真正的Windows程序员都应该拥有这本书

Windows 95

用软件组件编写
难以置信的
Windows 95
应用程序

用户控件的程序设计

[美] Paul Cilwa 著
杨开成 罗秀川 译
邢文华 校

人民邮电出版社

Windows 95
用户控件的程序设计

[美] Paul Cilwa 著
杨开成 罗秀川 译
邢文华 校

人民邮电出版社

Windows 95 Programming with Custom Controls

Windows 95
用户控件的程序设计

◆著 [美]Paul Cilwa
译 杨开成 罗秀川
校 邢文华
责任编辑 周雁飞

◆人民邮电出版社出版发行 北京崇文区夕照寺街14号
北京顺义振华印刷厂印刷
新华书店总店北京发行所经销

◆开本：787×1092 1/16
印张：23
字数：571 千字 1997年10月第1版
印数：1－6 000 册 1997年10月北京第1次印刷
图字01-96-1613号 ISBN7-115-06589-6/TP·173

定价：34.00元

译 者 序

Windows 程序员都知道，虽然 Windows 标准控件提供给用户许多标准、方便的交互手段，但对程序员来说，编写出漂亮、友好的应用程序界面只依靠那些标准控件是远远不够的。按照传统的做法，您可以按自己的意图将各种标准控件定义成具有定制风格的控件以满足应用程序的需要。但这种一切都从头开发的做法除给程序员带来单调冗长的工作外没有任何价值。长期从事这种开发工作的程序员不会再感受到创作软件的快乐。

现在不同了，软件组件的时代到了！

您可以根据需要先购买或开发那些可用的软件组件，再在“软件建造厂”中创造你的软件。Visual Basic 和 Deiphi 等是“软件建造厂”典型代表。那么当买不到您所需要的组件时，您又如何开发这种组件呢?这就是这本书的教学任务。

VBX 和 OCX 是两种通用的软件组件。这本书的主要内容就是讲解如何编写您自己的组件，而且，您从中还可以学习到如何编写一个代码生成器。本书的 VBX Genie 就是一个很好的例子。此外，您如果想要顺利地阅读完本书，至少应熟悉下面两种程序开发环境：Visual Basic 和 Visual C++。只会使用一种环境的程序员是不适合读这本书的，但愿您不是。

本书共有四个部分。第一部分简单介绍了 VBX 和 OCX 的基本概念；第二部分阐述了如何创建一个“与软件商无关”的 Windows GDI 类库，这个类库是为编写 VBX 服务的；第三部分详细讲解了 VBX 的代码生成器 VBX Genie 的编写；第四部分告诉您如何编写 OCX 组件。其中第一、第二以及第四部分由杨开成译；第三部分由罗秀川译；全书由邢文华校。由于译者水平有限，如有错误不当之处，请读者见谅。

序　言

中国有句古老的祝福："祝你天天快乐"；从某种意义上讲，这句话在我们身上应验了，因为过去的日子很少有现在这样有趣过。作为程序员，我们更是过着有趣的生活。因为我们的工作就是改变这个世界，使得我们的生活越发有趣。而且我们本身也在改变，那些我们用来改变世界的工具也同样在改变，有时它们会以我们难以跟上的速度在变化。

VBX(Visual Basic Extension 的缩写)就是一个这样的例子。VBX 的前身是 Visual Basic，现在已被许多应用程序开发环境和工具所使用，包括 Microsofe Visual C++，Borand C++，Delphi，PowerBuilder 等等。然而，VBX 虽然只有四年的历史，但已经被认为是过时了，正被 OCX(OLE Control Extension)所代替。OCX 标准正在迅速地发展之中。

以前，您可能指望您写的库函数过几年还会有用；现在，如果它们在几个月后仍然有用，您就很幸运了。早在1994年，我在 Coriolis Group(出版商)出版了一本叫做《使用用户自定义控件进行 Windows 程序设计》的书。这本书教 C/C++程序员如何使用 Windows 标准控件在单个动态连接库(DLL)中建造组合的标准控件和 VBX。这本书也介绍了一种有用的 VBX 框架，它允许程序员将注意力集中于自定义控件特定的要求上，而不必为那些每个 VBX 都需要的繁杂工作而浪费时间和精力。

