FOXPRO 2.5 程序设计讲座(四)

罗 辉 (湖南省双峰工行) 何丹鸿 (湖南省娄底人行)

第五讲 全屏幕编辑命令的使用

FOXPRO2.5 对全屏幕编辑命令@…SAY…GET 的功能作了很大的扩充,为用户设计各种友善美观的人机界面提供了方便。

本讲将频繁地涉及到 GET 对象和控制钮的概念,全屏幕编辑命令的 GET 编辑域、各种控制钮都可称为 GET 对象。有关控制钮的知识请参阅下一讲。

一、全屏幕编辑命令

1.全屏幕编辑命令

全屏幕编辑命@…SAY…GET 是继承以往 FOX 产品的一个传统的命令。但是 FOXPRO2.5 对它的功能作了很大的扩充。

正如我们所知,FOXPRO 全屏幕编辑命令同样支持功能子句 FUNCTION 和图像子句 PICTURE 的使用,但其控制代码较之 FOXBASE+2.1 要丰富。譬如,代码"T",可用于过滤要输的表达式前后空格;代码"^",可使数值以科学表示法显示;等。有关控制代码请参阅技术资料。

全屏幕编辑命令还可带 SIZE <expN1>,<expN2> 子句,以控制显示或打印区域的长度和宽度,限制输出固定在这一区域内显示或打印。这在用全屏幕编辑命令编辑备注字段或长字符串数据时,将有很不错的效果。

如果是在 FOXPRO2.5 FOR WINDOW 下使用这条命令,则还可用 FONT <expC3>[,<expN3>]子句设定输出的字体<expC3>和字体大小<expN3>,并用 STYLE <expC4>子句定义字体显示风格。FONT 和 STYLE 子句的使用如第二讲所示。

为了取得不同的效果,还可用 COLOR SCHEME COLOR 子句设定显示文本的前/后景颜色。

FOXPRO2.5 的全屏幕编辑命令不同于以往 FOX 产品的最有特色之处是,FOXPRO2.5 提供的编辑权效验子句 WHEN 和有效性确定子句 VALID。FOXBASE+2.1 的全屏幕编辑命令也提供一个 VALID 子句,但其只能带表达式,使用能力不强。而 FOXPRO2.5 提供的 WHEN 和 VALID 子句不仅可使用校验表达式,还可带用户自定义函数。在自定义函数中,除可完成需要的编辑数据效验外,还可以作任何其它子程序能实现的功能,从而极大的丰富了全屏幕编辑命令的能力。对它们运用得好,常常使用户收到意想不到的效果。

2.WHEN 和 VALID 子句

(1)WHEN 子句

带 WHEN 子句的全屏幕编辑命令,将使用户在欲将光标移进该编辑域前进行编辑权限检查。系统将根据 WHEN 子句中指定的逻辑表达式、数值表达式或用户自定义函数对权限进行校验,以决定是否有权编辑该域。

当用户选择的是逻辑表达式 EXPL1 时,如果 EXPL1 为假,该域只能显示,无权编辑,系统自动跳下一编辑域。只有在 EXPR1 为真时,才能进入该编辑域编辑。而当用户选择的是数值表达式 EXPN1 时,如果 EXPN1 值为 0,表示无权编辑,光标仍截留在原编辑域;当 EXPN1 的运算结果是一个正值或负值时,表示输入有效,该值指示 READ 语句要激活的下一个 GET 域的相对域数,其中正值表示向后继续,负值向前返回。

(2)VALID 子句

在用全屏幕编辑命令@…SAY…GET 编辑一个字段或变量时,可通过 VALID 子句来确认输入数据的合法性。当编辑退出该编辑域时,系统将用 VALID 子句中指定的逻辑表达式、数值表达式或用户自定义函数对输入值进行校验,以决定输入是否有效。

当用户选择的是逻辑表达式 EXPL1 时,如果

EXPL1 为假，输入无效，系统将显示一串固定的或用户自定义的错误报警信息，同时将截留在原编辑域等待重新编辑。只有在 EXPR1 为真即输入有效后，才能退出该编辑域。

