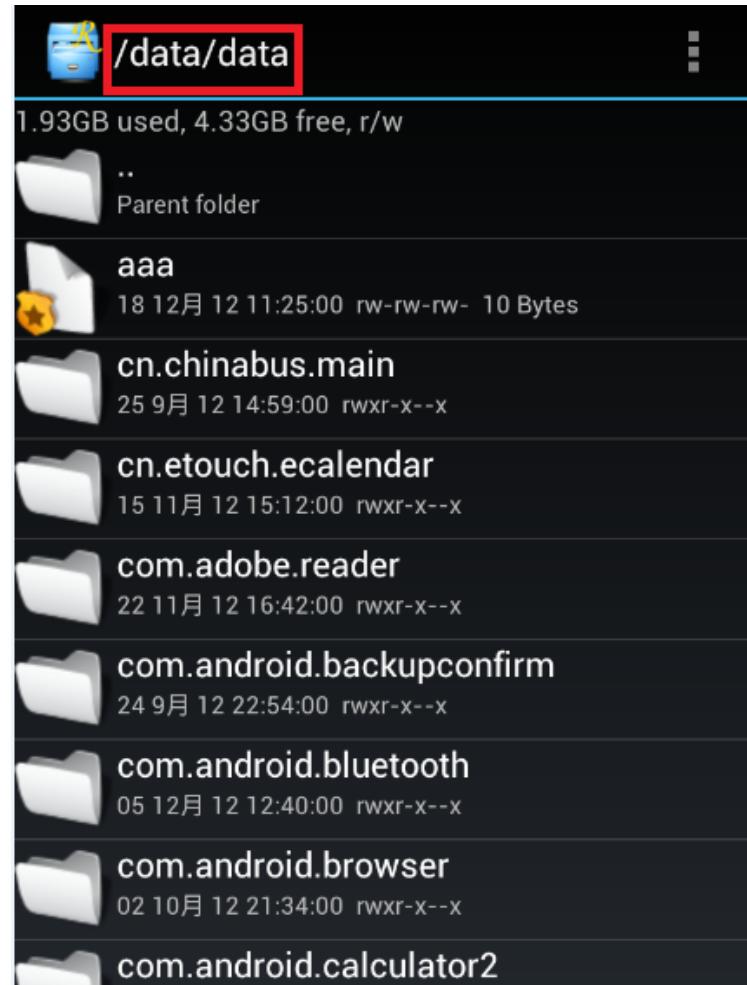




# Android的系统文件夹结构

- **/data/data**

—存放所有软件（包括  
/system/app 和  
/data/app 和 /mnt/asec  
中装的软件）的一些lib  
和xml文件等数据信息

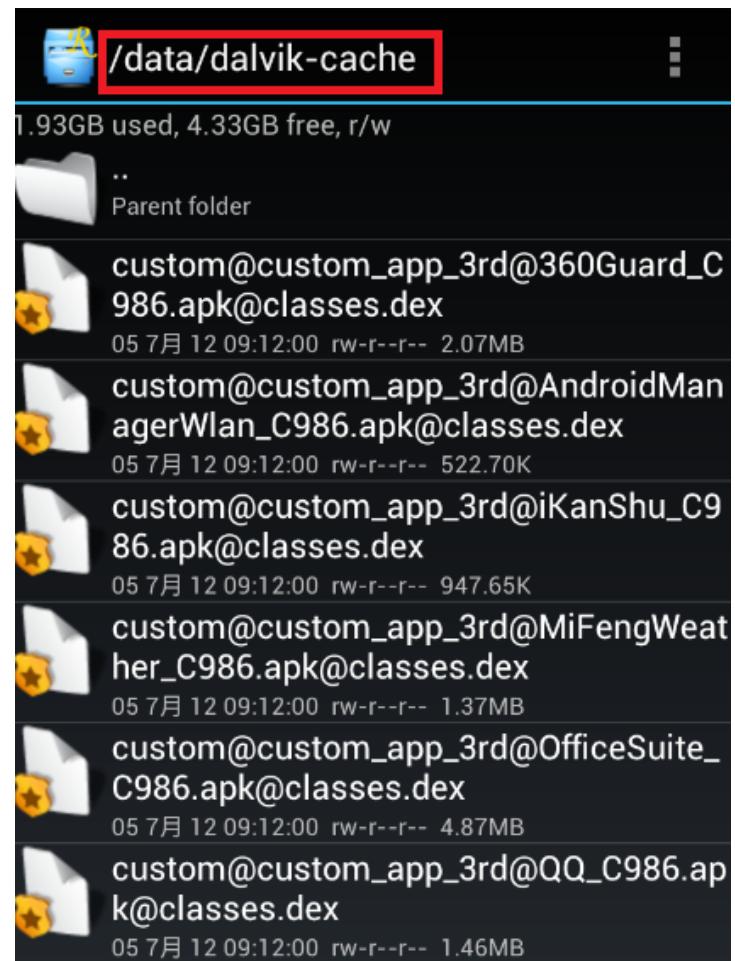




# Android的系统文件夹结构

- **/data/dalvik-cache**

—存放程序的缓存文件，  
这里的文件都是可以删除的





# 本章内容提要

- Android操作系统介绍
- Android的系统文件夹结构
- AndroidManifest.xml文件的解析
- Android系统的安全与权限
- Android系统的用户管理
- Android系统的软件安全
- Android应用程序下载源及安全机制



