清华电脑学堂

Visual Basic .NET 程序设计 实践教程

纪多辙 刘万军 李白萍 等编著

- 总结了作者长期教学经验，难易适中，实用性强
- 深入剖析了 Visual Basic.NET 的技术要点和难点
- 围绕丰富的案例讲解，代码规范清晰
- 精心编写了大量“实验指导”，引导学生深入练习编程实践
- 课后提供丰富的习题，巩固学习成果
- 网站提供代码下载和课件支持

清华大学出版社

清華 电脑学堂

Visual Basic .NET 程序设计 实践教程

■ 纪多辙 刘万军 李白萍 等编著

清华大学出版社
北 京

内 容 简 介

Visual Basic.NET 是 Windows 环境下简单、易学、高效的编程语言，其快速开发的特性深受程序员的喜爱。本书介绍了使用 Microsoft Visual Basic.NET 和 Visual Studio.NET 集成开发环境开发面向对象应用程序的知识，包括.NET Framework 的工作原理（公共语言运行时和类库）、程序集与中间语言、面向对象的编程基础、使用 VB.NET 类进行编程的知识、VB.NET 的 GUI 设计、VB.NET 的图形和文件处理、使用 VB.NET 访问数据库等，本书开发部署了一个 VB.NET 范例应用程序，包括对项目进行案例分析、部署 Windows 和 Web 三层应用程序等内容。

本书可以作为读者学习 VB.NET 语言和面向对象开发的教程，适合作为普通高校计算机专业和非计算机专业的程序设计教程，也可供自学读者使用。

图书在版编目(CIP)数据

Visual Basic .NET 程序设计实践教程 / 纪多辙等编著. —北京：清华大学出版社，2006.8
ISBN 7-302-13415-4

Ⅰ. V…　Ⅱ. 纪…　Ⅲ. BASIC 语言－程序设计－教材　Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2006）第 079698 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社 总 机：010-62770175　**客户服务**：010-62776969
组稿编辑：夏兆彦
文稿编辑：王冰飞
印 刷 者：北京市清华园胶印厂
装 订 者：三河市新茂装订有限公司
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印张**：29　**字数**：721 千字
版　　次：2006 年 8 月第 1 版　2006 年 8 月第1 次印刷
书　　号：ISBN 7-302-13415-4/TP・8425
印　　数：1～4000
定　　价：39.80 元

前言

FOREWORD

Visual Basic 是 Windows 环境下简单、易学、高效的编程语言，其快速开发的特性深受程序员的喜爱。Visual Basic.NET 是微软公司基于.NET 框架重新设计的语言，新增和加强了许多新的面向对象的特性，例如引入了继承、接口和重载等特性。此外，还增加了结构化异常处理、托管代码和符合通用语言运行时（CLS）等特性，大大提高了 VB 应用程序的稳定性和可伸缩性，可用于大型应用程序的开发，完成复杂而艰巨的任务。这些新的特性使得 Visual Basic .NET 再次成为程序员关注的焦点。

本书内容

本书介绍了使用 Microsoft Visual Basic.NET 和 Visual Studio.NET 集成开发环境开发面向对象应用程序的知识，包括.NET Framework 与 VS.NET IDE 的关系、.NET Framework 的工作原理（公共语言运行时和类库）、程序集与中间语言、VS.NET 的开发环境、面向对象的编程基础。

此外，本书还介绍了使用 VB.NET 类进行编程、使用类接口进行编程、VB.NET 的 GUI 设计、在 VB.NET 中创建多文档窗口应用程序、VB.NET 的图形和文件处理、创建.NET 控件、使用 VB.NET 访问数据库等，以及在 VB.NET 中处理多个数据表的知识。

本书第 13 章介绍了 VB.NET 在 ASP.NET 中的应用，内容包括创建一个 Web 应用程序、开发 Web 页面、创建 Web 控件、使用 Web 数据绑定和 Web 服务等内容。第 14 章开发部署了一个 VB.NET 范例应用程序，包括对“图书管理系统”项目进行案例分析、部署 Windows 和 Web 三层应用程序等内容。

