HZ BOOKS
华章科技

国内首部UEFI专著，资深UEFI专家兼布道者撰写，英特尔中国研究院院长吴甘沙及大数据和数据经济实验室研究总监周鑫联袂推荐

以实战为导向，全面介绍UEFI的使用、深入剖析UEFI的原理，为开发UEFI应用和驱动程序提供了翔实的指导

實戰

戴正华 著

U ng

UEFI 原理与编程

机械工业出版社
China Machine Press

實戰

UEFI: Principles and Programming

UEFI 原理与编程

戴正华 著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

UEFI 原理与编程 / 戴正华著 . —北京：机械工业出版社，2015.1
（实战）

ISBN 978-7-111-48729-6

I. U… II. 戴… III. 程序设计 IV. TP311.1

中国版本图书馆 CIP 数据核字（2014）第 282707 号

UEFI 原理与编程

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）
责任编辑：高婧雅　　责任校对：殷　虹
印　　刷：三河市宏图印务有限公司　　版　　次：2015 年 1 月第 1 版第 1 次印刷
开　　本：186mm × 240mm　1/16　　印　　张：26
书　　号：ISBN 978-7-111-48729-6　　定　　价：89.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991　88361066　　投稿热线：010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzjsj@hzbook.com

Preface 序

收到书稿，看到熟悉的作者名字，立刻有种开始仔细阅读的冲动。

翻完最后一页，掩卷体会，其中充满了正华苦心钻研技术的精神与回馈社会的热情。

在 IT 行业各个细分领域里，系统程序员需要埋头于汇编语言 /C 语言的编辑器、调试器和体系结构的大部头手册，堪称最枯燥、最辛苦的工种之一。在系统领域取得成就的程序员需要有丰富的知识、扎实的技巧和踏实的态度，缺一不可。正华在我们团队开始系统程序员的职业生涯，他从一个普通的 C 程序员，以孜孜不倦的学习态度，快速成长为一个出色的编程语言和编译器系统的开发骨干。然后，他去了更广阔的天地，直到今天成为 UEFI 专家。正华的钻研精神和成长经历，是本书在专业内容以外，更值得读者学习和体会的精神食粮。

UEFI 是一个很出色的方向。尤其是对有志于学习计算机体系结构和操作系统的初学者来说，UEFI 是一个复杂度适中、内容涵盖度超高的学习平台。我们这一代的系统程序员，都是从奔腾 /586 CPU、DoS、Windows95、Linux Kernel 2.0 时代成长起来的。仔细阅读、修改、调试这些操作系统平台，是我们成长最有效的途径。但是，现代主流桌面操作系统，比如 Windows、Linux Ubuntu、Fedora，经过 20 多年的发展，功能超级丰富，系统设计超级复杂、代码量超级多，已经不适合作为体系结构和操作系统初学者的学习对象。UEFI 在恰当的时候填补了这个空缺。相比它的前辈 BIOS，UEFI 已经涵盖现代操作系统的核心内核功能和外设模块，可以视为一个最最简化版的操作系统。学通、学透 UEFI，可以帮助操作系统初学者快速入门。同时，对于安全领域和 X86 嵌入式领域来说，UEFI 是不可或缺的知识构成部分。

本书是正华热情回馈社会的成果。书里详细、具体地再现了一个学习者对 UEFI 的学习历程。相对于枯燥的手册，这种历程回顾方式的学习材料，更具有可阅读性、可学习性、可

操作性。事无巨细地列举了所有环境细节、命令参数，能极大地帮助初学者绕过障碍，专注于核心内容的学习，缩短学习过程。这种貌似简单，实则烦琐的学习总结，需要极大的精力和时间来准备与撰写，正华的回馈精神体现无疑。

