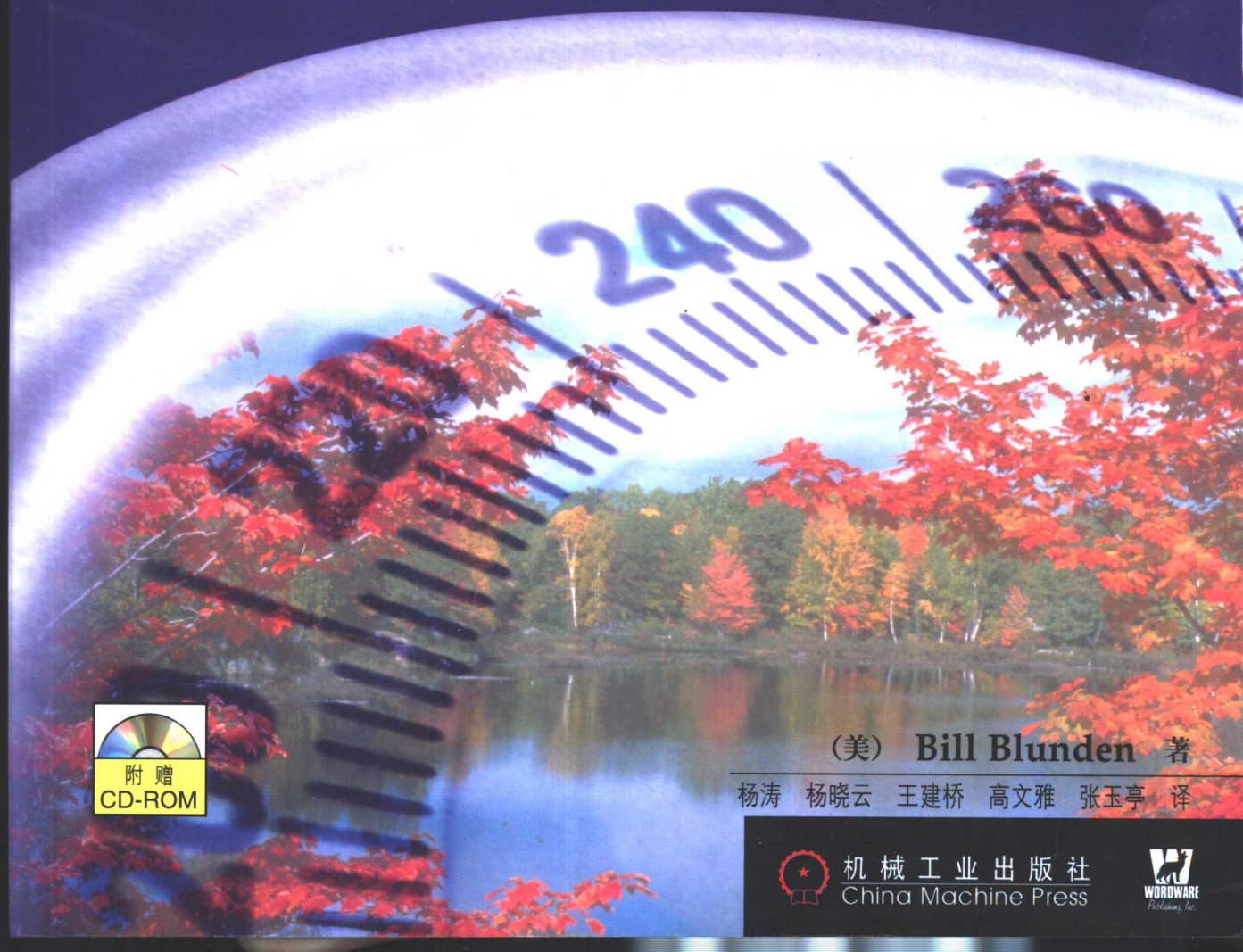
HZ BOOKS

Virtual Machine Design and Implementation in C/C++

开发人员专业技术丛书

虚拟机的设计与实现
——C/C++



附赠 CD-ROM

(美) Bill Blunden 著

杨涛 杨晓云 王建桥 高文雅 张玉亭 译

机械工业出版社
China Machine Press

WORDWARE Publishing Inc.

开发人员专业技术丛书

虚拟机的设计与实现

——C/C++

（美） Bill Blunden 著

杨涛 杨晓云 王建桥 高文雅 张玉亭 译

机械工业出版社
China Machine Press

本书的作者曾经是一位物理学家，所以在计算机领域有着坚实、严谨的理论基础，他从自己的实践出发，采用了数学中的SOP方法（命题－证明－示例），深入浅出地论述了本书的三大部分：概述、HEC虚拟机、HEC汇编语言，对虚拟机的设计目标、HEC虚拟机、HEC汇编器、HEC调试器、HEC中断、HEC汇编语言的使用方法依次进行了描述，对HEC虚拟机在Windows和UNIX上的实现进行了比较。

本书是第一本对虚拟机及其全套开发工具做出完整细致介绍的工具书。它能帮助系统工程师摆脱对计算机硬件制造商的依赖，也适用于学习计算机工作原理的学生，是虚拟机方面不可多得的一本好书。

Bill Blunden: Virtual Machine Design and Implementation in C/C++ (ISBN: 1-55622-903-8).

