# 自己动手写操作系统

于渊 编著
尤晋元 审校

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

# 自己动手写操作系统

于 渊 编著
尤晋元 审校



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

## 内 容 简 介

本书在详细分析操作系统原理的基础上，用丰富的实例代码，一步一步地指导读者用 C 语言和汇编语言编写出一个具备操作系统基本功能的操作系统框架。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。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。全书共分 7 章。

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**

自己动手写操作系统 / 于渊编著. —北京：电子工业 出版社，2005.8

ISBN 7-121-01577-3

Ⅰ. 自… Ⅱ. 于… Ⅲ. 操作系统 Ⅳ. TP316

中国版本图书馆 CIP 数据核字（2005）第 079890 号

责任编辑：张 毅 zhangyi@phei.com.cn
印 刷：北京智力达印刷有限公司
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
经 销：各地新华书店
开 本：787×980 1/16 印张：24.75 字数：511 千字
印 次：2005 年 8 月第 1 次印刷
印 数：5000 册 定价：48.00 元（含光盘 1 张）

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系电话：（010）68279077。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。

# 序

一年多以前，电子工业出版社的张毅编辑告诉我说，有一位年轻的程序员，正在写一本《自己动手写操作系统》的书。知道这个消息，我既有点好奇，又有些担忧。如果是在十年前，这样题材的书将会是读者争相传阅的对象，毕竟 20 世纪 90 年代是软件的理想主义年代。但是在理想褪尽、实务未兴的尴尬的这两年，这样一本书在市场上究竟会遇到怎样的待遇，确实让人不敢乐观。不过，在阅读了样章之后，我深为作者清新的文笔、流畅的思路和扎实的技术功底所折服，于是请张毅为我引见了这位作者，即本书的作者于渊。

于渊非常年轻，却有着高人一筹的表达能力和技术视野，我觉得他是难得的技术写作人才，就鼓励他在《程序员》杂志开辟了一个技术专栏，专门剖析操作系统相关的技术。一年来这个专栏陆续发表了一系列文章，获得了不少读者的正面反馈。

然而，事实证明，我最初的担忧并不是没有道理的。一年多来，不断有人表达过他们对这样一个题材的不同看法。他们认为，相对于 90 年代中后期，现在的软件产业已经务实了很多，今天的程序员更关心的是如何尽可能快、尽可能简单地用软件解决实际问题，创造实际价值，在一个既定的秩序中寻找自己的生存空间，而不是异想天开地憧憬能成为 Linus Torvalds 式的旧秩序的“破坏者”。因此诸如软件过程、开发方法、系统集成、应用架构等“高级”的话题受到关注和欢迎，而诸如操作系统、编译原理之类的基础技术，已经是关心者寥寥了。他们非常怀疑，这样的一本书，对于一线的开发者是否有实际的意义？对于尚在寻找自己职业发展方向的初学者是否构成一种误导？这个问题相当尖锐，必须面对。我想这样一本书，至少在以下几个方面是具有重要的正面意义的。

首先，对于正在大学里学习计算机科学的学生来说，“操作系统原理”是重要的专业基础课。为了达到大学阶段教育的标准，这方面的知识应当认真学习。一些比较严肃的学校鼓励学生在学习这门课程的同时自己动手开发一个具体而微的操作系统。这种实习对于学生充分掌握书本知识、打下扎实的基本功有非常大的好处。在我认识的比较有成就的开发者中，有不少人自己动手写过小的操作系统，他们认为编写操作系统的实践使他们最终消除了对编写软件系统的心理障碍，实在地消化和理解了书本上的知识，学会了解决问题的思路，收获非常巨大。可惜的是，大部分的学生都没有进行过这样的实践，这主要是因为目前的课本偏重操作系统理论，把大量的笔墨放在对操作系统运行机制的剖析或者现成源代码的分析上，对于那些想自己动手写一个操作系统的同学来说，从课本上反而得不到实际的指导。即使是一些世界级的名著，在“How”上也是语焉不详。

