Broadview
www.broadview.com.cn

畅销书升级版

Orange'S
一个操作系统的实现

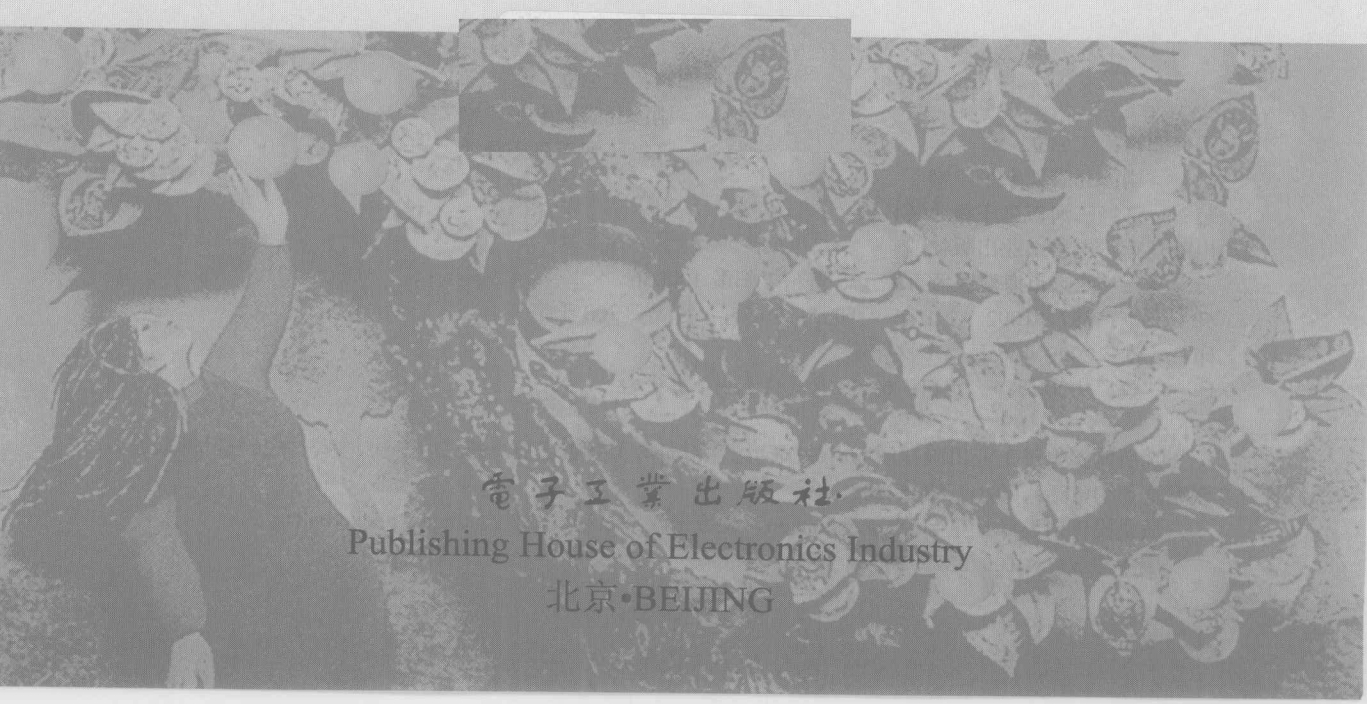
于渊 著

《自己动手写操作系统》第2版

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Orange'S
一个操作系统的实现

于渊 著



電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书从只有二十行的引导扇区代码出发，一步一步地向读者呈现一个操作系统框架的完成过程。书中不仅关注代码本身，同时关注完成这些代码的思路和过程。本书不同于其他的理论型书籍，而是提供给读者一个动手实践的路线图。读者可以根据路线图逐步完成各部分的功能，从而避免了一开始就面对整个操作系统数万行代码时的迷茫和挫败感。书中讲解了大量在开发操作系统中需注意的细节问题，这些细节不仅能使读者更深刻地认识操作系统的核心原理，而且使整个开发过程少走弯路。本书分上下两篇，共 11 章。其中每一章都以前一章的工作成果为基础，实现一项新的功能。而在章的内部，一项大的功能被分解成许多小的步骤，通过完成每个小的步骤，读者可以不断获得阶段性的成果，从而让整个开发过程变得轻松并且有趣。