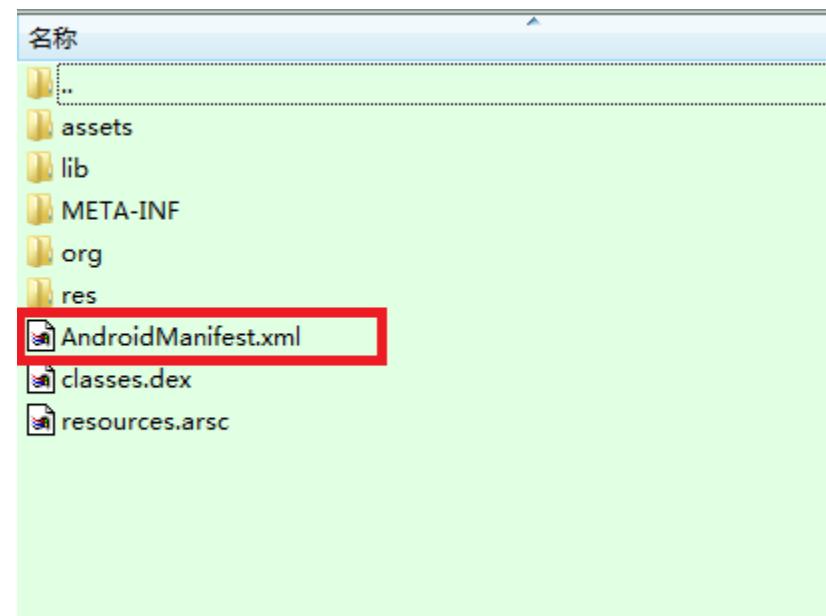
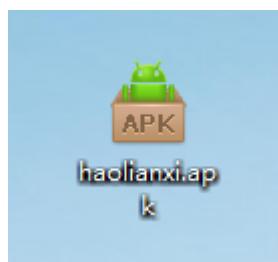
# AndroidManifest.xml文件的解析

- AndroidManifest.xml文件是Android程序的全局配置文件，是每个 android程序中必须的文件
- 位于我们开发的应用程序的根目录下，描述了 package 中的全局数据，包括 package 中暴露的组件 (activities, services, 等等)，以及他们各自的实现类，各种能被处理的数据和启动位置等重要信息
- 声明应用程序必须拥有哪些权限和许可，以便访问 API 的被保护部分，以及与其他应用程序交互



# AndroidManifest.xml文件的解析

- 查看AndroidManifest.xml文件的方法
  - Apk安装文件使用解压缩工具打开
  - 目录中有AndroidManifest.xml文件，把它提取出来，但是文件本身是压缩的，不可以直接查看

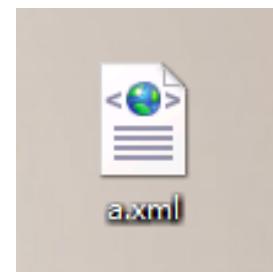




# AndroidManifest.xml文件的解析

- 使用java组件AXMLPrinter2.jar来解析该文件
- 使用以下命令来输出解析后的xml文件，输出的文件名为a.xml

```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7600]
版权所有 <c> 2009 Microsoft Corporation。保留所有权利。
C:\Users\santiago>java -jar C:\Users\santiago\Desktop\AXMLPrinter2.jar C:\Users\santiago\Desktop\AndroidManifest.xml > C:\Users\santiago\Desktop\a.xml
C:\Users\santiago>
```





