移动操作系统体系结构的研究分析

胡忠望

(肇庆学院计算机学院 肇庆 526061)

摘 要 移动操作系统是国际上研发竞争激烈的基础软件。移动操作系统是移动设备中最基本的系统软件,它控制移动设备的所有资源并提供服务和应用程序开发的基础。体系结构设计是研发新移动操作系统最主要的任务。分析主流移动操作系统 Andriod、iOS、Symbian、Windows Phone 的体系结构,并对体系结构及实现技术的异同进行研究比较,提出自主移动操作系统的技术路线,探讨移动操作系统的发展趋势,为移动操作系统发展提供技术支持和帮助。

关键词 移动操作系统,体系结构,移动互联网,层次化架构

中图法分类号 TP316.8 **文献标识码** A

Research on Mobile Operating System Architectures

HU Zhong-wang

(School of Computer, Zhaoqing University, Zhaoqing 526061, China)

Abstract Mobile operating system(MOS) is the foundational software based on current international competitive field. MOS is the basic system software for mobile devices, it controls all resources of mobile device and provides basic services and application development. The architecture design is a main task of the new MOS development. This paper analyzed the structures of the mainstream MOS Andriod, iOS, Symbian and Windows Phone, compared the similarities and differences of the architecture and implementation technology, gived the technical routes of autonomous MOS, discussed the development trend of MOS, and pointed out the development trend of the future MOS. This paper is proposed to provide technical support and help for the development of MOS.

Keywords Mobile operating system, Architecture, Mobile internet, Layered architecture

移动互联网时代已经到来,移动设备大行其道。我们把移动设备上运行的操作系统称为移动操作系统(Mobile Operating System, MOS)。MOS 是移动设备中最基本的系统软件,它控制移动设备的所有资源并提供服务和应用程序开发的基础,可以说,MOS 是移动设备的核心。当今,智能手机无疑已经成为移动互联网发展的重要载体,以智能手机操作系统为代表的 MOS 在移动互联网产业里占据了核心的位置,它连接着硬件开发者、软件开发者、运营商以及终端用户。伴随移动互联网的迅猛发展,MOS 的发展日新月异,Google、Apple、Microsoft、Nokia 等公司展开了 MOS 的争霸战,逐渐显现出对整个移动互联网应用产业的垄断。虽然多年来我国政府对操作系统非常重视并大量投入,但成效并不显著。整体来讲,我国自主知识产权的 MOS 发展还处于起步阶段。

从设计人员的角度来看,操作系统是一大堆模块和它们之间的相互联系,即操作系统的体系结构。实际上,构建一个新的操作系统最主要的任务就是体系结构的设计[1]。因此,研究分析主流 MOS 的体系结构,对于更好地开发移动应用以及研发自主的 MOS,非常必要也非常重要。

1 当今主流 MOS 体系结构分析

1.1 Android 操作系统

Android 体系结构[2]分为 4 层,从高到低分别是 Applications layer、Application Framework layer、Libraries & Android Runtime layer、Linux Kernel layer,如图 1 所示。

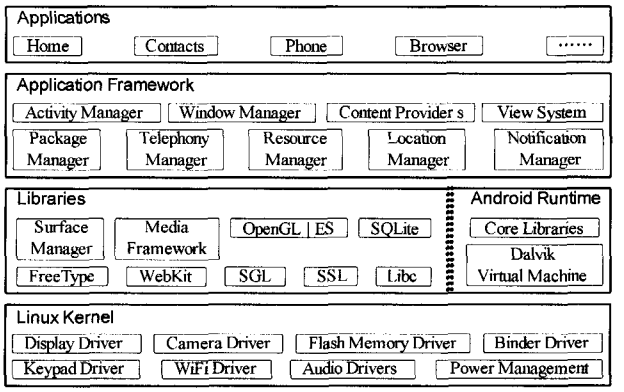

图 1 Android 操作系统体系结构

(1) Applications layer(应用程序层)。该层包含了一系列核心应用程序,如主屏幕、电子邮件、短信服务、日历、浏览器、联系人管理、地图等。这些应用程序采用 Java 语言编写,运行于 Google 自己研发的 Dalvik 虚拟机上,并且都可以被他人开发的其他应用程序所替换,使得系统更加灵活和个性化。