本书适合各类程序员、程序开发爱好者阅读，也可作为高等院校操作系统课程的实践参考书。

图书在版编目（CIP）数据

Orange' S：一个操作系统的实现 / 于渊著. —北京：电子工业出版社，2009.6
ISBN 978-7-121-08442-3

I. O… Ⅱ. 于… Ⅲ. 操作系统 Ⅳ. TP316

中国版本图书馆 CIP 数据核字（2009）第 030117 号

责任编辑：江 立
印 刷：北京天宇星印刷厂
装 订：三河市皇庄路通装订厂
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编 100036
开 本：787×1092 1/16 印张：30.75 字数：700 千字
印 次：2009 年 6 月第 1 次印刷
印 数：5000 册 定价：69.00 元（含光盘 1 张）

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

做真正 Hacker 的乐趣——自己动手去实践

2004 年我听编辑说有个年轻人写了本《自己动手写操作系统》，第一反应是不可能，恐怕是翻译稿，写这种书籍是要考作者硬功夫的，不但需要深入掌握操作系统的原理，还需要实际动手写出原型。

历史上的 Linux 就是这么产生的，Linus Torvalds 当时是一名赫尔辛基大学计算机科学系的二年级学生，经常要用自己的电脑去访问大学主机上的新闻组和邮件，为了方便读写和下载文件，他自己编写了磁盘驱动程序和文件系统，这成为了 Linux 第一个内核的雏形。

我想中国有能力写出内核原型的程序员应该也有，但把这个题目写成一本书，感觉上不会有人愿意做这件事情，作者要花很多时间，加上主题比较硬，销售量不会太高，经济上回报有限。

但拿来文稿一看，整个编辑部大为惊艳，内容文笔俱佳，而且绝对原创，马上决定在《程序员》连载。2005 年博文视点出版的第一版也广受好评。

不过有很多读者还是质疑：现在软件编程主要领域是框架和应用，还需要了解操作系统底层吗？

经过四年的磨练成长，于渊又拿出第二版的书稿《Orange'S：一个操作系统的实现》，这本书是属于真正 Hacker 的。我虽然已经有多年不写代码了，但看这本书的时候，让我又重新感受到做程序员的乐趣：用代码建设属于自己的系统，让电脑听从自己的指令，对系统的每个部分都了如指掌。

黑客（hacker）实际是褒义词，维基百科的解释是喜欢用智力通过创造性方法来挑战脑力极限的人，特别是他们所感兴趣的领域，例如软件编程或电气工程。个人电脑、软件和互联网等划时代的产品都是黑客创造出来的，如苹果的 Apple 电脑、微软的 Basic 解释器、互联网的 Mosaic 浏览器。

回答前面读者的质疑，学软件编程并不需要看这本书，想成为优秀程序员和黑客的朋友，我强烈建议你花时间来阅读这本书，并亲自动手实践。正如于渊在本书结尾中所说“我们写自己的操作系统是出于一种好奇，或者说一种求知欲。我希望这样不停地‘过把瘾’能让这种好奇不停地延续”。

好奇心是动力的源泉，追究问题的本质是优秀黑客的必备素质，只有充分掌握了系统原理，才能在技术上游刃有余，才能有真正的创新和发展。中国需要更多真正的黑客，也希望更多的程序员能享受属于黑客的创造乐趣。

蒋涛

2009 年 4 月

作者自序

本书是《自己动手写操作系统》的第二版，通过一个具体的实例向读者呈现一个操作系统雏形的实现过程。有关操作系统的书籍资料可以找到很多，但是关注如何帮助读者实现一个试验性操作系统的书籍却不多见，本书便是从一个简单的引导扇区开始，讲述一个操作系统成长的故事，以作读者参考之用。

本书面向实践，通过具体实例教读者开发自己的操作系统。书中的步骤遵循由小到大、由浅入深的顺序，跟随这些步骤，读者可以由一个最简单的引导扇区开始，逐渐完善代码，扩充功能，最后形成一个小的操作系统。

本书不仅介绍操作系统的各要素，同时涉及开发操作系统需要的各个方面，比如如何建立开发环境、如何调试以及如何在虚拟机中运行等。书中的实例操作系统采用 IA32 作为默认平台，所以保护模式也作为必备知识储备收入书中，而这是传统的操作系统实践书籍经常忽略的。总之，只要是开发自己的操作系统中需要的知识，书中都尽量涉及，以便于读者参考。

