HZ BOOKS
华章科技

畅销书全新升级和大幅优化，第1版广获好评，被翻译为繁体中文和英文（美国）

以真实操作系统的实际运行为主线，以图形图像为核心，突出描述操作系统在实际运行过程中内存的运行时结构；从操作系统设计者的视角，用体系的思想方法，深刻解读操作系统的架构设计与实现原理

华章原创精品

第2版

Linux内核设计的艺术

图解Linux操作系统架构设计与实现原理

The Art of Linux Kernel Design, Second Edition

新设计团队 著

机械工业出版社
China Machine Press

Linux内核设计的艺术

图解Linux操作系统架构设计与实现原理（第2版）

The Art of Linux Kernel Design, Second Edition

新设计团队 著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Linux 内核设计的艺术：图解 Linux 操作系统架构设计与实现原理 / 新设计团队著 . —2 版 . —北京：机械工业出版社，2013.5

ISBN 978-7-111-42176-4

Ⅰ . L… Ⅱ . 新… Ⅲ . Linux 操作系统 – 图解 Ⅳ . TP316-64

中国版本图书馆 CIP 数据核字（2013）第 076059 号

本书的第 1 版广获好评，版权被中国台湾和美国两家大型出版社引进，第 2 版根据读者的反馈和作者对操作系统的最新研究成果对第 1 版进行了大幅优化和重写，使其内容质量更上一层楼。本书在众多关于 Linux 内核的书中独树一帜，它在世界范围内首次提出并阐述了操作系统设计的核心指导思想——主奴机制，这是所有操作系统研究者的一笔宝贵财富。它也是一本能真正引导我们较为容易地、极为透彻地理解 Linux 内核的经典之作，也可能是当前唯一能从本质上指引我们去设计和开发拥有自主知识产权的操作系统的著作。

本书的最大特点是它的写作方式和内容组织方式与同类书完全不同。它在深刻地分析了传统讲解方法的利弊之后，破旧立新，从认知学的角度开创了一种全新的方式。以操作系统的真实运行过程为主线，结合真实的内核源代码、300 余幅精确的内核运行时序图和具有点睛之妙的文字说明，对操作系统从开机加电到系统完全准备就绪，及运行用户程序的整个过程进行了系统而完整地分析，深刻地揭示了其间每一个动作的设计意图和实现原理，完美地再现了操作系统设计者的设计思路。阅读本书就如同跟随着操作系统设计者一起去思考，我们会在阅读的过程中发现 Linux 内核设计的精妙，会发现原来处处都“暗藏玄机”，哪怕是一行很短的代码。

本书在所有细节上都力求完美。为了保证知识的准确性，操作系统运行过程中的每个动作都经过了严格的考证；为了让我们真正理解 Linux 内核的原理，它突破传统，以 Linux 的真实运行过程为主线进行讲解；为了做到真正易于理解，创新性地使用了图解的方式，精心绘制了 300 余幅分辨率 600dpi 的时序图，图中表现的运行时结构和状态与操作系统实际运行时的真实状态完全吻合；为了提高阅读体验，本书采用了双色印刷，以便于我们更清楚地观察每一幅图中的细节。

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：朱秀英
北京京师印务有限公司印刷
2013 年 5 月第 2 版第 1 次印刷
186mm × 240mm · 29.25 印张
标准书号：ISBN 978-7-111-42176-4
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

前　言

为什么写这本书

很早就有一个想法，做中国人自己的、有所突破、有所创新的操作系统、计算机语言及编译平台。

我带领的“新设计团队”（主要由中国科学院研究生院毕业的学生组成）在实际开发自己的操作系统的过程中，最先遇到的问题就是如何培养学生真正看懂 Linux 操作系统的源代码的能力。开源的 Linux 操作系统的源代码很容易找到，但很快就会发现，培养学生看懂 Linux 操作系统的源代码是一件非常困难的事。

