第一章 简 介

本书是微软高级技术培训中心中文版系列教材之一，书中介绍了 FoxPro 中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。

本课程所需教学材料包括：

- 一本教科书(本书)
- 样例程序的源程序代码清单
- 练习的源程序代码清单
- 附有样例程序、库例程和练习答案的磁盘

第二章　入　门

本课目的

- Sales Rep Entry(单表数据入口)。
- Customer List(单表 SQL SELECT 和多栏式报表)。
- 一组客户的 Sales Orders,包括平均订单量和总订单量(该报表需要使用报表变量和 UDF)。

注:在我们不断探讨若干新概念的时候,我们总是要回到代码应用程序,来看看这些新概念在实际应用程序开发中是如何被应用的。

本课的目的是向学员提供开发可运行的、多用户的 Microsoft FoxPro 2.5 应用程序所需的基本技能。为了编写 FoxPro 的应用程序,程序员必须掌握各种对象(如 READ,BROWSE,MENU,REPORT 等)和把这些对象联系到某一应用程序中的命令。

本课中所用到的应用程序是一个非常简单的 Sales Order Entry 系统,它包括下面几个模块:

- Customer Entry
- Inventory Entry
- Sales Order Entry

2.1　命名规则

本节目的

- 命名应用程序
- 命名应用程序中的表
- 命名表中字段
- 命名表中标识(索引)
- 区别 PUBLIC 和 PRIVATE 变量
- 应用程序设计概述

通常我们在现有项目基础之上继续开发,或者经过相当长的一段时间又回到该项目,所以建立常用开发模块命名规则是很重要的。

"命名规则"让读者看到名字就知道对象的实质。这样能快速看到旧的程序或其他人的源代码。

下面所讲的命名规则是 Micro Endeavors 所用的主要规则。其后再介绍其他规则。教程

的最后部分有所有命名规则的完整列表。

应用程序

所有属于某一应用程序的表、程序、屏幕、菜单和报表都以相同的两个字符开头，这两个字符是应用程序的“呼叫字母”。

例如，在课程示例应用程序中的所有表都以“ex”开头。

“呼叫字母”的用途有两点。

首先，因为应用程序的所有元素都以相同字符开头(“EX”)，所以简单的“ex*.*”就能备份全部应用程序。

第二，能立即区别出某一元素属于哪一个应用程序，而不需要看它的目录。

表

命名一个表时，前两个字符是该应用程序的呼叫字母，剩下的6个字符用来表明表的内容。例如存储客户数据的表叫“excust”。

字段

表中字段的命名要能同时表明表和字段内容。像应用程序呼叫字母一样，每一个表也有两个字符的指示器。指示器后面带有一个下划线，剩下的7个字符代表字段的内容。

例如，客户表中所有字段以“cu_”开头。字段“c_custid”表明该字段内容是客户标识码。

字段级命名规则有几个优点：

第一，立即辨别出某一字段是属于哪一个表，而不需使用数据目录或列出所有表和它们的数据结构。

第二，如果剩余字符命名工作做得好以后，可以判断出字段的内容，而不用查看文档解释和数据字典。

第三，表明各个不同表中哪一些字段是关系关键字。

例如，在订单表(exordh)中命名客户标识码字段“oh_custid”，那么我们很快就看出字段cu_custid和oh_custid包括相同信息，以及它们分别属于的表。

第四，因为这种字段命名规则确保不会出现字段名模糊性，所以在写SQL_SELECT时，不需用表别名做前缀就能说明字段。

例如，“oh.custid=cu_custid”就能连接客户和订单表，如果只用“id做字段名，那么就必须写“excust.id=exordh.id”。

最后，在使用SCATTER和GATHER MEMVAR时不会出现问题。因为这些命令是基于变量和字段匹配的前提上，所以在SCATTER若干个表时字段名唯一。但是如果字段名不唯一，那么在GATHER字段信息回记录时，就不能保证把正确的字段值放入正确的表中。

假设我们在客户文件中使用“id”代表客户标识码，在订单文件中使用“id”代表发票号，和SCATTER两个表，最后SCATTER的是m.id值，但是如果要GATHER第一个表，那么“id”的值就会是错的。