Authorized translation from the English language edition published by Wordware Publishing, Inc.

本书版权登记号：图字：01-2002-4719

图书在版编目(CIP)数据

虚拟机的设计与实现——C/C++ / (美) 布朗登 (Blunden, B.) 著；杨涛等译. – 北京：机械工业出版社，2002.11

(开发人员专业技术丛书)

书名原文：Virtual Machine Design and Implementation in C/C++

ISBN 7-111-11111-7

Ⅰ.虚… Ⅱ.①布… ②杨… Ⅲ.C语言－程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字(2002)第082912号

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：张金梅

北京第二外国语学院印刷厂印刷 · 新华书店北京发行所发行

2003年1月第1版第1次印刷

787mm × 1092mm 1/16 · 43.25印张

印数：0 001-4 000册

定价：76.00元（附光盘）

前　言

原来作为物理学家的时候，我痛苦地发现：几乎一切现有领域——如粒子物理方面——的突破性研究和实验都离不开联邦预算的支持，没有大笔的预算，你就什么工作也开展不了。但计算机科学方面的研究工作却不是这样。你只需破费一两百美元就能买到一台二手电脑，再多花50美元就能买到一套开发工具并投入工作。一台廉价的机器再加上现有的各种工具就足以让你开展重要的研究工作。正是这一点使我在1994年末从计算物理领域跳槽到了计算机科学领域里来。有一天，我从公寓大楼外的垃圾堆里刚捡回来一台废弃的80386电脑，当时的我正忙于填写各种物理方面的研究生课程申请表。在更换了硬盘之后，我让那台废电脑又焕发了青春。接下来，我决定撕掉那些申请表去学习计算机科学。我认为自己用来买那块硬盘的200美元是我这一生中最好的投资之一。

从根本上讲，计算机科学直到现在还是一门很容易进入的领域——坐在电脑前摆弄各种想法和方法是一件非常容易的事情。只要拥有了一台个人电脑，你就等于拥有了一座超一流的实验室。当然，我也不是一点儿后悔都没有，毕竟我一生中已经有7年光阴花在了物理学方面——这个领域里的顶级实验设备动辄就是数百万美元。这本书基本上是对我两年来的研究工作的一个总结，其内容主要与HEC运行时系统（run-time system）的设计和实现有关。

“HEC”这个名字是从喜剧连环漫画《CPU Wars》（CPU战争）借用来的，这部漫画虚构了一个Human Engineered Computer Company（人力计算机公司，HEC公司）的员工与来自Impossible to Purchase Machines（买不到机器公司，IPM公司）的极权主义刺客进行斗争的故事。那是在里根（Reagan）担任美国总统的20世纪80年代，当时的我还是一名中学生。我错过了计算机软件发展史上最具承前启后意义的二十年。我本人从没使用穿孔卡片编写过程序，也从没使用电传打字机输入过程序命令。我也没听说过我们那一代人里的任何一位软件工程师曾经使用过小型计算机（minicomputer）。事实上，在1984年，我能够接触到的真正的商用电脑是一台采用8088芯片的IBM PC。我给自己的运行时系统起名为“HEC”是想让那些年轻的工程师能够回想起他们的前辈，也是想对那些使用16进制代码编写程序并开创了90年代因特网繁荣局面的老一辈软件大师致以崇高的敬意。

任何一个在60年代和70年代编写过程序的人都知道，喜剧漫画《CPU Wars》里虚构出来的两个公司名称“HEC”和“IPM”实际上暗指着Digital Equipment Corporation（数字设备公司，DEC）和International Business Machines（国际商用机器公司，IBM）。DEC公司在1961年推出的PDP-1开创了小型计算机的先河。此前的计算机都是些庞大而又昂贵的大型设备，它们通常都安装在一个独立的建筑物，并且往往会有一队武装人员来守护着它们。向这类大型机（mainframe）提交作业就像是去梵帝冈面见教皇。不止一位老工程师给我讲过诸如不得不在凌晨三点签字进入机房去使用计算机之类的“恐怖”故事。

小型计算机把一切都改变了。体积更小、价格更低的小型计算机提供了一个方便的时间共

享环境。程序员们再也用不着把他们的穿孔卡片交给不可一世的大型机操作人员了。通过小型计算机，软件开发人员不再需要看别人的脸色，也不再有浪费生命的等待。与大型机相比，小型计算机对广大的软件开发人员更友善，而且也更容易使用。小型计算机的迅速流行可能给IBM的销售人员带来了很大的压力。

