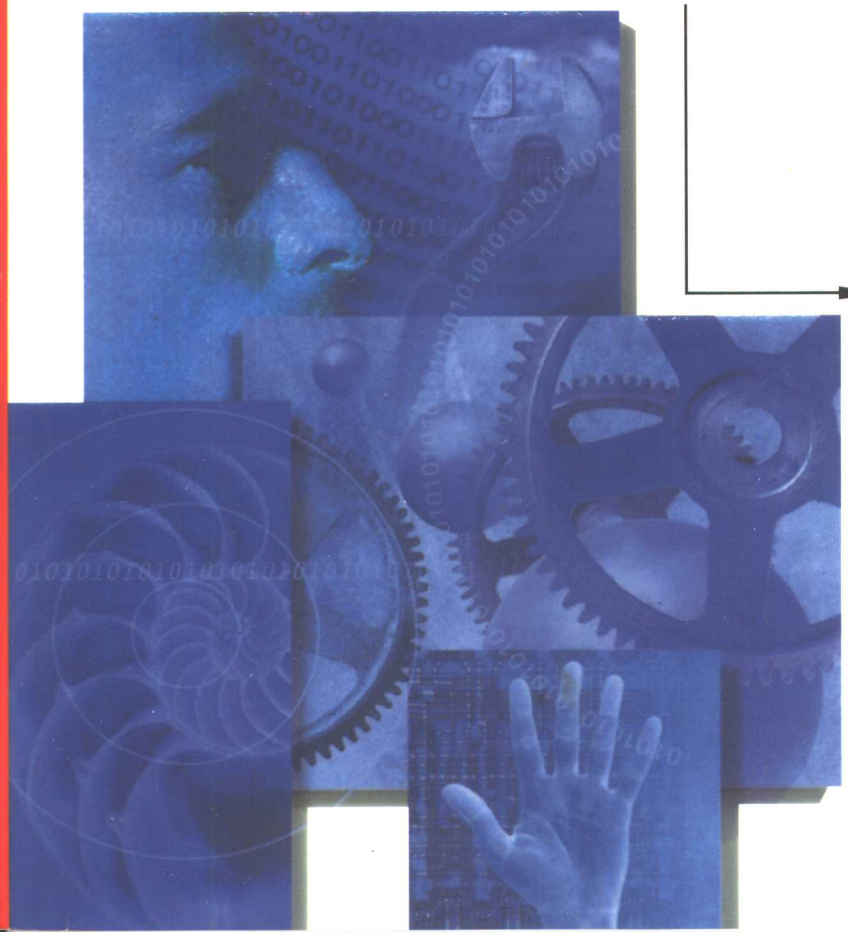
Microsoft 编程系列

Microsoft® Press

配光盘

应用程序调试技术

[美] John Robbins 著
潘文林 陈 武 译



查找并解决基于 Microsoft Windows 的应用程序编程错误的指南。

清华大学出版社
http://www.tup.tsinghua.edu.cn

Microsoft 编程系列

应用程序调试技术

[美]John Robbins 著

潘文林 陈 武 译

清 华 大 学 出 版 社

（京）新登字 158 号

内 容 简 介

《应用程序调试技术》以作者自己的多年编程和调试经验着重介绍了各种语言的程序调试工作。其内容包括程序错误和故障的类型、小组调试所需要的基础结构要求和在编写代码时如何进行预先的调试；什么是调试器，并描述了调试器的工作原理，作者还以相当深度讨论了 Visual C++调试器和 Visual Basic 调试器以使您能最大限度地发挥它们的功用；提供了一些在编写基于 Windows 的应用程序时遇到的常见的调试问题的解决方案。在本书的“附录”部分提供了您在进行调试冒险时会发现十分有用的一些附加信息。