在我写那本书的时候，VBX 的标准，特别是它们被不同编译平台(如 Visual C++，Borland C++等等)所支持的方式还处于不断变化之中。幸运的是，我收到了许多来自世界各地的程序员很有价值的反馈。他们正使用那本书所提供的代码和技术开发他们自己的用户自定义控件(以后简称"用户控件")和应用程序。

随着 OCX 和新开发平台(包括 Visual Basic 4，Visual C++ 4，和32位 Delphi 2)的出现，我认为应该进一步扩展创建软件组件的概念和阐述如何利用新的控件技术来开发 Windows 应用程序。

软件组件——下一代软件

当今您所遇到的大多数应用软件，特别是使用流行的可视(visual)开发环境开发出的软件都可能是在用户控件的帮助下开发出来的。我敢肯定的，您已经意识到了，用户控件正把成千上万的程序员带进了利用用户组件(custom component)开发软件的艺术殿堂。

什么人需要这本书

如果您是这样一类人：您宁愿在业余时间去科罗拉多大峡谷划橡皮筏或在科罗拉多河上用假蝇作饵钓鱼或去喜玛拉雅山脉做长途旅行，也不愿意再编写另一个 Windows 应用程序开发工具，那么，这本书就是为您写的。我的目标是通过教您如何创建自己的可重用软件组件来帮助您节省时间。为了能顺利阅读这本书，您需要具备一些用 C 和 C++编写 Windows 程序

的经验。本书呈现的用户控件代码是用这两种语言写的。此外，介绍一些 Visual Basic 对您是有帮助的，因为它目前仍然是 VBX 风格软件组件的主要使用者，并且也是第一个使用 OCX 的开发环境。

您将需要什么

为了编译一个 VBX，您必须有一个能生成16位代码的 C/C++编译器。如果您是微软 Visual C++用户，那么您可以用1.52或更早版本的编译器。后期版本不能用，因为2.0版以及更高版本的编译器只能产生32位代码。另一方面，Borland 公司的 C/C++编译器，只要能生成 Windows 代码的都可以。Borland C++4.0或4.2版有一些严重的错误(bugs)，如果您拥用这些版本，您应该要求升级版本。

要运行或测试一个 VBX，您需要 Visual Basic 3.0或 Visual Basic 4.0专业版的16位版本(其它版不含有16位版本)。其它使用 VBX 的平台实际上只能运行1.0版的 VBX。如果您能接受这个限制，您也可以选择它们。

为了在 Microsoft 环境中创建 OCX，您需要安装 Control Wizard。这个软件与 Visual C++同在一张光盘上，但它需要单独的安装。在 Borland C++环境中创建 OCX 需要 Borland C++5.0。

注意，就在我编写这本书的时候，并非所有的平台都使用了 OCX。运行或测试一个 OCX 需要 Visual Basic 4.0或 Visual C++4.0或 Borland C++5.0或32位的 Delphi。

我曾努力使这本书能与尽量多的工具配合使用。但坦率地说，由于两个主要软件商(Borland 和 Microsoft)每经几个月就升级软件——Borland 公司在去年就单独发行了五个版本的 C++环境——我是不可能跟得上的。所以，我没有提供读者如何编译的菜谱式指导，而是力图解释清楚诸如“为什么要设置各种编译开关”等等这方面的问题。因此，您可以根据自己的开发工具来作出明智的决定。

如何使用这本书

本书不是那种包含关于处理所有程序设计问题的资料；但它也不只是一个创建 VBX 或 OCX 的手册(尽管它提供了这方面的信息)。它试图描述1996年组件式软件世界的发展变化，而且不但要帮助您过渡到32位程序代码的编写，还要使您学会处理将来会遇到的这种过渡。

