

微软高级技术培训中心中文版系列教材之九

希望



高级篇

Microsoft 著
亦 欧 译

中文版

微软高级培训教材

FOXPRO 2.5

轻松学习

FOXPRO 2.5

Microsoft®

学苑出版社

微软高级技术培训中心中文版系列教材之九

轻松学习 FoxPro 2.5

(高级篇)

Microsoft 著

亦 欧 译

尤晓东 梁 超 审校

Microsoft

学苑出版社

(京)新登字 151 号

内 容 简 介

本书是微软大学 Microsoft FoxPro 高级培训班的高级参考读物,书中介绍了进行 FoxPro 高级程序设计的有关要点,全部例子均附在练习盘中。

本书对从事软件设计、开发和应用的技术人员具有重要的参考价值。

需要本书的用户,可与北京海淀 8721 信箱书刊部联系,邮码 100080,电话 2562329。

版 权 声 明

本书中文版由美国 Microsoft 公司北京代表处授权出版。未经出版者书面许可,本书的任何部分不得以任何形式或手段复制或传播。

微软高级技术培训中心中文版系列教材之九

轻松学习 FoxPro 2.5

(高级篇)

著 者: Microsoft

译 者: 亦 欧

审 校: 尤晓东 梁 超

责任编辑: 甄国宪

出版发行: 学苑出版社 邮政编码: 100036

社 址: 北京市海淀区万寿路西街 11 号

印 刷: 双青印刷厂

开 本: 787×1092 1/16

印 张: 26.25 字数: 622 千字

印 数: 1~5000 册

版 次: 1994 年 10 月北京第 1 版第 1 次

I S B N: 7-5077-0975-2/TP·34

本册定价: 39.00 元

学苑版图书印、装错误可随时退换

目 录

第一章 绪论	1
1.1 命名约定	1
第二章 事件驱动程序设计入门	4
2.1 应用程序模型	4
2.2 事件驱动模型	7
复习	9
复习思考题	9
第三章 事件循环	10
3.1 事件管理器——Exevmain. PRG	11
复习	18
复习思考题	18
第四章 形式化对话	19
4.1 建立 READ CYCLE(数据录入)对话	19
4.2 建立形式化数据录入对话	23
复习	25
复习思考题	25
第五章 共存对话	26
复习	28
复习思考题	28
第六章 窗口可视化	29
6.1 保持对话窗口可视:RELEASE WINDOWS	29
6.2 窗口注册栈	31
6.3 窗口移动(READ DEACTIVATE)	36
6.4 重新设置事件请求标志	37
复习	38
复习思考题	39
第七章 再激活先前的对话	40
7.1 为历史栈增加一个对话	40
7.2 从历史栈中删除对话	45
7.3 再激活先前对话	48
复习	49
复习思考题	49
第八章 记住窗口的位置	51
8.1 保存屏幕位置	51

8.2 恢复窗口位置.....	54
复习	57
复习思考题	57
第九章 记住对话变量	58
9.1 创建一个用户专用的系统表.....	58
9.2 恢复值.....	60
9.3 保存或取消对话值.....	62
复习	64
复习思考题	64
第十章 关闭所有对话	65
10.1 请求关闭一个对话	65
10.2 自关闭对话	68
复习	69
复习思考题	69
第十一章 控制 BROWSE	70
11.1 将 BROWSE 集成到一个对话中	71
11.2 BROWSE 和 BROWSE 窗口	73
11.3 新对话还是 BROWSE	75
11.4 保存和刷新 GET	77
11.5 BROWSE 和改变中的对话	78
复习	79
复习思考题	79
第十二章 数据录入体系结构	81
12.1 录入时显示	81
12.2 防止编辑	82
12.3 创建一个编辑对话	84
12.4 退出一个编辑对话	87
12.5 消息处理(13msg)	88
12.6 录入时编辑	91
复习	92
复习思考题	92
第十三章 多用户结构	93
13.1 SET EXCLUSIVE OFF/ON	93
13.2 RLOCK()	94
13.3 SET REPROCESS TO	95
13.4 UNLOCK	96
13.5 SET MULTILOCKS ON/OFF	96
13.6 数据现时性	98
13.7 文件	98