对于倍受程序调试折磨的编程人员来说，本书将能成为您不可多得的良师益友。

书　　名：应用程序调试技术
作　　者：[美] John Robbins
译　　者：潘文林　陈武
责任编辑：夏兆彦
出 版 者：清华大学出版社(北京清华大学学研大厦，邮编：100084)
　　http://www.tup.tsinghua.edu.cn
印 刷 者：北京国马印刷厂
发 行 者：新华书店总店北京发行所
开　　本：787×960　1/16　印张：29.5　字数：641 千字
版　　次：2001 年 4 月第 1 版　2001 年 4 月第 1 次印刷
书　　号：ISBN 7-900631-59-3
印　　数：0001~5000
定　　价：65.00 元(含 1 张光盘)

前　言

如果按照通常的惯例，我应该联系有关程序中充满了错误与故障的故事并提醒大家有错误与故障的应用程序是社会麻烦的根源，以此来开始前言。然后就滔滔不绝的列出从调查报告中得出的有关每 100 LOC（代码行）中就有多少错误与故障的平均统计数。这些都是正统的主题，但是我认为读者们已经知道了消除错误与故障的重要性才来读这本书。这样，我想在这本书中需要完成三项任务：

- 说明黑客 John Robbins 是谁
- 说明为什么你需要读这本书
- 讲述有关 John Robbins 的经典故事

第一次与 John 相遇是在 CompuServe 论坛上。当时他正致力一个国产调试器的研究工作，并提出了一个关于 Microsoft Win32 Debugging API 的问题。他开玩笑地补充说，如果他完成了这个国产调试器，也许就能够在 NuMega 公司找到一份工作。我留意到了这一消息并给了他答复。我还顺便提及 NuMega 公司正在物色人员（难得的精通调试的人才）。

现在开始讲述 John 兴冲冲地到 NuMega 公司面试的故事。他身着制服（这也许是他唯一的一套制服），并带着一叠精心格式化了的磁盘，磁盘里是他完成的调试器的代码样本。面试时，NuMega 公司的一名领导睡着了（或者是 John 以为他睡着了。而事实上那位领导非常专心致志的在听）。John 以为自己就这样失去了这份工作的机会，但是事实上，他给所有人都留下了深刻的印像。John 被录用的几个月之后，他发现我使用的咖啡杯垫正是他的代码样本磁盘。事实上，我从与 John 的交谈中就发现了他是调试方面的中坚人才，我无须对他的代码进行检验。

那时，NuMega 公司的规模还很小，完全处在致力于为开发人员取得得力的调试工具的阶段。John 为这场改革作出了贡献。他是那么急切地想要投入到 NuMega 公司的工作中，所以他马上就从佛吉尼亚搬迁到了 New Hampshire 来，在他所找到的第一处租居，即消防站后面的一所房子里住下来。每次当消防站的大门呼啸着打开时，John 的 PC 机就会断电。在工作中，John 接受的第一项任务就是编写 BugBench，这是演示 BoundsChecker 能够找到的所有错误的一个实用程序。在那段时间里，我们都善意地称 John 为世界上错误与故障最多地程序设计师。

几个月后，John 的 PC 机开始差不多每隔一小时就会冻结大约 10 秒钟。无论他对网络管理员说多么尖锐刻薄的话都无济于事。最终，John 熬不了 Windows NT 的 SoftICE

（NuMega 公司的内核调试器）的一个很早的版本。没过多久，John 就查出了原来是有个工程师使用了一个具有时间判断力的线程，在随机间隔的时间内将总计达 175M 的 Windows NT 业务，放置到 John 的机器上。即使他竭尽全力，也无法战胜这样的恶作剧。那时 NuMega 公司的大部分工作就是派工程师去参加贸易展览。John 的外向个性使他成为销售部瞄准的一个天然目标，销售部急切地想让他也成为销售代表。哈，还有一些更无知的想法颇具讽刺意味，是想让 John 作为“亚特兰大错误与故障控制中心的研究员”而闪闪发光。John 丝毫不想参与这个闹剧，于是他和另外一个工程师提出了一种完全不同的想法， 这种想法就是举世闻名的“高年级男孩和低年级男孩”幽默剧的起源。John 是“高年级男孩”，这是他在 NuMega 公司得到的许多外号之一。我将“Sporkie 男孩”的起源留给读者来想像吧。