VBX 和 OCX 只是一个借助手段，您要学习的是如何编写紧凑、强壮、真正可重用的组件风格的代码(component-style code)。在这本书中您将要学习面向对象的代码编写技术(可以用 C，也可以用 C++)、如何使 C++重载运算符真正为您所用、如何最大限度地利用新的 Visual Basic 类以及如何设计多用途的 DLL。

如果您只懂一种计算机语言，并且想一直这样下去，这本书就不适合您。而且当今的时代也不适合您。目前，快速开发应用程序的最佳方法是：用 C 或 C++或两者都用来创建工具；再用诸如 Visual Basic 或 Deiphi 这些开发平台作为粘合剂将那些工具捆绑在一起。尽管可以用一个程序员来编写工具；用另一个程序员来使用“粘合剂”，但如果您两者都会做，您的身价就会更高；您的薪水将会告诉您这一点。

本书力图做到读起来很风趣，并且书中的代码也力图做到使您在编写和运行这些代码时

感到很有意思。通常，我会随着代码告诉您一些编程和设计的建议。这些建议是基于我15年来的专业编程和设计经验，它们主要是关于 Windows 的。我的读者和学生普遍认为这些经验是很有用的。

好!现在您可以坐到转椅上，启动计算机，开始阅读吧。如果您在运行 Winsows 95，您应该在启动计算机前花一点时间浏览一下第一章的内容。

本书的内容

本书分为四个主要部分。在第一部分，我们将简要阐述什么是 VBX，如何建造一个 VBX 以及为什么您会需要 VBX。接下来我们以同样的方式来介绍 OCX。在结束第一部分以前，我们简单回顾一下前一本书所谈到的 VBX 框架。

下一部分，我们将开辟一个新的天地：为 VBX 创建 C++类库。为了使它用途更广，这个库必须既不依赖于 Borland 开发环境，也不依赖于 Microsoft 开发环境。我称之为“与软件商无关”的类库。为了创建这样一个类库，我们必须先为 Windows GDI 建一个小的类库。当您做完这部分工作时，您就会感到创建一个有用的类库是多么容易。这时您可以会放下手头的工作，做一个自己的类库玩玩。

第三部分，我们要转换一下所使用的语言。您是不是觉得为重复的编码任务而手工复制代码很讨厌呢?编写一个代码生成器是很容易的，而且也蛮有意思的。用这部分的工程所生成的工具来编写它会更有趣。这个工程就是 VBX Genie，它能自动生成我们前面开发出的代码。

下一部分我们将进一步来学习 OCX。它与 VBX 有什么区别?又有哪些相同之处?在我们学习 OCX 时，我们会先创建两个 OCX。您会发现 OCX 控件非常有用。

走您自己的路

中国还有句俗话：“千里之行始于足下”。我相信您只要一步一个脚印，就一定能学好。所以赶快开始第一章吧，就会很快完成这个特别旅行的。穿上舒适的鞋子!这会很好玩的。

目　录

第一部分　组件式软件的奥秘

第一章　软件组件的威力

1869年，John Wesley Powell 领导了第一次科罗拉多大峡谷飘流探险。这是真正的首次探险，因为没有证据表明当地的美国人曾架舟穿过科罗拉多大峡谷(尽管他们肯定在那里生活过)。所以当 Powell 和他的队员踏上旅程时，他们正迈向人们未曾去过的地方。

Powell 是带领九名队员和四艘重木船出发的。Powell 非常肯定在路上会遇到急流，而那些宽底船则是他们通过急流险段的最大希望。尽管如此，飘流结束时，Powell 丧失了两艘船和四名队员。(掉队的四名队员中，有三名至今下落不明)这次探险用了他三个多月的时间。去年，我花了17天顺大峡谷飘流而下，只是为了好玩。同 Powell 一样，我们开进 Lee' Ferry 渡口，从 Meade 湖离 Powell 出发地不远的地方开跋。路上我们有美味的食物、舒适的睡眠并且有很多时间上岸攀岩或到 Colorado 和 Deer Creek 支流玩跳水。

这是怎么回事呢?

