基于 UEFI Shell 的 Pre- OS Application 的开发与研究

吴　广　何宗键
(同济大学软件学院　中国　上海　201203)

【摘　要】UEFI(统一可扩展固件接口)是 Intel 提出的下一代 PC Bios(Basic Input/Output System)架构。由于其除了完成传统 Bios 硬件启动检测和初始化等功能外,还构建了一个 C 语言执行环境,可以调用设备驱动,不用依赖操作系统就能在 UEFI Shell 的环境下进行磁盘管理和启动管理,以 PC 操作系统的备份为例,重点探讨 UEFI Shell 下无须操作系统的实现一般 Application 的方法和原理。

【关键词】UEFI;Shell;Bios;Application;系统备份

【Abstract】UEFI (Unified Extensible Firmware Interface) is next generation Bios (Basic Input/Output System) Architecture which promoted by Intel. Except implementation legacy Bios function, UEFI setup advanced language excutive environment, can directly recall device driver and acess Disk & Boot management without OS supported. With above advantage, try to discuss the UEFI Application development theory , apply UEFI into PC OS back up , do the research on backup without OS supported principle and way

【Key words】UEFI;Shell ;Bios ;Application; System backup

0　引言

1981 年 IBM 推出第一台 PC-XT 架构的个人电脑 5150,这台在当时标价为 2880 美元的电子产品拥有 4.77MHZ 的 CPU, 64K 的内存,两个 160KB 的软驱, 拥有彩色显示器还配有 Microsoft 的 MS-DOS 操作系统,IBM 在推出这个产品时已经勾画出了现代 PC 的基本架构, 硬件和软件的结合下的 IBM-XT 标准让其它 PC 制造商和零部件供应商去生产个人电脑, IBM 把一些开机时的硬件启动和检测代码, 以及一些外围 I/O 子程序代码集成到主板上一块 32 KB 大小的只读存储器中,这些程序代码就叫做 Bios, 全称是“Basic Input/Output System”,也就是基本输入/输出系统,它是由汇编语言编写的一段代码,主要负责开机时硬件启动和初始化等工作,这段神奇的代码是控制硬件并且跟操作系统沟通的桥梁, Bios 的功能非常强大,因为操作系统与硬件之间是隔离的,这样操作系统对硬件的调用都必须经过 Bios,甚至很多硬件设计的问题甚至都可以通过 Bios 加以解决,由于 Bios 用的是 16 位汇编代码编写,所有对操作系统的服务都是通过中断来完成的,即使目前的 32 位和 64 位处理器还必须使用实模式来寻址, 启动时间长,代码运行慢等问题,已经成为了传统 Bios 的致命缺陷。

在 2000 年的 Intel 春季 IDF(Intel Develop Forum)上, Intel 提出了 EFI(Extensible Firmware Interface)这一概念,Intel 在当时推出 EFI 主要为将来电脑固件能兼容于各种不同架构的平台, 传统 Bios 已经成为阻碍 PC 向前发展的制约因素之一。由于 UEFI 采用 C 语言而不是传统 Bios 使用的汇编语言,开发人员不需要掌握繁琐的底层硬件语言,因此开发起来更为简单方便,与传统 Bios 相比,UEFI 给带给用户最直观的两个感受是图形化界面和支持鼠标操作, 启动 PC 后 UEFI 进行初始化的时候,不仅对硬件设备进行检测,还能加载硬件设备的驱动程序,而并不需要通过操作系统来加载。这也是 UEFI 跟传统 Bios 很大的区别,UEFI 可以让开发人员在不依赖操作系统的情形下提供各种诊断测试和应用。

1　UEFI 的框架和结构

UEFI 是介于操作系统和底层硬件之间的 Firmware 接口, 基本由 UEFI Application , UEFI Driver , UEFI Framwork, UEFI OS Loader 等部分组成:

1.1　UEFI Application

是指系统的核心应用和调用, 比如 Bios 设置, 一些应用程式,配置管理等等,UEFI Shell (类似传统 Bios 环境下的 DOS) , 一些平台诊断程序和测试程序等等也都属于 UEFI 的 Application;

1.2　UEFI Driver

如同操作系统中的驱动, Driver 用来支持硬件,UEFI Driver 具有跨平台性, 在不同的操作系统下都能应用, 每一个 UEFI Driver 都要求有它独立的数据结构和签名,它会消耗同时也会制造 Protocols ;