(2) Application Framework layer(应用程序框架层)。应用程序框架是进行 Android 开发的基础,提供了应用程序开发的各种 API,很多核心应用程序也是通过这一层来实现其核心功能的。该层简化了组件的重用,开发人员可以直接使

本文受广东省高等学校人才引进专项资金项目(2010-343)资助。

胡忠望(1965—),男,教授,CCF 会员,主要研究领域为计算机网络、移动互联网、信息安全,E-mail:zhongwanghu@163.com。

用其提供的组件来进行快速的应用程序开发，也可以通过继承实现个性化的拓展。每个应用程序都可以发布自身的功能块，而其它应用程序在遵循框架的安全性限制的前提下，可以使用 Android 已发布的功能块。由于这样的重用机制，用户可以对 Android 系统本身提供的各种应用程序组件进行替换。

(3)Libraries & Android Runtime layer(函数库和运行时层)。该层提供函数库和运行环境，分为两个部分。1)函数库(Libraries)。由大多数开源的函数库组成，如标准的 C 函数库 Libc、OpenSSL、SQLite 等。其中的 WebKit 负责 Android 网页浏览器的运行，2DSGL/OpenGL 图形与多媒体函数库分别支持各种影音与图形文件的播放；SQLite 提供了轻量级的数据库管理系统。Libraries 是应用程序框架的支撑，这些库能被 Android 系统中的各式组件使用，通过 Android 应用程序框架为开发者提供服务。2)运行时(Android Runtime)。Android 采用自有的 Android Runtime 来执行 Java 应用程序，而不是使用 J2ME 来执行。Android Runtime 由核心库和 Dalvik 虚拟机两部分组成。①核心库。其中一部分是功能函数，以便开发者使用 Java 语言编写 Android 应用程序时调用；另一部分是 Android 的核心库如 android. os、android. net、android. media 等。②Dalvik 虚拟机。主要为 Android 应用程序提供运行环境，其作用相当于 JVM。Dalvik 虚拟机是专门为移动设备设计的基于寄存器的虚拟机，支持同时执行多个虚拟机程序，执行的是. dex 文件，这种文件由转换工具 dx 对 Java 字节码进行转换后得到。

(4)Linux Kernel layer(Linux 内核层)。该层基于 Linux 2.6 内核实现，它负责硬件驱动、网络管理、电源管理、系统安全、内存管理等。该层在软件层和硬件层之间建立了一个抽象层，使得应用程序开发人员无需关心硬件细节。

1.2 iOS 操作系统

iOS 体系结构[3,4]分为 4 层，从高到低分别是 Cocoa Touch layer、Media layer、Core Services layer、Core OS layer，如图 2 所示。

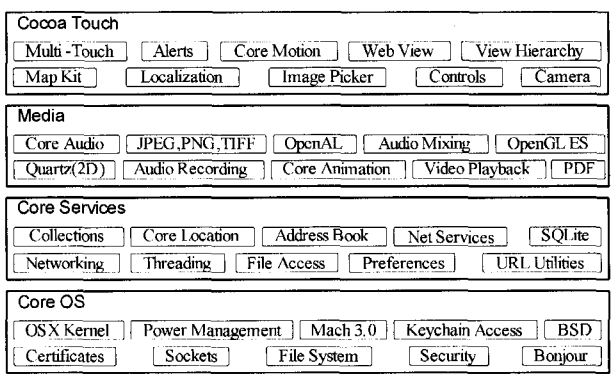

图 2 iOS 操作系统体系结构

(1)Cocoa Touch layer(Cocoa 触摸框架层)。该层是创建 iOS 应用程序所需的关键框架。应用程序的可视界面以及与高级系统服务交互都是构建在该层提供的接口之上，最基本的就是各个界面控件，一般的开发者开发应用程序使用的接口都是该层接口。在开发应用程序的时候，为了减少程序开发的工作，同时也是出于效率考虑，Apple 公司推荐尽量使用该层提供的技术支持。

