# Windows™ 应用程序集成技术

学习如何集成主要的
Windows 应用程序

最大发挥对象链接
与嵌入(OLE)技术的功能

快速容易地建立多应用
程序报表和文档

掌握 OLE 客户和服务器
应用程序的内外环境和技术

[美] Rob Krumm 著


希望
科学出版社
龍門書局

# Windows 应用程序集成技术

〔美〕Rob Krumm 著

张小力 张 群 夏春和等 译

施 丽 审校

科学出版社

龍門書局

(京)新登字 092 号

## 内 容 简 介

Windows 提供了可以在两个同时运行的应用程序中"交换信息"的能力，它可以处理那种需要不止一种数据类型的复杂项目。Windows 还有许多精彩的特性，这些特性向用户提供了各种各样的方法，用户使用这些方法可以实现应用程序之间的信息交换，并可将他们的命令结构集合起来，以使两个应用象一个单独的程序一样运行。

Windows 为这些特性取名为 DDE、OLE 以及 OLE Automation 等等。本书的目的就是要将这些功能强大但通常看似晦涩难懂的抽象的特性，通过一步一步详细地讲解，最后让读者能够了解并实在地使用它们。在这个过程中，我们还将提供给大家一些实用示例，让大家能够学会怎样用这些特性来解决日常工作中遇到的一些计算问题。

本书的读者对象包括计算机初级用户和高级程序设计人员。

**欲购本书的用户，请直接与北京海淀 8721 信箱书刊部联系，邮码 100080，电话 2562329。**

## 版 权 声 明



# Windows 应用程序集成技术

[美] Rob Krumm 著

张小力 张 群 夏春和等 译

施[illegible] 校

责任编辑：董芳明

[illegible] 出版

北京东黄城根北街 16 号

邮政编码 100717

三空印刷厂印刷

新华书店北京发行所发行 各地新华书店经售

*

1995 年 1 月第 版 开本 787×1092 1/16

1995 年 1 月第一次印刷 印张 20.75

印数 1—10000 字数 503 千

ISBN 7-03-004731-1/TP·444

定价：53.00 元

# 献 词

——致 Morgan

在这里，我想问一下，生命的意义究竟是什么？从我们降生到这个世上开始，我们的生命只不过是历史长河中的一瞬间而已，然后我们就会死去。一个蜘蛛的生命看起来免不了就是一直忙于捕食苍蝇。如果在我的生命中尽量去帮助别人，或许这样可以提高我生命的意义，上帝能够洞察每一个人一生所做的事情并且也将认同这种有意义的生活。

*E. B. White, Charlotte's Web*

# 致　谢

我首先要向我的同伴 Carolyn S. Rigiero 表示感谢，感谢她对这本书（以及其他的 40 多本书）所付出的辛勤的劳动和心血。如果世上每个人都象她那样具有诚实的性格、献身精神以及那种一定要尽可能献给大家最好工作成果的愿望，那么，我们的世界将变得更加美好。

在此，我还向 Michael Sprague 致以最诚挚的谢意，感谢他对此书付出的辛勤劳动以及对这本书出版所提供的大力支持。

# 译者序

Windows 提供了可以在两个同时运行的应用程序中“交换信息”的能力，它可以处理那种需要不止一种数据类型的复杂项目。Windows 还有许多精彩的特性，这些特性向用户提供了各种各样的方法，用户使用这些方法可以实现应用程序之间的信息交换，并可将他们的命令结构集合起来，以使两个应用象一个单独的程序一样运行。

Windows 为这些特性取名为 DDE、OLE 以及 OLE Automation 等等。为帮助我国计算机用户熟练地使用 Windows，为提高我国计算机的应用水平，我们翻译了这本《Windows 应用程序集成技术》。本书的目的就是要将这些功能强大但通常看似晦涩难懂的抽象的特性，通过一步一步详细地讲解，最后让读者能够了解并实在地使用它们，在这个过程中，我们还将提供给大家一些实用示例，让大家能够学会怎样用这些特性来解决日常工作中遇到的一些计算问题。

在本书的翻译过程中，周方、刘丽华、张小力等翻译了第一至第五章，王小明、张莉、黄大山等翻译了第六至第八章，马焕峰、张群等翻译了第九至第十一章，朱晓军、刘丽等翻译了第十二和第十三章，梁卫权、吴谋等翻译了第十四和第十五章。施丽同志进行了全书的统编和审校工作。