希望读者们从内容中获取知识的同时，也体会学习和回馈的精神。

周鑫

英特尔中国研究院　大数据和数据经济实验室　研究总监

2014 年 11 月 23 日

Preface 前　言

BIOS 已经逐渐成为历史，UEFI 目前开始全面取代 BIOS。

UEFI 全称统一可扩展固件接口，是 UEFI 论坛发布的一种操作系统和平台固件之间的标准。它之所以能迅速取代 BIOS，源于硬件平台的发展以及 UEFI 相对于 BIOS 的巨大优势。UEFI 可编程性好，可扩展性好，性能高，安全性高。

随着 64 位 CPU 取代 32 位 CPU，UEFI 也完成了对 BIOS 的取代。市场已经完成了这种转变，对固件开发者来说，这是一种挑战，固件开发模式已经发生了巨大的改变；这也是一个机遇，UEFI 为计算机系统提供了更丰富的功能，为开发者提供了更强大的开发手段。

为什么写这本书

21 世纪什么最重要？大家都知道答案。人才是需要培养的。培养 UEFI 开发者最重要的是相关开发资料。但是目前 UEFI 相关的开发资料十分匮乏。UEFI 是一种全新的技术，是对 BIOS 的一种革命，BIOS 时代的技术积累很难转移给 UEFI。在 BIOS 时代，BIOS 开发采用汇编语言并且与硬件特性息息相关，技术被几大公司垄断，进入 BIOS 开发领域的门槛高，离开的代价大，相关的技术资料很难在广大程序员中流传。进入 UEFI 时代后，这种开发特点因为惯性的作用依然会延续一段时间。

UEFI 开发对广大 C/C++ 开发者敞开了大门，进入和离开 UEFI 的门槛降低了很多。相信会有更多的开发者进入和离开固件开发领域，这种流动性也会促进这个领域的繁荣，相关的开发资料也会越来越多，越来越精彩。

本书特色

本书是国内第一本介绍 UEFI 开发的书籍，希望这本书能改善 UEFI 开发者无处学习的困境。本书以实战为主，辅以相关理论知识，重要的章节还附有完整的应用程序。力图让读

者不仅知道如何开发 UEFI 程序，还让读者了解为什么这样编程。本书提供了丰富的源代码，让读者可以在实践中快速掌握编程要点。

读者对象

本书内容循序渐进，非常适合刚刚接触 UEFI 开发的初学者、大专院校在校的学生，也适合对 UEFI 有一定了解的专业开发者。

如何阅读本书

本书假定读者有基本的 C 和 C++ 知识，但并不要求读者有固件开发的相关知识。本书将一步一步引导读者熟悉 UEFI，随着本书的不断深入，使读者成为 UEFI 开发专家。

本书从实战的角度介绍如何开发 UEFI 应用和驱动，用生动的实例介绍如何使用 UEFI 提供的服务，同时我们将讲述所需的理论知识以及 UEFI 的内部实现。本书既可以作为 UEFI 爱好者的入门教材，也可以帮 UEFI 开发者更加深入了解 UEFI 内部实现。

本书不是 UEFI 开发的参考手册，所以读者在学习过程中最好准备 UEFI Spec 2.4 以备参考。本书提供了大量的实例程序，读者还需下载 TianoCore 的源码，边学习边实践。

本书内容编排如下：

第 1 章 UEFI 概述介绍 UEFI 发展的历史及 UEFI 理论知识，UEFI 系统启动到结束可分为 7 个过程，本章从程序开发的角度阐述了这个过程的执行流程。

第 2 章 介绍 UEFI 开发环境的搭建，EDK2 提供了 Linux、Windows、Cygwin 等多种开发环境，本章逐步讲述了 Linux 和 Windows 下 UEFI 开发环境的搭建，如何制作 OVMF 固件，以及如何制作启动盘。

第 3 章 介绍 EDK2 工程模块及包的概念，主要包括 .inf 文件、.dsc 文件和 .dec 文件的格式与用法。其中，主要的工程模块包括 UEFI 应用程序模块、驱动程序模块、库模块、Shell 应用程序模块。本章最后讲述了如何在模拟器下调试应用程序。