众所周知，一个成型的操作系统往往非常复杂。如果考虑到操作系统作为软硬件桥梁的特殊地位，那么它可能看上去比一般的软件系统更难理解，因为其核心部分往往包含许多直接针对 CPU、内存和 I/O 端口的操作，它们夹杂在一片代码汪洋之中，显得更加晦涩。

我们有许多源代码公开的操作系统，可供随时下载和阅读，看上去好像让实现一个供自己把玩的微型操作系统变得容易很多，但事实往往不尽人意，因为这些代码动辄上万甚至几十几百万行，而且细节之间经常互相关联，要理解它们着实不易。我们有许多容易得到的操作系统教程，但读来好像总觉得跟我们有隔膜，不亲近。造成这些的根本原因，在于学习者一开始就面对一个完整的操作系统，或者面对前辈们积累了几十年的一系列理论成果。而无论作者多么擅长写作，读者多么聪明，或者代码多么优秀，要一个初学者理清其中的头绪都将是非常困难的。

我并非在此危言耸听，因为这曾经是我的亲身体会。当然，如果只是为了考试，几本操作系统理论书籍就足够了，你不需要对细节那么清楚。但如果是出于兴趣呢？如果你是想编写自己的操作系统呢？你会发现理论书籍好像一下子变得无用武之地，你会发现任何一个细节上的理解错误都可能导致自己辛辛苦苦编写的代码运行异常甚至崩溃。

我经历过这一切！我曾经翻遍了一本《操作系统：设计与实现》，也没有找到实现一个操作系统应该从何处着手。并不是这些书不好，也不是前人的代码不优秀，而是作为一无所知的初学者，我们所不了解的不仅是高居庙堂的理论

知识，还有让我们举步维艰的实践细节。

可能在这些教科书作者的眼里，操作的细节不属于课程的一部分，或者这些细节看上去太容易，根本不值一提，甚至作者认为这些属于所谓“经验”的一部分，约定俗成是由读者本人去摸索的。但是实际情况往往是，这些书中忽略掉的内容恰恰占去了一个初学者大部分的时间，甚至影响了学习的热情。

我至今仍记得当我开始编写自己的操作系统时所遭受的挫败感，那是一种不知道如何着手的无助的感觉。还好我坚持了下来，克服了各种困难，并完成了自己的操作系统雏形。

进而我想到，一定不只是我一个人对编写自己的操作系统怀有兴趣，也一定不只是我一个人在实践时遇到困难。或许我应该把自己的经历写下来，从而可以帮助跟我相似的后来者，就这样，我编写了本书的第一版，也就是《自己动手写操作系统》。我相信，如果你也对神奇的计算机世界充满好奇，并且希望通过自己编写操作系统的方式来了解背后发生的故事，那么你一定可以在这本书中得到一些帮助。而假如你真的因为我的书而重新燃起实践的热情，从而开始一段操作系统旅程，我将会感到非常高兴。

不过我得坦白，在写作《自己动手写操作系统》的时候，我并不敢期待它能引起多少反响，一方面因为操作系统并不是时尚的话题，另一方面我也是走在学习的路上，或许只是比读者早走了一小步而已。然而出乎我的意料，它面世后重印多次，甚至一度登上销量排行榜的榜首，这让我觉得它的确有一定的参考价值，我要借此机会感谢所有支持我的读者。

在我写作《自己动手写操作系统》的时候，并没有想过今天会有一个第二版。原因在于，我希望这本书是用来填补空白的，而不是重复去做别人已经做得很好的事情。所谓填补空白，具体说就是让像我一样的操作系统爱好者在读完本书之后，能够有信心去读其他比较流行的开源的操作系统代码，有能力从零开始自己动手写操作系统，而这个任务第一版已经完成了。