最后还不需阅读大量的代码来判断哪一个程序引用的是哪一个“id”。

综上所述，正确的字段命名可以提高出应用程序的维护性。

标识

以单一字段表达式建立的标识以字段名命名(例如 cu_custid 是基于单一字段 cu_custid 的标识)。这样区别标识就很容易。

使用复杂表达式时(表达式基于若干个字段,字段的一部分或有效的 FoxPro 表达式)尽量起一个有说明性的名字。

内存变量

FoxPro 环境中有两种内存变量,PUBLIC 和 PRIVATE。了解变量的实质是很重要的。

公用变量

PUBLIC 变量在被释放前在应用程序的任何部分都可以使用。所以重写变量并赋给一个新值就能很容易地改变它的值。

公用变量名必须是"动作"。如果使用它来判断要执行哪一个程序(系统"动作"由最终用户选择),那么在选择之前必须维护该值。

然而,我还想使用变量"动作"来保持该程序(如 add)内当前活动值,这样改变程序内变量"动作"值就会改变当前系统选择值,这样系统变糊涂了。为了避免出现这种系统值混乱,在 PUBLIC 变量前加上"K",这样系统变量"动作"变成了"K 动作"。

这种命名规则能够立即从程序指出 PUBLIC 变量。

局部变量

PRIVATE 变量只能用在某一程序或过程中,只要程序或过程被调用,那么变量就存在了。

因为 FoxPro 缺省是说明变量是 PRIVATE,所以在一个应用程序中可以重复使用变量名,也能保证每一个程序或过程内有当前程序值,但不影响其他程序或过程中具有相同名字的变量。

现在我要解决意外覆盖 PRIVATE 变量的问题。

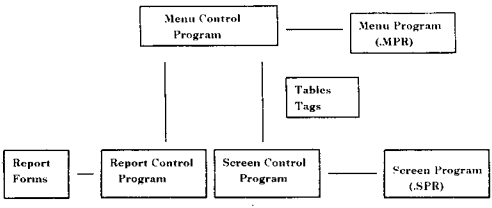
我在 PRIVATE 命令后面写了一串某一过程/程序的局部变量。(如 PRIVATE ans)。

现在,在另一个过程中使用相同的变量名(PRIVATE ans),FoxPro 将维护两个不同的变量"ans"。退出某个过程时,只有在这个过程中建立的"ans"被释放,其他过程建立的仍然存在,并保持各自的值。

这种变量通常从 l_ 开头,并在程序和过程的开头说明,它们不会与上级调用模块中相同名字的变量冲突。

如果变量被若干个程序使用,但不需在应用程序结束时释放它,那么它的名字不能以"a"和"l_"开头。

2.2 FoxPro 2.5 程序设计



大多数的开发人员都有一套开发应用程序的方法，MEI 使用下面的方法：

- 设计菜单系统
- 设计菜单控制程序
- 设计表和索引标识

本教程是中级教程，所以不讨论理论级（设计 DBMS）和实用级（命令语法）的表和标识的建立：

- 设计输入屏幕
- 设计屏幕控制程序
- 设计报表/输出

上述每一步骤都包含许多编程概念和 FoxPro 命令。

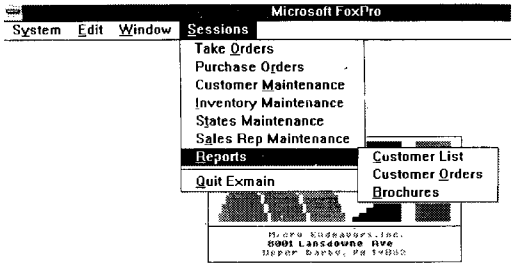
2.3 复习

- 命名规则
- 应用程序设计

2.4 复习题

1. 什么是应用程序的“呼叫字母”，使用它们有什么好处？
2. 交叉表中字段命名唯一有什么好处？
3. 如何区别 PUBLIC 和 PRIVATE 内存变量？
4. 设计应用程序的步骤是什么？

