返回总目录

DEFINE MENU 命令

创建菜单栏。

语法

```
DEFINE MENU MenuBarName
  [BAR [AT LINE nRow]]
  [IN [WINDOW] WindowName | IN SCREEN]
  [FONT cFontName [, nFontSize]]
  [STYLE cFontStyle]
  [KEY KeyLabel]
  [MARK cMarkCharacter]
  [MESSAGE cMessageText]
  [NOMARGIN]
  [COLOR SCHEME nSchemeNumber
  | COLOR ColorPairList]
```

参数描述

MenuBarName

指定要创建的菜单栏的名称。给菜单栏命名使您能够在其他命令和函数中引用该菜单栏。

BAR [AT LINE nRow]

创建一个与 Visual FoxPro 系统菜单栏类似的菜单栏，该菜单栏有以下特征：

- 水平菜单栏的高度为一个行高，宽度为它所在的 Visual FoxPro 主窗口或用户自定义窗口的宽度。
- 自动处理菜单栏上菜单标题的位置。
- 如果您所定义的菜单标题的大小与数目超出了菜单栏所在的屏幕或窗口的大小，可以滚动菜单栏。

行号用 *nRow* 指定。

IN [WINDOW] WindowName

在用户自定义窗口中放置一个菜单栏。*WindowName* 指定用户自定义窗口名。如果省略 IN WINDOW，除非有一个活动的用户自定义窗口，在默认的情况下，菜单栏放置在主窗口中。如果有一个活动的用户自定义窗口，菜单栏就放在其中。

IN SCREEN

在 Visual FoxPro 主窗口中放置菜单栏。

FONT cFontName [, nFontSize]

为菜单栏中所有的菜单标题指定默认的字体，在 DEFINE PAD 中包含 FONT 子句可以改写单个菜单标题的默认字体。

cFontName 指定字体名称，*nFontSize* 指定磅值。例如，下面的命令创建一个菜单栏，

它的菜单标题使用 12 磅 Courier 字体：

```
DEFINE MENU mnu 示例 FONT 'Courier', 12
```

如果在 FONT 子句中省略磅值 *nFontSize*，就使用 10 磅字体。如果指定的字体不可用，就用相似的字体代替。

添加到 Visual FoxPro 系统菜单 _MSYSMENU 中的菜单标题忽略 FONT 子句。注意菜单设计器使用的是 Visual FoxPro 系统菜单。

STYLE cFontStyle

为菜单栏中所有的菜单标题指定一个默认的字形，在 DEFINE PAD 命令中包含 STYLE 子句可以改写单个菜单标题的默认字形。

如果省略了 STYLE 子句，就使用常规字形。在 Visual FoxPro 中，如果指定的字形不可用，那么就用相似的字形代替。

下表列出了可以使用 *cFontStyle* 指定的字形：

字符	字形
B	粗体
I	斜体
N	常规
Q	不透明
–	删除线
T	透明
U	下划线

可以组合多个字符来指定一个字形。例如，下列命令指定粗斜体：

```
DEFINE MENU mnuexample STYLE 'BI'
```

添加到 Visual FoxPro 系统菜单 _MSYSMENU 中的菜单标题忽略 STYLE 子句。注意菜单设计器使用的是 Visual FoxPro 系统菜单。

KEY KeyLabel

指定用于激活菜单栏的键或组合键。有关可用键或组合键，以及它们的键标记名的内容，请参阅 ON KEY LABEL。

包含 KEY 子句等于使用了下列命令：

ON KEY LABEL KeyLabel ACTIVATE MENU MenuName

注意 如果一个键同时具有键标记和键盘宏，则键盘宏优先，不能用它指定的键或组合键来激活菜单栏。

MARK cMarkCharacter

指定在菜单栏的菜单标题左端出现的标记字符。可以使用 MARK 将默认的标记字符改为用 *cMarkCharacter* 指定的标记字符。如果 *cMarkCharacter* 包含一个以上字符，则仅有第一个字符用作标记字符。

在 Visual FoxPro 中，默认的标记字符是对勾号。如果菜单栏是 Visual FoxPro 系统菜单，就忽略 MARK 子句，而使用默认的标记字符。如果菜单栏所在的 Visual FoxPro 主窗口或用户自定义窗口中的字体不是 FoxFont，则忽略 MARK 子句。

注意 指定一个标记字符并不能在菜单栏上标记菜单名。可使用 SET MARK OF 和指定的字符在菜单栏上标记菜单标题。

DEFINE PAD 指定的标记字符优先于 DEFINE MENU 中用 MARK 子句指定的标记字符。SET MARK OF 用来切换标记字符的打开或关闭，它也能用来为单个菜单项或所有的菜单项指定标记字符。

MESSAGE cMessageText

当您选择菜单标题时显示一条信息。在 Visual FoxPro 中，信息放在图形方式状态栏中。如果图形方式状态栏被 SET STATUS BAR OFF 关闭，信息就显示在 Visual FoxPro 主窗口中最后一行的中央。

NOMARGIN

默认情况下，删除每个菜单名左端和右端的空格。

COLOR SCHEME nSchemeNumber

为单个菜单栏指定颜色。

COLOR ColorPairList

为单个菜单栏指定颜色。默认情况下，菜单项的颜色由当前颜色集中配色方案 2 决定。

有关颜色方案和颜色对的详细内容，请参阅稍前部分的“颜色概述”。

说明

使用 DEFINE MENU 为应用程序的菜单系统创建菜单栏。使用 DEFINE PAD 在菜单栏上创建菜单标题。使用 ON PAD ... ACTIVATE 指定在每个菜单标题下显示的菜单。使用 DEFINE POPUP 在每个菜单标题下创建菜单。使用 ACTIVATE MENU 激活整个菜单系统。

如果使用菜单设计器创建菜单，就完全没有必要使用这些命令，菜单设计器自动地为菜单创建命令。菜单设计器使用 Visual FoxPro 系统菜单，您可以通过添加自己的菜单项更新它。

有关创建菜单的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十一章“设计菜单和工具栏”中的“创建菜单系统”。

示例

以下示例使用 DEFINE MENU 命令创建用户自定义的菜单系统。首先使用 SET SYSMENU SAVE 命令将当前系统菜单栏存入内存，然后使用 SET SYSMENU TO 命令清除系统菜单标题。

使用 DEFINE MENU 命令创建菜单栏，使用 DEFINE PAD 命令创建两个菜单标题。DEFINE POPUP 为每个菜单标题创建菜单。DEFINE BAR 创建在每个菜单上的菜单项。当选择菜单标题时，ON PAD 使用 ACTIVATE POPUP 激活相应的菜单。ACTIVATE MENU 显示并激活菜单栏。

当从菜单中选取菜单项时，执行 CHOICE 过程。CHOICE 显示已选取菜单项的名称和包含菜单项的菜单名。

```
*** Name this program DEFIMENU.PRG ***
CLEAR
SET SYSMENU SAVE
SET SYSMENU TO
ON KEY LABEL ESC KEYBOARD CHR(13)
DEFINE MENU example BAR AT LINE 1
DEFINE PAD convpad OF example PROMPT '\<Conversions' COLOR SCHEME 3 ;
    KEY ALT+C, ''
DEFINE PAD cardpad OF example PROMPT 'Card \<Info' COLOR SCHEME 3 ;
    KEY ALT+I, ''
ON PAD convpad OF example ACTIVATE POPUP conversion
ON PAD cardpad OF example ACTIVATE POPUP cardinfo
DEFINE POPUP conversion MARGIN RELATIVE COLOR SCHEME 4
DEFINE BAR 1 OF conversion PROMPT 'Ar\<ea' ;
    KEY CTRL+E, '^E'
DEFINE BAR 2 OF conversion PROMPT '\<Length' ;
    KEY CTRL+L, '^L'
```

```
DEFINE BAR 3 OF conversion PROMPT 'Ma\<ss' ;
    KEY CTRL+S, '^S'
DEFINE BAR 4 OF conversion PROMPT 'Spee\<d' ;
    KEY CTRL+D, '^D'
DEFINE BAR 5 OF conversion PROMPT '\<Temperature' ;
    KEY CTRL+T, '^T'
DEFINE BAR 6 OF conversion PROMPT 'T\<ime' ;
    KEY CTRL+I, '^I'
DEFINE BAR 7 OF conversion PROMPT 'Volu\<me' ;
    KEY CTRL+M, '^M'
ON SELECTION POPUP conversion DO choice IN defimenu WITH PROMPT( ), POPUP( )
DEFINE POPUP cardinfo MARGIN RELATIVE COLOR SCHEME 4
DEFINE BAR 1 OF cardinfo PROMPT '\<View Charges' ;
    KEY ALT+V, ''
DEFINE BAR 2 OF cardinfo PROMPT 'View \<Payments' ;
    KEY ALT+P, ''
DEFINE BAR 3 OF cardinfo PROMPT 'Vie\<w Users' ;
    KEY ALT+W, ''
DEFINE BAR 4 OF cardinfo PROMPT '\-'
DEFINE BAR 5 OF cardinfo PROMPT '\<Charges '
ON SELECTION POPUP cardinfo;
    DO choice IN defimenu WITH PROMPT( ), POPUP( )

ACTIVATE MENU example
DEACTIVATE MENU example
RELEASE MENU example EXTENDED
SET SYSMENU TO DEFAULT
ON KEY LABEL ESC
PROCEDURE choice
PARAMETERS mprompt, mpopup
WAIT WINDOW 'You chose ' + mprompt + ;
     ' from popup ' + mpopup NOWAIT
```

请参阅

ACTIVATE MENU, CNTPAD( ), CREATE MENU, DEACTIVATE MENU, DEFINE PAD, GETPAD( ) , HIDE MENU, MRKPAD( ) , ON PAD, ON SELECTION PAD, PRMPAD( ) , RELEASE MENUS, RELEASE PAD, SET MARK OF, SET SYSMENU, SHOW MENU

DEFINE PAD 命令

在用户自定义菜单栏或 Visual FoxPro 系统菜单栏上创建一个菜单标题（主菜单名）。

语法

```
DEFINE PAD MenuTitle1 OF MenuBarName PROMPT cMenuTitleText
  [AT nRow, nColumn]
  [BEFORE MenuName2 | AFTER MenuName3]
  [NEGOTIATE cContainerPosition [, cObjectPosition]]
  [FONT cFontName [, nFontSize]]
  [STYLE cFontStyle]
  [KEY KeyLabel [, cKeyText]]
  [MARK cMarkCharacter]
```

[SKIP [FOR lExpression]]
[MESSAGE cMessageText]
[COLOR SCHEME nSchemeNumber
| COLOR ColorPairList]

参数描述

MenuTitle1

指定要创建的菜单的标题。指定菜单标题使您能够在其他的命令和函数中引用此菜单。

OF MenuBarName

指定菜单标题所在的菜单栏名。

PROMPT cMenuTitleText

指定真正显示的菜单标题文本。

在作为快捷键的字符前加反斜杠和小于号(\<)，可以为菜单标题创建快捷键。在下面的例子中，用户可按下“I”键从“Receive”菜单中选择“Invoices”，按下“Q”键从同样的菜单中选择“Inquiry”：

```
DEFINE MENU mnuReceive
DEFINE PAD padInvoice OF mnureceive PROMPT "\<Invoices"
DEFINE PAD padInquire OF mnureceive PROMPT "In\<quiry"
ACTIVATE MENU mnuReceive
```

AT nRow, nColumn

指定菜单标题出现在菜单栏中的位置。*nRow*，*nColumn* 是菜单标题的左端在

Visual FoxPro 主窗口或用户自定义窗口中的坐标。

如果省略了 AT 子句，第一个菜单标题的左端就放在 Visual FoxPro 主窗口或用户自定义窗口的第 0 行，下一个菜单标题放在第一个标题的右端，依此类推。

注意 对于由 DEFINE MENU 中的 BAR 子句创建的菜单栏，不能通过包含 AT 子句指定菜单标题的位置。

BEFORE MenuName2

将菜单栏上的菜单标题放到 *MenuName2* 指定的菜单标题的左边。通过键盘访问菜单标题的顺序与菜单栏中菜单标题的位置一致。

AFTER MenuName3

将菜单栏上的菜单标题放到 *MenuName3* 指定的菜单标题的右边。通过键盘访问菜单标题的顺序与菜单栏中菜单标题的位置一致。

首先必须创建在 BEFORE 或 AFTER 子句中指定的菜单标题。如果不创建菜单标题，那么菜单栏上的菜单标题的放置就由创建的顺序或用 AT 子句指定的位置决定。

不是用 BAR、BEFORE 或 AFTER 子句创建的菜单栏决定了由键盘访问菜单标题的顺序，菜单标题的位置由 AT 子句指定。

运行下面两个 example，并注意当分别用 AT 子句和不用 AT 子句定义菜单标题时菜单标题的位置与访问顺序的区别：

```
*** 程序示例 1 不用 AT 子句 ***
DEFINE MENU mnuBefAft
DEFINE PAD padOne OF mnuBefAft PROMPT '1111'
DEFINE PAD padTwo OF mnuBefAft PROMPT '2222'
```

```
DEFINE PAD padThree OF mnuBefAft PROMPT '3333'
DEFINE PAD padFour  OF mnuBefAft PROMPT '4444' BEFORE padTwo
ACTIVATE MENU mnuBefAft
```

***程序示例 2 使用 AT 子句 ***

```
DEFINE MENU mnuBefAft
DEFINE PAD padOne OF mnuBefAft PROMPT '1111' AT 1,5
DEFINE PAD padTwo OF mnuBefAft PROMPT '2222' AT 1,15
DEFINE PAD padThree OF mnuBefAft PROMPT '3333' AT 1,25
DEFINE PAD padFour  OF mnuBefAft PROMPT '4444' BEFORE padTwo AT 1,35
WAIT WINDOW 'Press ESC to erase menu' NOWAIT
ACTIVATE MENU mnuBefAft
```

NEGOTIATE cContainerPosition [, cObjectPosition]

cContainerPosition 指定当对 Visual FoxPro 表单中的一个 ActiveX 控件进行 OLE 可视编辑时，菜单标题在 Visual FoxPro 菜单栏中的位置。

cObjectPosition 指定菜单标题在 Active Document 宿主程序的菜单栏中的位置。

cContainerPosition 的设置有，

设置	说明
NONE	不显示菜单标题。
LEFT	菜单标题放在“文件组”的右边。

MIDDLE	菜单标题放在“容器组”的右边，“编辑”菜单的后面。
RIGHT	菜单标题放在“窗口组”的左边。

cObejectPosition 的设置有，

设置	说明
NONE	不显示菜单标题。
LEFT	菜单标题放在“文件组”的右边。
MIDDLE	菜单标题放在“容器组”的右边，“编辑”菜单的后面。
RIGHT	菜单标题放在“帮助”菜单中。

注意，在 Active Document 应用程序中可能只有一个“RIGHT”菜单标题。如果指定了一个以上“RIGHT”菜单标题，那么所有的菜单标题都将放置在“帮助”菜单的左方。

如果省略 NEGOTIATE 子句，则当进行 OLE 可视编辑时从菜单栏中删除菜单标题。ONE 是 cContainerPosition 和 cObjectPosition 的默认值。

FONT cFontName [, nFontSize]

为菜单标题指定字体。*cFontName* 指定字体名称，*nFontSize* 指定磅值。例如，下列命令创建一个 12 磅 Courier 字体的菜单标题。

```
DEFINE PAD padPageAccts OF mnuReceive FONT 'Courier', 12
```

如果在 FONT 子句中省略磅值 *nFontSize*，则使用 10 磅的字体。如果指定的字体不可用，那么就用相似的字体代替。

添加到 Visual FoxPro 系统菜单 _MSYSMENU 中的菜单标题忽略 FONT 子句，注意菜

单设计器使用 Visual FoxPro 系统菜单。

STYLE cFontStyle

为菜单标题指定字形。如果省略 STYLE 子句，那么就使用常规字形。在 Visual FoxPro 中，如果指定的字形不可用，那么就用相似的字形代替。

下表列出了可以使用 *cFontStyle* 指定的字形：

字符	字形
B	粗体
I	斜体
N	常规
Q	不透明
–	删除
T	透明
U	下划线

可以组合多个字符来指定字形。例如，下列命令指定粗斜体：

```
DEFINE PAD padPageAccts OF mnuReceive STYLE 'BI'
```

添加到 Visual FoxPro 系统菜单 _MSYSMENU 中的菜单标题忽略 STYLE 子句，注意菜单设计器使用 Visual FoxPro 系统菜单。

KEY KeyLabel [, cKeyText]

为菜单标题指定访问键或组合键。有关可用键和组合键以及它们的键标记名的详细内容，请参阅 ON KEY LABEL。

注意 如果一个键具有同样的键标记和键盘宏，则键盘宏优先，此时不能用指定的键或组合键去选择菜单标题。

对于不是用 BAR 子句创建的菜单栏，键标记放在菜单标题的右边。用 BAR 子句创建的菜单栏中，不显示键标记。Visual FoxPro 系统菜单栏中菜单标题的键标记也不显示。

包含 *cKeyText* 可以用自己的文本替换键标记。例如，包含 KEY ALT+B 将文本 ALT+B 放到菜单标题的右端。如果指定的不是 KEY ALT+B，而是“^B”，则 ^B 就出现在菜单标题的右端。通过为 *cKeyText* 指定空字符串可以不显示一个键标记。

MARK cMarkCharacter

指定出现在菜单标题左端的标记字符。包含 MARK 子句可以将默认的标记字符改写为 *cMarkCharacter* 指定的字符。如果 *cMarkCharacter* 包含一个以上字符，则仅用第一个字符作为标记字符。

默认的标记字符为对勾号。当包含菜单标题的菜单栏是 Visual FoxPro 系统菜单时，就忽略 MARK 子句，且将使用默认的标记字符。如果菜单栏的 Visual FoxPro 所在的主窗口或用户自定义窗口中的字体不是 FoxFont，则忽略 MARK 子句。

用 DEFINE PAD 指定的标记字符优先于 DEFINE MENU 中用 MARK 子句指定的标记字符。SET MARK OF 用来切换标记的打开或关闭。它也能用来为单个菜单标题或所有的菜单标题指定标记字符。

注意 指定一个标记字符并不能给菜单标题做标记，可以使用 SET MARK OF 命令以及指定的字符去标记一个菜单标题。

SKIP [FOR lExpression]

指定一个条件。如果 *lExpression* 为“真”(.T.)，菜单标题被禁止，禁止用户

选择；如果 *lExpression* 为“假”(.F.)，则允许选择菜单标题。
也可以在菜单标题文本前放置反斜杠 (\) 来禁止一个菜单项。For example:

```
DEFINE PAD padPageAccts OF mnuReceive PROMPT '\Age Accounts'
```

菜单标题 padPageAccts 以暗灰色显示，表明不能选择它。
在 Visual FoxPro 中禁止的菜单标题可以显示但不能选择。但是会显示由 MESSAGE 指定的菜单信息。

MESSAGE cMessageText

选择一个菜单标题时显示信息。在 Visual FoxPro 中，消息放在图形方式状态栏中。如果图形方式状态栏用 SET STATUS BAR OFF 关闭，信息就显示在 Visual FoxPro 主窗口最后一行的中央。

COLOR SCHEME nSchemeNumber

为单个菜单标题指定颜色，它将取代默认的或用 DEFINE MENU 指定的颜色。

COLOR ColorPairList

为单个菜单标题指定颜色。它将取代默认的或用 DEFINE MENU 指定的颜色。

默认情况下，菜单栏中菜单标题的颜色由当前颜色集中的配色方案 2 决定。
有关颜色方案和颜色对的详细内容，请参阅稍前部分的“颜色概述”。

说明

必须用 DEFINE PAD 命令创建放置在菜单栏上的每个菜单标题。在放置菜单标题之前，必须用 DEFINE MENU 定义菜单栏，且必须在 DEFINE PAD 中包含菜单栏名。

如果使用菜单设计器创建菜单，根本没有必要使用这些命令，菜单设计器会自动为菜单创建这些命令。菜单设计器使用的是 Visual FoxPro 系统菜单。您可以通过添加自己的菜单项来更新它。

有关创建菜单的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十一章“设计菜单和工具栏”中的“创建菜单系统”。

示例

以下示例使用 DEFINE PAD 命令在 Visual FoxPro 系统菜单栏中放置菜单标题。首先使用 SET SYSMENU SAVE 命令将当前系统菜单栏存入内存，然后使用 SET SYSMENU TO 命令清除系统菜单标题。

使用 DEFINE MENU 命令创建菜单栏，使用 DEFINE PAD 命令创建两个菜单标题。DEFINE POPUP 为每个菜单标题创建菜单。DEFINE BAR 创建在每个菜单上的菜单项。当选择菜单标题时，ON PAD 使用 ACTIVATE POPUP 激活相应的菜单。ACTIVATE MENU 显示并激活菜单栏。

当从菜单中选取菜单项时，执行 CHOICE 过程。CHOICE 显示已选取菜单项的名称和包含菜单项的菜单名。

使用 DEFINE MENU 命令创建几个菜单栏。当从菜单中关闭菜单项时，执行 CHOICE 过程。CHOICE 显示已关闭菜单项的名称和包含菜单项的菜单名，并切换菜单标题的标记字符“开”和“关”。如果关闭了“Exit”菜单标题，恢复原始的 Visual FoxPro 系统菜单。

```
*** Name this program DEFINPAD.PRG ***
CLEAR
SET TALK OFF
SET SYSMENU SAVE
```

```
SET SYSMENU TO
PUBLIC markpad
markpad = .T.
DEFINE PAD syspad OF _MSYSMENU PROMPT '\<System' COLOR SCHEME 3 ;
   KEY ALT+S, ''
DEFINE PAD editpad OF _MSYSMENU PROMPT '\<Edit' COLOR SCHEME 3 ;
   KEY ALT+E, ''
DEFINE PAD recordpad OF _MSYSMENU PROMPT '\<Record' COLOR SCHEME 3 KEY
ALT+R, ''
DEFINE PAD windowpad OF _MSYSMENU PROMPT '\<Window' COLOR SCHEME 3 ;
   KEY ALT+W, ''
DEFINE PAD reportpad OF _MSYSMENU PROMPT 'Re\<ports' COLOR SCHEME 3 ;
   KEY ALT+P, ''
DEFINE PAD exitpad OF _MSYSMENU PROMPT 'E\<xit' COLOR SCHEME 3 ;
   KEY ALT+X, ''
ON SELECTION MENU _MSYSMENU ;
   DO choice IN definpad WITH PAD( ), MENU( )
PROCEDURE choice
PARAMETER mpad, mmenu
WAIT WINDOW 'You chose ' + mpad + ;
   ' from menu ' + mmenu NOWAIT
SET MARK OF PAD (mpad) OF _MSYSMENU TO ;
   ! MRKPAD('_MSYSMENU', mpad)
markpad = ! markpad
IF mpad = 'EXITPAD'
   SET SYSMENU TO DEFAULT
ENDIF
```

请参阅

ACTIVATE MENU, CREATE MENU, DEACTIVATE MENU, DEFINE

MENU, GETPAD(), HIDE MENU, MRKPAD() , ON PAD, ON SELECTION PAD, PRMPAD() , RELEASE PAD, SET MARK OF, SET MESSAGE, SET SYSMENU, SHOW MENU

DEFINE POPUP 命令

创建菜单。

语法

```
DEFINE POPUP MenuName
  [FROM nRow1, nColumn1]
  [TO nRow2, nColumn2]
  [IN [WINDOW] WindowName | IN SCREEN]
  [FONT cFontName [, nFontSize]]
  [STYLE cFontStyle]
  [FOOTER cFooterText]
  [KEY KeyLabel]
  [MARGIN]
  [MARK cMarkCharacter]
  [MESSAGE cMessageText]
```

[MOVER]
[MULTISELECT]
[PROMPT FIELD FieldName | PROMPT FILES [LIKE FileSkeleton]
| PROMPT STRUCTURE]
[RELATIVE]
[SCROLL]
[SHORTCUT]
[TITLE cMenuTitleText]
[COLOR SCHEME nSchemeNumber
| COLOR ColorPairList]

参数描述

MenuName
指定要创建的菜单的名称。

FROM nRow1, nColumn1 TO nRow2, nColumn2
指定菜单放置的位置。 *nRow1*、*nColumn1* **指定菜单左上角的坐标。如果省略** FROM **子句，** Visual FoxPro **就将菜单的左上角放在** Visual FoxPro **主窗口或用户自定义窗口的第一行第一列。**

若要用指定的尺寸创建菜单，可包含 TO *nRow2, nColumn2* **来指定菜单右下角的位置。如果包含了** FROM *nRow1, nColumn1* **而省略了** TO *nRow2, nColumn2*，Visual FoxPro **自动设置菜单大小。菜单的宽度与最长的菜单项一样宽（如果菜单项是用** DEFINE BAR **创建的），菜单的长度受到所在的** Visual FoxPro **主窗口或用户自定义窗口的限制。若菜单的长度不够容纳所有的菜单项，就会出现一个滚动条允许在菜单项间滚动。**

IN [WINDOW] WindowName

在 *WindowName* 指定的用户自定义窗口中放置菜单。如果省略这个子句，在默认情况下，菜单放在 Visual FoxPro 主窗口中，除非有一个活动的用户自定义窗口。如果有一个活动的用户自定义窗口，菜单就放在该活动窗口中。

IN SCREEN

在 Visual FoxPro 主窗口中放置菜单。

FONT cFontName [, nFontSize]

为菜单指定默认的字体。可以通过在 DEFINE BAR 中包含 FONT 子句改写单个菜单项的默认字体。

cFontName 指定字体名称，*nFontSize* 指定字体磅值。例如，下列命令创建 12 磅 Courier 字体的菜单：

```
DEFINE POPUP popMyPopup FONT 'Courier', 12
```

如果在 FONT 子句中省略磅值 *nFontSize*，就用 10 磅字体。如果指定的字体不可用，就用相似的字体代替。

STYLE cFontStyle

为菜单指定默认的字形。可在 DEFINE BAR 中包含 FONT 子句改写单个菜单项的默认样式。

如果省略 STYLE 子句，或指定的字形不可用，就使用常规字形。

下表列出了可用 *cFontStyle* 指定的字形：

字符	字形
B	粗体
I	斜体
N	常规
Q	不透明
–	删除线
T	透明
U	下划线

可以组合多个字符来指定字形。例如，下列命令指定粗斜体：

```
DEFINE MENU popMyPopup STYLE 'BI'
```

FOOTER cFooterText

用 *cFooterText* 指定的文本在菜单底部中央创建一个脚注。

KEY KeyLabel

为菜单指定访问键或组合键。请参阅 ON KEY LABEL 命令，可得到可用键和组合键以及它们的键标记名称的列表。

在此使用 KEY 子句等效于发出下列命令：

```
ON KEY LABEL KeyLabel ACTIVATE POPUP MenuName
```

注意 如果已经使用相同的键盘标签定义了键盘宏，键盘宏优先执行，使用指定键或键组合不能激活菜单。

MARGIN

在每个菜单的两边设置额外的空间，标记字符显示在菜单项左端的空间里，表明有一个附加的层叠式子菜单的箭头显示在菜单项的右端。如果省略 MARGIN，标记字符改写菜单项名的第一个字符，表示菜单层次的箭头改写菜单项的最后一个字符。

MARK cMarkCharacter

指定出现在菜单项左端的标记字符。默认的标记字符为对勾号。如果菜单并入 Visual FoxPro 系统菜单中，就忽略 MARK 子句，并使用默认的标记字符。如果菜单所在的 FoxPro 主窗口或用户自定义窗口中的字体不是 FoxFont，则忽略 MARK 子句。

可以包含 MARK 将默认的标记字符改写为 *cMarkCharacter* 指定的字符。如果 *cMarkCharacter* 包含多个字符，则仅第一个字符用作标记字符。

注意 指定一个标记字符并不能给一个菜单项做标记，可用 SET MARK OF 给菜单项做标记。

MARK 子句可为菜单上所有的菜单项设置标记字符。DEFINE BAR 命令指定的标记字符优先于 DEFINE POPUP 中用 MARK 子句指定的标记字符。SET MARK OF 用来切换标记字符的打开或关闭，它也可用来为单个菜单项或为所有菜单项指定标记字符。

MESSAGE cMessageText

当选择一个菜单项时显示信息。消息放在图形方式状态栏中。如果基于字符方式的状态栏用 SET STATUS ON 打开时，消息就显示在 Visual FoxPro 主窗口最后一行的中央。

MOVER

将双向箭头(↕)放到菜单中被选定菜单项左边的移动钮中。可拖着这个双向箭头将一个菜单项移到菜单上的另一个位置。GETBAR()可用来确定菜单上每个菜单项的位置。

不能重新调整用 PROMPT 子句创建的菜单中的菜单项。

MULTISELECT

允许用户同时从一个菜单中选择多个菜单项，当用户从一个菜单中选择一项时，菜单项的左端就放置一个标记字符。

不能从一个用 PROMPT 子句创建的菜单中选择多项。

MRKBAR()可用来确定从菜单中选择的菜单项。

如果在 DEFINE POPUP 中包含了 MULTISELECT，可以包含 MARGIN 子句，在每个菜单项中为标记字符保留空间。

在以下示例中，创建了一个名为 popFruits 的菜单。包含 MULTISELECT 以创建允许选择多个菜单项的菜单。

四个菜单项的每一个都有一个不同的标记字符，当用户从菜单中选取菜单项时，标记了菜单项并有一个名为 yourchoice 例程显示已关闭的菜单项。

```
CLEAR
IF NOT _DOS
   MODIFY WINDOW SCREEN FONT 'foxfont', 12
ENDIF
ACTIVATE SCREEN
DEFINE POPUP popFruits FROM 5,5 ;
```

```
   MULTISELECT MARGIN          && 创建多选目录
DEFINE BAR 1 OF popFruits ;
   PROMPT '\<Apples'  MARK CHR(3)   && 第一项
DEFINE BAR 2 OF popFruits ;
   PROMPT '\<Bananas' MARK CHR(4)   && 第二项
DEFINE BAR 3 OF popFruits ;
   PROMPT '\<Grapes'  MARK CHR(5) && 第三项
DEFINE BAR 4 OF popFruits ;
   PROMPT '\<Lemons'  MARK CHR(6)   && 第四项
@ 12,5 SAY 'Your choices:'
ON SELECTION POPUP popFruits DO yourchoice   && 选择程序
ACTIVATE POPUP popFruits
PROCEDURE yourchoice          && 执行选择
@ 13,5 CLEAR
FOR gnCount = 1 TO CNTBAR('popFruits')     && 循环选项
   IF MRKBAR('popFruits', gnCount) = .T.     && 标记选项
     ? PRMBAR('popFruits', gnCount) AT 5     && 显示说明
   ENDIF
NEXT
```

PROMPT FIELD FieldName

指定一个已打开的表中的字段名，该表的记录作为菜单上的菜单项。对于表中的每个记录，菜单都对应一个菜单项。当激活该菜单时，就选定了该表的工作区。

提示 如果为 PROMPT FIELD 指定的、用于菜单中的字段设置了筛选条件，可以使用 Rushmore 优化。

有关 Rushmore 优化的详细内容，请参阅稍后部分的“SET OPTIMIZE 命令”和《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十五章“优化应用程序”中的“掌握 Rushmore 技术”。

FieldName 可包含多个用加号操作符 (+) 连接的字段名和表达式。*FieldName* 也可以是在另一个工作区打开表中的字段名或是一个用户自定义函数。

在 Visual FoxPro 中，对可以出现在 PROMPT FIELD 创建的菜单上的菜单项数目没有限制。

PROMPT FILES [LIKE FileSkeleton]

创建一个显示当前目录中可用的文件名的菜单。

LIKE *FileSkeleton* 允许使用通配符指定在菜单上显示的文件。例如，要创建一个显示默认驱动器和路径下的表名的菜单，可使用以下命令：

```
PROMPT FILES LIKE *.DBF
```

可以包含驱动器标识符、路径标识符或同时包含两者，来创建显示在其他驱动器、其他路径上的文件名的菜单。例如，若要创建的菜单中包含 C 驱动器 PROGRAMS 子目录下的文件名，可使用下列命令：

```
PROMPT FILES LIKE C:\PROGRAMS\*.PRG
```

PROMPT STRUCTURE

根据表的字段结构，在菜单上显示当前表中的字段名。当激活菜单时，也就

选择了表所在的工作区

RELATIVE

指定菜单项在菜单上放置的顺序。如果菜单不是用 RELATIVE 子句创建的，那么菜单项在菜单上放置的顺序由该菜单项的编号决定。菜单为没有定义的菜单项保留空间。例如，如果定义了第一项和第三项且激活了菜单，则菜单上就会为第二项保留一空行。

如果用 RELATIVE 创建菜单，菜单项就以定义它们的顺序出现在菜单上。菜单不为未定义的项保留空间。

用 RELATIVE 定义菜单允许您在 DEFINE BAR 中使用 BEFORE 和 AFTER 子句在菜单上相对于其他项的位置放置菜单项。如果菜单不是由 RELATIVE 创建，在 DEFINE BAR 中包含 BEFORE 或 AFTER 会产生错误。

运行以下两段程序示例并比较在各个菜单上菜单项的位置。

*** 关于定义后的事例 ***

```
DEFINE POPUP popRelatYes RELATIVE FROM 1,1
DEFINE BAR 4 OF popRelatYes PROMPT '4444'
DEFINE BAR 3 OF popRelatYes PROMPT '3333'
DEFINE BAR 2 OF popRelatYes PROMPT '2222'
DEFINE BAR 1 OF popRelatYes PROMPT '1111'
DEFINE BAR 6 OF popRelatYes PROMPT '6666' BEFORE 4
ACTIVATE POPUP popRelatYes
```

*** 关于无定义的事例 ***

```
DEFINE POPUP popRelatNo FROM 1,1
DEFINE BAR 4 OF popRelatNo PROMPT '4444'
DEFINE BAR 3 OF popRelatNo PROMPT '3333'
DEFINE BAR 2 OF popRelatNo PROMPT '2222'
DEFINE BAR 1 OF popRelatNo PROMPT '1111'
DEFINE BAR 6 OF popRelatNo PROMPT '6666'
```

ACTIVATE POPUP popRelatNo

SCROLL

在创建的菜单右端放置滚动条。只有当菜单项太多而菜单装不下，或菜单太长而它所在的 Visual FoxPro 主窗口或用户自定义窗口中放不下时，才出现滚动条。

SHORTCUT

创建快捷菜单，典型情况下，当使用鼠标右键单击选中部分、工具栏或任务栏时，出现快捷菜单。快捷菜单列出与鼠标右键单击的屏幕区域相关的命令。

您可以在 FORM 子句中包含 MROW( ) 和 MCOL( ) 函数以激活鼠标单击位置的弹出菜单。

TITLE cMenuTitleText

在菜单的顶边中央显示一个标题。*cTitleText* 指定菜单标题。

COLOR SCHEME nSchemeNumber

为菜单上所有元素指定颜色。默认情况下，DEFINE POPUP 创建的菜单颜色由配色方案 2 控制。

COLOR ColorPairList

为菜单上所有元素指定颜色。

有关颜色方案和颜色对的详细内容，请参阅稍前部分的“颜色概述”。

说明

要在菜单上放置一组定义的菜单项，请使用一系列 DEFINE BAR 命令。要在菜单中放置记录、文件或字段，使用 DEFINE POPUP 中的选项 PROMPT FIELD、PROMPT

FILES 或 PROMPT STRUCTURE。

当菜单显示出来并由 ACTIVATE POPUP 激活时，可在菜单上选择一个菜单项。根据选择的菜单项，可能会执行一个例程或显示并激活另一个菜单。当选择一个菜单项时显示的另一个菜单称为层叠式子菜单。有关创建子菜单的详细内容，请参阅 ON BAR。

如果使用菜单设计器创建菜单，没有必要使用这些命令。菜单设计器自动为菜单创建这些命令。菜单设计器使用 Visual FoxPro 系统菜单，可以通过添加自己的菜单项更新它。

有关创建菜单的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十一章“设计菜单和工具栏”中的“创建菜单系统”。

示例

以下示例使用了 DEFINE POPUP 命令以创建当选择菜单栏中的菜单标题时激活的菜单。首先使用 SET SYSMENU SAVE 命令将当前系统菜单栏存入内存，然后使用 SET SYSMENU TO 命令清除所有系统菜单标题。

使用 DEFINE PAD 命令创建两个菜单标题，并且 DEFINE POPUP 为每个菜单标题创建下拉式菜单。DEFINE BAR 创建在每个菜单上的菜单项。当选择菜单标题时，ON PAD 使用 ACTIVATE POPUP 激活相应的菜单。

当从菜单中选取菜单项时，ON SELECTION POPUP 使用 PROMPT( ) 和 POPUP( ) 向 CHOICE 过程传递菜单项数目和菜单项名称。CHOICE 显示已关闭菜单项的名称和包含菜单项的菜单名。CHOICE 显示已选取菜单项的文本和包含菜单项的菜单的名称。

如果从 Card Info 菜单中选取 Exit 项，恢复原始的 Visual FoxPro 系统菜单。

```
*** DEFINPOP.PRG 程序 ***
CLEAR
```

```
SET SYSMENU SAVE
SET SYSMENU TO
DEFINE PAD convpad OF _MSYSMENU PROMPT '\<Conversions' COLOR SCHEME 3 ;
    KEY ALT+C, ''
DEFINE PAD cardpad OF _MSYSMENU PROMPT 'Card \<Info' COLOR SCHEME 3 ;
    KEY ALT+I, ''
ON PAD convpad OF _MSYSMENU ACTIVATE POPUP conversion
ON PAD cardpad OF _MSYSMENU ACTIVATE POPUP cardinfo
DEFINE POPUP conversion MARGIN RELATIVE COLOR SCHEME 4
DEFINE BAR 1 OF conversion PROMPT 'Ar\<ea' KEY CTRL+E, '^E'
DEFINE BAR 2 OF conversion PROMPT '\<Length' ;
    KEY CTRL+L, '^L'
DEFINE BAR 3 OF conversion PROMPT 'Ma\<ss' ;
    KEY CTRL+S, '^S'
DEFINE BAR 4 OF conversion PROMPT 'Spee\<d' ;
    KEY CTRL+D, '^D'
DEFINE BAR 5 OF conversion PROMPT '\<Temperature' ;
    KEY CTRL+T, '^T'
DEFINE BAR 6 OF conversion PROMPT 'T\<ime' ;
    KEY CTRL+I, '^I'
DEFINE BAR 7 OF conversion PROMPT 'Volu\<me' ;
    KEY CTRL+M, '^M'
ON SELECTION POPUP conversion;
    DO choice IN definpop WITH PROMPT( ), POPUP( )
DEFINE POPUP cardinfo MARGIN RELATIVE COLOR SCHEME 4
DEFINE BAR 1 OF cardinfo PROMPT '\<View Charges' ;
    KEY ALT+V, ''
DEFINE BAR 2 OF cardinfo PROMPT 'View \<Payments' ;
    KEY ALT+P, ''
DEFINE BAR 3 OF cardinfo PROMPT 'Vie\<w Users' ;
    KEY ALT+W, ''
DEFINE BAR 4 OF cardinfo PROMPT '\-'
```

```
DEFINE BAR 5 OF cardinfo PROMPT '\<Charges '
DEFINE BAR 6 OF cardinfo PROMPT '\-'
DEFINE BAR 7 OF cardinfo PROMPT 'E\<xit '
ON SELECTION POPUP cardinfo;
    DO choice IN definpop WITH PROMPT( ), POPUP( )
PROCEDURE choice
PARAMETERS mprompt, mpopup
WAIT WINDOW 'You chose ' + mprompt + ;
    ' from popup ' + mpopup NOWAIT
IF mprompt = 'Exit'
    SET SYSMENU TO DEFAULT
ENDIF
```

请参阅

ACTIVATE POPUP, CNTBAR( ) CREATE MENU, DEFINE BAR, GETBAR( ) HIDE POPUP, MOVE POPUP, MRKBAR( ) , ON BAR, ON SELECTION BAR POPUP( ) , PRMBAR( ) , PROMPT( ) , RELEASE POPUPS, SET MARK OF SET MESSAGE SIZE POPUP SHOW POPUP

DEFINE WINDOW 命令

创建一个窗口，可以同时指定它的属性。

语法

DEFINE WINDOW WindowName1

FROM nRow1, nColumn1 TO nRow2, nColumn2
| AT nRow3, nColumn3 SIZE nRow4, nColumn4
[IN [WINDOW] WindowName2 | IN SCREEN | IN DESKTOP
[NAME ObjectName]
[FONT cFontName [, nFontSize]]
[STYLE cFontStyle]
[FOOTER cFooterText]
[TITLE cTitleText]
[HALFHEIGHT]
[DOUBLE | PANEL | NONE | SYSTEM | cBorderString]
[CLOSE | NOCLOSE]
[FLOAT | NOFLOAT]
[GROW | NOGROW]
[MDI | NOMDI]
[MINIMIZE | NOMINIMIZE]
[ZOOM | NOZOOM]
[ICON FILE FileName1]
[FILL cFillCharacter | FILL FILE FileName2]
[COLOR SCHEME nSchemeNumber
| COLOR ColorPairList]

参 数 描 述

WindowName1

指定要创建的窗口的名称。在 Visual FoxPro 中窗口名长度可以达到 254 个字符（早期 FoxPro 版本中为 10 个字符）。必须以字母或下划线开头而不能以数字开头。它们可以包含字母、数字和下划线的任意组合。