在这方面，我相信于渊的这本书在国内算是填补了一个空白。这本书最大的特点是明白、实在，将学习编写操作系统的每一个步骤都清清楚楚地交代出来，丝毫没有含糊其辞之处。可以说，只要读者能够耐心阅读学习，按照书上交代的步骤一步步来，就肯定能够进入操作系统的大门，把书本上的知识与实践紧密联系起来。毕竟写自己的操作系统是一个让所有程序员心动的事情。如果当年我学习操作系统知识的时候能够有这样一本书，那该有多好！

其次，对于那些希望通过分析 Linux 源代码学习与研究操作系统，进而在开源软件天地里有所作为的研究者和开发者来说，这本书是非常好的入门阶梯。目前研究 Linux 内核的图书，一般局限在对现有内核源代码的分析上，不但理解起来很困难，而且没有给读者以自己实践的机会。有人想到去分析 Linux 早期的版本，降低了读者理解的难度，但是总的来说还是纸上谈兵。本书的风格截然不同，不但行文活泼清新，叙理简明清晰，而且完全着眼于动手，以一种夹叙夹议的方式，对于编写操作系统过程中可能遇到的各种问题“逢山开路，遇水架桥”，读者可以在实际的语境中理解问题，解决问题。通过这种方式学习操作系统的实现技术，无疑要比其他方式更为有效。而且，于渊在这本书中构造的这个微型的操作系统，跟 Linux 有微妙的相关，读者细心品味便知。

另外，虽然目前国内软件产业的主流是做下游的生产性集成，但是对于程序员个体来说，也有不少从事系统级软件开发的机会。有幸从事系统级软件开发的朋友，更是可以直接地从本书中学到不少实用的知识和技能。特别是作者在解决一个又一个问题的过程中所体现出来的思路和方法，可能是更值得大家学习的东西。

众所周知，操作系统是计算机软件领域中核心的工程性技术，尽管它的理论相对成熟，但是在工程实施和维护上，仍然是体现一个国家软件技术水平的“两弹、一星、大飞机”级的标志性核心技术。世界上凡是在软件产业方面存有雄心壮志的国家，无不非常重视操作系统技术的研究和积累。比如法国在他们的一个国家级实验室中，自己研发了包括操作系统和编译器在内的全套基础软件，并由国家投入资金不断维护和发展。德国拥有大批 Linux 黑客，其政府因势利导，通过一系列的大型工程将自己的 Linux 软件人才组织起来，希望依托 Linux 重建自己的软件核心技术力量。20 世纪 80 年代中期，日本在美国的压力下而放弃了自己的“BTRON”操作系统，此后软件产业的发展让日本追悔莫及。痛定思痛之后，日本希望牢牢地把握自己在消费电子产品上的优势，一方面继续发展国产的 ITRON OS，另一方面把握住 Linux 的机会，希望在未来占据消费类嵌入式操作系统的制高点。我国在这个方面走过一些弯路，但是现在已经认识到了掌握核心软件技术的重要性，并且有了一定的投入，相信今后国家在这方面的支持力度会越来越强。我本人见过国内的一些操作系统方面的专家，切实地感到，就个体而言，国内的技术专家在理论和实践上都达到了很高的水平，但是由于缺乏一个质量高、并且有一定规模的

团队和社群，他们基本处于单打独斗或者小组作战的状态，不仅个人的技术不能够得到充分地发挥，而且也不能形成有规模的成果，无法从根本上扭转我国在软件核心技术领域上的劣势。

我认为，只有在中国出现一大批关心操作系统、熟悉操作系统的程序员，才有可能逐渐缩小我们与世界先进水平的差异。

作为中国软件产业中的普通一员，我非常希望看到这本书能够在这个过程中发挥一点作用。

孟　岩

《程序员》杂志技术主编

2005 年 7 月

# 审校者的话

这是一本编程爱好者编写的别具一格、颇有特色的操作系统原理与实现的书。该书作者对操作系统具有特殊爱好，在大量实践和反复钻研下积累了丰富而可贵的经验，为了与广大读者分享这些经验写成了此书。

本书对一般的操作系统原理教材不很重视的部分，例如，系统初启、保护模式、控制权如何转入 OS Kernal 等都写得具体详细，对操作系统的爱好者以及涉足于操作系统设计、实现和应用的读者有很好的参考价值。