第三章　菜　单



内容概要

- 用 Menu Builder 建立和修改菜单
- 热键和简化键
- Menu Builder Meta 字符集
- 菜单动作
- 菜单 READ
- QUICK MENU
- FoxPro 系统菜单笺

课程目的

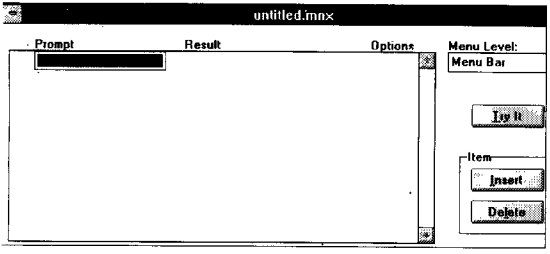
本课中将学习：

- 建立一个工作菜单
- 提供菜单热键和简化键
- 使用 Quick Menu
- 在读者应用程序中使用 FoxPro 菜单选项

3.1 建立菜单

本节目的

- 使用 Menu Builder 建立和修改一个 FoxPro 2.5 菜单
- 建立单一和多层弹出式菜单
- 建立菜单和弹出式热键和键盘简化键
- 使用 Menu Builder Meta 字符集



FoxPro 提供了建立菜单和弹出式(某一菜单笺的菜单选项列表)的不同方法。

在本课中我们只学习 System Style Pull Downs(SSPD)技术，Menu Builder 构造这种类型的菜单。

在 CREATE MENU 命令已存在时，发 MODIFY MENU 命令初始化 Menu Builder。

如果命令后的菜单已存在，Menu Builder 打开该菜单，如果菜单不存在，那么这条命令被视为 CRIATE MENU 命令。本课中我们键入：

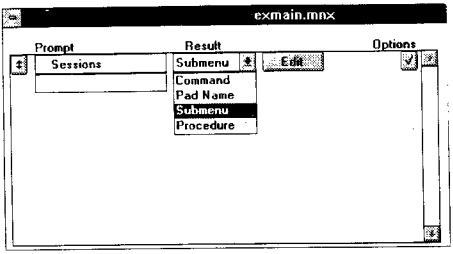
```
MODIFY MENU exmain
```

在不能确定要被修改菜单的名字时，键入下面命令：

```
MODIFY MENU?
```

问号产生 Select Menu 对话框，其中显示当前驱动器和目录下所有菜单名。单击要修改的菜单后，屏幕上出现准备修改的菜单，如果没有匹配的菜单，单击(new)建立新的菜单。

现在到了 MENU BAR 级，Menu Builder 的提示符出现在主菜单条上。FoxPro 菜单包括许多菜单笺，像 HELP，PROGRAM 和 EDIT。



如果在 exmain. mnx 中建立菜单笺，不会有什么结果 因为菜单笺没有选项。

Prompt 栏旁边是 Submenu(Submenu 也叫 Popups)，在菜单笺有缺省的弹出式菜单时，可以通过菜单笺立即执行 Commands 或 Procedures。

下面我们选择 CREATE 建立 Sessions 菜单笺的弹出式菜单。

在这张表上，输入与 Sessions 菜单相关的任选项，有“Take Orders”，“Customer Maintenance”，“sales Rep Maintenance”，“States Maintenance”，“Inventory Maintenance”，“Reports”和“Quit”。然后选择(Try it)从菜单中选择不同的任选项试着看菜单和弹出式。

另外，Sessions 子菜单中的 Reports 提示符还要有一些事情要做，因为有若干张表，所以 Reports 要建立一个子菜单。

为菜单弹出式任选项建立弹出式的技术叫做嵌套弹出式或“多层”弹出式。

读者可能已经注意到了，Session 菜单笺替代了 FoxPro 系统菜单，这样向已存在的菜单增加菜单笺就很容易，操作步骤如下：

- 从系统菜单选择“Menu”菜单笺
- 选择“General Options…”任选项
- 选择 Location 框中的 Append 按钮。

使用 PROGRAM 菜单 GENERATE 任选项生成菜单程序，这样菜单就可放在屏幕上。

执行生成任选项时，建立下列文件：

Exmain. MNX	包含菜单版面信息表。
Exmain. MNT	与 MNX 有关的内存字段。
Exmain. MPR	genmenu. prg 根据 MNX 中信息生成的程序，发出 DO 菜单名. MPR 命令才能执行该程序(如 DO exmain. MPR)。
Exmain. MPX	DO . mpr 时才生成这个扩展文件，它是菜单程序(MPR)的编译版本，FoxPro 真正使用的是它，就像 FoxPro 运行程序时使用 FXP 而不是 PRG。