13.8	FLOCK()	99
13.9	UNLOCK	99
13.10	记录	99
13.11	SET REFRESH TO	99
	复习	100
	复习思考题	100
第十四章	多用户编辑	101
14.1	主动与被动锁模式	101
14.2	主动模式结构	101
14.3	被动模式结构	103
	复习	106
	复习思考题	106
第十五章	多用户增加	107
15.1	建立一个 ADD 对话	107
15.2	唯一标识符和记录增加	108
15.3	结束一个 ADD 对话	118
	复习	119
	复习思考题	119
第十六章	多用户删除	120
16.1	创建一个 DELETE 对话	120
16.2	参照完整性	123
16.3	重用与消除删除的记录	124
16.4	PACK	125
16.5	COPY TO	126
	复习	126
	复习思考题	127
第十七章	多用户终止器	128
17.1	READ TIMEOUT	128
17.2	READ VALID	129
17.3	READKEY(1)	130
	复习	130
	复习思考题	130
第十八章	多表处理	131
18.1	入门	131
18.2	设置代码	133
18.3	临时表	136
18.4	更新屏幕信息和值	138
18.5	移动记录指针	144
18.6	定单头编辑	144

02	18.7 定单头增加.....	162
08	18.8 删除.....	170
00	复习.....	172
00	复习思考题.....	172
	第十九章 嵌入 BROWSE	173
001	19.1 嵌入 BROWSE	173
101	19.2 BROWSE 和 READ	186
101	19.3 嵌入 BROWSE 总结	187
101	19.4 BROWSE 杂集	188
001	复习.....	189
001	复习思考题.....	190
	第二十章 事务跟踪系统	191
001	20.1 适当管理方法.....	191
101	20.2 NetWare. FLL/PLB(Novell 的 TTS).....	193
001	20.3 远程登录服务器.....	195
011	20.4 版本号.....	196
011	复习.....	197
011	复习思考题.....	197
	第二十一章 远程服务器	199
001	21.1 服务器需求.....	199
001	21.2 为一个请求服务.....	200
101	21.3 登录一个请求.....	203
001	21.4 客户通知.....	204
001	复习.....	204
001	复习思考题.....	204
	第二十二章 对象连接和嵌入	205
001	22.1 综合字段.....	205
001	22.2 Windows 对象数据录入.....	210
001	22.3 Windows 数据录入窗口.....	211
001	22.4 Windows 数据录入:综合字段.....	212
001	22.5 图片控制.....	221
001	22.6 图片对象和 Report Writer	223
101	复习.....	225
101	复习思考题.....	225
	第二十三章 动态数据交换	226
001	23.1 DDE 作为一个会话.....	226
001	23.2 DDE 命令.....	227
101	23.3 与 Word 的约定.....	228
101	23.4 与 Microsoft Excel 的一个会话.....	229

172	23.5 FoxPro 作为 DDE 服务器——从 Microsoft Word 调用	232
872	23.6 FoxData: 一个一般意义上的 DDE 服务器	234
772	复习	235
872	复习思考题	235
	第二十四章 高级 SQL-SELECT	236
502	24.1 子查询	236
202	24.2 查询条件	237
602	24.3 存在性测试	238
102	24.4 数量测试	239
702	24.5 复杂查询	241
802	24.6 外部联接	248
502	复习	249
502	复习思考题	249
	第二十五章 API 与 C 语言	250
802	25.1 创建一个 API 函数	250
302	25.2 编译和连接代码	252
	25.3 在 FoxPro 中挂接一个库	252
	25.4 使用 API 提高性能	252
	25.5 FoxPro for Windows: FLL 文件	253
	25.6 通过 FOXTOOLS. FLL 访问 Windows API	254
	25.7 访问任何 Windows 函数或 DLL	255
	复习	255
	复习思考题	255
	第二十六章 数据字典	256
	26.1 数据字典	257
	26.2 13dict-MEI 数据字典	257
	复习	262
	复习思考题	262
	第二十七章 高级索引技术	263
	27.1 简单的降序索引	263
	27.2 降序日期	264
	27.3 降序数据	264
	27.4 降序字符	265
	27.5 嵌入 UDF	266
	27.6 嵌入式 IIF	267
	复习	267
	复习思考题	267
	第二十八章 高级性能的问题	268
	第二十九章 低级文件 I/O	271