操作系统的代码量通常都是非常庞大的，动辄几百万行，即使浏览一遍也要很长时间。比庞大的代码量更让学习者绝望的是操作系统有着极其错综复杂的关系。看上去，代码的执行序时隐时现，很难抓住脉络。代码之间相互牵扯，相互勾连，几乎无法理出头绪，更谈不上理解代码背后的原理、意图和思想。

对于学生而言，选择从源代码的什么地方开始分析，本身就是一个难题。通常，学生有两种选择：一种是从 main 函数，也就是从 C 语言代码的总入口开始，沿着源代码的调用路线一行一行地看下去，学生很快就会发现源代码的调用路线莫名其妙地断了，但直觉和常识告诉他操作系统肯定不会在这个地方停止，一定还在继续运行，却不知道后续的代码在哪里，这种方法很快就走进了死胡同；另一种则是从某一模块入手，如文件系统，但这样会无形中切断操作系统源码之间复杂的关系，如文件系统与进程管理的关系，文件系统与内存管理的关系，等等。学生如果孤立地去理解一个模块，往往只能记住一些名词和简单概念，难以真正理解操作系统的全貌。用学生的话讲，他们理解的操作系统变成了“文科”的操作系统。

由于操作系统是底层系统程序，对应用程序行之有效的调试和跟踪等手段对操作系统的源代码而言，几乎无效。学生就算把每一行源代码都看懂了，对源代码已经烂熟于心，知道这一行是一个 for 循环，那一行是一个调用……但仍然不知道整个代码究竟在做什么以及起到什么作用，更不知道设计者的意图究竟是什么。

学生在操作系统课程上学习过进程管理、内存管理、文件系统等基础知识，但是对这些空洞的理论在一个实际的操作系统中是如何实现的却不得而知。他们在源代码中很难看出进程和内存之间有什么关联，内核程序和用户程序有什么区别，为什么要有这些区别；也很难从源代码中看清楚，我们实际经常用到的操作，比如打开文件，操作系统在其中都做了哪些具体的工

作。想在与常见的应用程序的编程方法有巨大差异的、晦涩难懂的、浩瀚如海的操作系统底层源代码中找到这些问题的答案，似乎比登天还难。

对熟悉操作系统源代码的学生而言，他们也知道像分页机制这样的知识点，知道若干级的分页及恒等映射，但是未必能够真正理解隐藏在机制背后的深刻意义。

这些都是学生在学习 Linux 操作系统源代码时遇到的实际问题。中国科学院研究生院的学生应该是年轻人中的佼佼者，他们遇到的问题可能其他读者也会遇到。我萌发了一个想法，虽然学生的问题早已解决，但是否可以把他们曾经在学习、研发操作系统的过程中遇到的问题和心得体会拿出来供广大读者分享。

当时，针对学生的实际问题，我的解决方法是以一个真实的操作系统为例，让学生理解源代码并把操作系统在内存中的运行时状态画出图来。实践证明，这个方法简单有效。

现在我们把这个解决方案体现在这本书中。这就是以一个真实的操作系统的实际运行为主线；以图形、图像为核心，突出描述操作系统在实际运行过程中内存的运行时结构；强调学生站在操作系统设计者的视角，用体系的思想方法，整体把握操作系统的行为、作用、目的和意义。

第 1 版与第 2 版的区别

第 2 版较第 1 版有较大的改动。

从总体结构上，将第 1 版的第 2 章拆分为第 2 版的第 2 章、第 3 章、第 4 章。这样的拆分对操作系统启动部分的系统初始化、激活进程 0、创建进程 1、创建进程 2 的层次划分更清晰。各章内容的分量也比较均衡，阅读感受会更好。

根据读者的反馈意见，第 2 版增加了一些示意图，在源代码中增加了大量的注释，对操作系统的架构表述得更直观，对源代码讲解得更细致。这些是第 2 版改动最大、下功夫最多的地方。希望我们的努力能给读者带来更多的帮助。

本书内容及特色

在全书的讲解过程中，我们不仅详细分析了源代码、分析了操作系统的执行序，还特别分析了操作系统都做了哪些“事”，并且对于“事”与“事”之间的关系和来龙去脉，这些“事”意味着什么，为什么要做这些“事”，这些“事”背后的设计思想是什么……都做了非常详细且深入的分析。