注意 让IBM公司感到头疼不已的公司并不仅仅是DEC公司一家。早在1964年，Control Data Corporation（控制数据公司，CDC）就已经向世界推出了CDC 6600计算机。CDC 6600是当时世界上速度最快、功能最强大的计算机，它使IBM产品线的顶级产品相形见绌。但这并不会令人感到惊讶——因为开发这一产品的带头人就是大名鼎鼎的Seymour Cray（塞缪尔·克雷）。CDC 6600最初的售价是7 000 000美元，CDC公司总共卖出了大约50台。据说，IBM公司的高层领导对CDC公司推出的这种掩盖了IBM光芒的产品大为紧张，于是想出了一个纸老虎的把戏——对外宣传说IBM很快会推出它自己的超级计算机。可6个月后的事实却证明了那只不过是IBM公司为了阻挠CDC 6600的流行而虚张的声势。

DEC首创的小型计算机也是催生出UNIX操作系统的重要因素之一。在1968年，贝尔实验室（Bell Labs）的一位研究员Ken Thompson刚接手了一台DEC PDP-7计算机，他认为那台机器是开发一个名为“Space Travel”（星际旅行）游戏的绝佳平台。Ken是MULTICS项目的一位参加者。MULTICS（MULTIplexed Information and Computing Service，多重信息与计算服务）是一个由贝尔实验室、通用电气（General Electric）和麻省理工学院（MIT）联合发起的计算机项目。但MULTICS项目却不得不面对这样一个问题，即当时的硬件水平无法真正支持该项目预计提供的操作系统级功能。贝尔实验室放弃了这个项目，给可怜的Ken留下了大把的“业余”时间。Ken最初只是想利用PDP-7来实现“Space Travel”（星际旅行）游戏，可他的开发工作却不断地因操作系统内部的bug而受到影响。随着bug数量的增加，开发工作受到的影响越来越严重，Ken建立一种新操作系统的想法也越来越强烈和成熟。Ken决定根据MULTICS的要求写一个规模略小的版本供自己使用。不久之后，UNIX操作系统就诞生了。UNIX这个名字原来叫做UNICS（UNIplexed Information and Computing Service，单一信息与计算服务）。Windows群体半开玩笑地把它称为“eunuchs”（中文的意思是“太监”），这可能是因为UNIX操作系统的用户界面很差劲。

现在是2002年了，DEC公司已经成为了一种记忆。在90年代末被Compaq公司兼并之后，DEC这个品牌逐渐退出了人们的视线。

本书采用的论述方法

本书将要介绍的HEC运行时系统由一个执行引擎（execution engine）、一个机器级调试器（machine-level debugger）、一个汇编器（assembler）以及一整套相关的开发工具组成。我从最基本的零件开始建立HEC运行时系统。在开发过程中，我遇到了几个体系结构方面的问题。对这些问题的思考最终形成了关于我所构造的程序的定义。我认为，先给出一堆理由再把源代码摆到你面前并不是一个好的方法，为了让大家更好地理解我的成果，我决定采用一种与众不同的思路。

专业数学家遵循的标准操作流程（standard operation procedure，SOP）是给出命题、做出证明、然后给出一个示例。大多数数学系的研究生甚至从没考虑过这个问题——他们对“命题-证明-示例”这种操作流程见得太多了，所以只要是写论文，他们就会不假思索地采用这个流程。为了能以统一的严谨和格式去解释我所遇到的体系结构方面的问题，我决定采用一种类似的方法。我采用的SOP包括三个基本步骤：

1) 给出一个设计问题以及必要的背景知识。

2) 提出一个解决方案。

3) 对其他替换方案进行讨论，比较它们之间的利弊。

第2章里的大部分内容都是按此流程论述的。我将以概述有关基本概念和理论来开始各论题的讨论。也就是说，我会先搭建一个舞台，然后再在这个舞台上对我做出的决定进行解释。我认为这个方法能够让大家更好地理解我在解决具体问题时的具体思路。事实上，我的设计目标会给我的工作加上一些基础性的限制条件，如果读者不熟悉这些限制条件，就可能会嘲笑我的某些决定——对这一点我是相当肯定的。

最后，每个决定都有它不利的方面。例如，如果某个列表数据结构是用一个数组（array）来实现的，你就可以获得很高的访问速度，但要以降低其灵活性为代价。如果某个列表数据结构是用一个链表（linked list）来实现的，你就可以方便地增减该列表的长度，但这将以牺牲访问速度为代价。很难找到一种能够在各种条件下都达到最优效果的解决方案。这种两难局面在计算机科学里十分常见，大家在这本书里也会经常遇到。因此，我在每项设计决定的后面会立刻对该决定的利弊做出分析，让大家既能随时了解该项决定的好处，也能随时明白我牺牲了些什么。