UEFI Driver 分为 Device Drivers, Bus Drivers, Hybrid Drivers, Service Drivers, Initializing Drivers, Root Bridge Drivers 等; 每个 Driver 目录下一般都包 Driver.c, ComponentName.c, DriverConfiguration.c, DriverDiagnostics.c, Driver.h, Make.inf 这几个文件;

Make.inf 像设备驱动安装的说明书,定义了驱动的入口,相关的库文件和驱动文件的安装目录,以下是 Make.inf 的示例

```
[Defines]
BASE_NAME     = IdeBus
FILE_GUID     = 69FD8E47-A161-4550-B01A-5594CEB2B2B2
COMPONENT_TYPE = BS_DRIVER

[sources.common]
  idebus.h
  ide.h
  idedata.h
  idebus.c
  ide.c
  ata.c
  atapi.c
  ComponentName.c

[libraries.common]
  EdkFrameworkProtocolLib
  EfiProtocolLib
  EfiDriverLib
  EdkGuidLib

[includes.common]
  $(EDK_SOURCE)\Foundation
  $(EDK_SOURCE)\Foundation\Framework
  $(EDK_SOURCE)\Foundation\Efi
  $(EDK_SOURCE)\Foundation\Core\Dxe
  $(EDK_SOURCE)\Foundation\Include
  $(EDK_SOURCE)\Foundation\Efi\Include
  $(EDK_SOURCE)\Foundation\Framework\Include
  $(EDK_SOURCE)\Foundation\Include\IndustryStandard
  $(EDK_SOURCE)\Foundation\Library\Dxe\Include
  $(EDK_SOURCE)\Sample\Include

[nmake.common]
  IMAGE_ENTRY_POINT=IDEBusControllerDriverEntryPoint
```

1.3　UEFI Framework

Framework 实际上是 intel 对 UEFI 核心组件的统称, 它包括平台组件的驱动,架构性的协议,像 ACPI(Advanced Configuration and Power Interface) ,CSM(Compatibility Support Module)等, 还包括各家不同 Bios 厂商加进去的一些独有的特性,像对传统 Bios 的向下兼容等等。

1.4　UEFI OS Loader

OS Loader 是一种特殊的能控制系统从 Firmware 到 OS 的一种 UEFI Application , 当被加载的时候, OS Loader 就像任何其它的

Application 一样必须使用内存而且必须调用 UEFI Services 和协议去调用装置和 Firmware，一般如果 OS Loader 碰到问题不能成功引导操作系统，它将会自动 release 相关资源后通过 Boot Service 调用 Exit() 来退出.

1.5 Handle & Protocal

这两个概念在 UEFI 中是必须要被提到的,Protocal 有些类似于 C++中的"类"，它拥有私有成员变量,并拥有内部和外部功能以及句柄(Handle)，是一组相关功能以及跟其相关的一些数据,Protocol 必须由驱动产生。

所有的 Protocal 都有跟其对应的 Handle，可以通过这些句柄指向不同的功能函数去驱动不同的硬件。

2 UEFI Application 的原理与开发

UEFI 的 Application 是不依赖系统硬件和软件的跨平台扩展 Firmware，以.efi 的文件存在，能够方便地移植到不同的平台，它的编程模型可以分为两种，一种是基于 EFI 模型的，仅仅能够使用 EFI 数据结构，代码比较小；一种是基于便携式模型的，拥有常用的 C 语言编程接口，更容易导入 ANSI/POSIX 编程模型的程序。

2.1 UEFI Application 的执行流程

本文是基于 UEFI Toolkit 组件来完成 UEFI 下的 Application 的开发，Toolkit 包含 Utilities,C Library,Network Stack,Platform Management, Compression, Database 等组件，包含了开发 Application 的各种库和头文件；包括常用 I/O,内存读写，TCP/UDP 网络协议以及 UEFI 的装置路径、协议和句柄等；

用 Toolkit 生成的.efi 文件可以在 UEFI Shell 中直接运行；

Application 的操作分为三个部分，如下图 1 所示，首先是它必须由 UEFI Loader 来加载，然后是进入 Application 的入口，最后是通过 Exit() 退出应用返回调用 Application 的 UEFI Component .