那么为什么我又写作了第二版呢？原因有几个方面。第一，虽然第一版未曾涉及的进程间通信、文件系统等内容在许多书中都有讲解，但阅读的时候还是感觉有语焉不详的通病，作者本人可能很清楚原委，但写得太简略，以至于读者看来未必清晰。第二，我自己想把这个圈画圆。第一版的书虽然完成了它的使命，但毕竟到书的结尾，读者看到的不是一个真正的操作系统，它没有文件系统，没有内存管理，什么也干不了。在第二版中，你将会看到，你已经可以通过交叉编译的方式为我们的实验性 OS 编写应用程序了，也就是说，它已经具备操作系统的基本功能，虽然仍然极其简陋，但第一个圈，毕竟是已经圆起来了。第三，实践类的操作系统书籍还是太少了，以至于你要想看看别人是怎么做的，除了读以《操作系统：设计与实现》为代表的极少数书籍之外，就是一头扎进源代码中，而结果有时相当令人气馁。我自己也气馁过，所以我在第二版中，仍然试图把话说细一点，把自己的经验拿出来分享。而且我选择我能想到的最精简的设计，以便让读者不至于陷入太多细节而无法看到全貌。我想这是本书可能具有的价值所在——简化的易懂的设计，还有尽量详细的文字。

在这一版中，内容被划分成上下两篇。上篇基本上是第一版的修订，只是做了一个调整，那便是在兼顾 Windows 和 Linux 两方面用户的基础上，默认在 Linux 下建立开发环境来编写我们的操作系统。至于这样做的原因，在本书第 2 章有比较详细的说明。当然，开发环境毕竟是第二位的，书中讲述的内容以及

涉及的代码跟第一版都是一致的。本书的下篇全部都是新鲜内容，主要是增加了进程间通信、文件系统和内存管理。跟第一版的做法相同，下篇仍然不仅关注结果，更加致力于将形成一个结果的过程呈现出来。与此同时，由于本书旨在分享和引路，所以尽可能地简化了设计，以便将最重要的部分凸显出来。读者将看到，一个操作系统的文件系统和内存管理可以简陋到什么程度。简陋不是缺点，对于我们初学者而言，正是需要从简陋入手。换言之，如果你已经对实现一个操作系统有了一定的经验，那么这本书可能不适合你。这本书适合从来没有编写过操作系统的初学者。

本书的排版是我用LaTeX自己完成的。在排版中我花了一些工夫，因为我希望读者购买的首先是一本易于阅读且赏心悦目的书，其次才是编写操作系统的方法。另外，书中列出的代码均由我自己编写的程序自动嵌入LaTeX源文件，从而严格保证书和光盘的一致性，读者可以根据文件名和行号方便地找到光盘中代码的准确位置。

此外，在第二版中还有一些小的变化。首先是操作系统的名字改变了，原因在于虽然我们的试验性 OS 从前辈们那里借鉴了很多东西，但其各个部分的设计（比如文件系统和内存管理）往往有其独特之处，所以我将原先的 Tinix（本意为 Try Minix）改成了新名字Orange'S（这个名字来自于我的妻子☺），以表示它们的不同。另外，书中的代码风格，有些地方也做了调整。

我想，虽然第二版有着这样那样的变化，但有一点没有变，那就是本书试图将我在编写自己操作系统的过程中的经验尽可能地告诉读者，同时尽可能将我当初的思路和编码过程呈现出来。很可能读者比我更聪明，有更好的解决问题的方法，但无论如何，我认为我自己的经验可以为读者所借鉴。如果真是如此，我将会非常欣慰。

在第二版的编写过程中，我同样要感谢许多人。感谢我的父母和爷爷对我的爱，并希望爷爷不要为我担心，写书是件辛苦的事，但同时也使我收获良多。爸爸在第二版的最后阶段帮我订正文字，这本书里有你的功劳。我要感谢博文视点的各位朋友，感谢郭老师的理解和支持，感谢李玲的辛勤工作，感谢江立和李冰，你们的高效让我非常钦佩。我还要感谢孟岩老师，你给我的鼓励我一直记在心里。我要感谢我的挚友郭洪桥，不仅仅因为你在技术上给我的帮助，更加因为你在精神上给我的支持。感谢我的同事和朋友张会昌，你在技术上的广度和深度总令我钦佩。另外，在第一版中帮助我的人，我要再次谢谢你们，因为没有第一版，也就没有第二版。

在所有人中我最应该感谢和最想感谢的，是我的妻子黄丹红，感谢你给我的所有建议，还有你帮我画的图。尤其是，当这本书在我预想的时间内没有完成的时候，当我遇到困难迟迟不能解决的时候，你总在一旁给我鼓励，在你那里，我从来都能感觉到一种温暖，我深知，如果没有你的支持，我无法坚持下来将书写完。谢谢你，这本书同样属于你。