而当用户选择的是数值表达式 EXPN1 时，如果 EXPN1 值为 0，表示输入无效，光标仍截留在原编辑域；当 EXPN1 的运算结果是一个正值或负值时，表示输入有效，该值指示 READ 语句要激活的下一个 GET 域的相对域数，其中正值表示向后继续，负值向前返回。

(3)WHEN 和 VALID 子句的设计

①在程序中直接嵌入条件。当程序简单、输入输出屏幕也不复杂且用于 WHEN 或 VALID 子句的条件也简单时，可简单地通过在程序中直接嵌入逻辑表达式或算术表达式作为判断条件。

②使用用户自定义函数。当用于 WHEN 或 VALID 子句的判断条件比较复杂时，将条件直接嵌入@命令将难于实现或会降低程序的可读性，这时可通过一个自定义判断函数来取代直接嵌入。

③设计公用判断函数。如果对每一个编辑域都设计一个 WHEN 或 VALID 判断函数，当编辑域较多时，类似的函数势必相应增多，增加了编程和测试工作量；同时使系统变得庞大，给阅读、修改和维护带来困难。鉴于数据的 WHEN 和 VALID 判断过程的相似性，可以抽出所有编辑域进行 WHEN 或 VALID 判断的不同逻辑或算术表达式，用一个数据库来统一管理所有的判断用表达式等资源，而判断过程则可用一个公用的自定义函数实现。

二、GET 命令的激活

1.READ 命令

正如我们所知，全屏幕编辑命令的激活是由 READ 命令实现的。READ 命令不再如 FOXPLUS+2.1 中的 READ 一样单调了，从其使用格式就可见一斑：

```
READ [CYCLE] [ACTIVATE <expL1>] [DEACTIVATE <expL2>]
     [MODAL] [WITH <window title list>]
     [SHOW <expL3>] [VALID <expL4  expN1>]
     [WHEN <expL5>] [OBJECT <expN2>]
     [TIMEOUT <expN3>] [SAVE] [NOMOUSE] [LOCK NOLOCK]
     [COLOR <color pair list>  COLOR SCHEME <expN4>]
```

READ 命令激活自上一个 READ 或 CLEAR GETS 命令之后定义的所有编辑域、各种控制按钮等对象。

CYCLE 子句使 READ 不被终止，除非你按了 Escape、Ctrl+W，或使用了请求终止 READ 的命令 CLEAR READ 或用了子句 TIMEOUT。

MODAL 子句确保使除 READ 命令调用的窗口之外的窗口不是活动的。

默认时，所有系统内部窗口(如 Browse 窗口、桌面附件和用 MODIFY FILE、MODIFY REPORT 等打开的窗口等)能在 READ 命令中共享。用 WITH 子句限制能被 READ 共享的窗口。

在执行 READ 所带任何子过程中，当遇到 SHOW GETS 命令时将执行 SHOW 子句。由 SHOW 例程返回的值被忽略。一个 SHOW 例程可被用来刷新@ ... SAY 命令组或激活或关闭控制钮。

用 VALID 子句决定 READ 命令的终止与否；而用 WHEN 子句可决定 READ 是否被执行。

OBJECT 子句指定当 READ 被执行时，哪个对象被初始选择，每一个编辑域、独立的按钮、无线按钮或不可见按钮等控制钮被分别认为是一个对象。

使用 TIMEOUT 决定 READ 命令的作用时间(以秒为单位)。如果 READ 由于 TIMEOUT 子句的作用而被终止，则若此时任何对象都未被修改，则 READKEY()函数将返回 20，若已有编辑域被改变，则返回值 276，且正在编辑的编辑域所作的任何改变都作废，但其它已作改变的编辑域被认可。

除非你使用了 SAVE 子句，否则所有对象的定义在 READ 命令执行后将自动清除(并不清除屏幕显示)。如果前一 READ 命令使用了 SAVE 子句，则其对象定义被保存，你可重新仅使用 READ 命令即可激活前面定义的对象。

用 NOMOUSE 子句防止对象被鼠标选择。这样，你只有用键盘从一个对象移到另一个对象。

LOCK 和 NOLOCK 子句使你可以标明包含由对象所指定的字段的记录，在 READ 期间是否自动加锁。它们主要用于网络环境的数据库共享。如果包含了 LOCK 子句，系统将试图给对象所用到的每一个记录上锁。如果上锁成功，你就可以放心编辑该记录。可用