本书的文字生动活泼，富有个性，可望提高青年学子的阅读兴趣。

尤晋元

尤晋元老师简介：

上海交通大学教授，博士生导师，国内操作系统领域著名权威专家，代表著作（含译著）有：《Windows 操作系统原理》，《操作系统：设计与实现》，《莱昂氏 UNIX 源代码分析》等。

# 作 者 自 序

你是否有过这样的经历，有一天你兴致勃勃买来一堆菜谱想学厨艺，翻开之后却发现自己根本没见过那些材料的名字，也不知道什么叫文火什么叫武火，什么叫上浆什么叫勾芡。而菜谱里根本没告诉你！你扔掉菜谱，垂头丧气，从此对厨艺失去兴趣。

你也可能会有这样的经历，当你在计算机课上学完了一堆 C 语言语法，想要大展身手实践一番的时候，突然发现你居然不知道源代码应该敲到哪里，是 Word 还是 NotePad？

很多计算机自学者可能有过这样的经历，由于不知道如何跟踪调试，在辛辛苦苦编写的程序得不出正确的结果时，要么束手无策，要么用打印语句输出很多东西，费时费力，而教科书根本没教你这些操作的细节。

有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，约定俗成是由学习者本人去摸索出来的。但是实际情况恰恰是，这些书中忽略掉的内容可能占去了一个初学者大部分的时间，甚至因此影响了学习的热情。

学 C 语言是很容易找到老师的，你会被详细地告知 IDE 是什么，以及如何使用。但是学习操作系统呢？你会发现绝大多数操作系统书籍都只讲原理，讲各种各样的算法和策略。如果是为了考试，你将内容背下来，最后可能得一个高分；如果是出于兴趣，怕是读了没几页就感到索然无味了。你或许的确能找到极少数的书籍告诉你怎样去写，比如 Andrew S. Tanenbaum 和 Albert S. Woodhull 的《操作系统：设计与实现》，但是两位先生还是没能告诉你从哪里开始。你还是不得不在一开始的时候在浩瀚的因特网上搜索一个 BootSector 的写法。

你或许听说过张五常，他为了研究经济学问题亲自跑到大街上去卖橘子，后来写成了著名的《卖橘者言》，成为了实证经济学的典范。他没有仅仅躲在房子里研究，因为他相信通过实践得来的经验才最可靠、最深刻的。我想他真的是一个喜欢追根究底的人。

你可能也喜欢探求问题的本质，想了解事情的各个细节。面对神奇的计算机世界，很想知道为什么打开电源，电脑屏幕上就能出现这样色彩斑斓的图像，很想知道操作系统理论书籍中讲到的进程管理到底怎样实现，很想知道 DOS 和 Windows

到底有什么本质上的区别，想知道怎样才能像那些伟大的黑客一样参与修改 Linux 的源代码。是的，我就是这样一个喜欢探求本质的人，对这一切怀有极大的兴趣，于是我想写一个自己的操作系统，因为我知道只有通过自己动手，才能对它有真正深刻的了解。经过一段时间的努力，我终于完成了一个雏形。回头想想，很庆幸自己能克服困难走了过来。同时，我在网上了解到还有很多朋友也在写自己的操作系统，也遇到了许多困难。为了跟大家分享其中的经验，让后来者不至于走同样的弯路，我把自己的开发过程记录下来，希望能为初学者们做参考。

本书有幸得到国内操作系统方面的研究权威、上海交通大学尤晋元教授的审校和指点，在此我要表示最真心的敬意和最诚挚的感谢！

在本书的写作过程中，我也有幸得到了许多人的帮助。在我遇到一些未曾预料的困难和变故时，得到了电子工业出版社的张毅编辑以及《程序员》杂志社的孟岩先生的理解、宽容和支持，在这里，我要对你们表示衷心的感谢！

我同样想把感谢献给朱风明、郝慎水、包立光、李雷、郭洪桥和张璐。在整个过程中，你们都在方法上、精神上和物质上给予我巨大而无私的帮助。没有你们的付出，我不可能完成这项工作。

