Android 系统简介

杨樾人 3160104875

软件工程

Android是一个基于Linux内核的开放源代码移动操作系统，由Google 持续领导与开发，主要应用在触屏移动设备如智能手机和平板电脑与其他便携式设备。本文将简单介绍Android的起源和市场规模以及Android的系统架构，在文章的最后，有作者关于Android的三个问题的思考。

一、简介

Android 是一个基于 Linux 内核的开放源代码移动操作系统，由 Google 成立的 Open Handset Alliance（OHA，开放手持设备联盟）持续领导与开发，主要设计用于触屏移动设备如智能手机和平板电脑与其他便携式设备。

2003 年，Android 之父安迪·鲁宾等人在美国加利福尼亚州成立了 Android 科技公司。2005 年被谷歌收购，谷歌借助 Android 正式进入移动领域，抗衡微软，试图阻止微软在移动市场复制桌面市场的成功。在谷歌鲁宾领导团队开发了基于 Linux 内核驱动的操作系统平台，Google 向手机厂商推出此平台，并承诺提供一个灵活可靠的系统，Android 自此打开了一个广阔的市场。

Android 系统非常庞大、错综复杂，底层采用 Linux 作为基底，上层采用包含虚拟机的 Java 层以及 Native 层，通过系统调用(Syscall)连通系统的内核空间与用户空间。用户空间主要采用 C++和 Java 代码，通过 JNI 技术打通用户空间的 Java 层和 Native 层(C++/C)，从而融为一体。

根据 Gartner 公司的数据，在 2017 年销售的智能手机中，99.9%搭载了 iOS 和 Android 两个操作系统，其他竞争对手已经被完全排除。这其中，Android 占据 86%的份额，iOS 占据 14%。 Android 设备覆盖了高中低价位，并且包含无数款设备，而 iOS 主打高端市场，故 Android 的市场份额明显高于 iOS。

二、安卓系统架构

Android的系统架构采用了分层架构的思想。从上层到底层共包括四层，分别是应用程序程序层、应用框架层、系统库和Android运行时和Linux内核。

1. 应用程序(Applications)

Android会同一系列核心应用程序包一起发布，该应用程序包包括email客户端，SMS短消息程序，日历，地图，浏览器，联系人管理程序等。开发人员在此层次进行编写代码，所有的应用程序使用JAVA语言编写的。

2. 应用程序框架(Application Frameworks)

应用程序框架提供应用程序开发的各种 API，其中包括：

1) 丰富而又可扩展的视图（Views）可以用来构建应用程序，它包括列表（lists），网格（grids），文本框（text boxes），按钮（buttons），甚至可嵌入的web浏览器。
2) 内容提供器（Content Providers）使得应用程序可以访问另一个应用程序的数据（如联系人数据库），或者共享它们自己的数据。
3) 资源管理器（Resource Manager）提供非代码资源的访问，如本地字符串，图形，和布局文件（layout files）。
4) 通知管理器（Notification Manager）使得应用程序可以在状态栏中显示自定义的提示信息。

5) 活动管理器（Activity Manager）用来管理应用程序生命周期并提供常用的导航回退功能。

3. 系统运行库与 Android 运行环境(Libraries and Android Runtime)

1) 系统运行库

它是连接应用程序框架层与 Linux 内核层的重要纽带。Android 包含一些 C/C++库，这些库能被 Android 系统中不同的组件使用。它们通过 Android 应用程序框架为开发者提供服务。

以下是一些核心库：

- WebKit：Android浏览器内核，用来渲染网页。
- SQLite：轻量级的关系型数据库，便于存储和共享应用程序数据。
- SSL库：负责互联网安全。
- 多媒体库：支持多种常用的音频、视频格式录制和回放，编码格式包括MPEG4、MP3、H.264、AAC、ARM等。
- FreeType：提供字体的描绘与显示服务。
- SGL：底层的2D图形引擎。
- Libc：从标准C系统函数库，专门为基于embedded linux的设备定制。

2) Android 运行环境

Android 应用程序都在 Android 运行时中执行，其运行时分为核心库和 Dalvik 虚拟机两部分。

核心库提供了 Java API 中的大多数功能，同时也包含了 Android 的一些核心 API。

Dalvik 虚拟机是 Google 公司自己开发的用于 Android 平台的 Java 虚拟机，它适合内存和处理器速度有限的系统。不同于 JDK 运行 .class 文件，Dalvik 虚拟机运行 .Dex 文件（由 .class 转换而来）。

在 Android 中可以同时运行多个 Dalvik 实例，并且每个 Dalvik 应用都是一个独立的 Linux 进程。独立的进程可以防止在虚拟机崩溃的时候所有程序都被关闭。

很长时间以来，Dalvik虚拟机一直被用户指责为拖慢Android运行速度、不如IOS的根源。不过Google在2014年的I/O大会上推出了Android L，删除了Dalvik，由传闻已久的ART代替，表现更佳。

ART执行程序的方式与Dalvik完全不同：Dalvik在程序运行时依靠一个即时编译器（Just-In-Time，简称JIT）将字节码翻译成机器码，因为多了一个环节，并不是非常高效。ART 在应用程序安装时就会把字节码转换成机器码，这种机制叫做Ahead-Of-Time，简称AOT。因为程序运行时没有了解释字节码这一环节，因而启动更快，执行效率更高，触控更加灵敏，体验更加流程。

