Android 系统介绍

作者:程姚根　　联系方式:chengyaogen123@163.com

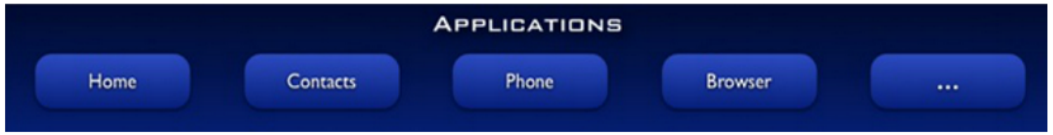
一、Android历史介绍

http://www.cnblogs.com/vamei/p/3726620.html

二、Android架构分析

下面我们来详细分析每一层的作用，以及每一层包含的内容!

1、Applications



位于最顶层，就是我们经常在Android手机上使用的app软件，这一层我们主要用Java语言开发

2、Applications Framework

这一层主要是为应用层提供相关的软件开发框架，让应用层的工程师开发APP软件更简单、快键。这一层相关的架构代码用Java语言编写。

(1)Activity Manager(活动管理器)

用来管理应用程序生命周期，并提供常用的导航回退功能

(2)Window Manager(窗口管理器)

用来管理窗口的一些状态、属性、窗口顺序等

(3)Content Providers(内容提供器)

使得一个程序可以访问另外一个应用程序的数据或者共享这个程序自己的数据

(4)View System（视图系统）

基本组件的实现，例如:list，button, text boxes

(5)Notification Manager(通知管理器)

一般用在电话、短信、邮件、闹钟铃声，在手机的状态栏上就会出现一个小图标，提示用户处理这个通知，这时从上方滑动状态就可以展开并处理这个快讯

(6)Package Manager(包管理器)

管理应用程序包,用来获取软件包信息、安装、卸载等

(7)Telephony Manager(电话管理器)

用于管理手机通话状态，获取电话信息(设备信息、SIM卡信息以及网络信息),侦听电话状态(呼叫状态、服务状态、信号强度等)以及可以调用电话拨号器拨打电话。

(8)Resource Manager(资源管理器)

管理应用程序使用到的字符串、图形、布局文件等。

(9)Location Manager(定位管理器)

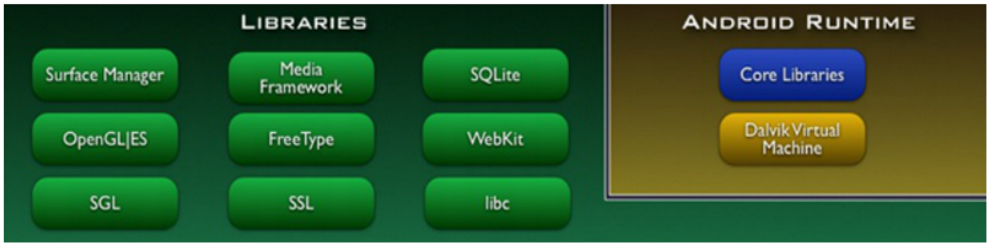
实现定位功能，它提供了两种方法定位当前设备所在的位置: GPS 和 NETWORK

(10)XMPP Service(Extensible Messaging and Presence Protocol可扩展消息与表示协议)

即时通信服务，为我们提供了很好的即时通信机制，实现了消息接收、推送、状态信息等。

3. Libraries & Android Runtime

这一层提供一些Android系统运行的时候需要的函数库以及Java语言运行的时候需要的运行环境。这一层相关代码都是用C/C++进行编写。



(1)Surface Manager

对显示子系统管理，并且为多个应用程序提供了2D和3D图层的无缝融合

(2)Media Framework

基于PacketVideo OpenCore实现的开源多媒体框架，支持多种常用音频(MP3、AAC和AMR)、视频格式(MPEG4、H264)等格式的媒体文件播放和录制，支持静态图片(JPG、PNG格式)。

(3)SQLite

轻量级数据库引擎,世界上速度最快、功能强大的数据库。

(4)OpenGL|ES(Open Graphics Library | Embedded Systems)

3D加速库，支持硬件加速及高度优化的软件加速

(5)FreeType

位图(Bitmap)和矢量(Vector)字体显示

(6)WebKit

web浏览器引擎,苹果的Safari、谷歌的Chrome浏览器都是基于这个框架来开发的。

(7)SGL

2D图形引擎库

(8)SSL(Secure Sockets Layer 安全套接层)

SSL在传输层对网络连接进行加密，有一下三个功能:

a、使用公钥证书对双端进行认证

b、通信加密

c、数据完整性检查

(9)Libc

一个从BSD继承来的标准C系统函数库,它是专门为基于embedded linux的设备定制的。

(10)Core Libraries

java语言核心库,这里主要通过JNI的方式向应用程序框架层提供调用底层程序库的接口。

(11)Dalvik Virtual Machine (Java虚拟机)

Davik 虚拟机是为了同时高效地运行多个VMS而实现的。Dalvik虚拟机执行.dex的Dalvik可执行文件，该格式的文件针对最小内存使用做了优化。Dalvik虚拟机是基于寄存器的，所有的类都经由Java汇编器编译，然后通过SDK中的dx工具转化成.dex格式并由虚拟机执行。Dalvik虚拟机依赖Linux的一些功能，比如线程机制和底层的内存管理机制。每个Android应用程序都在它自己的进程中运行，都拥有一个独立的Dalvik虚拟机实例。