(2)Media layer(媒体层)。该层包含图形技术、音频技术和视频技术，这些技术相互结合就可为移动设备带来最好的多媒体体验。该层提供的接口让开发者开发基于图形图像、音频或者视频的应用程序变得更加容易，提供了统一的视图供开发者直接使用。开发者既可以使用高级框架更快速地创建高级的图形和动画，也可以通过底层框架访问必要的工具。

(3)Core Services layer(核心服务层)。该层是 iOS 上层框架和系统构建的基础，它提供所有应用程序使用的基础系统服务，这些服务很多应用程序都不可以直接访问。iOS 系统的许多部分也是构建在这一层的服务之上。

(4)Core OS layer(核心操作系统层)。该层位于 iOS 层次结构的最底层，是最为核心的系统，包括了多种硬件管理、安全管理等内容，是很多其他技术的构建基础。通常情况下，这些功能不会直接应用于应用程序，而是应用于其他框架。但是，在直接处理安全事务或和某个外设通讯的时候，则必须应用到该层的框架。

1.3 Symbian 操作系统

Symbian OS V9.3 体系结构[5]分为 5 层，从高到低分别是 UI Framework layer、Application Services layer、OS Services layer、Base Services layer、Kernel Services & Hardware Interface layer，如图 3 所示。

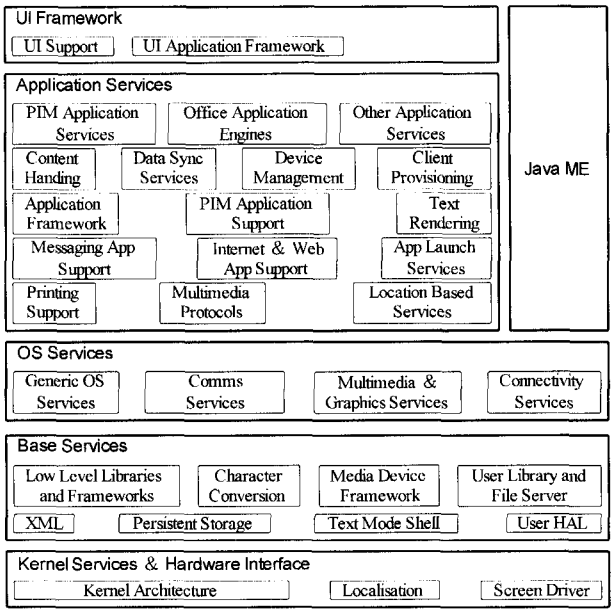

图 3 Symbian 操作系统体系结构

(1)UI Framework layer(UI 框架层)。该层为构建用户界面提供了框架和库，包括用户界面控件的基类层次、其他的框架和实用程序(包括界面组件使用的具体的 widget 类)。

(2)Application Services layer(应用程序服务层)。该层为 Symbian OS 的应用程序提供独立于用户界面的支持。该层服务分为 3 个主要的部分：①系统级别的服务，如基本的应用程序架构，可以被所有应用程序使用。②提供专门技术逻辑的服务，如消息和多媒体协议，可以被应用程序的多个类使用。③支持专门的单个应用程序的服务，如个人信息管理(PIM)和办公应用程序，也包括许多由获得许可的厂商使用或扩展的应用程序引擎。

与 UI 框架和应用程序服务两个层都相关的 Java ME 包，为 Java 应用程序抽象(并实现)了两层的元素。Symbian

的 Java 实现基于 Java ME MIDP 2.0 和 CLDC 1.1，Symbian OS 从一开始就提供对 Java 的支持，但是早期的 Java 系统是基于 Personal Java 和 JavaPhone，基于 Java ME 的标准系统最初在 Symbian OS v7.0 中出现。自 Symbian OS v8 开始，Java VM 成为了 Sun CLDC HI 的一个端口。