更重要的是，对于所有重要的阶段，我们几乎都用图解的方式把操作系统在内存中的实际运行状态精确地表示了出来。我们用 600 dpi 的分辨率精心绘制了 300 多张图，图中表现的运行时结构和状态与操作系统实际运行的真实状态完全吻合。对每一条线、每一个色块、每一个位置、每一个地址及每一个数字，我们都经过了认真反复地推演和求证，并最终在计算机上进行了核对和验证。看了这些绘制精美的图后，读者的头脑中就不再是一行行、一段段枯燥的、令人眩晕的源代码，而是立体呈现的一件件清晰的“事”，以及这些“事”在内存中直截了当、清晰鲜活的画面。用这样的方法讲解操作系统是本书的一大特色。理解这些图要比理解源代码和文字容易得

多。毫不夸张地说，只要你能理解这些图，你就理解了操作系统的80%。这时你可以自豪地说，你比大多数用别的方法学过操作系统的人的水平都要高出一大截。

作者和机械工业出版社的编辑做了大量的检索工作。就我们检索的范围而言，这样的创作方法及具有这样特色的操作系统专著在世界范围都是第一次。

本书分三部分来讲解Linux操作系统：第一部分（第1～4章）分析了从开机加电到操作系统启动完成并进入怠速状态的整个过程；第二部分（第5～8章）讲述了操作系统进入系统怠速后，在执行用户程序的过程中，操作系统和用户进程的实际运行过程和状态；第三部分（第9章）阐述整个Linux操作系统的设计指导思想，是从微观到宏观的回归。

第一部分中，我们详细讲解了开机加电启动BIOS，通过BIOS加载操作系统程序，对主机的初始化，打开保护模式和分页，调用main函数，创建进程0、进程1、进程2以及shell进程，并且具备用文件的形式与外设交互。

第二部分中，我们设计了几个尽可能简单又有代表性的应用程序，并以这些程序的执行为引导，详细讲解了安装文件系统、文件操作、用户进程与内存管理、多个进程对文件的操作以及进程间通信。

我们将操作系统的原理自然而然地融入了讲解真实操作系统的实际运行过程中。在读者看来，操作系统原理不再是空对空的、“文科”概念的计算机理论，而是既有完整且体系的理论，又有真实、具体、实际的代码和案例，理论与实际紧密结合。

第三部分是全书水平最高的部分，详细阐述了主奴机制以及实现主奴机制的三项关键技术：保护和分页、特权级、中断，分析了保障主奴机制实现的决定性因素——先机，还详细讲解了缓冲区、共享页面、信号、管道的设计指导思想。我们尝试从操作系统设计者的视角讲解操作系统的设计指导思想。希望帮助读者用体系的思想理解、把握、驾驭整个操作系统以及背后的设计思想和设计意图。

在本书中，我们详细讲解了大家在学习操作系统的过程中可能会遇到的每一个难点，如main函数中的pause()调用，虽然已经找不到后续代码，但该调用结束后，程序仍然执行的原因是：中断已经打开，进程调度就开始了，而此时可以调度的进程只有进程1，所以后续的代码应该从进程1处继续执行……

我们还对读者不容易理解和掌握的操作系统特有的底层代码的一些编程技巧做了详细的讲解，如用模拟call的方法，通过ret指令“调用”main函数……

总之，我们所做的一切努力就是想真正解决读者遇到的实际问题和难题，给予读者有效的帮助。我们盼望即使是刚刚考入大学的学生也有兴趣和信心把这本书读下去；我们同样希望即使是对操作系统源代码很熟悉的读者，这本书也能给他们一些不同的视角、方法和体系性思考。