另外，王晓江等同志也参加了部分翻译和审校工作，刘莉、邵红、张芹等负责本书的录入与排版工作，在此一并感谢。

由于时间仓促，翻译过程中难免出现错误，欢迎广大读者指正。

译　者

1994. 12 于北京

# 引 言

当一个 DOS 系统用户开始接触 Windows 系统时，这个系统给他的第一个惊喜就是用户在该系统中居然可以将输入到某个应用中的内容拷贝并粘贴到另一个应用中去。Windows 提供的这种可在两个同时运行的应用程序中“交换信息”(exchange information)的能力对一个 DOS 系统用户来讲，无异于“美梦成真”。在这个系统中，可以处理那种需要不止一种数据类型的复杂的项目——如一份同时要求有电子表格和商务图表的报告。DOS 系统用户在此之前为找到这样一种解决办法已经奋斗了将近十年的时间，现在，这样一个完善的系统出现在他们眼前，怎能不令他们欣喜若狂！

就象多重拷贝和粘贴一样，刚才所提到的这种功能只不过是“冰山之一角 (tip of the iceberg)”。在 Windows 的这些表层之下，还有许多精彩的特性，这些特性向用户提供了各种各样的方法，用户使用这些办法可以实现应用程序之间的信息交换并可将他们的命令结构集合起来以使两个应用象一个单独的程序一样运行。

Windows 为这些特性取名为 DDE、OLE 以及 OLE 自动化等等。本书的目的就是要将这些功能强大但通常看似晦涩难懂的抽象的特性，通过一步一步详细地讲解，最后让用户能够了解并实在地使用它们。在这个过程中，我们还将提供给大家一些实用示例，让大家能够学会怎样用这些特性来解决日常工作中遇到的一些计算问题。

在写这本书的过程中，我一直处于一种紧迫、激动、兴奋的心情中。因为我在书中要向大家介绍的是一种可将两个以前必须要用不同的工具才能完成的将应用合并在一起使用的技术，掌握了这种技术之后，我们可以更快捷、更轻松地完成工作。就象我五岁的女儿所说的，“真棒，爸爸！”。此外，这个工作对我来说就好象只有通过这本书才能使 Windows 进行工作一样地重要。Windows，这个词在本书中将经常用到，它其实是一个系统的名称，在这个系统中，传统的各个独立应用程序之间的界限将荡然无存。使用最新的 OLE 2.0 特性，用户就可以在对一个诸如 Access 的数据库应用进行操作的同时，使用 Word 和 Exel 的特性，而且使用这些特性时，根本就可以不要在屏幕上进行显示，一切都是悄然进行的。

这本书非常有用的一个原因就是，现在绝大多数的软件手册和计算机书籍都是着重讲述一个单独产品的，书店老板也比较喜欢那种可以整齐地分类放到某个特定的书架上的书，例如 Windows 中的字处理部分。但是，在讨论数据交换和应用合并的定义时，我们至少会在同时考虑这两个程序——当然，如果包含 Windows 的话，就是三个了，因为 Windows 在所有数据交换和应用合并任务中都是不可缺少的部分。

由于在本书中讨论的所有这些任务都要求用户在同一时间使用至少两个不同的应用程序，因此我在书中包括了在字处理、电子表格以及数据库领域中大多数居于领先地位的应用软件的示例。当然，并非所有的这些程序都能提供同一级别的数据交换和应用合并特性。我将尽量向大家演示各种各样不同的应用程序是怎样执行同样的一种任务的。并

且由于大多数用户不可能都有上述的这些应用程序，我认为即使没有使用过其中的某种软件，在看到用于该软件的示例时，同样可以从中学到东西。

尽管我尽最大的努力要在编写本书的过程中保持客观的态度，但是在讨论最高级的特性和技术时，读者仍可能会发觉我比较着重于 Microsoft 应用，如最新发布的 Word 6.0 和 Exel 5.0 以及 Access 2.0 等。从我自己来讲，这个原因很简单。因为上述的这几种应用程序在利用 Windows 中的数据交换和应用合并特性时，当然比其竞争对手要占更大的优势。因为同是 Microsoft 公司的产品，所以他们使用的是同一种宏编程语言，而且该语言是专为这些高级特性而设计的。