本书特色

本书通过实例介绍 VB.NET 程序开发知识，具有实用性教程的特色。

- 本书使用 VB.NET 语言开发了“图书管理系统”，读者可以从中学习使用 UML 建模、三层设计等逼真的开发过程。
- 介绍了 UML 语言的应用知识，UML 是面向对象开发过程中描述示例的系统要求语言，在真实的开发过程中非常重要。
- 强调了三层设计思想，本书内容和实例严格按照图形用户接口类、问题域类和数据访问类三层的思想划分和实现。
- 各章编写了大量“实验指导”，引导读者应用该章知识独立练习编程项目。
- 思考与练习题可以帮助学生检查对 VB.NET 开发理论知识的掌握程度。

本书读者对象

本书围绕范例介绍了 VB.NET 程序开发过程的完整知识，突出了面向对象实现方式，

读者可以从本书中得到完美的编程体验。本书可以作为普通高校计算机相关专业的VB.NET 编程初级教程，也可以作为接触过 VB.NET 基础知识，需要深入学习面向对象开发的中级教程。

编写过程难免会有错误，欢迎读者与我们联系，帮助我们改正提高。

编　者

2006 年 4 月

目 录

CONTENTS

第 1 章　Visual Studio.NET 的核心概念

随着 Internet 的飞速发展，软件开发的难度也在逐步加大，现在的开发平台和开发环境，不论是从开发技术还是从开发模式上，都无法满足 Internet 时代的基于 Web 的应用程序和 Web 服务的开发需要。在这种环境下，Microsoft 公司推出了.NET 开发平台。该平台使得 Windows 上 Web 应用程序的开发更为容易。

本章将大致介绍一些.NET 的设计原理，以及 Visual Studio.NET 元素之间的相互关系，让用户去理解 Visual Studio.NET 编译器的内部原理和 Visual Studio.NET 应用程序的组成元素。最后将使用诸如记事本之类的文本编辑器创建一个简单的应用程序，然后从命令提示窗口编译和执行该程序。

本章学习要点：

- 了解 VS.NET 与.NET Framework
- 了解公共语言运行时的作用
- 命名空间
- 创建 VB.NET 应用程序

1.1 VB.NET 与 Visual Studio.NET 简介

随着 Microsoft.NET 平台的发布，Visual Basic.NET 开始成为一种完全面向对象的语言，简称为 VB.NET。它是基于.NET Framework 的，其设计目的是为了快速而简洁地开发包括 Web 服务和 ASP.NET Web 应用程序在内的.NET Framework 程序。VB.NET 的主要特点有：

- VB.NET 有两种新的窗体方式——Windows 窗体和 Web 窗体。
- VB.NET 可以通过新的 ADO.NET 访问离线的数据源。
- VB.NET 具备了面向对象的所有特征，包括类、接口、封装、继承和多态性等。
- VB.NET 支持结构化异常处理和多线程。
- VB.NET 支持与其他.NET Framework 语言的集成。

开发 VB.NET 需要 Visual Studio.NET 集成开发环境。这样创建大多数应用程序方便快捷。但是也可以不使用 Visual Studio.NET 集成开发环境创建一些简单的应用程序，而完全利用记事本等这样的文本编辑器创建部分或者全部的.NET 应用程序，然后通过命令提示窗口调用合适的编译器来编译应用程序。

Visual Studio.NET IDE 与以前的版本相比有了明显的改进，它提供了许多新的功能。而且支持的语言的语法和实现方式也有所改进。下面列出了 Visual Studio.NET IDE 的一些新的功能。

- 所有的 Visual Studio.NET 语言都共享相同的集成开发环境。当安装 Visual

Studio.NET 时，默认有 3 种语言（VB .NET、C++和 C#）与.NET Framework 一起安装。根据用户所使用的 Visual Studio.NET 版本的不同，Visual J#默认时也可能会安装。用户可以自己选择安装何种语言，也可以选择安装特定的工具。