SET REPROCESS 命令设定试上锁的次数和时间。READ 命令默认用 LOCK 子句。NOLOCK 子句使对象涉及到的记录不上锁。使用这些记录中字段的对象被置为只读,按钮不能被选择,且以非活化的暗色显示。如果你用 READ NOLOCK 修改一个备注字段,则该记录自动被上锁。

当由 READ 命令调用的窗口被提到前面时,其中的动作可由 ACTIVATE 和 DEACTIVATE 子句控制。你可通过点按该窗口将之提到前面来,也可通过按 Ctrl+F1 或选择激活窗口的菜单选项转到该窗口。

ACTIVATE 作为窗口级 WHEN 子句,在首次执行 READ 命令或切换当前 READ 窗口之前,将先执行 ACTIVATE 子句;而当你试着从当前 READ 作用范围内含有编辑或控制对象的最前面的窗口移到另一窗口并将之提到最前面来时,DEACTIVATE 将被执行。DEACTIVATE 子句的返回值(.T.或.F.)决定 READ 命令是否终止或继续。DEACTIVATE 可被看成窗口级 VALID 子句。

当 DEACTIVATE 子句被执行时,函数 WONTOP()返回正被提到前面的窗口名,而 WLAST()返回在此之前的最前面的窗口名。

如果 DEACTIVATE 子句使一个含有 READ 对象的新窗口到最前面时,将执行 ACTIVATE 子句。

当一个窗口被提到最前面时,如果在 DEACTIVATE 子句中使用命令: ACTIVATE WINDOW (WLAST( )),将使该窗口回到它原来的位置。

2.由一个 READ 控制多个编辑窗口

一个 READ 能同时激活多个窗口中的各种对象。对象按它们建立的顺序激活,而不论是放在哪个窗口中。

当将光标移到某一对象时,它所在窗口被激活,并变成当前输出窗口。当光标在一个窗口的最后一个对象上时,可通过按 Tab、Enter 或↓键将光标移到下一窗口的第一个对象上;而按 Shift+Tab 或↑键可使在一个窗口的第一个对象上的光标移到上一窗口的最后一个对象上。也可用鼠标点按不同窗口上的 GET 对象在不同对象间直接移动。

每一个 GET 对象可出现在不同的窗口中,能在不同窗口中的 GET 对象间随意移动。当用键盘从一个对象移到另一个对象时,光标将按程序中对象出现的顺序遍历两个对象间的其它对象,而不管这些对象是在那个窗口中。当前所遍历的对象所在的窗口将被激活并成为当前输出窗口。

在一个 READ 活动期间存取另外的窗口,可通过 READ MODAL 子句实现。当 READ 命令中 MODAL 子句时,你不能关闭、缩放、最小化、移动不被 READ 作用的窗口。

3.READ 的嵌套应用

READ 命令可嵌套应用,嵌套深度最多为 5 级。其嵌套是由 READ 的 VALID 或 WHEN 子句内含 READ 来实现的。可用 RDLEVEL( )函数返回当前 READ 嵌套深度,取值在 0–5 之间。如无 READ 执行,RDLEVEL( )返回 0。

【例 5.1】READ 三级嵌套。按回车键进入下一嵌套级的 READ,按 ESC 退回到上级 READ。

```
CLEAR
@ 4,2 SAY "第一级 READ 嵌套:" GET level1 VALID(proc1( )) DEFAULT 1
READ
RETURN
PROCEDURE proc1
@ 6,2 SAY "第二级 READ 嵌套:" GET level2 VALID(proc2( )) DEFAULT 2
READ
RETURN"
PROCEDURE proc2
@8,2 SAY "第三级 READ 嵌套:" GET level3 DEFAULT 3
READ
RETURN"
```

4.READ 事件和子句的执行顺序

当出现 READ 时,READ 事件或子句的执行顺序:执行 READ 级的 WHEN;第一个 GET 对象激活;执行 READ 级 ACTIVATE 子句;执行 READ 级 SHOW 子句;执行第一个 GET 级的 WHEN 子句。

当一个新对象被激活之前,READ 事件或子句被调用的顺序是:退出一个对象时执行 VALID 子句;被退出的对象非活化;新 GET 对象被活化;执行 READ 级 DEACTIVATE 子句;执行 READ 级 ACTIVATE 子句;执行新对象的 WHEN 子句。