作为贸易展览的一个脚注，正是从这里 John 明白了那些表面看起来充满沙子的重力救援气球里面真正装了些什么。在这个特殊的例子里，填充物看起来好像是些胡桃壳的小碎片。当 John 将气球举过头顶时，他设法刺破了气球，结果弄得我们浑身都是胡桃壳，不仅如此，整个旅馆房间里也到处都是。这些胡桃壳甚至还掉到了地毯和床罩中。

最初 John 在 NuMega 公司的一半时间都用来完成 BoundsChecker 各个方面的工作。他的工作包括开发调试器循环体、编写 API 确认占位程序、修正符号表的错误与故障、埋头于 COM 界面的引用计数算法，以及将流程序跟踪信息的代码扩展到磁盘上。一天之内要干这么多事情！对调试工具进行调试，尤其是那些对操作系统没有严格限制的工具，这些工作提供了大量磨练调试技能的机会。

我和 John 在遇到困难的问题时经常不能使用调试器来解决。我还记得有个很特别的问题，在发布一个很重要的测试版的前夜，是我在 Microsoft Visual SourceSafe 上运行 BoundsChecker 时遇到的。开始看起来运行状况很好，但是几分钟以后，另一位工程师发现 Visual SourceSafe 中的数据文件看起来像“@@1?70”。常规的备份当时还不属于工程进程的一部分，所以最后几乎完全毁坏了保存着源代码的唯一拷贝的版本控制数据库。毋庸多言，当时整个公司都差点要剥了我的皮，但是我和 John 费尽心机，最终发现 Visual SourceSafe 使用了一些传统的 MS-DOS 代码，破坏了我们的文件句柄。

你可能会不明白，“讲这些故事干吗”？我的意思是 John 并不是学院式或抽像的理论空谈家。John 是个十足的实干家。我和 John 经常悲叹，有那么多编程方面的书都是由极其缺乏或根本没有商业开发经验，并且只编写过代码不超过 100 行的程序的人编写的。John 是几个获奖产品的主要开发者。在编写软件时，他是在传达自己从得之不易的经验中所领悟的信息。

当 John 决定要做什么事时，他就会全力以赴。他会对某个问题紧抓不放，直到完全将它解决。在 John 的字典里，没有“我无法解决”这样的话。在 John 开始他的编程生涯之前，曾主动提出帮我做我所负责的《Microsoft Systems Journal》(MSJ)杂志中一个专栏的几个函数。我认为他在代码上所花的时间甚至比我还要多。由于我们都在为 MSJ 的栏

目写东西，所以就经常交流各自在为栏目内容编写相应的代码时所遇到的令人头痛的问题。很多次听他描述他为探索一个问题而穷追不舍的情形都令我目瞪口呆。

要写一本书不仅需要了解与主题相关的知识，还需要了解更多别的知识。并且还要有一种希望与读者有效地交流这些知识的愿望。在 John 的第一篇文章里，我就注意到了他那简单明了的语言表达方式。我同时还认为阅读 John Robbins 所写的任何东西都是一种享受。当我知道文章的作者真正了解文章所谈的主题，也很希望能与读者进行交流，并且不会使用毫无必要的晦涩复杂的表达时，阅读这样的文章无疑是令人愉快的。

John 的写作热情远在他的这些文章、这一栏目以及这本书之外。John 在他所编写的所有代码上都留下了自己的印记，下面这段文字就摘自我在工作中收到的典型的有关 John 电子邮件：

