Android系统架构及其驱动研究

胡 伟
(广州大学华软软件学院, 广东 广州 510990)

摘 要: Android作为 Google公司推出的专为智能终端定制的操作系统, 已经成为目前智能手机中增长最快的操作系统, 并且必将对手持终端操作系统的发展产生重要而深远的影响。文章研究、分析了该操作系统的架构、代码结构, 并着重研究了其驱动的原理和特点。

关键词: Android; Linux; 架构; 驱动程序
中图分类号: TP316 **文献标识码**: A **文章编号**: 1672 - 0385(2010)04 - 0096 - 06

一、引言

Android是 Google公司在 2007年 11月发布的基于 Linux系统的手机系统平台, 是 HTC、Motorola等企业多款智能手机的操作系统。据世界消费和零售市场研究领导者 NPD Grop数据报告显示, 从今年第一季度开始, Android已经取代苹果的 iPhone成为美国第二大智能手机操作系统和增长速度最快的智能手机操作系统。凭借 Google公司强劲的创新和开发能力以及其开源和免费的特性, 其必将是移动终端操作系统的一场革命。而国内目前研究和介绍该系统的文章较少, 本文着重介绍了该系统的系统架构和驱动工作机理。

二、Android系统架构

Android是为移动设备设计的软件平台, 包括操作系统、中间件和一些关键应用。Android SDK提供了必须的工具和进行应用开发所必须的 Java接口 API。

Android是一个开放的软件系统, 为用户提供了丰富的移动设备开发功能, 从下至上包括 4个层次, 如图 1所示。其中第一层是 Linux内核层, 包括 Linux操作系统及驱动, 依赖于 Linux2.6内核, 不支持 linux2.4内核。如 Android1.0 (release - 1.0) 使用 Linux2.6.25, Android1.6 (sdk - 1.6) 使用 Linux2.6.29。除了标准的 Linux内核外, Android系统还增加了 Binder IPC驱动、WiFi驱动、蓝牙驱动等驱动程序, 为系统运行提供了基础性支持。

第二层是核心的扩展类库, 如 SQLite、WebKit、OpenGL等, 它们可以通过 JAVA本地调用 JNI (Java Native Interface) 的接口函数实现和上层之间的通信。该层由 Android的 Java虚拟机 Dalvik和基础的 Java库为 Java运行环境提供了 Java编程语言核心库的大多数功能。

第三层是包含所有开发所用的 SDK类库和某些未公开接口类库的框架层, 是整个 Android平台核心机制的体现。

第四层是应用层。系统部分应用和第三方开发的应用都是位于这个层次上, 但两者不完全相同, 其中系统应用会用一些隐藏的类, 而第三方的应用, 是基于 SDK基础上开发的。一般 Android开发是在 SDK基础上用 Java编写应用程序, 但本机开发程序包 NDK提供了应用层穿越 Java框架层直接和底层包含了 JNI接口的 C/C++库直接通信的方法。

其中第一层由 C语言实现, 第二层由 C和 C++实现, 第三、四层主要由 Java代码实现。从 Linux操作系统的角度来看, 第一、二层次之间是内核空间与用户空间的分界线, 第一层运行于内核空间, 第二、三、四层运行于用户空间。第二、三层之间, 是本地代码层和 Java代

收稿日期: 2010 - 05 - 27
作者简介: 胡伟 (1973—), 男, 河南信阳人, 讲师, 研究生, 硕士, 主要研究方向为嵌入式 Linux系统软硬件设计、开发。