FROM nRow1, nColumn1 TO nRow2, nColumn2

在 Visual FoxPro 主窗口中指定用户自定义窗口的位置和大小。FROM *nRow1, nColumn1* 指定用户自定义窗口左上角在 Visual FoxPro 主窗口中的位置。TO *nRow2, nColumn2* 指定用户自定义窗口的右下角在 Visual FoxPro 主窗口中的位置。

定义窗口时，坐标可以超出 Visual FoxPro 边框，窗口可以比 Visual FoxPro 主窗口更大。

在 Visual FoxPro 中，窗口的位置和大小由父窗口的字体决定。父窗口可以是另一个用户自定义窗口或 Visual FoxPro 主窗口。

AT nRow3, nColumn3 SIZE nRow4, nColumn4

指定一个用户自定义窗口的位置和大小。

AT *nRow3, nColumn3* 指定用户自定义窗口的左上角在 Visual FoxPro 主窗口中的位置，这个位置是由父窗口中当前字体决定的。因为 AT 子句在各方面都与 FROM 子句相同，所以这两个子句可以互换使用。

SIZE *nRow4, nColumn4* 以行和列为单位指定用户自定义窗口的大小，它确保以特定字体显示的文本能在创建的窗口中显示。

可以使用 FONT 和 STYLE 子句为用户自定义窗口指定字体和字形。如果为窗口指定了字体并包含了 SIZE 子句，那么窗口的大小就由窗口字体的高度和宽度决定。在 Visual

FoxPro 中，如果没有为窗口指定字体，窗口的字体就是默认的系统 10 磅 FoxFont 字体。

IN [WINDOW] WindowName2

将用户自定义窗口放入一个父窗口。用户自定义窗口变成一个子窗口，它不能移到父窗口之外。如果父窗口移动，子窗口随之移动。

当子窗口放在父窗口中时，用 FROM 和 TO 子句或 AT 和 SIZE 子句指定的子窗口的坐标是相对于父窗口的，而不是相对于 Visual FoxPro 主窗口。

在以下示例中，创建父窗口 wParent。在父窗口中创建子窗口 wChild。

```
CLEAR
DEFINE WINDOW wParent ;
   FROM 1, 1 TO 20, 30 ;
   TITLE "Parent"       && 父窗口。
ACTIVATE WINDOW wParent
DEFINE WINDOW wChild ;
   FROM 1, 1 TO 20, 20 ;
   TITLE "Child" ;
   IN WINDOW wParent     && 子窗口。
ACTIVATE WINDOW wChild
ACTIVATE SCREEN
WAIT WINDOW 'Press a key to clear the windows'
RELEASE WINDOW wParent, wChild
CLEAR
```

IN SCREEN

在 Visual FoxPro 主窗口中放置用户自定义窗口。如果省略 IN SCREEN，默认情况下，用户自定义窗口放在 Visual FoxPro 主窗口中。

在 ACTIVATE WINDOW 中包含 IN WINDOW 子句，可以在另一个用户自定义窗口中放置窗口并将忽略此 IN SCREEN 子句。

IN DESKTOP

用户自定义窗口可以放在 Microsoft Windows 桌面上，但位于 Visual FoxPro 主窗口外。窗口的位置与 Windows 桌面和 Visual FoxPro 主窗口中的当前字体有关。

NAME ObjectName

为窗口创建一个对象引用，这允许您使用表单对象的面向对象属性对窗口进行操作。可以为 NAME 子句创建的窗口指定表单对象属性。

有关面向对象的程序设计的其他内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第三章“面向对象的程序设计”。有关使用 NAME 子句为创建的窗口指定表单对象属性的其他内容，请参阅稍后部分的“Form 对象”。

FONT cFontName [, nFontSize]

为放在窗口中的文本指定字体。*cFontName* 指定字体名称，*nFontSize* 指定磅值。如果省略 *nFontSize*，就使用 10 磅字体。

例如，在 Visual FoxPro 中，此命令创建一个窗口，用 16 磅 Courier 字体显示定向到窗口中的输出信息：

```
DEFINE WINDOW wDisplayFont FROM 2,2 TO 12,22 FONT 'Courier', 16
```

如果省略 FONT 子句，就使用 10 磅 FoxFont 字体。如果指定的字体不可用，就用相似的字体代替。

STYLE cFontStyle

为放置在窗口中的文本指定字形。*cFontStyle* 指定字形。如果省略了 STYLE 子句，或指定的字形不可用，就使用常规字形。

下表列出了字形和它们相应的字符。

字符	字形
B	粗体
I	斜体
N	常规
Q	不透明
–	删除线
T	透明
U	下划线

可以组合多个字符指定字形。在 Visual FoxPro 中，下列命令指定粗斜体字形：

```
DEFINE WINDOW w DisplayStyle FROM 2, 2 TO 12, 22 STYLE 'BI'
```

TITLE cTitleText

用 TITLE 子句加一个标题。*cTitleText* 指定标题文本，它显示在窗口底边的中央。如果标题比窗口宽，就截断它。

HALFHEIGHT

所创建的窗口具有对分标题栏。此参数提供了与 FoxPro 早期版本所建窗口

的兼容性，使这些窗口能导入到 Visual FoxPro、FoxPro for Windows 和 FoxPro for Macintosh 中。

当使用 DEFINE WINDOW 在 Visual FoxPro 中创建窗口时，除非包含 SYSTEM 关键字或包含 FONT 子句，否则会使用对分标题栏。

如果包含 HALFHEIGHT 关键字，则不管是否包含了 SYSTEM 或 FONT 子句，都会使用对分标题栏。

DOUBLE | PANEL | NONE | SYSTEM | cBorderString

为用户自定义窗口指定边框样式，默认的边框为一条单线。

参数	说明
DOUBLE	指定窗口周围用双线边框
PANEL	指定窗口周围用宽边框。
NONE	彻底地隐藏边框。
SYSTEM	指定用户自定义窗口，使其看上去像一个系统窗口，当包含某些其他子句（GROW，ZOOM 等）时，要在窗口的边框上设置适当的窗口控制。
cBorderString	指定一个自定义边框。有关自定义边框的详细内容，请参阅 SET BORDER。

在 Visual FoxPro 中，包含 DOUBLE 或自定义边框字符串可以创建带有 PANEL 边框的窗口。即使不包含 SYSTEM 窗口定义子句，包含 CLOSE、FLOAT、GROW 或 MINIMIZE 子句也会在窗口中设置适当的控制。

CLOSE

允许使用键盘或界面关闭用户自定义窗口。关闭窗口是将它从 Visual FoxPro

主窗口或用户自定义的父窗口中移去，并将它的定义从内存中删除。如果省略了 CLOSE 子句，就不能关闭窗口。

NOCLOSE

禁止关闭一个窗口。

FLOAT

允许您使用键盘或鼠标移动一个用户自定义窗口。如果省略了 FLOAT，不能移动窗口。MOVE WINDOW 也可用来移动一个窗口。

NOFLOAT

禁止移动一个用户自定义窗口

GROW

允许您使用键盘或鼠标调整用户自定义窗口的大小。如果省略 GROW，不能调整窗口大小，除非在程序或命令窗口中使用 SIZE WINDOW 命令。

NOGROW

禁止调整窗口的大小除非在程序或命令窗口中使用 SIZE WINDOW 命令。。

MDI

创建一个适合 MDI 的用户自定义窗口。MDI（多文档界面）允许使用多文档窗口并决定它们的结构和行为。如果省略 MDI，您所创建的窗口就不适合 MDI。

当一个适合 MDI 的窗口最大化时：

- 窗口假定为 Visual FoxPro 主窗口的大小。窗口控制消失且控制菜单框出现在 Visual FoxPro 系统菜单栏中，窗口的“复原”按钮也放在 Visual FoxPro 系统菜单栏中。

- 窗口标题放在 Visual FoxPro 标题栏中，用连字符将它与 Visual FoxPro 标题分隔开。
- 如果激活了另一个适合 MDI 的窗口，它将自动最大化。

NOMDI

创建一个不适合 MDI 的窗口。

MINIMIZE

允许使用键盘或鼠标最小化一个用户自定义窗口。

NOMINIMIZE

禁止最小化一个窗口。

ZOOM

允许使用键盘或鼠标最大化一个用户自定义窗口。也可以将窗口复原到它原先的大小。

NOZOOM

禁止最大化一个窗口。

ICON FILE FileName

指定当窗口最小化时显示的图标。必须在 DEFINE WINDOW 中包含 MINIMIZE 关键字。只能指定一个图标 (.ico) 文件，而不能指定一个位图 (.bmp) 文件。

FILL FILE FileName2

为窗口指定壁纸（背景）。窗口由指定的 *FileName2* 平铺填充，需要指定它的位图文件。

COLOR SCHEME nSchemeNumber

为用户自定义窗口指定颜色。默认情况下，DEFINE WINDOW 创建的窗口的颜色由配色方案 1 控制。

COLOR ColorPairList

为用户自定义窗口指定颜色。

有关颜色方案和颜色对的详细内容，请参阅稍前部分的“颜色概述”。

说明

DEFINE WINDOW 创建用户自定义窗口后，就可以用 ACTIVATE WINDOW 或 SHOW WINDOW 在 Visual FoxPro 主窗口中显示它们。可创建的用户自定义窗口的数量仅受可用内存和系统资源的限制。

被激活的窗口保存在 Visual FoxPro 主窗口中，直到执行 DEACTIVATE WINDOW 或 HIDE WINDOW 命令。DEACTIVATE WINDOW 和 HIDE WINDOW 命令从 Visual FoxPro 主窗口中移去窗口，但不从内存中删除窗口定义。可使用 ACTIVATE WINDOW 或 SHOW WINDOW 命令将窗口放回到 Visual FoxPro 主窗口中。

使用 CLEAR WINDOWS 或 RELEASE WINDOWS 从 Visual FoxPro 主窗口中移走窗口，并从内存中删除窗口定义。要重新显示已从内存中删除定义的窗口，必须用 DEFINE WINDOW 重新创建该窗口。

示例

在以下示例中，创建了一个名为 output 的窗口并将其激活。程序等待您的键盘输入，然后隐藏窗口。再次等待您的键盘输入，随后重新显示窗口。

```
CLEAR
DEFINE WINDOW output FROM 2,1 TO 13,75 TITLE 'Output' ;
   CLOSE FLOAT GROW ZOOM
```

```
ACTIVATE WINDOW output
WAIT WINDOW 'press any key to hide window output'
HIDE WINDOW output
WAIT WINDOW 'press any key to show window output'
SHOW WINDOW output
WAIT WINDOW 'press any key to release window output'
RELEASE WINDOW  output
```

请参阅

ACTIVATE WINDOW, DEACTIVATE WINDOW , 字体概述, Form 对象, HIDE WINDOW , MODIFY WINDOW MOVE WINDOW, RELEASE WINDOWS, SETBORDER , SHOW WINDOW

DefOLELCID 属性

指定表单或 Visual FoxPro **主窗口中默认的** OLE Locale ID **值。设计和运行时可用。**

语法

Form.DefOLELCID[= nValue]

参数描述

nValue

指定表单或 Visual FoxPro **主窗口中默认的** OLE Locale ID **值。当** OLE **绑定型**

控件和 OLE 容器控件在表单上或 Visual FoxPro 主窗口中，此默认值决定了它们的 Locale ID 值。

说明

如果表单或 Visual FoxPro 主窗口中默认的 OLE Locale ID 值改为一个新值，则在新的默认的 OLE Locale ID 值起作用之后，其中的 OLE 绑定型控件和 OLE 容器控件将启用该值。

表单或 Visual FoxPro 主窗口的 DefOLELCID 设置成 0 时，SYS(3004) 将决定在表单上或 Visual FoxPro 主窗口中的 OLE 绑定型控件和 OLE 容器控件的默认的 Locale ID 值。

有关 Locale ID 的详细内容，请参阅 SYS(3005)。

注意 DefOLELCID 属性仅影响显示 OLE 控件的用户接口语言，而不是 OLE 自动化命令语言。OLE 自动化命令语言仅受全局 LocaleID 影响，LocaleID 由 SYS(3005) 设置。

应用于

Form，_SCREEN

请参阅

OLELCID 属性，SYS(3005)

DELETE 命令

给要删除的记录做标记。

语法

```
DELETE
  [Scope] [FOR lExpression1] [WHILE lExpression2]
  [IN nWorkArea | cTableAlias]
  [NOOPTIMIZE]
```

参数描述

Scope

指定要做删除标记的记录范围，范围子句有：ALL、NEXT *nRecords*、RECORD *nRecordNumber* 和 REST。

有关范围子句的详细内容，请参阅“帮助”中的“范围子句”和“语言概述”。

DELETE 的默认范围是当前记录 (NEXT 1)。

FOR lExpression1

指定一个条件，仅给满足逻辑条件 *lExpression1* 的记录做删除标记。

如果 *lExpression1* 是一个可优化表达式，且表在 DECETED( ) 上建立索引时，可以用 Rushmore 优化 DELETE ... FOR 创建的查询。要得到最佳性能，可在 FOR 子句中使用一个可优化表达式。

有关 Rushmore 优化表达式的内容，请参阅稍后部分的“SET OPTIMIZE 命令”和《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十五章“优化应用程序”中的“掌握 Rushmore 技术”。

WHILE lExpression2

只要 *lExpression2* 计算为“真”(.T.)，就给这些记录做删除标记。

IN nWorkArea

指定要加记录删除标记的表所在的工作区。

IN cTableAlias

指定要加记录删除标记的表的别名。

如果省略 *nWorkArea* 和 *cTableAlias*，就给当前选定工作区中的表记录做删除标记。

NOOPTIMIZE

禁止 DELETE 进行 Rushmore 优化。

说明

标有删除标记的记录在使用 PACK 前并不从表上做物理删除。标有删除标记的记录可以用 RECALL 恢复（清除标记）。

示例

下列示例打开 testdata 数据库中的 customer 表。对于 country 字段中包含 USA 的所有记录，使用 DELETE 命令标记为删除。显示所有标记为删除的记录。使用 RECALL ALL 命令恢复所有标记为删除的记录。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer && 打开 Customer 表
```

```
DELETE FOR country = 'USA' && 删除标记
CLEAR
LIST FIELDS company, country FOR DELETED( ) && 列出已标记删除的记录
RECALL ALL && 取消全部记录的删除标记。
```

请参阅

DELETE-SQL, DELETED(), PACK RECALL, SET DELETED

DELETE-SQL 命令

标记删除记录。

语法

```
DELETE FROM [DatabaseName!]TableName
  [WHERE FilterCondition1 [AND | OR FilterCondition2 ...]]
```

参数描述

FROM [DatabaseName!]TableName

指定表中哪些记录标记为删除。

DatabaseName! **指定包含表的非作用数据库名称。如果数据库不是当前使用的数据库，您必须要包含表所在数据库的名称。在数据库名称之后与表名之前使用叹号 (!) 分隔。**

WHERE FilterCondition1 [AND | OR FilterCondition2 ...]

指定 Visual FoxPro 标记删除的记录的规则。

FilterCondition 指定一定条件，符合的该条件的记录将标记为删除。您可以包含随意数量的过滤条件，对它使用 AND 或 OR 操作符连接。您还可以使用 NOT 操作符将逻辑表达式的值取反，或使用 EMPTY() 函数检查一个空字段。

说明

在执行 PACK 命令之前，标记为删除的记录实际上并没有从表中删除。使用 RECALL 命令可以恢复（取消标记）已做删除标记的记录。

如果 SET DELETED 设置为 ON，所有包含范围的命令均忽略已做删除标记的记录。

与 DELETE, DELETE 不同——当在为共享而打开的表中标记多个删除记录时，SQL 使用记录锁定。这样做减少了记录在多用户情况下的冲突，但是可能会降低性能。如果要获取最大性能，应以独占方式打开表或使用 FLOCK() 函数锁定表。

示例

以下示例打开了 testdata 数据库中的 customer 表。使用 DELETE-SQL 将 country 字段中值为 USA 的记录标记为删除。显示所有标记为删除的记录。使用 RECALL ALL 命令将所有标记为删除的记录取消标记。

```
CLOSE DATABASES
CLEAR

OPEN DATABASE HOME(2)+"Data\testdata"
USE customer  && Open Customer table

DELETE FROM customer WHERE country = "USA"  && 标记删除
```

```
CLEAR
LIST FIELDS company, country FOR DELETED( ) && 列出标记的记录
* 如果文件在此时整理，将删除记录
WAIT WINDOW "Records currently marked for deletion"+CHR(13) + ;
    "Press any key to revert..."

* 取消所有标记删除记录的标记
RECALL ALL
CLEAR
* 验证恢复的记录
COUNT FOR DELETED( )=.T. TO nDeleted

* 将 nDeleted 转化为字符串并显示信息
WAIT WINDOW ALLTRIM(STR(nDeleted)) + " records marked for deletion."
```

请参阅

DELETE, DELETED( ), PACK, RECALL, SET DELETED

DELETE CONNECTION 命令

从当前数据库中删除一个命名连接。

语法

DELETE CONNECTION ConnectionName

参数描述

ConnectionName

指定要从当前数据库中删除的命名连接的名称。

说明

删除一个命名连接定义并不关闭任何活动的连接

DELETE CONNECTION 要求独立使用数据库。要以独占方式打开数据库，需在 OPEN DATABASE 中包含 EXCLUSIVE。

示例

下面的示例假定名为 MyFoxSQLNT 的 ODBC 数据源是可用的，数据源的用户标识 (ID) 为“sa”，打开 testdata 数据库，创建名为 Myconn 的连接。DISPLAY CONNECTIONS 用来显示数据库中的命名连接。然后用 DELETE CONNECTION 将连接从数据库中删除。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')

CREATE CONNECTION Myconn DATASOURCE "MyFoxSQLNT" USERID "sa"
DISPLAY CONNECTIONS  && 显示数据库中已命名的连接
DELETE CONNECTION Myconn && 移动刚刚建立的连接
```

请参阅

CREATE CONNECTION, MODIFY CONNECTION, RENAME CONNECTION

DELETE DATABASE 命令

从磁盘上删除数据库。

语法

DELETE DATABASE DatabaseName | ?
[DELETETABLES] [RECYCLE]

参数描述

DatabaseName

指定要从磁盘上删除的数据库的名称。指定的数据库不能打开。*DatabaseName* 可以包含数据库的路径和数据库名。

?

显示“打开”对话框，从中可以指定要从磁盘上删除的数据库名。

DELETETABLES

从磁盘上删除包含在数据库中的表和包含表的数据库。

RECYCLE

指定不将数据库从磁盘中立即删除并将其放置 Windows 95 回收站。

说明

最好使用 DELETE DATABASE 从磁盘上删除一个数据库。DELETE DATABASE 从数据库的表中删除对数据库的引用。

如果 SET SAFETY 设为 ON，Visual FoxPro 会提示是否要删除指定的数据库。如果 SET SAFETY 设置为 OFF，自动从磁盘上删除数据库。

示例

此示例创建了一个名为 people 的数据库。创建一个名为 friends 的表并将其自动添加到数据库中。使用 DISPLAY TABLES 命令显示数据库中的表，并使用 DISPLAY DATABASES 命令显示数据库中有关表的信息。

使用具有 DELETETABLES 选项的 DELETE DATABASE 命令从磁盘移除数据库及其中的 friends 表。

```
CLOSE ALL
CREATE DATABASE people
CREATE TABLE friends (FirstName C(20), LastName C(20))
CLEAR
DISPLAY TABLES  && 显示数据库中的表
DISPLAY DATABASES  && 显示表的信息
CLOSE ALL
DELETE DATABASE people DELETETABLES
```

请参阅

ADD TABLE, CLOSE DATABASES, DBC( ), DBGETPROP( ), DBSETPROP( ), DISPLAY TABLES, OPEN DATABASE, REMOVE TABLE

DELETE FILE 命令

从磁盘上删除文件。

语法

DELETE FILE [FileName | ?] [RECYCLE]

参数描述

FileName

指定要删除的文件。FileName可以包含如 * 和 ? 这样的通配符。例如，如果要删除当前目录中扩展名为 .bak 的备份文件，则可执行 DELETE FILE *.bak 命令。

?

显示“删除”对话框，可以从中选择要删除的文件。

RECYCLE

指定不将文件立即从磁盘中删除，而是将其放入 Windows 95 回收站中。

注意 使用此命令删除的任意文件均不能恢复。即使 SET SAFETY 的设置为 ON，在删除文件之前，您也不能得到警告信息。

说明

当执行 DELETE FILE 命令时，想删除的文件不能是打开的。如果文件所在的驱动器

或路径与默认的不同，文件名必须包含路径和扩展名。文件名中不能包含通配符。

在删除数据库表之前，应该以表名为参数执行 REMOVE TABLE 从数据库中删除对表的引用。如果删除的表带有 .fpt 备注文件，要确保删除此备注文件。

示例

在以下示例中，Customer.dbf 的结构和所有 country 字段为 USA 的记录都复制在名为 backup 的表中。然后将 backup 中的数据复制到一个 temp 文本文件中，打开此文件并在其关闭时将其删除。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer  && 打开 Customer 表

COPY STRUCTURE TO backup
USE backup
APPEND FROM customer FOR country = 'USA'
COPY TO temp TYPE DELIMITED

WAIT WINDOW 'Press Esc to close and erase temp.txt' NOWAIT
MODIFY FILE temp.txt NOEDIT
DELETE FILE temp.txt
? IIF(FILE('temp.txt'),'File not deleted','File deleted')
USE
DELETE FILE backup.dbf
```

请参阅

ERASE, REMOVE TABLE, SET SAFETY

DELETE TAG 命令

从复合索引 (.cdx) 文件中删除标识。

语法

DELETE TAG TagName1 [OF CDXFileName1]
[, TagName2 [OF CDXFileName2]] ...

–或者–

DELETE TAG ALL [OF CDXFileName]

参数描述

TagName1 [OF CDXFileName1] [, TagName2 [OF CDXFileName2]] ...

指定要从复合索引文件中删除的标识。可以使用包含一系列用逗号隔开的标识名的 DELETE TAG 命令删除多个标识。如果在打开的多个索引文件中有两个或更多同名的标识，可以通过包含 OF *CDXFileName* 从某一特定的索引文件中删除一个标识。

ALL [OF CDXFileName]

从复合索引文件中删除所有标识。如果当前表中有结构复合索引文件，就从该索引文件中删除所有的标识，并从磁盘上删除该索引文件。表头上标明有一个关联的结构复合索引文件存在的标记同时被删除。使用 ALL 时带有 OF *CDXFileName* 可从另一个打开的复合索引文件中删除所有标识，而不是从结

构复合索引文件中删除标识。

说明

用 INDEX 创建的复合索引文件包含与索引项相对应的标识。DELETE TAG 用来从打开的复合索引文件中删除一个或多个标识，可以只删除在当前工作区中打开的复合索引文件的标识。如果删除了一个复合索引文件中的所有标识，文件就从磁盘上删除。.Visual FoxPro 首先在结构复合索引文件（如果已打开）中查找一个标识，如果此标识不在结构复合索引文件中，Visual FoxPro 就在其他打开的复合索引文件查找。

如果试图删除一个主索引标识或候选索引标识，且 SET SAFETY 设置为 ON，Visual FoxPro 就会发出警告。

请参阅

COPY INDEXES, COPY TAG, INDEX, TAG()

DELETE TRIGGER 命令

从当前数据库的表中删除触发器。

语法

DELETE TRIGGER ON TableName FOR DELETE | INSERT | UPDATE

参数描述

TableName
指定要删除触发器的表名。

FOR DELETE | INSERT | UPDATE
指定要删除的触发器。包含 FOR DELETE 删除“删除”触发器，包含 FOR INSERT 删除“插入”触发器，包含 FOR UPDATE 删除“更新”触发器。

说明

使用 CREATE TRIGGER 为表创建“删除”、“插入”或“更新”触发器。

示例

以下示例创建了一个更新触发器，该触发器防止 customer 表中的 maxordamt 字段输入超过 50 的值。使用 DISPLAY DATABASE 命令显示更新触发器。然后使用 DELETE TRIGGER 命令移除更新触发器，并再次执行 DISPLAY DATABASE 命令验证移除的更新触发器。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')  && 打开 testdata 数据库
USE CUSTOMER  && 打开 customer 表

CREATE TRIGGER ON customer FOR UPDATE AS maxordamt <= 50
CLEAR
DISPLAY DATABASE
DELETE TRIGGER ON customer FOR UPDATE
DISPLAY DATABASE
```

请参阅

ADD TABLE, AERROR(), CREATE DATABASE, CREATE TRIGGER, DISPLAY DATABASE, LIST DATABASE, OPEN DATABASE

DELETE VIEW 命令

从当前数据库中删除一个 SQL 视图。

语法

DELETE VIEW ViewName

参数描述

ViewName

指定从当前数据库中删除的视图名。

说明

可使用 CREATE SQL VIEW 创建一个 SQL 视图，并向当前数据库中添加该视图。如果打开了一个 SQL 视图，然后删除它，那么包含 SQL 视图结果的临时表不会被关闭。

DELETE VIEW 要求独占地使用数据库。要以独占方式打开数据库，可在 OPEN DATABASE 中包含 EXCLUSIVE。

示例

以下示例打开了 testdata 数据库。使用 CREATE SQL VIEW 命令创建名为 myview 本地 SQL 视图，显示“视图设计器”，允许您为 SQL 视图指定表和条件。在保存 SQL 视图之后，使用 DISPLAY DATABASE 命令显示有关 SQL 视图的信息。然后使用 DELETE VIEW 命令删除名为 myview 的 SQL 视图。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')

CREATE SQL VIEW myview
CLEAR
DISPLAY DATABASE
DELETE VIEW myview
```

请参阅

CREATE SQL VIEW, OPEN DATABASE

DeleteColumn 方法

从一个表格控件中删除一个列对象

语法

Grid.DeleteColumn[(nIndex)]

参数描述

nIndex

指定在表格中的列的编号。如果不指定 *nIndex*，就删除表格中活动的列。

应用于

Grid

请参阅

AddColumn 方法, AddObject 方法, RemoveObject 方法

Deleted 事件

当用户在记录上做删除标记、清除一个删除标记，或执行 DELETE 命令时，此事件发生。

语法

```
PROCEDURE Grid.Deleted
LPARAMETERS nRecNo
```

参数描述

nRecNo

返回被删除的行记录号。

应用于

表格

请参阅

DELETE, DeleteMark 属性

DELETED( ) 函数

若当前记录被标识删除，返回真（.T.）。

语法

DELETED([cTableAlias | nWorkArea])

返值类型

逻辑值

参数描述

cTableAlias | nWorkArea

可以用 *nWorkArea* 指定工作区号，或用 *cTableAlias* 指定表别名检查在另一工作区打开的表中当前记录的状态。如果在指定的工作区中没有打开的表，DELETE( ) 就返回“假”。

如果省略了 *cTableAlias* 和 *nWorkArea*，该函数就返回当前工作区中当前记录的删除状态。

说明

如果记录标有删除标记，DELETED( ) 函数就返回“真”(.T.)；否则，DELETED( ) 返回“假”(.F.)。

可以用 DELETE 和 DELETE - SQL 给记录标上删除标记，用 RECALL 清除它们的标记。

如果已对 DELETED( ) 建立了索引，那么 Rushmore 将对测试记录删除状态的查询进行优化。

有关使用 Rushmore 优化查询的内容，请参阅稍后部分的“SET OPTIMIZE 命令”及《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十五章“优化应用程序”中的“掌握 Rushmore 技术”。

示例

以下示例打开了 testdata 数据库中的 customer 表。使用 DELETE-SQL 将 country 字段中值为 USA 的记录标记为删除。使用 DELETED( ) 显示所有标记为删除的记录。使用 RECALL ALL 命令将所有标记为删除的记录取消标记。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer  && 打开 Customer 表

DELETE FROM customer WHERE country = 'USA'  && 标记删除
CLEAR
LIST FIELDS company, country FOR DELETED( ) && 列出标记为删除的记录
RECALL ALL  && 取消已做删除标记记录的标记
```

请参阅

DELETE, DELETE-SQL, PACK, RECALL, SET DELETED

DeleteMark 属性

指定在表格控件中是否出现删除标记列。设计和运行时可用。

语法

Grid.DeleteMark[= lExpr]

参数描述

lExpr

下表列出了 DeleteMark 属性的设置：

设置	说明
“真”(.T.)	（默认值）删除标记列作为最左列出现在表格上。
“假”(.F.)	删除标记列不出现。

说明

删除标记列使用户能标记要删除的记录。用户可在记录的删除列中单击来删除记录。

应用于

表格

请参阅

DESCENDING( ) 函数

返回逻辑值，指定是否使用 DESCENDING 关键字创建了索引标记，或在 USE、SET INDEX 或 SET ORDER 等命令中是否包含了 DESCENDING 关键字。

语法

DESCENDING([CDXFileName,] nIndexNumber [, nWorkArea | cTableAlias])

返值类型

逻辑值

参数描述

CDXFileName

可用 *CDXFileName* 指定一个复合索引文件名。所指定的复合索引文件可以是和表一起自动打开的结构复合索引文件，或者是一个独立的复合索引文件。

nIndexNumber

数值型表达式 *nIndexNumber* 指定 DESCENDING( ) 测试的索引标识或索引文件。*nIndexNumber* 通常是一个从 1 开始且以 1 为单位递增的整数，代表了各

个索引标识。

如果 *nIndexNumber* 为 1，就为 .idx 主控单项索引文件或主控索引标识（如果存在）返回一个值。

随着 *nIndexNumber* 递增，可以为结构复合索引文件（如果存在）中的每个标识返回值。为标识返回的值按照它们在结构复合索引文件中创建的顺序返回。

为结构复合索引文件中的所有标识返回了值以后，再为所有打开的独立复合索引文件中的每个标识返回值。为标识返回的值按照它们在独立复合索引文件中创建的顺序返回。

如果 *nIndexNumber* 大于打开的单项索引 .idx 文件和结构复合以及独立复合索引标识的总数，就返回空字符串。

nWorkArea | cTableAlias

为在非当前工作区的索引文件或标识返回值。*nWorkArea* 指定工作区号，*cTableAlias* 指定表别名。

如果没有指定的表别名，Visual FoxPro 将产生错误信息。

说明

可以用两种方法对一个表中的记录进行降序排列：

- 可以在 INDEX 命令中包含 DESCENDING 关键字实现在复合索引 .cdx 中创建一个降序索引标识。
- 可以在 USE、SET INDEX 或 SET ORDER 中包含 DESCENDING 关键字，

指定主控索引标识或主控单项索引 (.idx) 文件为降序索引。

DESCENDING( ) 可以确定一个索引标识是否以降序创建。如果所指定的索引标识是用 DESCENDING 关键字创建的，则 DESCENDING( ) 返回“真” (.T.)。

DESCENDING( ) 也可以确定主控索引标识或主控索引文件是否为降序。如果指定的主控索引标识或一个单项索引 (.IDX) 文件在 USE、SET INDEX，或 SET ORDDER 中包含了 DESCENDING 关键字，DESCENDING( ) 就返回“真” (.T.)。

如果不包含任何可选的参数，DESCENDING( ) 就为主控索引标识或主控索引文件返回一个值。如果不包含任何可选的参数，且主控索引标识或 .IDX 文件无效（例如，执行了 SET ORDER TO 按物理记录顺序对表排序），DESCENDING( ) 返回“假” (.F.)。

示例

以下示例打开了 testdata 数据库中 customer 表。使用 FOR...ENDFOR 创建循环语句，在其中包含检查 customer 结构化索引中每个索引标记的降序状态。显示每个结构化索引标记的名称和它们的降序状态。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE Customer    && 打开 customer 表
CLEAR

FOR nCount = 1 TO 254
   IF !EMPTY(TAG(nCount))  && 检查索引中的标记
   ? TAG(nCount) + ' Descending? ' && 显示标记名称
   ?? DESCENDING(nCount)  && 显示降序状态
   ELSE
      EXIT  && 当不再发现标记时退出循环语句
```

```
    ENDIF
ENDFOR
```

请参阅

INDEX, SET INDEX, SET ORDER, USE

Description 属性

对于文件对象，是该文件的说明。对于一个服务程序对象，是服务程序类的说明。设计和运行时可用。

语法

Object.Description[= cExpression]

参数描述

cExpression

指定文件或服务程序对象的说明。

对于文件对象，默认值为空字符串。通过在“项目”菜单中选择“编辑说明”，可以为项目中的一个文件指定说明。

对于服务程序。默认值是服务程序类的名称。也可以在“项目信息”对话框的“服务程序”选项卡中为一个服务程序指定说明。在注册表中也出现这个

服务程序对象的说明。

应用于

文件对象，服务程序对象

请参阅

Name 属性

Desktop 属性

指定表单是否放在 Visual FoxPro 主窗口中。设计时可用，运行时只读。

语法

Object.Desktop[= lExpr]

参数描述

lExpr

下表列出了 Desktop 属性的设置：

设置	说明
“真” (.T.)	表单可放在 Windows 桌面的任何位置。
“假” (.F.)	（默认值）表单包含在 Visual FoxPro 主窗口中。

说明

当 ShowWindow 属性被设置为 2 时，Desktop 属性将被忽略。

应用于

表单，_SCREEN

请参阅

AlwaysOnTop 属性，AutoCenter 属性，MaxHeight 属性，MaxLeft 属性，MaxTop 属性，MaxWidth 属性，MinHeight 属性，MinWidth 属性，ShowWindow 属性

Destroy 事件

当释放一个对象时发生。

语法

```
PROCEDURE Object.Destroy
[LPARAMETERS nIndex]
```

参数描述

nIndex

在控件数组中，唯一地指定一个控件。

说明

一个容器对象的 Destroy 事件在它所包含的任何一个对象的 Destroy 事件之前触发；容

器的 Destroy 事件在它所包含的各对象释放之前可以引用它们。

应用于

ActiveDoc 对象，复选框，组合框，命令按钮，容器对象，控件对象，临时表，自定义，数据环境，编辑框，表单，表单集，表格，图像，标签，线条，列表框，OLE 绑定型控件，OLE 容器控件，选项按钮，页面，页框，ProjectHook 对象，关系，形状，微调，文本框，计时器，工具栏

请参阅

AddObject 方法, CREATEOBJECT(), Init 事件, RELEASE

_DIARYDATE 系统变量

包含“日历/日记”中的当前日期。

语法

_DIARYDATE = dExpression

参数描述

dExpression

为“日历/日记”指定日期。

说明

用 _DIARYDATE，可用选择的特定日期显示“日历/日记”，或返回选定的“日历/日记”日期。

默认情况下，当前日期存储在 _DIARYDATE 中。当打开“日历/日记”时就选定了当前日期。也可在 _DIARYDATE 中存储一个不同的日期，用新选定的日期打开“日历/日记”。

当“日历/日记”打开时，选择新的日期会导致在 _DIARYDATE 中存储新的日期。

当关闭“日历/日记”时，_DIARYDATE 中就包含最近选定的日期。

示例

在下面的程序示例中，用选定的 March，28，2001 打开“日历/日记”，_DIARYDATE 的值显示在下面。将 July，4，1776 存入 _DIARTDATE，重新显示 _DIARYDATE。然后该程序使用 DATE( ) 函数选择当前日期，并显示 _DIARYDATE。

```
SET CENTURY ON
STORE {^2001-03-28} TO _DIARYDATE
=MESSAGEBOX(DTOC(_DIARYDATE),64)
ACTIVATE WINDOW calendar
=MESSAGEBOX("Change date to July 4, 1776",48))
STORE {^1776-07-04} TO _DIARYDATE
=MESSAGEBOX(DTOC(_DIARYDATE),64)
=MESSAGEBOX("Change date to today's date",48)
STORE DATE( ) TO _DIARYDATE
=MESSAGEBOX(DTOC(_DIARYDATE),64)
RELEASE WINDOW calendar
```

请参阅

DIFFERENCE( ) 函数

返回 0 到 4 间的一个整数，表示两个字符表达式间的发音差别 (phonetic difference)。

语法

DIFFERENCE(cExpression1, cExpression2)

返值类型

数值型

参数描述

cExpression1, cExpression2

指定用 DIFFERENCE( ) 比较的两个字符表达式。

说明

当对表进行搜索而又不知道某一项的确切拼写时，DIFFERENCE( ) 很有用处。两个表达式拼写越相似，DIFFERENCE( ) 返回的数值就越高。如果字符表达式拼写非常相似，DIFFERENCE( ) 就返回 4。对于两个在语音上几乎不同的表达式，DIFFERENCE( ) 返回 0。

示例

```
STORE 'Smith' TO gcName1
```

```
STORE 'Smythe'  TO gcName2
STORE 'Smittie' TO gcName3
STORE '' TO gcName4
CLEAR
? DIFFERENCE(gcName1, gcName2)  && 显示数值 4
? DIFFERENCE(gcName1, gcName3)  && 显示数值 4
? DIFFERENCE(gcName1, gcName4)  && 显示数值 1
```

请参阅

SOUNDEX()

DIMENSION 命令

创建一维或两维数组。

语法

DIMENSION ArrayName1(nRows1 [, nColumns1])
[, ArrayName2(nRows2 [, nColumns2])] ...

参数描述

ArrayName1

指定数组名。可以通过包含多个数组名（*ArrayName2*, *ArrayName3* 等）用一个 DIMENSION 命令创建多个数组。

nRows1 [, nColumns1]

指定要创建的数组大小。如果只包含 *nRows1*，就创建一维数组。一维数组含有一列和 *nRowsl* 行。例如，下列命令创建一个名为 gaArrayOne 的一列十行的数组。

```
DIMENSION gaArrayOne(10)
```

若要创建二维数组，应包含 *nRows1* 和 *nColumn1*。*nRows1* 指定数组中的行数，*nColumns1* 指定列数。下列命令创建一个名为 gaArrayTwo 的二行四列的二维数组：

```
DIMENSION gaArrayTwo(2,4)
```

使用 DIMENSION 创建数组时，必须指定大小。在下面的示例中，创建了三个数组：前面两个示例中的数组 gaArrayOne 和 gaArrayTwo，以及第三个名为 gaArrayThree 的数组：

```
DIMENSION gaArrayOne(10), gaArrayTwo(2,4), gaArrayThree(3,3)
```

在 DIMENSION 或 DECLARE 中，可以用方括号或圆括号括起表达式。例如，下面的两个命令创建相同的数组：

```
DIMENSION gaArrayOne(10), gaArrayTwo[2,4], gaArrayThree(3,3)
DIMENSION gaArrayOne[10], gaArrayTwo(2,4), gaArrayThree[3,3]
```

说明

DIMENSION 的操作和语法与 DECLARE 相同。

数组元素... 一个数组的大小决定了它包含的元素数目。数组中的每个元素可以存储

一条信息。要确定一个数组中包含的元素数目和存储的信息量，可用数组的行(*nRows1*)数乘以数组的列数(*nColumns1*)。

数组元素可以包含任何类型的数据，并在最初创建数组时将它们初始化为“假”(.F.)。如果SET COMPATIBLE设置为FOXPLUS或OFF（默认设置），那么可以用STORE将一个数组中所有元素初始化为相同的值。例如，

```
DIMENSION gaArray(10,3)
STORE 'initial' TO gaArray
```

数组下标... 通过下标来引用数组中的元素。每个数组元素都有唯一一个可以识别它的数值下标。如果数组是一维的，则数组元素的下标与它的行号相同。例如，一个一维数组第三行中元素的下标就是3。

通过两个下标来引用二维数组中的元素。第一个下标表明元素的行位置，第二个下标表明元素的列位置。例如，一个二维数组第三行第四列中的元素下标为3，4。有关数组元素下标的进一步讨论，请参阅ASUBSCRIPT()。

数组中第一个元素的下标通常从1开始。如果数组是二维的，也可以用单个下标引用它。使用AELEMENT()根据数组的行列下标返回单个下标；使用ASUBSCRIPT()根据单个下标返回行列下标。

重新定义数组的维数... 重新执行DIMENSION命令可以改变数组的大小和维数。数组的大小可以增加或减小，一维数组可以转换为二维数组，二维数组可以降低为一维数组。

如果数组中元素的数目增加了，就将原数组中所有元素内容复制到维数重新调整过的数组中，增加的数组元素初始化为“假”(.F.)。

示例

示例 1 演示了一维数组增量的结果（注意，如果您是在“命令”窗口中输入这些命令将是“PUBLIC”的。但是如果您将它们复制到程序中并运行，这些命令将是“PRIVATE”的。

如果数组中元素的数目是递减的，删除元素和它们包含的数据。当一维数组重新定义为二维数组时，原始一维数组的内容按照从元素到行的顺序复制到新的数组中。

在示例 2 中，一个一维数组转换为一个二维数组。一维数组元素的内容首先复制到新的数组的第一行中，再向数组的第二行复制，以此类推。附加的元素初始化为 (.F.)。

当二维数组转换为一维数组时，原始的二维数组的内容按照行到元素的顺序复制到新的数组中。第一行第一个元素成为一维数组的第一个元素，第一行第二个元素成为第二个元素，以此类推。

使用 ADEL( ) 或 AINS( ) 删除或插入数组元素、行和列。使用 APPEND FROM ARRAY、COPY TO ARRAY、SCATTER 和 GATHER 传输在表记录和数组之间的数据。

在示例 3 中，创建了一个二维数组并加载了数据。显示数据元素和它们包含的数据。