一句话，递增式发展的结果。第一步是 Powell 证明了漂流是可以做到的。

很奇怪，有些事情如果前人已经做过了，而当你再去做它时，会很有安全感的。

下一步是二战期间充气橡皮筏的发展。(这一步也是建立在前面几步的基础上，比如硫化橡胶和充气泵的发明)。

再下一步，也是天才的一步，发生在传奇式人物 Georgie White 把出售战争中剩余橡皮筏的生意和她对大峡谷的热爱结合起来的时候。她首次开创了乘坐橡皮筏飘流科罗拉多河的旅游业。乘客的旅行路线以前已经有人走过了，但是他们是乘坐木制的平底小渔船完成旅行的。这种小船在大的波浪中很容易翻船。而 Georgie 的橡皮筏只会在波浪中吓唬吓唬乘客，乘客们会浑身湿透，感到惊心动魄，但却安然无恙。

1.1　可重用代码的诞生

当70岁的 Georgie White 仍在做着探险旅游业生意的时候，我正忙于在 Tandem 主机(当时我们称它们为 '小玩意')上工作，使用一种专门语言 TAL 编程。我很快地意识到我可能是在一遍又一遍地编写着许多相同的代码。由于 TAL 是块结构(block-structured)语言(类似 C 语言)，所以我试着将代码放到集中那些完成某一种功能的小函数和过程中。这样这些函数和过程就很容易被重新使用了。我还做了其他程序员很少做的事情：把我的库全部文档化。一年过后，我积累了400多个函数，涉及的范围包括从把二进制值转换到 ASC Ⅱ 码的小函数到能执行全部功能的“报告书写器”的一组过程。

我经常会使上司和同事感到惊奇，我会在一个下午做完一个月的工作。我能做到这一点是

因为我不再去编太多的程序，只是简单地用我以前写的函数来‘组装’新的程序。四年多过去了，我一直在使用着那个库。

1.2 进入 VBX

可现在有了 Visual Basic。VB 的一部分魅力(也是它成功的最重要的原因)是它的用户控件 VBX。这些控件允许我们程序员扩展程序设计环境。VBX 做到了 Windows 标准控件从未努力去做的事情：建立了生产第三方的、可重用的、可单独发行的程序组件的小作坊式软件工业。软件组件的时代到来了。

第三方 VBX 市场尤如科罗拉多河的十级急流席卷了整个软件开发行业。成百上千的公司投入其中，迅速出现了上千种控件。使用现成控件进行软件开发已经成为了流行的时尚。大量控件等待着程序员们去选择，包括数据库存取、多媒体控制、网络和 Internet 通讯、动画和图形、用户界面工具、文字处理机、文本编辑器、拼写检查器和实时控制器等等，甚至包括专家系统那样的组件。有关 Visual Basic 和其它可视语言的杂志、书籍和学术研讨会迅速涌现出大量文章，纷纷强调用 VBX 建造的 Windows 应用程序更具灵活性、功能更强、可大量节省时间等方面的优点。

但是，没过多久 Microsoft 公司就声称 VBX 已过时了。Microsoft 公司就好象一个发脾气的孩子，拿到了他要的东西，可马上又说那不是他/她最终想要的。尽管 Microsoft 公司自己的产品 Visual Basic 正在使用3.0版的 VBX，但它的另一个程序设计环境 Visual C++却只使用1.0版的 VBX。在我写这本书的时候，Visual C++的32位版本根本就不使用 VBX。

与那些90年代初兴起的软件开发工具和技术相比，VBX 更是担负着对 Windows 软件开发前景的变革。为什么这么说呢?因为 VBX 第一次实现了软件组件的实际应用。这些软件组件可以被嵌入开发平台中，用于建造任何一种应用程序。

Microsoft 公司认为 VBX 标准的缺点是它不能用在32位的环境中。但你要明白，这个缺点是软件厂商人为造成的。没有理由不可以把软件调度层设计成能将32位参数转换成16位参数。毕竟，Microsoft 公司的其它 Windows 代码是这样设计的，允许其它类型的16位 DLL 在诸如 Windows 95和 Windows NT 那样的32位操作系统中正常运行。