为什么本书选用Linux 0.11内核

这本书选用的是Linux 0.11操作系统源代码。对为什么选用Linux 0.11而不是最新版本，赵炯先生有过非常精彩的论述。我们认为赵先生的论述是非常到位的。

我们不妨看一下Linux最新的版本2.6，代码量大约在千万行这个量级，去掉其中的驱

动部分，代码量仍在百万行这个量级。一个人一秒钟看一行，一天看 8 小时，中间不吃、不喝、不休息，也要看上几个月，很难想象如何去理解。

就算我们坚持要选用 Linux 2.6，就算我们写上 2000 页（书足足会有十几厘米厚），所有的篇幅都用来印代码，也只能印上不到十分之一的代码。所以，即使是这么不切实际的篇幅，也不可能整体讲解 Linux 2.6。读者会逐渐明白，对于理解和掌握操作系统而言，真正有价值的是整体、是体系，而不是局部。

Linux 0.11 的内核代码虽然只有约两万行，但却是一个实实在在、不折不扣的现代操作系统。因为它具有现代操作系统最重要的特征——支持实时多任务，所以必然支持保护和分页……而且它还是后续版本的真正的始祖，有着内在的、紧密的传承关系。读者更容易看清设计者最初的、最根本的设计意图和设计指导思想。

Linux 0.11 已经问世 20 多年了，被世人广为研究和学习。换一个角度看，要想对众人熟悉的事物和领域讲出新意和特色，对作者来说也是一个强有力的挑战。

致谢

首先，感谢机械工业出版社华章公司的副总经理温莉芳女士以及其他领导，是他们的决心和决策成就了这本书，并且在几乎所有方面给予了强有力的支持。特别令人感动的是他们主动承担了全部的出版风险，同时给予了作者最好的条件，让我们看到一个大出版社的气度和风范。

其次，特别感谢机械工业出版社华章公司的编辑杨福川。杨先生的鉴赏力和他的事业心以及他对工作认真负责的态度为这本书的出版打开了大门。杨先生对读者的理解以及他的计算机专业素养使得他有能力对这本书给予全方位的指导和帮助，使我们对这本书整体修改了多次，使之更贴近读者，可读性更好。

还要感谢我们和杨福川共同的朋友张国强先生和杨缙女士。

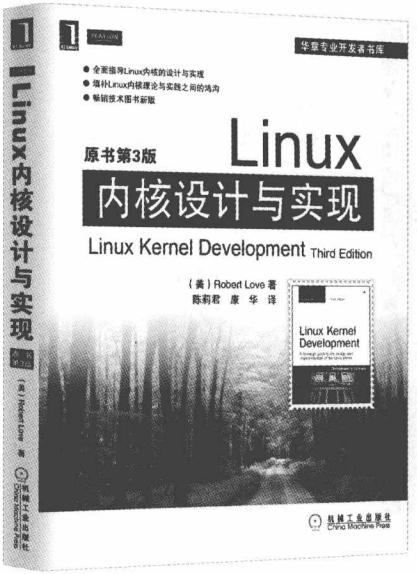
最后，感谢我们的家人和朋友，是他们坚定的支持才使得整个团队能够拒绝方方面面、形形色色的诱惑，放弃普遍追求的短期利益；我们在常人难以想象的艰苦条件下，长时间专注于操作系统、计算机语言、编译器、计算机体系结构等基础性学科的研究。因为我们认认真真、踏踏实实、不为名利，只为做一点实在、深入的工作，积累了十年的经验，打造了一支敢想、敢干、敢打、敢拼、不惧世界顶级强敌的队伍。这些是本书的基础。