# AndroidManifest.xml文件的解析

- 解析出来的xml文件结构如下图所示

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest package="com.cleaderwin.contact" android:installLocation="0" android:versionName="3.1.3888" android:versionCode="7" xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-sdk android:targetSdkVersion="8" android:minSdkVersion="7"> </uses-sdk>
    <uses-feature android:name="android.hardware.camera"> </uses-feature>
    <uses-permission android:name="android.permission.CAMERA"> </uses-permission>
    <uses-permission android:name="android.permission.INTERNET"> </uses-permission>
    <uses-permission android:name="android.permission.GET_TASKS"> </uses-permission>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"> </uses-permission>
    <uses-permission android:name="android.permission.READ_CONTACTS"> </uses-permission>
    <uses-permission android:name="android.permission.WRITE_CONTACTS"> </uses-permission>
    <uses-permission android:name="android.permission.WRITE_CALENDAR"> </uses-permission>
    <uses-permission android:name="android.permission.CALL_PHONE"> </uses-permission>
    <uses-permission android:name="android.permission.SEND_SMS"> </uses-permission>
    <uses-permission android:name="android.permission.READ_SMS"> </uses-permission>
    <uses-permission android:name="android.permission.RECEIVE_SMS"> </uses-permission>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"> </uses-permission>
    <uses-permission android:name="android.permission.INTERNET"> </uses-permission>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"> </uses-permission>
    <uses-permission android:name="android.permission.MODIFY_PHONE_STATE"> </uses-permission>
    <uses-permission android:name="android.permission.MANAGE_ACCOUNTS"> </uses-permission>
    <uses-permission android:name="android.permission.ACCOUNT_MANAGER"> </uses-permission>
    <uses-permission android:name="android.permission.GET_ACCOUNTS"> </uses-permission>
    <uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS"> </uses-permission>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"> </uses-permission>
    <uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"> </uses-permission>
    <uses-permission android:name="android.permission.READ_LOGS"> </uses-permission>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"> </uses-permission>
    <uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"> </uses-permission>
    <uses-feature android:name="android.hardware.location.gps"> </uses-feature>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"> </uses-permission>
    <uses-permission android:name="android.permission.INTERNET"> </uses-permission>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"> </uses-permission>
    <uses-permission android:name="android.permission.PROCESS_OUTGOING_CALLS"> </uses-permission>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"> </uses-permission>
    <uses-permission android:name="com.android.launcher.permission.READ_SETTINGS"> </uses-permission>
```



# AndroidManifest.xml文件的解析

- uses-permission标签声明了那些由你定义的权限，而这些权限是应用程序正常执行所必需的
- 在安装程序的时候，你设定的所有权限将会告诉给用户，由他们来决定同意与否
- 对很多本地Android服务来说，权限都是必需的，特别是那些需要付费或者有安全问题的服务

```
<uses-permission android:name="android.permission.CAMERA"> </uses-permission>
<uses-permission android:name="android.permission.INTERNET"> </uses-permission>
<uses-permission android:name="android.permission.GET_TASKS"> </uses-permission>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"> </uses-permission>
<uses-permission android:name="android.permission.READ_CONTACTS"> </uses-permission>
<uses-permission android:name="android.permission.WRITE_CONTACTS"> </uses-permission>
```



# AndroidManifest.xml文件的解析

- Permission标签用来自定义声明自己的权限，用来限制应用程序组件的访问权限
- 应用程序组件就可以通过添加android:permission属性来要求这些权限
- 再后，其他的应用程序就需要在它们的清单中包含uses-permission标签(并且通过授权)，之后才能使用这些受保护的组件