```
* 示例 1
DIMENSION marray(2)
STORE 'A' TO marray(1)
STORE 'B' TO marray(2)
CLEAR
DISPLAY MEMORY LIKE marray
DIMENSION marray(4)
DISPLAY MEMORY LIKE marray
WAIT WINDOW

* 示例 2
DIMENSION marrayone(4)
```

```
STORE 'E' TO marrayone(1)
STORE 'F' TO marrayone(2)
STORE 'G' TO marrayone(3)
STORE 'H' TO marrayone(4)
CLEAR
DISPLAY MEMORY LIKE marrayone
DIMENSION marrayone(2,3)
DISPLAY MEMORY LIKE marrayone
WAIT WINDOW
```

* **示例** 3

```
DIMENSION sample(2,3)
STORE 'Goodbye' TO sample(1,2)
STORE 'Hello' TO sample(2,2)
STORE 99 TO sample(6)
STORE .T. TO sample(1)
CLEAR
DISPLAY MEMORY LIKE sample
```

请参阅

ACOPY( ), ADEL( ) , ADIR( ) , AELEMENT( ) , AFIELDS( ) , AFONT( ) , AINS( ) , ALEN( ) , APPEND FROM ARRAY, ASCAN( ) , ASORT( ) , ASUBSCRIPT( ) , COPY TO ARRAY , DECLARE, EXTERNAL, GATHER, PRIVATE, PUBLIC, SCATTER, SET COMPATIBLE , STORE

DIR 或 DIRECTORY 命令

显示磁盘目录（或文件夹）中文件的信息。

语法

DIR | DIRECTORY [ON Drive]
[[LIKE] [Path] [FileSkeleton]]
[TO PRINTER [PROMPT] | TO FILE FileName]

参数描述

ON Drive

指定磁盘目录（或文件夹）所在的驱动器名。

[LIKE] [Path] [FileSkeleton]

指定文件所在目录（或文件夹）的路径。如果省略了 ON *Drive*，路径可以包含驱动器名。

可包含 *FileSkeleton* 显示其他文件类型而不仅是表的信息。*FileSkeleton* 是支持通配符的文件梗概。例如，要想列出当前目录中的所有程序文件，请执行下列命令：

```
DIR *.PRG
```

在 Visual FoxPro 中，可执行以下命令列出所有不带扩展名的文件：

DIR *.

TO PRINTER [PROMPT]

将 DIRECTORY 的输出定向到打印机。

在 Visual FoxPro 中，包含可选的 PROMPT 子句可在打印开始前显示一个打印对话框，在此对话框中可以调整打印机设置。PROMPT 关键字直接放置在 TO PRINTER 后面。

TO FILE FileName

将 DIR 的结果定向到 *FileName* 指定的文件中。如果文件已经存在，且 SET SAFETY 设置为 ON，则会提示您是否要改写此文件。

说明

可使用 DIR 显示文件的信息。

不带 LIKE 子句的 DIR 显示如下信息：

- 目录（或文件夹）中所有表的名称
- 每个表中记录的数目
- 每个表最近更新的日期
- 以字节数表示的每个表的大小（早期的 FoxBASE 格式中的表就以此表示）
- 每个表是否为数据库的组成部分
- 表所占的磁盘空间的总字节数（不包括相关的 .FPT 备注文件）
- 显示表的数目
- 磁盘上剩余的字节总数

除非用 *Drive* 或 *Path*（或两者同时使用）指定了驱动器和目录（或文件夹），否则显示默认的驱动器和目录（或文件夹）中有关表的信息。

示例

```
CLEAR
DIR &&目录或文件夹中的表。
DIR *.CDX &&显示在当前目录或文件夹中的索引文件。
DIR A*.DBF &&显示以“A”打头的表
DIR *.* &&显示所有文件，包括不带扩展名的文件。
```

请参阅

DISPLAY FILES

DIRECTORY( ) 函数

若在磁盘上存在指定的目录，返回真 (.T.)。

语法

DIRECTORY(cDirectoryName)

返值类型

逻辑值

参数描述

cDirectoryName

指定要查找的目录。如果您在此处指定的不是一个绝对路径，Visual FoxPro 将在 Visual FoxPro 的默认目录中寻找这个路径。

说明

Visual FoxPro 默认路径由 SET DEFAULT 命令指定。

请参阅

FILE( ), GETDIR( ) , GETFILE( ) , SET DEFAULT

DisabledBackColor, DisabledForeColor 属性

为一个禁止的控件指定背景色和前景色。设计和运行时可用。

语法

Control.DisabledBackColor[= nColor]

–或者–

Control.DisabledBackColor = RGB(nRedValue, nGreenValue, nBlueValue)

Control.DisabledForeColor[= nColor]

–或者–

Control.DisabledForeColor = RGB(nRedValue, nGreenValue, nBlueValue)

参数描述

nColor

指定一个数值来代表颜色。

nRedValue, nGreenValue, nBlueValue

分别指定组成控件的前景色或背景色的三种颜色深度，它们必须和 RGB( ) 一起使用，将三种颜色合并到一个颜色号中。

说明

所指定的颜色与灰色一起抖动。

注意 在属性窗口中，可双击任何颜色属性来显示“颜色”对话框。可以从这个对话框中选择或定义颜色。当关闭“颜色”对话框后，与所选颜色相应的红、绿、蓝的深度就变成这些属性的设置。

应用于

复选框，组合框，命令按钮，编辑框，标签，列表框，选项按钮，微调，文本框

注意 DisabledBackColor 属性不能应用于 CommandButton 控件。

请参阅

BackColor, ForeColor **属性**, **颜色概览**

DisabledItemBackColor, DisabledItemForeColor 属性

为列表框或组合框中的不可用项指定背景色或前景色。设计和运行时可用。

语法

Control.DisabledItemBackColor[= nColor]

–**或者** –

Control.DisabledItemBackColor RGB(nRedValue, nGreenValue, nBlueValue)

Control.DisabledItemForeColor[= nColor]

–**或者** –

Control.DisabledItemForeColor RGB(nRedValue, nGreenValue, nBlueValue)

参数描述

nColor

指定一个数值代表颜色。

注意 **在属性窗口中，可双击任何颜色属性来显示“颜色”对话框。可以在此对话框中选择或定义颜色。当关闭颜色对话框后，与所选颜色相应的红、绿、蓝的深度就变成这些属性的设置。**

有关详细内容，请参阅稍前部分的“BackColor，ForeColor **属性”中的颜色表。**

应用于

组合框，列表框

请参阅

BackColor, ForeColor **属性**, DisabledForeColor, DisabledBackColor **属性**, RGB(), SelectedBackColor, SelectedForeColor **属性**, SelectedItemBackColor,

SelectedItemForeColor 属性

DisabledPicture 属性

指定当禁止控件时要显示的图形。设计时和运行时可用。

语法

Control.DisabledPicture[= cPicture]

参数描述

cPicture

指定位图的完整路径和文件名，或指定数据库表中的通用字段名。

说明

如果在设计时设置了 DisabledPicture 属性，且所指定的文件不存在，Visual FoxPro 将显示一条错误信息，但属性设置仍然保留。如果 DisabledPicture 属性设置为一个不存在的文件，Visual FoxPro 在运行时将忽略该属性。

如果不设置 DisabledPicture 属性，Visual FoxPro 就使用 Picture 属性的设置来确定禁止控件时显示的图形。

对于 CheckBoxes 和 OptionButtons，如果 Style 属性设置成 1（图形方式），控件仅显

示位图。

应用于

复选框，命令按钮，选项按钮

请参阅

DownPicture 属性，Enabled 属性，Picture 属性，Style 属性

DISKSPACE( ) 函数

返回默认磁盘驱动器或卷 (Volume) 上可用的字节数。

语法

DISKSPACE([cVolumeName])

返值类型

数值型

参数描述

cVolumeName

指定一个驱动器或卷。若缺省，DISKSPACE( ) 则返回默认驱动器或卷中的可用字节。

说明

此参数可用来确定是否有足够的可用空间来备份文件，或执行诸如 SORT 这样的、需要为临时工作文件提供额外磁盘空间的命令。

默认的驱动器或卷由 SET DEFAULT 命令指定。

若在读磁盘或卷时发生错误，DISKSPACE( ) 返回 -1。在一些网络上，DISKSPACE( ) 返回的值对于大的网络驱动器可能是不精确的。

示例

以下示例使用 DISKSPACE( ) 函数以确定是否有足够的磁盘空间执行排序。

```
*** 排序前确认磁盘空间 ***
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer  && Opens Customer table

*** 确认表头大小 ***
gnTableHead = HEADER( )

*** 计算表的大小***
gnFileSize = gnTableHead + (RECSIZE( ) * RECCOUNT( ) + 1)
IF DISKSPACE( ) > (gnFileSize * 3)
    WAIT WINDOW 'Sufficient diskspace to sort.'
ELSE
    WAIT WINDOW 'Insufficient diskspace. Sort cannot be done.'
ENDIF
```

请参阅

HEADER( ), RECSIZE( ) , SET DEFAULT

DISPLAY 命令

在 Visual FoxPro 主窗口或用户自定义窗口中，显示与当前表有关的信息。

语法

DISPLAY
[[FIELDS] FieldList]
[Scope] [FOR lExpression1] [WHILE lExpression2]
[OFF]
[NOCONSOLE]
[NOOPTIMIZE]
[TO PRINTER [PROMPT] | TO FILE FileName]

参数描述

FIELDS FieldList

指定要显示的字段。如果省略了 FIELDS *FieldList*，就默认显示表中所有的字段。

除非明确地将备注字段名包含在字段列表中，否则不显示备注字段的内容。备注字段的显示宽度由 SET MEMOWIDTH 决定。

Scope

指定要显示的记录范围。只显示在指定范围中的记录。范围子句有：ALL、

NEXT *nRecords*、RECORD *nRecordNumber* 和 REST。包含 *Scope* 的命令只对活动工作区中的表进行操作。

有关范围子句的详细内容，请参阅“帮助”中的“范围子句”。

DISPLAY 默认的范围是当前记录 (NEXT 1)。

FOR lExpression1

指定只显示满足逻辑条件 *lExpression1* 的记录。这就允许您筛选出不想要的记录。

如果 *lExpression1* 是一个可优化表达式，Rushmore 优化用 DISPLAY ... FOR 创建的查询。要获得最佳效果，请在 FOR 子句中使用可优化表达式。

有关详细内容，请参阅稍后部分的“SET OPTIMIZE 命令”和《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十五章“优化应用程序”中的“掌握 Rushmore 技术”。

WHILE lExpression2

指定一个条件，只要逻辑表达式 *lExpression2* 求值为“真”(.T.)，就显示记录。

OFF

不显示记录号。如果省略了 OFF，就在每个记录前显示记录号。

NOCONSOLE

不向 Visual FoxPro 主窗口或活动的用户自定义窗口输出。

NOOPTIMIZE

使 DISPLAY 的 Rushmore 优化无效。

有关详细内容，请参阅稍后部分的“SET OPTIMIZE 命令”和《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十五章“优化应用程序”中的“掌握 Rushmore 技术”。

TO PRINTER [PROMPT]

将 DISPLAY 的结果定向输出到打印机。

在 Visual FoxPro 中，可以包含可选的 PROMPT 子句，在打印开始前显示一个对话框。在此对话框中可以调整打印机的设置，包括打印的数目和要打印的页。可调整的打印机设置依赖于当前安装的打印机驱动程序。请将 PROMPT 直接放在 TO PRINTER 后面。

TO FILE FileName

将 DISPLAY 的结果定向输出到 *FileName* 指定的文件中。如果文件已经存在，且 SET SAFETY 设置为 ON，将提示您是否要改写此文件。

说明

DISPLAY 显示当前表记录的内容和表达式的结果。如果有更多信息需要显示，而在窗口中显示不完，就显示第一屏信息，然后暂停。按任意键或在任意位置单击鼠标可以看下一屏的信息。DISPLAY 与 LIST 相似，只是 LIST 在显示同样信息时连续输出而不暂停。

DISPLAY 也可用来显示表达式的结果，它可以包括字母和数字的组合、变量、数组元素、字段和备注字段。如果 SET HEADINGS 为 ON，字段名和表达式将显示出来。

示例

下面的示例打开 testdata 数据库的 customer 表，显示第一个记录的内容。

```
CLOSE DATABASES
```

```
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer  && 打开 Customer 表

CLEAR
DISPLAY FIELD cust_id, company, contact OFF NEXT 10
```

请参阅

LIST, SET HEADINGS, SET MEMOWIDTH

DISPLAY CONNECTIONS 命令

显示当前数据库中与命名连接有关的信息。

语法

DISPLAY CONNECTIONS
[TO PRINTER [PROMPT] | TO FILE FileName]
[NOCONSOLE]

参数描述

TO PRINTER [PROMPT]

将 DISPLAY COMNNECTIONS 的结果定向输出到打印机。

在 Visual FoxPro 中，可以包含可选的 PROMPT 子句，在打印开始前显示“打印”对话框。应将 PROMPT 直接放在 TO PRINTER 后面。

TO FILE FileName

将 DISPLAY CONNECTIONS 的结果定向输出到 *FileName* 指定的文件中。如果文件已经存在，且 SET SAFETY 为 ON，Visual FoxPro 将提示您是否要改写此文件。

NOCONSOLE

不向 Visual FoxPro 主窗口或活动的用户自定义窗口输出。

说明

DISPLAY CONNECTIONS 显示当前数据库中的连接名、数据源和连接字符串。请使用 DBGETPROP( ) 函数返回有关当前数据库中连接的详细信息。

示例

下面的示例假定一个名为 MyFoxSQLNT 的 ODBC 数据源是可用的，数据源的用户标识 ID 为“sa”。打开 testdata 数据库并创建一个名为 Myconn 的连接。DISPLAY CONNECTIONS 用来显示数据库中的命名连接。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')

CREATE CONNECTION Myconn DATASOURCE "MyFoxSQLNT" USERID "sa"
CLEAR
DISPLAY CONNECTIONS  && 显示数据库中的命名连接
```

请参阅

CREATE CONNECTION, DELETE CONNECTION, DBGETPROP( ), LISTCONNECTIONS , RENAME CONNECTION

DISPLAY DATABASE 命令

显示有关当前数据库的信息。

语法

```
DISPLAY DATABASE
  [TO PRINTER [PROMPT] | TO FILE FileName]
  [NOCONSOLE]
```

参数描述

TO PRINTER [PROMPT]

将 DISPLAY DATABASE 的结果定向输出到打印机。在 Visual FoxPro 中可以包含可选的 PROMPT 子句，在打印开始前显示“打印”对话框。应将 PROMPT 直接放在 TO PRINTER 后面。

TO FILE FileName

将 DISPLAY DATABASE 的结果定向输出到 *FileName* 指定的文件中。如果文件已经存在，且 SET SAFETY 设置为 ON，Visual FoxPro 将提示您是否要改写此文件。

NOCONSOLE

不向 Visual FoxPro 主窗口或活动的用户自定义窗口输出。

说明

可使用 DBGETPROP( ) 函数返回有关当前数据库的详细信息。

示例

下面的示例创建了一个名为 people 的数据库，同时创建了一个名为 friends 的表并自动添加到数据库中。DISPLAY TABLES 用来显示数据库中的表，DISPLAY DATABASES 用来显示数据库中有关表的信息。

```
CREATE DATABASE people
CREATE TABLE friends (FirstName C(20), LastName C(20))
CLEAR
DISPLAY TABLES &&显示数据库中的表
DISPLAY DATABASES &&显示表信息
```

请参阅

LIST DATABASE

DISPLAY DLLS 命令

在 Visual FoxPro 中，显示与 32 位 Windows 动态链接库 (DLL) 函数有关的信息。

语法

```
DISPLAY DLLS
  [TO PRINTER [PROMPT] | TO FILE FileName]
  [NOCONSOLE]
```

参数描述

TO PRINTER [PROMPT]

将 DISPLAY DLLS 的结果定向输出到打印机。

在 Visual FoxPro 中，可以包含可选的 PROMPT 子句，在打印开始前显示打印对话框。应将 PROMPT 直接放在 TO PRINTER 后面。

TO FILE FileName

将 DISPLAY DLLS 的结果定向输出到 *FileName* 指定的文件中。如果文件已经存在，且 SET SAFETY 设置为 ON，Visual FoxPro 将提示您是否要改写此文件。

NOCONSOLE

不向 Visual FoxPro 主窗口或活动的用户自定义窗口输出。

请参阅

DECLARE-DLL, LIST DLLS

DISPLAY FILES 命令

显示关于文件的信息。

语法

DISPLAY FILES

[ON Drive]
[LIKE FileSkeleton]
[TO PRINTER [PROMPT] | TO FILE FileName]

参数描述

ON Drive
指定文件所在的驱动器。

LIKE FileSkeleton
指定一个条件，Visual FoxPro只显示那些与文件梗概 *FileSkeleton* **匹配的文件的有关信息。文件梗概可以包含通配符，例如“?”和“*”。**

TO PRINTER [PROMPT]
将 DISPLAY FILES **的结果定向输出到打印机。**

在 Visual FoxPro **中，可以包含可选的** PROMPT **子句，在打印开始前显示“打印”对话框。在此对话框中，可以调整打印机的设置，包括打印的数目和要打印的页。可以调整的打印机设置依赖于当前安装的打印机驱动程序。应将** PROMPT **关键字直接放在** TO PRINTER **的后面。.**

TO FILE FileName
将 DISPLAY FILES **的结果定向输出到** *FileName* **指定的文件中。如果文件已经存在，且** SET SAFETY **设置为** ON**，**Visual FoxPro **将提示您是否要改写此文件。**

说明

可用 DISPLAY FILES **显示有关磁盘文件的信息。可以显示某一特定驱动器、路径或文件夹中的所有文件的信息，或者仅显示与某一文件梗概（包含“?”和“*”）相匹**

配的文件的有关信息

不带任何参数执行 DISPLAY FILES 将显示当前目录中表的信息。显示的信息包含以下内容：

- 表名
- 表中记录的数目
- 最近更新表的日期和时间
- 每个表的字节数
- 表是否为数据库的组成部分

示例

下面的示例显示了在目录 SAMPLES\DATA 中的数据库名。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')

CLEAR
DISPLAY FILES LIKE *.DBC
```

请参阅

DIR 或 DIRECTORY, LIST

返回总目录

DTOC（） 函数
DTOR（） 函数
DTOS（） 函数
DTOT（） 函数
DynamicAlignment 属性
DynamicBackColor, DynamicForeColor 属性
DynamicCurrentControl 属性
DynamicFontBold,DynamicFontItalic, DynamicFontStrikethru,
DynamicFontUnderline 属性
DynamicFontName 属性
DynamicFontOutline 属性
DynamicFontShadow 属性
DynamicFontSize 属性
DynamicInputMask 属性

DISPLAY MEMORY 命令

显示变量和数组的当前内容。

语法

```
DISPLAY MEMORY
  [LIKE FileSkeleton]
  [TO PRINTER [PROMPT] | TO FILE FileName]
  [NOCONSOLE]
```

参数描述

LIKE FileSkeleton

可以通过包含 LIKE 子句有选择地显示变量和数组的信息。如果包含 LIKE *FileSkeleton*，Visual FoxPro 只显示匹配 *FileSkeleton* 的变量和数组的内容。*FileSkeleton* 支持诸如“?”和“*”的通配符。例如，要显示所有以字母 A 开头的变量，可执行下面的命令：

```
DISPLAY MEMORY LIKE A*
```

TO PRINTER [PROMPT]

将 DISPLAY MEMORY 的结果定向输出到打印机。

可包含可选的 PROMPT 子句，在打印开始前显示“打印”对话框。在此对话框中，可调整打印机设置，包括打印数目和要打印的页数。可调整的打印机设置依赖于当前安装的打印机驱动程序。应将 PROMPT 关键字直接放在 TO PRINTER 后面。

TO FILE FileName

将 DISPLAY MEMORY 的结果定向输出到 *FileName* 指定的文件中。如果文件已经存在，且 SET SAFETY 设为 ON，Visual FoxPro 提示您是否要改写此文件。

NOCONSOLE

不向 Visual FoxPro 主窗口或活动的用户自定义窗口输出。

说明

DISPLAY MEMORY 显示所有当前定义的变量和变量数组的名称、类型、内容和状态。也显示已定义的变量的数目、已使用的字节总数和可用的额外变量数目。注意，已使用的字节总数表示字符类型的变量使用的内存。字符型变量是唯一需要使用附加内存的变量，附加内存是超出 MVCOUNT 配置项所指定的变量分配空间的内存。

DISPLAY MEMORY 也显示有关系统变量、菜单、菜单栏、菜单标题和窗口的信息。

示例

下面的示例创建了几个变量并赋值。首先 DISPLAY MEMORY 显示所有以“sam”开头的变量的值，然后再显示所有包含五个字母并以“exit.”结尾的变量的值。

```
STORE 'Goodbye' TO sample1
STORE 'Hello' TO sample2
STORE .T. TO texit
```

```
STORE .F. TO mexit

CLEAR
DISPLAY MEMORY LIKE sam*
DISPLAY MEMORY LIKE ?exit
```

请参阅

DECLARE, DIMENSION, LIST, STORE

DISPLAY OBJECTS 命令

显示有关一个对象或一组对象的信息。

语法

```
DISPLAY OBJECTS
  [LIKE cObjectSkeleton]
  [TO PRINTER [PROMPT] | TO FILE FileName]
  [NOCONSOLE]
```

参数描述

LIKE cObjectSkeleton

显示对象的信息。*cObjectSkeleton* 是支持通配符（“*”和“?”）的对象说明梗概。例如，要显示所有以 A 开头的对象，可使用以下命令：

DISPLAY OBJECTS LIKE A*

TO PRINTER [PROMPT]

将 DISPLAY OBJECTS 的结果输出到打印机。包含 PROMPT 子句会在打印开始前显示打印对话框。应将 PROMPT 关键字直接放在 TO PRINTER 后面。

TO FILE FileName

将 DISPLAY OBJECTS 的结果定向输出到 *FileName* 指定的磁盘文件中。如果文件已经存在，且 SET SAFETY 设为 ON，Visual FoxPro 就提示您是否改写此文件。

NOCONSOLE

不向 Visual FoxPro 主窗口或活动的用户自定义窗口输出。

说明

DISPLAY OBJECTS 显示所有已存在对象的下列信息：

- 属性和它们的值
- 方法
- 成员对象以及它们所基于的类或子类
- 对象所基于的类或子类
- 对象的类层次

用 DISPLAY OBJECTS 显示的信息将整个 Visual FoxPro 主窗口或用户自定义窗口填满后暂停，按任意键或在任意位置单击鼠标可看下一屏信息。DISPLAY 与 LIST 不同的是，LIST 在显示信息时连续输出而不暂停。

示例

下面的示例用 DEFINE CLASS 和 CREATEOBJECT() 从 Visual FoxPro 的 Form 基类创建了两个自定义类，FormChild 和 FormGrandChild。DISPLAY OBJECTS 显示关于对象和它们的属性的信息。

```
CLEAR
frmMyForm = CREATEOBJECT("FormGrandChild")
DISPLAY OBJECTS LIKE frm*
RELEASE frmMyForm

DEFINE CLASS FormChild AS FORM
ENDDEFINE

DEFINE CLASS FormGrandChild AS FormChild
ENDDEFINE
```

请参阅

LIST OBJECTS

DISPLAY PROCEDURES 命令

显示当前数据库中内部存储过程的名称。

语法

DISPLAY PROCEDURES
[TO PRINTER [PROMPT] | TO FILE FileName]
[NOCONSOLE]

参数描述

TO PRINTER [PROMPT]

将 DISPLAY PROCEDURES 的结果定向输出到打印机。

包含 PROMPT 可以在打印开始前显示“打印”对话框。应将 PROMPT 关键字直接放在 TO PRINTER 后面。

TO FILE FileName

将 DISPLAY PROCEDURES 的结果定向输出到 *FileName* 指定的磁盘文件中。如果文件已经存在，且 SET SAFETY 设为 ON，Visual FoxPro 提示您是否改写此文件。

NOCONSOLE

不向 Visual FoxPro 主窗口或活动的用户自定义窗口输出。

说明

可用 APPEND PROCEDURES、COPY PROCEDURES 或 MODIFY PROCEDURE 创建内部存储过程。

示例

下面的示例打开 testdata 数据库，用 DISPLAY PROCEDURE 显示数据库中的内部存储过程（如果存在）。如果数据库中没有内部存储过程，可运行 APPEND PROCEDURES 的示例向数据库中添加一个过程。

CLOSE DATABASES

```
OPEN DATABASE (HOME(2) + 'Data\testdata')

CLEAR
DISPLAY PROCEDURES && 显示数据库中的内部存储过程
```

请参阅

APPEND PROCEDURES 命令, COPY PROCEDURES 命令, CREATE DATABASE 命令, DISPLAY DATABASE 命令, LIST PROCEDURES 命令, MODIFY PROCEDURE 命令

DISPLAY STATUS 命令

显示 Visual FoxPro **环境的状态。**

语法

```
DISPLAY STATUS
  [TO PRINTER [PROMPT] | TO FILE FileName]
  [NOCONSOLE]
```

参数描述

TO PRINTER [PROMPT]

将 DISPLAY STATUS **的结果定向输出到打印机。**

包含可选的 PROMPT 子句可在打印开始前显示打印对话框。在此对话框中可以调整打印机设置，包括打印份数和要打印的页数。可调整的打印机设置依赖于当前已安装的打印机驱动程序。应将 PROMPT 关键字直接放在 TO PRINTER 后面。

TO FILE FileName

将 DISPLAY STATUS 的结果定向输出到 *FileName* 指定的文件中。如果此文件已经存在，且 SET SAFETY 设为 ON，Visual FoxPro 提示您是否要覆盖此文件。

NOCONSOLE

不向 Visual FoxPro 主窗口或活动的用户自定义窗口输出。

说明

DISPLAY 列出了有关当前 Visual FoxPro 环境的信息。信息的分类和每一类中的信息如下所示：

表和索引文件信息：

- 已打开的表
- 已打开的备注文件
- 表别名
- 表代码页
- 表关系
- 活动索引
- 索引文件关键字

- 主控索引文件或标识
- 已打开的结构复合文件
- 已打开的复合索引标识
- 每个已打开表的共享属性状态
- 每个表中当前锁定的记录
- EXCLUSIVE 使用设置
- LOCK 设置
- MULTILOCKS 设置
- SET REFRESH 值
- SET REPROCESS 值

已打开的低级文件信息：

- 已打开的低级文件
- 每个低级文件的文件句柄号
- 每个低级文件的文件指针位置
- 每个低级文件的“读-写”属性

附加的 Visual FoxPro 环境信息：

- 正在使用的过程文件
- 处理器类型
- Visual FoxPro 路径

- Visual FoxPro 默认的目录
- 打印目标
- 页边距设置
- 当前工作区
- SET 命令设置
- 当前加载的二进制模块
- Visual FoxPro 的 DDE 信息
- 当前代码页
- 当前排序序列
- 编译器代码页
- 当前数据格式
- 键盘宏和组合键
- UDF 参数的传递方法
- 文本合并选项
- 外部注册的 .DLL 函数

请参阅

LIST

DISPLAY STRUCTURE 命令

显示一个表文件的结构。

语法

```
DISPLAY STRUCTURE
  [IN nWorkArea | cTableAlias]
  [TO PRINTER [PROMPT] | TO FILE FileName]
  [NOCONSOLE]
```

参数描述

IN nWorkArea | cTableAlias

显示非当前工作区中的表的结构。*nWorkArea* 指定工作区号，*cTableAlias* 指定表别名。

TO PRINTER [PROMPT]

将 DISPLAY STRUCTURE 的结果定向输出到打印机。

在 Visual FoxPro 中可以包含可选的 PROMPT 子句，在打印开始前显示“打印”对话框。在此对话框中可调整打印机设置，包括打印份数和要打印的页数。可调整的打印机设置依赖于当前安装的打印机驱动程序。应将 PROMPT 关键字直接放在 TO PRINTER 的后面。

TO FILE FileName

将 DISPLAY STRUCTURE 的结果定向输出到 *FileName* 指定的文件中。如果此文件已经存在，且 SET SAFETY 设为 ON，Visual FoxPro 提示您是否要改写此文件。

NOCONSOLE

不向 Visual FoxPro 主窗口或活动的用户自定义窗口输出。

说明

可显示一个表的字段结构，表中每个字段名及其类型和宽度一起显示。如果字段是数值型、双精度型或浮点型的，还将显示小数点的位数。DISPLAY STRUCTURE 也显示每个字段是否支持 null 值。

DISPLAY STRUCTURE 还显示表中当前记录的编号和最近更新的日期。如果表中有一个相关备注字段，还显示备注字段块的大小。此外，还显示所有字段的总宽度和表的代码页。

表可以有一个和它一起打开的结构复合索引。如果结构复合索引中的标识与表中的一个字段同名，标识的顺序（升序或降序）和标识的排序序列就显示在字段名旁边。

如果用 SET FIELDS 限制了对表中字段的访问，一个尖括号就会出现在可以被访问的字段名旁边。

示例

下面的示例打开 testdata 数据库中的 customer 表，并用 DISPLAY STRUCTURE 显示表结构。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer  && 打开 Customer 表
```

CLEAR
DISPLAY STRUCTURE

请参阅

LIST

DISPLAY TABLES 命令

显示包含在当前数据库中所有的表和表的信息。

语法

```
DISPLAY TABLES
  [TO PRINTER [PROMPT] | TO FILE FileName]
  [NOCONSOLE]
```

参数描述

TO PRINTER [PROMPT]

将 DISPLAY TABLES 的结果定向输出到打印机。

可以包含 PROMPT 在打印开始前显示打印对话框。应将 PROMPT 关键字直接放在 TO PRINTER 后面。

TO FILE FileName

将 DISPLAY TABLES 的结果定向输出到 *FileName* 指定的磁盘文件中。如果此文件已经存在，且 SET SAFETY 设为 ON，Visual FoxPro 提示您是否要改写此文件。

NOCONSOLE

不向 Visual FoxPro 主窗口或活动的用户自定义窗口输出。

说明

返回的信息是 DISPLAY STATUS 所显示信息的一个子集。但是，用 DISPLAY TABLES 显示的信息只包含有相关表的信息，且不论表是否打开都显示信息。

显示的信息有：

- 表名
- 表路径

示例

下面的示例打开了 testdata 数据库中的 customer 表，DISPLAY TABLES 用来显示数据库中有关表的信息。

```
CLOSE DATABASES
SET PATH TO (HOME(2) + 'Data\')  &&设置到数据库的路径
OPEN DATABASE testdata  &&打开 testdata 数据库

CLEAR
DISPLAY TABLES  &&显示关于数据库中表的信息
```

请参阅

ADD TABLE, CLOSE DATABASES, CREATE DATABASE, LIST TABLES, OPEN DATABASE , REMOVE TABLE

DISPLAY VIEWS 命令

显示当前数据库中关于 SQL 视图的信息。

语法

```
DISPLAY VIEWS
  [TO PRINTER [PROMPT] | TO FILE FileName]
  [NOCONSOLE]
```

参数描述

TO PRINTER [PROMPT]

将 DISPLAY VIEW 的结果定向输出到打印机。

包含 PROMPT 可以在打印开始前显示“打印”对话框。应将 PROMPT 关键字直接放在 TO PRINTER 的后面。

TO FILE FileName

将 DISPLAY VIEWS 的结果定向输出到 *FileName* 指定的磁盘文件中。如果文件已经存在，且 SET SAFETY 设为 ON，Visual FoxPro 提示您是否要改写此文件。

NOCONSOLE
不向 Visual FOxPro 主窗口或活动的用户自定义窗口输出。

说明

使用 DBGETPROP() 可以返回当前数据库中关于 SQL 视图的详细信息。

SQL 视图可用 CREATE SQL VIEW 创建。

示例

下面的示例打开了 testdata 数据库，然后用 CREATE VIEW 创建一个名为 myview 的本地 SQL 视图。再显示视图设计器，为 SQL 视图指定表和条件。最后在保存创建的 SQL 视图后，显示数据库中关于 SQL 视图的信息。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
CREATE SQL VIEW myview

CLEAR
DISPLAY VIEWS
```

请参阅

CREATE DATABASE, CREATE SQL VIEW, DBGETPROP(), DISPLAY DATABASE, LIST VIEWS

DisplayCount 属性

指定在一个组合框控件的列表部分所显示项的最大数目。

语法

Object.DisplayCount[= nExpression]

参数描述

nExpression

指定指定在一个组合框控件的列表部分所显示项的最大数目。默认值为零；如果 nExpression 为零，则在列表中最多显示 7 项。

如果 nExpression 为 1，则在组合框控件的列表部分不显示滚动箭头。

应用于

组合框

请参阅

ColumnCount 属性, DisplayValue 属性

DisplayValue 属性

指定在一个列表框或组合框中选定项的第一列的内容。设计和运行时可用。

语法

[Form.]Control.DisplayValue[= Expr]

参数描述

Expr

如果 DisplayValue 是一个字符串，就指定选定项的第一列的值；如果 DisplayValue 是数值，就指定选定项的索引。

说明

设计和运行时可用。

当组合框或列表框中的项不止一列并且控件的 BoundColumn 属性设置大于 1 时，应使用 DisplayValue 属性。当 DisplayValue 是一个字符串时，组合框的文本框中的文本由 DisplayValue 指定。

注意 当 ComboBox 或 ListBox 只有一列时，如果控件的 DisplayValue 属性和 Value 属性都包含字符串，则两者有相同的设置。

应用于

组合框，列表框

请参阅

BoundColumn 属性, Value 属性

DMY( ) 函数

从一个日期型或日期时间型表达式返回一个“日-月-年”格式的字符表达式（例如，31 May 1995）。月名不缩写。

语法

DMY(dExpression | tExpression)

返值类型

字符型

参数描述

dExpression

指定的日期型表达式，DMY( ) 函数从该表达式返回“日-月-年”格式的字符串。

tExpression

指定的日期时间表达式，DMY( ) 函数从该表达式返回“日-月-年”格式的字符串。

说明

如果 SET CENTURY 设置为 OFF，DMY( ) 函数以 dd Month yy 格式（例如，16

February 95，不带世纪）返回一个字符串。如果 SET CENTURY 设为 ON，格式为“dd-month-yyyy”（例如，16 February 1995，带世纪）。

示例

```
CLEAR
SET CENTURY OFF
?DMY(DATE())
SET CENTURY ON
?DMY(DATE())
```

请参阅

MDY（）,SET CENTURY,SET DATE

DO 命令

执行一个 Visual FoxPro 程序或过程。

语法

```
DO ProgramName1|ProcedureName
  [IN ProgramName2]
  [WITH ParameterList]
```

参数描述

ProgramName1

指定要执行的程序的名称。

如果执行的程序不包含扩展名，Visual FoxPro 就以下列顺序查找并执行这些版本的程序：

- ..EXE（可执行版本）
- . APP（一个应用程序）
- .FXP（已编译的版本）
- .PRG（程序）

若要使用 DO 执行一个特定的菜单程序、表单程序或者查询，则必须包含它的扩展名（.MPR、.SPR、或者 .QPR）。

ProcedureName

指定要执行的过程的名称。Visual FoxPro 首先在当前执行的程序中查找此过程，如果在该程序中找不到此过程，Visual FoxPro 就在用 SET PROCEDURE 打开的过程文件中查找过程。.

可以包含 IN *ProgramName2* 子句，通知 Visual FoxPro 在指定的文件中查找过程。

在一个可执行文件 (.EXE) 或应用程序 (.APP) 中，多个过程可以有相同的过程名。当使用 DO 去启动一个可执行文件或一个应用程序的过程时，Visual FoxPro 只在可执行文件或应用程序的主程序中搜索指定的过程。

IN ProgramName2

执行 *ProgramName2* 指定的程序文件中的一个过程。

当找到该文件时，执行该过程。如果找不到该程序文件，就会显示“文件不存在”信息。如果找到了程序文件，但指定的过程不存在，就会显示“找不到过程”信息。

WITH ParameterList

指定要传递给程序或过程的参数。列在 *ParameterList* 中的参数可以是表达式、变量、字母和数字、字段或用户自定义函数。默认情况下，参数按引用传递给程序和过程，也可以将参数放在括号中按值传递。

关于按值或引用传递参数的内容，请参阅 SET UDFPARMS。传递给程序或过程的参数的最大数目为 27。有关参数传递的详细内容，请参阅 LPARAMETERS 和 PARAMETERS。

说明

DO 执行一个程序或一个过程文件中的 Visual FoxPro 程序或过程。一个程序文件自身又可以包含其他的 DO 命令，这种嵌套最多可允许 128 级。

当使用 DO 运行一个程序时，包含在程序文件中的命令一直执行，直到下列某一事件发生：

- 遇到 RETURN 语句。
- 执行了 CANCEL 命令。
- 执行了另一个 DO 命令。
- 到达文件末尾。
- 执行了 QUIT 命令。

当程序执行结束时，控制可以返回到：

- 调用的程序。
- 命令窗口。

- 操作系统。

  如果是从“程序”菜单中选择“执行”命令，并在非当前的某个目录或驱动器上执行一个程序，Visual FoxPro 就自动地将默认的目录和驱动器改为包含该程序的目录和驱动器。

  请参阅

  CLEAR 命令，LPARAMETERS 命令，PARAMETERS 命令，PARAMETERS（）函数，PRIVATE 命令，PROCEDURE 命令，PUBLIC 命令，SET DEFAULT 命令，SET DEVELOPMENT 命令，SET PATH 命令，SETPROCEDURE 命令

DO CASE...ENDCASE 命令

根据不同的条件表达式结果执行不同的命令。

语法

```
DO CASE
  CASE lExpression1
     Commands
  [CASE lExpression2
     Commands
  ...
```

CASE lExpressionN
Commands]
[OTHERWISE
Commands]
ENDCASE

参数描述

CASE lExpression1 Commands ...

当遇到第一个结果为“真”的 CASE 表达式时，就执行它后面的命令集合。命令集合连续执行，直到遇到下一个 CASE 或 ENDCASE。然后就从 ENDCASE 后面的第一个命令恢复程序的执行。

如果一个 CASE 表达式为“假”(.F.)，就忽略它与下一个 CASE 子句之间的命令集合。此命令只能执行一组命令，这些命令是 CASE 表达式计算为“真”(.T.)的第一个命令集合，而其他计算为“真”(.T.) 的 CASE 表达式被忽略。

OTHERWISE Commands

- 如果所有的 CASE 表达式计算为“假”(.F.)，就由 OTHERWISE 确定是否执行一组额外的命令。
- 如果包含 OTHERWISE，就执行 OTHERWISE 后面的命令集，执行后跳到 ENDCASE 后的第一条命令执行。
- 如果省略了 OTHERWISE，就跳到 ENDCASE 后面的第一条命令处执行。

说明

DO CASE 根据逻辑表达式的值执行一组 Visual FoxPro 命令。当执行 DO CASE 时，

先计算它后面的逻辑表达式，表达式的值决定了执行哪一组命令集。可在 DO CASE 和 END CASE 之后放置注释，在编译和执行过程中，这些注释被忽略。

示例

本示例中，Visual FoxPro 计算每个 CASE 子句，直到在列表中找到 MONTH 变量。将合适的字符串存储到变量 rpt_title 中，并退出 DO CASE 结构。

```
STORE CMONTH(DATE()) TO month && 今天所在的月份

DO CASE && 开始循环

   CASE INLIST(month,'January','February','March')
      STORE 'First Quarter Earnings' TO rpt_title

   CASE INLIST(month,'April','May','June')
      STORE 'Second Quarter Earnings' TO rpt_title

   CASE INLIST(month,'July','August','September')
      STORE 'Third Quarter Earnings' TO rpt_title

   OTHERWISE
      STORE 'Fourth Quarter Earnings' TO rpt_title
ENDCASE && 结束循环
WAIT WINDOW rpt_title NOWAIT
```

请参阅

DO WHILE...ENDDO 命令，EXIT 命令，FOR...ENDFOR 命令，IF...ENDIF 命令，IIF() 函数，SCAN...ENDSCAN 命令

DoCmd 方法

对于 Visual FoxPro 应用程序自动服务程序的一个实例，执行一个 Visual FoxPro 命令。

语法

ApplicationObject.DoCmd(cCommand)

参数描述

cCommand

指定要执行的 Visual FoxPro 命令。

说明

cCommand 必须是一个有效的 Visual FoxPro 命令。

应用于

Application 对象，_VFP 系统变量

请参阅

DO, Eval 方法, SetVar 方法

DODEFAULT( ) 函数

在子类（派生类）中，执行父类的同名的事件或方法。

语法

DODEFAULT([eParameter1 [, eParameter2] ...])

返值类型

字符型，数值型，货币型，日期型，日期时间型，逻辑型，备注型

参数描述

eParameter1 [, eParameter2] ...

传递给父类事件或方法的参数。

说明

在子类中，DOEFAULT( ) 执行父类的同名事件或方法。例如，如果在某子类的 Click 事件中包含 DOEFAULT( ) 函数，则执行父类的 Click 事件。DOEFAULT( ) 函数与 :: 限定符不同，DOFAULT( ) 执行父类的同名程序，:: 引用父类中不同名的程序。

DODEFAULT( ) 返回的值由父类中的事件或方法的返回值决定。

DODEFAULT( ) 只能在事件或方法中使用。

请参阅

DOEVENTS 命令

执行所有等待的 Windows **事件。**

语法

DOEVENTS

说明

如果您将 AutoYield **属性设置为假** (.F.)**，则在用户程序执行时，所有** Windows **事件将不被响应，而是排成一个队列，等待响应。直到执行** DOEVENTS **时，所有等待的** Windows **事件都被响应，执行用户为这些事件编写的代码。**

请参阅

AutoYield **属性**

DO FORM 命令

运行用表单设计器创建或编译过的表单或表单集。

语法