跟第一版相比，这本书涉及的内容触及操作系统设计的更多方面，而由于笔者的水平实在有限，难免有纰漏甚至错误。如果读者有任何的问题、意见或建议，请登录 http://www.osfromscratch.org，让我们共同探讨，共同进步。

本书导读

这本书适合谁

本书是一本操作系统实践的技术书籍。对于操作系统技术感兴趣，想要亲身体验编写操作系统过程的实践主义者，以及 Minix、Linux 源代码爱好者，都可以在本书中得到实践中所需的知识和思路。

本书以“动手写”为指导思想，只要是跟“动手写”操作系统有关的知识，都作为介绍对象加以讨论，所以，从开发环境的搭建，到保护模式，再到 IBMPC 中有关芯片的知识，最后到操作系统本身的设计实现，都能在本文中找到相应介绍。所以如果你也想亲身实践的话，本书可以省去你在书店和互联网寻找相应资料的过程，使你的学习过程事半功倍。在读完本书后，你不但可以获得对于操作系统初步的感性认识，并且对 IBMPC 的接口、IA 架构之保护模式，以及操作系统整体上的框架都将会有一定程度的了解。

笔者相信，当你读完本书之后，如果再读那些纯理论性的操作系统书籍，所获得的体验将会完全不同，因为那些对你而言不再是海市蜃楼。

对于想阅读 Linux 源代码的操作系统爱好者，本书可以提供阅读前所必要的知识储备，而这些知识储备不但在本书中有完整的涉及，而且在很多 Linux 书籍中是没有提到的。

特别要提到的是，对于想通过阅读 Andrew S. Tanenbaum 和 Albert S. Woodhull 的《操作系统：设计与实现》来学习操作系统的读者，本书尤其适合作为你的引路书籍，因为它翔实地介绍了初学者入门时所必需的知识积累，而这些知识在《操作系统：设计与实现》一书中是没有涉及的，笔者本人是把这本书作为写操作系统的主要参考书籍之一，所以在本书中对它多有借鉴。

你需要什么技术基础

在本书中所用到的计算机语言只有两种：汇编和 C 语言。所以只要你具备汇编和 C 语言的经验，就可以阅读本书。除对操作系统常识性的了解（比如知道中断、进程等概念）之外，本书不假定读者具备其他任何经验。

如果你学习过操作系统的理论课程，你会发现本书是对于理论的吻合和补充。它是从实践的角度为你展现一幅操作系统画面。

书中涉及了 Intel CPU 保护模式、Linux 命令等内容，到时候会有尽可能清晰的讲解，如果笔者认为某些内容可以通过其他教材系统学习，会在书中加以说明。

另外，本书只涉及 Intel x86 平台。

统一思想——让我们在这些方面达成共识

道篇

让我们有效而愉快地学习

你大概依然记得在你亲自敲出第一个“Hello world”程序并运行成功时的喜悦，那样的成就感助燃了你对编写程序浓厚的兴趣。随后你不断地学习，每学到新的语法都迫不及待地在计算机上调试运行，在调试的过程中克服困难，学到新知，并获得新的成就感。

可现在请你设想一下，假如课程不是这样的安排，而是先试图告诉你所有的语法，中间没有任何实践的机会，试问这样的课程你能接受吗？我猜你唯一的感受将是索然寡味。

原因何在？只是因为你不再有因为不断实践而获得的源源不断的成就感。而成就感是学习过程中快乐的源泉，没有了成就感，学习的愉快程度将大打折扣，效果于是也将变得不容乐观。

每个人都希望有效而且愉快的学习过程，可不幸的是，我们见到的操作系统课程十之八九令我们失望，作者喋喋不休地讲述着进程管理存储管理I/O控制调度算法，可我们到头来也没有一点的感性认识。我们好像已经理解却又好像一无所知。很明显，没有成就感，一点也没有。笔者痛恨这样的学习过程，也决不会重蹈这样的覆辙，让读者获得成就感将是本书的灵魂。

其实这本书完全可以称作一本回忆录，记载了笔者从开始不知道保护模式为何物到最终形成一个小小 OS 的过程，这样的回忆录性质保证了章节的安排完全遵从操作的时间顺序，于是也就保证了每一步的可操作性，毫无疑问，顺着这样的思路走下来，每一章的成果都需要努力但又尽在眼前，步步为营是我们的战术，成就感是我们的宗旨。

