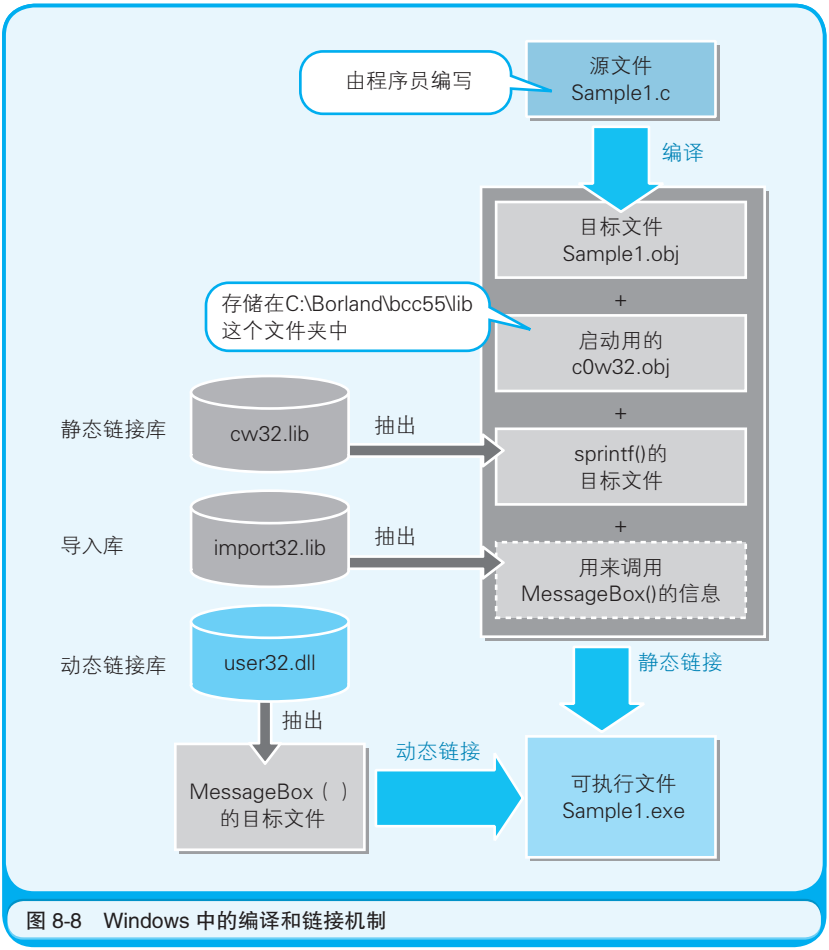

图 8-8　Windows 中的编译和链接机制

8.7　可执行文件运行时的必要条件

在了解了通过程序的编译及链接来生成 EXE 文件的机制后，接下来看一下 EXE 文件的运行机制。EXE 文件是作为单独的文件储存在硬盘中的。通过资源管理器找到并双击 EXE 文件，就会把 EXE 文件的内容加载到内存中运行。

请大家思考一下下面的问题。本地代码在对程序中记述的变量进行读写时，是参照数据存储的内存地址来运行命令的。在调用函数时，程序的处理流程就会跳转到存储着函数处理内容的内存地址上。EXE 文件作为本地代码的程序，并没有指定变量及函数的实际内存地址。在类似于 Windows 操作系统这样的可以加载多个可执行程序的运行环境中，每次运行时，程序内的变量及函数被分配到的内存地址都是不同的。那么，在 EXE 文件中，变量和函数的内存地址的值，是如何来表示的呢?

下面就让我们来揭晓答案。那就是 EXE 文件中给变量及函数分配了虚拟的内存地址。在程序运行时，虚拟的内存地址会转换成实际的内存地址。链接器会在 EXE 文件的开头，追加转换内存地址所需的必要信息。这个信息称为**再配置信息**。

EXE 文件的再配置信息，就成为了变量和函数的相对地址。相对地址表示的是相对于基点地址的偏移量，也就是相对距离。实现相对地址，也是需要花费一番心思的。在源代码中，虽然变量及函数是在不同位置分散记述的，但在链接后的 EXE 文件中，变量及函数就会变成一个连续排列的组。这样一来，各变量的内存地址就可以用相对于变量组起始位置这一基点的偏移量来表示，同样，各函数的内存地址也可以用相对于函数组起始位置这一基点的偏移量来表示。而各组基点的内存地址则是在程序运行时被分配的 (图 8-9)。