三、GET 对象的重显

GET 命令建立的编辑域和控制钮等对象，在对象值发生变化后，还可刷新重新进行显示，而不必再次用 GET 命令重现这些对象。

FOXPRO2.5 提供了三条用于 GET 对象重显的命令：SHOW GETS、SHOW GET 、SHOW OBJECT。

SHOW GETS 命令的格式：

```
SHOW GETS [ENABLE DISABLE] [LEVEL <expN1>] [OFF ONLY] [LOCK]
[WINDOW <winname>][ COLOR SCHEME <expN2>
COLOR <color pair list>]
```

SHOW GET / SHOW OBJECT 的命令格式：

```
SHOW GET <var> [,<expN1> [PROMPT <expC>]]
[ENABLE DISABLE]
[LEVEL <expN2>] [COLOR SCHEME <expN3>
COLOR <color pair list>]
```

SHOW GETS 用于重显全部 GET 对象(编辑域或控制钮)的值。而 SHOWGET 或 SHOW OBJECT 命令则用于个别对象的重显；SHOW OBJECT 与 SHOW GET 之不同在于前者通过对象号刷新对象，而后者通过变量名进行刷新。

在 GET 级 VALID 或 WHEN、或在 READ 级 ACTIVATE 或 DEACTIVATE 子句所带的自定义函数中可以使用 GET 对象重显命令。

SHOW GETS 命令将置 UPDATED()返回值为假(.F.).

各子句的作用是：

ENABLE | DISABLE 允许或阻止重显的对象有被选中的能力。如命令：SHOW GET fruit,2 DISABLE，它重显第二个按钮但禁止被选。LEVEL 指明 SHOW GETS 作用的 READ 嵌套级，默认仅对当前的 READ 级。OFF 使 SHOW GETS 只执行 READ SHOW 所带过程而不刷新对象；而 ONLY 则只刷新控制而不执行 READ SHOW 所带过程。WINDOW 子句指明特定的窗口中的对象被刷新显示。LOCK 给 READ 调用的记录上锁。使用 PROMPT 可替代不可见按钮、无线按钮或复选框等控制钮的屏幕提示符及其属性(通过在其中包含特殊控制字符实现)。在 FoxPro for Windows 下，可用 PROMPT 标明特定的.BMP 文件以替代一个图象控制钮的显示。在 PROMPT 中至少要标明三个.BMP 文件名：第一个.BMP 文件用于控制钮是可选择的情况；第二个文件用于该钮被选中时的情况；而第三个文件用于该钮不可选择的情况。

【例 5.2】SHOW GETS 命令的应用。本例由一个 READ 命令控制两个窗口：数据录入窗口和控制窗口。在 READ 活动期间，你可以在两个窗口间任意移动；当两个窗口的数据改变了时，由 SHOW GETS 命令重显新的内容。注意：重显命令只刷新显示 GET 对象，对 SAY 内容不刷新，因此必须另外编程刷新显示。

```
CLOSE ALL
SET TALK OFF
DEFINE WINDOW customer FROM 3,3 TO 13,57 FLOAT
SHADOW DOUBLE
DEFINE WINDOW panel2 FROM 2, 61 TO 14,74 ;
    FLOAT SHADOW DOUBLE COLOR SCHEME 5
USE customer
ACTIVATE WINDOW customer NOSHOW    &&第一个
窗口
@3,3 SAY '客户名:'
@3,14 GET customer.company SIZE 1,35
@5,3 SAY '联系人:'
@5,14 GET customer.contact SIZE 1,35
@7,3 SAY '城市 / 州名:'
@7,14 GET customer.city SIZE 1,21
@7,36 GET customer.state
@7,39 GET customer.zip
@0,3 SAY '客户号 #'
@0,15 SAY cno SIZE 1,6
@0,36 SAY '记录号 #'
@0,46 SAY RECNO() SIZE 1,3
ACTIVATE WINDOW panel2    &&第二个窗口
@1,2 GET act PICTURE '@ * VN 顶;上;下;底;&?退出' ;
   SIZE 1,8,1 DEFAULT 0 VALID actvalid( )
READ CYCLE SHOW readshow() COLOR ,R / BG
RELEASE WINDOWS customer,panel2
FUNCTION actvalid    &&有效性确定子过程
DO CASE
   CASE act = 1
     GO TOP
   CASE ACT = 2
     SKIP -1
     IF BOF( )
        GO TOP
     ENDIF
   CASE ACT = 3
     SKIP 1
     IF EOF( )
```

```
        GO BOTTOM
    ENDIF
    CASE act = 4
        GO BOTTOM
    CASE act = 5
        CLEAR READ    &&退出 READ 命令
ENDCASE
SHOW GETS    &&用新位置的记录数据刷新屏幕显示
RETURN 0
FUNCTION readshow    &&对 SAY 命令刷新显示子过程
STORE WOUTPUT( ) TO currwind
IF SYS(2016) = 'customer' OR SYS(2016) = '*'
    ACTIVATE WINDOW customer SAME
    @0,15 SAY cno SIZE 1,6
    @0,46 SAY RECNO( ) SIZE 1,3
ENDIF
IF NOT EMPTY(currwind)
    ACTIVATE WINDOW (currwind) SAME
ENDIF
RETURN .T.
```