John 在程序当中东一块西一块的废话讨论到底有什么用？每一条该死的函数标题前都有一整章的关于社会学的暗示，即如何使用例程以及如果不正确地使用他们会如何地使行星错位的暗示——上个月的 MSJ 几乎都是一本蠢书，整本书都充斥这种讨论的废话——在此书中，他几乎整本地谈论他的 Form Load 处理程序。这个人是一名作家还是工程师？天哪……

正如你将在本书中看到的那样，问题的答案就是两者都是！

调试问题被一层神秘的面纱覆盖着，在许多方面都只是一个口头惯例而已。我们急需好的书籍。仅有的几本书多数都只是集中讨论无错误与故障的情况，而没有深入到 Windows 调试过程中可能会遭遇到的困境。书籍发行人不时地催促我写一本调试方面的书，但我都婉言拒绝了。这并不是因为我对调试方面的知识了解得不够，而是因为我觉得很难将自己在调试方面的直接经验付诸于文字。而 John 却不存在这种问题。

从我的观点来看，做到有效的调试其实非常简单。我通常将之归于两条原则：

- 了解可能会出现的问题
- 懂得如何使用工具来查明究竟出了什么问题

如果你能够做到这两点，调试就很简单了。问题就在于，上述两点都不是那么容易就能做到的。当我说“了解可能会出现的问题”时，这并不是说你所了解的知识只停留在源代码的水平上。你不仅要了解大概的内容，还要熟知所有的细枝末节。编译器会将代码转换成什么？在 API 的调用过程中将会发生什么情况？如果不能回答这些问题，你就是在盲目地进行调试工作。

我也并不是说要进行调试就要通晓所有的指令，而是说，在需求不断提高时，你应该能够将问题分解成越来越小的部分，就像 John 在本书第 1 章中所介绍的那样。你甚至还要深究不是由你控制的那些部分，找出程序崩溃的原因。如果挖掘的深度足够，你最终肯定会找到原因所在。跟踪错误与故障的许多工作都需要你了解语言、操作系统和 CPU 的错综复杂之处，这也是 John 在第 1 章中谈到的另一个话题。书中讨论你至少应该了解的 x86 汇编语言的有关部分的那一章（第 6 章），对读者们来说也是极有价值的。

同样地，知道如何有效地使用工具也很关键。调试工具已经变得相当成熟了，很多开发人员还从未有机会（或者有机会也不用）去熟悉这些工具的所有性能。我想起一位朋友，他手下有一位年轻的图像处理程序设计师为他工作。当这位神童的代码发生了程序崩溃时，他竟然不知道调试器能够显示出调用堆栈。

很多开发人员只要了解了足够的调试技能，能够脱离当前的困境，他们就满足了。很多用户常常会忽视调试工具的一些重要功能，因为需要使用这些功能的情形并没有在恰当的上下文中出现。如果一项功能一项功能地通读文件，到后来就会感到厌烦，最终你只能够了解到一些最基本的功能。John 所写的“使用 Microsoft Visual C++调试器进行的调试”和“使用 Microsoft Visual Basic 调试器进行强有力的调试”这两章（第 5 章和第 7 章），将会使你彻底摆脱“一步、一步、一步，然后检查。然后又一步、一步、一步地调试”的沉闷状况。

我说调试是一项口头惯例，是指多数人只有在他们听说别人使用某种技能或技术解决了某个问题时，才会学习这种新的技能或技术。当你在精通调试工具的人旁边观看他工作时，只需观看半个小时，就能学到 10 倍于自己看一天技术资料所学到的东西。John 正是这样一位精通调试的人。他了解他所使用的工具有哪些优点和缺点。同时，他还预先投入时间来编写对症的诊断代码，而不是在出现问题之后才来补偿损失。将调试当作一项技术而不是什么繁琐之事来不断地提高自己的调试技术，这样，你也可以作到这一点。本书是一座丰富的金矿，即 John 关于调试的金科玉律，会帮助你更快地掌握调试技术。