练习

- 修改 exmain 菜单
- 建立 Sessions 菜单笺
- 建立 Sessions 菜单笺的弹出式，包括下列任选项：
  - Take Orders
  - Customer Maintenance
  - Sales Rep Maintenance
  - States Maintenance
  - Inventory Maintenance
  - Reports
  - Quit
- 建立 Report 子菜单，包括下列任选项：
  - Customer List By State
  - Customer Orders
  - Brochures
- 使用(Try it)按钮
- 把 Sessions 加到 FoxPro 主菜单中，但不替代它
- 生成菜单

MODI COMM exmain. mpr 命令可修改程序 exmain. mpr，下一次的 MODIFY MENU 和重新生成会覆盖这次所写的程序。关闭 Menu Builder 窗口，DO exmain. mpr 运行新菜单。但是这时我们需要回到 FoxPro 系统菜单：

SET SYSMENU TO DEFAULT

这个命令的重要性不仅在于可返回到 FoxPro 系统菜单，当读者的菜单是活动的时候，可能丢掉了许多原有的键盘简化键。

例如，这时 CTRL+F2 键不再产生 Command 窗口，帮助键 F1 也不再工作。总之，如果在读者的菜单是活动的时候，程序"失败"了，这时必须发出 SET SYSMENU TO DEFAULT 恢复所有 FoxPro 开发工具。

下面我们看一下 exmain. mpr 程序的当前状态。

在程序开发过程中可向原 FoxPro 系统环境增加读者自己的任选项。

Sessions 菜单笺没有热键也没有简化键。

从 Sessions 的下拉菜单中选择任何一项没有执行什么程序。

下面我们对 exmain. mpr 做一些修改：

3.1.1 热键和简化键

首先向 Sessions 菜单笺增加\<，成为"Sess\<图标"。\<是 Menu Builder 三个可用的 meta 字符之一，它使得字母"I"在"Sessions"中高亮，高亮的字母叫做"热键"。支持 FoxPro 2.5 的 Windows 版本是自动配置好的，称为 CUA(Common Architecture)，热键与 ALT 键一起使用建立简化键。

但是在 FoxPro 2.5 DOS 版本下这样做不行，必须按下列步骤增加简化键：

- 单击"Sessions"提示符旁的 Options[]
- 在 Options 对话框，选择[] Shortcut…复选框，按键 ALT1 然后选择"OK"。

现在热键(高亮显示的字母)和简化键(与某一任选项有关的击键动作)都建立了，它们之间的区别如下。

简化键不考虑当前在屏幕的什么地方，只要击键就能触发相关动作。

例如，无论在程序中，Menu Builder 中还是对话框中，F2 总是产生 Command 窗口，而 F1 总是带来 FoxPro Help 系统。所以，最终用户的弹出式任选项由简化键操作是很方便的，简化键在系统的任何地方都能执行。

热键则在它被显示在屏幕上后才能触发执行。例如，只有屏幕上出现 Sessions Pad "1"才能激活菜单笺，但若在 Command 窗口中敲入"1"，Command 窗口中只是一个字母"1"。

3.1.2 Menu Builder Meta 字符

在建立谈者自己的菜单之前，首先介绍一下其他的 Menu Builder Meta 字符：\和\-。

\使菜单和菜单任选项不再活动，通常我们不在 Menu Builder 中使用这个符号，而是要么执行 Menu Builder Options 对话框中的 SKIP FOR 复选框，要么禁止菜单上的某些任选项。这两种方法都是让我们根据某些条件禁止或激活菜单和任选项。后面我们将详细讨论这部分内容。

\-在菜单任选项之间放分隔条(或线)，它用来对菜单任选项进行逻辑分组，而不是再建立另一个菜单笺。例如，在我们的菜单上，QUIT 与其他任选项需要分开，因此在 QUIT 线上写\-。