29.1 低级文件 I/O 函数	271
复习	276
复习思考题	277
第三十章 跨平台问题	278
30.1 转向跨平台	279
复习	293
复习思考题	293
第三十一章 项目管理和分配	294
31.1 项目管理器(Project Manger)	294
31.2 FoxPro 应用程序	297
31.3 FoxPro 2.5 可执行程序	298
复习	302
复习思考题	302
第三十二章 FoxPro 2.5 快速参考	303
32.1 FoxPro 命令和函数快速参考	306
32.2 系统内存变量快速参考	406

第一章 绪 论

1.1 命名约定

在本章开始时,需要了解一下贯穿全书的命名约定。

1. 应用程序(Applications)

隶属于一个给定的应用程序的所有表、程序、屏幕、菜单以及报表都以相同的两个字符开头。这些字符就是这个应用程序的“调用字母”。

例如,你会发现本书示例的应用程序中的所有表都是以“ex”开头的(由于“example”的缘故)。

“调用字母”的作用是双重的。首先,由于所有的应用程序部件的开头字符(“EX”)都是一样的,因而,可以很容易地使用简单的“ex*. *”备份整个应用程序。其次,我们可以很快地指出谁属于这个应用程序,而不必再查找其他任何目录项。

2. 表(Tables)

在给一个表命名时,头两个字母是为应用程序调用字母保留的,剩下的六个字符用来标识内容。

例如,存有客户数据的表的名字可以是“excust”。

3. 字段(Fields)

一个表中的字段的命名既是为了标识该表,也是为了标识该字段的内容。

正如应用程序调用字母一样,每个表都有一个两字符的唯一标识符。标识符后面紧跟的是一个下划线,它的作用相当于一个空格,因为中间嵌入空格是不允许的。剩下的七个字符用于表示这个字段的内容。

例如,客户表中的所有字段都以“cu_”开头。字段“cu_custid”表明这个字段的内容是来自客户表的客户标识码。

这些字段级的命名约定的好处如下:

首先,我们可以立刻知道这个字段所在的表,而不必去利用数据字典或列出所有的表以及它们的结构。

其次,如果我们对剩下的有效字符利用得好的话,就可以知道这个字段的内容,而不必再在整个文件或数据字典中查寻。

第三,我们可以利用这个命名约定去表明来自不同的表的那个字段是相关(或外部)键字段。

例如,通过将定单表(exordh)中的客户标识码命名为“oh_custid”,就可以很明显地看出

字段 `cu_custid` 和 `oh_custid` 包含的信息是相同的以及它们分别来自的表。

第四, 由于使用这个字段命名约定保证了明确的字段名, 因此我们就可以在写一个 `SQL-SELECT` 时, 直接写这个字段, 而不必加表的别名前缀。

例如, 我们能简单地通过写“`oh_custid=cu_custid`”将客户表和定单表连接起来。如果我们仅仅使用“`id`”作为字段名, 那我们将不得不写成“`excust.id=exordh.id`”。

最后, 我们在使用 `SCATTER` 和 `GATHER MEMVAR` 时不会出问题。由于这些命令都是基于这样一个前提, 那就是变量与字段是匹配的, 因而如果我们打算 `SCATTER` 多个表时, 我们必须使字段名唯一。如果字段名不唯一的话, 那么当把这些字段信息 `GATHER` 回记录时, 将不能保证正确的字段值放到了正确的表中。

假定我们为客户文件中的客户 `id` 和定单文件中的发票 `id` 使用了字段名“`id`”。如果我们将两个表都 `SCATTER` 了, 那么最后一个 `SCATTER` 的将是 `m.id` 的值。然而, 如果我们现在 `GATHER` 到第一个表中, 那么“`id`”就会被一个错误的值所代替。

另一方面的益处就是在决定程序引用的“`id`”是客户的, 还是定单的时, 不必再查阅大量的代码。

总而言之, 合适的字段命名可以提高的应用程序的维护速度和新应用程序的开发速度。

4. 标识(Tags)

在简单的一个字段表达式上创建的标识(Tags)的名字就可以是这个字段的名称(例如, 如果主要基于 `cu_custid` 字段的话, 那么标识的名字就是 `cu_custid`)。这就使识别标识的根据更容易了。

当使用复杂表达式(表达式基于多个字段, 字段的部分或任何一个有效的 `FoxPro` 表达式)时, 要尽可能给出一个名字。

5. 内存变量(Memory Variables)