注意 *这种两难局面并不仅仅局限于计算机科学的各个领域。除非你拥有无限的资源，否则，加在其上的限制条件和要求就会不可避免地导致一些不利因素。*

因为一个运行时系统必须具备一些通常需要由操作系统和硬件平台来负责实现的功能，所以我所涉及的背景资料的来源很广、很杂。我已经尽力使这本书做到尽可能的全面和自成体系。有兴趣做进一步钻研的读者可以去阅读我在各章结尾处列出的参考资料。

本书的适用读者

这本书主要供两类读者参考：

- 系统工程师。
- 计算机科学系的学生。

之所以说这本书对系统工程师很有用，是因为它能帮助他们摆脱对独断的计算机硬件制造商的依赖。如今，计算机硬件的体系结构越来越复杂，要想跟上摩尔定律（即一块芯片所能容纳的晶体管数量每隔18到24个月就会翻一番），工程师就不得不面对更大的挑战。例如，对负责开发编译器（compiler）的系统程序员们来说，与80年代的处理器相比，如今的EPIC（Explicitly Parallel Instruction Computing，并行指令计算）处理器架构绝对是一个严峻得多的挑战。为了支持诸如指令级并行处理（instruction-level parallelism）之类的功能，提高执行效率的大部分责任

都已经从处理器芯片转移到了系统工程师（他们可真了不起）肩上。要想提高执行效率，就必须预测计算机指令的执行情况并尽可能地让它们并行执行。这类预测必须满足两项要求：一要尽量避免因预测错误而降低执行效率，二要生成能够通过预测而提高执行效率的代码。因此，实现这类预测功能的工作量是非常大的。不夸张地说，这足以把任何一位工程师逼得跳楼。

更糟糕的是，计算机的处理器芯片和这些芯片的指令集总是处于变化和发展当中。当一个稳定、高效的编译器被推向市场并被接纳为一种标准化工具的时候，计算机硬件肯定又有了新的进展，这就迫使设计该编译器的工程师们不得不埋头于改进该编译器的后端（back end）。为了跟上硬件开发人员的步伐，像我这样的系统软件开发人员就不得不一刻不停地跑下去，根本没有喘息的机会——这是一种永远的痛。

在不断改进开发工具后端的办法之外并不是没有其他的道路。具体地说，你可以选择一台虚拟机（virtual machine）为目标。一台虚拟机与一台真实存在的计算机的不同之处在于前者只是一个技术规范。这类技术规范由一系列规则构成，而软件工程师可以采用任何他自己认为适当的手段来实现这些规则。这就使虚拟机能够做到与具体的计算机平台无关。对一台虚拟机来说，只要它能够遵从其技术规范里的各项规则，就可以存在于任何一种计算机平台上，就可以用任何一种计算机语言来编写。我就是受到了这种思路的启发才开始构造HEC执行引擎的。我的主要目标是建立一个稳定的目标机器，让系统工程师不必再每隔两年就不得不去重新编写他们的开发工具。我的另一个目标是想为中级水平的系统工程师提供一个既容易掌握又容易应用的运行时系统，就像70年代里DEC公司的PDP-11对广大程序员们的意义那样。

这本书对那些希望进一步了解计算机工作原理却又不打算钻研直接内存访问（direct memory access）或各种计时器（interval timer）繁复细节的学生们也很有用。不管某个系统基于哪一种硬件平台，程序在其上执行的基本机制都是完全一样的：指令从二级存储器被加载到内存并由处理器来执行它们。本书用了很多篇幅来解释这种机制。我希望这本书能够使学生们对程序执行机制有一个基本的了解，这样，当他们在今后遇到某个新系统时，就能够以此为框架来分析新系统。

此外，这本书对利用汇编语言进行的程序设计也做了大量坚实的论述。完全采用汇编语言来开发程序当然不是什么明智之举，但对汇编语言的深入了解能够帮助我们对问题的本质有一个更清晰的认识，这是其他程序设计方法做不到的。例如，有时候要想彻底弄清楚编译器（compiler）中的优化器（optimizer）的细节，惟一的办法就是对处理器执行机器编码指令的有关情况做细致的研究和分析。

“不要去管那个躲在窗帘后面的人……”——童话《Oz》里的巫师

在刚接触到Borland公司Turbo C编译器的时候，我对其底层工作情况的了解是十分差劲的。因此，我通常会在写好代码并运行编译器后闭上双眼并默默祈祷。等我觉得安全了，我才会睁开双眼查看其结果。后来，我看到了Turbo C的汇编代码清单，从那以后，我才对发生在幕后的事情有了一个更好的把握——有几次，我编写出来的代码甚至比编译器的优化器做得还好。