在写这本书的过程中，我的打算就是要将使用 DDE 和 OLE Automation 这些在任何单独的应用或 Windows 手册或书中都找不到的应用之间的操作集中在一起来讨论，并尽量使我的讲述直接和实用，以便使具有一般 Windows 知识的读者能够学会怎样开发和使用在 Windows 表层之下的储存的这些丰富特性的“金矿”。

# 写这本书的目的

这本书是为那些想学会怎样使用 Windows 高级特性的用户而写的，书中所述的这些特性是内嵌在 Windows 应用中的，用它可以实现两个程序之间的数据交换。本书并不着重于面向职业的程序员，更确切地来讲，该书是针对希望通过合并在不同的应用中的数据从而解决某个日常问题的普通 Windows 用户。

本书主要包括了下面的这几项内容：

（1）连接在不同的应用中建立的文档。

（2）使用在许多流行的程序中都有的 Insert Object 特性。

（3）用一个程序来控制另一个程序的运行。

（4）DDE——Dynamic Data Exchange（动态数据交换）的含义及其用法。

（5）OLE 2.0 Automation 的含义及其用法。

本书向一个普通的 Windows 用户解释了应用是怎样合并工作的，并提供了一些特定的、可靠的示例程序，以准确地告诉用户需要做哪些事情才能使这些特性为自己的程序工作。

一旦学会了怎样使自己的 Windows 应用合并在一起工作，用户就可以更加高效地完成某项任务，因为你已经尝试了怎样将一个程序的某种好的特性取出来，填进另一个应用的命令组中。这样，通过使两个程序协调在一起运行，即可避免许多重复的过程。

# 目 录

# 第一章　让应用程序协同工作

如果我们去问 Windows 用户，Windows 与 DOS 相比起来最大的、独特的优势是什么，你可能会得到各种各样的答案，其中包括：图标的使用、有统一标准的下拉式菜单、或是可以使用不同字型和正文风格的能力。在这些答案中，肯定会包括同时运行一个以上程序的能力。

DOS 系统的一个最本质的弱点在于，该系统在设计之初，就假定了在同一时刻只能对一个程序进行操作。当用户将某个程序加载到 DOS 系统中时，用户同时就被自动限制到了内嵌在该应用程序中的命令和特性组中。事实上，即使当时在用户自己的硬盘上存有 2 个到 2000 个其他的程序，也无权使用。在 DOS 模式下，在同一时间内，用户只能使用所有存储在磁盘上的程序中的一个。

这就意味着，如果用户当前正在使用的是一个字处理软件，但此时又需要建一个图表，在这种情况下，用户必须先退出这个字处理程序，并将其从内存中有效地清除。只有在这一系列的工作完成之后，用户才可以将图形程序装入以前由那个字处理程序占用的内存空间中。

在八十年代中期，软件编制人员在 DOS 系统系统中发现了一个漏洞（loophole），并利用这个漏洞实现了内存驻留功能。驻留功能的作用是在退出一个程序的同时，允许其程序代码保留在内存中——即使此用户装入了另一个应用程序。这种程序被称为“中断并驻留（TSR）”，因为它们即使是在装入和运行另一个应用时都将一直保留在内存中。TSR 程序将一个钩（hook）放置到内存中，这个钩将对键盘输入进行扫描，以寻找一个特殊的组合键（例如：Alt＋Shift＋\）。如果用户键入了这个特定的键，那么对计算机的控制权将从当前的应用程序移交给 TSR 程序，该程序一直静静地伺服在内存中，并随时等待特定的键来激活。在随后的几年中，TSR 程序就成了软件市场上最热门的产品，TSR 程序为用户提供了一种全新的方法，可将在两个程序中实现的操作合并在一起运用。在这些 TSR 程序中，最成功、最流行的一种是 Borland 出品的 Sidekick。Sidekick 的典型特性就是，当用户在运行一个字处理软件时，可以弹出一个计算器菜单，用户可通过这个计算器进行某些计算，然后将运算结果自动插入到正在进行字处理的文档中。

而在九十年代，Windows 就取代了 DOS 系统和 TSR 成了操作环境。在这个环境中，用户可以轻松地在运行字处理程序、电子表格程序或其他应用的同时，使用诸如计算器之类的桌面附件，而且 Windows 的剪贴板（Clipboard）在某种程度上来讲，允许用户将某个应用中的数据拷贝并粘贴到其他应用中。Windows 的这些特性运行起来比旧的 DOS/TSR 方式更好，因为 Windows 不象 DOS 系统，它本来就是为同时运行一个以上的应用而设计的。