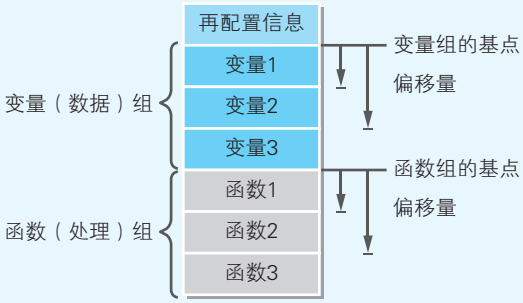

图 8-9　链接后的 EXE 文件的构造

8.8　程序加载时会生成栈和堆

EXE 文件的内容分为再配置信息、变量组和函数组，这一点想必大家都清楚了吧。不过，当程序加载到内存后，除此之外还会额外生成两个组，那就是栈和堆。**栈**是用来存储函数内部临时使用的变量（局部变量[①]），以及函数调用时所用的参数的内存区域。**堆**是用来存储程序运行时的任意数据及对象的内存领域（图 8-10）。

EXE 文件中并不存在栈及堆的组。栈和堆需要的内存空间是在 EXE 文件加载到内存后开始运行时得到分配的。因而，内存中的程序，就是由用于变量的内存空间、用于函数的内存空间、用于栈的内存空间、用于堆的内存空间这 4 部分构成的。当然，在内存中，加载 Windows 等操作系统的内存空间又是另外一回事了（图 8-10）。

① 局部变量是指只在调用函数时存在于内存中的变量。例如，在代码清单 8-1 中，WinMain 函数的处理中的 ave 和 buff 都是局部变量。全局变量是指程序运行时一直存在于内存中的变量。代码清单 8-1 中的 title 就是全局变量。

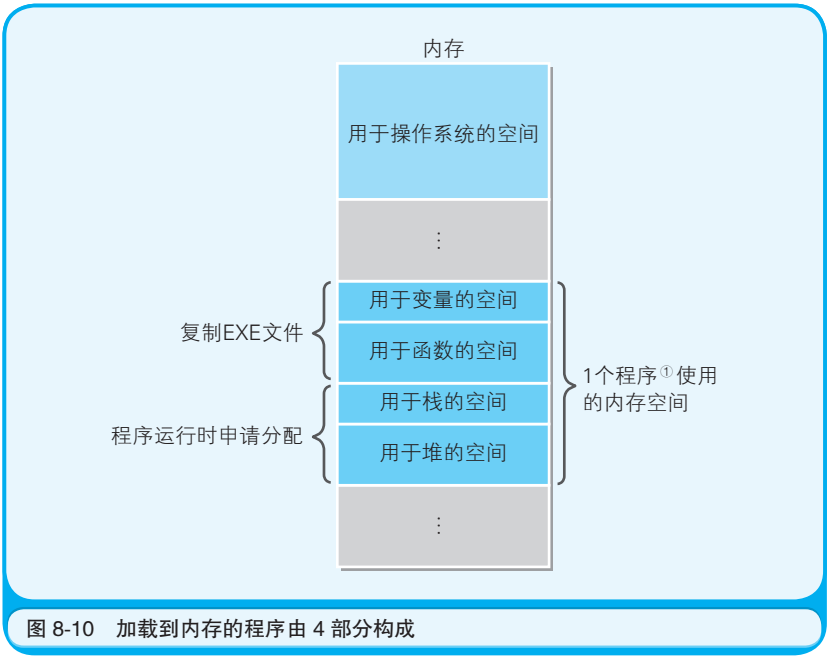

图 8-10 加载到内存的程序由 4 部分构成

栈及堆的相似之处在于，他们的内存空间都是在程序运行时得到申请分配的[2]。不过，在内存的使用方法上，二者存在些许不同。栈中对数据进行存储和舍弃（清理处理）的代码，是由编译器自动生成的，因此不需要程序员的参与。使用栈的数据的内存空间，每当函数被调用时都会得到申请分配，并在函数处理完毕后自动释放。与此相对，堆的内存空间，则要根据程序员编写的程序，来明确进行申请分配或释放。

① 不管是什么程序，程序的内容都是由处理和数据构成的。大多数编程语言都是用函数来表示处理、用变量来表示数据。

② 栈和堆的大小，可以由程序员任意指定。在高级编程语言中，编译器会自动生成指定栈和堆大小的代码，并将其附加到程序中。

根据编程语言的不同，对堆用的内存空间进行申请分配和释放的程序的编写方法也是多种多样的。C 语言中是通过 malloc() 函数来进行申请分配、通过 free() 函数来释放的。而 C++ 中则是通过 new 运算符来申请分配、通过 delete 运算符来释放的。无论是 C 语言还是 C++，如果没有在程序中明确释放堆的内存空间，那么即使在处理完毕后，该内存空间仍会一直残留。这个现象称为**内存泄露**（memory leak），它是令 C 语言及 C++ 的程序员们十分头疼的一个 bug（程序的错误）。如果内存泄露一直存在的话，就有可能会造成内存不足而导致宕机。这就好比，如果水龙头一直嘀嗒嘀嗒地漏水，那么一晚上的时间水桶就可能会装满并溢出。

8.9　有点难度的 Q&A

Q：编译器和解释器有什么不同?

A：编译器是在运行前对所有源代码进行解释处理的。而解释器则是在运行时对源代码的内容一行一行地进行解释处理的。

Q：“分割编译”指的是什么?