需要的预备知识

本书假设读者都能熟练使用C和C++编程语言。如果你不熟悉C和C++语言，但又希望向系

统工程师方向发展，我建议你尽快开始学习这两种程序设计语言。C语言是实现系统软件的首选语言。在各种程序设计语言里，C语言可以说是一个轻量级选手，但它的功能却十分强大，在提供了许多计算机底层操作访问手段的同时，又具备足够的抽象性，非常适用于软件移植工作。

C++是C语言的一个扩展，它的面向对象（object-oriented）技术使它能够对更复杂的问题进行描述和处理。C++是三种最优秀的面向对象的程序设计语言（Smalltalk、C++、Java）之一。我在本前言的末尾列出了几本参考书，希望它们对不甚了解C或C++语言的读者有所帮助。

对任何一位打算从事系统程序设计的人来说，掌握C语言将是一项必不可少的工作。这是因为UNIX操作系统从传统上讲都是用C语言实现的。UNIX的第一个版本是由Ken Thompson用汇编语言在一台PDP-7机器上实现的。当语句规模超过几千行之后，任何一个汇编程序都会变得难以维护。Ken能够坚持完成第一版UNIX操作系统体现了他做为一名程序员的毅力。但一个用汇编语言编写出来的操作系统难于移植，在认识到这一点之后，Ken与Dennis Ritchie和Brian Kernighan合作创造出了C语言。在1973年，UNIX内核在DEC公司新推出的PDP-11机器上重新用C语言写了一遍。如果你认为C语言是一种已经被比较现代的程序设计语言所超越了的“古董”，请再好好想想。看看Linux操作系统内核的源代码吧，它是免费的，在因特网上随处可见，但几乎全是用C语言写出来的。

本书的布局

本书对HEC的体系结构及其具体实现两方面的哲学动机都进行了论述。为此，我对它的设计问题进行了分解和分析。我相信，一幅图画抵得过千言万语；所以我会在我认为适当的地方给出一份图表或一张示意图。相关的源代码段落在书中也随处可见。

本书分为三大部分。

第一部分——概述 本书开始的两章内容是全书的基础。第1章对计算技术的发展史进行了回顾。在第1章里，我还给出了HEC虚拟机的设计目标，这些目标对HEC虚拟机本身做出了定义。在第2章里，我对HEC虚拟机的基本功能以及促使我做出各项具体决定的各种限制条件进行了简要的分析。

第二部分——HEC虚拟机 在第3和第4章里，我对HEC虚拟机及其调试器（debugger）的操作情况分别进行了解释。HEC虚拟机其实要比HEC汇编器（assembler）简单得多，所以这两章对后续的学习来说是一个很好的热身。第3章介绍的是HEC虚拟机的操作情况。第4章则对调试器进行了细致的分析。因为调试器是内嵌在虚拟机里的，所以这两章在内容上结合得非常紧密。

第三部分——HEC汇编语言 在本书最后的四章内容里，我对与HEC汇编器进行了介绍和讨论。第5章是对HEC汇编器本身的研究和分析。HEC到宿主操作系统的接口是用一组中断来提供的。第6章对这些中断依次进行了描述。HEC虚拟机汇编语言的使用方法在第7章里进行了介绍。第8章对如何使用HEC汇编语言来实现各种面向对象的结构提出了一些看法。

本书附带的光盘

软件工程可不是一项旁观者的运动，你必须亲自动手才能真正有所收获——至于下海之后

要游多远倒完全取决于你个人。如果你只想拥有并使用HEC虚拟机，本书附带的光盘里的二进制文件可以让你立刻开始工作。如果你想研究HEC虚拟机的源代码，本书附带的光盘里也包含所有二进制代码的源代码。

我现在居住于加利福尼亚州，只能利用紧缺的私人预算（即我的工资）开展工作。因此，我最初的虚拟机实现是在Windows上完成的。有些读者可能会对我的这一做法感到失望，我也很能理解他们的心情。我之所以会选择Windows，是因为我认为它比KDE或GNOME更容易使用。另一条路是购买Sun公司的硬件——可惜我又拿不出那么多钱来。但这并不意味着HEC只能老老实实地待在Windows上。要想移植这个运行时系统并不困难，我在第8章对此也进行了讨论。

反馈

金无足赤，人无完人。但这并不意味着我们不必去追求完美。对很多事情来说，从实践中学习是获得知识的最佳道路。事后诸葛最容易做，所以我的目标是准备好足够的代码让大家来做我的事后诸葛。

有一种古老的东方游戏叫做“围棋”，这种游戏非常复杂，必须经过多年的认真学习才能登堂入室。“先学输，再学赢”是人们告诫围棋初学者的一句基本格言，而这也将是我告诫软件工程师们的一句格言。当你还是一个新手时，也许会犯大量的错误。大多数经理都希望年轻的软件工程师能够在错误中进步。

我在构造HEC虚拟机的时候也遵循了这条忠告。从几年前开始，我就埋头于我的代码，力争把我能找到的缺陷都纠正过来。可如果你现在让我再来一次，我知道自己肯定还能在里面再找出一些缺陷来。但是，丑媳妇总要见公婆，你编写的代码迟早要拿出来给别人看。