```
DO FORM FormName | ?
  [NAME VarName [LINKED]]
  [WITH cParameterList]
  [TO VarName]
  [NOREAD] [NOSHOW]
```

参数描述

FormName

运行用表单设计器创建或编译过的表单或表单集。

?

显示“执行”对话框，从对话框中选择要运行的表单或表单集。

NAME VarName [LINKED]

指定一个变量或数组元素，可通过它们引用表单或表单集。如果指定的变量不存在，Visual FoxPro 就自动创建它。如果指定一个数组元素，在执行 DO FORM 前数组必须存在。如果指定的变量或数组元素已经存在，就将改写它的内容。

如果省略 NAME 子句，Visual FoxPro 就创建一个与表单或表单集文件同名的对象类型的变量。

包含 LINKED 可用来链接表单和相关联的变量。当变量超出作用域时就释放表单。如果不包含 LINKED，即使没有与表单相关联的变量，表单仍可以是活动的。

WITH cParameterList

指定传递给表单或表单集的参数。

运行表单时，参数传递给表单的 Init 方法。

运行表单集时，如果表单集的 WindowType 属性设置为无模式 (0) 或模式 (1)，参数就传递给表单集的 Init 方法；如果表单集的 WindowType 属性设为读 (2) 或写 (3) 模式，参数就传递给 Setup 方法。

TO VarName

指定存放表单返回值的变量。如果变量不存在，Visual FoxPro 就自动创建它。可在表单的 Unload 事件过程中使用 RETURN 命令来指定返回值。如果不指定返回值，就返回默认值“真”(.T.)。要使用 TO 命令，表单的 WindowType 属性必须设置为 1(Modal)。

NOREAD

创建并显示表单集，但在执行 READ 前不激活控制。如果表单集对象的 WindowType 属性没有设为 2（读），就忽略 NOREAD。

NOSHOW

指定当表单正在运行时，不能调用表单的 Show 方法。当您包含了 NOSHOW 并运行表单时，直到表单的 Visible 属性设置为真(.T.) 或调用了

表单的 Show 方法，表单才可见。

说明

DO FORM 执行表单或表单集的 Show 方法。

示例

下面的示例运行了停止监视 (Swatch.scx) 的控件示例。

```
DO FORM (HOME(2) + 'Solution\Controls\Timer\Swatch.scx')
```

请参阅

COMPILE FORM, CREATE FORM, MODIFY FORM

DO WHILE...ENDDO 命令

在一个条件循环里执行一组命令。

语法

```
DO WHILE lExpression
  Commands
  [LOOP]
  [EXIT]
ENDDO
```

参数描述

lExpression

指定一个逻辑表达式，它的值决定是否执行 DO WHILE 和 ENDDO 之间的命令集。如果 *lExpression* 计算为“真”(.T.)，就执行命令集。

Commands

指定当 *lExpression* 计算为“真”(.T.) 时，要执行的 Visual FoxPro 命令集。

LOOP

直接将程序控制返回到 DO WHILE。LOOP 可放在 DO WHILE 和 ENDDO 间的任何位置。

EXIT

将程序控制从 DO WHILE 循环的内部转到 ENDDO 后的第一个命令。EXIT 可放在 DO WHILE 和 ENDDO 间的任何位置。

说明

只要逻辑表达式 *lExpression* 为“真”(.T.)，就执行位于 DO WHILE 和 ENDDO 间的命令集。每个 DO WHILE 语句必须有一个相应的 ENDDO 语句。可在 DO WHILE 和 ENDDO 之后放置注释。在程序编译和执行期间注释被忽略。

示例

下面的示例用 DO WHILE loop 语句对库存中价格超过 20 美元的产品进行汇总，直到遇到文件尾 (EOF)。然后退出 DO WHILE loop 语句，并显示总和。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE products  && 打开 Products 表
SET TALK OFF
```

```
gnStockTot = 0

DO WHILE .T. && 开始循环
    IF EOF()
        EXIT
    ENDIF
    IF unit_price < 20
        SKIP
        LOOP
    ENDIF
    gnStockTot = gnStockTot + in_stock
    SKIP

ENDDO   && 结束循环

CLEAR
? 'Total items in stock valued over 20 dollars:'
?? gnStockTot
```

请参阅

DO CASE...ENDCASE, FOR...ENDFOR, IF...ENDIF, IIF(), SCAN...ENDSCAN

Dock 方法

沿着 Visual FoxPro 主窗口的边界停放“工具栏”对象。

语法

ToolBar.Dock(nLocation [, X, Y])

参数描述

nLocation

指定工具栏停放的位置。*nLocation* 的值有：

值	常量	说明
	TOOL_NOTDOCKED	不停放工具栏。
	TOOL_TOP	在 Visual FoxPro 主窗口的顶部停放工具栏。
	TOOL_LEFT	在 Visual FoxPro 主窗口的左边停放工具栏。
	TOOL_RIGHT	在 Visual FoxPro 主窗口的右边停放工具栏。
	TOOL_BOTTOM	在 Visual FoxPro 主窗口的底部停放工具栏。

X, Y

指定工具栏停放位置的水平坐标和垂直坐标。

说明

当停放一个工具栏时将隐藏它的标题栏，它的边框变成单线边框。工具栏的大小调整为一个单行的按钮，窗口的大小也自动调整，使工具栏不掩盖屏幕上任何信息。例如，如果工具栏停放在上边，屏幕就根据工具栏的高度下移。

使用 Move 方法可移去工具栏。当使用 Move 方法时，指定的坐标必须在停放区的外部。

应用于

工具栏

请参阅

AfterDock 事件, BeforeDock 事件, Docked 属性, Undock 事件

Docked 属性

包含一个逻辑值，表明是否停放用户自定义的工具栏对象。设计和运行时可用。

语法

ToolBar.Docked[= lExpr]

参数描述

lExpr

下表列出了 Docked 属性的设置：

设置	说明
“真” (.T.)	停放工具栏。
“假” (.F.)	不停放工具栏。

应用于

工具栏

请参阅

DockPosition 属性

DockPosition 属性

指定用户自定义工具栏对象停放的位置。设计和运行时只读。

语法

ToolBar.DockPosition[= nPosition]

参数描述

nPosition

下表列出了 *nPosition* 的值：

设置	位置
	不停放
	顶部
	左边
	右边
	底部

应用于

工具栏

请参阅

Docked 属性

ocumentFile 属性

返回文件名，由该文件创建一个嵌入或链接的对象。设计时，指定链接文件名，对于一个已有的对象，运行和设计时只读，但在对象生成时可以设置。

语法

Object.DocumentFile[= cFileName]

参数描述

cFileName

指定文件名，由该文件创建一个嵌入或链接的对象。该文件名包括该文件的完整路径。

说明

DocumentFile 包含了嵌入（不是链接）对象的空字符串。

最初向表单上添加 OLE 容器时，通过“插入对象”对话框为链接的 OLE 对象设置 DocumentFile 属性。也可以在 APPEND GENERAL 命令创建 OLE 对象或以代码作为类定义的一部分来定义对象时，设置该属性。

在用 DocumentFile 属性指定 OLE 对象的内容前，通过设置对象的 OLEClass 属性来指定 Automation 服务应用程序。

示例

下面的示例向一个表单中添加一个 OLE 容器控件，并使用 DocumentFile 和 OleClass 属性来指定一个 Excel 工作表作为要编辑的文件，指定 Excel 作为 OLE 服务器。DocumentFile 属性在 EXCEL 目录中指定一个名为 BOOK1.XLS 的工作表。如果在 DocumentFile 属性中指定的文件和目录不存在，这个示例将不能正确地执行；DoVerb 方法用于激活要编辑的工作表。

```
frmMyForm = CREATEOBJECT('form') && 创建一个表单
frmMyForm.Closable = .F. && “控件”菜单框无效

frmMyForm.AddObject('cmdCommand1','cmdMyCmdBtn') && 添加命令按钮
frmMyForm.AddObject("oleObject","oleExcelObject") && 添加 OLE 对象

frmMyForm.cmdCommand1.Visible=.T. && 显示“退出”命令按钮
```

```
frmMyForm.oleObject.Visible=.T. &&显示 OLE 控件
frmMyForm.oleObject.Height = 50 && OLE 控件高度

frmMyForm.Show &&显示表单

frmMyForm.oleObject.DoVerb(-1) && -1用于编辑

READ EVENTS &&启动事件处理

DEFINE CLASS oleExcelObject as OLEControl
    OleClass ="Excel.Sheet" &&服务器名
    DocumentFile = "C:\EXCEL\BOOK1.XLS" &&此文件必须存在
ENDDEFINE

DEFINE CLASS cmdMyCmdBtn AS CommandButton &&创建命令按钮
    Caption = '\<Quit' &&命令按钮的标签
    Cancel = .T. &&默认“取消”按钮 (Esc)
    Left = 125 &&命令按钮列
    Top = 210 &&命令按钮行
    Height = 25 &&命令按钮高度
    PROCEDURE Click
        CLEAR EVENT &&  停止事件处理，关闭表单
ENDDEFINE
```

应用于

OLE 绑定型控件，OLE 容器控件

请参阅

APPEND GENERAL, CREATEOBJECT（）, OLEClass

_DOS 系统变量

如果使用 FoxPro for MS-DOS，该命令为“真”(.T.)。如果使用一个不同平台的 FoxPro 或 Visual FoxPro 版本，该变量为“假”(.F.)。

语法

_DOS = lExpression

说明

不能用 STORE 或“=”更改 _DOS 中的值。

请参阅

_MAC, _UNIX, VERSION（）, _WINDOWS

DoScroll 方法

滚动表格控件来模拟用户单击滚动条操作。

语法

Grid.DoScroll(nDirection)

参数描述

nDirection

下表列出了 *nDirection* 的设置：

滚动命令	滚动动作
0	向上滚动
1	向下滚动
2	向上翻页滚动
3	向下翻页滚动
4	向左滚动
5	向右滚动
6	向左翻页滚动
7	向右翻页滚动

说明

使用 DoScroll 方法后触发 Scrolled 事件。

应用于

表格

请参阅

Scrolled 事件

DoVerb 方法

在指定的对象上执行一个动作。

语法

Object.DoVerb[(Verb)]

参数描述

Verb

指定在 OLE 容器控件中对象要执行的动作。如果没有指定，就执行默认的动作。此参数值可以是所有对象都支持的某个标准动作，或是 ObjectVerbs 属性数组的一个索引。每个对象可支持它自身的动作集。

下列值表示每个对象支持的标准的动作：

值	动作
	对象的默认动作。
	激活要编辑的对象。如果创建对象的应用程序支持现场激活，就在 OLE 容器控件中激活对象。
	在一个独立的应用程序窗口中打开对象。如果创建对象的应用程序支持现场激活，就在对象自身的窗口中激活对象。
	对于嵌入的对象，隐藏创建对象的应用程序。

续表

如果对象支持现场激活，就以现场激活方式激活对象并显示任何用户界面工具。如果对象不支持现场激活，对象就不激活并产生错误。
如果用户将焦点移到 OLE 容器控件中，就创建一个窗口，并准备要编辑的对象。如果对象不支持以鼠标单击的方式激活，就产生错误。
当激活要编辑的对象时，用来放弃所有记录的更改，对象的应用程序可以撤消这些更改。

说明

如果将 AutoActivate 属性设置为 2（双击），那么，当用户双击控件时，OLE 容器控件就自动激活当前的对象。

提示 尽管可以使用动作名（编辑、打开、播放等）来指定要和 Doverb 一起使用的动作，但使用索引（0，1，2 等）会更快。

示例

The 下面的示例向一个表单中添加一个 OLE 容器控件，并使用 OleClass 和 DocumentFile 属性指定 Excel 作为 OLE 服务器，指定一个 Excel 工作表作为要编辑的文件。

The DocumentFile 属性在驱动器 C 上的 EXCEL 目录中指定一个名为 BOOK1.XLS 的工作表。如果 DocumentFile 指定的文件和目录不存在，此示例将不能正确地执行，可能有必要更新 DocumentFile 属性来指定一个存在的目录和工作表文件。

使用 DoVerb 方法激活要编辑的工作表。

```
*frmMyForm = CREATEOBJECT('form') && 创建一个表单
*frmMyForm.Closable = .F. && 使“控件”菜单框无效
```

```
frmMyForm.AddObject('cmdCommand1','cmdMyCmdBtn') && 添加命令按钮
frmMyForm.AddObject("oleObject","oleExcelObject") && 添加 OLE 对象

frmMyForm.cmdCommand1.Visible=.T. && 显示“退出”命令按钮

frmMyForm.oleObject.Visible=.T. && 显示 OLE 控件
frmMyForm.oleObject.Height = 50 && OLE 控件的高度

frmMyForm.Show && 显示表单

frmMyForm.oleObject.DoVerb(-1) && -1 用于编辑

READ EVENTS && 启动事件处理

DEFINE CLASS oleExcelObject as OLEControl
    OleClass ="Excel.Sheet" &&服务器名
    DocumentFile = "C:\EXCEL\BOOK1.XLS" && 此文件必须存在
ENDDEFINE

DEFINE CLASS cmdMyCmdBtn AS CommandButton && 创建命令按钮
    Caption = '\<Quit' && 命令按钮的标题
    Cancel = .T. && 默认的“取消”命令按钮 (Esc)
    Left = 125 && 命令按钮列
    Top = 210 && 命令按钮行
    Height = 25 && 命令按钮的高度
    PROCEDURE Click
        CLEAR EVENTS && 停止事件处理，关闭表单
ENDDEFINE
```

应用于

OLE 绑定型控件，OLE 容器控件

请参阅

OLE 绑定型控件，OLE 容器控件

DOW（） 函数

从日期表达式或日期时间表达式返回一个数值型的星期几。

语法

DOW(dExpression | tExpression [, nFirstDayOfWeek])

返值类型

数值型

参数描述

dExpression

指定的日期表达式。

tExpression

指定的日期时间表达式。

nFirstDayOfWeek

指定一周的第一天。 *nFirstDayOfWeek* **可以是下列某个值：**

一周的第几天	说明
0	DOW（）使用当前在“星期开始于”列表框中选定的日期，该列表框出现在“选项”对话框中的“国际”选项卡上。
1	星期日。当省略 *nFirstDayOfWeek* 时这是默认值。它是早期 FoxPro 版本使用的一周的第一天。
2	星期一
3	星期二
4	星期三
5	星期四
6	星期五
7	星期六

示例

```
STORE DATE() TO gdDayNum
CLEAR
? DOW(gdDayNum)
? CDOW(gdDayNum)
```

请参阅

CDOW（）,DAY（）,SET FDOW,SET FWEEK,SYS（） Functions Overview,WEEK（）

DownClick 事件

当单击控件的下箭头时，此事件发生。

语法

PROCEDURE Control.DownClick
[LPARAMETERS nIndex]

参数描述

nIndex
在控件数组中唯一标识一个控件。

应用于

组合框，微调

请参阅

UpClick 事件

DownPicture 属性

指定选择控件时显示的图形。设计和运行时可用。

语法

Control.DownPicture[= cPicture]

参数描述

cPicture

指定位图的完整路径和文件名或数据库表中的通用字段名。

说明

若在设计时设置 DownPicture 属性，而指定的文件不存在，则 Visual FoxPro 显示错误信息，但是该属性仍保留。若在运行时，DownPicture 属性设置给不存在的文件，则 Visual FoxPro 忽略此属性。

若不指定 DownPicture 属性的设置，则当选择控件时 Visual FoxPro 使用 Picture 属性指定的图形。

对于 CommandButton 控件，Style 属性必须设置为 0（标准），以使位图在控件上显示；对 CheckBox 和 OptionButton 控件，Style 属性必须设置为 1（图形）。

应用于

复选框，命令按钮，选项按钮

请参阅

DisabledPicture 属性, Enabled 属性, Picture 属性, Style 属性

Drag 方法

启动、结束或取消拖动操作。

语法

Control.Drag [(nAction)]

参数描述

nAction

指示要执行的动作。若省略 *nAction*，**则** *nAction* **设置为** 1。**下表列出了** Drag **的设置：**

设置	说明
0	取消拖动操作；恢复控件的原始位置。
1	（默认值）启动对控件的拖动操作。
2	结束拖动—即放下控件。

说明

通常， MouseDown **事件过程调用** Drag **方法来启动拖动操作。**

只有当控件的 DragMode 属性设置为 0（人工）时，才需要使用 Drag 方法来控件拖动。在 DragMode 属性设置为 1（自动）的控件上也可以使用 Drag，但并不触发 MouseDown 和 MouseUp 事件。

若想在拖动控件时更改鼠标指针，可使用 DragIcon 属性或 MousePointer 属性定义指针。只有在未设置 DragIcon 属性时，MousePointer 属性才有效。.

应用于

复选框，组合框，命令按钮，命令组，容器对象，控件对象，编辑框，表格，图像，标签，线条，列表框，OLE 绑定型控件，OLE 容器控件，选项按钮，选项组，页面，页框，形状，微调，文本框

请参阅

DragDrop 事件，DragIcon 属性，DragMode 属性，MousePointer 属性，Move 方法

DragDrop 事件

当完成拖放操作时发生。

语法

```
PROCEDURE Object.DragDrop
LPARAMETERS [nIndex,] oSource, nXCoord, nYCoord
```

参数描述

在事件处理中必须包含一个 LPARAMETERS 或 PAPAMETERS 语句，并且为每一个参数指定一个名称。Visual FoxPro 按下列顺序把三个或四个参数传递给 DragDrop 事件：

nIndex

唯一标识控件数组中的控件。

oSource

引用被拖动的控件。可用此参数引用控件的属性和方法。

nXCoord,nYCoord

包含鼠标指针在目标表单或控件中的当前水平（*nXCoord*）和垂直（*nYCoord*）坐标。这些坐标通常使用目标坐标系来表达，度量单位由 ScaleMode 属性确定。

说明

如果将控件拖动到另一个控件或表单上后释放鼠标按钮，或者调用了 Drag 方法并对 *nAction* 参数设置为 2（放下），拖放操作就算完成。

使用 DragDrop 事件可以控件拖动操作完成后所发生的事情。例如，可将源控件移动到新位置，或把文件从一个位置复制到另一个位置。

注意 DragDrop 事件涉及两个对象，即被拖动的控件和目标对象。DragDrop 事件是被目标对象触发，而不是由被拖动的控件触发。

可使用 DragMode 属性和 Drag 方法指定如何开始拖动。一旦开始拖动，则可用 DragOver 事件处理在 DragDrop 事件之前的事件。

应用于

复选框，组合框，命令按钮，命令组，容器对象，控件对象，编辑框，表单，表格，图像，标签，线条，列表框，OLE 绑定型控件，OLE 容器控件，选项按钮，选项组，页面，页框，形状，微调，文本框，工具栏

请参阅

Drag 方法，DragIcon 属性，DragMode 属性，DragOver 事件，MouseDown 事件，MouseUp 事件，MouseMove 事件

DragIcon 属性

指定在拖放操作时作为指针显示的图标。设计和运行时可用。

语法

Control.DragIcon[= cIcon]

参数描述

cIcon

指定用作鼠标指针的图标文件路径。一般来说，在设计时可用属性窗口指定图标文件。指定的文件必须有 .CUR 扩展名，并且按 VGA-Mono 2-Color 32 × 32 格式保存。若想使用彩色的 .CUR 文件，则产生错误信息。Visual FoxPro 专业版中包含的 ImageEdit（图像编辑），可用来创建一个两色的 CUR 文件。若省略 *cIcon*，则使用的指针为矩形内带一个箭头。

说明

设计和运行时可用。若想在拖动操作期间提供可见的反馈信息，比如指示源控件已在合适的目标之上，DragIcon 很有用处。当用户开始拖动时 DragIcon 有效。一般来说，可在 MouseDown 或 DragOver 事件中设置 DragIcon。

若在设计时设置了 DragIcon 属性，但指定的文件并不存在，则 Visual FoxPro 产生错误信息，但是属性仍保留。若运行时设置的文件不存在，则 Visual FoxPro 忽略 DragIcon 属性。

应用于

复选框，组合框，命令按钮，命令组，容器对象，控件对象，编辑框，表格，图像，标签，线条，列表框，OLE 绑定型控件，OLE 容器控件，选项按钮，选项组，页面，形状，微调，文本框

请参阅

Drag 方法，DragDrop 事件，DragMode 属性，MouseDown 事件，MouseMove 事件，MousePointer 属性，MouseUp 事件

DragMode 属性

指定拖放操作的拖动方式为人工或自动。设计和运行时可用。

语法

Control.DragMode[= nMode]

参数描述

nMode

下表列出了 DragMode **属性的设置：**

设置	说明
0	（默认值）人工。要求使用 Drag 方法开始源控件的拖动。
1	自动。单击源控件则自动开始拖动。

说明

当 DragMode 设置为 0（人工）时，控件响应鼠标事件，并且必须使用 Drag 方法开始拖动操作。

当 DragMode 设置为 1（自动）时，控件不响应鼠标事件，并且当用户在控件上按下鼠标主（左）键时，自动开始拖动操作。

在拖动操作中，当鼠标指针在目标控件或表单上时，释放鼠标按钮会生成目标对象的 DragDrop 事件，并且结束拖动操作。拖动也可触发 DragOver 事件。

注意 在拖动控件时，控件不会接受其他用户生成的鼠标或键盘事件（KeyPress、MouseDown、MouseMove 或 MouseUp）。

应用于

复选框，组合框，命令按钮，命令组，容器对象，控件对象，编辑框，表格，图像，标签，线条，列表框，OLE 绑定型控件，OLE 容器控件，选项按钮，选项组，页面，页框，形状，微调，文本框

请参阅

DragDrop 事件, DragIcon 属性, DragOver 事件, KeyPress 事件, MouseDown 事件, MouseMove 事件, MouseUp 事件, OLEDragMode 属性

DragOver 事件

控件拖过目标对象时发生此事件。

语法

PROCEDURE Object.DragOver
LPARAMETERS [nIndex,] oSource, nXCoord, nYCoord, nState

参数描述

在事件过程中必须包含一个 LPARAMETERS 或 PARAMETERS 语句，并且为每一个参数指定名称，否则产生错误。Visual FoxPro 按下列顺序把四或五个参数传递给 DragOver 事件：

nIndex

唯一标识控件数组中的控件。

oSource

包含对被拖动控件的引用。可用此参数引用控件的属性和方法。

nXCoord, nYCoord

包含鼠标指针在目标表单或控件内的当前水平 (*nXCoord*) 和垂直 (*nYCoord*)

位置。这些坐标通常使用目标的坐标系来表达，度量单位由 ScaleMode 属性确定。

nState

包含一个数值，表示被拖动控件相对于目标对象的移动状态。

设置	说明
0	进入。控件拖入了目标范围之内。
1	离开。控件拖出了目标范围之外。
2	经过。控件从目标上的一个位置移动到另一个位置。

用 *nState* 可以确定关键移动点处的动作。例如，可在 *nState* = 0（进入）时突出显示可能的目标，并在 *nState* = 1（离开）时恢复对象外观。

当对象在 *nState* = 0（进入）的情况下接受 DragOver 事件时：

- 如果源控件在目标对象上放下，则触发 DragDrop 事件。
- 如果源控件不在有效目标上放下，则用 *nState*=1（离开）触发一个 DragOver 事件。

说明

在拖动图标下方的对象是目标对象，它响应 DragOver 事件。当鼠标指针进入、离开或经过目标对象时可使用这个事件进行监控。

DragOver 事件确定在拖动开始之后和控件放到目标上之前所发生的一切操作。例如，通过设置 BackColor 或 ForeColor 属性，或显示一个专门的鼠标指针来突出显示目标，可以检验有效的目标范围。

应用于

复选框，组合框，命令按钮，命令组，控件对象，编辑框，表单，表格，图像，标签，线条，列表框，OLE 绑定型控件，OLE 容器控件，选项按钮，选项组，页面，页框，形状，微调，文本框，工具栏

请参阅

Drag 方法, DragDrop 事件, DragIcon 属性, DragMode 属性, MouseDown 事件, MouseUp 事件, MouseMove 事件

Draw 方法

重画表单对象。

语法

Object.Draw

应用于

表单，_SCREEN

请参阅

Paint 事件

DrawMode 属性

与颜色属性一起确定如何在屏幕上显示形状或线条控件。

语法

Object.DrawMode[= nMode]

参数描述

nMode

下表列出了 DrawMode 属性的设置：

设置	说明
1	Blackness 笔。用黑色画形状。
2	NotMerge 笔。与设置 15 相反。
3	Mask Not 笔。背景色和反前景色中共有颜色的合成。
4	Not Copy 笔。与设置 13 相反。
5	Mask Pen Not。前景色和反背景色中共有颜色的合成。
6	相反。反背景色。
7	XOR 笔。前景色或背景色中颜色的合成。
8	NotMask 笔。与设置相反。
9	Mask 笔。前景色和背景色共有颜色的合成
10	Not XOR 笔。与设置 7 相反。

续表

11	NOP。输出保持不变。实际上，此设置关闭了绘图。
12	Merge Not 笔。背景色和反前景色的合成。
13	（默认值）Copy 笔。由 ForeColor 属性指定颜色。
14	Merge Pen Not。前景色和反背景色的合成。
15	Merge 笔。前景色和背景色的合成。
16	Whiteness 笔。用白色画形状。

说明

使用 DrawMode 属性，可在使用形状和线条控件或用图形方法画图时增强视觉效果。当画新形状时，Visual FoxPro 比较图案中的每个像素和已有背景中的相应像素，然后进行逐位操作。例如，设置 7 用异或操作 (XOR) 组合绘图图案的像素和背景像素。DrawMode 属性设置的效果取决于运行时画线的颜色和屏幕颜色如何结合。设置 1、6、7、11、13、16 可以产生可预测的结果。

应用于

表单，线条，_SCREEN，形状

请参阅

Box 方法，Circle 方法，DrawWidth 属性，DrawStyle 属性，FillColor 属性，FillStyle 属性，Line 方法，Pset 方法

DrawStyle 属性

指定用图形方法绘图时的线条样式。设计和运行时可用。

语法

Object.DrawStyle[= nStyle]

参数描述

nStyle

下表列出了 DrowStyle **属性的设置：**

设置	说明
0	（默认值）实线
1	虚线
2	点线
3	点划线
4	双点划线
5	透明
6	内实线

注意 若 DrawWidth 设置为 1，对每个设置 DrawStyle 可产生上表所述的效果。但是若 DrawWidth 设置为大于 1 的值，则设置 1 到 4 都产生一条实线。

应用于

表单, _SCREEN

请参阅

Circle 方法, DrawMode 属性, DrawWidth 属性, FillColor 属性, FillStyle 属性, ForeColor 属性, Line 方法, Pset 方法

DrawWidth 属性

指定图形方法输出时所用的线条宽度。设计和运行时可用。

语法

Object.DrawWidth[= nSize]

参数描述

nSize

指定线宽的像素数。DrawWidth 有效设置值从 1 到 32,767。默认值是 1 个像素宽。

增加此属性的值会增加线条宽度。

应用于

表单, _SCREEN

请参阅

Circle 方法, DrawMode 属性, DrawStyle 属性, FillColor 属性, FillStyle 属性, ForeColor 属性

DRIVETYPE（） 函数

返回指定驱动器的类型。

语法

DRIVETYPE(*cDrive*)

返值类型

数值型

参数描述

cDrive

cDrive 是驱动器指示符。驱动器名中的分号是可选的（例如， C ）。

说明

下表解释了 DRIVETYPE（） 的返回值，以及相应的驱动器类型说明。

数值	驱动器类型
1	无类型
2	软驱

续表

3	硬盘
4	可移动硬盘或网络驱动器
5	CD-ROM
6	RAM 磁盘*

*由于有多种 RAM 磁盘，您可能得到不一致的返回结果。

请参阅

SYS(5)

DROP TABLE 命令

把一个表从数据库中移出，并从磁盘中删除它。

语法

DROP TABLE TableName | FileName | ? [RECYCLE]

参数描述

TableName

指定要移出并删除的表的名称。

FileName

指定一个自由表，从磁盘中删除它。

?

显示“移去”对话框，您可以从当前数据库中移出并删除您在对话框中选定的表。

RECYCLE

将删除的文件放到 Windows 95 回收站中，以后可以恢复。

说明

执行了 DROP TABLE 之后，所有与被删除表有关的主索引，默认值，验证规则都将丢失。当前数据库中的其他表若与被删除的表有关联，比如规则引用了被删除的表或与被删除的表建立了关系，这些规则和关系也都将无效。

表被删除后，Visual FoxPro 将无法访问它，而且即使 SET SAFETY 设置为 ON，在删除表时也不会提示警告信息。

请参阅

ADD TABLE 命令, CLOSE DATABASES 命令, CREATE DATABASE 命令, FREE TABLE 命令, OPEN DATABASE 命令

DROP VIEW 命令

从当前数据库中删除指定的 SQL 视图。

语法

DROP VIEW ViewName

参数描述

ViewName

指定要删除的视图。

说明

DROP VIEW **的作用与** DELETE VIEW **相同。但** DROP VIEW **是** ANSI SQL **标准语法。使用** CREATE SQL VIEW **创建** SQL **视图，并把视图加入到数据库中。如果** SQL **视图在打开时被删除，则包含** SQL **查询结果的临时表并不关闭。**

请参阅

CREATE SQL VIEW, DELETE VIEW, OPEN DATABASE

DropDown **事件**

单击组合框控件中的下箭头后、列表部分即将下拉时此事件发生。

语法

PROCEDURE ComboBox.DropDown
[LPARAMETERS nIndex]

参数描述

必须在事件过程中包含 LPARAMETERS 或 PARAMETERS 语句，并且为每个参数指定名称，否则产生错误。Visual FoxPro 将下列参数传递给 DropDown 事件：

nIndex

包含一个数值，唯一标识控件数组中的控件。

说明

使用 DropDown 事件过程，可在用户做选择之前，对组合框的列表部分进行最后的更新，从而可以用 AddItem 或 RemoveItem 方法从列表中添加或删除菜单项。在运行时需要对此控件进行交互操作时，这种灵活性很有用处。例如，可以让在组合框列表部分中显示的内容取决于用户在 OptionGroup 中的选择。

应用于

组合框

请参阅

AddItem 方法, DownClick 事件, RemoveItem 方法

DROPOFFLINE( ) 函数

放弃对游离视图的所有修改，并把游离视图放回到数据库中。

语法

DROPOFFLINE(cViewName)

返值类型

逻辑值

参数描述

cViewName

指定视图名称。视图在数据库中的名称，且数据库应处于打开状态。

说明

若游离视图被成功地放回到数据库中，则 DROPOFFLINE() 返回真 (.T.)；否则返回假 (.F.)。

请参阅

CREATEOFFLINE(), DROP VIEW, USE

DTOC() 函数

由日期表达式返回字符型日期。

语法

DTOC(dExpression | tExpression [, 1])

返值类型

字符型

参数描述

dExpression
指定的日期型变量、数组元素或字段，DTOC（）由此返回字符型日期。

tExpression
指定的日期时间型变量、数组元素或字段，DTOC（）由此返回字符型日期。

1
以适合作为索引的格式返回日期。对于按时间顺序维护表的记录特别有用。例如，要按录入顺序排列表中记录，可执行命令：

```
INDEX ON DTOC(gdInvDate, 1) + gnInvTime TAG Timeindx
```

当记录中输入数据时，gdInvDate 和 gnInvTime都是包含着日期和时间的数据。

说明

DTOC（）返回对应于日期或日期时间表达式的字符串。日期格式由 SET CENTURY 和 SET DATE 确定。.

示例

```
SET STRICTDATE TO 1
STORE CTOD({^1998-10-31}) TO gdThisDate
CLEAR
? DTOC(gdThisDate)
STORE DTOC({^1998-10-31}+90) TO gcExpireDate
? 'Your 90-day warranty expires ', gcExpireDate
? DTOC({^1998-10-31},1)
```

请参阅

CTOD（）,SET CENTURY,SET DATE,SYS（） Functions Overview,TTOC（）

DTOR（） 函数

将度转换为弧度。

语法

DTOR(nExpression)

返值类型

数值型

参数描述

nExpression

指定的数值表达式，将其值转换为弧度。一个用“度：分：秒”格式表示的度数应先转换为实数形式。

说明

DTOR（）把以度数表示的数值表达式转换为弧度值。DTOR（）在使用 Visual FoxPro 的三角函数 ACOS（）、ASIN（）、COS（）、SIN（）、TAN（）时很有用。用 RTOD（）可将弧度转换为度。

示例

```
CLEAR
?DTOR(0) &&显示数值 0.00
?DTOR(45) &&显示数值 0.79
?DTOR(90) &&显示数值 1.57
?DTOR(180) &&显示数值 3.14
?COS(DTOR(90)) &&显示数值 0.00
```

请参阅

ACOS（）,ASIN（）,COS（）,RTOD（）,SIN（）,TAN（）

DTOS（） 函数

从指定日期表达式中返回 yyyymmdd **格式的字符串日期。**

语法

DTOS(dExpression | tExpression)

返值类型

字符型

参数描述

dExpression

指定的日期表达式， DTOS（） **将其转换为八位字符串**

tExpression
指定的日期时间表达式，DTOS（） 将其转换为八位字符串。

说明

若要按日期或日期时间型字段对表索引，此函数很有用。它与包含参数 1 的 DTOC（） 相同。用 DTOS() 返回的字符串不受 SET DATE 或 SET CENTURY 的影响。

示例

```
CLEAR
?DTOS(DATE（）)
```

请参阅

DATE（）,DATETIME（）,CTOD（）,DTOC（）

DTOT（） 函数

从日期型表达式返回日期时间型的值。

语法

DTOT(dDateExpression)

返值类型

日期时间型

参数描述

dDateExpression
指定要返回日期时间型值的日期型表达式。

说明

DTOT() 返回的日期时间型值的格式由 SET DATE 和 SET CENTURY 当前设置值决定。若未提供世纪值，则假定为二十世纪。

DTOT() 向日期中加上午夜 (12:00:00 A.M.) 的默认时间来生成有效的日期时间值。

示例

```
? DTOT({^1998-02-16})     && 显示数值 02/16/1998 12:00:00am
```

请参阅

CTOT(), DATE(), SEC(), SECONDS(), SET SECONDS, TIME()

DynamicAlignment 属性

指定列对象中文本和控件的对齐方式。运行期间每次刷新表格控件时都重新计算对齐方式。设计时可用，运行时只读写。

语法

Column.DynamicAlignment[= cAlign]

参数描述

cAlign

指定运行时重新计算的字符串表达式，其结果可以是下列某一值：

设置	说明
0	左
1	右
2	居中
3	（默认值）自动
4	左上。文本在左侧、列的顶部对齐。
5	右上。文本在右侧、列的顶部对齐。
6	顶部居中。文本在列的顶部、左右距离相等的中央对齐。
7	左下。文本在左侧、列的底部对齐。
8	右下。文本在右侧、列的底部对齐。
9	底部居中。文本在列的底部、左右距离相等的中央对齐。

应用于

列

请参阅

Alignment 属性

DynamicBackColor, DynamicForeColor 属性

指定列对象的背景和前景色。运行期间每次刷新表格控件时，都重新计算颜色值。

语法

Column.DynamicBackColor[= cExpression]
Column.DynamicForeColor[= cExpression]

参数描述

cExpression

用引号指定表达式。运行期间每次刷新表格控件时都重新求值。运行时的计算结果必须是单个颜色值。

说明

可使用 DynamicBackColor 和 DynamicForeColor 属性创建特殊效果。例如，用绿色显示奇数行，而用灰色显示偶数行。

示例

下面的示例使用 DynamicBackColor 属性和 SetAll 方法指定表格控件中记录的背景颜色。若显示在表格中的记录号为偶数，则该记录的 DyamicBackColor 是白色，否则 DynamicBackColor 是绿色。

在表单上放置一个表格控件后，打开 customer 表并且在表格上显示该表的内容。Caption 属性用来为 CUST_ID 字段指定不同的标头标题。在表单上放置一个命令按钮

```
    来关闭表单。
CLOSE ALL && 关闭表和数据库
OPEN DATABASE (HOME(2) + 'Data\testdata')

USE customer IN 0 &&打开 Customer 表

frmMyForm = CREATEOBJECT('Form') && 创建表单
frmMyForm.Closable = .f. && 使控件菜单框无效

frmMyForm.AddObject('cmdCommand1','cmdMyCmdBtn') && 添加命令按钮
frmMyForm.AddObject('grdGrid1','Grid') && 添加表格控件

frmMyForm.grdGrid1.Left = 25 && 调整表格位置

frmMyForm.grdGrid1.SetAll("DynamicBackColor",;
    "IIF(MOD(RECNO(),2)=0,RGB(255,255,255);
    ,RGB(0,255,0))","Column") && 交替白色和绿色记录

frmMyForm.grdGrid1.Visible = .T. && 表格控件可见
frmMyForm.cmdCommand1.Visible =.T. && “退出”命令按钮可见
frmMyForm.grdGrid1.Column1.Header1.Caption = 'Customer ID'

frmMyForm.SHOW && 显示表单
READ EVENTS && 启动事件处理

DEFINE CLASS cmdMyCmdBtn AS CommandButton && 创建命令按钮
    Caption = '\<Quit' && 命令按钮上的标题
    Cancel = .T. && 默认取消命令按钮(Esc)
    Left = 125 && 命令按钮的列
    Top = 210 && 命令按钮的行
    Height = 25 && 命令按钮的高度
```

```
    PROCEDURE Click
       CLEAR EVENTS &&结束事件处理，关闭表单
       CLOSE ALL &&关闭表和数据库
ENDDEFINE
```

应用于

列

请参阅

BackColor_ForeColor 属性

DynamicCurrentControl 属性

指定用包含在列对象中的哪个控件来显示活动单元的值。每次刷新表格控件时都重新计算控件名称。设计时可用，运行时只读写。

语法

Column.DynamicCurrentControl[= cExpression]

参数描述

cExpression

在运行时计算得到控件的名称。该控件显示和接收列中活动单元的数据。

说明

默认控件是 Name 属性为“Textl”的文本框。可使用 AddObject 方法添加其他控件。若列的 Sparse 属性设置为“真”(.T.)，则只有列中的活动单元才使用 DynamicCurrentControl 属性设置中指定的对象来显示数据，其他单元用文本框显示数据。若 Sparse 属性设置为“假”(.F.)，则列中的所有单元都使用 DynamicCurrentControl 属性设置中指定的对象来显示数据。

应用于

列

请参阅

AddObject 方法, CurrentControl 属性

DynamicFontBold,DynamicFontItalic,

DynamicFontStrikethru, DynamicFontUnderline 属性

指定显示在列对象中的文本具有下列一种或多种字形：粗体，*斜体*，~~删除线~~或下划线。每次刷新表格控件时，都要重新计算此逻辑表达式。设计时可用，运行时只读写。

语法

Column.DynamicFontBold[= “lExpr”]
Column.DynamicFontItalic[= “lExpr”]
Column.DynamicFontStrikeThru[= “lExpr”]
Column.DynamicFontUnderline[= “lExpr”]

参数描述

“lExpr”
值可以是“真”(.T.)或“假”(.F.)。下表列出了动态字体属性的设置：

设置	说明
True (.T.)	字形是粗体、斜体、删除线或下划线。
False (.F.)	（默认值，FontBold 除外）字形不是粗体、斜体、删除线，也不是下划线。

说明

可在运行时根据数据的情况使用不同的属性。例如，可设置 DynamicFontUnderline 属性来指示组织中的新成员姓名。

应用于

列

请参阅

DynamicFontSize 属性

DynamicFontName 属性

指定列对象中显示文本所用的字体。每次刷新表格控件时，都将重新计算。设计时可用，运行时只读写。

语法

Column.DynamicFontName[= cName]

参数描述

cName

计算得到的字体名称。

说明

在设计期，如果在属性窗口中选择 FontName 属性，并单击属性框右边的下箭头，就可以显示一个可用字体的列表。

应用于

列

请参阅

FontName 属性

DynamicFontOutline 属性

指定与列对象相关的文本是否以轮廓方式显示。运行期间每次刷新表格控件时，都要重新计算此逻辑表达式。

语法

Column.DynamicFontOutline[= “lExpr”]

参数描述

“lExpr”

结果为“真”(.T.)**或“假”**(.F.)。**下表列出了** DynamicFontOutline **属性的设置：**

设置	说明
True (.T.)	**文本以轮廓方式显示。**
False (.F.)	**（默认值）文本不以轮廓方式显示。**

应用于

列

请参阅

FontOutline **属性**, DynamicFontShadow **属性**

DynamicFontShadow 属性

指定与列对象相关的文本是否带阴影。运行时刷新表格控件，都要重新计算此逻辑表达式。

语法

Column.DynamicFontShadow[= “lExpr”]

参数描述

“lExpr”

结果为“真”(.T.)或“假”(.F.)。下表列出了 DynamicFontShadow 属性的设置：

设置	说明
True (.T.)	文本带阴影显示。.
False (.F.)	（默认值）文本不带阴影显示

应用于

列

请参阅

FontShadow 属性, DynamicFontOutline 属性

DynamicFontSize 属性

指定列对象中显示文本的字体大小。运行期间每次刷新表格控件时，都要重新计算字体大小。设计时可用，运行时只读写。

语法

Column.FontSize[= “nSize”]

参数描述

“nSize”

计算结果等于字体大小。默认字体大小是 10 磅。

说明

nSize 的最大值是 2048 磅。1 英寸等于 72 磅

该属性用于改变字体的大小。

应用于

列

请参阅

DynamicAlignment 属性, DynamicFontName 属性

DynamicInputMask 属性

指定在一个 Column 对象中如何输入和显示日期。每当刷新 Grid 控件时都重新计算该属性的值。设计时可用，运行时可读写。

语法

Column.DynamicInputMask[= cInputMask]

参数描述

cInputMask

指定一个列中的文本和控件的动态格式。cInputMask 必须是一个符合下列格式的字符串：

[@cFunction][cMask]

以下代码表明了 cInputMask 的一种典型格式：

```
Column1.DynamicInputMask = "@R$ ###,###,###.##"
```

有关 cFunction 和 cMask 的格式的详细内容，请参阅本书稍后的“Format”和“InputMask”属性。

说明

DynamicInputMask 属性比 Format 与 InputMask 属性有优先权。

如果列的 CurrentControl 属性指定的控件是文本框、微调或组合框，则

DynamicInputMask 表达式的 Format 和 InputMask 属性被传递到文本框、微调或组合框的 Format 和 InputMask 属性。

应用于

列

请参阅

Format 属性, InputMask 属性

返回总目录

EXIT 命令
EXP（）函数
EXPORT 命令
EXTERNAL 命令
FCHSIZE（）函数
FCLOSE（）函数
FCOUNT（）函数
FCREATE（）函数
FDATE（）函数
FEOF（）函数
FERROR（）函数
FFLUSH（）函数
FGETS（）函数
FIELD（）函数
FILE（）函数
FileClass 属性
FileClassLibrary 属性
File 对象
文件集合（Files Collection）
FILETOSTR（）函数
FillColor 属性

FillStyle 属 性
Filter 属 性
FILTER（）函 数
FIND 命 令
FirstElement 属 性
FKLABEL（）函 数
FKMAX（）函 数

EDIT 命令

显示要编辑的字段。.