1.3 OCX 出现了

Microsoft 公司一个最新的观点是所有的东西都是 OLE(对象连接与嵌入，Object Linking and Embedding)世界的一部分。虽然 VBX 可能要被迫顺应这种观点，但 Microsoft 公司已决定放弃它，而用 OLE 控件来代替。这些 OLE 控件通常被称为 OCX。

在下一章我会多讲一些关于 OCX 的知识，并且本书的后两部分有关于 OCX 的更详细阐述。现在你只需知道 OCX 在应用于软件组件的功能上与 VBX 是等价的，但它需要的系统资源是 VBX 的两倍到十倍，却没有为程序员提供更多的价值。

1.4 现代的控件怪物

作为一个 Windows 用户——您肯定是，对吗?——您一定很熟悉控件。它们就是那些你在

处理数据时所指向、点击和要拖动的东西。从前，这些特征曾属于卡片分类机或打印机上的硬件控件。现在，他们只是屏幕上的一些图，但他们可以对键盘和鼠标做出反应。我们喜欢使用这些控件。

作为一个 Windows 程序员，您可能熟悉控件的内部机制：每个控件都是一个 Windows 窗口，这个窗口有自己的窗口处理函数，可以接收来自其它窗口的消息和向其它窗口发送消息，当用户指向、点击或拖动它时，它会将用户的操作通知给父窗口。作为一个 VBX 程序员，你可能会意识到在标准 Windows 窗口上面还另外有一个层次将控件挂到 Visual Basic、Delphi、PowerBuilder 或 Borland C++开发环境内。这个层次不但支持工具箱(toolbox)中各种控件的图形显示，而且支持以“属性”和“事件”来命名控件的功能以及将对属性和事件的引用转换成向底层控件发送的消息。

OCX 标准也是建立在 Windows 窗口基础上的。尽管在许多方面它重复了 VBX 的功能，但它实现的方式是如此的复杂和深奥，以至于在 Microsoft 公司将那些复杂的东西用 C++类库隐藏起来之前，是没有人能够弄得懂它或让它正常运行。事实上，第一部有关 OLE2.0程序设计的书的作者已经承认虽然他在写书，但他的确弄不太懂。

然而，Micorsoft 类库确实使我们能够编写 OCX。去向神灵祈祷吧，千万别发现那些只有深入 Microsoft 源代码才能确定的错误！

写得好的用户控件有一个优点，就是它把要完成的功能全部封装起来。为了更清楚地说明这一点，我们把用户控件比做电视机。你知道电视机内部充满了元器件、电路板以及发热的和带电的东西。你当然不会愚蠢地打开后盖，在里面焊点什么东西。你也知道你不必那样做。所有与你有关的控制器都被放置在机器的前部、后部或边上，或放在遥控器里。

好的用户控件也应提供给程序员清晰的流水线式外观。程序员可以而且也应该能控制一些东西(如属性、事件和方法)，但那些程序员并不需要接触的东西都应隐藏在控件内部，不能让程序员见到。

使用这个模型，你可以通过用户控件传递任何有用的东西。控件并不一定非是终端用户使用或看到的，有时它只是为了方便程序员而已。比如，Visual Basic 的计时器控件在运行时是不可见的，它只是为了方便程序员而简单地将计时器函数封装起来。

1.5 封装控件

你来封装一个标准 Windows 控件吧，要确保 Windows 消息是与它进行通讯的唯一方式。老兄，那里可没有全程变量！

当在标准 Windows 控件基础上建造一个 VBX 或 OCX 时，你要为控件定义你自己的属性(对 VBX 和 OCX 来说)和方法(只对 OCX 来说)；当那些属性和方法实际运行时，你要把适当的 Windows 消息传递给底层的 Windows 窗口。你还要定义一些事件(event)，当底层窗口向 VBX 或 OCX 层(底层窗口的父窗口)发送通知消息时这些事件会被激活。