- 帮助系统有了很大的改变。大多数索引主题通过命名空间和命名空间中定义的类型组织。
- Visual Studio.NET 引入了一些新的控件。有一组控件能够访问诸如向系统消息日志写数据的 Windows 服务，其他控件能够通过 ADO.NET 访问数据库。
- Visual Studio.NET 允许创建不同类型的应用程序。例如，可以创建 ASP.NET 和 ASP.NET Web 服务应用程序。ASP.NET 是 ASP 的升级，ASP.NET Web 服务是 Visual Studio.NET 的新功能，它提供了在 Internet 上创建分布式应用程序的通用模型。Visual Studio.NET 还允许创建控制台应用程序和作为桌面服务运行的应用程序。桌面服务是一种在 Windows 启动时自动启动的应用程序，无须用户干预即可响应请求。
- Visual Studio.NET 依赖于可扩展标记语言（eXtensible Markup Language，XML）来通过 Web 保存、发送和接收数据，并且在应用程序之间通信。
- Microsoft 公司添加了一些新的调试窗口和调试工具，使得错误检测更加容易。
- Visual Studio.NET 开发工具为开发者提供了让用户在其目标计算机上运行的安装程序的自定义方式。用户可以选择安装部分应用程序，并且自定义应用程序在计算机上的安装方式。

1.2 .NET Framework 概述

.NET Framework 是支持生成、运行下一代应用程序和 XML Web 服务的内部 Windows 组件，是 Visual Studio.NET 应用程序开发环境的核心。它定义了语言之间互操作的规则，以及如何把应用程序编辑为可执行代码，它还负责管理任何 Visual Studio.NET 语言创建的应用程序的执行。.NET Framework 旨在实现下列目标：

- 提供一个一致的面向对象的编程环境，而无论对象代码在本地存储和执行，还是在本地执行但在 Internet 上分布，或者在远程执行。
- 提供一个将软件部署和版本控制冲突最小化的代码执行环境。
- 提供一个可提高代码（包括由未知的或不完全受任的第三方创建的代码）执行安全性的代码执行环境。
- 提供一个可消除脚本环境或解释环境性能问题的代码执行环境。
- 使开发人员的经验在面对类型大不相同的应用程序（如基于 Windows 的应用程序和基于 Web 的应用程序）时保持一致。
- 按照工业标准生成所有通信，以确保基于.NET Framework 的代码可与任何其他代码集成。

.NET 开发平台是为简化在第三代 Internet 分布式环境下的应用程序开发，基于开发互联网标准和协议之上，实现异构语言和平台高度交互性，而构建的新一代的通信平台。

.NET 开发平台使得开发者创建运行在 IIS（Internet Information Server）Web 服务器

上的 Web 应用程序更为容易，也使创建稳定、可靠而又安全的 Windows 桌面应用程序更加容易。.NET 开发平台如图 1-1 所示。

.NET 开发平台包括.NET Framework 和.NET 开发者工具等组成部分。.NET Framework 是整个开发平台的基础，包括两个主要组件：公共语言运行时（Common Language Runtime，CLR）和.NET Framework 类库（FCL）。.NET 开发者工具包括 Visual Studio.NET 集成开发环境和.NET 编程语言。其中 Visual Studio.NET 集成开发环境用来开发、测试和部署应用程序。.NET 编程语言包括 VB.NET、Visual C++和 Visual C#等用来创建运行在 CLR 下的应用程序。

如果想要开发和运行.NET 应用程序，就必须安装.NET Framework。.NET Framework 包含把.NET 应用程序转换为可执行文件所需要的所有编译器。开发者可以在文本编辑器中创建应用程序，不必使用 Visual Studio.NET。

Visual Studio.NET 包含了创建窗体和管理 XML 文档的可视化设计器，以及管理大型多文件工程，并把这些工程编译为可执行文件的工具。Visual Studio.NET 还包含部署应用程序以及把.NET Framework 部署到其他计算机的工具。尽管不使用 Visual Studio.NET 也可以开发出复杂的应用程序，但是使用 Visual Studio.NET 无疑会更高效。.NET Framework 和 Visual Studio.NET 之间的关系如图 1-2 所示。