第 4 章 介绍 Protocol 的概念和用法。UEFI 提供的绝大多数服务都是以 Protocol 的形式提供的，由此可见其重要性。

第 5 章 介绍 UEFI 中的基础服务，包括系统表，启动服务和运行时服务。系统表是应用程序进入内核的接口，启动服务用于在启动过程中管理计算机软硬件资源，运行时服务是为操作系统访问固件而提供的服务。

第 6 章 介绍了事件及异步操作方式。除了介绍事件的使用方法，本章还介绍了事件及异步操作在 UEFI 内核的实现机制。最后本章以具体实例介绍了如何使用键盘、鼠标和定时

器事件。

第 7 章 介绍了 GPT 硬盘以及文件系统和文件的操作方法。相比 BIOS，UEFI 的一个重大进步就是开始支持文件系统，FAT32 文件系统是 UEFI 内建文件系统。本章重点讲述 FAT32 文件系统之上的文件操作。

第 8 章 以视频解码服务为例介绍如何利用 Protocol 提供服务。服务型 Protocol 在 UEFI 中占有重要地位，服务型 Protocol 的开发是 UEFI 开发的一项基本功，一定要熟练掌握。

第 9 章 介绍驱动开发模型，并以 AC97 驱动为例介绍开发设备驱动的具体步骤。

第 10 章 介绍如何支持 C++ 语言特性。UEFI 内核是 EDK2 采用 C 语言实现的，UEFI 提供的服务也是以 C 语言形式提供的，因而 C 天然适用于 UEFI 开发，但 C++ 会使得开发更有效率。本章首先讲述了如何使用 C++ 基础语法开发 UEFI 应用，然后讲述如何支持全局构造函数、析构函数，支持 new、delete 等操作符，支持标准模板库等高级特性。

第 11 章 介绍开发 GUI 的基础知识，包括在 UEFI 中如何使用字符串资源、字体和图像。通过使用字符串资源，可以实现字符串的本地化。本地化后，字符串的显示需要字体的支持，EDK2 默认支持英文和法文字符的显示，要显示其他语言的字符串还需要开发者自行开发相应字符的字体。

第 12 章 以视频播放器为例介绍 GUI 开发。目前还没有成熟的 GUI 库可以用于 UEFI 开发。本章从零开始利用第 11 章介绍的基础知识开发 GUI 系统，主要分两大部分：GUI 事件的派遣和响应；GUI 控件的绘制。

第 13 章 介绍多任务的开发。本章分为两大块：多核和多线程。首先主要介绍 MPP(MP Services Protocol)，MPP 用于管理多核，本节重点讲述了 MPP 如何在其他 CPU 核心上执行任务。UEFI 没有提供对多线程的支持，对此我们讲述了如何使用 LongJmp 技术实现简单的多线程。

第 14 章 介绍如何开发网络应用。本章首先简单介绍了 UEFI 提供的网络协议栈，然后讲述了使用 TCP 协议开发网络应用的基本框架。

第 15 章 介绍如何使用 C 标准库中的函数。

第 16 章 介绍应用程序中的 Shell 服务、Shell 脚本的语法以及常用 Shell 命令。

资源和勘误

由于笔者水平与时间有限，书中可能出现一些错误或不准确的地方，恳请读者批评、斧正。如果您对本书有任何的意见和指正，请发邮件至 djx.zhenghua@gmail.com。本书附带的

源代码，可以从如下网址获取：https://uefi-programming-guider.googlecode.com/svn/trunk/，或者从华章网站（www.hzbook.com）下载。