四、GET 对象的拾取

在程序设计中，往往希望能直接获得当前正在编辑的 GET 对象的名字，以实现某种特殊的目的，比如说针对具体的 GET 对象建立在线式帮助机制之类的应用。在 FOXPRO2.5 中有两个函数用于完成这一功能：VARREAD()和 SYS(18)。这两个函数所完成的功能是一样的，即返回当前 GET 对象(编辑字段或控制钮)的名字，名字用大写方式。

还可用 READKEY()函数返回 READ 命令退出时所按的键值，如果退出前，对数据没作修改，则该函数返回值在 0–36 之间的值；否则返回值在 256–292 之间。该命令当然对其它编辑命令 APPEND、BROWSE、CHANGE、CREATE、EDIT、MODIFY、INSERT 等一样有效。

最后，以一个对 GET 对象的在线式帮助的应用实例作为本讲的结束。

【例 5.3】为 GET 对象提供在线式帮助。

```
SET TALK OFF
ON KEY LABEL F1 DO Help WITH VARREAD( )  &&设定 HELP 热键为 F1
USE customer
DEFINE WINDOW input FROM 6,10 to 18,70 PANEL
ACTIVATE WINDOW input
@1,3 SAY '客户名:' GET company
@3,3 SAY '地址:' GET address
@6,7 SAY '按 Esc 键退出，或按 F1 键获得帮助'
READ
RELEASE WINDOW input
ON KEY LABEL F1
RETURN
PROCEDURE Help    &&在线式帮助子过程
PARAMETERS fieldname
DO CASE
    CASE fieldname = 'COMPANY'
        WAIT WINDOW '录入客户公司名' NOWAIT
    CASE fieldname = 'ADDRESS'
        WAIT WINDOW '录入客户通讯地址' NOWAIT
ENDCASE
RETURN
```

第六讲 控制钮的设计

FOXPRO2.5 系统提供七种特殊控制钮的设计：按钮、无线按钮、不可见按钮、复选框、弹出控制、滚动列表、旋转器。其中旋转器只对 FOXPRO2.5 FOR WINDOWS 有效。

这七种控制钮的设计都是通过 …GET 命令来实现的，与@…SAY…GET 全屏幕编辑命令的使用类似，但不包括 SAY 子句。除了滚动列表和旋转器是靠子句来标识其控制特性外，不同的控制钮有不同的控制代码，靠相应的图像子句 PICTURE 或功能子句 FUNCTION 来标识。正如全屏幕编辑命令一样，其控制既可用图像子句也可用功能子句来实现。

对这七种控制钮的设计，其共同点是：它们都可带图像子句或功能子句来标识控制方式；可以用 VALID 或 WHEN 子句进行有效性确定或编辑权校验；用 SIZE 子句限定控制钮的区域大小；也可用 DEFAULT 设定默认值，用 MESSAGE 给用户提供选择信息；还可用[ENABLE DISABLE]子句使能／禁止该控制钮的选择，此时按钮以不活动的暗色显示；都可对控制钮的颜色进行任意设计；对 FOXPRO2.5 FOR WINDOWS 版本，都可设置字体的大小、位置和风格。