`FoxPro` 环境中有两种类型的内存变量, `PUBLIC` 和 `PRIVATE`。知道变量的类型是很重要的。

6. Public(公共)

`Public` 变量在应用程序中的任何一个地方都是有效的, 直到它们被明显地释放掉。由于这个缘故, 因而就可以很容易地通过给这个名字赋一个新值而偶然地改变这个变量的值。

一个公用的变量名可以是“`action`”。如果使用这个变量去决定运行哪个程序(最终用户选择的系统“`action`”)的话, 我们必须去维持这个值, 直到又有一个新选择出现。因而我们可以定义一个 `PUBLIC` 变量, 并且现在可在任何一个程序中访问当前系统选项。

但是, 我们也许还想利用变量“`action`”去保存程序中的当前活动的值(例如增加)。一个程序中变量“`action`”值的改变同样会引起当前系统选项值的改变。这样, 系统就会混乱起来。

为了避免这种变量值的冲突, `MEI` 在所有的 `PUBLIC` 变量前都加了一个“`K`”。这样, 系统变量“`action`”就变成了“`Kaction`”。

7. Private(私有)

Private 变量意味着它仅在一个给定的程序或过程内才存在。一旦这个程序或过程终止了,该变量就不存在了。

由于在 FoxPro 中,缺省情况下,变量被定义成 PRIVATE,因而我们就能在一个应用内反复使用变量名,并能保证在每个程序或过程内我们都拥有当前程序的值,且不会与任何其他同名变量发生冲突。

但现在我们必须解决一下 PRIVATE 变量的偶然覆盖写的问题。

我们将通过使用 PRIVATE 命令完成它,在 PRIVATE 命令后面紧跟的是对该局部过程/程序而言完全私有的变量表(例如 PRIVATE ans)。

现在,如果我们在一个过程中使用了相同的变量名(PRIVATE ans),FoxPro 将保持有两个不同的“ans”变量。当退出这个过程后,只有在这个例程中创建的“ans”变量才会被释放掉,另一个仍然存在,且仍保持原有的值。

这种类型的变量有时也称作“局部”(local)变量。为了方便在一个 PRIVATE 语句中命名局部变量,MEI 使用下面的约定:

所有的局部变量都以 L_ 开头。PRIVATE 语句变成:PRIVATE ALL LIKE L_*。

我们现在可以在程序或过程的开头使用这个声明了,并且自信不会危害到调用树中较高层的同名变量的值。

最后,如果我们想让一个变量被多个程序使用,但又不想在应用结束时显式地释放它,我们就可以给它任一个命名,只要它不以一个“K”或“L_”开头。

第二章 事件驱动程序设计入门

本章要点:

- 应用程序模型
- 事件驱动组件

本章目的:

完成本章后,你将会:

- 应用程序设计模型间的差异
- 事件循环(Event Loop)工具
- 事件(Event)
- FoxPro 触发器(Trigger)
- FoxPro 对话(Session)
- FoxPro 标志(Flag)

2.1 应用程序模型

本小节目的:

当你完成本小节的学习后,你会理解下列程序模型间的差异:

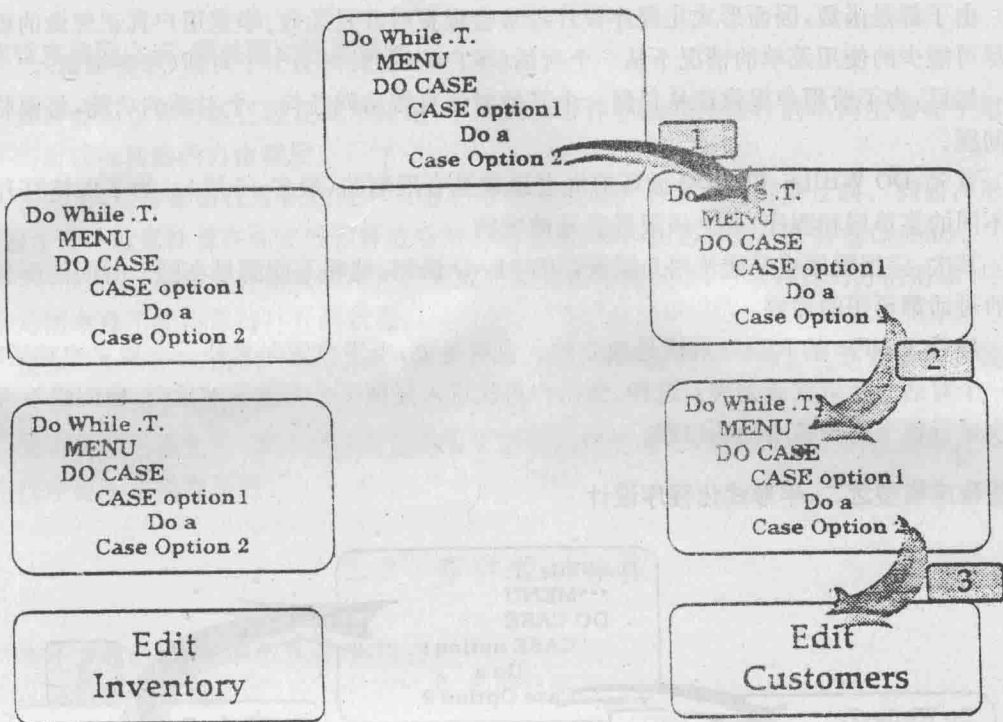
- 形式化应用程序(Modal Applications)
- 半形式化应用程序(Semi-modal Applications)
- 无模型应用程序(Modeless Applications)

