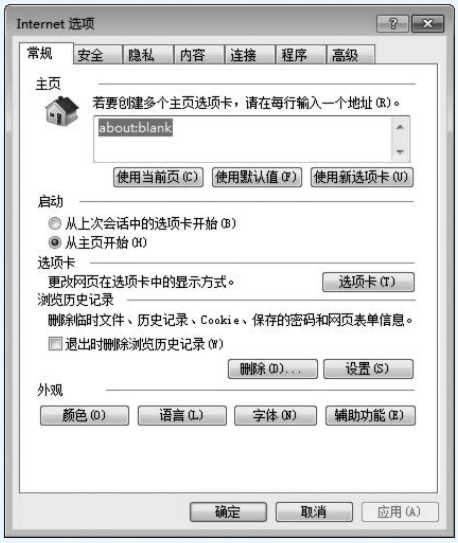


图 9-8　用户可以按照任意顺序来操作窗口的任意元素

在像 MS-DOS 这种没有使用 GUI 的操作系统中，应用的处理流程由程序员决定，用户按照定好的流程来进行操作即可。与此相反，采用 GUI 的操作系统中运行的应用，则是由用户决定处理流程的。因此，程序员就必须要制作出在任何操作顺序下都能运行的应用。这就要求以前的程序员要改变观念。这就是 GUI 的难点。如果程序员最初接触的操作系统就是 Windows 的话，那他或许会认为 GUI 是理所当然的。

(4) 通过 WYSIWYG 实现打印输出

WYSIWYG 指的是显示器上显示的内容可以直接通过打印机打印输出。在 Windows 中，显示器和打印机是被作为同等的图形输出设备处理的，而该功能也就为 WYSIWYG 的实现提供了条件。

借助 WYSIWYG 功能，程序员可以轻松不少。最初，为了实现在

显示器中显示和在打印机中打印，就必须分别编写各自的程序。而在Windows 中，借助 WYSIWYG 功能，基本上在同一个程序中就可以实现显示和打印这两方面的操作了（当然，也可以将显示和打印的内容放在不同的程序中处理）。

（5）提供多任务功能

多任务指的是同时运行多个程序的功能。Windows 是通过**时钟分割**技术来实现多任务功能的。

时钟分割指的是在短时间间隔内，多个程序切换运行的方式。在用户看来，就是多个程序在同时运行。也就是说，Windows 会自动切换多个程序的运行（图 9-9）。此外，Windows 中还具有以程序中的函数为单位来进行时钟分割的**多线程**[①]功能。

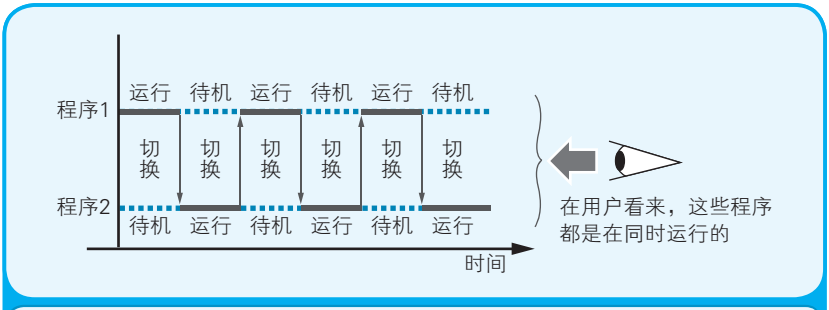


图 9-9　通过时钟分割实现多任务的机制

（6）提供网络功能及数据库功能

Windows 中，网络功能是作为标准功能提供的。数据库（数据库服务器）功能有时也会在之后进行追加。网络功能和数据库功能，虽并不是操作系统本身不可欠缺的功能，但因为它们和操作系统很接近，所

① 关于多线程，我们会在第 10 章进行说明。

以被统称为**中间件**而不是应用。意思是处于操作系统和应用的中间（middle）。操作系统和中间件合在一起，也称为**系统软件**。应用不仅可以利用操作系统，也可以利用中间件的功能（图 9-10）。