语法

```
EDIT
  [FIELDS FieldList]
  [Scope] [FOR lExpression1] [WHILE lExpression2]
  [FONT cFontName [, nFontSize]]
  [STYLE cFontStyle]
  [FREEZE FieldName]
  [KEY eExpression1 [, eExpression2]]
  [LAST | NOINIT]
  [LPARTITION]
  [NAME ObjectName]
  [NOAPPEND]
  [NODELETE]
  [NOEDIT | NOMODIFY]
```

[NOLINK]
[NOMENU]
[NOOPTIMIZE]
[NORMAL]
[NOWAIT]
[PARTITION nColumnNumber [LEDIT] [REDIT]]
[PREFERENCE PreferenceName]
[REST]
[SAVE]
[TIMEOUT nSeconds]
[TITLE cTitleText]
[VALID [:F] lExpression3 [ERROR cMessageText]]
[WHEN lExpression4]
[WIDTH nFieldWidth]
[WINDOW WindowName1]
[IN [WINDOW] WindowName2 | IN SCREEN
[COLOR SCHEME nSchemeNumber]

参数描述

FIELDS FieldList

指定出现在编辑窗口中的字段。这些字段按照 *FieldList* **中指定的顺序显示。在字段列表中可以包含其他相关表的字段。当包含相关表的字段时，应在字**

段名前加上表别名和一个句点。

如果省略 FIELDS，将按照字段在表结构中出现的顺序显示表的所有字段。

字段列表可以指定字段或计算结果字段的任意组合，包括其他工作区中打开表的字段。字段列表的语法是：

```
FieldName1
    [:R]
    [:nColumnWidth]
    [:V = lExpression1 [:F] [:E = cMessageText]]
    [:P = cFormatCodes]
    [:B = eLowerBound, eUpperBound [:F]]
    [:H = cHeadingText]
    [:W = lExpression2]
    [, FieldName2 [:R]...]
```

Calculated Fields

字段列表中可以包含语句，来创建计算结果字段。计算结果字段中包含用表达式创建的只读数据。表达式可以是任何形式，但它必须是一个有效的 Visual FoxPro 表达式。

用于创建计算结果字段的语法是：

```
CalculatedFieldName = eExpression
```

以下示例创建一个名为 location 的计算结果字段：

```
CLOSE DATABASES
```

```
OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer && 打开 customer 表
EDIT FIELDS location = ALLTRIM(city) + ', ' + country
```

FIELDS 子句的字段列表包含有 8 个选项，可以对编辑窗口中显示的字段进行特殊处理。

:nColumnWidth

以列为单位指定字段的显示宽度。*nColumnWidth* 的值并不影响表中该字段的大小，只是改变此字段在编辑窗口中的显示。

:R

下面的示例打开编辑窗口，并显示 cust_id 和 company 这两个字段。其中，字段 cust_id 是只读的，不能更改。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer && 打开 customer 表
EDIT FIELDS cust_id:R, company
```

:V = lExpression1

指定在编辑窗口内执行字段级数据有效性检查的选项。当把光标从此字段上移走时，如果 *lExpression1* 计算为“真”(.T.)，则说明向此字段输入的数据是正确的，光标将移动到下一字段。

如果 *lExpression1* 取值为“假”(.F.)，则输入的数据不正确，光标仍停留在此字段上并显示错误信息。如果 *lExpression1* 取值为 0，则所输入的数据不正确，光标仍停留在此字段上，但不显示任何错误信息。

默认情况下，只有当修改此字段时，才计算 *lExpression1* 的值。要强制进行检查，可加入 :F 选项。

如果加入 :E 选项，则可以显示自己定义的错误信息。

对于备注字段，不执行此检验选项。

:F

指定强制有效性检查选项，此选项决定从一个字段移走光标或另一个窗口被激活时，是否计算检验选项表达式 (*lExpression1*) 的值。如果不包含 :F 选项，则只有修改此字段时，才计算 *lExpression1* 的值。如果包含 :F 选项，即使此字段未作修改，也要计算 *lExpression1* 的值。

:E = cMessageText

显示由 *cMessageText* 指定的错误信息而不是默认的系统信息。

如果有效性检查表达式：V= *lExpression1* 计算为“真”(.T.)，则光标从字段中正常移出。如果该表达式计算为“假”(.F.)，则光标仍停留在字段上，并显示错误信息。

如果有效性检查表达式：V= lExpression1 计算为 0，则不显示错误信息，且光标仍停留在被检查的字段上，允许您在有效性检查例程中显示自定义的错误信息。

只有当 SET NOTIFY 为 ON 时，才能显示此错误信息。当 SET BELL 为 ON 时，将会响铃。

下面的示例打开 products 表并显示 product_id 和 prod_name 这两个字段。在 product_id

字段中输入一个大于 100 的值来执行字段有效性检验。
:V 指定了有效性检验准则。:F 说明无论数据是否更改，都要强制进行有效性检查。:E 把 Visual FoxPro 的系统错误信息替换为用户自定义的错误信息。
在 Visal FoxPro 中，错误信息都显示在主窗口底部的状态栏上。
按 ESC 键关闭编辑窗口。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE products  && 打开 products 表
IF _WINDOWS OR _MAC
    SET STATUS BAR ON
ENDIF
USE products
EDIT FIELDS in_stock :V = in_stock < 100 ;
    :F ;
    :E = 'The stock amount must be less than 100'
```

:P = cFormatCodes

指定一个模板选项，允许您创建由 *cFormatCodes* 指定的编辑模板，模板控制编辑窗口中各个字段数据的显示和输入。
有关使用模板编辑代码的详细内容，请参阅 @ ... GET 和 @ ... SAY。除了 M 代码以外，在 @ ... GET 和 @ ... SAY 中可用的所有函数和图片代码都能在 PICTURE 和

FUNCTION 子句中使用。

下面的示例使用了模板选项，从而在 unit_price 字段上只允许输入特定格式的数值型数据：

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE products  && 打开 products 表
EDIT FIELDS unit_price :P = '99,999.99'
```

:B = eLowerBound, eUpperBound [:F]

指定一组边界值，数据应在这两个边界值之间取值。边界值表达式 *eLowerBound* 和 *eUpperBound* 应该与此字段的数据类型相匹配，并且不能是用户自定义函数的函数名。如果输入的数据没有落在 *eLowerbound* 和 *eUpperBound* 之间，将会显示一条系统信息，指明数据的取值范围。

默认情况下，只有当更改字段的内容时，才用边界值检查输入的数据。要强制进行边界值检查，应选用强制检验项 (:F)。

下面的示例保证 in_stock 字段的值在 1 到 100 之间。按 ESC 键关闭编辑窗口。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE products  && 打开  products 表
EDIT FIELDS in_stock :B = 1, 100 :F
```

:H = cHeadingText

指定标题选项，允许用户用 *cHeadingText* 指定的标题替换默认的字段名称。默认情况下，编辑窗口中字段名放置在字段左边。

下面的示例为所显示的字段提供用户自定义标题。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE products && 打开 products 表
EDIT FIELDS prod_name :H = 'Product Name:', ;
     unit_price :H = 'Price per Unit:'
```

:W = lExpression2

指定 WHEN 选项，根据逻辑表达式 *lExpression* 的值有条件地禁止光标移动到某个字段上。(:W) 计算 *lExpression* 的值。如果 *lExpression2* 取值为“假”(.F.)，则不能把光标移到该字段上。如果 *lExpression2* 取值为“真”(.T.)，则可以把光标移到该字段上。*lExpression2* 也支持用户自义函数。

如果当前字段标记为只读，则禁止向任一字段移动光标。仅当每一字段都包含计算为“假”的 WHEN 子句时，才发生这种情况。

Scope

指定在编辑窗口中显示记录的范围。说明范围的子句有：ALL、NEXT *nRecords*、RECORD *nRecordNumber* 以及 REST。EDIT 命令默认的作用范围是所有记录 (ALL)。

详细内容，请参阅帮助中的“Scope Clauses”。

FOR lExpression1

规定只有满足逻辑条件 *lExpression1* 的记录才能在编辑窗口中显示。这有助于用户筛选出不需要的记录。

如果 *lExpression1* 是可优化的表达式，Rushmore 将优化 EDIT FOR 查询。为达到最佳性能，应在 FOR 子句中使用可优化表达式。有关 Rushmore 可优化表达式的详细内容，请参阅稍后的 SET OPTIMIZE 命令与《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十五章“优化应用程序”中的“掌握 Rushmore 技术”。

WHILE lExpression2

指定条件，只要逻辑表达式 *lExpression2* 计算为“真”(.T.)，记录就显示在编辑窗口中。

FONT cFontName [, nFontSize]

指定编辑窗口的字体和字体大小，其中字符表达式 *cFontName* 指定字体名，数值表达式 *nFontSize* 指定字体大小。例如，下面子句为编辑窗口中显示的字段指定了 16 磅的 Courier 字体：

```
FONT 'Courier', 16
```

如果包含了 FONT 子句，但省略了字体大小 *nFontSize*，编辑窗口将使用 10 磅的字体。

如果省略了 FONT 子句，则将使用 9 磅的 Arial 字体。如果指定的字体无效，则用类似的字体代替。

STYLE cFontStyle

指定在编辑窗口的字形。如果省略了 STYLE 子句，将使用常规字形。

如果您所指定的字形无效，则使用类似的字形代替。

字符	字形
B	粗体
I	斜体
N	常规
O	轮廓
Q	不透明
S	阴影
–	删除线
T	透明
U	下划线

可以包含一个以上的字符以指定组合字形。下面的示例打开编辑窗口，并使用加下划线的字形：

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + ‘data\testdata’)
USE customer  && 打开 customer 表

IF _WINDOWS
    EDIT FIELDS contact FONT ‘System’, 15  STYLE ‘NU’
```

```
ENDIF
IF _MAC
    EDIT FIELDS contact FONT 'Geneva', 14  STYLE 'NU'
ENDIF
```

FREEZE FieldName

在编辑窗口中只允许更改 *FieldName* **中指定的一个字段。其余字段能够显示，但不能编辑。**

KEY eExpression1 [, eExpression2]

限制在编辑窗口中所显示记录的范围。使用 KEY **子句，可以指定能够在编辑窗口中显示的记录的索引关键字值** (*eExpression1*) **或关键字值的范围** (*eExpression1*,*eExpression2*)**。此表应已建立索引，并且** KEY **子句中包含的索引关键字值的数据类型应该与主控索引文件或主控索引标识中索引表达式的数据类型一致。**

在下面的示例中，编辑窗口只显示邮政编码在 10000 **到** 30000 **范围内的记录：**

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer   && 打开 customer 表
SET ORDER TO postalcode
EDIT KEY '10000', '30000'
```

LAST | NOINIT

保存对编辑窗口外观设置所做的更改。这些更改将保存在 FOXUSER 文件中，其中包括对字段列表、每个字段的大小以及编辑窗口的位置和大小的更改。有关此文件的详细内容，请参阅 SET RESOURCE。

如果在 EDIT 命令中包含 LAST 子句，编辑窗口按照最近一次保存在 FOXUSER 文件中的配置打开。这将还原最近一次用 EDIT 命令创建的编辑窗口的配置。如果最近一次在命令窗口中发出的 EDIT 命令包含一长串子句，使用 EDIT LAST 命令可以避免重复键入这些命令。

如果按 CTRL+Q 退出编辑窗口，则不会保存当前工作期中对编辑窗口配置的更改。

LPARTITION

把光标放置在编辑窗口左分区的第一个字段上。使用 PARTITION 子句，可以把编辑窗口拆分为左、右两个分区。默认情况下，编辑窗口打开时光标放置在右分区的第一个字段上。

如果包含 LPARTITION 子句而无 PARTITION 子句，则光标将放置在编辑窗口的右分区内。

NAME ObjectName

创建编辑窗口的对象引用，允许使用表格控制面向对象的属性操作编辑窗口。

有关 Visual FoxPro 中的面向对象程序设计的附加内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》的第三章“面向对象程序设计”。有关可以为通

过 NAME 子句创建的“编辑”窗口指定的网格控件属性的详细内容，请参阅稍后的“Grid Control”。

NOAPPEND

禁止用户通过按 CTRL+Y 或从“查看”菜单中选择“追加方式”向表中添加记录。

重要提示 包含 NOAPPEND 并不禁止用户在编辑窗口中通过例程（用 VALID、WHEN 或 ON KEY LABEL 创建）追加记录。.

NODELETE

禁止在编辑窗口中为记录做删除标记。默认时，可以通过按 CTRL+T、从“表”菜单中选择“切换删除标记”，或单击待删除记录最左边的一列为记录作删除标记。

重要提示 包含 NODELETE 并不禁止用户在编辑窗口中通过例程（用 VALID、WHEN 或 ON KEY LABLE 创建）为记录做删除标记。

NOEDIT | NOMODIFY

禁止用户修改表。NOEDIT 与 NOMODIFY 是等价的。无论包含哪一个子句，都可以浏览或搜索此表，但不能编辑。不过，可以追加或删除记录。

NOLINK

解除编辑窗口中左、右分区的链接。默认时，编辑窗口中的左、右分区是链接的；当在某一分区滚动时，另一分区也相应滚动。

NOMENU

从系统菜单栏中移去 Visual FoxPro 的“表”菜单标题以及 FoxPro 早期版本

的“浏览”菜单标题，禁止访问“编辑”菜单。

NOOPTIMIZE

关闭 Rushmore 对 EDIT 命令的优化。

关闭 Rushmore 对 EDIT 命令的优化。有关 Rushmore 可优化表达式的详细内容，请参阅稍后的 SET OPTIMIZE 命令与《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十五章“优化应用程序”中的“掌握 Rushmore 技术”。

NORMAL

按照正常默认设置打开编辑窗口，这些设置包括颜色、大小、位置、标题和控制选项（GROW、FLOAT、ZOOM 等等）。如果省略 NORMAL，并且当前输出窗口是有其自己设置的用户自定义窗口，编辑窗口将使用这些自定义设置。

NOWAIT

编辑窗口打开后，立刻继续程序的执行。程序并不等待关闭编辑窗口，而是继续执行 EDIT NOWAIT 所在程序行下面的程序行。如果省略 NOWAIT，则在程序中发出 EDIT 命令时，将打开编辑窗口并暂停执行程序，直到编辑窗口关闭为止。

NOWAIT 只用于程序中。从命令窗口发出 EDIT 命令时，加入 NOWAIT 将不起作用。

PARTITION nColumnNumber

把编辑窗口拆分成左、右两个分区，*nColumnNumber* 指定拆分条所在的列。例如，如果 *nColumnNumber* 等于 20，则拆分条放置在编辑窗口的第 20 列上。

LEDIT
指定编辑窗口的左分区处于浏览方式。
REDIT
指定编辑窗口的右分区处于浏览方式。下面的示例将打开一个编辑窗口，拆分条放置在第 20 列上，并且右分区在打开时处于浏览方式。
同时包含两个关键字可以使左、右分区在打开时均处于浏览方式。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer  && 打开 customer 表

EDIT PARTITION 30 REDIT
```

PREFERENCE PreferenceName
保存编辑窗口的属性和选项供以后使用。LAST 能够还原前一工作期中出现的编辑窗口，与此不同，PREFERENCE 在 FOXUSER 资源文件中保存编辑窗口的属性而没有时间限制，可随时获取。有关 FOXUSER 资源文件的详细内容，请参阅 SET RESOURCE。
参数设置名称最多可包含 10 个字符，并应以字母或下划线开头，可包含字母、数字和下划线的任意组合。
如果您刻意要使用某个参数设置名称，可以禁止对其进行更改。关闭编辑窗口，发出 SET RESOURCE OFF 命令，并以表的方式打开 FOXUSER，然后把逻辑字段

READONLY 的值改为“真”(.T.)，就可以把包含参数设置的字段改为只读字段。有关 FOXUSER 资源文件的详细内容，请参阅 SET RESOURCE。

REST

禁止记录指针从当前位置移动到表的头部。默认情况下，EDIT 命令把记录指针定位于表的头部。

SAVE

使编辑窗口和所有备注字段的文本编辑窗口处于活动状态并可见（打开）。在使用鼠标或键盘循环遍历其他打开的窗口后，即返回到编辑窗口。

SAVE 只可用在程序中。在命令窗口中，使用 EDIT 命令时包含 SAVE 不起作用，因为 EDIT SAVE 在交互方式时总是默认的。

TIMEOUT nSeconds

指定编辑窗口等待输入的时间。数值表达式 *nSeonds* 指定经过多少秒仍没有输入，编辑窗口就自动关闭。

TIMEOUT 只在程序中可用。从命令窗口中发出带此子句的 EDIT 命令时没有什么作用。下面的示例中，如果 10 秒钟内没有输入，则编辑窗口关闭。

```
DEFINE WINDOW wEdit FROM 1,1 TO 24,40 ;
    CLOSE ;
    GROW ;
    COLOR SCHEME 10
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
```

```
USE customer  && 打开  customer  表
EDIT WINDOW wEdit ;
    FIELDS phone :H = ‘Phone Number:’ , ;
    company :H = ‘Company:’ ;
    TIMEOUT 10
RELEASE WINDOW wEdit
```

TITLE cTitleText

用 *cTitleText* 指定的标题改写编辑窗口标题栏中出现的默认表名或别名，否则，正被浏览的表的名称或别名将出现在标题栏上。

如果发出 EDIT WINDOWS 命令把编辑窗口放在某个用户自定义窗口中，则编辑窗口的标题将替换用户自定义窗口的标题。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + ‘data\testdata’)
USE customer  && 打开 customer 表
EDIT;
    TITLE ‘My Edit Window’ ;
    FIELDS phone :H = ‘Phone Number’ , ;
    company :H = ‘Company:’
```

VALID lExpression3

在编辑窗口中执行记录级有效性检查。只有在更改了记录，并且把光标移到

另一个记录上时才执行 VALID 子句。如果只是对备注字段作了更改，则不执行 VALID 子句。

如果 VALID 子句返回“真”(.T.)，那么可以把光标移到另一个记录上。如果 VALID 子句返回“假”(.F.)，则光标仍停留在当前字段上，Visual FoxPro 显示错误信息。如果包含 ERROR 子句，则可以在 VALID 子句返回“假”值时显示自定义的错误信息。字符表达式 *cMessageText* 作为错误信息显示。如果 VALID 子句的返回值为 0，则光标仍然停留在当前字段上，但不显示错误信息。

不要把 VALID 子句与启用字段有效性检查的检验选项 (:V) 混淆起来。

:F

强制在用户把光标移到下个一记录之前执行 VALID 子句。在这种情况下，即使当前记录未作更改，也执行 VALID 子句。

ERROR cMessageText

用 *cMessageText* 定义错误信息的内容，并指定用此错误信息改写系统默认信息。当 VALID 子句返回“假”(.F.) 时，Visual FoxPro 显示自定义的错误信息。

WHEN lExpression4

当用户把光标移到另一个记录上时，计算某个条件是否满足。如果 *lExpression4* 为“真”(.T.)，则用户可以修改光标所在的记录。如果 *lExpression4* 为“假”(.F.) 或 0，则光标所在的记录成为只读记录，不能修改。

当激活另一个窗口时，不执行 WHEN 子句。

WIDTH nFieldWidth

限制编辑窗口每个分区中所有字段所能显示的字符数不超过 *nFieldwidth*。包含 WIDTH 子句并不更改表中字段的大小，只改变字段在编辑窗口中的显示方式。如果已经用 FIELDS 子句指定了某个字段的宽度，则此宽度值将忽略由 WIDTH 子句为这个字段指定的宽度值。

WINDOW WindowName1

指定一个用户自定义窗口，编辑窗口具有与之相同的特征。例如，如果创建此用户自定义窗口时使用了 FLOAT 子句，则编辑窗口也可以移动。所指定的窗口不一定要活动或可见，但必须已定义。

IN [WINDOW] WindowName2

指定一个父窗口，编辑窗口在其中打开。编辑窗口并不继承父窗口的特征。在父窗口中激活的编辑窗口不能移出父窗口外。如果父窗口移动，则编辑窗口也随之移动。

要访问编辑窗口，必须首先用 DEFINE WINDOW 定义其父窗口，并且父窗口应该活动并可见。

IN SCREEN

当一个用户自定义窗口活动时，把编辑窗口明确地放置在 Visual FoxPro 主窗口中。

COLOR SCHEME nSchemeNumber

指定编辑窗口颜色的配色方案编号。在 Visual FoxPro 中，编辑窗口选用由“控制面板”的“颜色”程序建立的配色方案。

说明

EDIT 允许在窗口中编辑所选的表。EDIT 的作用等同于 CHANGE。
如果按下 ESC 键退出“编辑”窗口，将放弃对上一个字段所做的修改。但是如果修改字段后移到另一个记录，则对此字段的修改将被保存。
程序中用 DEACTIVATE WINDOW 来保存更改并关闭“编辑”窗口。在 DEACTIVATE WINDOW 中包含“编辑”窗口的名称。有关“编辑”窗口名称的详细内容，请参阅稍后的 WTITLE（）函数。

SET SKIP Support

SET SKIP 可以在两个表中建立一对多的关系（参见示例）。父表中的每个记录在子表中可以有多个相关记录。如果创建了一对多关系，就可以用 EDIT 查看父表和子表中的记录。
随同匹配的子表中的记录，父记录出现一次。后面的匹配记录接着父记录和第一个匹配的子记录在行中显示。在 MS-DOS 版本的 FoxPro 中，包含第一个匹配记录以下的父表信息的列中会显示阴影块。在 FoxPro 中，重复的父信息的填充字符取决于当前的“编辑”窗口字体。
详细内容，请参阅稍后的 SET SKIP 命令。

COL（）与 ROW（）Support

用 COL（）与 ROW（）返回“编辑”窗口中光标在屏幕上的当前行和列。如果在 Visual FoxPro 主窗口中打开“编辑”窗口，返回的光标位置与 Visual FoxPro 主窗口相关，而不是“编辑”窗口。如果在用户自定义窗口中打开“编辑”窗口，COL（）and ROW（）返回与用户自定义窗口相关的光标位置。

示例

下面的示例用 SET SKIP 在两个表间创建了一对多关系。对每个发票在父表(customer)中都有一个单独的记录。子表(orders)对每个记录包含多个记录。

创建关系后，将打开显示父表和子表的记录的“编辑”窗口。

```
CLEAR
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer ORDER cust_id IN 0  && 父表
USE orders ORDER cust_id IN 0  && 子表
SELECT customer    && 返回父工作区
SET RELATION TO cust_id INTO orders    && 建立关系
SET SKIP TO orders  && 一对多关系
WAIT WINDOW 'Scroll to see shipping dates for each customer' NOWAIT
EDIT FIELDS customer.cust_id :H='Customer Number', ;
    customer.city :H='Customer City', orders.shipped_on
```

请参阅

BROWSE, 表格控件, SET SKIP, WTITLE( )

EditBox 控件

创建一个编辑框。

语法

EditBox

说明

编辑框控件可用来编辑字符类型的变量、数组元素、字段或备注字段。
所有标准的 Visual FoxPro **编辑功能，如剪切、复制和粘贴，在编辑框中都可以使用。**
编辑框中的文本在垂直方向上可以滚动，在水平方向上可以自动换行。
有关创建编辑框的详细内容，请参阅《Microsoft Visual FoxPro 6.0 **中文版程序员指南》的第十章“使用控件”。**

属性

Alignment	AllowTabs	Application
BackColor	BackStyle	BaseClass

续 表

BorderColor	BorderStyle	Class
ClassLibrary	ColorScheme	ColorSource
Comment	ControlSource	DisabledBackColor
DisabledForeColor	DragIcon	DragMode
Enabled	FontBold	FontCondense
FontExtend	FontItalic	FontName
FontOutline	FontShadow	FontSize
FontStrikeThru	FontUnderline	ForeColor
Format	Height	HelpContextID
HideSelection	IMEMode	IntegralHeight
Left	Margin	MaxLength
MouseIcon	MousePointer	Name
NullDisplay	OLEDragMode	OLEDragPicture
OLEDropEffects	OLEDropHasData Property	OLEDropMode
OLEDropTextInsertion	Parent	ParentClass
PasswordChar	ReadOnly	RightToLeft
ScrollBars	SelectedBackColor	SelectedForeColor
SelectOnEntry	SelLength	SelStart

续 表

SelText	SpecialEffect	StatusBarText
TabIndex	TabStop	Tag
TerminateRead	Text	ToolTipText
Top	Value	Visible
WhatsThisHelpID	Width	

事 件

Click	DblClick	Destroy
DragDrop	DragOver	Error
ErrorMessage	GotFocus	Init
InteractiveChange	KeyPress	LostFocus
Message	MiddleClick Event	MouseDown
MouseMove	MouseUp	MouseWheel
OLECompleteDrag	OLEDragDrop	OLEDragOver
OLEGiveFeedBack	OLESetData	OLEStartDrag
ProgrammaticChange	RightClick	UIEnable
Valid	When	

方 法

AddProperty	CloneObject	Drag
Move	OLEDrag	ReadExpression

续表

ReadMethod	Refresh	ResetToDefault
SaveAsClass	SetFocus	ShowWhatsThis
WriteExpression	WriteMethod	ZOrder

请参阅

CREATE CLASS, CREATE FORM, DEFINE CLASS

EJECT 命令

向打印机发送换页符。.

语法

EJECT

说明

EJECT 命令要求打印机走纸到下一页的顶端。如果系统变量 _PADVANCE 设置为 FORMFEED，则 EJECT 命令向打印机发送换页符。如果 _PADVANCE 设置为 LINEFEEDS，则 EJECT 命令将不断发送换行符，直至打印机走纸到下一页的顶端。EJECT 命令把 PCOL（）和 PROW（）重置为打印机打印头当前的行、列位置值，但

并不影响系统变量 _PAGENO 或 _PLINENO 的值。

示例

下面的示例打印了 customer 表的 company 和 phone 字段。（请确认本示例中有打印机相连并打开）打印的行数超过 62 时会走纸。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer  && 打开 customer 表

SET DEVICE TO PRINTER
SET PRINT ON
DO WHILE NOT EOF( )
   @ PROW( )+1,10 SAY 'Company: ' + company
   @ PROW( )+1,10 SAY 'Phone: ' + phone
   @ PROW( )+1,1  SAY ''
   IF PROW（） > 62
      EJECT
   ENDIF
   SKIP
ENDDO
SET PRINT OFF
SET DEVICE TO SCREEN
```

请参阅

EJECT PAGE, ON PAGE, SET DEVICE, SET PRINTER, PCOL( ) , PROW( ) , 系统变量概览

EJECT PAGE 命令

向打印机发出有条件走纸的指令。

语法

EJECT PAGE

说明

使用 EJECT PAGE 命令可以连续走纸。走纸取决于 _PADVANCE 的值以及 ON PAGE 例程是否有效。

如果 _PADVANCE 设置为 FORMFEED 而 ON PAGE 例程没有运行，则 EJECT PAGE 命令执行下列操作：

- 如果打印机已经联机，则向打印机发送一个换页符。
- 根据系统变量 _PLENGTH 和 _PLINENO，向屏幕或者某个替补文件发送换行符。
- _PAGENO 加 1。
- PLINENO 设置为 0。
- 如果系统变量 _PADVANCE 设置为 LINEFEEDS，而 ON PAGE 例程正在运行，并且 _PLINENO 的值小于 ON PAGE 例程所指定的行数，则 EJECT

PAGE 命令将不断向打印机、Visual FoxPro 主窗口或某个替补文件或者以上两者发送换行符，直到打印机走纸到下一页顶端为止。

如果 ON PAGE 例程未执行，或当 _PADVANCE 设置为 LINEFEEDS，并且 _PLINENO 大于 ON PAGE 例程所指定的行号，则 EJECT PAGE 命令执行下列操作：

- 根据系统变量 _PLENGTH 和 _PLINENO，向打印机、Visual FoxPro 主窗口或其替补文件发送换行符。
- PAGENO 加 1。
- PLINENO 设置为 0。

请参阅

EJECT, ON PAGE, SET ALTERNATE, SET DEVICE, SET PRINTER, PCOL( ) , PROW( ) , 系统变量概览

EMPTY（）函数

确定表达式是否为空。

语法

EMPTY(eExpression)

返值类型

逻辑值

参数描述

eExpression

指定 EMPTY（）函数作用的表达式。可包含字符、数值、日期或逻辑表达式，也可以是已打开表的备注字段或通用字段的名称。当表达式取如下值时，EMPTY（）函数返回“真”(.T.)。

表达式类型	取值
字符型	空字符串、空格、制表符、回车、换行符或以上各字符的任意组合
数值型	0
货币型	0
浮点型	0
整型	0
双精度型	0
日期型	空（例如 CTOD(''））
日期时间型	空（例如 CTOT(''））
逻辑值	“假”(.F.)
备注字段	空（没有内容）
通用字段	空（没有 OLE 对象）

续表

图片	空（没有图片）

不能用 EMPTY（）函数确定变量对象引用是否为空。例如，变量中可以包含表单引用的对象。如果单击表单的弹出式菜单中的“关闭”命令，或者执行 CLEAR WINDOWS 来关闭表单，此变量包含 null 值。

下面的示例演示了如何使用 TYPE（）和 ISNULL（）函数判定变量的对象引用是否有效。

```
goMyForm = CREATEOBJECT('Form')
WAIT WINDOW IIF(TYPE('goMyForm') = 'O' AND !ISNULL(goMyForm), ;
    'goMyForm has valid object reference',;
    'goMyForm does not have valid object reference')
```

说明

如果表达式 *eExpression* 取值为空，则 EMPTY（）返回“真”(.T.)；否则，EMPTY（）返回“假”(.F.)。

示例

下面的示例打开 testdata 数据库的 customer 表。用 FOR ... ENDFOR 创建一个循环，在这个循环中用 EMPTY（）函数确定 TAG（）函数是否返回空字符串。显示每个结构索引标识的名称及其候选状态。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
```

```
USE customer   && 打开 customer 表

FOR nCount = 1 TO 254
   IF !EMPTY(TAG(nCount)) && 检查空字符串
   ? TAG(nCount) && 显示标识名
   ? CANDIDATE(nCount) && 显示候选状态
   ELSE
      EXIT && 当没有发现更多标识时退出循环
   ENDIF
ENDFOR
```

请参阅

LEN()

Enabled 属性

指定对象能否响应用户引发的事件。设计和运行时可用。

语法

Object.Enabled[= lExpr]

参数描述

lExpr

Enabled 属性的设置为：

设置	说明
“真”(.T.)	（默认值）对象可以响应事件。.
“假”(.F.)	对象不响应事件。

说明

Enabled 属性允许在运行时将对象设置为启用或禁用状态。例如，可以禁用应用程序当前状态中不使用的对象，也可以禁用某个控件以限制其使用权。例如，可以禁用编辑框，使之显示只读信息。如果禁用了某个控件，则不能再选择它。

当某个容器对象的 Enabled 属性设置为“假”(.F.)时，将同时禁用它所包含的所有控件。用户单击已禁用表单所包含的控件时，不会触发任何事件。

把计时器控件的 Enabled 属性设置为“假”(.F.)，可以禁用计时器控件，从而取消由计时器控件 Interval 属性指定的计时。

应用于

复选框，列，组合框，命令按钮，命令组，容器对象，控件对象，编辑框，表单，表格，图像，标签，线条，列表框，OLE 绑定型控件，OLE 容器控件，选项按钮，选项组，页面，页框，_SCREEN，形状，微调，文本框，计时器，工具栏

请参阅

Click 事件, DblClick 事件, DragDrop 事件, DragOver 事件, DropDown 事件, KeyPress 事件, MouseDown 事件, MouseMove 事件, MouseUp 事件, RightClick 事件, Scrolled 事件, Undock 事件

Encrypted 属性

指定项目中已编译的源代码是否加密。只在设计时可用。

语法

Object.Encrypted[= lExpression]

参数描述

lExpression

Encrypted **属性的设置有：**

设置	说明
“真” (.T.)	**已编译的源代码加密。将** lExpression **设置为“真”** (.T.) **可以为您的源代码提供附加的保护，这与在** COMPILE **命令中包含** ENCRYPT **参数相同。**

续表

“假” (.F.)	（默认值）已编译的源代码不加密

说明

项目中的源代码包括包括程序和格式文件、表单、标签、报表和可视类库的源代码和数据库中的存储过程。Encrypted 属性对应于“项目信息”对话框中“项目”选项卡的“加密”复选框。

应用于

项目对象

请参阅

COMPILE, Project Information 对话框

END TRANSACTION 命令

结束当前事务。

语法

END TRANSACTION

说明

END TRANSACTION命令保存事务过程中对表、表的备注文件或索引文件所作的任何修改并结束此事务。此时将提交 BEGIN TRANSACTION命令与 END TRANSACTION命令之间对数据库所作的更新。如果当前事务是第一个事务，或者是唯一的事务（即事务并不嵌套），则将所作的更改写入磁盘。

如果事务有嵌套，则 END TRANSACTION命令将已缓冲的更新放入更高层事务中。嵌套事务有可能改写在更高层事务中对数据的更改。

如果 END TRANSACTION命令产生错误（例如，没有足够磁盘空间写入所作的修改），则取消事务中所作的修改，并结束该事务。

请参阅

BEGIN TRANSACTION, ROLLBACK, TXNLEVEL()

EOF（）函数

确定记录指针是否超出当前表或指定表中的最后记录。

语法

EOF([nWorkArea | cTableAlias])

返值类型

逻辑值

参数描述

nWorkArea

指定表所在的工作区号。

cTableAlias

指定表的别名。

如果指定工作区中没有打开的表，则 EOF（）函数返回“假”(.F.)。

如果没有指定工作区或别名，则检查当前选定工作区中打开的表，看是否到达了表的最后一个记录。

说明

如果记录指针已指向表文件的末尾(EOF)，则 EOF（）返回“真”(.T.)。当记录指针超过表中的最后记录时，即到达表的末尾。例如，当 FIND、LOCATE 或 SEEK 命令不成功时，Visual FoxPro 将把记录指针移动到最后一个记录之后，EOF（）函数返回“真”(.T.)。当记录指针并不指向表的末尾时，EOF（）函数返回“假”(.F.)。

示例

下面的示例打开 customer 表，并且一次一页地列出公司名称，直到文件尾或者选择了“取消”才结束。

```
CLOSE DATABASES
CLEAR
OPEN DATABASE (HOME() + "samples\data\testdata")
```

```
USE customer
GO TOP
local recCtr, btnValue
recCtr = 0
btnValue = 1
DO WHILE btnValue = 1 AND NOT EOF()
    ? "Company : " + company
    recCtr = recCtr + 1
    if (recCtr % 20) = 0 then
        btnValue =MESSAGEBOX ("Click OK to continue, ;
            Cancel to quit.",33)
        clear
    endif
    Skip 1    && 下移一个记录
ENDDO
=MESSAGEBOX("Listing complete.",48)
```

请参阅

BOF(), GO | GOTO, SKIP

ERASE 命令

从磁盘上删除文件。

语法

ERASE FileName | ? [RECYCLE]

参数描述

FileName

指定要删除的文件。必须包括文件的扩展名。如果该文件在当前驱动器或目录以外的某驱动器或目录上，则应在文件名前包含路径。

FileName中包含诸如*和？之类的通配符。例如，要删除备份文件，使用ERASE *.bak命令。

?

显示“删除”对话框，从中可以选择要删除的文件。

RECYCLE

指定文件不会立即从磁盘中删除，并放置在 Microsoft Windows 回收站。

注意 当使用 ERASE 命令时，必须十分小心。此命令删除的文件不能再恢复。在删除文件前，即使 SET SAFETY 已设置为 ON，也不会有任何警告。

示例

下面的示例中，CUSTOMER.DBF 的结构和所有 country 为 USA 的记录被复制到表 backup 中。再复制 backup 中的数据到已打开的文本文件 temp 中，该文件在关闭时会被删除。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer  && 打开 customer 表
```

```
COPY STRUCTURE TO backup
USE backup
APPEND FROM customer FOR country = 'USA'
COPY TO temp TYPE DELIMITED

WAIT WINDOW 'Press Esc to close and erase temp.txt' NOWAIT
MODIFY FILE temp.txt NOEDIT
ERASE temp.txt
? IIF(FILE('temp.txt'),'File not deleted','File deleted')
USE
ERASE backup.dbf
```

请参阅

DELETE FILE

ERROR 命令

生成一个 Visual FoxPro 错误。

语法

ERROR nErrorNumber
| nErrorNumber, cMessageText1
| [cMessageText2]

参数描述

nErrorNumber

指定要生成错误的编号。在指定错误编号时，应使用标准的 Visual FoxPro 错误信息。

有关 Visual FoxPro 的错误信息和错误编号，请参阅联机帮助中的“错误信息”。

cMessageText1

指定在错误信息中显示的一段文本，并提供该错误的附加信息。例如，如果引用了一个并不存在的变量，Visual FoxPro 将在错误信息中提供此变量的名称。

cMessageText2

指定在错误信息中显示的一段文本。当指定 *cMessageText2* 而不是 *nErrorNumber* 时，将生成 Visual FoxPro 的错误编号 1098（用户自定义错误）。在 *cMessageText2* 中使用回车 (CHR(13)) 可以将部分错误信息移到下一行。

说明

可用 ERROR 命令测试错误处理例程或显示自定义错误信息。

如果在发出 ERROR 命令时，某个 ON ERROR 错误处理例程正在运行，Visual FoxPro 将执行这个 ON ERROR 例程。如果某个对象发生错误，则执行该对象的 Error 事件。

如果是从命令窗口发出 ERROR 命令，而 ON ERROR 错误处理例程未运行，Visual FoxPro 将显示错误信息。如果从程序中发出 ERROR 命令，而 ON ERROR 错误处理例程未运行，Visual FoxPro 将显示错误信息，并允许取消程序、挂起程序，或忽略此错误。

示例

下面的示例生成三条错误信息。第一条错误信息是 Visual FoxPro 错误信息“找不到变量”（错误编号 12）。第二条错误信息再生成 12 号错误，并包含有变量名 Myvariable。

最后一条错误信息是用户自定义错误信息（错误编号 1098）“My error message”。

```
ERROR 12 && 生成 the Visual FoxPro 错误“找不到变量”。
ERROR 12, 'Myvariable' && 变量 'Myvariable' 未找到，出错。
ERROR 'My error message' && 生成错误“My error message”。
```

请参阅

ON ERROR

Error 事件

当某方法在运行出错时，此事件发生。

语法

PROCEDURE Object.Error
LPARAMETERS [nIndex,] nError, cMethod, nLine

参数描述

Visual Foxpro **按下列顺序向** Error **事件传送三或四个参数。**

nIndex

用以唯一标识控件数组中的某个控件。

nError

Visual FoxPro **的错误编号。有关错误编号的列表，请参阅“帮助”中“技术参考”的“错误信息”部分。**

cMethod

存放造成此错误的方法。但是，如果方法调用了用户自定义函数，且正是在此函数中发生了错误，则 *cMethod* 包含的是这个用户自定义函数名，而不是调用此函数的方法名。

nLine

存放方法中或自定义函数中造成此错误的程序行号。

说明

Error 事件使得对象可以对错误进行处理。此事件忽略当前的 ON ERROR 例程，并允许各个对象在内部俘获并处理错误。

重要提示 只有当错误发生在代码中时，才调用 Error 事件。

应用于

ActiveDoc 对象，复选框，组合框，命令按钮，命令组，容器对象，控件对象，临时表，自定义，数据环境，编辑框，表单，表单集，表格，图像，标签，线条，列表框，OLE 绑定型控件，OLE 容器控件，选项按钮，选项组，页面，页框，ProjectHook 对象，关系，形状，微调，文本框，计时器，工具栏

请参阅

ON ERROR

ERROR（）函数

返回触发 ON ERROR 例程的错误编号。

语法

ERROR( )

返值类型

数值型

说明

ERROR（）函数返回最近一次错误的编号。必须有一个 ON ERROR 例程处于活动状态，才能使 ERROR（）函数返回非零值。

在程序执行中俘获了一个错误时，在 ON ERROR 例程中可以通过 ERROR（）函数返回错误类型。相应的错误信息可由 MESSAGE（）函数返回。

ERROR（）函数的返回值可被 RETRY 或 RETRY 命令重置。.