我们将从二十行代码开始，让我们最简单的操作系统婴儿慢慢长大，变成一个翩翩少年，而其中的每一步，你都可以在书中的指导下自己完成，不仅仅是看到，而是自己做到！你将在不断的实践中获得不断的成就感，笔者真心希望在阅读本书的过程中，你的学习过程可以变得愉快而有效。

学习的过程应该是从感性到理性

在你没有登过泰山之前，无论书中怎样描写它的样子你都无法想象出它的真实面目，即便配有插图，你对它的了解仍会只是支离破碎。毫无疑问，一千本对泰山描述的书都比不上你一次登山的经历。文学家的描述可能是华丽而优美的，可这样的描述最终产生的效果可能是你非去亲自登泰山不可。反过来想

呢，假如你已经登过泰山，这样的经历产生的效果会是你想读尽天下描述泰山的书而后快吗？可能事实恰恰相反，你可能再也不想去看那些文字描述。

是啊，再好的讲述，又哪比得上亲身的体验？人们的认知规律本来如此，有了感性的认识，才能上升为理性的理论。反其道而行之只能是事倍功半。

如果操作系统是一座这样的大山，本书愿做你的导游，引领你进入它的门径。传统的操作系统书籍仅仅是给你讲述这座大山的故事，你只是在听讲，并没有身临其境，而随着这本书亲身体验，则好像置身于山门之内，你不但可以看见眼前的每一个细节，更是具有了走完整座大山的信心。

值得说明的是，本书旨在引路，不会带领你走完整座大山，但是有兴趣的读者完全可以在本书最终形成的框架的基础上容易地实现其他操作系统书籍中讲到的各种原理和算法，从而对操作系统有个从感性到理性的清醒认识。

暂时的错误并不可怕

当我们对一件事情的全貌没有很好理解的时候，很可能会对某一部分产生理解上的误差，这就是所谓的断章取义。很多时候断章取义是难免的，但是，在不断学习的过程中，我们会逐渐看到更多，了解更多，对原先事物的认识也会变得深刻甚至不同。

对于操作系统这样复杂的东西来说，要想了解所有的细节无疑是非常困难的，所以在实践的过程中，可能在很多地方，会有一些误解发生。这都没有关系，随着了解的深入，这些误解总会得到澄清，到时你会发现，自己对某一方面已经非常熟悉了，这时的成就感，一定会让你感到非常愉悦。

本书内容的安排遵从的是代码编写的时间顺序，它更像是一本开发日记，所以在书中一些中间过程不完美的产物被有意保留了下来，并会在以后的章节中对它们进行修改和完善，因为笔者认为，一些精妙的东西背后，一定隐藏着很多中间的产物，一个伟大的发现在很多情况下可能不是天才们刹那间的灵光一闪，背后也一定有着我们没有看到的不伟大甚至是谬误。笔者很想追寻前辈们的脚步，重寻他们当日的足迹。做到这一点无疑很难，但即便无法做到，只要能引起读者的一点思索，也是本书莫大的幸事。

挡住了去路的，往往不是大树，而是小藤

如果不是亲身去做，你可能永远都不知道，困难是什么。

就好像你买了一台功能超全的微波炉回家，研究完了整本说明书，踌躇满志想要烹饪的时候，却突然发现家里的油盐已经用完。而当时已经是晚上十一点，所有的商店都已经关门，你气急败坏，简直想摸起铁勺砸向无辜的微波炉。

研究说明书是没有错的，但是在没开始之前，你永远都想不到让你无法烹饪的原因居然是十块钱一瓶的油和一块钱一袋的更加微不足道的盐。你还以为困难是微波炉面板上密密麻麻的控制键盘。

其实做其他事情也是一样的，比如写一个操作系统，即便一个很小的可能受理论家们讥笑的操作系统雏形，仍然可能遇到一大堆你没有想过的问题，而

这些问题在传统的操作系统书籍中根本没有提到。所以唯一的办法，便是亲自去做，只有实践了，才知道是怎么回事。

术篇

用到什么再学什么

我们不是在考试，我们只是在为了自己的志趣而努力，所以就让我们忠于自己的喜好吧，不必为了考试而看完所有的章节，无论那是多么的乏味。让我们马上投入实践，遇到问题再图解决的办法。笔者非常推崇这样的学习方法：