A：将整个程序分为多个源代码来编写，然后分别进行编译，最后链接成一个 EXE 文件。这样每个源代码都相对变短，便于程序管理。

Q：“Build”指的是什么?

A：根据开发工具种类的不同，有的编译器可以通过选择“Build”菜单来生成 EXE 文件。这种情况下，Build 指的是连续执行编译和链接。

Q：使用 DLL 文件的好处是什么?

A：DLL 文件中的函数可以被多个程序共用。因此，借助该功能可以节约内存和磁盘。此外，在对函数的内容进行修正时，还不需要重

新链接（静态链接）使用这个函数的程序[①]。

Q：不链接导入库的话就无法调用 DLL 文件中的函数吗？

A：通过使用 LoadLibrary() 及 GetProcAddress() 这些 API，即使不链接导入库，也可以在程序运行时调用 DLL 文件中的函数。不过使用导入库更简单一些。

Q：“叠加链接”这个术语指的是什么？

A：将不会同时执行的函数，交替加载到同一个地址中运行。通过使用“叠加链接器”这一特殊的链接器即可实现。在计算机中配置的内存容量不多的 MS-DOS 时代，经常使用叠加链接。

Q：和内存管理相关的“垃圾回收机制”指的是什么呢？

A：垃圾回收机制（garbage collection）指的是对处理完毕后不再需要的堆内存空间的数据和对象[②]进行清理，释放它们所使用的内存空间。这里把不需要的数据比喻为了垃圾。进行该处理时，C 语言用的是 free() 函数，C++ 用的是 delete 运算符。在 C++ 的基础上开发出来的 Java 及 C# 这些编程语言中，程序运行环境会自动进行垃圾回收。这样就可以避免由于程序员的疏忽（忘了记述内存的释放处理）而造成内存泄露了。

① 关于 DLL 文件可以被多个程序共用的好处，第 5 章中有详细介绍。

② 堆中的 object（对象）不是 object 文件（目标文件），而是面向对象编程语言的 object（对象，数据和处理的集合体）。

第 9 章
操作系统和应用的关系

热身问答

阅读正文前，让我们先回答下面的问题来热热身吧。

1. 监控程序的主要功能是什么?
2. 在操作系统上运行的程序称为什么?
3. 调用操作系统功能称为什么?
4. Windows Vista 是多少位的操作系统?
5. GUI 是什么的缩写?
6. WYSIWYG 是什么的缩写?

怎么样？是不是发现有一些问题无法简单地解释清楚呢？下面是笔者的答案和解析，供大家参考。

答案

1. 程序的加载和运行
2. 应用或应用程序
3. 系统调用（system call）
4. 32 位（也有 64 位的版本）
5. Graphical User Interface（图形用户界面）
6. What You See Is What Your Get（所见即所得）

解析

1. 监控程序也可以说是操作系统的原型。
2. 文字处理软件和表格计算软件等都是应用。
3. 应用通过系统调用（system call）间接控制硬件。
4. Windows Vista 有 32 位 CPU 用的版本，也有 64 位 CPU 用的版本。
5. 显示器中显示的窗口及图标等通过鼠标点击可以直观操作的用户界面。
6. WYSIWYG 是指可以直接将显示器中显示的内容在打印机上打印出来。这也是 Windows 的特征之一。

本章重点

利用计算机运行程序大部分都是为了提高处理效率。例如，Microsoft Word 这样的文字处理软件，是用来提高文本文件处理效率的程序，Microsoft Excel 等表格计算软件，是用来提高账本处理效率的程序。类似于文字处理软件及表格计算软件这样，为了提高特定处理效率的程序总称为“应用”。

程序员的工作就是编写各种各样的应用来提高业务效率。而应用的运行环境，也就是操作系统，则直接从软件商店等处购买就可以了。不过，一定不能忽略操作系统，否则就无法编写应用。这是因为，程序员是通过利用操作系统提供的功能来编写应用的。本章中，我们会对操作系统的角色，以及应用利用操作系统功能的方法进行说明。关于操作系统的类型，这里我们选取了用户人数较多的 Windows 作为示例。

9.1 操作系统功能的历史

首先，在简单回顾操作系统[①]的历史的同时，我们来看一下操作系统到底是怎样的软件。

在计算机中尚不存在操作系统的年代，完全没有任何程序，因此程序员就需要编写出处理相关的所有程序。用机器语言编写程序，然后再使用开关将程序输入，这一过程非常麻烦。于是，有人开发出了仅具有加载和运行功能的**监控程序**，这就是操作系统的原型。通过事先启动监控程序，程序员就可以根据需要将各种程序加载到内存中运

① 操作系统（Operating System）也称为基础软件。操作系统是计算机运行时不可或缺的控制程序，以及在控制程序下运转的为其他软件运行提供操作环境的软件的统称。另外，在操作系统上运行的应用也称为“应用程序”。