最后出现 OPTIONS 对话框，检查 PAD NAME。为什么要检查呢？我们只要看看菜单程序 exmain.mpr，就能看到 FoxPro 已经给 Sessions 菜单笺一个唯一的名字_PDZ4WY230。

当然我们不需要知道这个名字，不考虑 FoxPro 的缺省，而让菜单生成器使用我们所识别的菜单笺名字，必须要检查 OPTIONS 对话框，在这个对方框中，选择 PAD NAME，由程序员指定 Pad Name，名字最多 10 个字符。

练习

- 修改 exmain 菜单
- 建立 Sessions 菜单笺的简化键和热键
- 在 QUIT 任选项之上插入分隔条(\-)
- 给 Session 菜单笺起名 Sessions
- 生成菜单
- DO 菜单程序

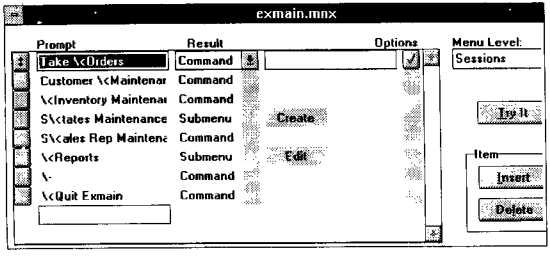
3.2 菜单动作

本节目的

- 菜单笺或弹出式的动作

- 激活或禁止菜单任选项
- 激活菜单
- 说明菜单笺名字

菜单动作



有三种方法指定菜单任选项的动作。

首先，给子菜单的每一个任选项赋一个动作；其次，给子菜单的所有任选项赋同一个动作；第三，给菜单的所有菜单笺赋同一个动作。

下面是给弹出式任选项赋一个动作的步骤：

- Edit Sessions 子菜单
- 确保 Result 是 COMMAND
- 在 Result 旁的文本框中键入“WAIT WINDOW [IORDERS]”[]告诉菜单生成器[]之间(如 Orders)的内容在命令行生成时作为文本，无论何时选择这个任选项，将执行 WAIT WINDOW 命令。

下面是给子菜单的所有任选项赋同一个动作的步骤：

- 从 Menu 弹出式中选择“SESSIONS Options…”
- 在 PROCEDURE 框中键入 WAIT WINDOW PROMPT()

无论何时选择子菜单的任何一项时，都将执行 WAIT WINDOW PROMPT()命令。

PROMPT()函数返回用户最后选择的提示符文本。最后，通过“Menu”弹出式的“General Options…”任选项给菜单中所有任选项赋同一个动作。

练习

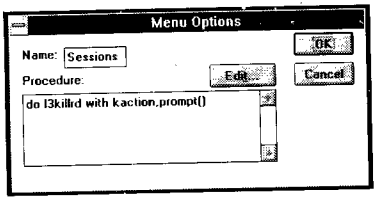
- 修改 exmain 菜单。
- 给某一 Sessions 任选项建立 .VAIT WINDOW [Orders]。
- 给全部 Sessions 菜单笺建立 WAIT WINDOW PROMPT()。

- 生成 exmain. mpr。
- DO exmain. mpr，观察菜单的动作。

3.3 菜单控制程序

本节目的

- 建立菜单控制程序
- 建立菜单 READ



3.3.1 激活菜单

DO exmain. MPR 就能激活 exmain. mnx 中建立的菜单系统，但是激活菜单与激活程序是不同的。

3.3.2 建立 READ

我们建立的菜单只要在读者发出 READ，BROWSE，MODIFY 和 ACTIVATE MENU 之后就能工作，但是标准的 genmenu. prg 菜单生成器不提供这些部分。

Exmain. prg 还包括检测内存变量"Kaction"值的"控制环"，因为 exmain. mpr 程序只是完整的应用程序的一部分，所以它建立的菜单改变了 Kaction 的值，在 Session Options Procedure 对话框中调用 13killrd 过程。