我还要感谢我的家人，爷爷、爸爸和妈妈，你们的爱和关怀是本书得以完成的保障。

要列出所有帮助过我的人的名字是不可能的，因为有些困难是通过因特网解决的，我甚至不知道他们的名字。在此，谨向他们一并表示感谢！

最后，以下面四句诗与读者共勉：

在你立足处深挖下去，
就会有泉水涌出！
别管蒙昧者们叫嚷：
“下面永远是地狱！”

——尼采

于　渊

# 本书导读

纸上得来终觉浅，绝知此事要躬行。

——陆游

## 本书适合谁读

本书是一本操作系统开发实践的技术书籍，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，都可以在本书中了解到实践所需的知识和思路。

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，都进行讨论，从开发环境的搭建到保护模式，再到 IBM PC 中有关芯片的知识，最后到操作系统本身的设计实现，都能在本书中找到相应介绍。所以，如果你也想亲身实践的话，本书可以使你的学习过程事半功倍。在读完本书后，你不但对于操作系统有一个初步的感性认识，并且对 IBM PC 的接口、IA 架构的保护模式，以及操作系统整体框架都会有一定程度的了解。

当你读完本书之后，再读那些纯理论性的操作系统书籍，体验将会完全不同，因为那些对你而言已不再是海市蜃楼了！

对于想阅读 Linux 源代码的操作系统爱好者来说，本书可以提供阅读前所必需的知识储备，这些知识也是很多 Linux 书籍没有提到的。

特别要说明的是，对于想通过 Andrew S. Tanenbaum 和 Albert S. Woodhull 所著的《操作系统：设计与实现》来学习操作系统的读者，本书尤其适合作为引路指南，因为它详细地介绍了初学者入门时所必需的知识，而这些知识在《操作系统：设计与实现》一书中是没有涉及的。笔者本人把该书作为写操作系统的主要参考书籍之一，所以在写作本书时对它多有借鉴。

## 你需要具备的基础

在本书中所用到的计算机编程语言只有两种：汇编语言和 C 语言。所以，只要你具备汇编语言和 C 语言的知识，就可以阅读本书。除对操作系统有常识性的了解（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，就会发现本书是针对理论知识的补充，它是从实践的角度为你展现一幅操作系统画面。

书中涉及到 Intel CPU 保护模式、Linux 命令等内容。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

- **有效而愉快地学习**

你大概依然记得，在亲自敲出第一个“Hello world”程序并运行成功时的喜悦，那样的成就感点燃了你对编写程序的浓厚兴趣。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，又学到新知识，并获得新的成就感。

可现在请你设想一下，假如课程不是这样安排的，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？我想惟一的感受就是索然无味。

原因何在？因为你体会不到通过不断实践而带来的一次一次的成就感。而成就感是学习过程中快乐的源泉。没有了成就感，学习效率将大打折扣。

每个人都希望有效且愉快地学习，可不幸的是，我们见到的操作系统课程十之八九都是在喋喋不休地讲述着进程管理、存储管理、I/O 控制、调度算法，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。笔者厌烦这样的学习过程，也绝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，本书完全可以称做一本回忆录，记载了笔者从一开始（不知道保护模式为何物）到最后（形成一个小小的操作系统）的全过程。回忆录性质保证了章节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近在眼前。步步为营是我们的战术，不断获取成就感是我们的宗旨。

我们将从 20 行代码开始，使最最简单的操作系统婴儿慢慢长大，长成一位翩翩少年。而其中的每一步，都可以在书的指导下自己完成。不仅仅是可以看到，而且是自己能够做到！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的提升过程**

在你没有登过泰山之前，无论书中怎样描写它的样子，你都无法想像出它的真实面目，即便配有插图，你对它的了解也仍是支离破碎的。毫无疑问，一千本描述泰山的书都比不上你一次登山的经历。文学家的描述是华丽而优美的，可这样的描述最终产生的效果是促使你非去亲自登泰山不可。反过来呢？假如你已 经登过泰山，这样的经历所产生的效果会使你想读尽天下所有描述泰山的书吗？恰恰相反，你可能再也不想去看那些文字描述了。

是啊，再好的讲述也比不上亲身的体验。人们的认知规律本来如此，有了感性的认识，才能上升为理性的思考。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，引领你进入其中。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。而随着本书去亲身体验，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。