码层的接口。第三、四层之间是系统 API接口。

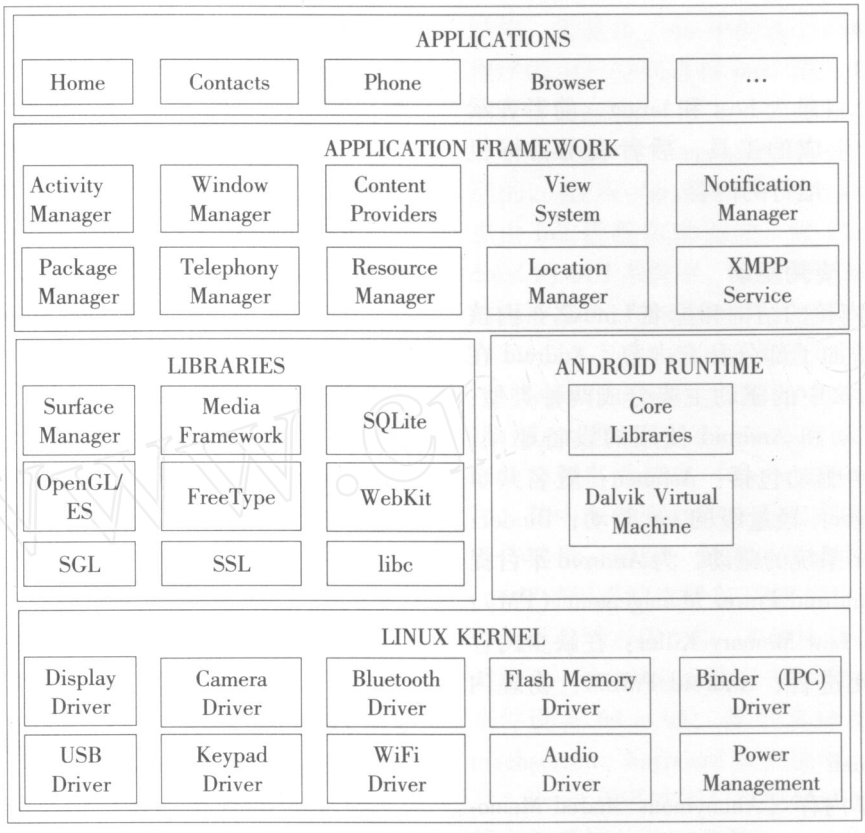

图 1 Android框架图

三、Android代码结构

Android代码包括 3个部分: ①核心工程(Core Project文件夹)是建立 Android系统的基础,在根目录的各个文件夹中; ②扩展工程(External Project文件夹中)是使用其他开源项目扩展功能; ③包 (Package)提供 Android的应用程序和服务。其中既包含了原始 Android的目标机代码,还包括了主机编译工具、仿真环境等。

代码包经过解压缩后,第一级别的目录和文件如下所示:

Mydroid/

——Makefile (全局的 Makefile)

——bionic (这里面是一些基础的库的源代码)

——bootloader (引导加载器)

——build (目录的内容不是目标所用的代码,而是编译和配置所需要的脚本和工具)

——dalvik (JAVA虚拟机)

——development (程序开发所需要的模板和工具)

——external (目标机器使用的一些库)

——frameworks (应用程序的框架层)

——hardware (与硬件相关的库)

——kernel (Linux2.6的源代码)

——packages (Android的各种应用程序)

——prebuilt (Android在各种平台下编译的预置脚本)

——recovery (与目标的恢复功能相关)

——system (Android的底层的一些库)

编译完成后,将在根目录中生成一个 out文件夹,所有生成的 Android代码结构内容均放置在这个文件夹中。out文件夹如下所示:

out/

——CaseCheck.txt

——casecheck.txt

——host

——common

——linux - x86

——target

——common

——product

主要的两个目录为 host和 target, 前者表示在主机 (x86) 生成的工具, 后者表示目标机(默认为 ARMv5) 运行的内容。

四、Android专用驱动

Android中内核的结构和标准 Linux2. 6内核基本相同, 但增加了部分私有内容。Android在标准的 Linux内核中的驱动主要分成两种类型: Android专用驱动和 Android使用的设备驱动。其中主要的专用驱动包括: Ashmen: 匿名共享内存驱动; Logger: 轻量级的 log驱动; Binder: 基于 OpenBinder系统的驱动, 为 Android平台提供 IPC支持; Android Power Management (PM): 电源管理模块; Low Memory Killer: 在缺少内存的情况下, 杀死进程; Android PMEM: 物理内存驱动。

(一) Ashmen

即匿名共享内存 (Anonymous Shared Memory), 为用户空间程序提供分配内存的机制, 实现类似 malloc的功能。其设备节点名称为: /dev/ashmen. 主设备号为 10, 次设备号动态生成。其驱动程序在内核中的头文件和代码路径如下:

Kernel/include/linux/ashmen. h

Kernel/mm/ashmen. c

在用户空间 C libutil库对 Ashmen进行封装并提供接口;

System/core/include/cutils/ashmen. h——简单封装头文件;

System/core/libcutils/ashmen - dev. c——匿名共享内存在用户空间的调用封装;