我想，标准 Windwos 窗口类型的用户控件一直没有过时的主要原因，其一是以前 Windows 程序是用 C 写的；其二是 C 程序员都有从不使用别人代码的怪癖。他们不喜欢封装的控件；他们更喜欢不带机箱、显像管没有固定在底座上的电视机。但当他们发现他们已被 Visual Basic 程序员淘汰时，他们就放弃了原来的习惯。或者说，旧风格 C 程序员在达尔文适者生存的竞争中就这么灭绝了。当今的程序员必须精通好几种语言(恐怕很少有人不是这样)，并且能使

用多种工具来开发应用软件。他或她首先尽可能地寻找各种可用的组件，当找不到适用的组件时，又能创建新的组件。并且只有他或她找不到合适的组件时，才去开发新的组件。

1.6 我们何去何从？

既然这样，你要做的是什么呢？是编写据说只有几天活头儿的 VBX 呢？还是编写目前只有屈指可数的几种环境中才能使用的 OCX 呢？还是放弃编程，去大峡谷当一名导游？如果我必须选择，我选择去当导游。然而，这里还有一种选择，第四个选择：VBX＋OCX。

完全可能编写出双倍 OCX 功能的 VBX。

只有你真正理解了 VBX 和 OCX 的编程要求并且你擅长编写结构化风格的程序，第四个选择才是可行的。这种结构化的编程风格要求程序员能编写出小型的、高聚焦（一个函数只完成一种功能）的函数。接下来让我们来看看为什么 VBX 还会生存下去（这使得我们值得去做第四个选择）和为什么 VBX 给 OCX 编程提供了一个坚实的基础。

1.7 VBX 还没有消亡的十大原因

除了写书和划船，我还从事 Visual Basic，VisualC＋＋和 BorlandC＋＋的教学工作。因为这三个编程环境都声称支持 VBX，所以班上的学生经常问“我一定要学习编写 VBX 吗？”、“VBX 还没过时吗？”

之所以有谣言说 VBX 已经过时或即将过时了，主要是因为 VBX 是按16位代码标准编写的，而现行的 Windows 版本（比如 Windows NT）是按32位标准运行的。由于 VBX 运行在16位代码段中，人们担心它不能适应新的环境。Microsoft 公司不想把 VBX 标准扩展到32位而是用另一种新的标准 OCX 来代替它的做法更加重了人们的这种担心。

所以，开发者到底应怎样做？是继续编写随时都会过时的工具呢？还是再次抛弃以前编写和购买的工具呢？我们到底要这样做多少次呢？看起来好象无论什么时候，只要我们准备贮备可重用代码，Microsoft 公司总是给我们来个釜底抽薪。所以，我们程序员总是生活在无措手足的状态中就不足为怪了。

但别灰心，事情并不象看起来那样糟糕。尽管你很快就会想学习编写 OCX（可能你在读完这本书后就会这样做），但这并不意味你在开发或使用 VBX 过程中所获得的技巧就毫无用处了。这里有一些理由足以说明“VBX 已消亡”的观点是被人夸大了。

#1：现在你的客户需要应用软件

我们很容易忽略这样一件小事情：按最初的计划，Windows 95应至少提前一年发行。正如你看到的，当 Microsoft 公司放弃使用所有原来的工具时，他们也遇到麻烦。可你的客户却不打算扔掉他们买的任何东西——至少现在不想立即这样做。一些客户仍在使用 Windows 3.1。使用 Visual Basic 3.0来开发软件将会持续一段时间。Microsoft 公司也清楚这一点，所以它的软件都设计成是向后兼容的，以允许这种情况的存在。

但现在你的客户是有需求的。并非我们每一个人都能等到神话的降临：我们用来开发应用程序的平台达到了尽善尽美、登峰造极、不需要再更新版本的程度。你想想看，这一天是不可能到来的。推出新版本软件的源动力不是对尽善尽美的追求，而是更高利润的追求。只要用户