Matt Pietrek
Hollis，New Hampshire
1999 年 12 月

绪　论

错误与故障影响系统安全与性能，并且周期性地出现。错误与故障是致使项目进展不利的原因，它会使项目不能按时提交，使开发者加班加点地工作并且会使同事产生不满的情绪。如果有很多的错误与故障蔓延到所开发的软件中，无疑会使你的生活变得很悲惨，因为用户再也不会使用你的产品，那么你就会失业。错误与故障是不容忽视的问题。

我们这个行业的人常常只是将错误与故障描述成令人讨厌的东西。事实远远还不止于此。所有工程师都能指出一些含有无法控制的错误与故障的项目，甚至不乏有这样的例子，即有些公司发布的产品中满是错误与故障，根本无法使用，公司也因此而倒闭。就在我写这本书的时候，美国国家航空和宇航局（NASA）因为在要求分析和设计阶段有错误与故障侵入而丢失了一个火星空间探测器。随着计算机开始控制越来越多的重要的任务系统、医疗设备和价格高昂的硬件，我们再也不能对错误与故障投以讥笑，把它当作开发过程中出现的一般现象而等闲视之。

我希望本书所讨论的内容将能够帮助你首先掌握如何编写应用程序才能使错误与故障更少；其次才是掌握在需要调试的时候，如何才能更迅速地进行调试。多数开发组没有意识到这一点，他们平均要花费开发周期一半的时间来进行调试工作。如果开始调试方法正确，可以大幅度地减少所需的时间，这意味着可以更早地提交产品。在需求收集和设计阶段无法走捷径，但是学会更加敏捷地调试软件无疑是可能的。这本书将调试作为一个整体的方法来讲。我不是将调试作为一个独立的步骤来看，而是把它看作整个产品周期中不可分割的一部分。我认为必须在需求收集阶段就开始调试工作，一直持续到将最终版本投入生产为止。

有两个问题使得在 Microsoft Windows 环境下进行调试比较困难并耗时。首先一个问题是调试是一种自学而来的技能，即基本上要靠自己去领悟。即使你有计算机学科的学位，我敢打赌你也从未上过任何一门专门讲授调试的课程。除了一些深奥的学科，比如为那些没有人使用过的语言设计自动程序校验，或者为原本就开放式的、整体的并行处理计算机开发调试器等这样的学科，调试这门学科应用于商业软件时，在教育机构中似乎并不受欢迎。有些教授指出首先要学会不去编写错误与故障。尽管这是一个很好的要求，是我们都应该努力争取的理想状态，但是现实却有些不同。学习系统知识、检验调试技术并不能够保证你永远不再编写错误与故障，但是遵循这样的实践经验却能够限制你在代码中可能添加的错误与故障的数目，同时，在确实出现了错误与故障时，可以使

你更快地追查到这些不经意造成的错误与故障。

第二个问题是，虽然有很多专门的关于 Windows 技术的好书可供利用，却没有一本在讨论到调试技术时能够深入到足够有用的程度。若想有效地调试所有技术，必须要了解的知识比一本集中讨论某项专门的技术的书所能提供的要多得多。了解如何编写 Microsoft Internet Explorer 的 ActiveX 控件是一回事，而能够调试 ActiveX 控件则完全又是另一回事。若要调试 ActiveX 控件，必须要了解 ActiveX 和组件对象模型(COM)的所有细节，动态链接库（DLL）如何映像到内存中以及 COM 如何进行查找并创建控件。有的书将一些成熟的功能的实施写得看起来很简单，比如象远程数据库链接使用当今最热门的技术就可以解决，但是，当 db.Connect（‘Foo’)在 Microsoft Visual Basic 程序中运行失败的时候(它终将会失败)，你只有靠自己去查找并修补技术链上的断裂链接。不仅如此，尽管有几本关于项目管理方面的书也的确谈到了调试问题，但是它们往往是侧重于管理和监督方面的问题而不是侧重于开发者所关心的问题。这些书可能包含了关于如何制定调试计划的真知灼见，可是当你盯着在回调函数的返回过程中所发生的系统崩溃时，它们却帮不了你什么忙。