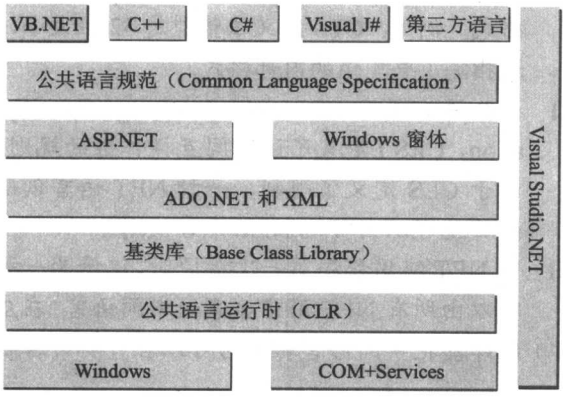

图 1-1 .NET 开发平台

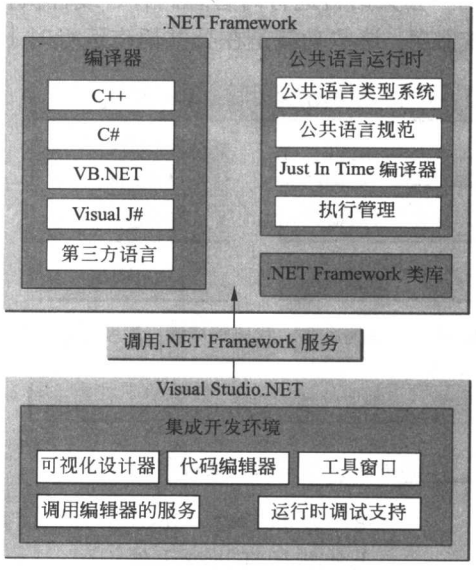

图 1-2 .NET Framework 和 Visual Studio.NET 之间的关系

从图 1-2 中可以看出，Visual Studio.NET 依赖于.NET Framework 提供的服务。这些服务包括 Microsoft 公司或者第三方提供的语言编译器。这些语言编译器是.NET Framework 自身的组成部分，而不属于 Visual Studio.NET。Visual Studio.NET 提供了大量的工具来调用某一种安装的编译器。

公共语言运行时是.NET Framework 的基础。它是执行时管理代码的代理，它提供内存管理、线程管理和远程处理等核心服务。首先，它分别通过公共类型系统（Common Type

System）和公共语言规范（Common Language Specification）定义了标准数据类型和语言间互操作性的规则。Just-In-Time 编译器在运行应用程序之前把中间语言（Intermediate Language）代码转换为可执行代码。公共语言运行时还管理应用程序，在应用程序运行时为其分配内存和解除分配内存。在下面的章节中我们将会详细讨论公共语言运行时及其提供的服务。

.NET Framework 的另一个主要组件是.NET Framework 类库，它是一个综合性的面向对象的可重用类型集合，用户可以使用它开发多种应用程序，这些应用程序包括传统的命令行或者图形用户界面（GUI）应用程序，也包括基于 ASP.NET 所提供的创新的应用程序（如 Web 窗体和 XML Web 服务）。

用户在执行由任何.NET Framework 语言开发的应用程序时，必须安装.NET Framework。.NET Framework 在安装 Visual Studio .NET 时自动安装。也可以从 Microsoft 公司的站点下载免费的.NET Framework。最新版本的 Windows，包括 Windows XP service pack 2，已经安装好了.NET Framework。

1.3 公共语言运行时

公共语言运行时管理内存、线程执行、代码执行、代码安全验证、编译，以及其他系统服务。这些功能是在公共语言运行时上运行的托管代码所固有的。它可以用图 1-3 表示。

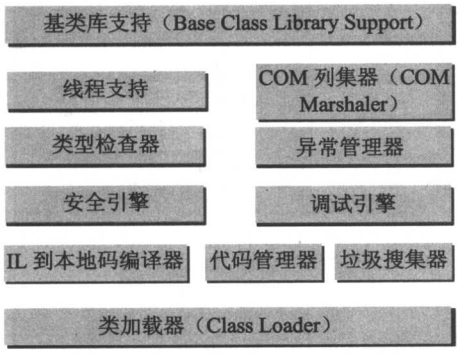

图 1-3 公共语言运行时

公共语言运行时提供了下列重要的服务：

- 公共类型系统（Common Type System，CTS）定义了所有.NET 语言的标准数据类型及其格式。例如，CTS 定义了整型是 32 位大小，还指定了整型值的内部格式。
- 公共语言规范（Common Language Specification，CLS）定义了语言间互操作性的规则。由于 CLS 定义了规则，一种.NET 语言创建的类就可以由其他.NET 语言使用。
- 当.NET 应用程序第一次编译时，编译为一种可以由所有.NET 语言共享的中间语言。在应用程序执行时，Just-In-Time（JIT）编译器把中间语言转换为可以在目标计算机上执行的可执行文件。
- CLR 管理应用程序的执行，也就是说，CLR 负责在创建和销毁对象时，为其分配和解除分配内存。
- 垃圾搜集器（Garbage Collector，GC）负责解除分配内存。