我们看一下 13killrd：

```
m1 = v1
DO control WITH "MENU"
```

现在，读者应该知道在从下拉式中选择一个选项后，PROMPT() 返回的值赋给了变量 kaction。

我们的菜单系统还有另外要考虑的，它有一个 Reports 子菜单。在提示符后带上子菜单名（如 REPORTS）能保证 Kaction 存储的值是唯一的。

```
DO l3killrd WITH kaction,"REPORTS" + PROMPT()
```

练习

- 修改 exmain 菜单
- 增加 Sessions 任选项的 l3killrd 调用
- 增加 Reports 的 l3killrd 调用
- 删除前面菜单的动作(如 WAITT WINDOW…)
- 生成菜单

现在回到菜单控制程序 exmain. prg。

READ VALID . T. 命令启动菜单,当用户选择菜单的一个选项后,FoxPro 执行 Menu Bui¹der 指定的程序。程序执行完后,控制返回到 READ VALID . T. 命令之后的那一行。

练习

1. 给每一个菜单任选项增加 CASE 语句(下面是 Reports 子菜单的例子)。

```
CASE kaction="Reports"
    DO CASE
        CASE kaction="REPORTSCustomer LIST"
```

3. 给尚未用的任选项的 CASE 语句增加 WAIT WINDOW 命令。
3. 增加退出 exmain 控制环的 CASE 语句。
4. 运行程序。

3.4 应用菜单程序

既然我们用 Menu Builder 已经做过一些练习了,那么我们把注意力移到半成品 exmain. prg 程序上。exmain. prg 将使用 FoxPro 中一些高级函数。

exmain. prg 中第一条命令是 DO exsetup…。

exsetup. prg 执行的第一个动作是设置环境变量。这些设置是全局的,直到设置改变或 QUIT FoxPro 这些设置一直有效。

其次,exsetup. prg 建立应用程序所用的 PUBLIC 内存变量。

然后,ON ERROR 命令建立错误陷井程序。

激活 Escape 键。程序员按{Escape}键就可中断程序。显然,当程序编辑完毕开始运行时,这个键要被禁止。

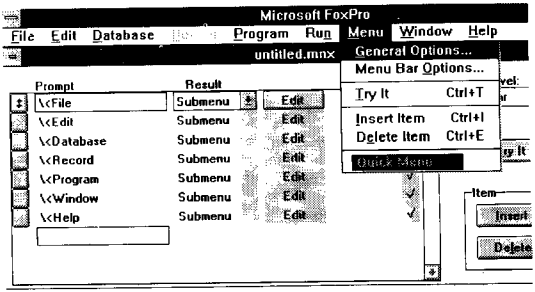
exmain. prg 建立"全局"ON KEY LABEL。无论何时按{F2},都将执行命令 DO exlook WITH VARREAD()。

3.5 菜单杂集

本节目的

- 使用 Quick Menu

- 把 FoxPro 菜单选项放到自己的菜单系统



exsetup. prg 执行 exmain1. MPR 菜单设置程序。在命令窗口中键入"DO exmain1. MPR"看这个程序的执行情况。请注意一些原 FoxPro 菜单笺不见了。

检查剩下的菜单笺的弹出式。

下面是在自定义菜单中包括原 FoxPro 菜单项的步骤：

- 修改新菜单(如 exmain2)
- 从 Menu 菜单笺中选择"Quick Menu"
- 编辑 Help 子菜单
- 删除 About FoxPro 和 Filer 任选项。

注：子菜单中的任选项被赋予"BAR #S"，而不是命令或子菜单。它们是"特殊"的 FoxPro 菜单选项。当一个提示符被赋与了 FoxPro "BAR #"，则选择这个提示符就会触发与 Bar 有关的原 FoxPro 动作。

- 从子菜单中删除 Calculator 提示符
- 回到 Menu Bar 级
- 菜单条中增加\(Mine 菜单笺
- 编辑有关子菜单
- 建立"\(Calculator"提示符
- 选择"Ban #'
- F 产生 HELP 对话框
- 选择 Search 键入 MENU
- 选择 MENU-System Menu Names