使用控制代码标识的控制钮，其控制代码由说明码开始，由特殊字符如：*、^或 等组成，或由星号"*"加字母组成。说明码后加一空格，然后是该控制钮的提示符，如果该命令可用于显示多个控制钮，则也可以跟一列用

分号隔开的控制钮提示符。在用 READ 命令激活这些屏幕控制语句之前，可以将任意@…GET 命令的组合放在屏幕上。

各种按钮的设计还可使用其它一些控制代码，常用到如下几个代码：

N　按钮选择后不终止 READ；　T　按钮选择后终止 READ 的执行；

H　按钮水平显示；　　V　按钮垂直显示。

控制钮可设置热键，通过在作热键的字符前加反斜杠和小于号"＼＜"标识某控制钮的选择热键；如果要使某个控制钮不能选择，可在该控制钮提示符前置两个反斜杠"＼＼"；在某提示符前置反斜杠和感叹号"＼!"，则可使该控制钮为缺省选项；为标识当你按 ESC 键退出时的默认选项，则请在该控制钮提示符前置"反斜杠和问号"＼?"。

1.按　钮（Push）

星号＊为按钮的控制代码。在图像子句中以＊开头，后跟一系列提示符。激活该语句后，将产生一组按钮或图像按钮。对 Foxpro2.5 for DOS 而言，各提示符将以一对尖括号"＜　＞"括住；对 Foxpro2.5 for Windows，各提示符上出现一个按钮图案，该按钮图案在你选中它时，具有形象的压真实按钮的动感。

按钮设计可设置热键、默认键、退出键等。它可使用 N、T、H、V 等控制代码及其组合。

【例 6.1】按钮设计。本例建立六个按钮。

```
SET TALK OFF
DEFINE WINDOW one FROM 3,5 TO 18,20 FLOAT DOUBLE COLOR SCHEME 5
ACTIVATE WINDOW one
@1,2 GET mchoice FUNCTION '＊NV 顶;前;后;底;＼?Q 退出';
    SIZE 2, 10, 1 DEFAULT 1
READ CYCLE
CLEAR WINDOW
```

2.无线按钮（Radio）

无线按钮的说明代码为＊R。图像子句的开头为＊R 加上用于建立无线按钮的一列提示符。

它产生一组无线按钮或图像无线按钮，在每一个提示符前附上一对圆括号"( )"，选择其中一个按钮后，将在该按钮提示符前的圆括号内置一个点号"."，表示该项被选中。在一组无线按钮中，同一时刻，只能选中一个控制。

无线按钮可用 N、T、H、V 等控制码及其组合。

【例 6.6】无线按钮设计。

```
SET TALK OFF
CLEAR
DEFINE WINDOW one FROM 3, 5 TO 13, 33 ;
    FLOAT DOUBLE COLOR SCHEME 5
ACTIVATE WINDOW one
@ 1,2 GET mchoice FUNCTION '＊RNV 前;后;顶;底';
    SIZE 1, 10, 1 DEFAULT '后'
READ CYCLE
WAIT WINDOW mchoice + '按钮被选择' NOWAIT
```