实践 → 遇到问题 → 解决问题 → 再实践

因为我们知道我们为什么学习，所以我们才会非常投入；由于我们知道我们的目标是解决什么问题，所以我们才会非常专注；由于我们在实践中学习，所以我们才会非常高效。而最有趣的是，最终你会发现你并没有因为选择这样的学习方法而少学到什么，相反，你会发现你用更少的时间学到更多的东西，并且格外的扎实。

只要用心，就没有学不会的东西

笔者还清楚地记得刚刚下载完 Intel Architecture Software Developer Manual 那三个可怕的 PDF 文件时的心情，那时心里暗暗嘀咕，什么时候才能把这些东西读懂啊！可是突然有一天，当这些东西真的已经被基本读完的时候，我想起当初的畏惧，时间其实并没有过去多少。

所有的道理都是相通的，没有什么真正可怕，尤其是，我们所做的并非创造性的工作，所有的问题前人都曾经解决，所以我们更是无所畏惧，更何况我们不仅有书店，而且有互联网，动动手脚就能找到需要的资料，我们只要认真研究就够了。

所以当遇到困难时，请静下心来，慢慢研究，因为只要用心，就没有学不会的东西。

适当地囫囵吞枣

如果囫囵吞枣仅仅是学习的一个过程而非终点，那么它并不一定就是坏事。大家都应该听说过鲁迅先生学习英语的故事，他建议在阅读的过程中遇到不懂的内容可以忽略，等到过一段时间之后，这些问题会自然解决。

在本书中，有时候可能先列出一段代码，告诉你它能完成什么，这时你也可以大致读过，因为下面会有对它详细的解释。第一遍读它的时候，你只要了解大概就够了。

本书的原则

1. 宁可啰嗦一点，也不肯漏掉细节

在书中的有些地方，你可能觉得有些很“简单”的问题都被列了出来，甚至显得有些啰嗦，但笔者宁可让内容写得啰嗦点，因为笔者自己在读书的时候有一个体验，就是有时候一个问题怎么也想不通，经过很长时间终于弄明白的时候才发现原来是那么“简单”。可能作者认为它足够简单以至于可以跳过不提，但读者未必那么幸运一下子就弄清楚。

不过本书到后面的章节，如果涉及的细节是前面章节提到过的，就有意地略过了。举个非常简单的例子，开始时本书会提醒读者增加一个源文件之后不要忘记修改 Makefile，到后来就假定读者已经熟悉了这个步骤，可能就不再提及了。

2. 努力做到平易近人

笔者更喜欢把本书称作一本笔记或者学习日志，不仅仅是因为它基本是真实的学习过程的再现，而且笔者不想让它有任何居高临下甚至是晦涩神秘的感觉。如果有一个地方你觉得书中没有说清楚以至于你没有弄明白，请你告诉我，我会在以后做出改进。

3. 代码注重可读性但不注重效率

本书的代码力求简单易懂，在此过程中很少考虑运行的效率。一方面因为书中的代码仅仅供学习之用，暂时并不考虑实际用途；另一方面笔者认为当我们对操作系统足够了解之后再考虑效率的问题也不迟。

本书附带光盘说明

本书附带光盘中有本书用到的所有源代码。值得一提的是，其中不止包含完整的操作系统代码，还包含各个步骤的中间产物。换句话说，开发中每一步骤的代码，都可在光盘中单独文件夹中找到。举例说明，书的开篇介绍引导扇区，读者在相应文件夹中就只看到引导扇区的代码；第 9 章介绍文件系统，在相应文件夹中就不会包含第 10 章内存管理的代码。在任何一个步骤对应的文件夹中，都包含一个完整可编译运行的代码树，以方便读者试验之用。这样在学习的任何一个阶段，读者都可彻底了解阶段性成果，且不必担心受到自己还未学习的内容的影响，从而使学习不留死角。

在书的正文中引用的代码会标注出出自哪个文件。以“chapter5/b/bar.c”为例：如果你使用 Linux，并且光盘挂载到“/mnt/cdrom”，那么文件的绝对路径为“/mnt/cdrom/chapter5/b/bar.c”；如果你使用 Windows，并且光盘是 X: 盘，那么文件的绝对路径为“X:\chapter5\b\bar.c”。

目　录

上　篇

下　篇