- 选择 Show Topics 和 Go To
- 从 Edit 菜单选择 Copy
- 在 Copy 对话框中找到入口(如_MST_CALCU)选择 Copy。关闭 HELP 窗口返回到菜单表
- 使用 Edit 菜单中 PASTE 任选项(CTRL+V)在文本区中放 BAR 入口
- 生成菜单
- 在命令窗口键入 DO exmain2. MPR
- 选择 Mine Pad 和 Calculator 任选项。

这样就把原 FoxPro 系统菜单的一部分放到了自定义的菜单系统中。

exmain1. mpr 减少了 FoxPro 菜单笺的数量，为应用程序菜单笺留出了许多空间。但是所有重要的任选项都已经放到了 Help 或 Edit Pad 中。

3.6 PUSH MENU

exmain 程序的第一行的命令 PUSH MENU _MSYSMENU。它执行了一个很重要的功能。备份了菜单系统口当前状态。在程序尾命令 POP MENU 把_MSYSMEU 恢复到 exmain. prg 开始执行的状态。

命令放的位置也很重要。PUSH MENU 命令放在 exsetup 之前。若放在它之后，exmain1 菜单被压入栈，而不是 FoxPro 系统菜单。如果程序中涉及多于一个菜单，那么一定要注意压入和弹出命令的位置。exmain. mpr 菜单把 Sessions 菜单笺加入到 exsetup. prg 建立的菜单中。

3.7 复 习

- 用 Menu Builder 建立和修改菜单
- 热键和简化键
- Menu Builder Meta 字符
- 菜单动作
- 菜单 READ
- QUICK MENU
- FoxPro System 菜单笺

3.8 复习题

1. 建立或修改菜单的命令是什么？
2. 在 Menu Builder 中在哪里向菜单笺任选项赋与动作？
3. 如何向弹出式任选项赋与某一动作？
4. 我们为什么要使用菜单控制程序？

第四章　数据录入屏幕

内容概要

- 屏幕控制程序设计
- 直接与间接 READ
- SCATTER MEMVAR MEMO
- Screen Builder

本章目的

- 设计屏幕控制程序
- 区别直接与间接 READ
- 字段分散到内存变量中
- 在 Screen Builder 中建立简单数据入口屏幕
- 使用 QUICK SCREEN
- 在屏幕格式上放文本和字段
- 生成屏幕程序
- 区别 INPUTT 和 OUTPUT 字段
- 使用 FORMAT 子句和函数
- 使用 READ

4.1　屏幕控制程序

本节目的

- 建立屏幕控制程序
- 保存初始环境
- 输入内存变量

为了更有效地建立、维护和使用屏幕，需要有一些程序模板，把屏幕、程序、表和索引等放在一起工作。在本课我们将使用下面的模型：

```
SETTUP CODE
DO Screen. SPR
CLEANUP CODE
PROCEDURES
```

SETUP CODE

为了运行应用程序，我们必须保存当前环境信息，打开应用程序表，确保工作区正确，然后让用户显示、增加、编辑或删除记录。该代码出现在屏幕控制程序的 SETUP CODE 部分。

现在我们已有一个打开表并选择工作区和索引的程序，下面花几分钟讨论如何保存初始环境。

因为运行应用程序时，我们不可能知道用户如何从一个程序转到另一个程序，所以保存工作区和索引就很重要了。程序结束时，恢复 CLEANUP CODE 部分中的环境，不用担心工作区是否错了，索引是否错了。环境 SETUP CODE 如下：

```
PRIVATTE ALL LIKE l_ *

* * save environment
PUSH KEY CLEAR
l_oldsele=SELECT()                    && previous WORK area
SET DELETED ON

* * reset order in data files
l_custord=" "
l_ordhord=" "
IF USED('excust')
  l_custord=ORDER('excust')
  IF EOF()
    GO BOTTOM
  ENDIF sitting AT EOF IN excust
ENDIF excust NOT already OPEN

IF USED('exordh')
  l_ordhord=ORDER("exordh")
ENDIF

IF ! USED('exsyst')
  USE exsyst iN SELECT(1) AGAIN
ENDIF

DO exopen

SET ORDER TO sy_tag IN exsyst
SET ORDER TO oh_custid IN exordh
```