3.不可见按钮（Invisible）

不可见按钮的说明代码为＊I。图像子句的开头为＊I 加上一列用于建立不可见按钮的由分号分隔的提示符。

激活该命令后，将建立一列不可见的按钮，其中的提示符不显示出来。它不仅可用 SIZE 定义按钮的大小，在 SIZE 子句中还可带第三个参数，以控制钮间的距离。

定义了不可见按钮后，即可在按钮位置用全屏幕编辑命令显示一些字符，作为该按钮的偶象，之后选择了该偶象，即表示选中了该不可见按钮。

它可使用 N、T、H、V 代码及其组合。

【例 6.3】不可见按钮设计。本例在屏幕上显示四张牌，选中某张牌，显示相应花样名。

```
STORE 0 TO mchoice
ACTIVATE SCREEN
CLEAR    &&水平建立四个不可见按钮的偶象(四张牌)
@ 2,2 SAY REPLICATE(CHR(3),2)    &&红桃 4
@ 3,2 SAY REPLICATE(CHR(3),2)
@ 1,1,4,4 BOX
@ 2,10 SAY REPLICATE(CHR(4),2)    &&方块 4
@ 3,10 SAY REPLICATE(CHR(4),2)
@ 1,9,4,12 BOX
@ 2,18 SAY REPLICATE(CHR(5),2)    &&梅花 4
@ 3,18 SAY REPLICATE(CHR(5),2)
```

```
@ 1,17,4,20 BOX
@ 2,26 SAY REPLICATE(CHR(6),2)    &&黑桃 4
@ 3,26 SAY REPLICATE(CHR(6),2)
@ 1,25,4,28 BOX
@ 1,1 GET mchoice PICTURE '@ * IH ;;;' SIZE 4, 4, 4 ;
    VALID SHOWCARD() MESSAGE '请选择一张牌!' ;
    COLOR ,,,,,R / W    &&在四个偶象上建立四个无线按钮
READ CYCLE
PROCEDURE SHOWCARD    &&显示选中的牌的花样名
@ 6,1 CLEAR
DO CASE
    CASE  CUROBJ = 1
      @ 6,1 SAY '红桃'
    CASE  CUROBJ = 2
      @ 6,9 SAY '方块'
    CASE  CUROBJ = 3
      @ 6,17 SAY '梅花'
    CASE  CUROBJ = 4
      @ 6,25 SAY '黑桃'
ENDCASE
RETURN .T.
```

4.复选框(Check)

复选框,又称检查框。其控制方式,由说明代码 * C 来标明:图像子句中包括 * C,其后跟复选框所用的提示符。一条语句只能标明一个复选框,如要设计多个复选框,必须用多条语句一一列出。

它产生一个复选框或图形复选框,即在按钮提示符的前面出现一对方括号"[ ]",当选中该复选框时,括号内出现一个"X"字母;否则括号内是一个空格。其中图形复选框只对 FOXPRO2.5 FOR WINDOWS 有效。

如用一个 READ 命令激活了多个复选框,可以同时选中多个复选框,或者不选。

复选框可使用控制代码 N 和 T。

【例 6.4】复选框的设计。本例建立两个复选框。

```
CLEAR
STORE 1 TO mchoice1
STORE 0 TO m.choice2, m.choice3
@ 2,2 GET mchoice1
@ 4,2 GET m.choice2 FUNCTION ' * C \主题' ;
    MESSAGE '.请用空格键选择该复选框.'
@ 6,2 GET m.choice3 FUNCTION ' * C \<T 帮助细节' ;
    MESSAGE '.请用空格键选择该复选框.'
READ CYCLE    &&按 Esc 键退出 READ
RETURN
```

5.弹出菜单(Popup)

弹出菜单的控制代码是^,图像子句以^开头,后跟一列弹出菜单将显示的提示符,或者用图象子句以^开头标识为弹出菜单控制,再用 FROM <数组名>子句从事先存储有全部选择项的数组中提出各选择项。

激活一个弹出菜单控制时,首先在屏幕上显示一个弹出菜单控制钮,它仅显示一个选择项,即仅显示默认选择项。如果在该钮上按空格或用鼠标点按该控制钮右边的选择钮,将在该控制钮下激活一个弹出菜单。在弹出菜单上移动选择某一选项,相应的选项将同时在上面的控制钮上显示出来。选中后,弹出菜单自动消失。