如果你在这本书里发现了一个错误，请来封信告诉我。如果我有钱的话，我也很愿意学学Don Knuth的做法——他为自己书里的每一个错误设立了2.56美元的奖金（一条有用的建议可以得到32美分的奖励），他甚至建议读者可以通过查找书中错误的办法把买书的钱挣回来。可惜，加利福尼亚州高昂的生活费用使我一直处于一种捉襟见肘的境地（我不知道Don的日子是怎样过的）。我能给予大家最好的奖励是献上我的感谢，或者在本书的下一版里提到你的名字。请把你的修正、建议或者批评按下面的地址寄给我：

Bull Blunden
c/o Wordware Publishing, Inc.
2320 Los Rios Blvd., Suite 200
Plano, Texas 75074

参考资料

• Charles Andres。《CPU Wars》（CPU战争），1980年。http：//e-pix.com/ CPUWARS/cpuwars.html

这部喜剧连载对20世纪60年代的软件文化进行了有趣的描写。任何一位出生在1969年以后的人都应该看看这个连载，好了解一下自己在婴儿时期都错过了哪些东西。

• Intel公司。《IA-64 Architecture Software Developer's Manual, Volume 1: IA-64 Application

Architecture》（IA-64体系结构软件开发手册，第一卷：IA-64应用体系结构）。定购代码：245317-001，2000年1月。http：//www.intel.com。

这是Intel公司论述其即将推出的64位处理器的4卷/套丛书中的第1卷。这一卷的讨论重点是编译器设计方面的问题。在看完这一卷之后，你就会明白人们为什么会说为IA-64处理器设计编译器是一项极其复杂的工作了。

- Scott Maxwell。《Linux Kernel Commentary》（Linux内核分析）。The Coriolis Group出版，1999年。ISBN：1576104699。

这本书对Linux操作系统中使用的进程管理基本机制进行了深入的分析。书中用大量的图表论述了怎样才能利用C语言来构造出质量达到实际应用标准的操作系统。这可不是一本你一坐下就能看完的书。

- Dennis M. Ritchie。《The Development of the C Language》（C语言的开发历史）。Association for Computing Machinery（计算机器联合会），Second History of Programming Languages Conference（第二届程序设计语言历史大会），1993年4月。
- _____。《The Evolution of Unix Time-sharing System》（Unix分时系统的演变）。AT&T Bell Laboratories Technical Journal 63 No. 6 Part 2（AT&T贝尔实验室技术论文集，63类，第6卷，第2部分），1984年。第1577-1593页。
- Herbert Schildt。《C: The Complete Reference》（C语言参考大全）。Osborne McGraw-Hill出版，2000年。ISBN：0072121246。

这本书很适合那些想学习C语言却又没有多少编程经验的人们阅读。这是一本由一位善于解释复杂概念的作者写出的入门级读物。

- _____。《C++ from the Ground Up》（C++入门）。Osborne McGraw-Hill出版，1994年。ISBN：0078819695。

在读完Herbert关于C语言的书之后，你应该再继续学习一下这本书。

- Peter van de Linden。《Expert C Programming: Deep C Secrets》（C语言专家程序设计技术：C语言的秘密）。Prentice Hall出版，1994年。ISBN：0131774298。

这是一本非常有价值的书。Peter论述了许多相当深奥的问题，这些东西是大师和初学者的分水岭。

作者简介

自从在1983年第一次接触到DOS的调试工具开始，Bill Blunden就一直沉迷于系统软件之中。他不满足于仅仅知道事物的表面现象，而是一门心思地去探求它们的本质。凭着初生牛犊的勇气，他开始尝试编写8259中断控制器程序，结果是把他自己的电脑弄得一团糟。直到获得计算物理（mathematical physics）学士学位和操作研究（operation research）硕士学位之后，Bill的才华才逐渐显露出来。在美丽的Cleveland市的一家保险公司里担任保险统计师期间，Bill平生第一次与一台工作异常的IBM大型机进行了面对面的“搏斗”——那台机器里有一个COBOL程序编写得不正确。虽然“战斗”十分“惨烈”，但Bill最终胜利了，而他的兴趣也由此从钻研数字转移到了软件研发方面。随着时间的推移，Bill逐渐成为美国中西部地区一位知名的ERP专家——他用Java语言开发了一个CASE工具，进行了大量的技术研究，并与Control Data Corporation（控制数据公司，CDC）的许多专家进行了切磋。Bill有权随意支配一台配备有4块处理器和2GB RAM的机器，这使他能够在自己的象牙塔里开展各种研究。因为拥有如此之多的内存，所以曾经有一个时期，Bill习惯于关掉操作系统的内存分页（paging）功能，让程序直接运行在SDRAM上。但他小侄子的出生使Bill不得不从中西部迁居到西部的硅谷。Bill现居住在经常发生停电和地震的硅谷地区，并逐渐从自己与COBOL当初的“战斗”中恢复过来。