```
<permission android:protectionLevel="0x00000002" android:name="com.tencent.mm.oauth.permission.SEND"> </permission>
<permission android:protectionLevel="0x00000002" android:name="com.tencent.mm.plugin.permission.WRITE"> </permission>
<permission android:protectionLevel="0x00000002" android:name="com.tencent.mm.plugin.permission.READ"> </permission>
<permission android:protectionLevel="0x00000002" android:name="com.tencent.mm.plugin.permission.SEND"> </permission>
<permission android:protectionLevel="0x00000002" android:name="com.tencent.mm.permission.MM_MESSAGE"> </permission>
<uses-permission android:name="com.tencent.mm.oauth.permission.SEND"> </uses-permission>
```



# AndroidManifest.xml文件的解析

- Permission-tree, permission-group
  - <permission-group> 就是声明一个标签，该标签代表了一组permissions
  - <permission-tree>是为一组permissions声明了一个namespace
- sharedUserId
  - 表明数据权限，因为默认情况下，Android给每个apk分配一个唯一的UserID，所以是默认禁止不同APK访问共享数据的。若要共享数据，可以采用sharedUserId了，将不同apk的sharedUserId都设为一样，则这些apk之间就可以互相共享数据了



# 本章内容提要

- Android操作系统介绍
- Android的系统文件夹结构
- AndroidManifest.xml文件的解析
- Android系统的安全与权限
- Android系统的用户管理
- Android系统的软件安全
- Android应用程序下载源及安全机制



# Android系统的安全与权限

- Android的安全机制
  - Android的安全机制有三个层次
    - Linux机制
    - Android特有的安全机制
    - 其他的保护机制



# Android系统的安全与权限

- Linux安全机制
  - 在Android Linux内核层包含两个基本的安全机制
    - 可移植操作系统接口（POSIX）用户
    - 文件访问控制
  - 这些机制的基本元素是用户（用户ID），用户拥有对象（进程和文件），用户再进一步分配到组



# Android系统的安全与权限

- POSIX用户
  - 安装在移动设备上的Android包（apk文件）都会分配一个唯一用户ID
  - 两个不同包的代码不能在同一个进程里运行，这就创造了一个沙箱，阻止应用程序对其他程序或系统其它部分施加恶意影响
  - 应用程序不允许访问其他资源，除非明确的授权允许访问



# Android系统的安全与权限

- 通过设置每个Android包的AndroidManifest.xml文件的manifest标签内的sharedUserId属性，给他们配备同一个用户ID
- 这两个包被认为是同一个程序，拥有同样的用户ID和文件存取权限
- 为保证安全，只有两个具有相同签名的应用程序才能分配相同的用户ID



# Android系统的安全与权限

- 文件访问控制
  - Android文件（包括应用程序文件和系统文件）管理属于Linux权限机制
  - 每个文件与用户ID和用户组ID以及Read, Write, Execute权限密切相关
  - Root用户拥有Android系统文件，特定应用程序用户拥有应用程序的文件
  - 把应用程序和系统文件分配给不同的用户，设定合适的文件访问安全权限，除非共用一个用户ID或被设定为全局的可读或者可写



# Android系统的安全与权限

- Apk文件包的权限设置





# Android系统的安全与权限

- Android特有的安全机制
  - 应用程序权限机制
  - 组件包装
  - 数字签名



# Android系统的安全与权限

## • 权限机制

- Android在设备上实施了基于权限的安全策略来处理安全问题，采用权限来限制安装应用程序的能力
- 权限是一段唯一的各不相同的字符串，当某个权限与某个操作和资源对象绑定在一起，必须获得这个权限才能在对象上执行操作
- Android框架提供一套默认的权限存储在 android.manifest.permission 类中，同时允许我们自己定义新的权限



# Android系统的安全与权限

- 设备上安装应用程序时，请求的权限在屏幕上以待用户审查，只有用户同意授权后，程序才会被安装





# Android系统的安全与权限

- 权限策略

- 权限分为两个类别

- 执行程序时被应用程序所请求的权限
    - 应用程序的组件之间通信时被其他组件请求的权限

- 开发者通过AndroidManifest.xml文件中编写权限标签来定义这两种类别的权限策略



# Android系统的安全与权限

- 权限声明
  - 应用程序可以用一个<permission>元素来声明权限，用于限制访问特定组件或应用程序
  - 安装程序时，这个已声明的权限被加入到系统中
- 权限请求
  - 应用程序列出所有需要用来完成任务的权限，分别用<uses-permission>元素标识
  - 这些权限在安装时被请求，同意安装意味着授权所有被请求的权限



# Android系统的安全与权限

- ICC (inter-component communication) 权限保护
  - <application>元素和组件元素都有 android:permission的属性，这个属性分别为应用程序和组件的权限标签
  - 组件启动ICC时，相关的访问控制器就会查看组件和组件所在应用程序的权限标签集合
  - 若目标组件的访问权限标签在集合内时，允许ICC的建立继续进行，否则将会被拒绝



# Android系统的安全与权限

- 防护等级
  - 防护等级是权限的一个属性，决定了权限怎样被授权
  - Android安全架构支持四种防护等级：normal, dangerous, signature, signatureOrSystem
  - 该等级声明在AndroidManifest.xml文件的标签 android: protectionLevel 中

```
<permission android:protectionLevel="0x00000002"
<permission android:protectionLevel="0x00000002"
<permission android:protectionLevel="0x00000002"
<permission android:protectionLevel="0x00000002"
<permission android:protectionLevel="0x00000002"
```



# Android系统的安全与权限

- Normal
  - 不是特别危险的权限拥有，在程序安装时，这些权限隐藏在屏幕折叠的菜单内
- Dangerous
  - 可能提供访问私有数据或有害功能的高危险度的权限拥有，如果没有用户的明确授权，应用程序将不会接受这种权限
  - 程序安装时，这些权限明显的罗列在屏幕上



# Android系统的安全与权限

- Signature
  - 当请求该权限的应用程序与声明该权限的应用程序拥有相同的数字签名证书时，才会授予该权限
- signatureOrSystem
  - 一种特殊的Signature权限，只会授权给安装在Android系统镜像的包



# Android系统的安全与权限

- 组件包装

- Android应用程序把组件包装在程序内容内，阻止其他应用程序访问它们
- 定义组件的exported属性来实现，设置为false时，这个组件只能被拥有它的应用程序访问，设置为true，该组件可以被其他外部程序调用