杨力祥

中国科学院研究生院

2013 年 1 月

推荐阅读



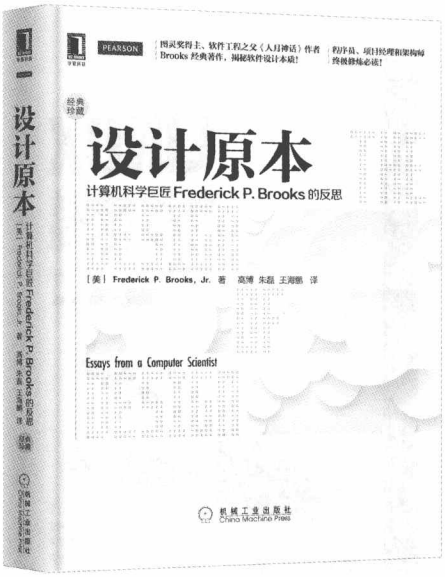
Linux内核设计与实现（原书第3版）

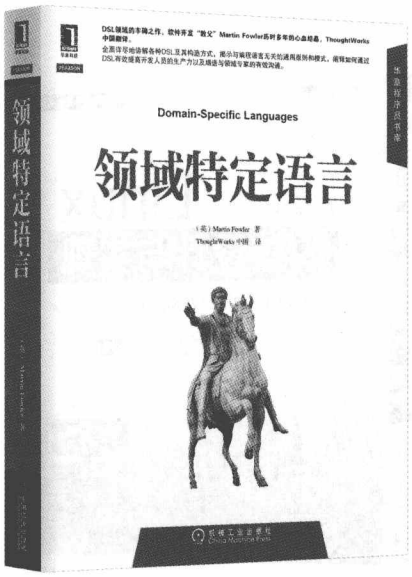
世界范围内公认的Linux内核经典著作，畅销全球多个国家

Linux内核精髓

畅销书，一线内核技术专家经验和智慧结晶，深刻解读Linux内核的资源管理、文件系统、网络、虚拟化、省电技术、调试、性能调优、分析与追踪等核心主题

推荐阅读





设计原本（精装本）

如果说《人月神话》是近40年来所有软件开发工程师和项目经理们必读的一本书，那么本书将会是未来数十年内从事软件行业的程序员、项目经理和架构师必读的一本书。它是《人月神话》作者、著名计算机科学家、软件工程教父、美国两院院士、图灵奖和IEEE计算机先驱奖得主Brooks在计算机软硬件架构与设计、建筑和组织机构的架构与设计等领域毕生经验的结晶，是计算机图书领域的又一史诗级著作。

领域特定语言

本书是DSL领域的丰碑之作，由世界级软件开发大师和软件开发“教父”Martin Fowler历时多年写作而成。全面详尽地讲解了各种DSL及其构造方式，揭示了与编程语言无关的通用原则和模式，阐释了如何通过DSL有效提高开发人员的生产力以及增进与领域专家的有效沟通，能为开发人员选择和使用DSL提供有效的决策依据和指导方法。

目 录

第 1 章

从开机加电到执行 main 函数之前的过程

从开机到 main 函数的执行分三步完成，目的是实现从启动盘加载操作系统程序，完成执行 main 函数所需要的准备工作。第一步，启动 BIOS，准备实模式下的中断向量表和中断服务程序；第二步，从启动盘加载操作系统程序到内存，加载操作系统程序的工作就是利用第一步中准备的中断服务程序实现的；第三步，为执行 32 位的 main 函数做过渡工作。本章将详细分析这三步在计算机中是如何完成的，以及每一步在内存中都做了些什么。

小贴士

实模式（Real Mode）是 Intel 80286 和之后的 80x86 兼容 CPU 的操作模式（应该包括 8086）。实模式的特性是一个 20 位的存储器地址空间（2^20 = 1 048 576，即 1 MB 的存储器可被寻址），可以直接软件访问 BIOS 以及周边硬件，没有硬件支持的分页机制和实时多任务概念。从 80286 开始，所有的 80x86 CPU 的开机状态都是实模式；8086 等早期的 CPU 只有一种操作模式，类似于实模式。

1.1 启动 BIOS，准备实模式下的中断向量表和中断服务程序

相信大家都知道一台计算机必须要安装一个所谓“操作系统”的软件，才能让我们使用计算机，否则计算机将是一堆毫无生命力的冰冷的硬家伙。在为计算机安装了操作系统后，当你按下计算机电源按钮的那一刻，计算机机箱传来了嗡嗡的声音。这时你感觉到，计算机开始启动工作了。然而，在计算机的启动过程中，操作系统底层与计算机硬件之间究竟做了哪些复杂的交互动作？下面我们将根据操作系统实际的启动和运行过程对此进行逐步的剖析和讲解。