System/core/libcutils/ashmen - host. c——没有使用。

用于为 Android系统提供内存分配功能。

(二) Binder

为用户层程序提供进程间通信 (IPC) 支持, Android整个系统的运行依赖 Binder驱动。其设备节点名称为: /dev/binder, 主设备号为 10, 次设备号动态生成。

Binder驱动程序在内核中的头文件和代码路径如下:

Kernel/include/linux/binder. h

Kernel/drivers/misc/binder. c

在用户空间 libutil工具库和 Service Manager守护进程中调用 Binder接口提供对整个系统的支持。

Frameworks/base/cmds/servicemanager/——ServerManager守护进程的实现;

Frameworks/base/include/utils/——Binder驱动在用户空间的封装接口;

Frameworks/base/libs/utils/——Binder驱动在用户空间的封装实现。

Binder是 Android中主要使用的 IPC方式, 使用时通常只需要按照模板定义相关的类即可, 不需要直接调用 Binder驱动程序的设备节点。

(三) Logger

该驱动程序为用户层程序提供 log支持, 作为一个工具使用。在用户空间中有 3个设备节点: /dev/log/main、/dev/log/event、/dev/log/radio, 主设备号为 10, 次设备号动态生成。

Logger驱动在内核中的头文件和代码路径如下:

Kernel/include/linux/logger. h

Kernel/driver/misc/logger. c

在 Android的用户空间 logcat程序调用 Logger驱动:

System/core/logcat/——可执行程序。

Logcat是一个可执行程序, 用于提取系统 log信息, 是系统的一个辅助工具。

五、Linux设备驱动在 Android上的使用分析

Linux设备驱动程序, 在为智能手机定制的 Android操作系统中得到了广泛的应用, 下面对几个比较有特色的设备驱动的使用情况进行分析。

(一) Framebuffer显示驱动

Framebuffer驱动在 Linux中是标准的显示设备的驱动。对于 PC系统, 它是显卡的驱动; 对于嵌入式 SOC处理器系统, 它是 LCD控制器或者其他显示控制器的驱动。它是一个字符设备, 在文件系统中设备节点通常是 /dev/fbx。每个系

统可以有多个显示设备，依次用 /dev/fb0、/dev/fb1等来表示。在 Android系统中主设备号为 29，次设备号递增生成。

该驱动在用户空间一般使用 ioctl、mmap等文件系统接口进行操作，ioctl用于获得和设置信息，mmap将 Framebuffer的内存映射到用户空间。驱动也可以直接支持 write操作，用写的方式输出显示内容。显示驱动的架构如图 2所示。

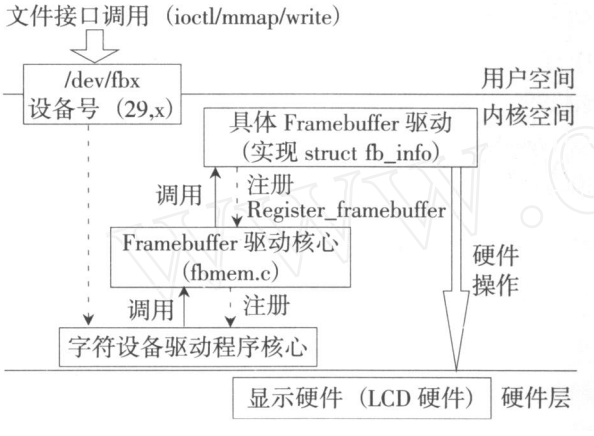

图 2 Framebuffer显示驱动架构图

驱动的主要头文件位于 include/linux/fb.h文件中；驱动核心实现位于 drivers/video/fbmem.c文件中，驱动中核心的数据接口是 fb_info在 fb.h中定义，如下所示。

```
Struct fb_ info {
Int node;
Int flags;
Struct fb_ var_ screeninfo var;      /*变化屏幕信息 */
Struct fb_ fix_ screeninfo fix;      /*固定屏幕信息 */
Struct fb_ ops * fbops;      /* Framebuffer驱动的操作 */
…….
}
```

该结构包含了驱动主要信息，struct fb_ var_ screeninfo和 Struct fb_ fix_ screeninfo两个数据结构对应 FBIOGET_ VSCREENINFO和 FBIOGET_ FSCREENINFO这两个 ioctl从用户空间获得的显示信息。Fb_ ops表示 Framebuffer驱动的操作，通常通过以下函数进行注册：

Int register_ framebuffer (struct fb_ info * fb_ info);