致谢

感谢华章公司的高婧雅与杨福川付出的巨大努力，高编辑耐心、细致地反复审阅书稿，使得本书从粗疏渐渐变得精准。

感谢宋风龙在本书写作过程中给予的技术支持。

本书最早起始于我在博客园发表的几篇博客，感谢博客园的众多网友的支持和意见，你们的建议让我受益颇深。

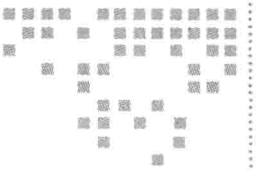
也要感谢工作在 CryptoMill 的同事。

特别鸣谢

最后要特别感谢我的妻子张一茁女士。从初稿到定稿近一年的时间里，我将大部分的节假日和周末用在了这本书上，这本书的完成，离不开妻子的付出和支持。

戴正华

Contents 目　录



第 1 章 Chapter 1

UEFI 概述

从我们按下开机键到进入操作系统，对用户来说是一个等待的过程，而对计算机来说是一个复杂的过程。在 BIOS 时代，这个过程重复了一年又一年，操作系统已经从枯燥的文本界面演化到丰富多彩的图形界面，BIOS 却一直延续着枯燥的过程，BIOS 设置也一直是单调的蓝底白字格式。BIOS 的坚持出于两个原因：外因是 BIOS 基本能满足市场需求，内因是 BIOS 的设计使得 BIOS 的升级和扩增变得非常困难。随着 64 位 CPU 逐渐取代 32 位 CPU，BIOS 越来越不能满足市场的需求，这使得 UEFI 作为 BIOS 的替代者，逐渐开始取代 BIOS 的地位。

1.1 BIOS 的前世今生

BIOS 诞生于 1975 年的 CP/M 计算机，诞生之初，也曾是一种先进的技术，并且是系统中相当重要的一个部分。随着 IBM PC 兼容机的流行，BIOS 也逐渐发展起来。它“统治”了计算机系统 20 多年的时间，在这段时间里，CPU 每 18 个月性能提升一倍。计算机软硬件都已经繁衍了无数代，BIOS 诞生之初与之配套的 8 位 CPU 和 DOS 系统都已经退出历史舞台，而 BIOS 依然顽强地存在于计算机中。

1.1.1 BIOS 在计算机系统中的作用

BIOS 全称为“基本输入 / 输出系统”，它是存储在主板 ROM 里的一组程序代码，这些代码包括：

- 加电自检程序，用于开机时对硬件的检测。
- 系统初始化代码，包括硬件设备的初始化、创建 BIOS 中断向量等。
- 基本的外围 I/O 处理的子程序代码。
- CMOS 设置程序。

BIOS 程序运行在 16 位实模式下，实模式下最大的寻址范围是 1MB，0x0C0000 ~ 0x0FFFFF 保留给 BIOS 使用。开机后，CPU 跳到 0x0FFFF0 处执行，一般这里是一条跳转指令，跳到真正的 BIOS 入口处执行。BIOS 代码首先做的是“加电自检”（Power On Self Test，POST），主要是检测关机设备是否正常工作，设备设置是否与 CMOS 中的设置一致。如果发现硬件错误，则通过喇叭报警。POST 检测通过后初始化显示设备并显示显卡信息，接着初始化其他设备。设备初始化完毕后开始检查 CPU 和内存并显示检测结果。内存检测通过以后开始检测标准设备，例如硬盘、光驱、串口设备、并口设备等。然后检测即插即用设备，并为这些设备分配中断号、I/O 端口和 DMA 通道等资源。如果硬件配置发生变化，那么这些变化的配置将更新到 CMOS 中。随后，根据配置的启动顺序从设备启动，将启动设备主引导记录的启动代码通过 BIOS 中断读入内存，然后控制权交到引导程序手中，最终引导进入操作系统。

1.1.2 BIOS 缺点

随着 CPU 及其他硬件设备的革新，BIOS 逐渐成为计算机系统发展的瓶颈，主要体现在如下几个方面：

1）**开发效率低**：大部分 BIOS 代码使用汇编开发，开发效率不言而喻。汇编开发的另一个缺点是使得代码与设备的耦合程度太高，代码受硬件变化的影响大。

2）**性能差**：BIOS 基本输入 / 输出服务需要通过中断来完成，开销大，并且 BIOS 没有提供异步工作模式，大量的时间消耗在等待上。

3）**功能扩展性差，升级缓慢**：BIOS 代码采用静态链接，增加硬件功能时，必须将 16 位代码放置在 0x0C0000 ~ 0x0DFFFF 区间，初始化时将其设置为约定的中断处理程序。而且 BIOS 没有提供动态加载设备驱动的方案。