1.3.1 公共类型系统

公共类型系统（CTS）是多信息类型系统，它被内置在公共语言运行时中，支持大多数编程语言中的类型和操作。它定义了声明和使用类型的标准，使得 CLR 可以在不同

语言开发的应用程序之间管理这些标准化的类型并且在不同计算机之间以标准化的格式进行数据通信。

1. 公共类型系统简介

时至今日，编程语言已经发展的丰富多彩，但是它们几乎都有类似的特点——支持各种数据类型（如整型、字符串型等），代码模块化及面向对象。.NET 平台利用了不同语言的这个相近的共性，抽象出公共类型系统（CTS）。公共类型系统构成了.NET 框架的公共语言运行时的基础，其中最重要的一个体现就是.NET 平台的多语言支持，而运行于.NET 平台的每一种语言又为了维护自己的语法特色，而使用别名来代替.NET 的基础数据类型，如 Visual Basic .NET 中的 Integer 类型以及 C#中的 int 类型实际上是基础数据类型 System.Int32 的别名。

公共类型系统不仅定义了所有的数据类型，而且提供了面向对象的模型以及各种语言需要遵守的标准。CTS 可以分为两个大类：值类型和引用类型，同时这两种类型之间还可以进行强制转换，这种转换被称为装箱（Boxing）和拆箱（UnBoxing）。从图 1-4 可以看出公共类型系统的基本结构，CTS 的每一种类型都是对象，并继承自一个基类——System.Object。

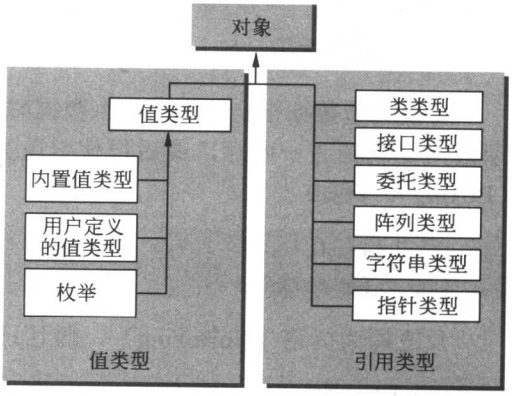

图 1-4 公共类型系统基本结构

2. 值类型和引用类型

值类型继承自 ValueType 类，值类型的变量直接存储数据，实例是被分配在栈中的，并且永远不可能为空。通过下面的例子可以看出，两个整形变量是分别存储数据的，无论哪个变化都不会影响另一个。

```
Dim iA As Int16 = 123
Dim iB As Int16 = iA
iA = 321
Console.WriteLine("A = {0}", iA)
Console.WriteLine("B = {0}", iB)
```

引用类型继承自 Object，引用类型变量存储的是数据内存的地址，而实例是被分配在可以进行垃圾回收的堆中的。一份数据可以被多个变量引用，使用这种变量类型能够起到节省内存资源的作用，同时也会由于一个引用变量的修改而导致其他引用的变更。引用类型的变量是可以为空的，表示它不指向任何对象。如下面的代码，可以看出 clsA 和 clsB 实际上都指向内存中同一个对象的实例。

```
Imports System.IO
Module Module1
    Sub Main()
```

```
        Dim clsA As New clsRef
        clsA.iValue = 123
        Dim clsB As clsRef = clsA
        clsB.iValue = 321
        Console.WriteLine("A = {0}", clsA.iValue)
        Console.WriteLine("B = {0}", clsB.iValue)
        Console.Read()
    End Sub
    Class clsRef
        Public iValue As Int16
    End Class
End Module
```

3. 装箱和拆箱

将值类型数据指定给引用类型数据时，系统会先从堆中配置一块内存，然后将值类型数据复制到这一内存，最后再使引用类型数据指向这一内存，这样的过程称为装箱（Boxing）。例如：

```
Dim I As Int16 = 123
Dim objRef As Object = I
```