UEFI Appilication 不会产生 protocols，但也许会消耗 protocols 。

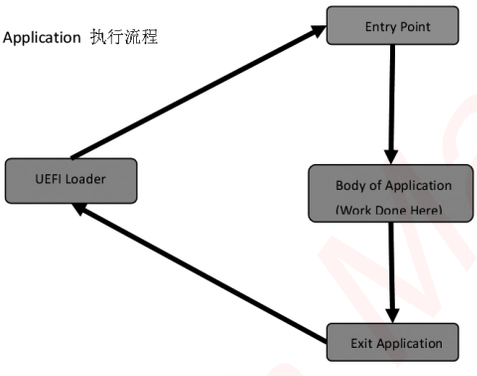

图 1

2.2 UEFI Application 实例——PC 操作系统的备份

以 PC 操作系统的备份为例，因为 UEFI 定义了一种新的分区表格式，被称为 GUID Partition Table (GPT)，GPT 是一种比 MBR(Master Boot Record) 更为灵活和安全的分区机制，可以将操作系统备份在这个 UEFI 支持的 GPT 分区上，用 C 语言写成的 Ghost 主文件因为要使用相应的 UEFI 的库函数，需要 EfiGpt.h,EfishellLib.h ,Efiapi.h 和 Efilib.h 的支援，可以在 EDK_Source/Other/Maintained/Application 下建立应用 Ghost 的目录，但相应的包括源文件列表的 make.inf，或者直接利用 Toolkit 在 Visual 2008 Command Promt 下用 Cl 命令编译和用 Link 命令链接(假设 Ghost.c 在 E 盘根目录下)，会生成 dll 文件格式的 PE/COFF 文件，接下来是将 dll 文件转换成.efi 文件，在 Toolkit 中 Efi_toolkit\build\tools\bin 下会有 Fwimage 的转换程式 (Fwimage app e:\ghost.dll peimage e:\ghost.efi)可以使用，生成的 efi 文件可以直接在 UEFI Shell 下使用。

如果是直接在 UEFI Shell 下写一些简单的应用，注意要将生成的.c 和.h 文件放在 EDK_Source/Other/Maintained/Application/UefiShell 下，并且 EDK_Source/Sample/Platform/DUET/build/DUET.dsc 文件也要做相应修改。

3 结束语

UEFI 是未来 Bios 的继任者，相信越来越多的以前必须在操作系统下运行的程序会被赋予新的生命力直接在 Pre-OS 的环境下运行，无论从安全还是可靠性方面都给用户带来全所未有的变革。

【参考文献】

[1]UEFI Specification Version 2.3, Errata A November, 2009 [S]. http://www.uefi.org/specs/, 2009-06-15

[2]Framework Open Source Community.EFI_Shell_Developer_Guide_Ver0 -91 [EB/OL] [2005-06-27] [Z]. https://efi-shell.tianocore.org/servlets/ProjectDocumentIist

[3]Framework Open Source Community. Edk Getting Started Guide [1]. 0.41 [EB/OL] [2005-01-14] [Z]. https://efi-shell.tianocore.org/servlets/ProjectDocumentList.

[4]UEFI Toolkit Open Source [Z]. https:// sourceforge.net/projects/efi-toolkit.

[5]Insyde H2O Architecture Version 1.2 [S].

[6]The Linux Bootdisk HOW TO (中译版) [M]. http://www.linux.org.tw/CLDP/gb/Bootdisk-HOWTO.html.

作者简介：吴广(1982—)，男，硕士生，同济大学软件学院，主要研究方向为计算机应用。

[责任编辑：汤静]

(上接第 51 页)答：孩子，然后是对方，再次是双方的父母。这也就注定了家庭关系的所有不和谐因素的源头所在，你颠倒了顺序，你忽略了自己。正确的排序应该是这样：首先爱自己，其次爱对方，再次爱双方的父母，最后是爱孩子。是的，你没有看错，对孩子的爱应该排在最后。你只有爱自己，对自己负责，使自己身心健康，乐观上进，有自己的事业基础和交际网络及娱乐生活，你才会变得健康快乐，只有你自己健康快乐，你的爱人才会对你维持最初的激情和兴趣，你才会赢得尊重和理解的基础，你的婚姻才有质量，你的生活才有厚度，你的家庭才会温馨，你的孩子才能成长。我们有很多家庭，把孩子摆在第一位。为了孩子牺牲自己的事业，时间，娱乐，爱人，甚至幸福和健康。在这样亲子环境中成长起来的孩子，就一定会出问题。很多妈妈把自己青春，事业，美丽，健康都当作牺牲品想换取孩子的成长，结果事与愿违，孩子没有培养好，家庭生活也弄得一团糟。原因再简单不过，你忽略了自己，自己的事业，自己的生活。你自己不美丽，不自信，弄得一团糟，你爱人会喜欢你么？夫妻关系搞不好，家庭关系就紧张，冷战，埋怨，赌气，甚至破坏性的行为。这样孩子就会受到影响，培养不出健康快乐的孩子。