目　录

第三部分 HEC汇编语言

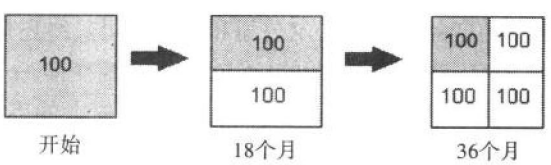
第一部分 概 述

第1章 历史与目标

1.1 历史回顾

Gordon Moore（戈登·摩尔）在1965年做出了这样一个预言：每隔18到24个月，能够容纳在一块芯片上的晶体管数量就会翻一倍，这就是有名的摩尔定律。后来的实际发展情况与戈登当时得出的结论的确相差无几。现在，人们已经习惯于利用摩尔定律来预测晶体管尺寸的发展方向了。

插曲 如果给定的处理器中的晶体管数量每隔18个月就增加一倍，就意味着晶体管的线性尺寸每三年就会缩小一半。在英特尔公司1989年推出的80486芯片中，晶体管的线性尺寸规模是1个微米（人的一根头发大约是100微米宽）。通过简单的数学计算（如图1-1所示）我们就能知道，再过40年，摩尔定律就会到达它的尽头。这是因为电子需要一条至少三个原子宽的通道，如果通道的宽度小于三个原子，量子力学的法则就会发生作用。也就是说，电子将不再像粒子那样活动，它们将开始像一个逃犯那样四处乱窜。电子不习惯于受到限制，如果人们给它们留出来的通道太窄，它们就会在“墙壁”上钻出“洞”来逃走——就像电影中的逃犯那样。既然IBM公司已经研制出一项能够利用碳元素的纳米通道来制作半导体的技术，所以人们很有可能很快就会制造出只有几个原子宽的晶体管来。也就是说，如果我足够幸运，就有可能在我的有生之年看到摩尔定律走到它的尽头。



如果CPU上的晶体管数量每隔18个月翻一倍，晶体管的线性长度就会每隔三年缩小一半。

1微米=10^{-6}米= 一个细菌的长度
1埃=10^{-10}米= 0.00001微米 = 一个水原子的直径

N(t) = $(0.5)^{t/3}$ = 晶体管的长度，以微米为单位，(t) 从1989年开始计算的年数
N(36) = 0.0002微米 = 2个埃（到2025年）

图1-1 摩尔定律

硬件不光是变得越来越小，它们还变得越来越便宜。随着功能廉价却足够强大的计算机的不断涌现，由传统意义上的大型主机所提供的集中式模型向分布式网络体系的转变正愈演愈烈。

这些事情是20世纪50年代的人们没有听说过的。在当时，人们估计的计算机市场容量大约是6台左右（我也不太清楚）。那个时候的计算机都是些巨大而又笨重的家伙，就像我们在老科幻电影中看到的那样。当时，做烟雾试验就是做烟雾试验——工程师们会接通机器的电源，然后仔细观察烟雾是从哪里冒出来的。

向分布式处理器转移的潮流实际是从1961年开始的，当时Digital Equipment Corporation（数字设备公司，DEC）推出了PDP-1计算机。这一事件标志着小型计算机（minicomputer）时代的来临。买不起大型主机（mainframe）的中规模公司可以购买小型计算机，而这些小型计算机在一个相对较小的范围内足以提供足够的性能。或者，即使某家公司已经拥有了一台大型主机，如果它想在不新添置一台大型主机的前提下提高吞吐量，就可以采用把大型主机的部分负担转移到各部门里的小型计算机上去的办法来解决问题。不管这些小型计算机的用途如何，与那些需要占据整个房间的大型主机相比，它们的成本只赶得上后者的一个零头。促成这种转变潮流的一个重要因素就是人们渴望能够更方便地使用计算机。

这种把计算处理工作转移到更为廉价的硬件上去的潮流在IBM公司于1981年推出其个人电脑之后变得更加迅猛。个人电脑（也叫做“microcomputer”——微型计算机或“微机”）的诞生导致了小型计算机的衰败，后人把这种现象称为“微型机杀手的进攻”。IBM的宣传口号是“没有人能躲过微型机杀手的进攻！”

我认识一位在大型主机还统治着地球的年代里在Unisys公司工作的工程师。他对自己所在部门制定的大型主机上机审批手续很反感，于是把自己编写的源代码转移到一台8088芯片的IBM PC上，他的大部分开发工作都是在那台微机上完成的。他的理由很简单：在自己的微机上工作，他能够控制整个事情，而这种控制是其他方法无法提供的。微机赋予人们更强大的能力。虽然当时的微机只有很少的RAM，其磁盘存储容量也只有几千字节，但坐在微机前面的人对这些东西却有着完全的控制权。对那些长期以来不得不看大型主机操作员脸色行事的工程师们来说，这绝对是一股新鲜的空气。