计算机的运行是离不开程序的。然而，加电的一瞬间，计算机的内存中，准确地说是 RAM 中，空空如也，什么程序也没有。软盘里虽然有操作系统程序，但 CPU 的逻辑电路被设计为只能运行内存中的程序，没有能力直接从软盘运行操作系统[㊀]。如果要运行软盘中的

㊀ Linus写Linux 0.11是在1991年年底。那时，很多计算机是从软盘启动的。他为Linux 0.11设计的系统启动盘是软盘。

操作系统，必须将软盘中的操作系统程序加载到内存（RAM）中。

特别注意

我们假定本书所用的计算机是基于 IA—32 系列 CPU，安装了标准单色显示器、标准键盘、一个软驱、一块硬盘、16 MB 内存，在内存中开辟了 2 MB 内存作为虚拟盘，并在 BIOS 中设置软驱为启动设备。后续所有的讲解都以此为基础。

小贴士

RAM（Random Access Memory）：随机存取存储器，常见的内存条就是一类 RAM，其特点是加电状态下可任意读、写，断电后信息消失。

问题：在 RAM 中什么程序也没有的时候，谁来完成加载软盘中操作系统的任务呢？

答案是：BIOS。

1.1.1 BIOS 的启动原理

在了解 BIOS 是如何将操作系统程序加载到内存中之前，我们先来了解一下 BIOS 程序自身是如何启动的。从我们使用计算机的经验得知：要想执行一个程序，必须在窗口中双击它，或者在命令行界面中输入相应的执行命令。从计算机底层机制上讲，其实是在一个已经运行起来的操作系统的可视化界面或命令行界面中执行一个程序。但是，在开机加电的一瞬间，内存中什么程序也没有，没有任何程序在运行，不可能有操作系统，更不可能有操作系统的用户界面。我们无法人为地执行 BIOS 程序，那么 BIOS 程序又是由谁来执行的呢？

秘诀是：0xFFFF0 ！ ！ ！

从体系的角度看，不难得出这样的结论：既然用软件方法不可能执行 BIOS，就只能靠硬件方法完成了。

从硬件角度看，Intel 80x86 系列的 CPU 可以分别在 16 位实模式和 32 位保护模式下运行。为了兼容，也为了解决最开始的启动问题，Intel 将所有 80x86 系列的 CPU，包括最新型号的 CPU 的硬件都设计为加电即进入 16 位实模式状态运行。同时，还有一点非常关键的是，将 CPU 硬件逻辑设计为加电瞬间强行将 CS 的值置为 0xF000、IP 的值置为 0xFFF0，这样 CS:IP 就指向 0xFFFF0 这个地址位置，如图[⊖] 1-1 所示。从图 1-1 中可以清楚地看到，0xFFFF0 指向了 BIOS 的地址范围。

⊖ 本书中的大部分图都是依照计算机实际运行时的内存真实状态，严格按比例以600 dpi分辨率精确绘制的，所以有些内存区域因为体积比较小，在图中占的位置也比较小。请大家阅读时仔细辨认。

小贴士

IP/EIP（Instruction Pointer）：指令指针寄存器，存在于 CPU 中，记录将要执行的指令在代码段内的偏移地址，和 CS 组合即为将要执行的指令的内存地址。实模式为绝对地址，指令指针为 16 位，即 IP；保护模式下为线性地址，指令指针为 32 位，即 EIP。

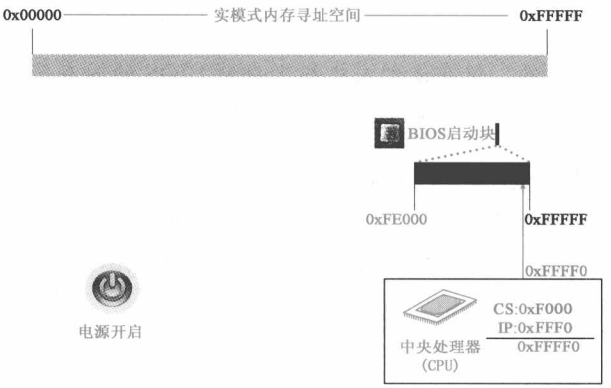


图 1-1　启动时 BIOS 在内存的状态及初始执行位置

小贴士

CS（Code Segment Register）：代码段寄存器，存在于 CPU 中，指向 CPU 当前执行代码在内存中的区域（定义了存放代码的存储器的起始地址）。