苏联教育家马卡连科说过，如果你想毁掉家庭，毁掉孩子，那么最好了的方法就是牺牲你的健康和美丽来为孩子付出吧。

而对于教育而言，如果把教育孩子的成果比作 100 分的话，那么超过 80 分的努力是由于良好的家庭关系和夫妻关系及亲子关系造成的，就是说教育孩子成功 80%的努力来源于亲子关系，只有不到 20%的努力来源于教育方法和科学理念，诸如：父母基本素质，母亲身体健康，顺产，母乳喂养，闪卡干预，七田真教室，蒙氏教育，好奶粉的选择，多带孩子出去玩，母亲陪伴，父亲游戏，至少半年的爬行，每天晚上讲故事，音乐熏陶，亲子课程，诸如此类。

父母是孩子最初也是最重要的老师，父母的言传身教对孩子的影响教育是无可比拟的，孩子最初的学习就是模仿，我们再多的说教和监管都比不上我们究竟是怎样一个人，有着如何的生活态度和生活状态来的重要。

给孩子一个健康良好的家庭环境，让孩子身心自由健康的成长吧

嵌入式资源免费下载

总线协议：

1. 基于 PCIe 驱动程序的数据传输卡 DMA 传输
2. 基于 PCIe 总线协议的设备驱动开发
3. CANopen 协议介绍
4. 基于 PXI 总线 RS422 数据通信卡 WDM 驱动程序设计
5. FPGA 实现 PCIe 总线 DMA 设计
6. PCI Express 协议实现与验证
7. VPX 总线技术及其实现
8. 基于 Xilinx FPGA 的 PCIE 接口实现
9. 基于 PCI 总线的 GPS 授时卡设计
10. 基于 CPCI 标准的 6U 信号处理平台的设计
11. USB30 电路保护
12. USB30 协议分析与框架设计
13. USB 30 中的 CRC 校验原理及实现
14. 基于 CPLD 的 UART 设计
15. IPMI 在 VPX 系统中的应用与设计
16. 基于 CPCI 总线的 PMC 载板设计
17. 基于 VPX 总线的工件台运动控制系统研究与开发
18. PCI Express 流控机制的研究与实现
19. UART16C554 的设计
20. 基于 VPX 的高性能计算机设计
21. 基于 CAN 总线技术的嵌入式网关设计
22. Visual C 串行通讯控件使用方法与技巧的研究
23. IEEE1588 精密时钟同步关键技术研究
24. GPS 信号发生器射频模块的一种实现方案
25. 基于 CPCI 接口的视频采集卡的设计
26. 基于 VPX 的 3U 信号处理平台的设计
27. 基于 PCI Express 总线 1394b 网络传输系统 WDM 驱动设计

VxWorks：

1. 基于 VxWorks 的多任务程序设计

2. 基于 VxWorks 的数据采集存储装置设计
3. Flash 文件系统分析及其在 VxWorks 中的实现
4. VxWorks 多任务编程中的异常研究
5. VxWorks 应用技巧两例
6. 一种基于 VxWorks 的飞行仿真实时管理系统
7. 在 VxWorks 系统中使用 TrueType 字库
8. 基于 FreeType 的 VxWorks 中文显示方案
9. 基于 Tilcon 的 VxWorks 简单动画开发
10. 基于 Tilcon 的某武器显控系统界面设计
11. 基于 Tilcon 的综合导航信息处理装置界面设计
12. VxWorks 的内存配置和管理
13. 基于 VxWorks 系统的 PCI 配置与应用
14. 基于 MPC8270 的 VxWorks BSP 的移植
15. Bootrom 功能改进经验谈
16. 基于 VxWorks 嵌入式系统的中文平台研究与实现
17. VxBus 的 A429 接口驱动
18. 基于 VxBus 和 MPC8569E 千兆网驱动开发和实现
19. 一种基于 vxBus 的 PPC 与 FPGA 高速互联的驱动设计方法
20. 基于 VxBus 的设备驱动开发
21. 基于 VxBus 的驱动程序架构分析