具体的 Framebuffer驱动需要实现 fb_ info结构，实现 fb_ ops中的各个函数指针。从驱动程序的用户空间进行 ioctl调用时，会转换成调用其中的函数。

Android对 Framebuffer驱动的使用方式是标准的，在 /dev/graphic/中的 Framebuffer设备节点由 init进程自动创建，被 libui库调用。Android的 GUI系统中，通过调用 Framebuffer驱动的标准接口，实现显示设备的抽象。

(二) Event输入设备驱动

Input驱动程序是 Linux输入设备的驱动程序，分为游戏杆 (joystick)、鼠标 (mouse和 mice)和事件设备 (Event queue) 3种驱动程序。其中事件驱动程序是目前通用的程序，可支持键盘、鼠标、触摸屏等多种输入设备。Input驱动程序的主设备号是 13，每一种 Input设备从设备号占用 5位，3种从设备号分配是：游戏杆 0～61；Mouse鼠标 33～62；Mice鼠标 63；事件设备 64～95，各个具体的设备在 misc、touchscreen、keyboard等目录中。

Event设备在用户空间使用 read、ioctl、poll等文件系统的接口操作，read用于读取输入信息，ioctl用于获取和设置信息，poll用于用户空间的阻塞，当内核有按键等中断时，通过在中断中唤醒内核的 poll实现。Event输入驱动的架构如图 3所示：

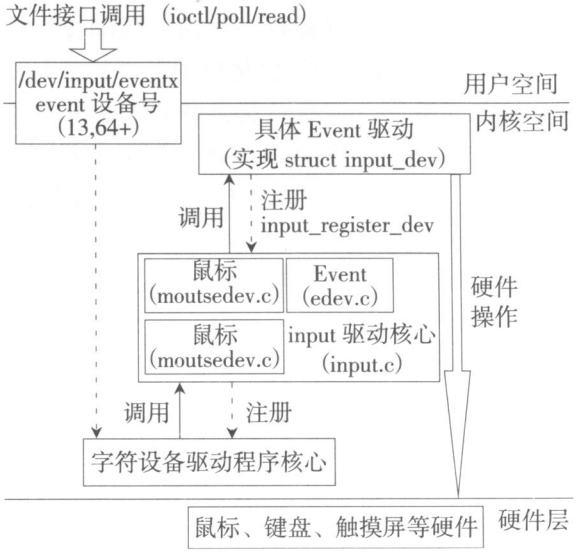

图 3 Event输入驱动架构图

Input驱动程序的头文件在 include/linux/input.h中。Input驱动程序的核心和 Event部分代码，分别位于 Drivers/input/input.c和 Drivers/input/evdev.c中。其中 Input.h中定义了 struct input_ dev结构，它表示 Input驱动程序的各种信息，对于 Event设备分为同步设备、键盘相对设备 (鼠标)、绝对设备 (触摸屏) 等。Event驱动程序需要定义 struct input_ dev结构体，并且通过 input_ register_ device ( ) 函数进行注册。

Int _ _ must_ chect input_ register_ device (struct input_ dev *);

Input设备驱动在内核 menuconfig配置时，配置选项为“Device Drivers” - >“Input Device Drivers”。Event驱动程序配置对应的文件是 driver/input/Kconfig，其配置选项是“Event Interface”，各个具体设备的接口在各自下面进行支持。

Android使用 Event驱动作为标准的输入设备，在 GUI系统中打开 Event驱动程序的设备节点，通常的输入设备是鼠标和触摸屏。Android的 init进程在 /dev/input自动建立 Event设备的节点，被 libui库调用作为系统的输入。

(三) ALSA音频驱动

高级 Linux声音体系 ALSA (Advanced Linux Sound Architecture) 是为音频系统提供驱动的 Linux内核组件，以替代原先的开发声音系统 OSS。它是一个完全开放源代码的音频驱动程序集，除了像 OSS那样提供一组内核驱动程序模块之外，ALSA还专门为简化应用程序的编写提供相应的函数库，与 OSS提供的基于 ioctl等原始编程接口相比，ALSA函数库使用起来要更加方便一些。利用该函数库，开发人员可以方便、快捷地开发出自己的应用程序，细节则留给函数库进行内部处理。所以虽然 ALSA也提供了类似于 OSS的系统接口，但建议应用程序开发者使用音频函数库，而不是直接调用驱动函数。