(3)OS Services layer(操作系统服务层)。该层提供了服务器、框架和库，从而将裸系统扩展成一个完整的操作系统。服务分为 4 个主要的块，这些块提供了所有面向专门技术但不独立于应用程序的服务：通用操作系统服务、通信服务、多媒体和图形服务、连通性服务。

(4)Base Services layer(基本服务层)。该层提供了用户端的最底层的服务。这些服务只依靠操作系统内核(和相关组件)。它将操作系统的内核扩展成为一个可用的(不过是最小的)系统。特别是，基本服务层中服务就可以满足新硬件最小基本移植的需要(即最小基本移植只需要系统的两个最低层)。

(5)Kernel Services & Hardware Interface layer(内核服务和硬件接口层)。该层包含了操作系统内核本身和抽象了下面硬件接口的支持组件，包括逻辑和物理设备驱动和为参考硬件平台实现预打包的各种支持。Symbian OS V8 以前的版本使用的都是内核体系结构 1(EKA1 内核)。Symbian OS V8.1 b 和 V9 以后的所有版本都是基于新的内核体系结构 2(EKA2)的实时内核。

1.4 Windows Phone 操作系统

Windows Phone 7(WP7)体系结构[6,7]分为两层，从高到低分别是 User Space layer、Kernel Space layer，如图 4 所示。

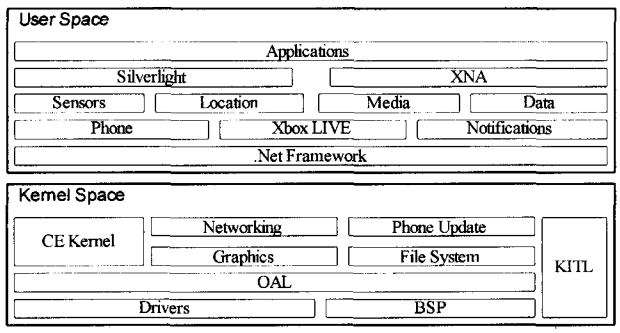

图 4 Windows Phone 7 操作系统体系结构

(1)User Space layer(用户空间层)。该层为应用程序提供独立于用户界面的支持，为应用程序开发提供两个开发平台：Silverlight Framework 和 XNA Framework。Silverlight 是以 XAML 文件为基础的应用程序设计框架，用来开发基本应用、网络应用、多媒体应用和控件。XNA 则是用来开发基础的游戏设计框架，用来开发 2D 游戏、3D 游戏和游戏控件。Silverlight 和 XNA 都基于. NET 平台，所有. NET 平台下的应用程序在用户空间中运行。

(2)Kernel Space layer(内核空间层)。该层基于 Windows Embedded CE 6.0 内核，提供基本的系统服务，包括内存模型及管理、进程/线程的调度、主版支持包、驱动程序，OEM 适配层、其他系统调用等。WP7 通过一个统一的内核去管理，而其他子系统通过加载为 DLL 的形式去实现其功能。操作系统内核、驱动和系统服务在内核空间中执行。

2 主流 MOS 体系结构及实现技术的分析比较

通过分析 Andriod、iOS、Symbian、WP 的体系结构，可以发现，它们既使用了操作系统的一些共性技术，又自主各具自身的特色。

2.1 共性特点

(1)层次化架构

Andriod、iOS、Symbian、WP 都采用了层次化平台式架构，向下适配硬件系统发挥终端基础效能，向上支撑应用软件决定用户最终体验。以操作系统为核心，上层集成必要的中间件、应用软件平台，并向第三方开放 API 接口。