从弹出菜单中选择某项后,根据变量的类型返回不同的值:如果变量为字符型,则将所选取的提示符赋给此变量;如变量为数字型,则将该提示符在弹出菜单中的编号返回给它。

它可使用控制代码:N、T。

【例 6.5】弹出菜单控制设计。使用弹出菜单选择 CUSTOMER 库的一个索引字段作为主索引以便 BROWSE 编辑。

```
SET TALK OFF
DEFINE WINDOW popupex FROM 5,10 TO 15,55 ;
    FLOAT DOUBLE COLOR SCHEME 5
CLOSE DATABASES
USE customer
DIMENSION dbftags(256)
dbftags(1) = '按记录顺序'
FOR i = 2 to 256    &&将所有索引字放到数组中
    IF empty(tag(i-1))
        i = i - 1
        DIMENSION dbftags(i)    &&重新调整数组大小仅容纳索引字个数
        EXIT
    ELSE
        dbftags(i) = tag(i-1)
    ENDIF
ENDFOR
dbforder = '按记录顺序'
ACTIVATE WINDOW popupex
@ 3,7 GET dbforder PICTURE '@^' FROM dbftags SIZE 3, 14
@ 0,2 TO 7,43
@ 0,5 SAY ' 按索引编辑 Customer '
@ 2,27 GET okcancel PICTURE '@ * VT \!OK;\?Cancel' ;
```

```
    SIZE 1, 10, 2 DEFAULT 0
READ CYCLE
RELEASE WINDOW popupex
IF okcancel = 1
    IF dbforder = "按记录顺序"
        SET ORDER TO
    ELSE
        SET ORDER TO (dbforder)
    ENDIF
    WAIT WINDOW '以'+dbforder+'为主索引' NOWAIT
    BROWSE WIDTH 10 NOWAIT
ELSE
    WAIT WINDOW 'Order not set' NOWAIT
ENDIF
```

6.滚动列表(List)

滚动列表的控制方式不同于其它控制钮的设计，其有效选择项不是在本语句@…GET 中列出，而必须用 FROM <数组名>子句从一个一维数组事先存储的各选项提示符，可通过 RANGE 子句限定数组的一部分作为列表选项；或用 POPUP <上弹菜单名> 子句直接使用事先定义的上弹菜单中的滚动列表选项。

激活该命令后，将在屏幕上事先设计好的弹出窗口上出现一个选择项列表，通过光条滚动可选择某一项，将它在列表中的编号赋给变量。

滚动列表可使用控制代码&与 N 或 T 的组合。

【例 6.6】滚动列表设计。以某数据库一字段内容为列表选项。

```
SET TALK OFF
DEFINE WINDOW example FROM 4,3 TO 21,76 TITLE '滚动列表举例';
    FLOAT SHADOW SYSTEM COLOR SCHEME 8
USE customer
* * * 以字段 company 的内容定义一个上弹菜单 * * *
DEFINE POPUP popfield PROMPT FIELD company SCROLL;
    MARGIN MARK CHR(16)    &&设置选中标志为一个三角形
ACTIVATE WINDOW example
@ 1,26 SAY 'company 字段内容列表:'
@ 2,25 GET listfield POPUP popfield SIZE 11, 20;
  DEFAULT company VALID dispitem(listfield);
  COLOR SCHEME 9
READ CYCLE
RELEASE WINDOW example
RELEASE POPUPS popfield

FUNCTION dispitem
PARAMETER item
WAIT WINDOW '你的选择是:' + ALLTRIM(item) NOWAIT
RETURN 0
```

7.旋转器控制(Spinner)

旋转器控制仅适用于 FOXPRO2.5 FOR WINDOWS，它使你可在一组数值中循环选择。

激活旋转器后，将在控制钮提示符右边，出现一个文本框，数值显示在一个文本框中；通过按↑(或↓)键使文本框中的数值顺序递增(减)，也可在文本框直接输入数值作为选择。它必须带 SPINNER 子句定义这组循环的数值。SPINNER 的使用格式是：

SPINNER <expN1> [,<expN2> [, <expN3>]]

每按一次↑(或↓)键数值的增(减)量由 expN1 指定，由 expN2 和 expN3 指定数值的变化范围。

一条语句只能标明一个旋转器，如要设计多个旋转器，必须相应由多条语句一一标明。

【例 6.7】变量 mchoice 的变化范围是：-5-24，以 1 为增量。

```
@ 2,2 GET mchoice SPINNER 1, -5, 24 DEFAULT 3
```

在旋转器的功能子句中还可以用如下代码以控制旋转器文本框中数值如何显示：

B　数值在文本框左对齐；　I　数值在文本框居中显示；

J　数值右对齐；

K　当直接输入选择值时，光标移动将选择整个值以便被编辑。

L　数值前置 0；　　Z　数值为 0 时显示空格；

^　用科学表示法显示数值；　¥　用货币格式显示；

可以在屏幕或窗口上随意使用任何控制钮，也可在同一屏幕上用一条 READ 命令控制各种控制钮的任意组合，为屏幕设计提供了很大的灵活性。再结合使用全屏幕编辑命令，即可设计出丰富多彩的用户界面来。