ALSA驱动的主设备号为 116，次设备号由各个设备单独定义，主要的设备节点如下：

/dev/snd/controlCX——主控制；

/dev/snd/pcmXXXc——PCM数据通道：

/dev/snd/seq——顺序器；

/dev/snd/timer——定义器。

在用户空间中，ALSA驱动通常配合 ALSA库使用，库通过 ioctl等接口调用 ALSA驱动程序的设备节点。对于 ALSA驱动的调用，调用的是用户空间的 ALSA库的接口，而不是直接调用 ALSA驱动程序。

ALSA音频驱动的架构如图 4所示。

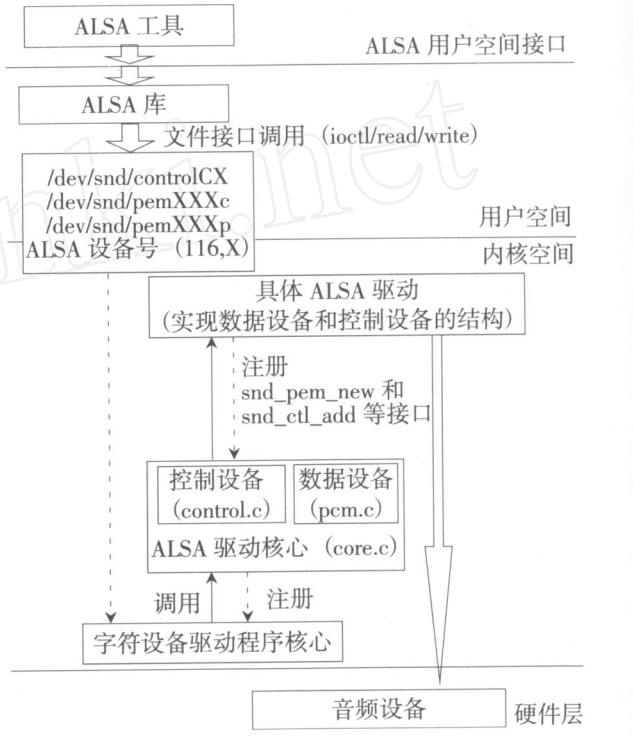

图 4 ALSA音频驱动架构图

ALSA驱动程序的主要头文件是 include/sound/asound.h，驱动核心数据结构和具体驱动的注册函数是 include/sound/core.h，驱动程序的核心实现是 Sound/core/sound.c文件。

ALSA驱动程序使用下面的函数注册控制和设备。

int snd_ pcm_ new (struct snd_ card * card, char * id, int device, int playback_ count, int capture_ count, struct snd_ pcm * * rpcm);

int snd ctl_ add (struct snd_ card * card, struct snd_ kcontrol * kcontrol);

ALSA音频驱动在内核进行 menuconfig配置时，配置选项为“Device Drivers” > “Sound card support” - >“Advanced Linux Sound Architecture”。子选项包含了 Generic sound devices

(通用声音设备)、ARM体系结构支持，以及兼容 OSS的几个选项。ALSA音频驱动配置对应的文件是 sound/core/Kconfig。

Android没有直接使用 ALSA驱动，可以基于 ALSA驱动和 ALSA库实现 Android Audio的硬件抽象层；ALSA库调用内核的 ALSA驱动，Audio的硬件抽象层调用 ALSA库。

六、小结

为智能终端定制的 Android系统，现在已经给世界智能手机带来了变革性的冲击，也为中国智能手机乃至智能终端的发展，提供了新的机遇和挑战。本文在作者对该系统研究和开发的基础上阐述了该操作系统的架构和驱动工作机理，对于项目开发具有实际的参考和借鉴意义。

参考文献:

[1] Google: Home Page, http: //www. android. com/

[2] 野泽哲生，南庭．全球首款 Android手机 GI拆解 [J]. 电子设计应用，2009 (1).

[3] 张仕成．基于 Google Android平台的应用程序开发与研究 [J]. 电脑知识与技术，2009 (28)

[4] 赵亮，张维．基于 Android技术的界面设计与研究 [J]. 电脑知识与技术，2009 (29).

[5] 王志国，侯银涛，石荣刚．Android智能手机系统的文件实时监控技术 [J]. 计算机安全，2009 (12).

[6] 张鹏．中国企业为什么青睐 Android [J]. 通信世界，2010 (3).