4. Linux 内核层

处于最底层的Linux内核层提供了基本的系统功能，例如流程管理、内存管理、设备管理、电源管理等。它还提供许多驱动，例如显示驱动、摄像头驱动、键盘驱动、wifi驱动、内存驱动等。

5. 扩展：硬件抽像层

Android 的 Hardware Abstraction Layer（硬件抽像层）是能以封闭源码形式提供硬件驱动模块。HAL 的目的是为了把 Android framework 与 Linux kernel 隔开，让 Android 不至过度依赖 Linux kernel，以达成 kernel independent 的概念，也让 Android framework 的开发能在不考量驱动程序实现的前提下进行发展。

Android 的硬件抽象层，是对 Linux 内核驱动程序的封装，它向上提供接口，屏蔽低层的实现细节。Android 把对硬件的支持分成了两层，一层放在用户空间，一层放在内核空间，其中，硬件抽象层运行在用户空间，而 Linux 内核驱动程序运行在内核空间。之所以这样安排，是因为 Android 的源代码版权遵循 Apache License，无需公布源码，而 Linux 内核源代码遵循 GNU License，必须公布源码。把硬件抽象层和内核驱动分离，可以不必公布手机硬件的相关参数，可以保护商业秘密。因此，Android 把对硬件的支持分成硬件抽象层和内核驱动层，内核驱动层只提供简单的访问硬件逻辑，例如读写硬件寄存器的通道，至于从硬件中读到了什么值或者写了什么值到硬件中的逻辑，都放在硬件抽象层中去了，这样就可以把商业秘密隐藏起来了。

三、关于 Android 的思考

1. Android 为什么会选择 Linux

成熟的操作系统有很多，但是Android选择采用Linux内核，这与Linux的一些特性有关，比如：

1) 强大的内存管理和进程管理方案
2) 基于权限的安全模式
3) 支持共享库
4) 经过认证的驱动模型
5) Linux 本身就是开源项目

2. 为什么 Android 不是 Linux

Android 设备的闪存被分成几个分区，例如/system 用于操作系统本身，而/data 是用于用户数据和应用程序的安装。跟 Linux 桌面发行版相比，Android 设备的拥有人都没有给予超级用户的进入操作系统的权限，以及例如/ system 是只读存储器的敏感分区。然而，文件系统层次结构标准是可以透过利用 Android 中的安全漏洞来获取，那是开源社区经常使用它来增强其设备的功能，恶意的一方还可以透过安装[计算机病毒及恶意软件来恶意获取系统数据。

1.它没有本地窗口系统

什么是本地窗口系统呢？本地窗口系统是指 GNU/Linux 上的 X 窗口系统，或者 Mac OX X 的 Quartz 等。不同的操作系统的窗口系统可能不一样，Android 并没有使用（也不需要使用）Linux 的 X 窗口系统，这是 Android 不是 Linux 的一个基本原因。

2.它没有 glibc 支持

由于 Android 最初用于一些便携的移动设备上，所以，可能出于效率等方面的考虑，Android 并没有采用 glibc 作为 C 库，而是 Google 自己开发了一套 Bionic Libc 来代替 glibc。

（Bionic 本身已跟特定于 Linux 内核的几个主要特点而设计。使用 Bionic 而不是 GNU C 库（glibc）或 uClibc 的主要好处是：它运行时间的足迹较小，以及对低频 CPU 进行优化。与此同时，Bionic 根据 BSD 许可条款而获得许可，当中 Google 找到更适合 Android 的整体许可模式）

3.它并不包括一整套标准的 Linux 使用程序

Android 并没有完全照搬 Liunx 系统的内核，除了修正部分 Liunx 的 Bug 之外，还增加了不少内容，比如：它基于 ARM 构架增加的 Gold-Fish 平台，以及 yaffs2 FLASH 文件系统等。由于 Android 的硬件抽象层和内核分层，Android 被踢出了 Linux 内核主线代码树中。Android 放在内核空间的驱动程序对硬件的支持是不完整的，把 Linux 内核移植到别的机器上去时，由于缺乏硬件抽象层的支持，硬件就完全不能用了。

4.Android 专有的驱动程序

除了上面这些不同点之外，Android 还对 Linux 设备驱动进行了增强，比如 Android Binder，Android 的电源管理，低内存管理，匿名共享内存，轻量级的日志设备和 USB Gadget 驱动等驱动程序。

3. 为什么 Android 能够成功

1、开放性

在优势方面，Android 平台首先就是其开放性，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，这里的人气包括消费者、和厂商，而对于消费者来讲，最大的受益正是丰富的软件资源。开放的平台也会带来更大竞争，如此一来，消费者将可以用更低的价位购得心仪的手机。

2、挣脱运营商的束缚

在过去很长的一段时间，特别是在欧美地区，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。自从 iPhone 上市，用户可以更加方便地连接网络，运营商的制约减少。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。

3、丰富的硬件选择

这一点还是与 Android 平台的开放性相关，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。好比你从诺基亚 Symbian 风格手机一下改用苹果 iPhone，同时还可将 Symbian 中优秀的软件带到 iPhone 上使用、联系人等资料更是可以方便地转移。

4、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。因此不会受到各种条条框框的阻挠，可想而知，会有多少新颖别致的软件会诞生。但也有其两面性，血腥、暴力、情色方面的程序和游戏如何控制正是留给 Android 难题之一。

5、无缝结合的 Google 应用（仅针对国外）

如今叱咤互联网的 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。

国内也有众多优秀厂商开发自己的应用商店，为 Android 的使用提供了便捷性。