愿意升级他们的操作系统，操作系统软件商就愿意提供升级。事情就这么简单。那么，你用什么工具来快速开发应用程序呢?还跟以前的一样用 Visual Basic、Visual C++、Borland C++、Borland Delphi?软件组件会帮助你进一步加快开发速度。你一定会想到它：VBX。

现在，如果你要编写 OCX，你可以把它嵌入到 VB4.0的应用软件中；你也可以在 Visual C++4.0和32位 Delphi 应用中使用它。但你现在知道如何编写一个 OCX 吗?不知道?我不这么看。是的，你最终是要掌握如何编写 OCX，并且这本书也会帮助你达到这个目标。尽管如此，VBX 仍会给你一些帮助。你在编写 VBX 过程中所付出的努力是不会白费的，因为：

#2：Windows 95 支持现存的16位代码

记住我先前所说的：只要人们愿意买，就会有新版本的操作系统被推出。如果新版本的操作系统迫使用户不得不放弃原有的应用软件，用户就不会买它。那些应用软件所含的价值太高了，不可能一下子被替换掉。(实际上，我的客户已经开始抱怨他们不得不频繁地更替应用软件了。比如，他们认为如果 Microsoft Office 新版本增加了工具条，但这些工具条并不意味系统功能的改进的话，花钱去升级这个软件是很不值得的。)所以，就象 Windows3.1完全支持 Windows3.0的应用程序一样，Windows 95不得不支持 Windosw 3.1的应用程序。那些应用程序有很多是用 Visual Basic 或 C 写的，并且简单地重新编译并不能将它们升级到32位。既然如此，我们还能做些什么呢?

要想明白这种局面是怎样形成的，你必须知道一点有关 IBM 兼容机的心脏部件 Intel 80x86芯片的知识。(PowerPC 芯片当运行与 Intel 兼容的操作系统时会模拟 Intel 芯片，所以我在这里所讲的也同样适用于 PowerPC)

你可能知道，DOS 程序都是在实模式(real mode)下运行的。“实”是指程序中的寻址是真正的存取地址。也就是说，地址0000：0000就是 RAM 的起始地址。地址0000：0001是下一个字节，等等以此类推(见图1.1)。由于实模式被设计成在某一时间只能运行一个程序，因此没有必要提供防止向另一个程序的内存空间进行意外写入的保护。当指针被破坏或程序员忘记初始化指针时通常会出现这种意外写入的情况。

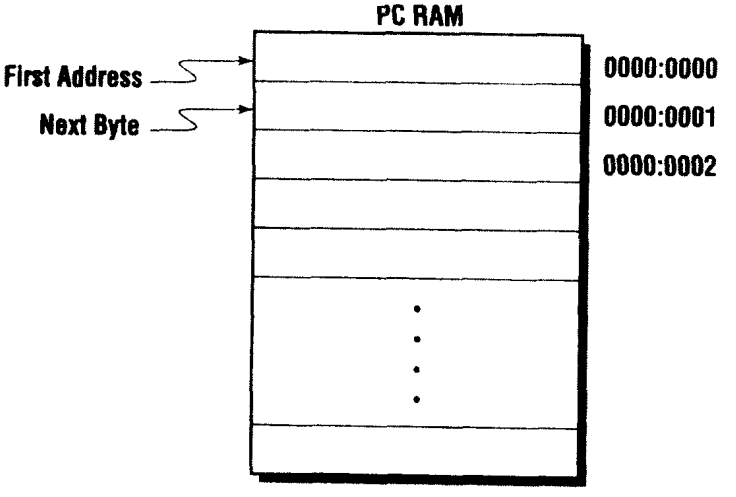

图1.1　在实模式下存取 PC 机内存

在保护模式下存在一个保护层。应用程序申请的每一块内存都由一个描述符来定义。描述符的索引被保存在32位的地址中最重要的13个位上。如图1.2所示，程序只能间接地使用内存。你不用担心，一切都会顺利运行的，因为 CPU 自己能以芯片层的速度自动管理这些描述符。