# 基于 Framework 的固件卷结构的分析

陈 楣[1],周振柳[2],许榕生[1,2]
CHEN Mei[1],ZHOU Zhen-liu[2],XU Rong-sheng[1,2]

1.福州大学 数学与计算机科学学院,福州 350002
2.中国科学院 高能物理研究所 计算中心,北京 100049
1.College of Mathematics and Computer Science,Fuzhou University,Fuzhou 350002,China
2.Computing Center,Institute of High Energy Physics,Chinese Academy of Sciences,Beijing 100049,China

**CHEN Mei,ZHOU Zhen-liu,XU Rong-sheng.Analysis of firmware volume format based on framework.Computer Engineering and Applications,2007,43(15):86-88.**

**Abstract**: The analysis of firmware volume format based on the framework is presented.We expound how one or more files constitute a whole firmware volume through hiberarchy encapsulation and analyse the features of encapsulation.Finally,the correctness of the analysis is validated through trial.

**Key words**: BIOS;EFI;UEFI;framework;firmware volume

**摘 要**:对固件卷的结构进行了系统的分析,详细阐述了多个文件数据进行层层封装有机构成一个整体的情况,并分析了此结构的特点。最后通过实验验证了分析的正确性。

**关键词**:BIOS;EFI;UEFI;架构;固件卷
**文章编号**:1002-8331(2007)15-0086-03 **文献标识码**:A **中图分类号**:TP311

## 1 引言

在 BIOS 出现的二十多年里,只有极少数的公司掌握 BIOS 的核心技术,国内没有 BIOS 生产厂商。随着 EFI(Extensible Firmware Interface Specification)规范[1]的提出,Intel 公司将 EFI 规范的一个具体实现的基础代码作为开放源代码,为我国在该领域的发展带来了新的契机。

基于 framework(架构)[5]的固件卷结构的分析是分析"架构"内部模块、执行流程、运行机制等的基础。本文首先论述了 EFI、Framework 和 Tiano 三者之间的关系,进而对固件卷进行简要的介绍,接着重点对固件卷的结构进行系统分析的同时,也分析了此结构的特点,最后通过实验验证了分析的正确性。

## 2 EFI、Framework 和 Tiano 三者的联系

EFI 为固件和操作系统的接口,是一种规范。它定义了固件必须实现的一系列接口和结构,同时它也定义了操作系统在启动时可以用到的一系列接口和结构。通过这种方式,操作系统不需要定制平台固件,就可以在多种系统设计上启动。同样,平台固件进行性能和功能的改进时,不必把新的代码写入操作系统启动过程。因此,EFI 为实现产品差异化,提供了极大的可扩展性和可定制性。现在 EFI 已发展成为 UEFI(Unified Extensible Firmware Interface)[2]。EFI 和 UEFI 规范的具体实现将是对 BIOS 的取代。

Framework(架构)是对 Intel 平台创新架构(the Intel Platform Innovation Framework for EFI)是简称。"架构"是 EFI 和 UEFI 规范的一种产品级实现,采用全新的启动模式取代传统 BIOS 的"PC-AT"型的启动模式。

Tiano 是 Intel 公司用来描述"架构"的代码名称,通常 Tiano 和 Framework 是等价的,可以互换使用。Intel 公司将"架构"的基础代码作为开放源代码,公布在网站 www.tianocore.org 上,希望更多的人参与其中,共同发展。

## 3 固件卷简介

在"架构"里,每个功能组件或模块都包含在固件卷(firmware volume,fv)里,一个简单的固件只包含一个固件卷,而复杂的固件通常包含多个固件卷。可在预启动过程中载入运行的固件映像(Firmware Image)也以固件卷作为存储格式,固件映像的存放位置不局限在固件里,可存放在光盘、硬盘、软盘等存储介质上。

固件作为计算机系统加电后,第一道程序运行的地点,有着不可忽略的重要性。目前我国国内还没有 BIOS 生产厂商,"架构"作为 EFI 和 UEFI 规范的产品级实现,对其固件卷结构的分析有着一定的必要性。对固件卷结构的分析有助于在已有的固件卷中分析出功能模块,或在已有的固件卷中增加或删除功能模块。