```
    es="0x000000A0" android:exported="true">> </activity>
    es="0x000000A0" android:exported="true">> </activity>
    le="1" android:exported="true">> </activity>
```



# Android系统的安全与权限

## • 数字签名

- 所有的应用程序都必须有数字证书，Android系统不会安装一个没有数字证书的应用程序
- Android程序包使用的数字证书可以是自签名的，不需要一个权威的数字证书机构签认认证
- 正式发布的Android应用程序，必须使用一个合适的私钥生成的数字证书给程序签名
- 数字证书是有有效期的，Android在应用程序安装时会检查证书的有效期
- 应用程序升级安装时会强制检查新应用程序的数字签名是否和已安装应用的数字签名匹配



# Android系统的安全与权限

- 其他的安全机制
  - 内存管理单元保障每个用户进程都拥有自己独立的地址空间，减少了特权升级的可能性
  - 编程语言的类型安全特性能使编写的应用程序更加安全



# 本章内容提要

- Android操作系统介绍
- Android的系统文件夹结构
- AndroidManifest.xml文件的解析
- Android系统的安全与权限
- Android系统的用户管理
- Android系统的软件安全
- Android应用程序下载源及安全机制



# Android系统的用户管理

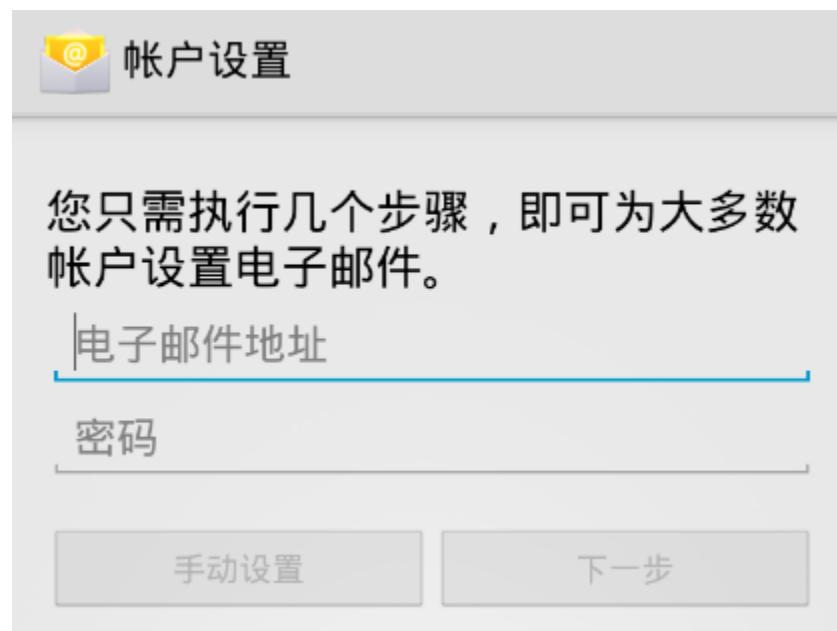
- 这里可以添加Google帐户，用来同步用户数据





# Android系统的用户管理

- 通过添加google电子邮件来绑定帐号





# Android系统的用户管理

- 填写帐户信息

帐户设置

用户名  
xxx@xxx.com

密码  
.....

POP3 服务器  
pop3.xxx.com

端口  
110

安全类型  
无

从服务器中删除电子邮件  
一律不

上一步

下一步



# Android系统的用户管理

- 可以通过网络使用google服务来同步通讯录，邮件，日程安排





# Android系统的用户管理

- Android Root权限用户

- Root是Linux系统的超级管理员用户帐号该帐户拥有系统最高的权限
- Root用户可以查看根目录下的文件，拥有对系统文件的修改，删除权限
- 针对Android用户来说，获取root权限不仅仅意味着用户获得了系统的最高权限，可以“为所欲为”。同时若是对系统本身不是很了解，会成为恶意软件或者病毒入侵提供便利



# Root的好处

- 可以备份系统。
- 使用高级的程序，例如屏幕截图、root explorer等等。
- 修改系统的内部程序。
- 将程序安装到SD卡中（Android2.2以下默认是不支持的）
- 可以使用一些软件的全部功能
- 可以更改主题



# Root的风险

- 如果在ROOT的过程中遇到问题，可能使手机变砖（无法开机）
- 如果不小心安装了恶意软件，可能使手机中的个人隐私信息暴露
- 个别手机厂商会检测用户是否曾进行ROOT操作，这可能与保修有关



# adb shell命令

- adb的全称为(Android Debug Bridge 调试桥) , 通过adb我们可以在Eclipse中方面通过DDMS来调试Android程序
- 管理设备或手机模拟器的状态
- 快速更新设备或手机模拟器中的代码, 如应用或Android 系统升级
- 在设备上运行shell命令
- 管理设备或手机模拟器上的预定端口
- 在设备或手机模拟器上复制或粘贴文件



# Android系统的用户管理

- 如果获得了Root的用户，可以通过adb shell命令来修改系统文件或者安装软件

The screenshot shows a Windows command prompt window titled 'C:\Windows\system32\cmd.exe - adb shell'. The window displays the following text:

```
Microsoft Windows [版本 6.1.7601]
版权所有 <c> 2009 Microsoft Corporation。保留所有权利。

C:\Users\yinyl>adb shell
shell@android:/ $ pwd
pwd
/
shell@android:/ $ ls
ls
acct
cache
config
custom
d
data
default.prop
dev
etc
ext_sdcard
factory
init
init.goldfish.rc
init.omap4blazeboard.rc
init.omap4blazeboard.usb.rc
半:
```

The first few lines of the output ('C:\Users\yinyl>adb shell', 'shell@android:/ \$ pwd', 'pwd', '/') are highlighted with a red rectangular box.



# Android系统的用户管理

- 有权限可以切换超级用户（root），以#符号表示

The screenshot shows a Windows command prompt window titled 'C:\Windows\system32\cmd.exe - adb shell'. The window displays the following text:

```
C:\Windows\system32\cmd.exe - adb shell
Microsoft Windows [版本 6.1.7601]
版权所有 © 2009 Microsoft Corporation。保留所有权利。

C:\Users\yinyl>adb shell
shell@android:/ $ su
su
shell@android:/ #
```

A red box highlights the command 'su' and its output '#', indicating the successful transition to root mode.



# Android系统的用户管理

## • 一键Root工具





# Android系统的用户管理

- 一键root工具获取root权限





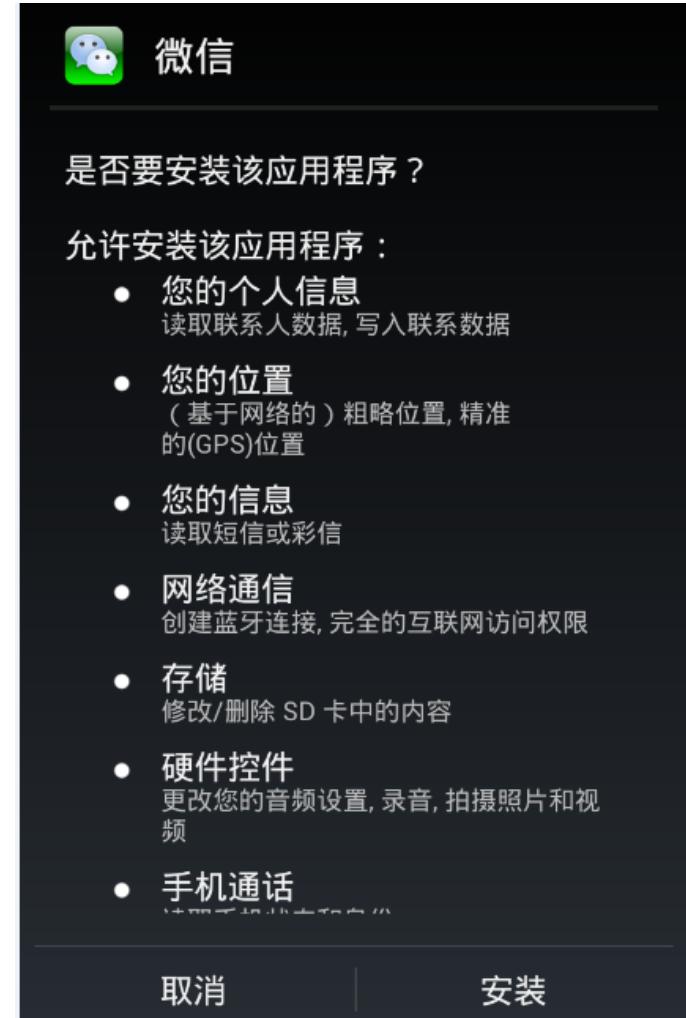
# 本章内容提要

- Android操作系统介绍
- Android的系统文件夹结构
- AndroidManifest.xml文件的解析
- Android系统的安全与权限
- Android系统的用户管理
- Android系统的软件安全
- Android应用程序下载源及安全机制



# Android系统的软件安全

- 在安装程序时会询问用户相关的权限，同意才可以安装





# Android系统的软件安全

- 点击应用程序可以查看相应程序所赋予的权限



应用程序信息

## 权限

此应用程序有权访问您手机上的以下内容：

- 您的个人信息  
读取联系人数据, 写入联系数据
- 您的信息  
读取短信或彩信
- 您的位置  
(基于网络的) 粗略位置, 精准的(GPS)位置
- 网络通信  
创建蓝牙连接, 完全的互联网访问权限
- 存储  
修改/删除 SD 卡中的内容
- 手机通话  
读取手机状态和身份
- 硬件控件  
更改您的音频设置, 录音, 拍摄照片和视频
- 系统工具  
防止手机休眠, 检索当前运行的应用程序, 蓝牙管理, 显示系统级警报, 修改全局系统设置



# Android系统的软件安全

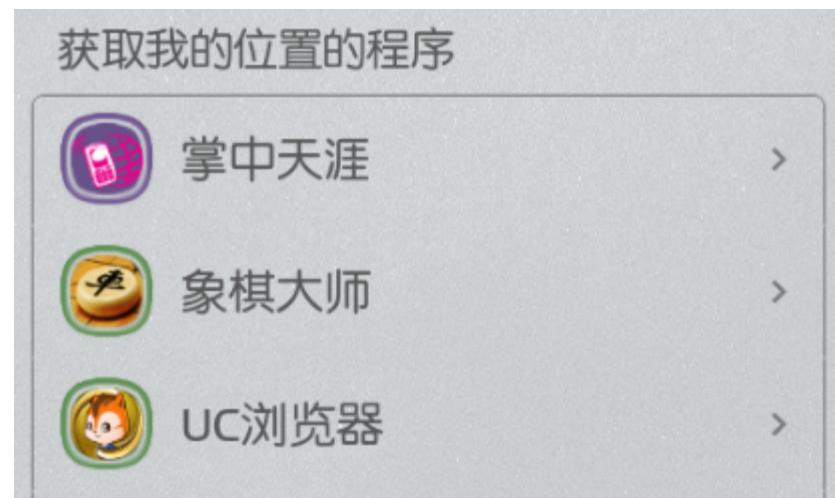
- 查看应用程序的信息





# Android系统的软件安全

## • 设置软件的GPS访问权限





# Android系统的软件安全

- 如果不做开发进行调试则需关闭该选项，这样就不允许用户使用adb shell命令来修改系统





# Android系统的软件安全

- Android软件安装于SD卡的安全性
  - Android系统是基于Linux平台进行设计的操作系统，而Linux支持的文件系统格式是ext2，我们一般购买的存储卡都是fat32格式的，并没有办法直接使用，因此在2.2固件之前官方固件都是不支持把软件装到存储卡的