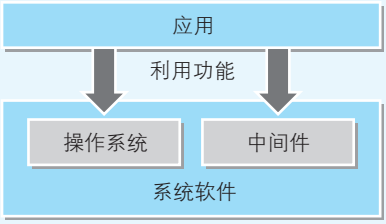


图 9-10　应用可以利用操作系统和中间件的功能

相对于操作系统一旦安装就不能轻易替换，中间件则可以根据需要进行任意的替换。不过，大多数情况下，中间件变更后应用往往也需要变更，因此中间件的变更也不是那么容易。

（7）通过即插即用实现设备驱动的自动设定

即插即用（Plug-and-Play）指的是新的设备连接（Plug）后立刻就可以使用（Play）的机制。新的设备连接到计算机后，系统就会自动安装和设定用来控制该设备的**设备驱动**程序。

设备驱动是操作系统的一部分，提供了同硬件进行基本的输入输出的功能。键盘、鼠标、显示器、磁盘装置等，这些计算机中必备的硬件的设备驱动，一般都是随操作系统一起安装的。如果之后再追加新的网卡（NIC[①]）等硬件的话，就需要向操作系统追加该硬件专用的设

① NIC（Network Interface Card）是计算机连接网络（LAN）时使用的设备。也称为网卡或者 LAN 卡。

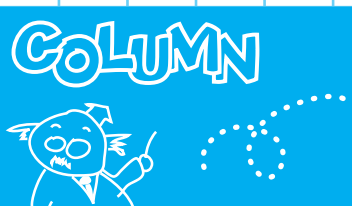
备驱动。大家购买的新的硬件设备中，通常都会附带着软盘或 CD-ROM，里面通常都收录着该硬件的设备驱动。

有时 DLL 文件也会同设备驱动文件一起安装。这些 DLL 文件中存储着用来利用该新追加硬件的 API（函数集）。通过 API，可以制作出运用该新硬件的应用。

可以任意追加设备驱动和 API 的机制使 Windows 操作系统变得非常灵活。这里所说的灵活，是指可以事后再对新追加的硬件进行处理。

本章中，为了明确区分应用和操作系统，在解说的过程中，当遇到想用“这个程序……”来表达的地方时，我们特意使用了“这个应用……”。这是因为，程序是操作系统、中间件、应用等所有软件的统称。因此，通常程序员制作的应该都是应用，而不是操作系统。不过，既然是应用，那么就肯定会通过某种形式来利用操作系统的功能。程序员一定要注意到这一点。例如，如果应用没有正常运行的话，那么很有可能就不是硬件的问题，而是操作系统的使用方法出现了偏差。而中间件和设备驱动，大家也可以把它们看作是操作系统的一部分。

在本书的解说中，到目前为止，“本地代码”这个术语已经出现过很多次。假如能用本地代码直接编写程序的话，那么程序的运行机制想必也就一目了然了。不过，能够直接用本地代码编写程序的人，实际上并不多见。大家的普遍做法都是使用汇编语言来代替本地代码。在接下来的下一章，我们将通过用汇编语言编写程序，来看一下程序的实际运行机制。



如果是你，你会怎样介绍？

向超喜欢手机的女高中生讲解操作系统的作用

笔者：你有手机吗？

女高中生：有啊。

笔者：什么机型啊？

女高中生：Docomo 的最新版。

笔者：是吗，真不错啊！那么，你都用手机做什么呢？

女高中生：当然是跟好朋友通电话了啊。有时候也会发邮件、查查音乐会信息等。

笔者：这样啊。不过，手机也是电话对吧。为什么这个电话能发邮件、查看音乐会信息呢？你知道原因吗？也就是说电话是如何连接互联网的呢？

女高中生：因为是电话，所以能连接互联网啊！

笔者：这么说也没错，但最近的手机并不是单纯的电话，更像是具有电话功能的计算机。因此，可以把手机看作是便携式计算机。

女高中生：感觉话题要转到大叔您擅长的领域了。

笔者：呵呵，这不是挺好的吗。计算机是运行程序的设备，这个你是知道的吧？

女高中生：知道啊。我用过计算机。

笔者：虽然手机不是手提电脑，但它里面也是有程序的。正是因为有了这些程序，手机才可以连网。显示文字和图片等也都是通过程序来实现的。

女高中生：这个当然啊，这个话题真没意思。

笔者：（不妙……换个话题看看）对了，用过 iApp 吗？

女高中生：用过！用过！通过它就可以在手机中玩游戏了。

笔者：（有戏有戏，那就从这里进入主题吧）iApp 的 i 以及 iMode 的 i，都是 Internet 的 i，App 指的是 Application，就是应用。