注意，这是一个纯硬件完成的动作！如果此时这个位置没有可执行代码，那么就什么也不用说了，计算机就此死机。反之，如果这个位置有可执行代码，计算机将从这里的代码开始，沿着后续程序一直执行下去。

BIOS 程序的入口地址恰恰就是 0xFFFF0 ！ 也就是说，BIOS 程序的第一条指令就设计在这个位置。

1.1.2　BIOS 在内存中加载中断向量表和中断服务程序

BIOS 程序的代码量并不大，却非常精深，需要对整个计算机硬件体系结构非常熟悉才能看得明白。要想把 BIOS 是如何运行的讲清楚，也得写很厚的一本书，这显然超出了本书的主题和范围。我们的主题是操作系统，所以只把与启动操作系统有直接关系的部分简单地讲解一下。

BIOS 程序被固化在计算机主机板上的一块很小的 ROM 芯片里。通常不同的主机板所用的 BIOS 也有所不同。就启动部分而言，各种类型的 BIOS 的基本原理大致相似。为了便于大家理解，我们选用的 BIOS 程序只有 8 KB，所占地址段为 0xFE000 ～ 0xFFFFF，如图 1-1

所示。现在 CS:IP 已经指向 0xFFFF0 这个位置了，这意味着 BIOS 开始启动了。随着 BIOS 程序的执行，屏幕上会显示显卡的信息、内存的信息……说明 BIOS 程序在检测显卡、内存……这期间，有一项对启动（boot）操作系统至关重要的工作，那就是 BIOS 在内存中建立中断向量表和中断服务程序。

小贴士

ROM（Read Only Memory）：只读存储器。现在通常用闪存芯片做 ROM。虽然闪存芯片在特定的条件下是可写的，但在谈到主机板上存储 BIOS 的闪存芯片时，业内人士把它看做 ROM。ROM 有一个特性，就是断电之后仍能保存信息，这一点和硬盘类似。

BIOS 程序在内存最开始的位置（0x00000）用 1 KB 的内存空间（0x00000 ~ 0x003FF）构建中断向量表，在紧挨着它的位置用 256 字节的内存空间构建 BIOS 数据区（0x00400 ~ 0x004FF），并在大约 57 KB 以后的位置（0x0E05B）加载了 8 KB 左右的与中断向量表相应的若干中断服务程序。图 1-2 中精确地标注了这些位置。

小贴士

一个容易计算的方法：0x00100 是 256 字节，0x00400 就是 4×256 字节 =1024 字节，也就是 1 KB。因为是从 0x00000 开始计算，所以 1 KB 的高地址端不是 0x00400，而是 0x00400−1，也就是 0x003FF。

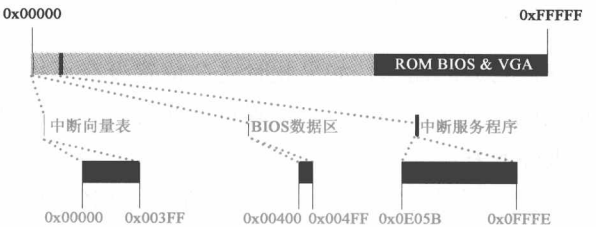


图 1-2　BIOS 在内存中加载中断向量表和中断服务程序

中断向量表中有 256 个中断向量，每个中断向量占 4 字节，其中两个字节是 CS 的值，两个字节是 IP 的值。每个中断向量都指向一个具体的中断服务程序。

下面将详细讲解后续程序是如何利用这些中断服务程序把系统内核程序从软盘加载至内存的。

小贴士

INT（INTerrupt）：中断，顾名思义，中途打断一件正在进行中的事。其最初的意思是：外在的事件打断正在执行的程序，转而执行处理这个事件的特定程序，处理结束后，回到被