但是，除此之外还有哪些功能更强大的应用合并方式呢？既然从某个程序中将某一条目拷贝和粘贴到另一个应用程序中的办法行之有效，而且节省时间，Microsoft 于是就继

续致力于开发出功能更强大的方式，以完成多应用的合并和程序间的数据交换。这种努力的结果就是动态数据交换（DDE）以及对象链接与嵌入（OLE）。

本书的目的就是要向用户介绍怎样运用 DDE 和 OLE 来实现应用程序之间的数据交换和合并，以及合并程序间的操作。

这本书中的各项内容归纳起来主要是下面两种类型的操作：

- **数据交换**。数据交换指的是将某个应用程序中的信息发送到另一个应用程序中，通常这种传送不需要将传送信息存到某个磁盘文件中。
- **应用程序合并**。Windows 提供了一种机制，在这种机制下，一个应用程序可以控制实际发生在另一个应用程序中的操作过程。这就是所谓的遥控方式，在此方式下，输入到某个程序中的命令实际由一个不同的应用程序来执行。

有一点需要牢记的是，在一个单独的宏或程序中，可能会同时使用上述的这两类操作。某一顶给定的任务也许同时需要数据交换和一个程序对另一个程序的控制操作。

## 1.1 本书的结构

这本书在下面这几个方面上与其他书有不同之处。首先，与大多数计算机类书籍不同，它讲述的内容并不局限于某个特定的软件应用程序，如 WordPerfect 或者一个 Windows 或 DOS 之类的操作环境。

### 1.1.1 所有的 Windows 应用

由于该书主要是讨论在 Windows 环境下操作的应用程序，因此在书中我们提到这些应用程序时，都没有加上 for Windows 这个标记。这就是说，我们在书中所指的 Word、Lotus 1-2-3 以及 Paradox 等分别应该是 Word for Windows、Lotus 1-2-3 for Windows 以及 Paradox for Windows 等等，这样可以节省掉许多不必要的文字。

此外，我们在书中所指的 Windows 全都是指 3.1 版本而言的。在该书中讨论的所有特性在 Windows for Workgroup 3.11 中均可运行。

然而，有些高级特性，如 OLE 2.0，虽然它需要 Windows 3.1 支持，但在 Windows 3.1 中却没有提供这个特性。OLE 2.0 是在 Windows 3.1 之后才发布的，它由 Access 2.0 之类的应用支持，只有在安装了 Access 2.0 之后，才可将其加入到 Windows 系统之中（通过使用第九章中所讲述的动态链接库技术）。

同时，读者还应该记住，在本书中所讲述的操作都是需要同时有一对应用程序才能进行的，因为只有两个程序才可能谈得上程序间的数据交换或合并运行操作。并且，在书中所述的所有的操作都应在至少有 8M 内存的机子上才能运行，如果你的系统只有 4M 或更少的内存，那么在同时运行两个大的应用程序时，就可能出现内存资源不够的情况。

数据交换和应用程序合并都是需要占用大量内存的操作，因为这种操作至少涉及到两个应用程序。而且，除此之外，Windows 动态链接库文件还需要占用附加的内存空间，以实施某些操作。

### 1.1.2 一个不可见的项目

本书所讨论的焦点内容是一组由 Windows 支持的特性，这些特性可以归纳为一个广义的范畴，即数据交换或程序合并。与其他如图标之类的 Windows 特性不同的是，数据交换或程序合并特性在没有实际对其进行使用之前是不可见的。因此，有关这些特性的内容和讨论就要复杂得多了。

举个例子来说吧，就象在书中第六章中要学到的，其他的应用程序可以通过使用一组特定的 Program Manager 指令来控制 Windows 的 Program Manager，而这些控制指令将不会显示在任何菜单中，或是列在帮助文件之内。然而，用户在安装一个新的 Windows 应用时，可以看一下这些命令是怎样进行运作的。在大多数 setup 或安装程序结束的时候，用户将能够看到一个新的 Program Manager 程序组被自动创建出来，并且自动有一组图标进入该程序组内。