## 4 固件卷的结构分析

总得来说,固件卷由一个或多个固件文件系统(firmware file system,ffs)封装而成,固件文件系统由一个或多个段

**作者简介**:陈楣(1978-),女,硕士研究生,主要研究方向:网络安全;周振柳,男,博士研究生,主要研究方向:网络安全、可信计算;许榕生,男,研究员,博导,主要研究方向:网络安全。

(section)封装而成,而段里面还可包含段。固件卷、固件文件系统和段分别由首部(header)和数据区构成,其中固件文件系统还可能有尾部。如图 1 所示。

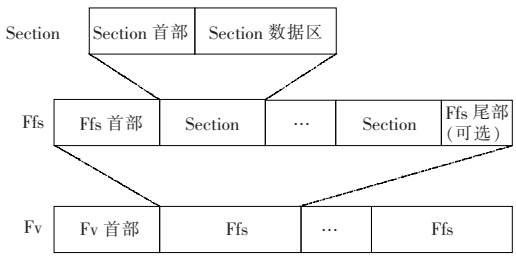


**图 1　固件卷结构图**

文件数据存放在段的数据区,多个文件数据通过层层封装构成固件卷。由于固件文件系统首部没有保存其数据区中各个段的位置及相关信息, 数据区中一个或多个段是紧密相连,中间的填充字节最多不超过 4 Byte,需要由各个段的首部识别出各自的段。固件卷首部与固件卷的数据区的关系亦是如此。因此在固件卷中定位某一文件数据时,需要对整个固件卷进行遍历。此结构的优点是存储紧凑,节省了存储空间,同时保证了模块化存储。由于固件的容量比较有限,通常固件卷的大小也是很有限的,因此,遍历所耗费的时间很少。

## 4.1　段的结构分析

段由段首部和段数据区构成。段首部的结构比较简单,长度为 4 Byte。前 3 个字节表示段的长度。第 4 个字节表示段的类型,指出了段数据区中内容所属的类型。第 4 字节中的数值与类型的对应关系如表 1 所示。其中类型 EFI_SECTION_COMPRESSION 表明段数据区中的内容可以为压缩的。当段的类型为 EFI_SECTION_COMPRESSION 或者 EFI_SECTION_GUID_DEFINED 时, 段数据区的内容可以是由一个或多个段组成。

**表 1　段首部 Type 域的数值与段数据区存放的文件类型的对应关系**

| Type | 类型 |
|---|---|
| 0x01 | EFI_SECTION_COMPRESSION |
| 0x02 | EFI_SECTION_GUID_DEFINED |
| 0x10 | EFI_SECTION_PE32 |
| 0x11 | EFI_SECTION_PIC |
| 0x12 | EFI_SECTION_TE |
| 0x13 | EFI_SECTION_DXE_DEPEX |
| 0x14 | EFI_SECTION_VERSION |
| 0x15 | EFI_SECTION_USER_INTERFACE |
| 0x16 | EFI_SECTION_COMPATIBILITY16 |
| 0x17 | EFI_SECTION_FIRMWARE_VOLUME_IMAGE |
| 0x18 | EFI_SECTION_FREEFORM_SUBTYPE_GUID |
| 0x19 | EFI_SECTION_RAW |
| 0x1B | EFI_SECTION_PEI_DEPEX |

由于类型为 EFI_SECTION_COMPRESSION 或者 EFI_SECTION_GUID_DEFINED 的段的数据区中包含着一个或多个段,因此,这种结构具有了跟文件系统中的目录同样的功能。另外,在段中没有进行有效性的校验,段的校验的任务是由它的上层——固件文件系统来完成的。

## 4.2　固件文件系统的结构分析

固件文件系统由固件文件系统首部、固件文件系统数据区和固件文件系统尾部构成。固件文件系统可以不包含尾部,尾部是可选的。

固件文件系统首部的结构如图 2 所示。

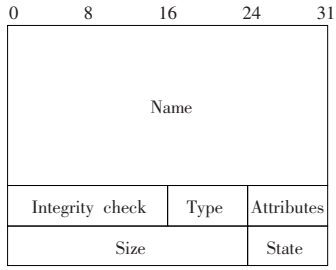


**图 2　固件文件系统首部的结构**

固件文件系统首部的各字段的含义如下:

(1)Name:固件文件系统的 guid,128 bit,占 16 Byte。

(2)Integrity check:固件文件系统的 CRC 校验和,占 2 Byte。完整性校验保证了固件文件系统的有效性。

(3)Type:固件文件系统的类型,占 1 Byte。Type 域的数值和类型的对应关系如表 2 所示。

**表 2　固件文件系统首部的 Type 域与固件文件系统类型的对应关系**

| Type | 类型 |
|---|---|
| 0x00 | EFI_FV_FILETYPE_ALL |
| 0x01 | EFI_FV_FILETYPE_RAW |
| 0x02 | EFI_FV_FILETYPE_FREEFORM |
| 0x03 | EFI_FV_FILETYPE_SECURITY_CORE |
| 0x04 | EFI_FV_FILETYPE_PEI_CORE |
| 0x05 | EFI_FV_FILETYPE_DXE_CORE |
| 0x06 | EFI_FV_FILETYPE_PEIM |
| 0x07 | EFI_FV_FILETYPE_DRIVER |
| 0x08 | EFI_FV_FILETYPE_COMBINED_PEIM_DRIVER |
| 0x09 | EFI_FV_FILETYPE_APPLICATION |
| 0x0b | EFI_FV_FILETYPE_FIRMWARE_VOLUME_IMAGE |
| 0Xf0 | EFI_FV_FILETYPE_FFS_PAD |

(4)Attributes:固件文件系统的属性,占 1 Byte。

(5)Size:固件文件系统的长度,占 3 Byte。

(6)State:固件文件系统的状态,占 1 Byte。

固件文件系统的首部的 Integrity check 域中既包含了对固件文件系统首部的校验,也包含了对其数据区的校验。

固件文件系统的数据区中存放着一个或多个段,可根据段的首部,识别出各个段。

固件文件系统的尾部占 2 Byte,当首部中的 attributes 域的最低位为 1 时,表示固文件系统具有尾部。具有尾部的固件文件系统的文件类型不能是 EFI_FV_FILETYPE_FFS_PAD,即用于填充的固件文件系统,也不允许数据区的长度为 0。

## 4.3　固件卷的结构分析

固件卷由固件卷首部和固件卷数据区构成。

固件卷首部的结构如图 3 所示。

固件卷首部的各字段的含义如下:

(1)ZeroVector:类型为 unsigned char 长度为 16 的一个数组,占 16 Byte。

(2)FileSystemGuid:固件卷的 guid,占 16 Byte。

(3)FvLength:固件卷的长度,占 8 Byte。

(4)Signature:固件卷首部的签名,占 4 Byte,内容为“_FVH”。

(5)Attributes:固件卷的属性,占 4 Byte。

(6)HeaderLength:固件卷首部的长度,占 2 Byte。目前的固件卷的首部的长度通常为 72 Byte。

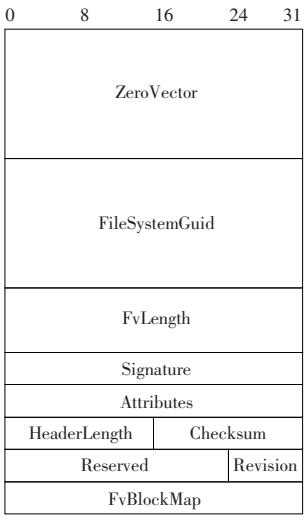


**图 3 固件卷首部结构图**

(7)Checksum:固件卷首部的校验和,占 2 Byte。用来校验固件卷首部的有效性。

(8)Reserved:类型为 unsigned char 长度为 3 的一个数组,占 3 Byte。目前未用,初始化为 0。

(9)Revision:版本号,占 1 Byte。

(10)FvBlockMap:将固件卷看作是一个逻辑的固件块设备。前 2 Byte 存放块的数目,后 2 Byte 存放块的长度,占 4 Byte。块的数目和块的长度的乘积等于固件卷的长度。

固件卷的数据区存放着一个或多个固件文件系统。固件卷数据区有可能包含一个特殊的固件文件系统——VTF(Volume Top File),该文件必须放在固件卷数据区的末尾,通过固件文件系统首部的 name 域可识别出 VTF。如果数据区里的固件文件系统没有占满整个数据区,则需要加入文件类型为 EFI_FV_FILETYPE_FFS_PAD 的填充固件文件系统。如果该固件卷有 VTF,那么用于填充的固件文件系统位于 VTF 之前,确保 VTF 处于固件卷数据区的末尾。