  - 在Android 2.2之后的版本允许将应用程序安装于SD卡，每一个安装在SD卡的应用程序，都可以在SD卡中的/sdcard/.android\_secure 目录里找到名称中有出现它的程序名，和副文件名为asec的经过特殊加密处理后的档案



# Android系统的软件安全

- Android 4.0版本自带APP2SD的功能





# Android系统的软件安全

- 有些软件由于权限的限制无法直接移动





# Android系统的软件安全

- 安装应用程序到SD卡的过程APP2SD
  - 目前比较流行有两种方案，分别是app2ext 和 data2ext
  - 用户程序安装到到SD卡上后，其内容可能分散到：/mnt/asec , /mnt/secure , /data/data
  - 把SD卡使用软件划分为两个区，放置视频照片的区划分为FAT32，安装程序的区划分为ext



# Android系统的软件安全

- app2ext的原理
  - 删除data区中的app文件夹，然后在sd卡的ext分区上创建一个app文件，并通过软链接映射到data区
  - 系统会以为，app这个软链接是一个真实的文件夹，会把程序都安装在里面，但实际上，这些程序都安装到卡上了。但由于操作系统并不知道，所以这种情况下，我们依然看到系统显示这个程序是安装在“内置空间”的



# Android系统的软件安全

- data2ext的原理
  - 它不是用软链接，而是直接用“挂载”功能
  - data文件夹本来是对应手机内部Flash中的一个分区，而data2ext则是修改了挂载对应关系，使data文件夹挂载的不是内置Flash，而是sd卡的整个ext分区
  - 这样，不仅是app，连存储程序设置的data和缓存dalvik-cache都会存储到sd卡中



# Android系统的软件安全

- 假如重刷ROM，使用了app2ext的话，所有的程序都可以保留，但是这些程序的配置信息和游戏的存档都会丢失
- data2ext则可以连同配置和存档都保留，data2ext由于是把整个data分区都放在SD卡上，因此，这个data分区的内容就可能不会被清除，这可以保存用户的个人资料，但是也可能造成系统莫名其妙的故障



# Android系统的软件安全

- 可以对外置的SD卡进行加密





# 本章内容提要

- Android操作系统介绍
- Android的系统文件夹结构
- AndroidManifest.xml文件的解析
- Android系统的安全与权限
- Android系统的用户管理
- Android系统的软件安全
- Android应用程序下载源及安全机制



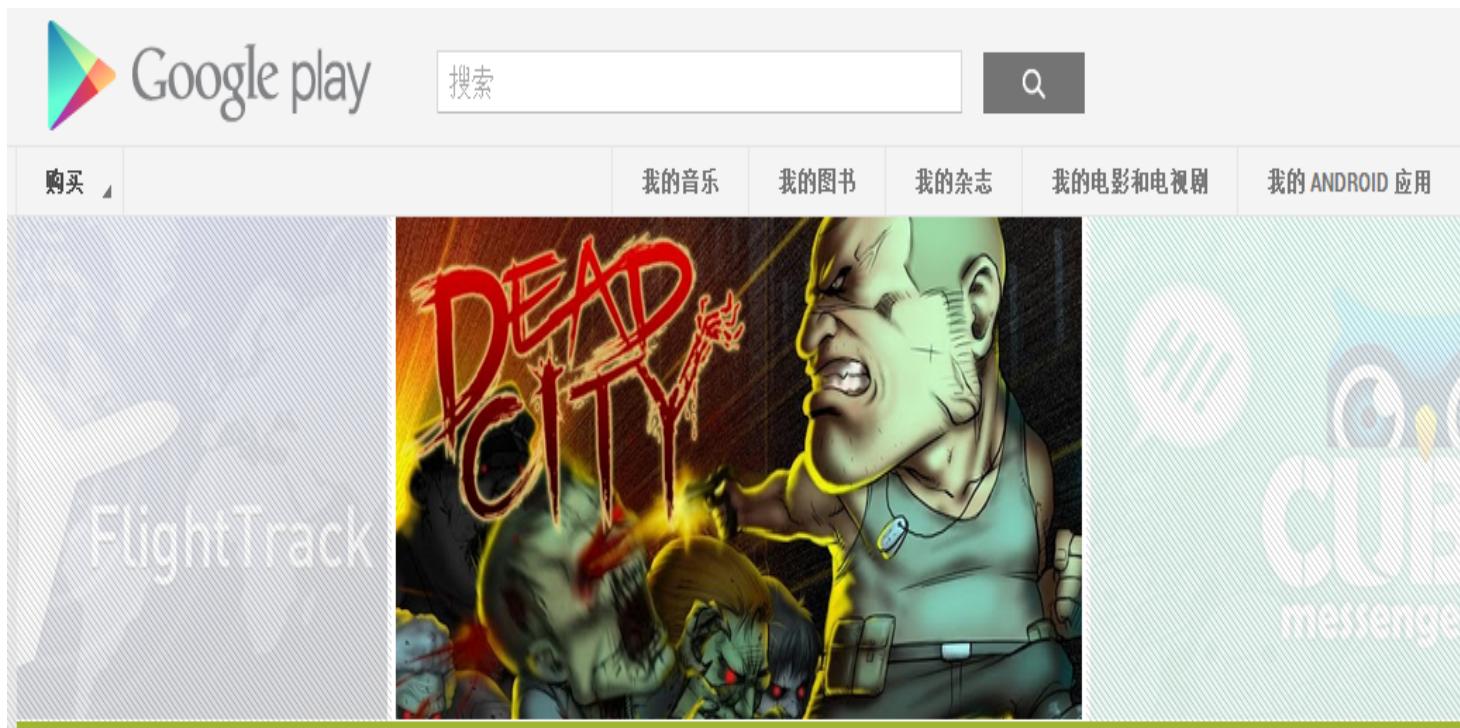
# Android应用程序下载源及安全机制

- Android应用程序的下载渠道
  - 官方的安卓市场 (Google Play)
  - 其他的第三方安装渠道



# Android应用程序下载源及安全机制

- <https://play.google.com> (安卓官方市场)