随着本书一步一步的讲述，读者就可以清楚这个自动创建的过程是如何实现的，并且能够对 Windows 下的应用程序合并的结构有个清晰的认识。

### 1.1.3 本书的两个部分

这本书的目的是要帮助读者学会并且掌握一些常用的概念以及实际的操作技术。在这一章之后的其余章节内，所讲述的内容大致可以分为以下的这两个部分：

- 第一部分——第二章至第九章。这些章节的主要内容是让读者对在 Windows 应用程序间数据交换，以及多个应用程序的功能合并的过程中可能用到的所有的特性、技术以及操作方法有个总的了解。
- 第二部分——第十章至第十五章。在这六章中，我们将应用在第一部分中讲解的那些数据交换和应用程序合并的方法、技术来完成某些特定的任务以及解决某些实际的问题。

### 1.1.4 Two To Tango

在这本书的组织和内部编排上还有另外一个比较奇特的地方，那就是在书中所讨论的所有操作都是采用“two to tango”的方式——这也就是说，我们在这里所讨论的通信或集成都需要同时涉及至少两个不同的应用程序。这就意味着，在书中不可能将某个要讨论的主题限定在某个单独的程序基础上。事实上，只要是对一个单独的应用程序进行的讨论，都不可能会涉及到本书中所讲的这些特性。

事实上，正是由于这些内容具备这些特殊的性质，我们才需要编写这本书。大多数的软件手册和计算机书籍都是针对某个单独的程序而编写的（例如：Word for Windows 或者是 Lotus 1-2-3 for Windows 等等）。即使是在某书中包含了几个应用程序方面的内容，其内容也与市面上出售的单独的某种应用程序使用手册上的内容大同小异。

有一点比较麻烦的是，一旦要讨论数据的交换和合并，就必须同时对一对应用程序进行运行和操作。在本书中，我尽了自己最大的努力，希望能够尽可能多地使用现在流行的每种应用程序，并且从中摘录出各种不同的函数、宏以及在数据交换和应用合并中用到的关键的特性和概念等内容。

然而，尽管存在这些固有的复杂性，读者仍然会发觉花点时间来学习怎样同时运行两个或两个以上的应用程序其实是非常值得的。这种程序间的直接交换数据或者将某个程序中的特性合并入另一个程序中的能力，是 Windows 系统中最强大的功能，也是一个最令人兴奋的特性。

## 1.2 控制应用程序

在大多数情况之下，应用程序的数据交换或者应用程序合并操作不允许一个应用程序对另外一个程序中的操作进行控制。但是，这种功能的实现需要进行某种程序编制才能实现。在这里，我们使用了“程序编制”这样一个比较广义的术语，这是因为应用程序使用了大量的术语来描述它们用来控制程序内和程序外操作的机制。

为了让读者能够了解这些不同的专业术语是从何处而得来的，以及它们确切的含义，我们还是先来看一下这些术语最初的使用，以及后来这些术语又是怎样合并到一起的。

### 1.2.1 宏

在 IBM PC 还没有出现的日子里，大多数的程序员都是使用一种叫做汇编语言（Assembly Language）的工具来编制程序。每个计算机系统的心脏都是一块特殊的芯片，也就是我们常常说到的微处理器。在这块芯片的内部，存储有一组简单的可以让芯片进行相应工作的命令，这一组命令就叫做“译码器（Decoder）”或者叫做“命令集（Command Set）”。不管用户在计算机上使用的是哪一种程序，所有的动作最终都将分解成一系列与该计算机的微处理器的译码器组相匹配的命令。PC 机上的软件不能在 Macintosh 机上运行，其核心的原因就是这两个不同的处理器有着不同的命令组，以致于为 PC 机制做的软件对 Mac 机来讲就起不了任何的作用。

有两种方法可以使 PC 机和 Mac 机实现兼容，这两种方法在现在或者不久的将来，都会以一种形式或者其他的方式出现。实现 PC 机和 Mac 机兼容，硬件上的解决方法是在 Mac 机上面加一个 PC 机的处理器（例如，一个 Intel 486 芯片），并且在想要运行某个程序时切换到这个 PC 处理器即可。而软件上的解决办法则是设置一个专用于解释 PC 命令的程序。在当前的情况来看，上面所述的这种硬件解决方法运行起来比软件方式要快得多。