Linux：

1. Linux 程序设计第三版及源代码
2. NAND FLASH 文件系统的设计与实现
3. 多通道串行通信设备的 Linux 驱动程序实现
4. Zsh 开发指南-数组
5. 常用 GDB 命令中文速览
6. 嵌入式 C 进阶之道
7. Linux 串口编程实例
8. 基于 Yocto Project 的嵌入式应用设计
9. Android 应用的反编译
10. 基于 Android 行为的加密应用系统研究
11. 嵌入式 Linux 系统移植步步通
12. 嵌入式 CC++语言精华文章集锦
13. 基于 Linux 的高性能服务器端的设计与研究
14. S3C6410 移植 Android 内核
15. Android 开发指南中文版
16. 图解 Linux 操作系统架构设计与实现原理（第二版）

17. 如何在 Ubuntu 和 Linux Mint 下轻松升级 Linux 内核
18. Android 简单 mp3 播放器源码
19. 嵌入式 Linux 系统实时性的研究
20. Android 嵌入式系统架构及内核浅析
21. 基于嵌入式 Linux 操作系统内核实时性的改进方法研究
22. Linux TCP IP 协议详解

Windows CE:

1. Windows CE.NET 下 YAFFS 文件系统 NAND Flash 驱动程序设计
2. Windows CE 的 CAN 总线驱动程序设计
3. 基于 Windows CE.NET 的 ADC 驱动程序实现与应用的研究
4. 基于 Windows CE.NET 平台的串行通信实现
5. 基于 Windows CE.NET 下的 GPRS 模块的研究与开发
6. win2k 下 NTFS 分区用 ntldr 加载进 dos 源代码
7. Windows 下的 USB 设备驱动程序开发
8. WinCE 的大容量程控数据传输解决方案设计
9. WinCE6.0 安装开发详解
10. DOS 下仿 Windows 的自带计算器程序 C 源码
11. G726 局域网语音通话程序和源代码
12. WinCE 主板加载第三方驱动程序的方法
13. WinCE 下的注册表编辑程序和源代码
14. WinCE 串口通信源代码
15. WINCE 的 SD 卡程序[可实现读写的源码]
16. 基于 WinCE 的 BootLoader 研究

PowerPC:

1. Freescale MPC8536 开发板原理图
2. 基于 MPC8548E 的固件设计
3. 基于 MPC8548E 的嵌入式数据处理系统设计
4. 基于 PowerPC 嵌入式网络通信平台的实现
5. PowerPC 在车辆显控系统中的应用
6. 基于 PowerPC 的单板计算机的设计
7. 用 PowerPC860 实现 FPGA 配置
8. 基于 MPC8247 嵌入式电力交换系统的设计与实现

9. 基于设备树的 MPC8247 嵌入式 Linux 系统开发
10. 基于 MPC8313E 嵌入式系统 UBoot 的移植

ARM：

1. 基于 DiskOnChip 2000 的驱动程序设计及应用
2. 基于 ARM 体系的 PC-104 总线设计
3. 基于 ARM 的嵌入式系统中断处理机制研究
4. 设计 ARM 的中断处理
5. 基于 ARM 的数据采集系统并行总线的驱动设计
6. S3C2410 下的 TFT LCD 驱动源码
7. STM32 SD 卡移植 FATFS 文件系统源码
8. STM32 ADC 多通道源码
9. ARM Linux 在 EP7312 上的移植
10. ARM 经典 300 问
11. 基于 S5PV210 的频谱监测设备嵌入式系统设计与实现
12. Uboot 中 start.S 源码的指令级的详尽解析
13. 基于 ARM9 的嵌入式 Zigbee 网关设计与实现
14. 基于 S3C6410 处理器的嵌入式 Linux 系统移植

Hardware：

1. DSP 电源的典型设计
2. 高频脉冲电源设计
3. 电源的综合保护设计
4. 任意波形电源的设计
5. 高速 PCB 信号完整性分析及应用
6. DM642 高速图像采集系统的电磁干扰设计
7. 使用 COMExpress Nano 工控板实现 IP 调度设备
8. 基于 COM Express 架构的数据记录仪的设计与实现
9. 基于 COM Express 的信号系统逻辑运算单元设计
10. 基于 COM Express 的回波预处理模块设计
11. 基于 X86 平台的简单多任务内核的分析与实现