在层次结构的操作系统中，系统由若干个层次构成，每一层都构建在其下的一层之上。最底层就是硬件裸机，最高层则是应用程序。每一层中包含了若干的数据和操作，所有的层次内的数据以及部分层次内的操作对其它层是不可见的。每一层均公布了一定的接口以供其它层调用，这些接口也是外层访问该层唯一的途径。层与层之间的调用关系严格遵守调用规则，每一层只能够访问位于其下层所提供的服务，利用它的下层提供的服务来实现本层的功能并为其上的层提供服务，每一层不能够访问位于其上的层次所提供的服务。

Andriod、iOS、Symbian、WP 高层框架为底层构造提供面向对象的抽象。这些抽象可以减少需要编写的代码行数，同时还对复杂功能进行封装，从而让编写代码变得更加容易。层次模式减少了各层之间的依赖性，方便各层的独立开发和调试。其好处是使得层与层互相分离，明确各层的分工，保证了层与层之间的低耦合。当下层内或者层下发生改变时，上层应用程序无需改变。

(2)继承中创新的思想

Andriod、iOS、Symbian、WP 都不是原始创新产品，都经过了较长历史的演进，继承了历史上操作系统的成果，是对历史操作系统的创新发展。图 5 示出 Symbian、Windows Phone 的版本演进，图 6 示出 Andriod、iOS 的版本演进。

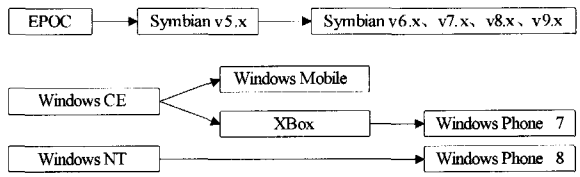

图 5 Symbian、Windows Phone 的版本演进

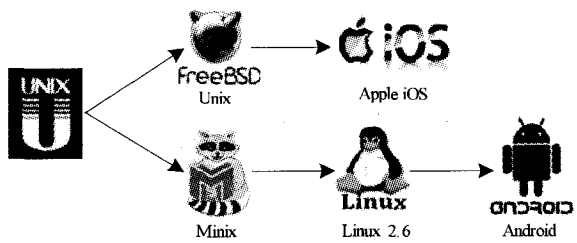

图 6 iOS、Andriod 的版本演进

2.2 差异性

(1)实现技术的差异性

从技术层面来看，Andriod、iOS、Symbian、WP 通过某种技术将自己和其他的操作系统区隔开来。1)Android 虽然采用 Java 语言，但使用了不同于 Sun(现在是 Oracle)JDK 的 API。为了让 Android 避开可能的版权问题，使用了自己的 JRE——Dalvik 虚拟机，不直接采用 Java 的虚拟机来运行 Android 程序。2)iOS 采用了 Object C 语言，为应用程序提供

的接口和框架甚至有别于苹果自己的 Mac OS X。3)Symbian 本身用 C++编写,兼顾效率的考虑其支持的 C++仅是 ANSI C++标准的一个子集。Symbian 采用了自己的异常处理机制和轻量级的模板库来代替标准 C++的异常处理和标准模板库(STL)。应用程序开发支持使用 C++和 Java,但需要掌握 Symbian 的一些特殊编程习惯如预防内存泄漏技术,目的是提高硬件资源的利用效率。4)WP 采用了 C#语言,在. Net 框架下进行开发。

(2)内核体系结构的差异性

内核是对硬件的首次扩充,是实现操作系统各项功能的基础。目前操作系统主要有两种系统内核模型:单体结构内核与微内核。

单体结构内核提供的主要功能有:文件管理、设备驱动、内存管理、CPU 调度以及网络协议等。微内核结构又称客户机/服务器结构,它尽可能多地从操作系统中去掉东西,只留下一个很小的内核。通常采用的方法是,由用户进程来实现大多数操作系统的功能。