汇编（Assembler）语言采用的是直接对应于机器指令的一些命令来建立程序的，所以这种编程方式使用起来就显得非常繁琐和困难。它除了要求编程人员具有对该语言和机器等知识全面的了解之外，还要求编程人员具有极大的耐心和毅力。即使汇编程序编程员，也不能真正地将所有的命令全部掌握并成功地运行。但是，有一个简捷的办法，即设置宏。汇编程序员可以创建并将一组指令存储在此宏里，而且可以在任何时候使用它。然后，每次当要进行这个操作例行过程时，只要键入这个例程的名称，就可以激活存储在宏里面的这组指令。我们将这组存储的指令称之为宏（macro），因为这个大的例程是由一系列简单得多的指令字符串组合在一起而创建出来的。

一个宏和一条命令相比起来，它们之间存在着一个细微但又相当重要和关键的区别。一条命令是内嵌在某个实在的语言中的，而一个宏实际上来讲并不是一条新的命令，它

只不过是由某个语言中本身包含的几条命令排列组合而成的命令序列而已。

### 1.2.2　Lotus 1-2-3 和键盘宏

随着 1983 年 IBM PC 版本的 Lotus 1-2-3 系统的发明，宏的含义又有了新的扩展。Lotus 1-2-3 中有这样一种特性，它可以允许用户输入代表某条命令或一系列命令的击键组成的一段正文。用户在此之后可以将这段正文作为一条命令来激活——那就是说，该程序将会以与键盘输入指令一样的方式来读取正文字符串。

举个例子来说，为了保存一份电子表格，用户就需要键入“/fs”。这里，/＝显示菜单、f＝文件以及 s＝存盘。如果在某个单元中输入正文“/fs”，这样就能够让 1-2-3 按键盘命令的方式来读取该段正文。Lotus 将这种特性称之为宏，因为它本身与宏的概念相符，尽管是在一种更加广义的范畴之内。描述这种特性的一个更加确切的术语应该是击键宏(keystroke macros)，因为这种宏实际上并不唤醒命令，但是它能够让程序认为用户当前是正在从键盘上输入指令，而不是把这段正文当成简单的字符串来读取。

准确地来讲，在 IBM PC 以前的 Visicalc 和 SuperCalc 中，就已经有了击键宏这个概念。事实上，1-2-3 的发明人在他们开始组建自己的公司之前就是在 Visicalc 及其相关的产品上进行工作的。然而，Visicalc 和 SuperCalc 中还没有把这种特性（当时，实际上使用和学习起这种特性来是很困难的）称之为宏。Lotus 1-2-3 与其他同类产品相比起来一个大的变化就是，它将该宏作为电子表格的一部分存起来，而在 Visicalc 中则是以单独文件的方式来存储宏的。

击键宏现在已经成了 Lotus 1-2-3 中一种最普遍和最流行的特性，并且，正是由于这个特性，才使这个系统以及 IBM PC 成了当前 PC 机市场上占主导地位的机型和系统。可以毫不夸张地说，是 1-2-3 使个人电脑真正成为了商务处理工具。事实上，许多用户是通过使用宏来创建他们自己所谓的程序的，大多数真正的程序员通常看不起这种带有击键宏的电子表格。但是，这些使用击键宏的用户代表着一个全新的用户群体，他们通过击键宏来设计某些自动生成的应用程序。以我的观点来看，在 Lotus 宏与汇编语言宏之间并没有什么明确的哲学意义上的区别，它只不过是工具的复杂性的不同方面的区别而已。

### 1.2.3　宏语言

随着象 Lotus 1-2-3 之类的应用程序中击键宏的普及使用，最终导致了用户对功能更强大的编程体系的要求。用户希望在这种体系中，能够直接用击键宏来写出结构化的程序。结构化（structured）程序是一种包含有可影响指令执行顺序的指令的程序类型。在所有的编程语言或宏语言中都存在有三种基本的结构，如下所示：

- **条件式(conditional)**。一个条件是用来控制其他的一些指令实际运行与否的一条指令。击键宏的一个特征就是，一旦命令宏开始执行，它必须将宏里面的每个击键从头至尾地执行一遍。一个条件结构（如 If …Endif）制造了这样一种可能性，就是在某些特定的条件下，程序将跳过且不执行某些指令。下面的这个例子就向用户显示了一个 Lotus 1-2-3 的 If 命令。该示例程序中的条件表达式的意思是说，如果单元指针（高亮的部分）处于一个空白单元内，程序就退出（中止该宏命令）。

  {If @ Cellpointer (" type") = " b"} {QUIT}