要强调一点，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轻松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地理解的时候，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完全不同。

对于操作系统这样复杂的事物来说，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产生。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，一定会让你备感愉悦。

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中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，精妙的背后一定隐藏着很多中间产 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，但即便不足以做到，只要能引起读者们的一点思索，也是莫大的欣慰。

- **挡住前路的，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什么是困难。就好像你买了一台功能很全

的微波炉回家，研究完整本说明书，踌躇满志地想要烹饪的时候，却突然发现家里的油盐已经用完。而当时已经是晚上 11 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 10 块钱 1 瓶的油，以及更加微不足道的 1 块钱 1 袋的盐。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，本书详述了这一大堆可能遇到却想不到的问题。所以，惟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，只是出于自己的兴趣，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。让我们马上投入实践，遇到问题再寻觅解决的办法。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

由于我们知道我们为什么学习，所以才会非常投入；由于我们知道我们的目标是解决什么问题，所以才会非常专注；由于我们在实践中学习，所以才会非常高效。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了更多的东西，并且格外扎实。

### • 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 3 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读完的时候，我想起当初的畏惧，算来时间其实并没有过去多久。

所有的道理都是相通的，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过，所以我们更应无所畏惧。更何况不仅有书店，而且有因特网，动动手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，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掠过，

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一下，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

- **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。因为笔者自己在读书的时候有一个体验，就是有时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”。可能该书的作者认为它足够简单以至于可以跳过不提，但读者未必一下子就能弄清楚。所以，本书在很多地方将尽量地细节阐述得很清楚，以节省、读者理解的时间。

在本书后面的章节中，如果涉及的细节是前面章节提到过的，就会有意地略过。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，就不再提及了。

- **努力做到平易近人**

笔者更喜欢把本书称做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。如果哪怕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。

- **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，暂时并不考虑用做实际用途；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也不迟。

## 光盘说明

书后所附光盘中有本书用到的所有源代码。源代码并不是只有一份，而是每增加一部分模块就有一份，后一部分所附代码是在前一部分基础上完成的，所以读者在阅读本书的时候可以做到彻底的步步为营，你不必从一大堆不必要的代码中找出自己想要的部分。

在书的正文引用的代码中会标注出出自哪个文件。以在 Windows 下为例，如果光盘的盘符是“F:”，则“\chapter5\b\bar.c”表示文件的绝对路径为“F:\Tinix\ chapter5\b\bar.c”。

另外，光盘中还有一些必要的小工具，以供读者方便地使用。

光盘内容摘要请见下表。

| 位　置 | 内　容 | 说　明 | |
|---|---|---|---|
| \Tinix | 书中涉及的全部源代码 | 书中的章节和代码对照表请见附录 | |
| | 其中很多目录中除了包含源代码（*.asm、*.inc、*.c、*.h）外，还有其他类型的一些文件 | boot.bin | 引导扇区（Boot Sector），可通过 FloppyWriter 写入软盘（或软盘映像） |
| | | loader.bin | Loader，直接复制至软盘（或软盘映像）根目录 |
| | | kernel.bin | 内核（Kernel），直接复制至软盘（或软盘映像）根目录 |
| | | bochsrc.bxrc | Bochs 配置文件，如果系统中安装了 Bochs 2.1.1 可直接双击运行它。其他细节请见 2.7 节 |
| | | godbg.bat | 调试时可使用此批处理文件。它假设 Bochs 2.1.1 安装在 D:\Program Files\Bochs-2.1.1\中 |
| | | TINIX.IMG | 软盘映像，可直接通过 Bochs 或者 Virtual PC 运行 |
| | | *.com | 可以在 DOS（必须为纯 DOS）下运行的文件 |
| \Tools | 一些小工具（在 VC 6 下编译通过） | DescParser | 描述符分析器，输入描述符的值，可以得出其基址、界限、属性等信息 |
| | | ELFParser | ELF 文件分析器，可以列出一个 ELF 文件的 ELF Header、Program Header、Section Header 等信息 |
| | | FloppyWriter | 用以写引导扇区，支持软盘和软盘映像 |
| | | KrnlChecker | 用以检查一个 Tinix 内核加载后位置是否正确 |

# 目　录