4. Linux Kernel

Android系统并没有使用原生态的Linux Kernel,而是对Linux Kernel针对嵌入式设备做了修改。主要的体现在以下几个方面:

(1)Android Binder

Android在以前的Linux 系统中的IPC机制中增加了一种新的进程间通信机制 Binder。可以看到Android并没有使用Linux内核自带的IPC而是自己重新实现一种新的IPC，从这点可以看到Binder的重要性。为什么重新实现呢?原先的Linux内核中的IPC，要么是效率低，要么是操作起来复杂，而且安全性差。也就是这些原因Android提供了Binder来解决这些问题。

(2)Android Low Memory Killer

低内存管理器是Linux标准OOM(Out Of Memory)改进版，当系统内存不足时，会杀掉一些不重要的进程，释放内存空间。

在Android中，即使当用户退出应用程序之后，应用程序的进程也还是存在于系统中，这样是为了方便程序的再次启动，但是这样的话，随着打开的程序数量的增加，系统的内存会变得不足，就需要杀掉一部分进程以释放内存空间。至于是否需要杀死一些进程和那些进程需要被杀死，是通过Low Memory Killer机制来进行判定的。

(3)Android Ashmem(Anonymous Shared Memory)

匿名共享内存，它是基于Linux共享内存机制实现的。它以驱动程序的形式实现在内核空间中。它有两个特点，一是能够辅助内存管理系统来有效地管理不再使用的内存块，二是它通过Binder进程间通信机制来实现进程间的内存共享。

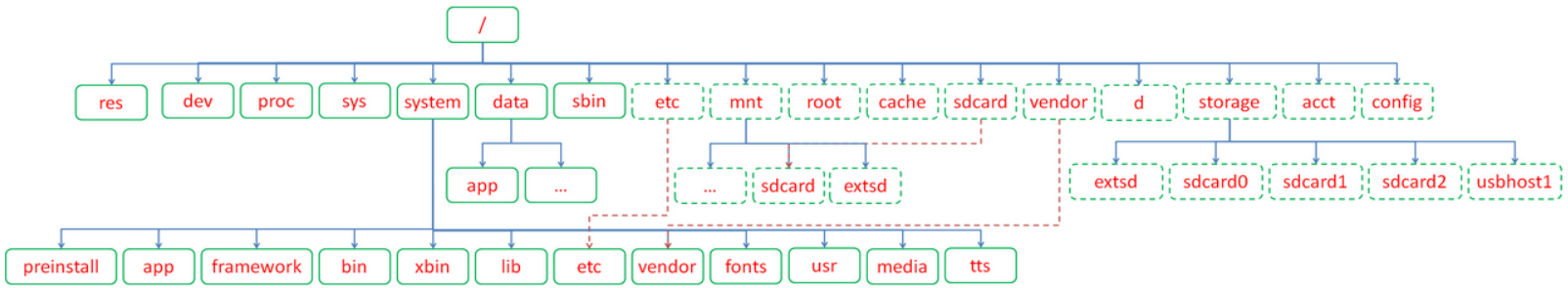
(4)Android Logger

我们在开发Android应用程序过程中可以很方便地使用Log信息来调试程序，这都归功于Android的Logger驱动为用户层提供的Log支持。

(5)Android Power Manager

基于标准Linux电源管理系统的轻量级电源管理驱动。

三 Android文件系统目录结构分析



下面我们介绍一下，每个目录的主要用途:

/res	系统资源目录(图片)
/dev	存放的是设备文件
/proc	用来挂载基于内存的procfs文件系统，用存放进程的所有信息和Linux内核部分的内核 参数以及系统配置信息
/sys	用来挂载基于内存的sysfs文件系统，它的作用与proc有些类似,但除了与proc相同的具有查看和设定内核参数功能之外，还有为Linux统一设备模型作为管理之用。
/system	Android系统主目录 /system/app android系统应用程序 /system/preinstall 预装android应用程序 /system/priv-app 私有的android应用程序 /system/framework 应用程序框架层java包 /system/bin 用户常用的工具程序，其中大部分链接到toolbox(类似嵌入式Linux中的busybox) /system/xbin 系统管理工具 /system/lib C/C++实现的动态库，被java虚拟机或C/C++程序调用 /system/etc 启动脚本和配置文件 /system/fonts 字库文件 /system/media 系统多媒体文件 /system/usr 用户文件夹，包含共享、键盘布局、时间区域文件等 /system/vendor 厂家文件(硬件驱动模块和动态库) /system/tts 语音合成信息
/data	用来存放用户的数据
/sbin	超级用户使用的一些命令
/etc	/system/etc的连接文件
/mnt	链接到外部存储设备或存放临时文件
/root	超级用户的用户主目录
/cache	用于存放应用程序临时数据，用于加速
/sdcard	链接到/storage/emulated/legacy
/vendor	链接到/system/vendor
/d	链接到/sys/kernel/debug用于内核调试
/acct	系统回收站，误删除的系统文件恢复
/config	计算机配置文件