# Android应用程序下载源及安全机制

- 其他第三方的安装渠道
  - 比较有名的豌豆荚
  - 其中下载的程序会提示你相应的安全属性





# Android应用程序下载源及安全机制

- 通过更改该选项的内容可以下载非官方市场的应用程序





# Android应用程序下载源及安全机制

- 官方市场和第三方市场的区别
  - 官方市场的应用程序都是经过google官方签名人认证的程序，在安全和稳定性方面有保障
  - 第三方应用程序不一定经过google认证，而且可能有各种不同的修改版本
  - 官方市场对于提交的app程序文件都会经过严格的审查，保证下载源的安全
  - 第三方的审查力度可能没有官方大，而且发布的程序可能会携带病毒



# Android应用程序下载源及安全机制

- 从google play购买和下载的app程序可以通过google帐号的和手机的绑定来方便的进行更新和管理，同步起来也很方便
- 第三方无法提供这样的服务，一旦下载的apk文件丢失或者损坏，查找和重新下载会很不方便



# 演示实验

中国传媒大学



# 安卓手机Root有风险，且看以下演示实验

- 实验一：手机所有连接过的wifi信息  
—包含ssid、密码
- 实验二：解锁屏密码



# 实验一：手机所有连接过的wifi信息

- /data/misc/wifi/wpa\_supplicant.conf

```
network={  
    ssid="lc****me"  
    scan_ssid=1  
    psk="lc****me*****12"  
    key_mgmt=WPA-PSK  
    priority=12  
    frequency=2412  
}  
  
network={  
    ssid="h****me"  
    psk="t****w"  
    key_mgmt=WPA-PSK  
    priority=7  
    frequency=2412  
}
```



## 实验二：解锁屏密码

- 1. 启动adb shell
  - adb shell
- 2. 获取root权限
  - su
- 3. 进入/data/system/
  - cd /data/system/
- 4. 删除gesture.key (图形密码) 或者password.key (数字及密码)
  - rm gesture.key
  - rm password.key
- 5. 重启手机
  - reboot



# 回顾：Android系统安全使用科普

- 尽可能只安装来自Google Play的APP
  - 国内安卓手机要避免从小市场安装APP
- 关闭USB调试模式
- Root后手机安装Super User（授权管理）
  - 禁止shell用户通过su获取最高权限
- 安装系统安全软件
  - 精细化关闭应用程序的非必要授权请求