4）**安全性**：BIOS 运行过程中对可执行代码没有安全方面的考虑。

5）**不支持从硬盘 2 TB 以上的地址引导**：受限于 BIOS 硬盘的寻址方式，BIOS 硬盘采用 32 位地址，因而引导扇区的最大逻辑块地址是 2^{32}（换算成字节地址，即 $2^{32} \times 512$=2TB）。

1.2 初识 UEFI

UEFI（Unified Extensible Firmware Interface，统一可扩展固件接口）定义了操作系统和平台固件之间的接口，它是 UEFI Forum 发布的一种标准。它只是一种标准，没有提供实现。

其实现由其他公司或开源组织提供，例如英特尔公司提供的开源 UEFI 实现 TianoCore 和 Phoenix 公司的 SecureCore Tiano。UEFI 实现一般可分为两部分：

- 平台初始化（遵循 Platform Initialization 标准，同样由 UEFI Forum 发布）。
- 固件 – 操作系统接口。

UEFI 发端于 20 世纪 90 年代中期的安腾系统。相对于当时流行的 32 位 IA32 系统，安腾是一种全新的 64 位系统，BIOS 的限制对这种 64 位系统变得不可接受（BIOS 也正是随着 32 位系统被 64 位系统取代而逐渐退出市场的）。因为 BIOS 在 64 位系统上的限制，1998 年英特尔公司发起了 Intel Boot Initiative 项目，后来更名为 EFI（Extensible Firmware Interface）。2003 年英特尔公司的安腾 CPU 计划遭到 AMD 公司的 x86_64 CPU 顽强阻击，x86_64 CPU 时代到来，市场更愿意接受渐进式的变化，英特尔公司也开始发布兼容 32 位系统的 x86_64 CPU。安腾虽然没有像预期那样独占市场，EFI 却显示出了它的价值。2005 年，英特尔公司联合微软、AMD、联想等 11 家公司成立了 Unified EFI Forum，负责制定统一的 EFI 标准。第一个 UEFI 标准——UEFI 2.0 在 2006 年 1 月发布。目前最新的 UEFI 标准是 2013 年发布的 UEFI 2.4。

1.2.1　UEFI 系统组成

UEFI 提供给操作系统的接口包括启动服务（Boot Services，BS）和运行时服务（Runtime Service，RT）以及隐藏在 BS 之后的丰富的 Protocol。BS 和 RT 以表的形式（C 语言中的结构体）存在。UEFI 驱动和服务以 Protocol 的形式通过 BS 提供给操作系统。

图 1-1 展示了基于 EFI 的计算机系统的组成。

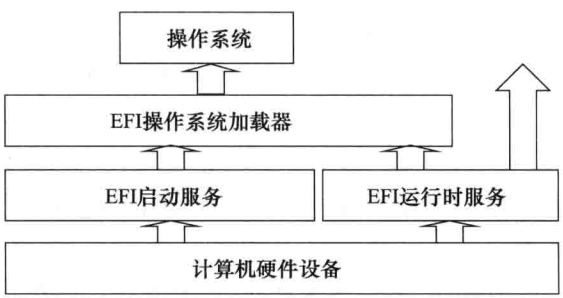


图 1-1　EFI 系统组成

从操作系统加载器（OS Loader）被加载，到 OS Loader 执行 ExitBootServices() 的这段时间，是从 UEFI 环境向操作系统过渡的过程。在这个过程中，OS Loader 可以通过 BS 和 RT 使用 UEFI 提供的服务，将计算机系统资源逐渐转移到自己手中，这个过程称为 TSL（Transient System Load）。

当 OS Loader 完全掌握了计算机系统资源时，BS 也就完成了它的使命。OS Loader 调用 ExitBootServices() 结束 BS 并回收 BS 占用的资源，之后计算机系统进入 UEFI Runtime 阶段。