示例

下面的示例演示了一个简单的错误处理例程，在出现错误时显示信息的值。

```
CLEAR
ON ERROR DO errhand WITH ERROR( ), MESSAGE( )
```

```
*** 下一行生成错误，没有 BRWSE 命令

BRWSE
ON ERROR
RETURN

*** 错误处理 ***

PROCEDURE errhand
PARAMETER errnum,message
? Message
? 'Error number: '+ ALLTRIM(STR(Errnum))
RETURN
```

请参阅

MESSAGE(), ON ERROR, RETRY, RETURN

ErrorMessage 事件

包含 ErrorMessage 事件是为了 FoxPro 2.X 的向后兼容性。请使用 Valid 事件。

Eval 方法

对于 Visual FoxPro 应用程序自动服务程序的一个实例，计算一个表达式的值，并且返回结果。

语法

ApplicationObject.Eval(cExpression)

返值类型

字符型、数值型、货币型、日期型或逻辑型

参数描述

cExpression

指定要计算的表达式。cExpression 可以是一个文字字符串，或者是一个有效的 Visual FoxPro 表达式、变量、数组元素或任意数据类型的字段，用引号引上。

说明

cExpression 必须是一个合法的 Visual FoxPro 字符表达式。

应用于

Application 对象，_VFP 系统变量

请参阅

DoCmd 方法, SetVar 方法

EVALUATE( ) 函数

计算字符表达式的值并返回结果。

语法

EVALUATE(cExpression)

返值类型

字符型、数值型、货币型、日期型、日期时间型、逻辑型或备注型

参数描述

cExpression

指定要计算的表达式。*cExpression* 可以是原义字符串，也可以是引号括起的各种数据类型的有效 Visual FoxPro 表达式、变量、数组元素或字段。*cExpression* 中的字符不能超过 255 个。

只要可能，就应使用 EVALUATE( ) 和名称表达式来代替使用 & 的宏替换，因为 EVALUATE( ) 函数或名称表达式比宏替换的执行速度快。

说明

EVALUATE() 函数与 TYPE() 函数类似，只不过返回的是表达式的值而不是其类型。包含 EVALUATE() 函数的表达式不能使用 Rushmore 技术优化。

请参阅

TYPE()

Exclude 属性

指定当根据一个项目连编一个应用程序 (.app)、动态链接库 (.dll) 或可执行文件 (.exe) 时，是否在其中排除一个文件。设计和运行时可用。

语法

Object.Exclude[= lExpression]

参数描述

lExpression

指定当连编一个 .app、.dll 或 .exe 时，是否在其中排除一个文件。如果 lExpression 为“真” (.T.)，则排除该文件；否则，就包含该文件。

文件类型确定了 lExpression 的默认值。例如，当添加到项目中时，表被自

动排除。

说明

在项目管理器中，被排除的文件在文件名的前面有一个带斜杠的圈。在项目管理器窗口中，列出被排除的文件供您参考，但是如果应用程序需要它们，您需要手工发布它们。您也可以通过在“项目”菜单中选择“排除”命令，来排除一个文件。

应用于

文件对象

请参阅

Build 方法

Exclusive 属性

指定是否以独占方式打开某个与临时表对象相关联的表。设计时可用，运行时只读写。

语法

DataEnvironment.Cursor.Exclusive[= lExpr]

参数描述

lExpr

Exclusive 属性的设置有：

设置	说明
“真”(.T.)	（默认值）当加载数据环境时，以独占方式打开与临时表关联的表。
“假”(.F.)	当加载数据环境时，不以独占方式打开与临时表关联的表.

说明

注意 当使用 CURSORSETPROP( ) 函数访问临时表对象时，在运行时 Exclusive 属性只读。

加载数据环境时，与临时表关联的每个表都能以独占方式打开（在多用户环境中不允许其他用户访问该表），或以共享方式打开。可以使用 Exclusive 属性指定访问表的方式。

注意 如果 DataSession 属性设置为 2（私有数据工作期）时，所有临时表对象 Exclusive 属性的默认值将更改为“假”(.F.)。

对于视图，视图对象本身总是以共享方式打开。不过，定义视图的表受 Exclusive 属性的影响。对于本地视图，以独占还是以共享方式打开定义视图的 Visual FoxPro 表，取决于 Exclusive 属性的设置。Exclusive 属性对远程视图不起作用。.

Exclusive 属性与 USE 命令的 EXCLUSIVE 子句和 SHARE 子句作用相似。

应用于

临时表

请参阅

DataSession 属性, SET EXCLUSIVE, USE

EXIT 命令

从 DO WHILE、FOR 或 SCAN 循环中退出

语法

EXIT

说明

EXIT 命令把控制权从循环 DO WHILE ... ENDDO、FOR ... ENDFOR 或 SCAN ... ENDSCAN 中转交给紧接在 ENDDO、ENDFOR 或 ENDSCAN 后的命令。

示例

下面的示例用 DO WHILE 循环语句对库存中价格超过 20 美元的产品进行汇总，直到遇到文件尾 (EOF)。然后退出 DO WHILE 循环语句，并显示总和。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE products  && 打开 Products 表
SET TALK OFF
gnStockTot = 0
```

```
DO WHILE .T.  && 循环开始
    IF EOF( )
        EXIT
    ENDIF
    IF unit_price < 20
        SKIP
        LOOP
    ENDIF
    gnStockTot = gnStockTot + in_stock
    SKIP
ENDDO  && 循环结束

CLEAR
? 'Total items in stock valued over 20 dollars:'
?? gnStockTot
```

请参阅

DO WHILE …… ENDDO， FOR …… ENDFOR， SCAN …… ENDSCAN

EXP( ) 函数

返回 e^x 的值，其中 x 是某个给定的数值型表达式。

语法

EXP(nExpression)

返值类型

数值型

参数描述

nExpression

指定指数表达式 e^x 中的指数 x。

说明

自然对数的底 e 约等于 2.71828。EXP() 函数返回的小数位数由 SET DECIMALS 命令指定。

示例

```
? EXP(0) && 显示数值 1.00
? EXP(1) && 显示数值 2.72
```

请参阅

LOG(), SET DECIMALS

EXPORT 命令

把 Visual FoxPro 表中的数据复制到其他格式的文件中。

语法

```
EXPORT TO FileName
  [TYPE] DIF | MOD | SYLK
  | WK1 | WKS | WR1 | WRK | XLS | XL5
  [FIELDS FieldList]
  [Scope]
  [FOR lExpression1]
  [WHILE lExpression2]
  [NOOPTIMIZE]
  [AS nCodePage]
```

参数描述

FileName

指定文件名，Visual FoxPro 向其中导入数据。如果该文件名中不包含扩展名，则赋给它指定文件类型的默认扩展名。

TYPE

指定待创建文件的类型。关键字 TYPE 是可选的，但指定的值必须是下列某一文件类型。.

文件类型	说明
DIF	如果指定 DIF 类型，则 Visual FoxPro 表中的每个字段成为 VisiCalc 所使用的 DIF (Data Interchange Format)文件中的一个矢量（列），而每个记录成为该文件中的一个元组（行）。如果 *FileName* 没有包含扩展名，则为新文件名指定 DIF 扩展名。
MOD	MOD 子句用于向 Microsoft Multiplan 4.0 版 MOD 格式的文件导入数据。如果 *FileName* 没有包含扩展名，则为新文件名指定 .MOD 扩展名。
SYLK	对于 SYLK 文件（符号链接交换格式，Microsoft Multiplan 中使用），Visual FoxPro 表中的每个字段成为电子表格中的列，而每条记录成为电子表格中的一行。默认情况下，SYLK 文件没有扩展名。
WK1	包含此选项用于从 Visual FoxPro 的表中创建 Lotus-1-2-3 的电子表格。对于使用 Lotus-1-2-3 修订版 2.X 的电子表格文件，指定扩展名为 .WK1。表中的每个字段成为新电子表格的列，而每条记录成为此电子表格的一行。

续表

WKS	包含此选项可从 Visual FoxPro 表中创建 Lotus-1-2-3 电子表格。对于使用 Lotus-1-2-3 修订版 1 - A 的电子表格文件，指定扩展名为 .WKS。表中每个字段成为新电子表格的一列，而每条记录成为此电子表格的一行。
WR1	包含此选项可以从 Visual FoxPro 表中创建 Lotus Symphony 电子表格。对于使用 Symphony 1.01 版的电子表格，指定扩展名为 .WRI。表中的每个字段成为新建电子表格的一列，而每条记录成为此电子表格的一行。
WRK	包含此选项可从 Visual FoxPro 表中创建 Lotus Symphony 电子表格。对于使用 Symphony 1.01 版的电子表格，指定扩展名为 .WRK。表中的每个字段成为新电子表格的一列，而每条记录成为此电子表格的一行。
XLS	包含 XLS 选项可以创建在 Microsoft Excel 中使用的工作表。选定表的每个字段成为该工作表的一列，而每条记录成为一行。除非用户指定另一个扩展名，否则将为新创建的工作表文件指定文件扩展名 .XLS。
XL5	包含 XL5 选项可以创建 Microsoft Excel 5.0 版的工作表文件。当前选定表的每个字段成为该电子表格的一列，而每条记录成为其中的一行。如果没有包含文件扩展名，则将为此新建的工作表指定扩展名 .XLS。

FIELDS FieldList

指定要复制到新文件中的字段。如果省略 FIELDS 子句，则把全部字段复制到新文件中。即使在字段列表中包含备注字段和通用字段，也不会将它们复制到新文件中。

Scope

指定要复制到新文件中的记录范围，只有此范围内的记录才复制到新文件中。范围子句有 ALL、NEXT *nRecords*、RECORD *nRecordNumber* 和 REST。

有关范围子句的详细内容，包含 *Scope* 子句的命令只能操作活动工作区中的表。

EXPORT 命令的默认范围是所有记录。

FOR lExpression1

指定只有满足逻辑条件 *lExpression1* 的记录才被复制到新文件中。这可以帮助您筛选出不想要的记录。

如果 *lExpression1* 是可优化表达式，则 Rushmore 将优化 EXPORT ... FOR *lExpression1* 命令。为达到最佳性能，应在 FOR 子句中使用可优化表达式。

有关 Rushmore 可优化表达式的详细内容，请参阅稍后的 SET OPTIMIZE 命令与《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十五章“优化应用程序”中的“掌握 Rushmore 技术”。

WHILE lExpression2

指定一个条件，只要逻辑表达式 *lExpression2* 取值为“真”(.T.)，就把记录复制到新文件中。

NOOPTIMIZE

关闭对 EXPORT 命令的 Rushmore 优化。有关 Rushmore 可优化表达式的详细内容，请参阅稍后的 SET OPTIMIZE 命令与《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十五章“优化应用程序”中的“掌握 Rushmore 技术”。

AS nCodePage

为 EXPORT 命令创建的文件指定代码页。Visual FoxPro 不仅复制当前选定表的内容，而且在复制数据的同时，自动把数据转换到为新文件指定的代码页上。如果可能，Visual FoxPro 使用指定的代码页来标记新创建的文件。

如果不支持指定的 *nCodePage* 值，Visual FoxPro 将产生错误信息。可以用 GETCP() 函数代替 *nCodePage*，显示“代码页”对话框，从中指定代码页。

如果省略 AS *nCodePage*，则不转换代码页。如果可能，Visual FoxPro 将使用被复制数据的表所在的代码页来标记新创建的文件。

如果 *nCodePage* 为 0，则不转换代码页，并且不用代码页标记新建的文件。

说明

EXPORT 命令可以让其他软件包使用 Visual FoxPro 的数据。

如果导出数据的表已建立索引，则按索引顺序创建新文件。

请参阅

APPEND FROM, COPY TO, GETCP(), IMPORT

EXTERNAL 命令

向项目管理器提示一个未定义的引用。

语法

EXTERNAL FILE FileList | ARRAY ArrayList
| CLASS | FORM | LABEL | LIBRARY | MENU
| PROCEDURE | QUERY | REPORT | SCREEN | TABLE

参数描述

FILE FileList

告知项目管理器包含在间接文件引用或宏替换中的文件（诸如文本文件、.BMP 位图文件等等）是独立文件。*FileList* 可以包含一系列文件，文件名之间用逗号分隔。

ARRAY ArrayList

在程序中创建了某个数组，然后要在低一层的程序中使用此数组时，应在这个低层程序中包含 ARRAY 子句和该数组名。*ArrayList* 可以包含一系列数组名，数组名之间用逗号分隔。

下面的示例中，第一个程序创建了数组 gaInvoice。初始化该数组并调用低级程序 dispinvo。dispinvo 显示在高级程序中创建的数组的内容。包含 EXTERNAL ARRAY GAINVOICE 命令以警告项目管理器。

```
DIMENSION gaInvoice(4)
STORE 'Paid' TO gaInvoice
DO dispinvo
***  dispinvo 程序***
PROCEDURE dispinvo
EXTERNAL ARRAY gaInvoice
? gaInvoice(1)
? gaInvoice(2)
? gaInvoice(3)
? gaInvoice(4)
RETURN
***dispinvo 程序结束***
```

当向一个自定义函数或过程传递一个数组时，自定义函数或过程中相应的数组在项目管理器中必须是同一的。包含 ARRAY 选项时，需要使用 PARAMETER 语句中的数组名。

```
DIMENSION gaArrayOne(2)     && 建立一个数组
EXTERNAL ARRAY gaArrayTwo   && 用于 UDF 中的数组名称
SET TALK OFF
STORE 10 TO gaArrayOne(1)
STORE  2 TO gaArrayOne(2)
= ADDTWO(@gaArrayOne)       && 通过 UDF 的指示传递数组
FUNCTION ADDTWO
PARAMETER gaArrayTwo
CLEAR
gaArrayTwo(1) = gaArrayTwo(1) + 2
gaArrayTwo(2) = gaArrayTwo(2) + 2
? gaArrayTwo(1)
? gaArrayTwo(2)
```

CLASS

告知项目管理器包含在一个间接文件引用或宏替换中的文件是一个可视类库。

```
EXTERNAL CLASS myvclass && 类 myvclass 必须存在
STORE 'myvclass' TO gcClassFile
```

MODIFY CLASS (gcClassFile)

FORM

如果在间接文件引用或宏替换中包含表单定义文件，应包含 FORM 子句及该表单文件名。FORM 子句与 SCREEN 子句等价。.

EXTERNAL FORM dataentr && 表单文件 dataentr 必须存在

STORE 'dataentr' TO gcFormFile

DO FORM (gcFormFile)

LABEL

告知项目管理器包含在间接文件引用或宏替换中的文件是一个标签定义文件。

EXTERNAL LABEL Maillabl && 标签文件 Maillabel 必须存在

STORE 'Maillabl' TO gcLabelFile

LABEL FORM (gcLabelFile) PREVIEW

LIBRARY

当在 SET LIBRARY 命令中通过间接文件引用或宏替换引用某个库文件时，应包含 LIBRARY 子句。

EXTERNAL LIBRARY regress && 库文件 regress 必须存在

STORE 'regress' TO gcStatFunc

SET LIBRARY TO (gcStatFunc)

MENU

如果在间接文件引用或宏替换中包含有菜单定义文件时，应包含 MENU 子句及菜单文件名。

```
EXTERNAL MENU pickfile  && 菜单文件  Pickfile 必须存在
STORE 'pickfile' TO gcSysMenPad
MODIFY MENU (gcSysMenPad)
```

PROCEDURE

标识一个外部过程或用户自定义函数。

```
EXTERNAL PROCEDURE delblank    && 过程  delblank 必须存在
STORE 'delblank' TO gcTrimBlanks
DO (gcTrimBlanks) WITH 'A B C D E'
```

QUERY

提示项目管理器包含在间接文件引用或宏替换中的文件是一个查询文件。

```
EXTERNAL QUERY sales  &&  查询文件 sales 必须存在
STORE 'sales.qpr' TO gcSalesFile
DO (gcSalesFile)
```

REPORT

提示项目管理器包含在间接文件引用或宏替换中的文件是报表定义文件。

```
EXTERNAL REPORT overdue    && 报表文件 overdue 必须存在
STORE 'overdue' TO gcReportFile
REPORT FORM (gcReportFile) PREVIEW
```

SCREEN

如果在间接文件引用或宏替换中包含有表单定义文件，则应包含 SCREEN 子句及表单文件名。SCREEN 子句与 FORM 子句等价。

```
EXTERNAL SCREEN dataentr  && 表单文件 dataentr 必须存在
STORE 'dataentr' TO gcScreenFile
MODIFY SCREEN (gcScreenFile)
```

TABLE

提示项目管理器包含在间接文件引用或宏替换中的文件是 Visual FoxPro 的一个表。

```
EXTERNAL TABLE customer && 表  customer 必须存在
STORE 'customer' TO gcMyTable
USE (gcMyTable)
```

说明

使用 EXTERNAL 命令，可以在项目管理器创建的项目中包含文件并解决未定义引用

的问题。只有项目管理器才使用 EXTERNAL 命令，在程序的执行中忽略此命令。
有关用项目管理器创建项目的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十三章“编译应用程序”。
项目管理器将 EXTERNAL 命令中指定的文件包含在项目中。在文件名前应加入 CLASS、FILE、FORM、LABEL、LIBRARY、MENU、PROCEDURE、QUERY、REPORT、SCREEN 或 TABLE，以告知项目管理器要包含在项目中的文件类型。
当在名称表达式或宏替换中包含文件名时，也应给项目管理器以提示。这可以保证在项目连编时包含所有必要的文件。另外，在其他过程或用户自定义函数中创建的数组也必须提示给项目管理器。
有关名称表达式和宏替换的详细内容，请参阅 &命令。只要可能，就应使用名称表达式，而不要使用宏替换，这有助于提高程序的性能。

请参阅

BUILD APP, BUILD PROJECT

FCHSIZE( ) 函数

更改用低级文件函数所打开文件的大小。

语法

FCHSIZE(nFileHandle,nNewFileSize)

返值类型

数值型

参数描述

nFileHandle

指定希望改变其大小的文件句柄，此句柄可以在打开文件时由 FOPEN（）函数返回，或者在创建文件时用 FCREATE（）函数返回。如果使用 FOPEN（）函数打开文件，该文件应以“写”或“读写”方式打开，以便更改其大小。

nNewFileSize

以字节为单位，指定新的文件大小。如果 *nNewFileSize* 小于文件的原始大小，文件将被截断；如果 *nNewFileSize* 大于文件的原始大小，则增大文件。

说明

FCHSIZE（）函数根据指定字节数增大文件或截断文件。

当增大文件时，在打开文件所在的驱动器上，Visual FoxPro 将为文件分配扇区。由于 FCHSIZE（）函数并不对新的文件存储空间初始化，因而这些空间可能包含以前的数据，此时应对新的文件存储空间进行妥善的管理。

返回文件最终的字节数。由于某些原因（例如磁盘空间不够），如果 FCHSIZE（）函数所指定的文件句柄无效，或文件为只读文件，则 Visual FoxPro 返回 –1。

提示 此函数可把文件长度截短为 0。

请参阅

FCLOSE( ), FCREATE( ), FEOF( ), FFLUSH( ), FGETS( ), FOPEN( ), FPUTS( ), FREAD( ), FSEEK( ), FWRITE( )

FCLOSE（）函数

刷新并关闭用低级文件函数打开的文件或通信端口。

语法

FCLOSE(nFileHandle)

返值类型

逻辑值

参数描述

nFileHandle

指定要关闭的低级文件句柄。在使用 FCREATE（）函数创建文件或在使用 FOPEN（）函数打开文件时，可返回数值型的文件句柄。

说明

如果成功地关闭文件，则 FCLOSE（）函数返回“真”(.T.)，并释放该文件句柄；如果文件不能关闭，FCLOSE（）函数返回“假”(.F.)。

CLOSE ALL 命令也用于关闭低级文件。

请参阅

CLOSE ALL, FCHSIZE( ), FCREATE( ), FEOF( ), FFLUSH( ), FGETS( ), FOPEN( ), FPUTS( ), FREAD( ), FSEEK( ), FWRITE( )

FCOUNT（）函数

返回表中的字段数目。.

语法

FCOUNT([nWorkArea | cTableAlias])

返值类型

数值型

参数描述

nWorkArea

指定一个表所在的工作区，FCOUNT（）函数返回该表的字段数目。

如果在指定工作区没有打开的表，则 FCOUNT（）函数返回 0。

cTableAlias

指定表的别名，FCOUNT（）函数返回该表的字段数目。

如果所指定的表别名不存在，Visual FoxPro 将产生错误信息。

说明

如果省略可选的参数，FCOUNT（）函数将返回当前选定工作区中已打开表的字段数目。

示例

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer  && 打开 customer 表
SELECT 0
USE employee  && 打开 employee 表

CLEAR
? FCOUNT('CUSTOMER')   && 显示数值 13, Customer 表的字段数
? FCOUNT('EMPLOYEE')  && 显示数值 22, Employee 表的字段数
```

请参阅

DBF( ), FIELD( ), FSIZE( )

FCREATE（）函数

创建并打开低级文件。

语法

FCREATE(cFileName [, nFileAttribute])

返值类型

数值型

参数描述

cFileName

指定要创建的文件名称，在文件名前可以加入驱动器指示符和路径。如果没有包括驱动器指示符或路径，则在默认目录下创建该文件。

注意 磁盘或目录名称含有感叹号(!)时，Visual FoxPro 不会正确辨认路径。

nFileAttribute

指定文件的属性。下表列出了可以指定的文件属性。

nFile Attribute	文件属性
0	（默认值）读写
1	只读
2	隐含
3	只读 / 隐含
4	系统
5	只读 / 系统
6	系统 / 隐含
7	只读 / 隐含 / 系统

注意 nFileAttribute 不为 0 时创建的文件不能用 FPUTS（）或 FWRITE（）写入，直到关闭文件再重新打开。

可以使用 DISPLAY STATUS 或 LIST STATUS 命令显示或打印由 FCREATE（）函数创建并打开的文件信息，每个用低级文件函数打开或创建的文件可由 DISPLAY STATUS 和 LIST STATUS 给出下列信息：

- 驱动器、目录和文件名
- 文件句柄编号
- 文件指针的位置
- 读写属性

说明

如果指定的文件已经存在，则覆盖此文件，并不作任何警告。

FCREATE() 函数为此文件指定一个文件句柄编号，在其他的 Visual FoxPro 低级文件函数中，可用此编号标识该文件。在创建一个文件时，FCREATE() 函数返回该文件句柄编号，如果不能创建文件，则返回 -1。

提示 可以把文件句柄的编号赋给变量，从而可以在其他的低级文件函数中用此变量访问该文件。

示例

```
IF FILE('errors.txt') && 文件存在吗？
    gnErrFile = FOPEN('errors.txt',12)   && 如存在，打开并读写
ELSE
    gnErrFile = FCREATE('errors.txt') && 如不存在，建立一个
ENDIF
IF gnErrFile < 0      && 检查打开文件的错误
    WAIT 'Cannot open or create output file' WINDOW NOWAIT
ELSE  && 如无错误，开始写文件
    =FWRITE(gnErrFile , 'Error information to be written here')
ENDIF
=FCLOSE(gnErrFile )   && 关闭文件
IF gnErrFile > 0
MODIFY FILE errors.txt NOWAIT  && 在编辑框中打开文件
ENDIF
```

请参阅

CLOSE ALL, FCHSIZE( ), FCREATE( ), FEOF( ), FFLUSH( ), FGETS( ), FOPEN( ), FPUTS( ), FREAD( ), FSEEK( ), FWRITE( )

FDATE( ) 函数

返回文件最近一次修改的日期。

语法

FDATE(cFileName [, nType])

返值类型

日期型

参数描述

cFileName

指定的文件名，由 FDATE( ) 函数返回其最近一次修改的日期，*cFileName* 可在文件名前包含路径。如果文件名前不包含路径，Visual FoxPro 将在默认目录和 SET PATH 命令指定的所有目录下搜索该文件。

nType

指定 FDATE( ) 返回值的类型，用 cFileName 指定的文件的最近一次修改日期或日期时间。如果 nType 为 0，返回最近一次修改的日期。包含 0 等同于省略 nType*e*。如果 nType 为 1，返回最近一次修改的日期时间。

说明

FDATE（）函数返回的日期是由操作系统指定给该文件的。FDATE（）函数返回值的格式由 SET DATE、SET MARK 和 SET CENTURY 的当前设置决定。

使用 LUPDATE（）函数可获得某个打开表最近一次修改的日期。

示例

下面的示例使用 FDATE（）函数显示 Visual FoxPro 资源文件 FOXUSER.DBF 最近一次修改的日期。

```
? FDATE('FOXUSER.DBF')  && 显示最近一次修改的日期
```

请参阅

FTIME( ), LUPDATE( )

FEOF（）函数

判断文件指针的位置是否在文件尾部。

语法

FEOF(nFileHandle)

返值类型

逻辑值

参数描述

nFileHandle

指定要进行文件尾部状况检查的文件句柄编号。当指定的文件句柄编号是用FOPEN()打开的通信端口所对应的文件句柄编号时，FEOF()函数的返回值总为“真”(.T.)。

说明

对于用低级文件函数打开的文件，如果文件指针位置在文件尾部，则这个低级文件函数的返回值为“真”(.T.)；如果文件指针并不在文件尾部，则FEOF()函数的返回值为“假”(.F.)。

示例

```
*** 打开 test.txt 文件 ***
gnFileHandle = FOPEN('test.txt')
*** 把文件的指示器指向 BOF ***
gnPosition = FSEEK(gnFileHandle, 0)
*** 如果文件的指示器同时指向 BOF 和 EOF,文件为空。 ***
*** 否则，文件不为空 ***
IF FEOF(gnFileHandle)
    WAIT WINDOW 'This file is empty!' NOWAIT
ELSE
    WAIT WINDOW 'This file has something in it!' NOWAIT
```

```
ENDIF
= FCLOSE(gnFileHandle)
```

请参阅

FCHSIZE( ), FCLOSE( ), FCREATE( ), FGETS( ), FOPEN( ), FPUTS( ), FREAD( ), FSEEK( ), FWRITE( )

FERROR（）函数

返回与最近一次低级文件函数错误相对应的错误号。

语法

FERROR( )

返值类型

数值型

说明

如果低级文件函数执行成功，则 FERROR（）**函数返回** 0；**如果函数执行不成功，此函数返回一个正值。下表列出了** FERROR（）**函数返回的各个错误编号及错误原因。**

错误编号	错误原因
2	文件没有找到
4	打开的文件太多（文件句柄不够）
5	不能访问
6	给出的文件句柄无效
8	内存不足
25	移动文件指针时出错（无法将指针移到文件开始位置之前）
29	磁盘满
31	打开文件时出错

请参阅

FCHSIZE( ), FCLOSE( ), FCREATE( ), FEOF( ), FFLUSH( ), FGETS( ), FOPEN( ), FPUTS( ), FREAD( ), FSEEK( ), FWRITE( )

FFLUSH（）函数

刷新低级函数打开的文件内容，并将它写入磁盘。

语法

FFLUSH(nFileHandle)

返值类型

逻辑值

参数描述

nFileHandle

指定输出到磁盘的刷新文件的句柄。

说明

FFLUSH（）函数释放此文件缓冲区所占用的内存。

FLUSH 命令与 FFLUSH（）函数不同。FLUSH 命令不对低级文件操作，而对表和索引进行操作。

示例

下面的示例打开并写文件 Input.dat。写入前两个字符串后，程序刷新缓冲区，以确保字符串写入到磁盘中。然后再写后两个字符串，重新刷新缓冲区并关闭文件。

```
IF FILE('input.dat')
    gnTestFile = FOPEN('input.dat',2)
ELSE
    gnTestFile = FCREATE('input.dat')
ENDIF
gnIOBytes = FWRITE(gnTestFile,'Test output')
gnIOBytes = FWRITE(gnTestFile,' for low-level file I/O')
glFlushOk = FFLUSH(gnTestFile)
gnIOBytes = FWRITE(gnTestFile,'Test output2')
gnIOBytes = FWRITE(gnTestFile,' for low-level file I/O')
```

```
glFlushOk = FFLUSH(gnTestFile)
glCloseOk = FCLOSE(gnTestFile)
MODIFY FILE input.dat NOWAIT NOEDIT
```

请参阅

FCHSIZE( ), FCLOSE( ), FCREATE( ), FEOF( ), FGETS( ), FOPEN( ), FPUTS( ), FREAD( ), FSEEK( ), FWRITE( )

FGETS（）函数

从低级文件函数打开的文件中返回一系列字节，直至遇到回车符。

语法

FGETS(nFileHandle [, nBytes])

返值类型

字符型

参数描述

nFileHandle

指定一个数值型文件句柄，FGETS（）函数根据此句柄，从相应的文件或通信端口返回数据。

nBytes

指定由 FGETS() 函数返回的字节数。如果在此之前没有遇到回车符，FGETS() 函数返回 *nBytes* 个字节。如果回车符在 *nBytes* 个字节当中，则 FGETS() 函数返回文件指针起始位置与回车符之间的数据。

如果省略 *nBytes*，则 FGETS() 函数默认最多可返回 254 个字节。

说明

通过使用一串 FGETS() 函数，可以逐行阅读文件。

FGETS() 函数将一串字节作为单个字符串返回，所返回的数据从文件指针的当前位置开始，直至遇到回车符为止，此时文件指针将定位在紧接此回车符的字节上。回车符不包含在所返回的字节中，其中的换行符也被放弃。

示例

```
*** TEST.TXT 文件必须存在 ***
STORE FOPEN('test.txt') TO gnFileHandle  && 打开文件
STORE FSEEK(gnFileHandle, 0, 2) TO gnEnd  && 移动指示器到 EOF
STORE FSEEK(gnFileHandle, 0) TO gnTop     && 移动指示器到 BOF
IF gnEnd <= 0  && 文件为空吗?
    WAIT WINDOW 'This file is empty!' NOWAIT
ELSE  && If not
    gcString = FGETS(gnFileHandle, gnEnd)  &&保存内容
    ? gcString
ENDIF
= FCLOSE(gnFileHandle)  && 关闭文件
```

请参阅

FCHSIZE( ), FCLOSE( ), FCREATE( ), FEOF( ), FFLUSH( ), FILETOSTR( ), FOPEN( ), FPUTS( ), FREAD( ), FSEEK( ), FWRITE( )

FIELD（）函数

根据字段编号返回表中的字段名。

语法

FIELD(nFieldNumber [, nWorkArea | cTableAlias])

返值类型

字符型

参数描述

nFieldNumber

指定的字段编号。如果 *nFieldNumber* **等于** 1，**则返回表中的第一个字段名；如果** *nFieldNumber* **等于** 2，**则返回第二个字段名，依此类推。如果** *nFieldNumber* **大于字段的数目，则返回空字符串。返回的字段名为大写。**

nWorkArea

字段所属表的工作区。

如果在指定工作区中没有打开的表，FCOUNT（）**函数将返回空字符串。**

cTableAlias

字段所属表的别名。.

如果指定的表别名不存在，Visual FoxPro将产生错误信息。

说明

如果省略可选参数，FIELD()返回当前选定工作区中已打开表的字段名。

示例

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer  && 打开 customer 表

CLEAR
FOR gnCount = 1 TO FCOUNT()  && Loop for number of fields
   ? FIELD(gnCount) && 显示每一个字段
NEXT
?
? 'Number of fields: ' + ALLTRIM(STR(gnCount –1))
```

请参阅

DISPLAY STRUCTURE, FCOUNT(), FSIZE()

FILE（）函数

如果在磁盘上找到指定的文件，则返回“真”(.T.)。

语法

FILE(cFileName)

返值类型

逻辑值

参数描述

cFileName

指定要查找文件的名称，必须包含文件的扩展名。Visual FoxPro首先在默认目录下查找该文件，如果在默认目录下未找到该文件，Visual FoxPro将按照SET PATH命令建立的Visual FoxPro路径进行搜索。

可以在文件名前加入路径，从而在某个非当前目录或驱动器上搜索文件。

说明

FILE（）函数在磁盘上查找文件。如果找到文件，则返回“真”(.T.)；否则，返回“假”(.F.)。

示例

下面的示例显示了信息数值，此信息指出 Visual FoxPro 资源文件是否在 Visual FoxPro 启动目录中。

```
SET PATH TO HOME( )
CLEAR
IF FILE('foxuser.dbf')
    WAIT WINDOW 'Visual FoxPro resource file present'
ELSE
    WAIT WINDOW 'Visual FoxPro resource file not present'
ENDIF
```

请参阅

FULLPATH( ), GETFILE( ), LOCFILE( )

FileClass 属性

包含项目中一个表单的表单类的名称。设计和运行时只读。

语法

Object.FileClass

说明

FileClass 属性只应用于项目中的表单，并且对于其他文件类型包含空字符串。

应用于

文件对象

请参阅

FileClassLibrary **属性**, Type **属性**

FileClassLibrary **属性**

包含类库的名称，在该类库中包含项目中一个表单的表单类。设计和运行时只读。

语法

Object.FileClassLibrary

说明

FileClassLibrary **属性只应用于项目中的表单，并且对于其他文件类型包含空字符串。**

应用于

文件对象

请参阅

FileClass **属性**, Type **属性**

File 对象

当一个文件添加到项目中时实例化。

语法

File

说明

对于每一个添加到项目中的文件，都创建并且实例化一个文件对象。一个文件对象提供了对项目中文件的一个对象引用，并且允许您确定有关该文件的信息，并且通过文件对象的属性和方法管理该文件。

项目的文件集合包含项目中的所有文件。

附注 文件对象是一个 COM 对象，将一个文件对象引用分配给一个变量，会创建一个 “Unknown Type” 类的变量。

属性

CodePage	Description	Exclude
FileClass	FileClassLibrary	LastModified

续表

Name	ReadOnly	SCCStatus
Type		

方法

AddToSCC	CheckIn	CheckOut
GetLatestVersion	Modify	Remove
RemoveFromSCC	Run	UndoCheckOut

请参阅

文件集合（Files Collection）

文件集合（Files Collection）

项目中文件对象的集合。

语法

Files

说明

文件集合包含了项目中的所有文件。每个文件都是一个对象，可以使用文件对象的属性和方法管理它。.

文件集合中的文件可以被索引编号或名称引用。例如，下面的代码打开了最早添加到项目的文件：

```
_VFP.ActiveProject.Files(1).Modify( )
```

下面的代码为 Main.prg 打开一个编辑窗口：

```
_VFP.ActiveProject.Files('Main.prg').Modify( )
```

注意 不必在文件名称中包含路径。

有关文件集合与项目的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第三十二章“应用程序开发和开发者的生产率”中的“项目管理器挂接程序”。

属性

Count

方法

Add Item

请参阅

File **对象**

FILETOSTR（）函数

将一个文件的内容返回为一个字符串。

语法

FILETOSTR(cFileName)

返值类型

字符型

参数描述

cFileName

指定文件的名称，该文件的内容返回为一个字符串。如果该文件不在当前默认目录中，在文件名中需要包含路径。

说明

附注注意，FILETOSTR（）返回的字符串可以很大。可用内存或硬盘的空间大小决定了是否可以将该字符串保存到一个变量、数组元素或备注字段中。另外，Visual FoxPro 中字符字段限制为 254 个字符。

有关字符型数据的限制，请参阅帮助中的“Visual FoxPro 系统容量”。

请参阅

FGETS( ), FREAD( ), STRTOFILE( )

FillColor 属性

指定图形例程在对象上所画图形的填充颜色。设计和运行时可用。

语法

Object.FillColor[= nColor]

Object.FillColor[= RGB(nRedValue, nGreenValue, nBlueValue)]

参数描述

nColor

指定一个数字代表颜色，默认时，FillColor **设置为** 0（**黑色**）。**有关颜色设置的详细内容，请参阅** BackColor, ForeColor **属性主题。**

nRedValue, nGreenValue, nBlueValue

分别指定填充颜色中红、绿、蓝三种颜色的比例，必须使用 RGB（） **函数把三种颜色成分合并为一个数字，作为** FillColor **属性。**

注意 **在属性窗口中，双击任一颜色属性可以显示“颜色”对话框，用户可在此对话框中选择颜色或定义颜色。在“颜色”对话框关闭后，与选定颜色相对应的红、绿、蓝三种颜色的比例成为这些属性当前的设置值。**

说明

当 FillStyle 属性设置为默认值 1（透明）时，忽略 FillColor 设置。
只有封闭形状（如圆、方框或椭圆之类的形状）才能被填充。
形状对象使用 BackColor 属性填充颜色。

应用于

表单，_SCREEN，形状

请参阅

BackColor 属性，Box 方法，Circle 方法，FillStyle 属性，ForeColor，属性，GETCOLOR(), RGB()

FillStyle 属性

指定图案，用来填充圆和方框图形方法创建的形状和图形。

语法

Object.FillStyle[= nStyle]

参数描述

nStyle

指定填充形状或图形的图案。FillStyle 的设置有：

设置	说明
0	实心
1	（默认值）透明。忽略 FillColor 属性
2	水平线
3	垂直线
4	向上对角线。向上对角线由左上角开始，到右下角为止。.
5	向下对角线。向下对角线由左下角开始，到右上角为止。
6	十字线。垂直线与水平线交叉构成许多小方块
7	对角交叉线。.

应用于

表单，_SCREEN，形状

请参阅

BorderStyle 属性，Box 方法，Circle 方法，DrawMode 属性，DrawStyle 属性，DrawWidth 属性，FillColor 属性

Filter 属性

排除不满足条件的记录，筛选条件由给定表达式指定。

语法

DataEnvironment.Cursor.Filter[= cExpr]

参数描述

cExpr

任意 Visual FoxPro 表达式，通常是能对一组记录进行操作的表达式。

说明

与 SET FILTER 命令的作用相类似。

注意 如果使用 CURSORSETPROP( ) 函数访问临时表对象，Filter 属性在运行时只读。

应用于

临时表

请参阅

SET FILTER

FILTER（）函数

返回 SET FILTER 命令中指定的表筛选表达式。

语法

FILTER([nWorkArea | cTableAlias])

返值类型

字符型

参数描述

nWorkArea

指定表所在的工作区，FILTER（）函数返回该表的筛选表达式。

在指定的工作区中如果没有打开表，FILTER（）函数返回空字符串。

cTableAlias

指定表的别名，FILTER（）函数返回其筛选表达式。如果所指定的表别名不存在，Visual FoxPro 将产生错误信息。

说明

如果省略可选参数，FILTER（）函数返回在当前选定工作区中已打开表的筛选表达式。有关创建过滤器的详细内容，请参阅稍后的 SET FILTER 命令。

示例

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer  && 打开 Customer 表
SET TALK ON
SET FILTER TO SUBSTR(cust_id,1) = 'B'

CLEAR
? FILTER（）  && 显示筛选表达式
STORE FILTER('customer') TO gcOldFilter && 保存筛选表达式
SET FILTER TO country = 'USA'
? FILTER（）  && 显示筛选表达式
SET FILTER TO &gcOldFilter  && 还原筛选表达式
? FILTER（）  &&  显示筛选表达式

LIST FIELDS cust_id, contact  && 显示筛选条件
```

请参阅

SET FILTER

FIND 命令

包含此命令是为了提供向后兼容性。可用 SEEK 代替。

FirstElement 属性

指定在组合框或列表框控件内所显示的数组的第一个元素。在设计和运行时可用。

语法

Control.FirstElement[= nElement]

参数描述

nElement

指定列表中第一个显示的数组元素。

说明

只有当 RowSource 是数组（RowSourceType = 5）时，此属性才可用。注意

FirstElement 属性只应用于有一列的组合框和列表框。

应用于

组合框，列表框

请参阅

NumberOfElements 属性，RowSource 属性

FKLABEL() 函数

根据功能键对应的编号，返回该功能键的名称 (F1, F2, F3 ...)。

语法

FKLABEL(nFunctionKeyNumber)

返值类型

字符型

参数描述

nFunctionKeyNumber

指定功能键编号。*nFunctionKeyNumber* 的取值从 0 开始，最大值为功能键的数值减去 1。如果 *nFunctionKeyNumber* 比功能键的数目减 1 要大，

FKLABEL（）函数返回空字符串。功能键的数目可以使用 FKMAX（）函数得到。

说明

可以用 SET FUNCTION 命令对功能键编程。

FKLABEL（）函数的返回值受 SET COMPATIBLE 的影响。当 COMPATIBLE 设置为 FOXPLUS（默认值）时，FKLABEL（）函数返回功能键的名称；而当 COMPATIBLE 设置为 DB4 时，FKLABEL（）函数返回功能键与组合功能键 (F1、CTRL+F1、SHIFT+F1、F2、CTRL+F2、SHIFT+F2...) 的名称。