在本节开始前,有必要先对事件(event)和对话(session)作一下定义。

一个“对话”就是一个独立的程序,比如编辑客户信息(如 excust)。

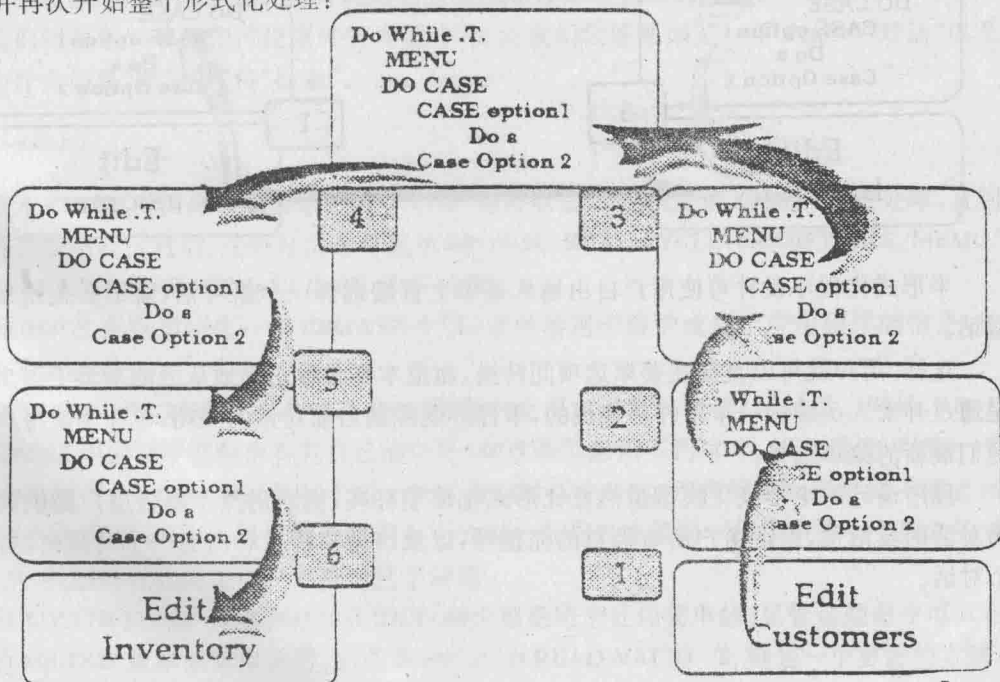
一个“事件”就是系统接收到的一个信号,它表明用户请求在当前对话中作一个更新。当“事件”被处理时,用户转向一个新的对话,此时不必关闭任何一个先前打开的对话,也不会从任何一个先前打开的对话中丢失信息。

应用程序模型之一:形式化程序设计



最早的程序设计模型是形式化模型。当你编写一个形式化应用程序时，用户必须通过一系列的菜单选择才能开始一个请求的对话(如编辑客户信息)。

为了选择另一个事件，比如像编辑库存，用户必须正确地当前的对话中退出，然后从一系列的菜单中返回，这时程序才仅仅到达主菜单。在那点上，用户才可以选择另一个对话选项并再次开始整个形式化处理：