本书的精髓，是我作为开发人员和管理人员为了能按时提交高质量的产品所经历的尝试及磨练的经验总结。这几年来，我已经掌握了处理开发和管理这两方面问题的能力和技术，这两方面的问题使开发基于 Windows 的应用程序成为一种挑战。为了解决第一个问题，即缺乏正规的调试方面的训练的问题，我编写了本书的第一部分，给读者们上一堂关于调试的应急课程，不过这是一门带着明显的商业开发倾向的调试课。关于第二个问题，即需要一本专门讨论在 Windows 环境下如何进行调试的书这一问题，我想我已经提供了这样的一本书，它架起一座连接专门技术和本质的、现实生活的调试技术之间的桥梁。

在过去的年中，我能够有机会几乎专门地致力于调试工作，实在是非常幸运。一些实际的工作经验帮助我树立起了对于调试问题的独特看法。最早的工作经验是在 NuMega Technologies（即现在的 Compuware NuMega）公司，在那儿我是首批工程师之一。我有幸能够进入编写自动错误检测工具（BoundsChecker）、性能工具（TrueTime）、代码覆盖工具（TrueTime）以及调试器（SoftICE）的顶尖小组中工作。我们所开发的产品属于这样的类型，即能够使用户在遇到错误与故障时就如同见到血腥谋杀事件时那样尖叫起来，这“鼓舞”着我们要尽自己所能地去编写和提交最优质的产品。

除了在 NuMega 公司作软件工程师和管理者的工作经验外，我还荣幸地为《Microsoft Systems Journal》杂志的 Bugslayer 栏目写文章。在这个栏目中，我集中讨论关于调试的问题，并经常与开发我们所能想象到的各种类型应用程序的软件工程师们交流，我从中所学到的东西甚至比直到今天提交产品时，开发该产品的工程师们所面对过的问题还要多得多。

哪些人应该阅读本书？

本书写给那些厌烦了整天深夜里还忙于调试工作，希望能够提高他们所编写的代码的质量以及公司产品的质量的人。同时，本书还写给那些希望开发出效率和效力更高的开发组的那些管理者和开发组领导者们。

从技术的角度来看，本书的“理想读者”是对 Microsoft Visual C++、Microsoft Visual Basic 或这两者有过 1 至 3 年的使用经验的人。同时，我还希望读者曾经是真正的开发组的成员，并至少提交过一种产品。软件行业将具有这种工作经验水平的软件开发人员称为“中级软件开发者”，不过对这一术语我并不在乎。

高级软件开发者或许也一样能从中学到很多东西。我非常有幸能够有一群了不起的朋友，他们是我的“评论组”，在将本书交付 Microsoft Press 之前，他们对其中的内容提出了很多批评意见，是他们教会我如何进行开发和调试的。他们真正是这个行业中最优秀的软件开发工程师。他们告诉我，读了我的书，他们也从中学到很多东西，这令我无比高兴。

对 Visual Basic 开发者的说明

如果很快地浏览一下本书，你会注意到我着重强调了 Visual C++调试器、C 语言和 C++语言以及汇编语言。强调这些有两个原因，首先，C 和 C++语言比 Visual Basic 更能提供更多的机会给开发者，进行自下向上的开发。其次，由于 Visual Basic 调试器无法对用自身语言编译的二进制代码进行调试，所以你需要了解 C++调试器，用它来调试 Visual Basic 应用程序。虽然根据粗略浏览一遍的印像，你也许会认为本书没有讨论 Visual Basic，但是不要这样断然地下结论，当你开始使用 Visual Basic 进行真正的开发工作时，就会发现几乎整本书对你都适用。调试就是调试，无论开发环境是怎样的。