示例

```
CLEAR
SET COMPATIBLE OFF
? 'COMPATIBLE OFF'
?
FOR nCount = 1 TO FKMAX（） && 为功能键循环
   ? FKLABEL(nCount) && 显示程序的功能键
ENDFOR
SET COMPATIBLE ON

?
? 'COMPATIBLE ON'
?
FOR nCount = 1 TO FKMAX（） && 为功能键循环
   ? FKLABEL(nCount) && 显示程序的功能键
ENDFOR
```

请参阅

FKMAX(), SET COMPATIBLE, SET FUNCTION

FKMAX（）函数

返回键盘上可编程功能键或组合功能键的数目。

语法

FKMAX( )

返值类型

数值型

说明

FKMAX（）函数的返回值受 SET COMPATIBLE 的影响。当 SET COMPATIBLE 设置为 FOXPLUS（默认值）时，FKMAX（）函数返回功能键的数目；当 SET COMPATIBLE 设置为 DB4 时，FKMAX（）函数返回功能键与组合功能键 (F1、CTRL+F1、SHIFT+F1、F2、CTRL+F2、SHIFT+F2...) 的数目。

示例

```
CLEAR
SET COMPATIBLE OFF
```

```
? 'COMPATIBLE OFF'
?
FOR nCount = 1 TO FKMAX（） &&为功能键循环
   ? FKLABEL(nCount) &&显示程序的功能键
ENDFOR
SET COMPATIBLE ON

?
? 'COMPATIBLE ON'
?
FOR nCount = 1 TO FKMAX（） &&为功能键循环
   ? FKLABEL(nCount) &&显示程序的功能键
ENDFOR
```

请参阅

FKLABEL(), SET COMPATIBLE, SET FUNCTION

返回总目录

FORCEEXT() 函数
FORCEPATH() 函数
Form 对象
Format 属性
FormCount 属性
Forms 属性
FormSet 对象
FOUND() 函数
_FOXDOC 系统变量
FPUTS() 函数
FREAD() 函数
FREE TABLE 命令
FSEEK() 函数
FSIZE() 函数
FTIME() 函数
FullName 属性
FULLPATH() 函数
Function 命令
FV() 函数
FWRITE() 函数
_GALLERY 系统变量

GATHER 命令
_GENGRAPH 系统变量
_GENHTML 系统变量
_GENMENU 系统变量
_GENPD 系统变量
_GENSCRN 系统变量
_GENXTAB 系统变量
GETBAR() 函数
GETCOLOR() 函数
GETCP() 函数
GetData 方法
GETDIR() 函数
GETENV() 函数
GETEXPR 命令
_GETEXPR 系统变量
GETFILE() 函数
GETFLDSTATE() 函数
GETFONT() 函数
GetFormat 方法
GETHOST() 函数

FLDLIST() 函数

包含此命令是为了提供对 dBASE 的兼容性。

FLOCK() 函数

锁定当前表或指定表。

语法

FLOCK([nWorkArea | cTableAlias])

返值类型

逻辑值

参数描述

nWorkArea

指定表所在的工作区，FLOCK() 函数对此表进行锁定。

如果指定工作区没有打开的表，FLOCK() 函数的返回值为“假”(.F.)

cTableAlias

指定待锁定表的别名。

如果指定的表别名不存在，Visual FoxPro 将产生错误信息。

说明

如果成功锁定该表，FLOCK() 函数返回“真”(.T.)；而当该表或表中的某个记录已经被另一用户锁定，FLOCK() 函数返回“假”(.F.)。

如果省略可选参数，FLOCK() 函数对当前选定工作区中的已打开表进行锁定。

如果 FLOCK() 函数锁定表的操作失败，返回“假”(.F.)，但并不产生错误信息。因此，FLOCK() 函数不触发 ON ERROR 例程。

当某个表被锁定时，对表加锁的用户可以对此表进行读和写访问，网络上的其他用户则只能对此表进行只读访问。

对于锁定的表，只有加锁的用户才能解锁。可以通过执行 UNLOCK 命令、关闭该表或退出 Visual FoxPro 来解开被锁定的表。使用 USE、CLEAR ALL 或 CLOSE DATABASES 命令可以关闭表。

默认情况下，FLOCK() 函数对一个表只进行一次锁定尝试，使用 SET REPROCESS 可以在第一次尝试失败后自动重试对表的锁定操作。SET REPROCESS 可以给定尝试加锁的次数，也可以给定在初次锁定操作不成功时，加锁尝试持续的时间。

可以用 SET RELATION 命令在两个或更多的表之间建立关系。如果某个表与一个或多个表相关，则在此表上设置文件锁定并不会引起在相关表上设置文件锁定。必须在相关表上明确设置文件锁或移去文件锁。

有关网络上表的记录和文件的锁定和共享的其他内容，请参阅《Microsoft Visual

FoxPro 6.0 中文版程序员指南》第十七章“共享访问程序设计”。

示例

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE products    && 打开 products 表
SET REPROCESS TO 3 SECONDS
SELECT * FROM products INTO TABLE newprods

IF FLOCK()
    *** 新产品初始化 ***
    REPLACE ALL in_stock  WITH 0.00
    REPLACE ALL on_order WITH 0.00
    WAIT 'Initialization Complete' WINDOW NOWAIT
ELSE
    ***提醒用户文件被锁***
    WAIT WINDOW 'Unable to open products file; try again later!' NOWAIT
ENDIF

BROWSE FIELDS in_stock, on_order   && 显示 newprods 表
USE
ERASE newprods.dbf
```

请参阅

LOCK(), RLOCK(), SET REPROCESS, SET RELATION, UNLOCK, USE, SETEXCLUSIVE

FLOOR() 函数

对于给定的数值型表达式的值，返回小于或等于它的最大整数。

语法

FLOOR(nExpression)

返值类型

数值型

参数描述

nExpression

指定的数值型表达式，FLOOR() 函数返回小于或等于此表达式值的最大整数。

示例

```
STORE 10.9 TO gnNumber1
STORE –10.1 TO gnNumber2

CLEAR
? FLOOR(gnNumber1) && 显示数值 10
? FLOOR(gnNumber2) && 显示数值 –11
? FLOOR(10.0) && 显示数值 10
? FLOOR(–10.0) && 显示数值 -10
```

请参阅

CEILING(), INT(), ROUND()

FLUSH 命令

将对表和索引所作的修改存入磁盘。

语法

FLUSH

说明

FLUSH 命令确保对所有打开表和索引的修改都存入磁盘。

在下列情况下，Visual FoxPro 自动保存所做修改：

- 用 USE、CLOSE ALL 或 CLOSE DATABASES 命令关闭表。只有被关闭文件的有关信息存入磁盘。
- 对记录或文件解锁。只有被解锁记录或文件的信息存入磁盘。

请参阅

CLOSE DATABASES, FFLUSH(), SET AUTOSAVE

字体概述

Visual FoxPro 可以使用您已经安装的字体。字体决定了显示或打印文本的外观。另外，字体决定了控件的位置和大小

字体控制大小和位置

在 Visual FoxPro 中，表单的 ScaleMode 属性确定了表单上控件的大小和位置。如果 ScaleMode 设置为 Pixels (3)，则控件的大小是以像素指定的。如果 ScaleMode 设置为 Foxels (0)，则控件的大小由表单的当前字体和字体大小确定。

Foxel 是 Visual FoxPro 术语，对应于当前字体一个字符的最大高度和平均宽度。行高对应于当前字体一个字符的最大高度；列宽度对应于当前字体一个字符的平均宽度。

在 Visual FoxPro 中，您可以使用行列坐标的小数部分，以便精确定义控件和输出的位置。在 FoxPro for MS-DOS 中，忽略行列坐标的小数部分。

在 Visual FoxPro 中，为了确定或更改 Visual FoxPro 主窗口的字体，可以在显示“格式”菜单时按住 SHIFT 键，然后选择“屏幕字体”选项。当使用 DEFINE WINDOW 创建一个自定义窗口时，使用 FONT 子句，可以指定该窗口的字体。

字体替换

如果您指定了一种不可用的字体，则 Windows 会使用类似字体。Windows 考虑到磅值、衬线特征以及所须字体的斜度。通常会替换为一种 TrueType 字体。只有当您所需的字体与光栅或矢量字体非常匹配时，才会替换为光栅或矢量字体。

字体函数

有几个函数可以用来返回特定字体的信息。
这些函数包括：

函数	说明
AFONT()	将可用字体的信息放在一个数组中。
FONTMETRIC()	返回已安装字体的字体特征。
GETFONT()	显示“字体”对话框，并且返回所选字体的名称。
SYSMETRIC()	返回一个显示元素的大小。
SCOLS()	返回 Visual FoxPro 主窗口中可用列的数目。用于在 Visual FoxPro 主窗口中居中对齐文本或控件。
SROWS()	返回 Visual FoxPro 主窗口中可用行的数目。用于在 Visual FoxPro 主窗口中居中对齐文本或控件。
WCOLS()	返回指定窗口中列的数目。用于在用户自定义窗口中居中对齐文本或控件。
WFONT()	返回一个窗口的当前字体的名称、大小和字型
WROWS()	返回指定窗口中行的数目。用于在用户自定义窗口中居中对齐文本或控件。

请参阅

AFONT( ),FONTMETRIC( ),GETFONT( ),SYSMETRIC( ),ScaleMode 属性,SCOLS( ),SROWS( ),WCOLS( ),WFONT( ),WROWS( )

FontBold, FontItalic, FontStrikethru, FontUnderline 属性

指定文本是否具有下列效果：粗体，*斜体*，~~删除线~~或<u>下划线</u>。

语法

Object.FontBold[= lExpr]
Object.FontItalic[= lExpr]
Object.FontStrikeThru[= lExpr]
Object.FontUnderline[= lExpr]

参数描述

lExpr

字体属性的设置有：

设置	说明
“真”(.T.)	字体效果为粗体、斜体、加删除线或下划线。
“假”(.F.)	（默认值，但 FontBold 属性除外）字体效果不是粗体、斜体、加删除线或下划线

说明

通常在使用 FontSize、FontBold、FontItalic、FontStrikethru 和 FontUnderline 属性设置字体大小与字形之前，先设置 FontName 属性。不过，当设置小于 8 磅的 TrueType 字体时，应该先设置 FontSize 属性的磅值大小，然后设置 FontName 属性，最后重新设置 FontSize 属性的磅值大小。当 TrueType 字体小于 8 磅时，Windows 环境使用另一种不同的字体。

注意 可用字体随系统配置、显示设备和打印设备的不同而不同，与字体相关的属性只能根据字体的实际情况进行设置。

应用于

复选框，列，组合框，命令按钮，编辑框，表单，表格，标头，标签，列表框，选项按钮，页面，_SCREEN，微调，文本框

注意 页框对象不使用 FontBold 属性。

请参阅

FontName 属性，FontSize 属性，SetAll 方法

FontCondense, FontExtend 属性

指定文本是否具有紧缩或扩展样式。这些属性保留下来供以后使用。

FONTMETRIC() 函数

返回当前操作系统已安装字体的字体属性。

语法

FONTMETRIC(nAttribute [, cFontName, nFontSize [, cFontStyle]])

返值类型

数值型

参数描述

nAttribute

确定 FONTMETRIC() 函数返回的字体属性。如果省略 *cFontName*、*nFontSize* 和 *cFontStyle* 选项，FONTMETRIC() 函数将返回活动输出窗口当前字体的属性。

下表列出 *nAttribute* 的值与返回的相应字体属性。

nAttribute	属性
1	以像素为单位的字符高度
2	以像素为单位的字符提升值（高于基线的像素数）
3	以像素为单位的字符降低值（低于基线的像素数）

续表

4	以像素为单位表示的前导空间（线条的间距）
5	以像素为单位的附加前置空间*
6	以像素为单位的字符平均宽度
7	以像素为单位的字符最大宽度
8	字体灰度 (Font Weight)。
9	斜体（0 为否，非 0 为是）
10	下划线（0 为否，非 0 为是）
11	删除线（0 为否，非 0 为是）
12	字体中定义的第一个字符
13	字体中定义的最后一个字符
14	默认字符（替代字体中没有的字符）*
15	断字符。
16	间距和族 (pitch and family)*
17	字符集*
18	突出（额外增加的宽度）*
19	水平方向字体的设备
20	垂直方向字体的设备

有关 FONTMETRIC() 返回数值的详细内容，请参阅《Microsoft Windows 中文版程序员参考手册》中的 TEXTMETRIC 函数。

CFontName

指定一个已安装字体的名称。

NFontSize

指定字体 *cFontName* 的磅值大小。

CFontStyle

指定字体 *cFontName* 的字形。如果省略 *cFontStyle* 选项，FONTMETRIC() 函数返回“常规”字体的属性。

CFontStyle 可以是下面字形表中列出的字符或字符组合。例如，字符组合 BI 指定了粗斜体字形。

字符	字形
B	粗体
I	斜体
N	常规
O	轮廓
Q	不透明
S	阴影
–	删除线
T	透明
U	下划线

说明

FONTMETRIC() 函数返回活动输出窗口当前字体的属性，可使用 WFONT() 函数确定当前窗口的字体。

请参阅

AFONT(), **字体概览**, GETFONT(), SYSMETRIC(), TXTWIDTH(), WFONT()

FontName 属性

指定显示文本的字体名。

语法

Object.FontName[= cName]

参数描述

cName

指定字体的名称，默认值为 Arial。

说明

FontSize **默认的设置是** 9 **磅。系统配置不同，可用的字体也不同。与字体相关的属性只能被设置为与该字体相关的值。在设计时，当在“属性”窗口中选定** FontName **属性，并单击“属性设置”框右侧的下箭头时，将显示可用字体列表。**

一般情况下，在用 FontSize、FontBold、FontItalic、FontStrikethru **和** FontUnderline **这些属性设置字体的大小或字形之前，先设置** FontName **属性。**

应用于

复选框，列，组合框，命令按钮，编辑框，表单，表格，标头，标签，列表框，选项按钮，页面，_SCREEN，**微调，文本框**

请参阅

FontBold **属性**, FontItalic **属性**, FontStrikethru **属性**, FontUnderline **属性**, FontSize **属性**

FontOutline 属性

指定与某个控件相关的文本是否加上轮廓。包含此命令是为了 FoxPro for Macintosh 的向后兼容性。

FontShadow 属性

包含此命令是为了 FoxPro for Macintosh 的向后兼容性。

FontSize 属性

指定对象文本的字体大小。

语法

Object.FontSize[= nSize]

参数描述

nSize

以磅为单位指定字体大小，默认值为 10 磅。

说明

设计和运行时均可用。

nSize 的最大值是 127 磅。72 磅相当于 1 英寸。

一般情况下，在用 FontSize、FontBold、FontItalic、FontStrikethru 和 FontUnderlined 属性设置字体的大小和字形之前，先设置 FontName 属性。不过在设置小于 8 磅的 TrueType 字体时，应该先设置 FontSize 属性，然后设置 FontName 属性，最后重新设置 FontSize 属性。当 TrueType 字体小于 8 磅时，Windows 环境使用另一种不同的字体。

注意 可用字体随系统配置和打印设备的不同而不同。与字体相关的属性只能根据字体的实际情况进行设置。

应用于

复选框，列，组合框，命令按钮，编辑框，表单，表格，标头，标签，列表框，选项按钮，页面，_SCREEN，微调，文本框

请参阅

FontBold 属性，FontItalic 属性，FontStrikethru 属性，FontUnderline 属性

FOPEN() 函数

打开文件，供低级文件函数使用。

语法

FOPEN(cFileName [, nAttribute])

返值类型

数值型

参数描述

cFileName

指定要打开的文件的名称。*cFileName* 中可包含 Visual FoxPro 搜索路径中未指定的驱动器名、目录名或文件夹名；如果不含路径，Visual FoxPro 将在以下位置搜索文件：

- **默认目录**
- **SET PATH 命令建立的路径。**

注意 **如果一个驱动器名或目录名包含一个惊叹号 (!)，则 Visual FoxPro 无法正确识别该名。**

nAttribute

指定打开文件的读写权限或缓冲方案，下表列出可包含在 *nAttribute* 中的数

字及其相应的文件读写权限和缓冲方案。

nAttribute	读写权限	缓冲 / 非缓冲
0	（默认值）只读	缓冲
1	只写	缓冲
2	读写	缓冲
10	只读	非缓冲
11	只写	非缓冲
12	读写	非缓冲

如果不包含 *nArribute* 或 *nAttribute* 等于 0，则以缓冲方式用只读权限打开文件。

注意 如果磁盘或目录名中包含一个惊叹号 (!)，Visual FoxPro 不能正确地识别路径名。

说明

如果 FOPEN() 函数成功地打开文件或通信端口，则函数返回文件或端口的文件句柄号；如果文件或端口无法打开，则 FOPEN() 函数返回 –1。可以把文件句柄号赋给某个变量，从而在其他低级文件函数中通过这个变量访问相应文件。

可以使用 DISPLAY STATUS 或 LIST STATUS 命令显示、打印有关 FOPEN() 函数打开的文件的信息。

- 驱动器、目录及文件名
- 文件句柄号
- 文件指针位置

- **读写属性**

示例

```
IF FILE('errors.txt') && 文件存在吗？
    gnErrFile = FOPEN('errors.txt',12) && 如果存在，打开读-写
ELSE
    gnErrFile = FCREATE('errors.txt') && 如果不存在，创建文件
ENDIF
IF gnErrFile < 0 && 检查打开文件错误
    WAIT 'Cannot open or create output file' WINDOW NOWAIT
ELSE && If no error, write to file
    =FWRITE(gnErrFile, 'Error information to be written here')
ENDIF
=FCLOSE(gnErrFile) && 关闭文件
MODIFY FILE errors.txt NOWAIT && 在编辑窗口中打开文件
```

请参阅

CLOSE ALL, FCHSIZE(), FCLOSE(), FCREATE(), FEOF(), FFLUSH(), FGETS(), FPUTS(), FREAD(), FSEEK(), FWRITE()

FOR ... ENDFOR 命令

按指定的次数重复执行一组命令。.

语法

FOR Var = nInitialValue TO nFinalValue [STEP nIncrement]
Commands
[EXIT]
[LOOP]
ENDFOR | NEXT

参数描述

Var

指定一个变量或数组元素作为计数器。在执行 FOR...ENDFOR **语句之前，变量或数组不必存在。**

nInitialValue TO nFinalValue

nInitialValue **为计数器的初始值，** nFinalValue **为计数器的终止值。**

STEP nIncrement

nIncrement**显示计数器增加或减少的数量。如果** nIncrement**显示为负数，计数器将减少。如果省略这一步，计数器默认为** 1**。**

Commands

指定要执行的 Visual FoxPro **命令，** *Commands* **可以包含任意数目的命令。**

EXIT

在 FOR ... ENDFOR**循环中，将控制权传递给紧接在** ENDFOR**后面的命令。可以在** FOR **与** ENDFOR**之间的任何地方放置** EXIT**。**

LOOP

将控制权直接返给 FOR**子句，不再执行** LOOP **与** ENDFOR**之间的语句。计**

数器正常递增或递减，就像执行到 ENDFOR 一样。可以在 FOR 与 ENDFOR 之间的任何地方放置 LOOP。

说明

可以用变量或数组元素作为计数器，指定 FOR ... ENDFOR 循环中 Visual FoxPro 命令的执行次数。

在遇到 ENDFOR 或 NEXT 之前，始终执行 FOR 后面的 Visual FoxPro 命令。执行过程中，每循环一次，计数器 *MemVarName* 都会递增，增量为 *nIncrement*（如果省略 STEP 子句，则计数器每次的增量为 1），然后把计数器的值与 *nFinalValue* 作比较。如果计数器的值小于或等于 *nFinalValue*，将再次执行 FOR 语句后的命令；如果计数器的值大于 *nFinalValue*，则退出 FOR ... ENDFOR 循环，程序继续执行 ENDFOR 或 NEXT 之后的第一条命令。

注意 *nInitialValue*、*nFinalValue* 和 *nIncrement* 只能是初次读入的值。在循环过程中更改计数器 MemVarName 的值将影响循环的执行次数。

如果 *nIncrement* 为负，且初始值 *nInitialValue* 大于最终值 *nFinalValue*，则每经过一次循环，计数器都将递减。

示例

在示例 1 中显示从 1 到 10 的数字。

示例 2 使用变量设置初始值、终止值和“步进”值以显示 customer 表中从 2 到 10 的所有奇数记录。

```
*示例 1
CLEAR
FOR gnCount = 1 TO 10
   ? gnCount
```

```
ENDFOR

* 示例 2
SET TALK OFF
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer  && 打开 Customer 表
STORE 2 TO gnI  && 初始值
STORE 10 TO gnJ  && 终止值
STORE 2 TO K  && 步进值
FOR gnCount = gnI TO gnJ STEP K
   GOTO gnCount  && 移动记录指针
   DISPLAY company && 显示 company 名称
ENDFOR
```

请参阅

DO CASE...ENDCASE,DO WHILE...ENDDO,IF...ENDIF, SCAN ... ENDSCAN

FOR EACH ... ENDFOR 命令

对 Visual FoxPro 数组或集合中的每一个元素，依次执行一组命令。

语法

```
FOR EACH Var IN Group
  Commands
  [EXIT]
  [LOOP]
ENDFOR | NEXT [Var]
```

参数描述

Var

一个变量或数组元素，用来依次引用 *Group* 中的每一个元素。

Group

可以是一个 Visual FoxPro 数组，OLE 数组，或是一个 Visual FoxPro 集合或 OLE 集合。

Commands

可以包含任意数目的命令。对于 *Group* 中的每一个元素，执行这些命令。

EXIT

跳出 FOR EACH ... ENDFOR 循环，执行 ENDFOR 后的第一条命令。EXIT 可以出现在 FOR EACH 和 ENDFOR 之间的任何地方。

LOOP

跳回到 FOR EACH 子句，而不去执行 LOOP 和 ENDFOR 之间的命令。LOOP 可以出现在 FOR EACH 和 ENDFOR 之间的任何地方。

示例

以下示例演示了如何使用 FOR EACH 语句枚举 Visual FoxPro 数组中的元素、一个 OLE 数组和赋给对象数组的一套命令按钮。

以下示例中创建了一个 Visual FoxPro 变量并使用 FOR EACH 语句显示数组中每个元素的内容。

```
DIMENSION cMyArray(3)
cMyArray[1] = 'A'
cMyArray[2] = 'B'
cMyArray[3] = 'C'
FOR EACH cMyVar IN cMyArray
    ? cMyVar
ENDFOR
```

在以下示例中，创建了一个 Microsoft Excel 实例，添加了一个新的工作簿。使用 FOR EACH 语句显示工作簿中每个工作表的名称。此示例要求在运行示例的机器上正确安装 Microsoft Excel。

```
oExcel = CREATE("Excel.Application")
oExcel.Workbooks.ADD

FOR EACH oMyVar IN oExcel.sheets
    ? oMyVar.name
NEXT oMyVar
```

在以下示例，在表单放置了五个命令按钮。使用 FOR EACH 命令显示表单上的按钮并指定它们的标题、字体和每个按钮的位置。

```
PUBLIC oMyObject
oMyObject = CREATEOBJECT("frmTest")
```

```
oMyObject.SHOW

DEFINE CLASS frmTest AS FORM
Height = 200
DIMENSION MyArray[5]
    PROCEDURE Init

        FOR i = 1 to 5
            THIS.AddObject('THIS.MyArray[i]',;
                'COMMANDBUTTON')
        ENDFOR

        ****** FOR EACH - NEXT ******
        FOR EACH oButton IN THIS.MyArray
            oButton.Visible = .T.
        NEXT

        ****** FOR EACH - NEXT element  ******
        FOR EACH oButton IN THIS.MyArray
            oButton.FontBold = .T.
        NEXT obutton

        j = 1
        ****** FOR EACH - ENDFOR ******
        FOR EACH oButton IN THIS.MyArray
            oButton.top = j * 30
            j = j + 1
        ENDFOR
        ****** FOR EACH - ENDFOR element ******
        FOR EACH oButton IN THIS.MyArray
            oButton.FontItalic = .T.
        ENDFOR obutton
```

```
        j = 1
        ****** EXIT  ******
        FOR EACH oButton IN THIS.MyArray
            oButton.Caption = "test" + str(j)
            j = j+1
            IF j > 3
                EXIT
            ENDIF
        NEXT

        j = 1
        ****** LOOP  ******
        FOR EACH oButton IN THIS.MyArray
            IF j > 3
                LOOP
            ENDIF
            j = j + 1
            oButton.Left = 25
        NEXT
    ENDPROC
ENDDEFINE
```

请 参 阅

FOR...ENDFOR

FOR() 函数

返回一个已打开的单项索引文件或索引标识的索引筛选表达式。

语法

FOR([nIndexNumber [, nWorkArea | cTableAlias]])

返值类型

字符型

参数描述

如果不包含任何可选参数，FOR() 函数将返回主控索引文件或主控索引标识的索引筛选表达式。如果主控索引文件或主控索引标识不发生作用（例如，已经发出 SET ORDER TO 命令，使表中记录按物理记录次序排列），FOR() 函数将返回空字符串。

nIndexNumber

指定返回筛选表达式的索引文件或索引标识。*nIndexNumber* 从 1 开始递增，直至等于已打开的单项索引文件、结构复合索引标识及独立复合索引标识的总数，在此过程中，FOR() 函数将按下列顺序返回筛选表达式：

1. 返回单项索引文件（如果已打开）的筛选表达式，在 USE 或 SET INDEX 中包含的单项索引文件顺序决定了筛选表达式返回的顺序。
2. 返回结构复合索引（如果存在）中每一标识的筛选表达式，标识的筛选表达式按照在结构索引中创建标识的顺序返回。

3. 返回所有打开的独立复合索引中各个标识的筛选表达式，标识的筛选表达式将按照在独立的复合索引中创建标识的顺序返回。

如果创建索引或索引标识时没有使用 FOR 子句，或 *nIndexNumber* 大于打开的单项索引文件、结构复合索引标识和独立复合索引标识的总数，函数将返回空字符串。

nWorkArea

指定表所在的工作区，FOR() 函数返回该表的索引筛选表达式。

如果在指定工作区中没有打开的表，FOR() 函数返回空字符串。

cTableAlias

指定表的别名，FOR() 函数返回其索引筛选表达式。

如果指定的表别名不存在，Visual FoxPro 将产生错误信息。

说明

在 Visual FoxPro 中可以创建筛选索引 (filtered Index)。如果在 INDEX 命令中包含可选的 FOR *lExpression* 子句，索引文件将成为表的过滤器。只能显示和访问与筛选表达式 *lExpression* 匹配的记录，而且在索引文件中只对与筛选表达式匹配的记录创建索引关键字。

USE 和 SET INDEX 命令均支持索引文件名列表，该列表允许您打开表的多个索引。在索引文件名列表中可以包含单项索引文件名、结构化复合索引或独立复合索引文件名的任意组合。FOR() 与 SYS(2021) 命令等同并提供了与 dBASE IV 相兼容的功能。

请参阅

INDEX

FORCEEXT() 函数

返回一个字符串，使用新的扩展名替换旧的扩展名

语法

FORCEEXT(*cFileName*, *cExtension*)

返值类型

字符型

参数描述

cFileName

指定文件名（带有或不带有路径或扩展名），该文件名将获得一个新的扩展名。

cExtension

指定了 cFileName **的新扩展名（带有点号）。**

请参阅

ADDBS(), DEFAULTEXT(), FILE(), FORCEPATH(), JUSTDRIVE(), JUSTEXT(), JUSTFNAME(), JUSTPATH(), JUSTSTEM()

FORCEPATH() 函数

返回一个文件名，使用新路径名代替旧文件名。

语法

FORCEPATH(*cFileName*, *cPath*)

返值类型

字符型

参数描述

cFileName

指定文件名（带有或不带有路径或扩展名），该文件名将获得一个新的路径。

cPath

指定 cFileName *的新路径*。

请参阅

ADDBS(), DEFAULTEXT(), FILE(), FORCEEXT(), JUSTDRIVE(), JUSTEXT(), JUSTFNAME(), JUSTPATH(), JUSTSTEM()

Form 对象

创建表单。

语法

Form

说明

使用 Form 对象可以创建能向其中添加控件的表单，您也可以使用表单设计器创建表单。表单属性决定了表单的外观（如位置、大小和颜色）及特性（例如是否可调整）。

表单还可以对用户启动或系统触发的事件作出响应。例如可以在表单的 Click 事件过程中编写程序，从而在单击表单时，更改表单的颜色。

除了属性和事件之外，还可以使用方法操作表单。例如，可以使用 Move 方法更改表单的位置和大小。

在设计表单时，可使用 BorderStyle 属性指定表单的边框，使用 Caption 属性指定标题栏的文本。BorderStyle 为 0 时移去边框。在程序中可以使用 Hide 和 Show 方法，使表单在运行时透明或可见。

有关创建表单的其他内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第九章“创建表单”。

属 性

ActiveControl
Application
BaseClass
Caption
ClipControls
Comment
ControlCount
CurrentY
DataSessionID
DrawMode
Enabled
FontBold
FontItalic
FontShadow
FontUnderline
Height
Icon
LockScreen
MaxLeft
ActiveForm
AutoCenter
BorderStyle
Class
Closable
ContinuousScroll
Controls
DataEnvironment
DefOLELCID
DrawStyle
FillColor
FontCondense
FontName
FontSize
ForeColor
HelpContextID
KeyPreview
MaxButton
MaxTop
AlwaysOnTop
BackColor
BufferMode
ClassLibrary
ColorSource
ControlBox
CurrentX
DataSession
Desktop
DrawWidth
FillStyle
FontExtend
FontOutline
FontStrikeThru
HalfHeightCaption
HscrollSmallChange
Left
MaxHeight
MaxWidth

续 表

MDIForm	MinButton	MinHeight
MinWidth	MouseIcon	MousePointer
Movable	Name	Objects
OLEDragMode	OLEDragPicture	OLEDropEffects
OLEDropHasData	OLEDropMode	Parent
ParentClass	Picture	ReleaseType
RightToLeft	ScaleMode	ScrollBars
ShowTips	ShowWindow	SizeBox
TabIndex	TabStop	Tag
TitleBar	Top	ViewPortHeight
ViewPortLeft	ViewPortTop	ViewPortWidth
Visible	VscrollSmallChange	WhatsThisButton
WhatsThisHelp	WhatsThisHelpID	Width
WindowState	WindowType	ZoomBox

事 件

Activate	Click	DblClick
Deactivate	Destroy	DragDrop
DragOver	Error	GotFocus
Init	KeyPress	Load

续 表

LostFocus	MiddleClick Event	MouseDown
MouseMove	MouseUp	MouseWheel
Moved	OLECompleteDrag	OLEDragDrop
OLEDragOver	OLEGiveFeedBack	OLESetData
OLEStartDrag	Paint	QueryUnload
Resize	RightClick	Scrolled
Unload		

方 法

AddObject	AddProperty	Box
Circle	Cls	Draw
Hide	Line	Move
NewObject	OLEDrag	Point
Print	PSet	ReadExpression
ReadMethod	Refresh	Release
RemoveObject	ResetToDefault	SaveAs
SaveAsClass	SetAll	SetViewPort

续表

Show	ShowWhatsThis	TextHeight
TextWidth	WhatsThisMode	WriteExpression
WriteMethod	ZOrder	

请参阅

CREATE CLASS, CREATE FORM, DEFINE CLASS, FormSet 对象

Format 属性

指定某个控件的 Value 属性的输入和输出格式。

语法

Control.Format[= cFunction]

参数描述

cFunction

指定数据输入的限制条件和显示的格式。

编辑框的 *cFunction* 设置有：

设置	说明
K	只允许字母字符（不允许空格或标点符号）

微调器控件的 *cFunction* 设置有：

设置	说明
$	显示当前的 SET DATE 格式。
^	使用科学记数法显示数值型数据，只用于数值型数据。
K	当光标移动到文本框上时，选定整个文本框。
L	在文本框中显示前导零，而不是空格。只对数值型数据使用。
R	显示文本框的格式掩码，该掩码在文本框的 InputMask 属性中指定。掩码格式用于更方便和更清晰地显示（例如，如果掩码为 99-999，数字 12345 显示为 12–345），但是掩码并没有作为数据的一部分来储存。仅适用于字符型或数值型数据。
Z	如果为 0，显示为空值，除非控件处于作用中。

文本框和栏对象的 cFunction 设置有：

设置	说明
!	把字母字符转换为大写字母。只用于字符型数据，且只用于文本框。
$	显示货币符号，只用于数值型数据或货币型数据。

续表

^	使用科学记数法显示数值型数据，只用于数值型数据。
A	只允许字母字符（不允许空格或标点符号）。
D	使用当前的 SET DATE 格式。
E	以英国日期格式编辑日期型数据。
K	当控件具有焦点时选择所有文本。
L	在文本框中显示前导零，而不是空格。只对数值型数据使用。
M	包含向后兼容的功能。
R	显示文本框的格式掩码，该掩码在文本框的 InputMask 属性中指定。掩码格式用于更方便和更清晰地显示（例如，如果掩码为 99-999，数字 12345 显示为 12–345），但是掩码并没有作为数据的一部分来储存。仅适用于字符型或数值型数据。
T	删除输入字段前导空格和结尾空格。
YS	显示短日期格式的日期值，该日期格式是通过“Windows 控制面板”短日期设置定义的。
YL	显示长日期格式的日期值，该日期格式是通过“Windows 控制面板”长日期设置定义的。

说明

设计和运行时可用。Format 属性与 @ ... GET 和 @ ... EDIT 命令中 FUNCTION 子句作用相类似。Format 属性指定整个输入区域的特性。可以组合使用多个格式代码，它们对输入区域的所有输入都有影响。这个属性与 InputMask 属性形成对照，后者中每种输入掩码对应输入域中的一个输入项。

应用于

编辑框，微调，文本框

请参阅

DynamicInputMask 属性，InputMask 属性

FormCount 属性

包含表单集中表单对象的数目。设计时不可用，运行时只读。

语法

Object.FormCount

说明

可利用这个属性循环遍历表单集中的所有表单，并执行某些操作。

应用于

表单集，_SCREEN

请参阅

Forms 属性

Forms 属性

访问表单集中单个表单对象的数组。.

语法

Object.Forms(nIndex)

参数描述

nIndex

唯一标识表单集中的一个表单对象。

说明

设计时不可用，运行时只读。利用 Forms 属性，不使用表单的 Name 属性也可以更改表单集中表单的属性设置。可以把 Forms 属性与 FormCount 属性结合起来，遍历表单集中的所有表单，并执行某种操作。例如，下面的程序更改表单集中所有表单的标题：

```
FOR x = 1 TO THISFORMSET.FormCount
    THISFORMSET.Forms(x).Caption = THISFORMSET.Forms(x).Caption;
       + "[Read Only]"
ENDFOR
```

应用于

表单集，_SCREEN

请参阅

FormCount **属性**

FormSet **对象**

创建表单集。

语法

FormSet

说明

表单集是容器对象，可包含一组表单。表单集类似于 FoxPro **早期版本使用的屏幕集。有关创建表单集的其他内容，请参阅《**Microsoft Visual FoxPro 6.0 **中文版程序员指南》第九章“创建表单”中的“创建表单集”。**

属 性

ActiveForm	Application	AutoRelease
BaseClass	Buffermode	Class
ClassLibrary	Comment	DataEnvironment
DataSession	DataSessionID	FormCount
Forms	Name	Parent
ParentClass	ReadBackColor	ReadCycle
ReadForeColor	ReadLock	ReadMouse
ReadSave	ReadTimeOut	Tag
Visible	WindowList	WindowType

事 件

Activate	Deactivate	Destroy
Error	Init	Load
ReadActivate	ReadDeactivate	ReadShow
ReadValid	ReadWhen	Unload

方法

AddObject　AddProperty　Hide
NewObject　ReadExpression　ReadMethod
Refresh　Release　RemoveObject
ResetToDefault　SaveAs　SaveAsClass
SetAll　Show　WriteExpression
WriteMethod

请参阅

CREATE CLASS, CREATE FORM, DEFINE CLASS, Form 对象

FOUND() 函数

如果 CONTINUE、FIND、WDEXSEEK()、LOCATE **或** SEEK **命令成功，函数返回值为“真”**(.T.)。

语法

FOUND([nWorkArea | cTableAlias])

返值类型

逻辑值

参数描述

nWorkArea

指定表所在的工作区。FOUND() 函数的返回值指明此表最近一次的 CONTINUE、FIND、WDEXSEEK()、LOCATE 或 SEEK 命令是否执行成功。

如果在指定工作区中没有打开表，FOUND() 函数的返回值为“假”(.F.)。

cTableAlias

指定表的别名。FOUND() 函数的返回值指明此表最近一次的 CONTINUE、FIND、WDEXSEEK()、LOCATE 或 SEEK 命令是否执行成功。

如果所指定的表别名不存在，Visual FoxPro 将产生错误信息。

说明

FOUND() 函数返回一个逻辑值，指明最近执行的 CONTINUE、FIND、WDEXSEEK()、LOCATE 或 SEEK 命令是否成功，或指明记录指针在相关表中是否移动。如果搜索成功，FOUND() 函数的返回值为“真”(.T.)，否则为“假”(.F.)。

如果省略可选参数，FOUND() 函数的操作对象是当前选定工作区中打开的表，其返回值指明最近一次的 CONTINUE、FIND、WDEXSEEK()、LOCATE 或 SEEK 命令执行是否成功。

提示 这个函数可以用来判定子表是否有记录和父表的记录相匹配。

示例

在以下示例中，统计所有来自德国 (Germany) 的顾客 (customer)。

```
SET TALK OFF
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
```

```
USE customer  && 打开 Customer 表

STORE 0 TO gnCount
LOCATE FOR UPPER(country) = 'GERMANY'
DO WHILE FOUND()
    gnCount = gnCount + 1
    CONTINUE
ENDDO
WAIT WINDOW 'Total customers from Germany: ' ;
    + LTRIM(STR(gnCount)) NOWAIT
```

请参阅

CONTINUE, EOF(), FIND, INDEXSEEK(), LOCATE, SEEK

_FOXDOC 系统变量

为了向后兼容性而包含。

FPUTS() 函数

向低级文件函数打开的文件或通信端口写入字符串、回车符及换行符。

语法

FPUTS(nFileHandle , cExpression [, nCharactersWritten])

返值类型

数值型

参数描述

nFileHandle

对 FPUTS() 函数要写入数据的文件或通信端口指定文件句柄号。

cExpression

指定写入文件或端口的字符表达式。

nCharactersW ritten

指定写入文件或端口的 *cExpression* 中的字符数目。

如果省略 *nCharacterWritten* 选项，FPUTS() 函数将把整个字符表达式 *cExpression* 写入文件或端口；如果包含 *nCharacterWritten* 选项，则向文件或端口写入 *nCharacterWritten* 个字符。如果 *nCharacterWritten* 小于 *cExpression* 中的字符数目，只向文件或端口中写入前 *nCharacterWritten* 个字符；如果 *nCharacterWritten* 等于或大于 *cExpression* 中的字符数目，则 *cExpression* 中所有字符都将写入文件或端口。

说明

FPUTS()函数返回写入文件或端口的字节数。如果由于某种原因，FPUTS()函数不能向文件或端口写入数据，其返回值为0。

请参阅

FCHSIZE(), FCLOSE(), FCREATE(), FEOF(), FFLUSH(), FGETS(), FOPEN(), FREAD(), FSEEK(), FWRITE()

FREAD()函数

从低级文件函数打开的文件返回指定数目的字节。

语法

FREAD(nFileHandle, nBytes)

返值类型

字符型

参数描述

nFileHandle

FREAD() 要返回数据的文件的句柄号

nBytes

FREAD() 函数返回的字节数。FREAD() 从文件指针的当前位置开始，返回 *nBytes* 个字节的数据，或是遇到文件尾为止。

示例

以下示例使用了 FREAD() 函数显示文件中的内容。如果文件为空，显示一条消息。