Android 内核是基于 Linux 2.6 内核的单体结构内核,功能包括硬件驱动、网络管理、电源管理、系统安全、内存管理等。

iOS 是单体结构内核,采用 FreeBSD 和 Mach 而改写的 Darwin,是一个符合 POSIX 标准的开源 Unix 核心。它提供整个 iOS 的基础功能:硬件驱动、内存管理、程序管理、线程管理(POSIX)、文件系统、网络(BSD Socket),以及标准输入输出等。

Symbian OS 采用微内核结构,只有很小一部分运行在最高优先级,其它的功能都以服务器的方式提供,这主要是基于效率以及低能耗的考虑。

WP7 是单体结构内核,功能包括内存模型及管理、进程/线程的调度、主版支持包、驱动程序,OEM 适配层、其他系统调用等。

(3)MOS 生态系统的差异性

从目前 MOS 的成功要素来看,产业生态体系的重要性远远超过技术本身。MOS 之战将会集中于生态系统的竞争,即由软件、硬件、应用程序和服务一体化构成的系统。Andriod、iOS、WP 采取不同的策略来构建自己 MOS 的生态系统[8]。

Apple 采取的是封闭策略,从手机芯片、操作系统到应用商店均是 Apple 的私有产品,控制了产业链大部分的利润。Apple 通过优秀的用户体验、系统稳定性和开发的简易性吸引了大量的应用开发商。

Android 则采取和 Apple 背道而驰的开源、开放路线,很好地满足了广大终端制造厂商渴望寻找一个低成本开放平台的需求。Android 的开源策略使得终端价格更为便宜,也满足了数量相当可观的部分消费者。Android 的应用开发商特性和 Apple 有较大的区别:Apple 的应用开发商多为中小软件企业或个人工作室,他们希望通过 App 的下载获得收益;Android 的应用开发商多为互联网企业,他们希望通过 Android 应用延伸其在移动互联网的用户渠道。

Microsoft 走的是中间路线——核心系统软件平台封闭,但提供给硬件制造商和第三方开发者一些 API 接口。基于 WP 的开发应用可以很容易地平移到微软平板电脑和传统电脑上,这是其他平台所不具备的优势,但 WP 昂贵的 License 阻止了大多数厂商的选择。

2.3 系统级别的比较

2012 年 5 月 27 日,诺基亚宣布放弃开发 Symbian。在此仅对 Andriod、iOS、WP3 个主要 MOS 在系统级别进行比较,如表 1 所列。

表 1 Andriod、iOS、Windows Phone 7 系统级别比较

项目	Android	iOS	Windows Phone 7
支持的 CPU 架构	ARM、X86	ARM	ARM
OS 内核	Linux 2.6 内核	Mac OS X 的 Mach 内核	Windows CE 6 内核
OS 底层内存管理	有	有	有
内存模型	分页,可选的 SD 卡交换区	分页,没有基于磁盘的备份存储	分页,没有交换区
管理运行时	Dalvik	无	.NET 框架公共语言(CLR)
应用程序沙盒	有	有	有
跨应用程序通信	Intent、广播接收器、内容提供器和服务	有限的,使用自定义的 URL	启动器和选择器

3 MOS 研发的技术路线及发展趋势

通过对主流 MOS 体系结构的分析比较,提出以下我国研发自主 MOS 的技术路线。

(1)基于 Android 的某一版本,通过各种定制方式分支以期获得与 Android 可竞争的市场份额,可能存在的问题是由于新生态系统的规模过小,很难吸引足够数量的应用开发商。如目前中移动 Ophone、阿里云 OS、小米科技 MIUI 等,采用的就是这种路线,它们都不能称为自主的操作系统。

(2)兼容演进路线。基于全球性的开源资源及 Google 的部分开源资源进行研发,初期与 Andriod 的应用保持兼容,但核心组件必须重新进行独立研发(如:基础 API、虚拟机,获得著作权证书和专利)。按照国人的业务需求和安全保障要求等,设计演进路线,不断地消化吸收再创新,最终打破国外产品的垄断而独立发展。