## 5 实验结果验证

在网站 www.tianocore.org 上,下载 EDK 代码。EDK 代码包含有“架构”(即 Tiano 代码)移植到 windows 应用平台的 NT32 仿真代码。用 Microsoft visual studio 编译运行 NT32 仿真代码,可得到对平台启动固件的 NT32 仿真。

在编译生成的文件夹里可找到用于恢复的固件卷 FvRecovery.fv。根据固件卷结构编写相应的结构分析软件对 FvRecovery.fv 二进制流文件进行分析,从中分析得到的模块与最初用于构建恢复固件卷的模块相一致,从而验证了固件卷结构分析的正确性。

## 6 结束语

在国内还未有自主研发的平台启动固件时,根据固件卷的结构,从固件卷中读出功能模块,进而分析“架构”的执行流程,运行机制或在“架构”里进行功能模块的增删等都将是非常有意义的工作。(收稿日期:2006 年 9 月)

## 参考文献:

[1] Intel Corporation.Extensible Firmware Interface Specification Version 1.10[S],2002.

[2] Unified EFI Inc.Unified Extensible Firmware Interface Specification Version 2.0[S],2006.

[3] Intel Corporation.EFI Developer Kit (EDK) getting started guid Version 0.41[S],2005.

[4] Vincent Girard-Reydet.EDK reference manual draft for review,2005.

[5] Intel Corporation.Intel platform innovation framework for efi architecture specification[S],2003.

(上接 36 页)

为缩水因子,$L$ 为搜索集合中的模板数。

## 3 实验及总结

在 Windows XP 平台下,CPU 主频为 PⅣ 2.8 G,用 VC++ 6.0 实现的地名识别系统中,样本集为 5 男 3 女训练的如北京、上海、昆明等 100 个城市地名孤立词,采用的采样频率为 8 kHz,帧长 30 ms,帧移 10 ms,每帧信号用 Hamming 窗相乘,特征提取使用的是 36 阶 MFCC(Mel 频率倒谱系数)参数,第一阶段的位置为 $N/3$ 向上取整处,$Ms$ 为 $N/6$ 向上取整,$Ml$ 为 $N/3$ 向上取整,缩水因子 $k$ 为 2/3,分别进行了特定人与非特定人的语音识别,实验结果如表 1。

**表 1 实验结果**

| 实验序号 | 参考模板数 | 传统的 DTW 算法 | | | 并行分段裁剪的 DTW 算法 | | |
|---|---|---|---|---|---|---|---|
| | | 平均耗时/s | 正确识别率/% | | 平均耗时/s | 正确识别率/% | |
| | | | 特定人 | 非特定人 | | 特定人 | 非特定人 |
| 1 | 10 | 0.134 | 100 | 98 | 0.071 | 100 | 98 |
| 2 | 50 | 0.668 | 99 | 97 | 0.325 | 99 | 96 |
| 3 | 100 | 1.322 | 98 | 95 | 0.644 | 97 | 93 |

可见,采用并行分段裁剪的 DTW 算法比传统的 DTW 算法的平均耗时有了显著的减少,以 100 个词为例,平均耗时节省了 53.2%,但该算法占用的存储空间较大,以 100 个词为例,占用的存储空间是传统的 DTW 算法的 50 倍左右。对于非特定人,随着词汇量的增加,正确识别率会有所下降,这与不同人的发音习惯,发音准确性有关,但平均识别率均在 93%以上,达到了预期的效果。

总之,DTW 算法在孤立词识别中是非常适用的。但如何更进一步减少 DTW 算法的运算量,提高识别率仍需继续探讨。(收稿日期:2006 年 12 月)

## 参考文献:

[1] 蔡莲红,黄得智,蔡锐.现代语音技术基础与应用[M].北京:清华大学出版社,2003:232-244.

[2] Quatieri T F.离散时间语音信号处理—原理与应用[M].北京:电子工业出版社,2004.

[3] Deller J R,Proakis J G,Hansen J H L.Discrete time processing of speech signals[M].England:Macmillan Publishing Company,1993.

[4] Rabiner L,Huang Juangbiing.Fundamentals of speech recognition.Pearson Education POD,1993.

[5] Wrigley S N.Speech recognition by dynamic time warping [EB/OL].http://www.dcs.shef.ac.uk/~stu/com326/index.html.

[6] 封伶刚,王秀萍.一种新的基于 LGB 和 DTW 的模板训练算法[J].计算机工程与应用,2005,41(26):85-88.