如何阅读本书？

我将本书分为三个独立的部分。应该按顺序阅读前两个部分（第 1 章到第 7 章），因为我是以逻辑前进的顺序安排本书内容的。

在本书的第 I 部分，即在“调试的概论”（从第 1 章到第 3 章）一部分中，我详细说明了各种不同类型的错误与故障，并逐步展开了一个调试过程，所有优秀的调试者都会遵循这个过程。同时，我还讨论了进行专门的小组调试所必需的基础结构要求问题。最后，因为可以（而且也应该）在代码编写阶段作大量的调试工作，所以我谈到了在编写代码时如何进行预先的调试。

在本书的第 II 部分，即在“高效率的调试”（从第 4 章到第 7 章）一部分中，我解释了什么是调试器，并描述了各种调试器的工作原理。对工具了解的越多，你就越能将其运用自如。我还相当深入地讨论了 Visual C++调试器和 Visual Basic 调试器，这样，你可以最大限度的发挥它们的功用。我与这一行业中具有不同水平的多位工程师（从没有经验者到经验丰富者）共事，从中了解到他们都只使用了 Visual C++调试器和 Visual Basic 调试器所有功能中很小的一部分。虽然这种观点从编写关于调试的书的作者口中说出来，显得很奇怪，但是，我希望能让读者尽可能地摆脱调试器的束缚。当你读这本书的时候，你会发现我写本书的主要目标不是告诉你如何修改程序崩溃，而是如何首先避免出现程序崩溃。但是，我还想教会读者如何发挥这些调试器的最大效力，因为有时你还是不得不使用它们。

在本书的第 III 部分，即在“功能强大的工具和技术”（从第 8 章到第 15 章）一部分中，我提供了一些在编写基于 Windows 的应用程序时会遇到的常见的调试问题的解决方案。我讨论的主题所覆盖的范围从查找只具有崩溃地址的源文件和代码行号、如何自动测试用户界面，到追查出多线程程序死锁的位置等这些方面。这一部分的多数章节中都包括有你能从中学习的实用程序和大量的源代码。

本书的最后一部分，即“附录”（附录 A 和附录 B）部分，提供了你在进行调试冒险时会发现十分有用的一些附加信息。在附录 A 中，我对如何阅读和解释 Dr. Watson 日志进行了说明。在附录 B 中，你将看到资料的注释清单，包括参考书目、工具和 Web 站点等，这些都曾经帮助我磨练了作为开发者和调试者的技能。

你会发现在整本书中，我在文章中用了黑方框来突出说明常见的调试问题和调试实战故事。虽然我乐意回答别人提出的常见的调试问题，但是真正的乐趣更在于听别人讲实战故事并将它们联系起来。我希望通过将一些我曾经参与解决的（并且有些还是我参与编写的）“上好”的错误与故障拿来与读者一起分享，你将会看到我所介绍的方法和技术的实际运用。另外，我希望能够帮助读者避免我所犯过的错误。

系统要求

使用这本书，需要有以下这些条件：

- Microsoft Windows 2000、Microsoft Windows NT 4 Service Pack 3 或更高的版本，或者是 Microsoft Windows 98
- Microsoft Visual Studio 6.0 Service Pack 3 或更高的版本
- DBGHELP.DLL（若要了解有关这个文件的更多信息，请参阅绪论中稍后提到的“务必阅读！关于 DBGHELP.DLL 符号引擎”一节）

本书配套 CD 的内容

我敢保证本书所附的源代码比你买过的任何一本书所附的都要多。除了使用 Microsoft Foundation Class(MFC)库实现图形用户界面（GUI）调试器的例子外，在附赠 CD 中还包括有 8 个完整的实用程序，使您能够更快更有效的进行调试。单单是源文件就总计有 2.5 兆字节。多数代码已在超过 50 个的真正的应用程序上进行过实战测试，其中有一台以使用 Windows 98 作为嵌入操作系统的钢琴。