(3)原创自主路线。基于 Linux 和相关开源技术完全自主开发,不与任何主要操作系统生态系统的应用兼容。可能存在的问题是开发一个 MOS 相对容易,但形成生态系统很难。MOS 的成功在于建立一个开放共享的产业生态环境。

从市场走向来看,未来一段时期内,MOS 市场仍将处于被若干主要品牌瓜分的状况。随着 Android、iOS、WP 等几大阵营的持续竞争,MOS 将逐步得到创新和完善。在这个过程中,MOS 存在以下发展趋势:

(1)兼容性仍将是各种平台竞争的焦点。随着高频处理器和大容量存储器等硬件的不断推广,兼容性问题一直是困扰智能手机操作系统的共同问题,因此,首先有效解决兼容性问题的平台将有望在应用环节上取得竞争优势。

(2)GPS、WiFi 以及其他一些多媒体和 Web2.0 的应用将会得到深入开发。如今 MOS 的竞争在一定程度上已经转化为各种应用程序和应用商店的竞争,功能和应用的丰富性能够直接影响 MOS 在市场中的竞争地位。

(3)MOS 逐渐走向开源化。目前,智能手机厂商和运营

(下转第 93 页)

表 1　命中概率变化表

E_yE_z	10,30	9,27	8,24	7,21	6,18
命中概率	0.7638	0.7916	0.8206	0.8507	0.8818
相比初始值变化百分比	0	3.64%	7.44%	11.38%	15.45%

表 2　命中概率变化表

E_yE_z	5,15	4,12	3,9	2,6	1,3
命中概率	0.9137	0.9457	0.9760	0.9969	0.9999
相比初始值变化百分比	19.63%	23.82%	27.78%	30.52%	30.91%

由表 1、表 2 可以看到，因目标跟踪误差的改善而使得导弹落点散布概率偏差减小，进而增大了目标的命中概率。对于本文给出的舰船目标而言，当落点散布概率偏差减小 10 倍时，命中概率可提高近 1/3，并且，若要使导弹达到 95%以上的命中概率，需要 y 轴落点散布概率偏差至少在 4m 以下，z 轴落点散布概率偏差在 12m 以下。

结束语　导引头的命中概率是考察精确制导武器作战效能的一个重要方面。本文分析了产生导引头命中概率误差的因素，并针对影响导引头命中概率的主要因素——目标跟踪误差进行了探讨，建立了命中概率模型并针对跟踪误差所产生的影响进行了理论分析。仿真试验说明对于一般舰船目标而言，若要保证导弹 95%以上的命中概率，应控制跟踪误差，使得落点散布在 $4\times12m^2$ 的目标区域内。应当指出的是，随着目标舰船体积的缩小，在同一跟踪误差条件下导弹的命中概率也会降低，因而本文所提供的算例只是为了说明本分析方法的有效性，对于其他类型的舰船目标本分析方法同样适用，可为导弹的指标设计提供相应理论基础。