把PC用做服务器方面的主力是1996年微软公司推出其Windows NT 4.0以后的事情了。在1996年之前，英特尔公司的硬件骨架没有足够的软件肌肉来承担服务器方面的负载，而Windows NT 3.51又不足以成为一个成熟的产品。但英特尔公司和微软公司最终还是合作开发出了一个初级的企业级服务器。当时，已经有一些比较复杂的UNIX操作系统变体——如FreeBSD和Linux——能够在PC上运行了。可惜的是，Berkeley没有微软公司那样强大的市场推广能力，而Linux当时还没有完全成长起来。到了1997年初，大多数ERP（Enterprise Resource Planning，企业资源规划）解决方案的供应商已经或者开始着手把它们的应用软件移植到Windows NT上。人们开始把NT看做是企业级解决方案的一个替换性平台。

插曲 DEC公司在桌面电脑的兴起中再次扮演了一个间接的角色。当比尔·盖茨打算开始开发Windows NT时，他聘请了一位名叫Dave Culter的人，他在操作系统的设计领域很有名，他就来自DEC公司。

Windows操作系统家族的最新成员Windows XP系列发布于2001年10月，其中有一个版本是以英特尔公司的64位Itanium处理器为目标的。业内普遍认为Itanium将是英特尔公司的下一件大事。在传统意义上，高端服务器的市场一直是惠普（Hewlett Packard）、Sun微系统（Sun Microsystems）和IBM等几个巨头的天下。英特尔凭借着自己的32位处理器已经在服务器市场上抢到了一定的份额，而Itanium被认为将使英特尔有能力进入高端服务器市场并与其他64位体系结构进行竞争。但我个人对Itanium处理器的前景并不乐观，它那733到800MHz之间的时钟速度和它的高价格（它的起价预定为1177美元）极有可能使它对市场构不成任何冲击。

英特尔和微软的紧密合作形成了所谓的Wintel联盟。促使各公司的CIO们采用Wintel服务器的最强烈的因素是希望能够把总拥有成本降到最低。他们的构想是让大量价格低廉的Wintel服务器（即运行着Windows的Intel服务器）彼此合作以分担计算工作。这种技术叫做“集群”（clustering），而这类服务器集合就统称为“集群”（cluster）或“服务器农场”（server farm）。

集群技术是一种以相对不太复杂的手段来提供扩展性和可靠性的办法。当某台机器因为某种原因无法继续运转时，在该机器的维修期间，集群中的其他机器可以分担它的工作。如果数据流量开始下降，只要在集群中增加一些服务器就可以了。与采用大型主机构成的服务器农场相比，采用Wintel机器的解决方案成本相对比较低廉，所以在集群中增加新节点的工作不会有太大的困难。

图1-2给出了著名在线银行E*trade公司部署的服务器农场的结构，基于HTTP协议的因特网数据流量（即网上某个浏览器客户所产生的数据）先经过一堵防火墙的过滤，然后到达一个负载平衡设备。再由该负载平衡设备根据预定的轮转（round-robin）算法选中一台服务器去完成与那个浏览器的会话。由某个客户程序起始的事务将在一组应用服务器上执行它们的业务逻辑，当数据提交到数据库服务器阵列时，与该客户程序之间的事务就算完成了。E*trade公司的服务器集群中的大部分硬件都是由Sun微系统公司提供的，但它们的股市行情实际是由一台大型主机提供的。

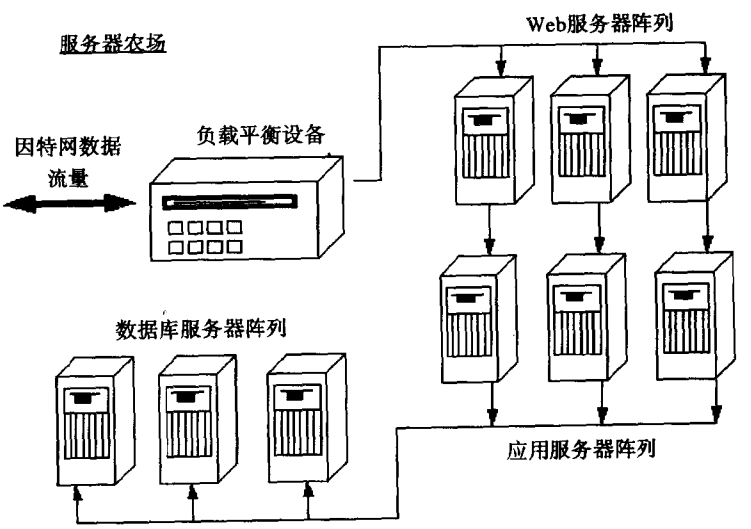


图1-2 服务器农场