配套 CD 中的 README.TXT 文件包含了与项目有关的所有特殊连编和执行问题。所有项目我都是使用 Microsoft Visual Studio 6 Service Pack 3 开发并测试的。同时，我还使用了 Windows 2000 RC2 Platform SDK 标题来连编所有项目，以确保读者们能够使用。

我用 Windows 98、Windows NT 4 Service Pack 4 以及 Windows 2000 RC3I（这是我完成本书时发布的最新版本）测试了所有代码。由于 Windows 98 已问世多年，所以我决定不再支持 Windows 95。不过，我并没有使用任何 Windows 98 特有的操作系统功能，所以如果你需要在 Windows 95 上运行本书附赠 CD 中的代码，应该不会有太多困难。

务必阅读！关于 DBGHELP. DLL 符号引擎

在几个实用程序中，我使用了 Windows 2000 DBGHELP.DLL 符号引擎。虽然该符号引擎过去通常是在 IMAGEHLP.DLL 文件中，现在却被放置到了 DBGHELP.DLL 文件中，这是为了将它从内核操作系统 DLL 中移出。必须用 DBGHELP.LIB 和 DBGHELP.H 来连编本书中的源代码。附赠 CD 里的是我写本书时的最新 Platform SDK 版本。

应该安装该版本的 Platform SDK，不仅是因为它有许多极好的调试实用程序，而且你还会发现 DBGHELP.LIB 和 DBGHELP.H 正是其中的一部分。安装之前，应该先对 MSDN Universal Subscription CD 或 msdn.microsoft.com/developer/sdk 进行仔细检查，才能得到最新版本。

为了使符号引擎能在 Windows NT 4 和 Windows 98 上运行，必须从 Windows 2000 或 <Platform SDK>\Redist\DbgHelp 目录下将 DBGHELP.DLL 文件拷贝到与使用该文件的程序相同的目录下。例如，如果想使用 CrashFinder 程序来查看出现崩溃的源文件和代码行号，需要将 DBGHELP.DLL 文件放到与 CRASHFINDER.EXE 文件相同的目录下。

为了使 DBGHELP.DLL 符号引擎找到你的符号，需要将 PDB 文件放在与你构建二进制码的目录相同的目录下，或者将 PDB 文件放在与二进制码所在的同一个目录下。另一种方法，则可以使用专门的环境变量来告诉该符号引擎 PDB 文件所在的位置。若要了解关于 SymInitialize API 函数的更多信息，请参阅 MSDN 文档资料。

作好准备，因为要使 DBGHELP.DLL 符号引擎进入运行状态，是得花些时间研究的。

判断 DBGHELP.DLL 符号引擎是否运行的惟一办法是，你是否能够从使用 PDB 调试符号编译的文件中获取函数信息或源文件和代码行号的信息。如果 DBGHELP.DLL 符号引擎无法正常运行的话，这正是你对我在本书中所介绍的调试技能进行尝试的一个绝好机会！在我的 Web 站点 www.jprobbins.com 上的 Frequently Asked Questions(FAQ)一部分中，保留着有关 DBGHELP.DLL 符号引擎最新最棒的信息。通过访问 FAQ 并且进行少许的调试工作，你就会获得自己的符号。

反馈信息

我非常希望能够了解读者朋友们对本书的看法。如果有什么问题或有自己的调试实战故事，我非常愿意洗耳恭听！请将它们发到 www.jprobbins.com 站点，我乐意回答朋友们提出的任何合理的问题。我非常愿意看到你的问题，因为当我努力回答这些问题时，我也会从中获益匪浅。尽管我很幸运，能够有机会从事很多的开发工作，但是这还远远不够（仍然不够！）。你的问题正给了我大好机会来了解你在最新最好的技术方面的得与失。

多谢惠阅，并祝调试成功！

John Robbins
1999 年 12 月
Hollis, New Hampshire
Bob in 2020

目 录

第 II 部分 高效率的调试