参 考 文 献

[1] 焦彦华，吴京，徐晖. 反舰导弹落点精度和命中概率的评估和仿真[J]. 电子对抗，2004，4：24-26

[2] 李邦杰，王明海. 基于弹着点椭圆散布的矩形目标命中概率计算[J]. 火力与指挥控制，2005，30：82-84

[3] 杨澍，朱建冲. 有源干扰条件下防空导弹单发命中概率统计模型[J]. 海军工程大学学报，2002，14(1)：62-65

[4] 李相民，张安，史建国. 防区外导弹对舰攻击的效能分析[J]. 火力与指挥控制，2004，29(4)：16-19

[5] 刘千里，朱建冲. 反舰导弹超视距攻击作战的计算机仿真研究[J]. 海军工程大学学报，2004，16(1)：107-110

[6] 徐德坤，张士峰，杨华波. 攻击移动目标的末制导导弹命中概率评估方法[J]. 飞行器测控学报，2008，27(3)：90-94

[7] 王光辉，严建钢，郭荔生，等. 目标位置误差存在条件下反舰导弹命中模型研究[J]. 弹箭与制导学报，2004，24(1)：88-90

[8] 杨春玲，倪晋麟，刘国岁. 转换坐标卡尔曼滤波器的雷达目标跟踪[J]. 电子学报，1999，3：121-123

[9] 徐毓，杨瑞娟，李锋，等. 极坐标到直角坐标转换对卡尔曼滤波的性能影响[J]. 空军雷达学院学报，2001，2：26-30

[10] 何友，修建娟，张晶炜，等. 雷达数据处理及应用[M]. 北京：电子工业出版社，2006

[11] 李红亮，宋贵宝，刘持胜，等. 反舰导弹命中概率模型研究[J]. 海军航空工程学院学报，2005，20(1)：181-184

(上接第 56 页)

商都宣布了自己的开源战略或产品。Android 正是基于开源的 Linux 来建立开放手机联盟，以吸引大批的开发者来对 Android 进行开发，从而取得了巨大的成功。

(4)目前手机和平板电脑已从双核向四核迈进，内存从 1GB 升至 2GB，整体性能已赶超 2～3 年前的主流 PC，屏幕从 4.5 吋向 5.0 吋看齐。面对硬件设备能力不断提升的状况，MOS 对于硬件设备的协调管理，使移动设备在转变到个人数字中心的过程中将起到更大的作用。更高的数据吞吐量、更高的数据处理能力也必然要求 MOS 能够充分挖掘硬件设备的能力并带给用户更多的体验和功能的提升。

(5)为了应对新的需求，一方面 MOS 从支持类似桌面应用风格向支持网页应用转移，如引入 HTML5、CSS、Javascript等元素。另一方面基于 HTML5 的新一代浏览器具有调动手机资源运行应用的能力，开始向网页操作系统演进，出现了手机操作系统和浏览器相互靠拢、融合的发展趋势，网页操作系统开始挑战桌面操作系统[9]。

结束语　研发一个新的操作系统最主要的任务就是体系结构的设计。主流 MOS 体系结构的设计思想给我国研发自主 MOS 提供了多种案例。基于 Linux、层次化平台式架构、优化使用开源软件、继承中创新的思想、开源的免费策略等等，是目前发展国产自主 MOS 的一种务实选择。

目前的 MOS 领域，Android、iOS、WP 三足鼎立。一个大国没有自主的操作系统，不仅信息安全无法得到保障，产业安全也无法得到保障。研发我国自主的 MOS 尤其对促进我国移动互联网产业的发展意义重大，有利于未来占据移动互联网技术与产业的制高点，有助于我国摆脱 Android、iOS、WP 等原生应用的垄断与限制。

参 考 文 献

[1] 石进，陆音，谢立. 操作系统体系结构的研究分析[J]. 计算机科学，2005，32(9)：234-238

[2] 薛进. 基于 Android 的 MSNV7_0 手机客户端的设计与实现. [D]. 北京：北京交通大学，2012

[3] 王伟. 基于 iOS 的旅游系统 ITC 产品的设计与实现[D]. 南京：南京大学，2012

[4] 开发者应知道 iOS 移动操作系统架构图[EB/OL]. http://www.580114.com/Forum/t-36409，2012-09-16

[5] Morris B. Symbian OS 架构手册——手机操作系统设计与演进[M]. 陈广辉，译. 北京：人民邮电出版社，2008

[6] 王仲远. 关于 Windows Phone 7 平台的三个看点 [EB/OL]. http://tech.it168.com/a2012/0401/1333/000001333249_2.shtml，2012-04-06

[7] Windows Phone 7 架构文件完整泄露 [EB/OL]. http://www.htc.cc/2245.html，2010-05-18

[8] 许洪波. 抓住产业技术变革契机发展我国新一代网络操作系统[J]. 信息技术与标准化，2012(11)：5-8

[9] 顾玉良. 智能手机操作系统和几点认识[EB/OL]. http://www.ccf.org.cn/sites/ccf/znzdjs.jsp#4，2012-05-17