(上接第 95页)

按照两种标准可以解释错误内容在错误形式上的更好成绩，那么除了在 DA形式上出现这种倾向外，在 AC形式上也应该表现出这种倾向，即 P3在 AC形式上的成绩比 P2好，但这里的结果刚好相反，P2在 AC形式上的成绩比 P3更高，这个结果便是“形式”和“内容”两种判断标准所无法解释，而这又只能用概率理论进行解释。

五、结论

综上所述，我们认为推理者在条件命题的推理过程中所表现出来的“AC和 MP形式的推理成绩比 MT和 DA形式高”这种不符合形式逻辑规则标准的现象在一定程度上可以用心理模型理论的观点来解释。本文中主要的研究结果是发现概率理论与“形式”和“内容”两种判断标准的观点存在一定的吻合性和相似性。在条件推理的 MP、DA、AC和 MT四种判断形式上，概率理论和两种判断标准都可以较好地解释在研究结果中出现的 MP和 MT在不同具体命题中所表现出来的差异，其实如果内容正确，那么一般情况下它的概率就比较高，反之则相反。被试或者根据不同的主观概率估计值或者根据“内容”正误的标准进行判断，从而对不同具体内容的命题作出不同的判断。在 AC形式上的结果则可以更好地用概率理论进行解释，DA形式的判断结果则可以更好地用两种判断标准进行解释。因此，概率理论和两种判断标准都可以在很大程度上对条件推理的结果进行很好的解释，但各自也都还存在着不足，而两种解释则刚好可以互补一方的不足对条件推理结果进行完整的解释。

当然，本研究只采用了三个条件命题，具体命题的内容与推理者的生活密切程度也存在着不一致性，而且没有对条件命题的前后件概率进行研究，这些问题在以后的研究中可以进一步深入探讨。

参考文献:

[1] 邱江，张庆林．条件推理与概率判断 [J]. 心理科学，2007 (2): 301 - 304.

[2] 胡竹菁，朱丽萍．人类推理的心理学研究 [M]. 北京：高等教育出版社，2007: 141.

[3] 方富熹，唐洪．12岁儿童充分条件假言推理能力发展的个体差异研究 [J]. 心理学报，2OO0 (3): 269 - 275.

[4] 邱江，张庆林．命题内容对青少年条件推理的影响 [J]. 心理发展与教育，2007 (3): 17 - 21.

[5] 胡竹菁．条件推理的条件概率模型述评 [J]. 心理学探新，2008 (2): 25 - 32.

The Insights of Foreign Industrialization Model to Guangdong Industrialization Process

HAN Jiang-bo

(School of Economics, Jinan University, Guangzhou 510632, China)

Abstract: Originating from American financial crisis in 2007, current international economic recession would not only produce an impact on the smooth implementation of strategy of industry and labor transformation, which was proposed by Guangdong Province in 2008, but also pose an unprecedented challenge to the export-oriented industrial development. In this post-crisis era when the global economic recovery is not assured, what alternative can be made? This paper answers this question based on study of foreign industrialization models.

Key words: industrialization; industrialization model; industry and labor transformation

Probability Theory and Dual Structure Model of Conditional Reasoning

LIN Zhu-mei

(Fujian Normal University, Fuzhou 350007, China)

Abstract: This paper uses different conditions as the experimental material to test the relationship between the probability theory and the two criteria of the form and content of dual structure model. The results show that there exists certain compatibility between them, since they both can well interpret the results of conditional reasoning to a large extent, and one can also make up the other's defects.

Key words: conditional reasoning; probability theory; dual structure model

The Study of Android System Architecture and its Drivers

HU Wei

(South China Institute of Software Engineering, GZU, Guangzhou 510990, China)

Abstract: Introduced by the Google Company as a customized intelligent terminal operating system used in the mobile phone, Android has undergone the fastest development and will surely produce an important and far-reaching impact on handset operating system. The article studies this system architecture, code structure, and especially its principles and characteristics of drivers.

Key words: Andriod; Linux; system architecture; driver

The Design and Implementation of .Net-based Educational Research Projects Evaluating System

FANG Bing-qin

(Guangzhou Education Science Institute, Guangzhou 510030, China)

Abstract: This thesis introduces the design and implementation of evaluating system of Educational Research Projects. It illustrates its characteristics, functions, approaches, and employment of core technologies like ASP.NET and FTP, all of which contribute to the great achievements in its practical application.

Key words: project evaluation; NET; B/S; Ftp