由于都是函数,因而形式化程序设计经常会很费时并且无效。毕竟用户真正想做的就是尽可能少的使用菜单的情况下从一个对话(客户编辑)转向另一个对话(库存编辑)。

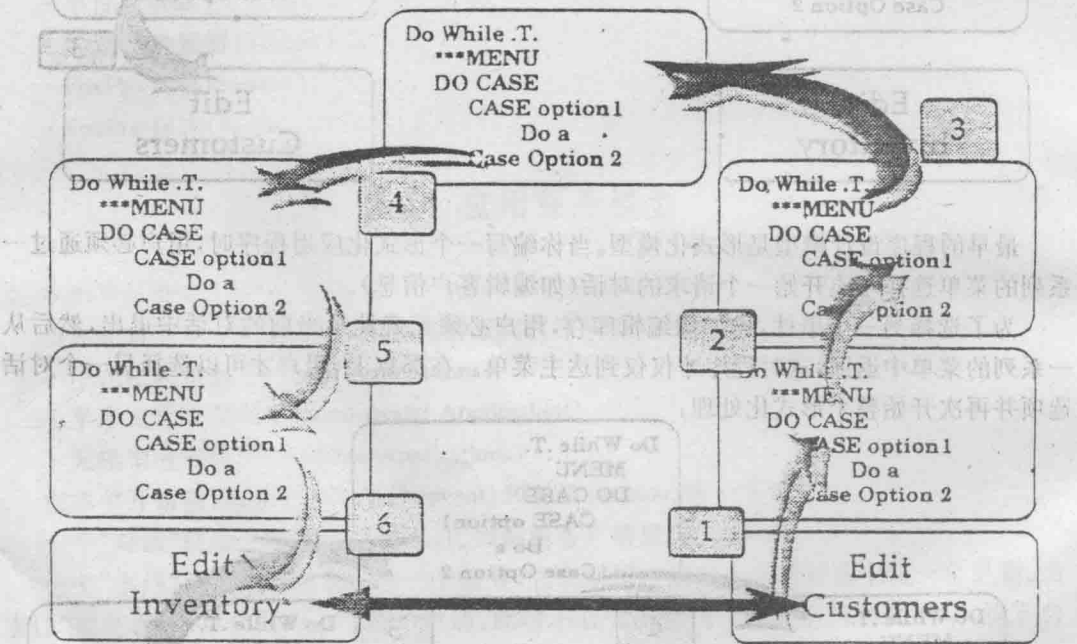
然后,为了给用户提能从一个其他对话直接访问任何一个对话的功能,将面临许多问题。

首先,DO WHILE/ENDDO 循环的嵌套层数是有限制的(最多 16 层)。为了保持打开数个不同的菜单层和程序,这个极限很容易被突破。

其次,应用程序也只能使用 5 层嵌套的 READ 语句。这将不能满足在同一时间使所有用户的对话都可用的需要。

最后,FoxPro 的 READ 对话是独立的。也就是说,当用户转向另外一个独立的对话时,前一个对话的变量就会丢失。这样,当用户再次进入任何一个先前的对话时,他们将不得不再次进行整个更新和增加的过程。

应用程序模型之二:半形式化程序设计



半形式化程序设计可使用户自由地从菜单上直接选择一个新对话,而不必关闭当前的对话。

这样,用户就可以直接在菜单选项间转换,而根本不必管正确地当前对话中退出。这是通过开发人员维护一个事件环实现的,事件环既控制当前对话的关闭,也控制使用户转向他们最新的菜单选择。

程序设计的半形式化模型虽然要比形式化模型好些,但它仍然不能为用户提供离开先前对话的灵活性,保证用户所有信息的完整性,以及当用户请求转向另一个对话时,打开这个对话。

应用程序模型之三:事件驱动程序设计

到目前为止你可能已经发现形式化,半形式和事件驱动程序设计的不同主要在于用户在不同对话间转换的自由程度。

实际上,这种自由程度在这些不同的应用程序模型的名字上已有所反映。例如,“形式化”程序设计就意味着在你完成和释放当前对话之前,你不能选择另一个行为(action)。

“半形式化”程序设计则表明用户没完成当前对话的情况下,可以选择另外的对话,尽管这个对话本身不能再保持打开的状态。

最后,事件驱动,或称“无模型化”程序设计,则为用户提供了一种直接在对话间转换并保证数据不受损害的能力。

你可能也已经发觉,如果没有大量的关于所要组件的考虑,要建立一个真正的事件驱动应用程序也不是很容易的。

2.2 事件驱动模型

本小节目的:完成本小节后,你将会:

- 事件驱动对话

- 事件驱动“触发器”

- 事件驱动“标志”

事件驱动应用程序“组件”(“component)包括:

- 一个处理对话请求的事件管理器

- 允许用户在对话内请求事件的触发器

- 应用程序用于维护各种打开的对话并辅助事件请求处理的标志。

我们将在下一节详细讨论事件管理器,因此让我们快速地浏览一下 FoxPro“对话”以及我们为什么需要“触发器”和“标志”。

对话(Session)

基本 FoxPro 对话命令都使系统处于一种“等待状态”。也就是说,系统会暂停处理,直到用户终止或中断该对话。这些对话命令包括 BROWSE, READ CYCLE, MODIFY FILE/MEMO/REPORT, ACTIVATE MENU 以及 READ VALID . T. 。

BROWSE 命令初始化一个 BROWSE 对话,该对话用于维护或显示应用程序的信息,比如像客户的信息。

正如 BROWSE 对话一样,READ CYCLE 用于使用户进行一个读/写对话。但是,READ 和 BROWSE 中的每个都包含有它自己的中断,并且谁都意识不到对方。换句话说,尽管一个 BROWSE 和一个 READ 可以共存于同一个普通的对话中(比如厂商维护),但谁都不能直接与对方通信。确实,开发人员必须把它们当中的每一个都当成是同一程序中的一个独立的对话,但是,我们将在后面更详细地谈论这个问题。

ACTIVATE MENU 和 READ VALID . T. 命令都是用于处理菜单的,尽管这些命令可以在产品的 MS-DOS 版本中交替使用,但在 Windows 中 READ VALID . T. 则是一个更好的方法,

ACTIVATE MENU 命令在这种环境功能要弱些。

一旦用户处于这些对话中的某一个,我们就必须依靠各种各样的中断这个途径给事件管理器传递事件请求。这些中断就是“触发器”。

触发器(Trigger)

无论何时当遇到一个“触发器”或对话中断时,就可以请求事件了。

一个“全程”(“global”)触发器是独立于特定的“等待状态”的,它用 ON KEY LABEL 来定义。由于 FoxPro 是在内循环解释时间(如在执行代码行之间)检查 ON KEY LABEL 的,因而用户可在任何时间“触发”一个事件请求。

ON KEY LABEL 仅在一个特定程序内还可以有效地使用。当使用这个方法时,ON KEY LABEL 用于执行程序的主控过程。一个“全程”ON KEY LABEL 要比一个程序的特定触发器更常用。

由于 OKL 都是在命令之间被激活的,因而有一个小问题(尤其是在最终用户那里)。

假定我们用一个 OKL 做增加新记录的处理,代码如下:

```
ON KEY LABEL F3 Do addproc.
```

当用户按下 {F3} 后,程序控制权就传给了过程 ADDPROC。但是,由于 FoxPro 仍需要做所有的内循环解释工作,因而执行也就没有开始。用户则认为缺乏响应是由于按键失败造成的,因而再次按下 {F3} 键。

程序控制权再次被传递了,同时也引起了程序栈内的一个递归调用。

最终,用户将反复按键以致于在递归调用中产生了一个 FoxPro 错误。

为了在 OKL 中避免这个问题的出现,可以使用下面的代码:

```
ON KEY LABEL {KEY} mvar=;  
IIF(PROGRAM() == "MAIN. PRG", myproc(), "")
```

其中:

{Key} 是一个 OKL 键(如 F2)

mvar 是任何一个内存变量的名字

“MAIN. PRG”是 PRG 的文件名

myproc() 是要执行的过程的名字

执行情况如下:

- 用户按下 OKL 键。
- PROGRAM() 返回主控 PRG 的名字,这样程序控制权就传给了过程 MYPROC()。程序执行尚未开始。
- 用户再次按下了 OKL 键。
- PROGRAM() 返回当前处于控制中的程序(如 MYPROC()) 的名字,这与主程序的名字不一样,因而从 OKL 返回的将是一个空串(“”)。
- FoxPro 只是忽略这个返回的空串并且当下一个命令准备好处理时,将执行 MYPROC() 中的第一行代码。

在一个菜单内,ON SELECTION POPUP 命令用于控制把请求事件分配给一个“标志”,这个标志然后就传给事件管理器。