女高中生：应用是什么啊？

笔者：问得好。我们总是笼统地说程序，其实程序可以根据功能的不同分为操作系统和应用。

女高中生：操作系统和应用？

笔者：iApp 中有各种游戏对吧。一个游戏程序是不是需要具备制定游戏规则的功能、使手机按键反应的功能、显示文字和图片的功能呢？

女高中生：？？？

笔者：游戏种类不同的话，当然游戏的规则也会不同，不过按键的响应功能及显示文字和图片的功能在任何游戏中都是相同的，没错吧？

女高中生：我怎么感觉不太一样呢……

笔者：对编写程序的来人说，是一样的！明明是同样的功能，可是如果每开发一个游戏都要生成一遍，就很浪费时间对吧。因此我们就可以把所有游戏共同的功能集合起来做成一个独立的程序，这个程序就称为操作系统。而像游戏的规则这种各游戏独有的程序，就是应用。

女高中生：程序就这样分成了两种类型了？

笔者：对，正是如此！手机中都会提前安装上操作系统。想要玩游戏的时候，只用下载游戏程序也就是应用并安装到手机上就可以了。游戏结束后，应用就消失了。不过操作系统是不会消失的。

女高中生：额，有点明白，又有点不明白……

笔者：那么，以计算机为例再来说明一下。在计算机中，程序同样分为操作系统和应用。Windows 知道吧。Windows 就是操作系统。后面安装上的文字处理软件及游戏等就是应用。

女高中生：Windows 中也自带纸牌游戏啊。

笔者：那个是 Windows 的附件中自带的应用，并不是操作系统本身。

女高中生：嗯……

笔者：怎么样，都明白了吗？

女高中生：差不多吧。

第10章 通过汇编语言了解程序的实际构成

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. 本地代码的指令中，表示其功能的英语缩写称为什么？
2. 汇编语言的源代码转换成本地代码的方式称为什么？
3. 本地代码转换成汇编语言的源代码的方式称为什么？
4. 汇编语言的源文件的扩展名，通常是什么格式？
5. 汇编语言程序中的段定义指的是什么？
6. 汇编语言的跳转指令，是在何种情况下使用的？

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. 助记符
2. 汇编
3. 反汇编
4. .asm
5. 构成程序的命令和数据的集合组
6. 将程序流程跳转到其他地址时需要用到该指令

解析

1. 汇编语言是通过利用助记符来记述程序的。
2. 使用汇编器这个工具来进行汇编。
3. 通过反汇编，得到人们可以理解的代码。
4. .asm 是 assembler（汇编器）的略写。
5. 在高级编程语言的源代码中，即使指令和数据在编写时是分散的，编译后也会在段定义中集合汇总起来。大家看过汇编语言的源代码后，就会清楚了。
6. 在汇编语言中，通过跳转指令，可以实现循环和条件分支。

本章重点

笔者在学生时代曾写过比较 C 语言源代码和汇编语言源代码的报告。这个报告的研究方法是，把 C 语言的各种语法变换成汇编语言，然后对这些内容进行调查。通过研究，笔者对程序的运行机制有了深刻的了解。

希望各位读者看完本章内容也能有同样的收获。在本章的前半部分，我们会对 CPU 解释运行的本地代码和汇编语言的一对一关系、汇编语言的源代码中包含的用来指示汇编器的伪命令、栈的 push/pop 以及调用函数的机制进行说明。

在本章的后半部分，会向大家介绍一下局部变量和全局变量的不同、循环等流程控制的实现方式等。在研究对象方面，我们选取了 Pentium 等 x86 系列 CPU 用的汇编语言，编程工具则依然使用前面章节中用到的 Borland C++。本章的内容相比其他章节多了不少，请大家耐心地阅读下去。

10.1 汇编语言和本地代码是一一对应的

接下来就让我们进入到本章的前半部分。在前面章节中已经多次提到，计算机 CPU 能直接解释运行的只有本地代码（机器语言）程序。用 C 语言等编写的源代码，需要通过各自的编译器编译后，转换成本地代码。

通过调查本地代码的内容，可以了解程序最终是以何种形式来运行的。但是，如果直接打开本地代码来看的话，只能看到数值的罗列。如果直接使用这些数值来编写程序的话，还真是不太容易理解。因而就产生了这样一种想法，那就是在各本地代码中，附带上表示其功能