反之即为拆箱（UnBoxing），而且只有引用类型数据需要拆箱。例如：

```
Dim J As Int16
J = Int(objRef)
```

在装箱的过程中，内存的配置是很费时的操作，虽然透明于程序员，但是考虑到程序的执行效能，我们应该尽量避免不必要的装箱操作。

1.3.2 公共语言规范

公共语言规范是一组结构和限制，用作库编写者和编译器编写者的指南。它使任何支持 CLS 的语言都可以完全使用库，并且使这些语言可以相互集成。公共语言规范是公共类型系统的子集。对于那些需要编写代码供其他开发人员使用的应用程序开发人员，公共语言规范也非常重要。如果开发人员遵循 CLS 规则来设计公共访问的 API（Application Programming Interface，应用程序编程接口），那么就可以在支持公共语言运行时的任何其他编程语言中很容易地使用这些 API。

CLS 和.NET 自身都依赖于 Windows API 提供的低级服务。Windows API 提供了诸如文本框、按钮、滚动条、列表框和组合框这样的基本控件的类，还提供了基本的 Windows 服务来管理文件、进程和内存。

CLS 定义了所有基于.NET Framework 的语言都必须支持的最小功能集。CLS 定义的规则可以概括如下：

- CLS 定义了命名变量的标准规则。例如，与 CLS 兼容的变量名都必须以字母开头，且不能包含空格。变量名之间必须有所区别（变量名的大小写除外）。

- CLS 定义了原语数据类型，如 Int32、Int64、Single、Double 和 Boolean。
- CLS 禁止无符号数值数据类型。有符号数值数据类型的一个数据位保留来指示数值的正负。无符号数据类型没有保留这个数据位。
- CLS 定义了对支持基于 0 的数组的支持。
- CLS 指定了函数参数列表的规则，以及参数传递给函数的方式。例如，CLS 禁止使用可选的参数。
- CLS 定义了事件名和参数传递给事件的规则。
- CLS 禁止内存指针和函数指针，但是可以通过委托提供类型安全的指针。

任何语言都可以扩展基本的 CLS 需求。例如，有些语言（如 C#）支持无符号整型。不鼓励使用非标准的功能，因为这样做就妨碍了语言之间的互操作性。完全符合 CLS 的语言称为兼容 CLS 的语言。

1.3.3 中间语言

使用.NET 语言开发的任何应用程序都必须在执行之前编译为可执行文件。传统的可执行文件包含了允许文件在特定 CPU 体系结构上执行的本机指令。不同厂商生产的 CPU 的体系结构都不同，也就是说，它们具有不同的寄存器，并且执行一套独有的指令。例如，Macintosh CPU 的指令集与 Intel Pentium 的就不相同。为了克服该问题，Apple 公司的 Virtual PC 应用程序允许苹果机用户通过把 Pentium CPU 指令转换为 Macintosh CPU 指令来运行 Windows 应用程序。这种类型的应用程序通常称为模拟器（emulator）。

其他计算机厂商也生产模拟器。尽管这些软件模拟器解决方案有效，但是它们的执行速度慢于本机的执行速度。另外，软件模拟器无法完全利用 CPU 的某些高级增强功能，例如从 Pentium Ⅲ到 Pentium Ⅳ的增强功能。

.NET Framework 解决了这个不同硬件体系结构之间的问题，方法是把应用程序编译为一种称为 Microsoft Intermediate Language（MSIL）的独立于硬件的格式。在本书中我们把它叫做中间语言（Intermediate Language，IL）。中间语言的主要特征如下：

- 面向对象和使用接口；
- 值类型和引用类型之间的巨大差别；
- 强数据类型；
- 使用异常来处理错误；
- 使用特性（attribute）。

中间语言的格式类似于程序集语言。程序集语言的语句直接与内部 CPU 体系结构支持的指令相关。但是中间语言的格式通常不依赖于特定 CPU 的体系结构。也就是说，中间语言不直接引用 CPU 寄存器或者执行 CPU 指令。当用户执行中间语言格式的应用程序时，另一个名为 Just-In-Time（JIT）编译器的实用程序进一步把中间语言转换为目标 CPU 可以执行的本机可执行文件。使用 JIT 编译器把中间语言文件转换为本机可执行文件的过程通常称为 JITting。

除了不同 CPU 厂商会制造各种不兼容的硬件之外，同一个 CPU 厂商，如 Intel，也常常会制造出带有支持特殊指令的附加功能的新的 CPU。通过把应用程序编译为独立于