```
* TEST.TXT must exist -- you can create this file
* 使用笔记本

Local gnFileHandle,nSize,cString
gnFileHandle = FOPEN("test.txt")
* 搜寻文件结尾，以确定文件的字节数
nSize =  FSEEK(gnFileHandle, 0, 2)    && 移动指针到文件头
IF nSize <= 0
    * If the file is empty, display an error message
    WAIT WINDOW "This file is empty!" NOWAIT
ELSE
    * 如果文件不为空，程序在内存中储存其内容
    * 然后在 Visual FoxPro 主窗口中显示文本
    = FSEEK(ggnFileHandle, 0, 0)    && 移动指针到文件尾
    cString = FREAD(gnFileHandle, nSize)
    ? cString
ENDIF
= FCLOSE(gnFileHandle)       && 关闭文件
```

请参阅

FCHSIZE(), FCLOSE(), FCREATE(), FEOF(), FFLUSH(), FGETS(), FILETOSTR(), FOPEN(), FPUTS(), FSEEK(), FWRITE()

FREE TABLE 命令

从一个表中删除数据库引用。

语法

FREE TABLE TableName

参数描述

TableName

指定表的名称，从该表中删除数据库引用。.

说明

如果偶然从磁盘上删除了一个数据库，该数据库中的表仍保留着对该数据库的引用。使用 FREE TABLE 可以从一个表中删除对数据库的引用，允许您打开该表，或者将该表添加到其他数据库中。

注意 如果一个数据库还在磁盘上，则一定不要使用 FREE TABLE 命令。如果一个数据库还在磁盘上，使用 FREE TABLE 会致使该数据库不可用。这时可使用 REMOVE TABLE 命令。与 FREE TABLE 不同，

REMOVE TABLE 从数据库中删除所有与表有关的主索引、默认值和有效性规则。

请参阅

ADD TABLE, CREATE DATABASE, OPEN DATABASE, REMOVE TABLE

FSEEK() 函数

在低级文件函数打开的文件中移动文件指针。

语法

FSEEK(nFileHandle, nBytesMoved [, nRelativePosition])

返值类型

数值型

参数描述

nFileHandle

指定文件句柄，FSEEK() 函数移动句柄所对应的文件的指针。可以在创建或打开文件时，由 FCREATE() 函数或 FOPEN() 函数返回句柄号。

nBytesMoved

指定文件指针移动的字节数。如果 *nByteMoved* 为正，则文件指针向文件尾移动；如果 *nByteMoved* 为负，文件指针向文件头移动。

nRelativePosition

在文件中把文件指针移动到某个相对位置。默认情况下，文件指针相对于文件头移动，也可以包含 *nRelativePosition* 使文件指针相对于文件指针的当前位置或文件尾移动。下表列出了 *nRelativePosition* 可能的取值及相对应的文件指针移动的起始位置。

NRelative Position	移动指针的相对起点
0	（默认值）文件头
1	文件指针当前位置
2	文件尾

说明

移动文件指针后，FSEEK() 函数返回从文件开始位置到文件指针位置的字节数。FREAD() 函数和 FWRITE() 函数也能够移动文件指针。

示例

下面的用户自定义函数使用 FSEEK() 返回一个文件的大小。如果不向这个用户自定义函数传递参数，则返回 –2。如果找不到该文件，则用户自定义函数返回 –1。

```
FUNCTION fsize2
PARAMETERS gcFileName  && 检查文件
PRIVATE pnHandle,pnSize
IF PARAMETERS() = 0
   RETURN -2  && 没有传递参数返回-2
```

```
ELSE
    IF !FILE(gcFileName)
        RETURN -1  && 文件不存在，则返回-1
    ENDIF
ENDIF
pnHandle = FOPEN(gcFileName)  && 打开文件
pnSize = FSEEK(pnHandle,0,2)  &&按照 pnSize 的标志决定文件大小
=FCLOSE(pnHandle) && 关闭文件
RETURN pnSize  && 返回值
```

请参阅

FCHSIZE(), FCLOSE(), FCREATE(), FEOF(), FFLUSH(), FGETS(), FOPEN(), FPUTS(), FREAD(), FWRITE()

FSIZE() 函数

以字节为单位，返回指定字段或文件的大小。

语法

FSIZE(cFieldName [, nWorkArea | cTableAlias] | cFileName)

返值类型

数值型

参数描述

cFieldName
指定字段名。

nWorkArea
指定表所在的工作区，FSIZE() 函数返回该表中某个字段的大小。

如果在指定工作区中没有打开的表，FSIZE() 函数的返回值为 0。

cTableAlias
指定表的别名，FSIZE() 函数返回该表中某个字段的大小。

如果指定的表别名不存在，Visual FoxPro 将产生错误信息。

cFileName
指定文件名，FSIZE() 函数以字节为单位返回其大小。

说明

SET COMPATIBLE 的当前设置决定了 FSIZE() 函数返回的是字段大小还是文件大小。如果 SET COMPATIBLE 设置为 OFF 或 FOXPLUS（默认值），则 FSIZE() 函数返回字段大小；如果 SET COMPATIBLE 设置为 ON 或 DB4，则 FSIZE() 函数返回文件大小。

下表列出了各种字段类型的默认大小（以字节为单位）。

字段类型	默认的字段大小（以字节为单位）
货币型	8
日期型	8
日期时间型	8
双精度型	8
整型	4
逻辑值	1
备注型	4
通用型	4

可以使用 DISPLAY STRUCTURE 或 LIST STRUCTURE 命令显示字段大小。

如果省略可选参数 *nWorkArea* 和 *cTableAlias*，FSIZE() 函数将返回当前表和工作区中字段的大小。

示例

以下示例使用 FSIZE() 函数返回 customer 表中两个字段的大小。

```
SET COMPATIBLE OFF
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer  && Open Customer table

CLEAR
? FSIZE('contact') && 显示数值 30
? FSIZE('cust_id') && 显示数值 6
```

请参阅

DISPLAY STRUCTURE, FCOUNT(), LIST

FTIME() 函数

返回最近一次修改文件的时间。

语法

FTIME(cFileName)

返值类型

字符型

参数描述

cFileName

要返回最近修改时间的文件名称，*cFileName*可在文件名前加入路径。如果在文件名前不包含路径，Visual FoxPro将在默认目录和SET PATH命令指定的目录下搜索文件。

说明

FTIME()**函数返回的时间值由操作系统指定给该文件。**

示例

以下示例使用 FTIME() 函数显示 FOXUSER.DBF 最后一次修改时间，此表为 Visual FoxPro 资源文件。

```
? FTIME('FOXUSER.DBF') && 显示上一次修改时间
```

请参阅

FDATE(), LUPDATE()

FullName 属性

包含 Visual FoxPro 的一个实例的文件名，以及启动 Visual FoxPro 的这个实例的目录名。运行时只读。

语法

ApplicationObject.FullName

说明

使用 FullName 属性，可以确定 Visual FoxPro 的一个实例的启动目录。.

应用于

Application 对象，_VFP 系统变量

请参阅

FULLPATH() 函数

返回指定文件的路径或相对于另一文件的路径。

语法

FULLPATH(cFileName1 [, nMSDOSPath | cFileName2])

返值类型

字符型

参数描述

cFileName1

指定 Visual FoxPro 要搜索的文件，应包含文件扩展名。

如果文件在 Visual FoxPro 路径上，则该路径随文件名一起返回。Visual FoxPro 路径可以用 SET PATH 命令来指定。

nMSDOSPath

搜索 MS-DOS 路径，而不是 Visual FoxPro 路径，*nMSDOSPath* 可取任何数值。如果文件不在 MS-DOS 路径上，将象在当前默认目录下找到文件一样返回路径及文件名。

cFileName2

指定要搜索的第二个文件名，应包含文件扩展名。FULLPATH() 函数将返回第一个文件相对于第二个文件的路径。

请参阅

DBF(), FILE(), LOCFILE(), SYS(2014)

Function 命令

定义一个用户自定义函数

语法

```
FUNCTION FunctionName
  Commands
  [RETURN [eExpression]]
ENDFUNC
```

参数描述

FunctionName

在 Visual FoxPro 中，函数名最长不超过 254 个字符。

如果要区分超过 10 个字符的程序文件名与以相同的 10 十个字符开头的函数，使用引号将程序文件名包围起来或在程序文件名后包含一个扩展名。

说明

许多程序中，都要经常重复运行若干例程。把公用例程定义为单独的函数可以减小程序的大小及其复杂性，也有利于程序的维护。

在程序中使用 FUNCTION *FunctionName* 语句，它指明程序中某个函数的开始，并用名称标识整个函数。

FUNCTION *FunctionName* 后面是组成该函数的一系列 Visual FoxPro 命令。可以一在函数的任何地方使用 RETURN 从函数返回调用程序或者另一个程序，并且返回一个由用户定义的函数值。如果没有 RETURN 命令，则当函数结束时自动执行隐式 RETURN 命令。如果 RETURN 命令不含返回值（或者如果执行隐式 RETURN 命令），将以真（.T.）作为返回值。

如果发出 DO 命令时包含某个函数名，Visual FoxPro 将按下列顺序搜索该函数：

1. Visual FoxPro 在包含 DO 命令的文件中搜索。
2. 如果在该文件中找不到此函数，Visual FoxPro 将在打开的过程文件中搜索。过程文件可通过 SET PROCEDURE 命令打开。
3. 如果在打开的过程文件中也找不到此函数，Visual FoxPro 将在执行链 (execution chain) 的各程序中搜索。首先搜索最近一次执行的程序，然后依次搜索，直到最开始执行的程序。
4. 如果仍未找到函数，Visual FoxPro 将搜索独立的程序。如果能够找到匹配的程序文件，则执行该程序，否则 Visual FoxPro 产生错误信息。

在 DO 命令中，包含 IN 子句可以执行指定文件中的函数。

默认情况下，参数以值传递方式传递给函数。有关以引用传递方式向函数传递参数的

详细内容，请参阅 SET UDFPARMS。最多可以向函数传递 27 个参数。这些参数可以传递给包含有 PARAMETERS 或者 LPARAMETERS 语句的函数，或者通过在函数姓名后的参数列表来传递。参数列表被包含在一个圆括号中，每一个参数用逗号隔开。

示例

此示例创建了一个名为 Hello 的自定义对象并且添加了一个名为 SayHello.的函数方法。SayHello 方法返回字符串“Hello World”，该字符串使用 MESSAGEBOX 函数显示。

注意：类定义代码放置在实例化对象的程序代码之后。

```
Local oHello
oHello=CREATEOBJECT("Hello")
=MESSAGEBOX(oHello.SayHello(),48)
RELEASE oHello

* Class definition code
DEFINE CLASS Hello AS CUSTOM
    FUNCTION SayHello
        RETURN "Hello World"
    ENDFUNC
ENDDEFINE
```

请参阅

LPARAMETERS, PARAMETERS, PARAMETERS(), PRIVATE, PROCEDURE, PUBLIC, RETURN, SET PROCEDURE, SET UDFPARMS

FV() 函数

返回一笔金融投资的未来值。

语法

FV(nPayment, nInterestRate, nPeriods)

返值类型

数值型

参数描述

nPayment

指定固定的周期性付款金额（可以为正数或负数）。

nInterestRate

指定的周期利率。如果已知年利率，但是每月付款一次，则 *nInterestRate* 等于年利率除以 12。

nPeriods

指定已经付款的周期数，FV() 函数假定周期性付款在每个周期末进行。

说明

FV() 函数按照不变复利计算一系列固定的周期性付款的未来值，未来值是所有已付款

与利息之和。

示例

```
STORE 500 TO gnPayment  && Monthly payment
STORE .075/12 TO gnInterest    && 7.5%每年的利率
STORE 48 TO gnPeriods  && 四年 (48月)

CLEAR
? FV(gnPayment, gnInterest, gnPeriods)   && 显示数值 27887.93
```

请参阅

CALCULATE, PV()

FWRITE() 函数

向低级文件函数打开的文件写入字符串。

语法

FWRITE(nFileHandle , cExpression [, nCharactersWritten])

返值类型

数值型

参数描述

nFileHandle
要写入字符串的文件的句柄号。
cExpression
指定 FWRITE() 函数写入的字符表达式。
nCharactersWritten
FWRITE() writes the entire character expression to the file unless you include 除非包含 *nCharacterWritten*，否则 FWRITE() 函数向文件写入整个字符表达式。当包含 *nCharacterWritten* 时，向文件写入 *nCharacterWritten* 个字符。如果 *nCharacterWritten* 小于 *cExpression* 中字符的数目，只向文件写入 *nCharacterWritten* 个字符；如果 *nCharacterWritten* 等于或大于 *cExpression* 中字符的数目，将把 *cExpression* 中的所有字符都写入文件。

说明

与 FPUTS() 函数不同，FWRITE() 函数并不在字符串的尾部放置回车和换行符。FWRITE() 函数返回向文件写入的字节数。如果由于某种原因，FWRITE() 函数不能向该文件写入数据，则函数的返回值为 0。.

请参阅

FCHSIZE(), FCLOSE(), FCREATE(), FEOF(), FFLUSH(), FGETS(), FOPEN(), FPUTS(), FREAD(), FSEEK(), STRTOFILE()

_GALLERY 系统变量

指定当选择“工具”菜单中的“组件管理库”命令时执行的程序。

语法

_GALLERY = ProgramName

参数描述

ProgramName

指定一个程序。如果您的程序不在当前默认目录中，在程序名中应该包含路径。

说明

在默认情况下，_GALLERY 包含 GALLERY.APP。请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第三十二章“应用程序的开发和开发者的生产率”中的“组件管理库”。

GATHER 命令

将当前选定表中当前记录的数据替换为某个数组、变量组或对象中的数据。

语法

GATHER FROM ArrayName | MEMVAR | NAME ObjectName
[FIELDS FieldList | FIELDS LIKE Skeleton | FIELDS EXCEPT Skeleton]
[MEMO]

参数描述

FROM ArrayName

指定一个数组，用它的数据替换当前记录中的数据。从数组的第一个元素起，各元素的内容依次替换记录中相应字段的内容。第一个数组元素的内容替换记录第一个字段的内容，第二个数组元素内容替换记录第二个字段的内容，依此类推。

如果数组的元素少于表的字段数目，则忽略多余的字段。如果数组的元素多于表的字段数目，则忽略多余的数组元素。

MEMVAR

指定一组变量或数组，把其中的数据复制到当前记录中。变量的数据将传送

给与此变量同名的字段。如果没有与某个字段同名的变量，则不替换此字段。

提示 在 SCATTER 命令中包含 MEMVAR 或 BLANK 子句，可以创建与字段同名的变量。

NAME ObjectName

指定某个对象，其属性与表的字段同名。每个字段的内容分别替换为与字段同名的属性的值。如果没有与某个字段同名的属性，则此字段的内容不做替换。

FIELDS FieldList

指定用数组元素或变量的内容替换其内容的字段。只替换在 *FieldList* 中指定的字段的内容。

FIELDS LIKE Skeleton | FIELDS EXCEPT Skeleton

选用 LIKE 子句或 EXCEPT 子句，或者同时包含以上两个子句，可以有选择地将字段内容替换为数组元素或变量的内容。如果包含 LIKE *Skeleton* 子句，Visual FoxPro 将替换与 *Skeleton* 匹配的字段；如果包含 EXCEPT *Skeleton* 子句，Visual FoxPro 将替换与 *Skeleton* 不匹配的所有字段。

Skeleton 支持通配符（* 和 ?）。例如，要替换所有以字母 A 和 P 开头的字段，可使用下列命令：

```
GATHER FROM gamyarray FIELDS LIKE A*,P*
```

MEMO

指定用数组元素或变量的内容替换备注字段的内容。如果省略 MEMO 子

句，则在用数组或变量的内容替换字段内容时，GATHER 命令将跳过备注字段。即使包含了 MEMO 关键字，GATHER 命令也忽略通用字段和图片字段。

示例

示例 1

此示例使用 GATHER 命令将数据复制到表的新记录中。在创建 Test 之后，使用 SCATTER 命令创建一套基于表中字段的变量。然后对每个字段赋值并向表中添加一个空记录。

```
CREATE TABLE Test FREE ;
    (Object C(10), Color C(16), SqFt n(6,2))
SCATTER MEMVAR BLANK
m.Object="Box"
m.Color="Red"
m.SqFt=12.5
APPEND BLANK
GATHER MEMVAR
BROWSE
```

示例 2

此示例使用了 GATHER 命令及 NAME 子句将数据复制到表的新记录。在创建表 Test 之后，使用 SCATTER 命令创建具有基于表中字段属性的对象。然后为对象的属性赋值并向表添加一个空记录。

```
CREATE TABLE Test FREE ;
    (Object C(10), Color C(16), SqFt n(6,2))
```

```
SCATTER NAME oTest BLANK
oTest.Object="Box"
oTest.Color="Red"
oTest.SqFt=12.5
APPEND BLANK
GATHER NAME oTest
RELEASE oTest
BROWSE
```

请参阅

APPEND FROM ARRAY, COPY TO ARRAY, DIMENSION, SCATTER

_GENGRAPH 系统变量

指定应用程序，该程序用于向 Microsoft Graph 中输出查询结果。包含此变量是为了提供向后兼容性。可用图形向导代替。

_GENHTML 系统变量

指定一个 HTML (Hypertext Markup Language) **生成程序。**

语法

_GENHTML = ProgramName

参数描述

ProgramName

指定生成 HTML **的程序。如果所指定的** HTML **生成程序不在当前默认目录中，在程序名中应该包含路径。**

说明

在默认情况下，_GENHTML **包含** Genhtml.prg**。当从“文件”菜单中选择“另存为** HTML**”命令时，会执行** Genhtml.prg**。并且创建一个包含** HTML **版本表单、报表或表的文本文件。只有当表单、菜单或报表设计器活动时，并且表单、菜单或报表已经保存了，“另存为** HTML**”选项才可用，或者当打开一个浏览窗口时“另存为** HTML**”选项才可用。**

请参阅

CREATE FORM, CREATE MENU, CREATE REPORT, File menu, USE

_GENMENU 系统变量

指定菜单生成程序。

语法

_GENMENU = ProgramName

参数描述

ProgramName

指定用来生成菜单代码的先前版本的程序。该程序从 .MNX 菜单定义表中生成菜单代码。如果此程序不在当前默认目录下，应该在程序名前包含路径。

说明

默认情况下，_GENMENU 中包含 GENMENU.PRG。

请参阅

CREATE MENU, _GENSCRN

_GENPD 系统变量

为 FoxPro for MS-DOS 中创建的基于字符的报表指定打印机驱动接口程序。包含此变量是为了提供向后兼容性。在 REPORT 中使用 TO FILE ASCII 参数。.

_GENSCRN 系统变量

指定表单生成程序。包含此变量是为了提供向后兼容性。可用表单设计器代替。

_GENXTAB 系统变量

指定用来以交叉表格式输出查询结果的程序。包含此变量是为了提供向后兼容性。可用交叉表向导代替。

GETBAR() 函数

返回用 DEFINE POPUP 命令定义的菜单或 Visual FoxPro 系统菜单上某个菜单项的编号。

语法

GETBAR(MenuItemName, nMenuPosition)

返值类型

数值型

参数描述

MenuItemName

指定菜单项。

nMenuPosition

指定菜单上的某一位置。*nMenuPosition* 取值范围为 1 到菜单中菜单项的数目。1 与菜单中的第一个菜单项相对应，而 2 与第二个菜单项相对应，依此类推。

说明

使用 GETBAR() 函数可以确定菜单特定位置上的菜单项。当在菜单中添加、删除或重新安排菜单项时，此函数很有用。使用 DEFINE BAR 命令可以向菜单中添加菜单项，用 RELEASE BAR 命令可从菜单中删除菜单项。用 DEFINE POPUP 命令创建菜单

时，如果包含 MOVER 子句，则可以更改菜单项在菜单中的位置。

示例

以下示例创建了一个名为 popDemo 的菜单。包含 MOVER 关键字是为了使菜单中的数据项可以重新组织。有关重新组织菜单项的内容，请参阅稍前部分的语言参考“DEFINE POPUP 命令中的 MOVER 子句”。

激活菜单，使用一系列 GETBAR() 函数以返回每个菜单的标题。在重新组织了菜单项之后，按下 CTRL+Z 键以显示新的菜单顺序。

```
CLEAR
ON KEY LABEL CTRL+Z DO showorder
WAIT WINDOW "Press CTRL+Z to refresh." NOWAIT

DEFINE POPUP popDemo MOVER FROM 2,2
DEFINE BAR 1 OF popDemo PROMPT 'One'
DEFINE BAR 2 OF popDemo PROMPT 'Two'
DEFINE BAR 3 OF popDemo PROMPT 'Three'
DEFINE BAR 4 OF popDemo PROMPT 'Four'

DO showorder
ACTIVATE POPUP popDemo

PROCEDURE showorder
CLEAR
@ 3,12 SAY  '1 ' + PRMBAR('popDemo', GETBAR('popDemo',1))
@ 4,12 SAY  '2 ' + PRMBAR('popDemo', GETBAR('popDemo',2))
@ 5,12 SAY  '3 ' + PRMBAR('popDemo', GETBAR('popDemo',3))
@ 6,12 SAY  '4 ' + PRMBAR('popDemo', GETBAR('popDemo',4))
RETURN
```

请参阅

DEFINE BAR, DEFINE POPUP, RELEASE BAR

GETCOLOR() 函数

显示 Windows 的“颜色”对话框，并返回选定颜色的颜色编号。

语法

GETCOLOR([nDefaultColorNumber])

返值类型

数值型

参数描述

nDefaultColorNumber

指定显示“颜色”对话框时初始选定的颜色。如果 *nDefaultColorNumber* 不与“颜色”对话框中的任何颜色对应，则选择“颜色”对话框中的第一种颜色。如果省略 *nDefaultColorNumber* 参数，则选择黑色。

说明

如果按 ESC 键、选择“取消”按钮，或从“控制”菜单中选择“关闭”，退出“颜色”对话框时，GETCOLOR() 函数返回 -1。

示例

以下示例显示了 Windows “颜色”对话框，您可以从中选取颜色。当您退出对话框时，显示与您选择的颜色相对应的数字。

```
CLEAR
? GETCOLOR(255)
```

请参阅

GETPICT(), RGB()

GETCP() 函数

显示“代码页”对话框，提示输入代码页，然后返回选定代码页的编号。

语法

GETCP([nCodePage] [, cText] [, cDialogTitle])

返值类型

数值型

参数描述

nCodePage

显示“代码页”对话框时，指定初始选择的代码页编号。如果 *nCodePage* 等于 0，或者省略 *nCodePage*，当显示“代码页”对话框时，没有选定代码页。

cText

指定显示在“代码页”对话框中的文本。如果省略参数 cText，Visual FoxPro 显示以下文本“请选择代码页供跨平台数据共享”。

cDialogTitle

指定“代码页”对话框标题栏上出现的标题。如果省略 *cDialogTitle*，则显示“代码页”。

说明

如果按 ESC 键、选择“取消”按钮，或从“控制”菜单上选择“关闭”项退出“代码页”对话框，GETCP() 函数返回 0。

在“代码页”对话框中列出的代码页由 FOXPRO.INT 决定，它是 Visual FoxPro 国际代码页支持文件。

可以在诸如 MODIFY COMMAND、APPEND FROM 和 COPY TO 等支持 AS *nCodePage* 子句的命令中包含 GETCP() 函数，此时将显示“代码页”对话框，并允许您为打开、追加或创建的文件指定代码页。因为不存在编号为 0 的代码页，所以在用户按 ESC 键、选择“取消”按钮或从“控制”菜单上选择“关闭”时，必须俘获 0 返回值。

示例

以下示例显示选取代码页为 1252 (Windows ANSI) 的“代码页”对话框。在“代码

页”对话框中显示“选择代码页”作为标题。在“代码页”标题栏中显示“代码页选取”。

```
? GETCP(1252, "Select a Code Page", "Code Page Selection")
```

请参阅

APPEND FROM, COPY TO, EXPORT, IMPORT, MODIFY COMMAND, MODIFY FILE, MODIFY QUERY

GetData 方法

从 OLE 拖放 DataObject 对象获取数据。只在设计时可用。.

语法

oDataObject.GetData(nFormat | cFormat [, @ ArrayName])

参数描述

nFormat | cFormat

指定要获取数据的格式。下表列出了每种数据格式的值以及说明。 DataObject **自动支持下列格式（还有其他格式可用，但是需要额外的编程）。有关可用数据格式的详细内容，请在** MSDN(Microsoft Developer

Network)中查看有关 Visual C++ 的文档。

数据格式*	nFormat | Format	说明
CF_TEXT	1	文本格式。
CF_OEMTEXT	7	包含 OEM 字符集中字符的文本格式。
CF_UNICODETEXT	13	Unicode 文本格式，只在 Windows NT 下可用。
CF_FILES or CF_HDROP	15	一个标识一组文件的句柄，例如从 Windows 资源管理器拖来的一组文件。
CF_LOCALE	16	与剪贴板上文本相关的本地标识符的句柄。
CFSTR_OLEVARIANTARRAY	“OLE Variant Array”	一个 Visual FoxPro 数组。使用这个格式在一次拖放中可以传送多个值。例如，这个格式可以用于将列表框中的一些项拖动到另一个列表框中。.

续表

CFSTR_OLEVARIANT	“OLE Variant”	一个 Visual FoxPro 变量。Visual FoxPro 中的所有数据类型都可以用变量代表。这个格式可以用于拖放 Visual FoxPro 数据，并且不丢失数据类型。
CFSTR_VFPSOURCEOBJECT	“VFP Source Object”	对一个 Visual FoxPro 对象的引用。

*在 FOXPRO.H 中定义。.

@ArrayName

指定数组的名称，当数据可以包含多个值时，将数据保存在这里。数据可以包含多个值的数据格式只有 CF_FILES、CF_HDROP 和 CFSTR_OLEVARIANTARRAY。例如，可以从 Windows 资源管理器中将一组文件拖动到 Visual FoxPro 的一个列表框中。使用列表框 OLEDragDrop 事件中的 GetData 方法，可以将这些文件的名称放到一个数组，然后在 FOR ... ENDFOR 循环中使用 AddItem 方法将数组中的内容添加到列表框中。

在 GetData 方法中指定数组名称时，该数组必须存在。如果该数组存在，但是不足以包含所需的数据，Visual FoxPro 自动增加该数组的大小。如果该数组比所需大，Visual FoxPro 会截断该数组。

说明

GetData 方法返回的值取决于 nFormat 或 cFormat 指定的数据格式。如果

DataObject 不包含在 nFormat 或 cFormat 指定的数据格式的数据，则返回“假”(.T.)。如果数据符合多值格式，例如 CF_FILES、CF_HDROP 或 CFSTR_OLEVARIANTARRAY，则返回“真”(.T.)。如果数据符合单值格式时，例如 CF_TEXT、CFSTR_OLEVARIANT 或 CFSTR_VFPSOURCEOBJECT，则返回 DataObject 中的值。.

如果使用 nFormat 或 cFormat 指定的数据格式存在，但是在 DataObject 中没有这种格式的数据，则触发一个拖动源的 OLESetData 事件。（在使用 SetData 方法将相应数据放在 DataObject 中之前，可以使用 SetFormat 方法指定一种数据格式。）

应用于

DataObject 对象

请参阅

ClearData 方法, GetFormat 方法, OLE drag-and-drop 概览, OLESetData 事件, SetData 方法, SetFormat 方法

GETDIR() 函数

显示“选择目录”对话框，从中可以选择目录或文件夹。

语法

GETDIR([cDirectory [, cText]])

返值类型

字符型

参数描述

cDirectory

指定在“选择目录”对话框中初始显示的目录。如果不指定 *cDirectory*，“选择目录”对话框打开时，将显示 Visual FoxPro 默认目录。.

cText

指定“选择目录”对话框中目录列表的文本。在 Windows 3.1 中，该文本显示为大会框标题栏中的标题。在 Windows 95 中，该文本显示在对话框中标题栏下面。

说明

GETDIR() 函数返回字符串，其内容为选定目录的名称。

如果没有选择目录（选择“取消”、按 ESC 键或从“窗口”菜单上选择“关闭”），GETDIR() 函数将返回空字符串。

请参阅

DIRECTORY, DIRECTORY(), GETEXPR, GETFILE()

GETENV() 函数

返回指定的 MS-DOS 环境变量的内容。

语法

GETENV(cVariableName)

返值类型

字符型

参数描述

cVariableName

指定 MS-DOS 环境变量的名称。如果所指定的 MS-DOS 环境变量不存在，则返回空字符串。

指定 MS-DOS 环境变量的名称。如果所指定的 MS-DOS 环境变量不存在，则返回空字符串。.

说明

有两个环境变量总是可用的：COMSPEC 和 PATH。您可以使用 MS-DOS 的 SET 命令创建自己的环境变量。

有关环境变量的其他内容，请参阅 MS-DOS 手册。

示例

```
CLEAR
? GETENV('PATH') && 显示 MS-DOS 路径
```

请参阅

DISKSPACE(), OS(), VERSION()

GETEXPR 命令

显示表达式生成器对话框，从中可以创建表达式并把此表达式存储在变量或数组元素中。

语法

```
GETEXPR [cCaptionText] TO MemVarName
  [TYPE cExpressionType [; cErrorMessageText]]
  [DEFAULT cDefaultExpression]
```

参数描述

cCaptionText

指定在“表达式生成器”中显示的标题。该标题可提示用户生成何种类型的表达式。

TO MemVarName

指定存储表达式的变量或数组元素。如果此变量不存在，Visual FoxPro 将创建该变量。GETEXPR 命令不创建数组元素。

如果按 ESC 键或选择“取消”按钮退出“表达式生成器”，将在变量或数组元素中存储空字符串。如果已经用 DEFAULT 子句创建了某个默认表达式，在按 ESC 键或选择“取消”按钮退出“表达式生成器”时，将把这个默认表达式存储在变量中。.

TYPE cExpressionType [; cErrorMessageText]

指定表达式类型。下表列出了在 *cExpressionType* 中用于指定各表达式类型的字符：

cExpressionType	表达式类型
C	字符型
D	日期型
T	日期时间型
N	数值型
F	浮点型
I	整型
B	双精度型
Y	货币型
L	逻辑值

可以指定当表达式无效时显示的错误信息 *cErrorMessageText*。如果同时包含 *cErrorMessageText* 和 *cExpressionType*，应该用分号（；）将它们分隔开。*cExpressionType*、分号以及 *cErrorMessageText* 的组合应该用成对的单引号或双引号括起来。

DEFAULT cDefaultExpression

指定在“表达式生成器”中显示初始的默认表达式。可以接受这个默认表达式，或用 *cDefaultExpression* 指定的表达式改写它。如果按 ESC 键或选择“取消”按钮退出“表达式生成器”，*cDefaultExpression* 的值将存储在变量或数

组元素中。

示例

下面的示例将显示“表达式生成器”，提示您输入相应的内容：

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer  && 打开 Customer 表

GETEXPR 'Enter condition to locate ' TO gcTemp;
    TYPE 'L' DEFAULT 'COMPANY = ""'
LOCATE FOR &gcTemp
IF FOUND()
    DISPLAY
ELSE
    ? 'Condition ' + gcTemp + ' was not found '
ENDIF
```

请参阅

_GETEXPR, GETFILE(), GETPICT(), LOCFILE(), PUTFILE()

_GETEXPR 系统变量

指定当发出 GETEXPR 命令时，或从 Visual FoxPro 中击活“表达式生成器”对话框时执行的程序

语法

_GETEXPR = ProgramName

参数描述

ProgramName

指定当发出 GETEXPR 命令时，或从 Visual FoxPro 中击活“表达式生成器”对话框时执行的程序。如果您的程序不在当前默认目录中，在程序名中应该包含路径。

说明

在默认情况下，_GETEXPR 包含空字符串；空字符串表明当发出 GETEXPR 命令时，或从 Visual FoxPro 中击活“表达式生成器”对话框时显示标准的 Visual FoxPro “表达式生成器”对话框。

您也可以创建自己的“表达式生成器”程序，当发出 GETEXPR 命令时，或从 Visual FoxPro 中击活“表达式生成器”对话框时就执行该程序。您的“表达式生成器”程序在第一行可执行代码行包含一个 LPARAMETERS 或 PARAMETERS 语句，以接受 Visual FoxPro 传递给该程序的四个参数。

下表按传递顺序列出了这些参数：

参数	说明
CExpression Type	指定表达式类型。

续表

CErrorMessageText	指定如果表达式无效所显示的错误信息。
CDefaultExpression	指定表达式生成器中的默认初始表达式。
cCaptionText	指定表达式生成器中显示的标题。

例如，您的表达式生成器程序可以将以下内容作为第一行可执行语句：

```
LPARAMETERS cExpressionType, cErrorMessageText, ;
    cDefaultExpression, cCaptionText
```

如果从 Visual FoxPro 中打开“表达式生成器”对话框时，执行您的表达式生成器程序，前三个参数包含空字符串，第四个参数包含 cCaptionText，即表达式生成器中显示的标题。

注意：Visual FoxPro 表达式生成器是一个模式对话框。为了创建一个模式对话框，您的表达式生成器程序应该按以下值设置它的表单属性：

表单属性	属性值
AlwaysOnTop	“真” (.T.)
Desktop	“真” (.T.)
WindowType	1 – Modal

请参阅

GETEXPR

GETFILE() 函数

显示“打开”对话框，并返回选定文件的名称。

语法

GETFILE([cFileExtensions] [, cText] [, cOpenButtonCaption]
[, nButtonType] [, cTitleBarCaption])

返值类型

字符型

参数描述

cFileExtensions

指定没有选择“所有文件”菜单项时，可滚动列表中显示的文件扩展名。

cFileExtensions 可具有多种形式：

- 如果 *cFileExtensions* 包含单一扩展名（例如 .PRG），只显示具有此扩展名的文件。
- 如果 *cFileExtensions* 为空字符串，并且不包含 *cCreatorType* 子句，则显示当前目录下的所有文件。
- *cFileExtensions* 也可以包含通配符（* 和 ?）。此时将显示扩展名满足通配符条件的所有文件。例如，如果 *cFileExtensions* 为 '?X?'，则显示扩展名。

- 在 Visual FoxPro for Windows 中，cFileExtensions 可以包含一个文件说明，后面带有一个或一列用逗号分隔的扩展名。这个文件说明出现在“文件类型”列表框中。使用一个冒号 (:) 将文件说明和扩展名分开。使用分号 (;) 将多个文件说明和它们的扩展名分开。
  例如，如果 cFileExtensions 是 "Text:TXT"，则文件说明 "Text" 出现在“文件类型”列表框中，而且显示所有具有 .txt 扩展名的文件。
  如果 cFileExtensions 是 "Tables:DBF; Files:TXT,BAK"，则文件说明 "Tables" 和 "Files" 出现在“文件类型”列表框中。当从“文件类型”列表框中选择 "Tables" 时，则显示所有具有 .dbf 扩展名的文件。当从“文件类型”列表框中选择 "Files" 时，则显示所有具有 .txt 和 .bak 扩展名的文件。.
- 如果 *cFileExtensions* 只包含分号 (;)，则显示所有不带扩展名的文件

cText

指定“打开”对话框中目录列表的文本。在 Windows 95 中，该文本显示在文件列表下面，而且长文本字符串可能会被截短。

cOpenButtonCaption

为“确定”按钮指定标题。

nButtonType

指定出现在“打开”对话框中按钮的数目与类型。当 *nButtonType* 等于 0、1 或 2 时，在对话框中分别出现下列按钮。

nButtonType	按钮
0	确定
(或省略)	取消
1	确定
	新建
	取消
2	确定
	无
	取消

如果 *nButtonType* 等于 1，而用户选择了“新建”按钮，此函数返回在“打开”对话框中指定的路径与“尚未命名”。如果 *nButtonType* 等于 2 而用户选定了“无”按钮，函数返回空字符串。

cTitleBarCaption

指定标题栏标题。

说明

如果按 ESC 键、选择“取消”按钮，或者从**控制**菜单上选择“关闭”退出“打开”对话框，GETFILE() 函数将返回空字符串。

示例

```
CLOSE DATABASES
SELECT 0

gcTable = GETFILE('DBF', 'Browse or Create a .DBF:', 'Browse', 1;
    'Browse or Create')
DO CASE
    CASE 'Untitled' $ gcTable
        CREATE (gcTable)
    CASE EMPTY(gcTable)
        RETURN
    OTHERWISE
        USE (gcTable)
        BROWSE
ENDCASE
```

请参阅

FULLPATH(), GETEXPR, GETPICT(), LOCFILE(), PUTFILE()

GETFLDSTATE() 函数

返回一个数值，标明表或临时表中的字段是否已被编辑，或是否有追加的记录，或者指明当前记录的删除状态是否已更改。

语法

GETFLDSTATE(cFieldName | nFieldNumber [, cTableAlias | nWorkArea])

返值类型

数值型

参数描述

cFieldName | nFieldNumber

指定要返回其编辑状态的字段的名称或编号。字段编号 *nFieldNumber* 对应于字段在表或临时表结构中的位置。可以使用 DISPLAY STRUCTURE 命令或 FIELD() 函数确定字段的编号。

指定 *nFieldNumber* 为 -1 可以返回一个字符串，该字符串包含表或临时表中所有字段的删除和编辑状态。例如，如果某个表有五个字段，而只编辑过第一个字段，则 GETFLDSTATE() 函数将返回下列值：

121111

第一个位置上的 1 表明删除状态尚未更改。

也可以把 *nFieldNumber* 设置为 0，从而确定自表或临时表打开以来是否更改过当前记录的删除状态。

注意 GETFLDSTATE() 函数只能用于判定是否更改过当前记录的删除状态。例如，如果对某个记录作了删除标记，然后又进行了恢复操作，即使此时该记录的删除状态已恢复为原始状态，GETFLDSTATE() 函数仍将指示删除状态已进行了更改。可以使用 DELETED() 函数确定记录当前的删除状态。

cTableAlias

指定表或者临时表的别名，函数将返回其字段的编辑状态或记录的删除状态。

nWorkArea

指定表或者临时表所在的工作区，函数将返回其字段的编辑状态或记录的删除状态。

如果没有指定别名或工作区，则 GETFLDSTATE() 函数返回当前选定表或者临时表中字段的状态。.

说明

下表列出了返回值及其相应的编辑或删除状态。

返回值	编辑或删除状态
1	字段未作编辑，或者删除状态未作更改。
2	已编辑了字段，或者更改了删除状态。.
3	追加记录的字段未做编辑，或者追加记录的删除状态未做更改。.
4	已编辑了追加记录的字段，或者已更改了追加记录的删除状态。

必须首先用 CURSORSETPROP() 函数启用行缓冲或表缓冲，才能使 GETFLDSTATE() 函数操作本地表。

如果 GETFLDSTATE() 函数未带可选的参数 *cTableAlias* 或 *nWorkArea*，则返回当前选定工作区中打开的表或临时表的编辑或删除状态。

示例

下面的示例演示了如何利用 GETFLDSTATE() 函数判定字段的内容是否已进行了更改。

为满足表缓冲的要求，将 MULTILOCKS 设置为 ON。打开 testdata 数据库中的 customer 表，然后用 CURSORSETPROP() 函数将缓冲方式设置为开放式表缓冲 (5)。

在 cust_id 字段修改之前发出 GETFLDSTATE() 函数，显示与此字段的未修改状态相对应的值 (1)。用 REPLACE 命令修改 cust_id 字段后，再次发出 GETFLDSTATE()函数，显示与此字段的修改状态相对应的值 (2)。之后调用 TABLEREVERT() 函数使表返回到原始状态，并且再次发出 GETFLDSTATE() 函数，显示与 cust_id 字段原始状态相对应的值 (1)。

```
CLOSE DATABASES
CLEAR

SET MULTILOCKS ON       && 允许表缓冲
OPEN DATABASE (HOME(2) + 'data\testdata')
USE Customer            && 打开 customer 表
=CURSORSETPROP("Buffering",5,"customer")  && 启用表缓冲

* 获取原始 cust_id 字段上的字段状态并显示状态
nState=GETFLDSTATE("cust_id")
DO DisplayState WITH nState

* 更改字段内容并显示状态
REPLACE cust_id   WITH "***"
nState=GETFLDSTATE("cust_id")
DO DisplayState WITH nState
```

```
*放弃表的更改并显示状态
= TABLEREVERT(.T.)      &&放弃所有的表更改
nState=GETFLDSTATE("cust_id")
DO DisplayState WITH nState

PROCEDURE DisplayState
PARAMETER nState
DO CASE
    CASE nState=1
        =MESSAGEBOX("Field has not been modified",0,"Results")
    OTHERWISE
        =MESSAGEBOX("Field has been modified",0,"Results")
ENDCASE
```

请参阅

CURVAL(), DELETED(), FIELD(), OLDVAL(), CURSORSETPROP(), SETFLDSTATE()

GETFONT() 函数

显示“字体”对话框，并返回所选定字体的名称。

语法

GETFONT(cFontName [, nFontSize [, cFontStyle)

返值类型

字符型

参数描述

cFontName

指定在“字体”对话框中初始选择的字体名称。如果没有安装您指定的字体，则初始选择默认的字体。

nFontSize

指定在“字体”对话框中初始选择的字体大小。如果不支持所指定的字体大小，则初始选择默认的字体大小。如果小于等于零，或者省略 nFontSize，则默认的字体大小为 10 磅。

cFontStyle

指定在“字体”对话框中初始选择的字型。如果不支持所指定的字型，则初始选择默认的字型。cFontStyle 是指定字型的一个或两个字符。下表列出了可以为 cFontStyle 指定的字符及相应的字型。

字符	字型
B	粗体
I	斜体
BI	粗斜体

说明

GETFONT() 返回所选择字体的名称、大小和字型。您的选择返回为一个字符串，字体的名称、大小和字型用逗号分隔。

如果通过选择“取消”按钮、从“控制”菜单选择了“关闭”命令，或者按了 ESC 键，则 GETFONT() 返回空字符串。

Cmyfont = GETFONT(—，，‘B’)

注意 Visual FoxPro 命令和函数可以截断为四个字符。而对于 GETFONT() 和 GETFILE()，优先权在 GETFILE()。发出 GETF()，会显示“打开”对话框。

请参阅

FONTMETRIC(), SYSMETRIC(), TXTWIDTH(), WFONT()

GetFormat 方法

确定在 OLE 拖放 DataObject 中指定格式的数据是否可用。只在设计时可用。

语法

oDataObject.GetFormat(nFormat | cFormat)

参数描述

nFormat | cFormat

指定要获取数据的格式。下表列出了每种数据格式的值以及说明。

DataObject 自动支持下列格式（还有其他格式可用，但是需要额外的编程）。有关可用数据格式的详细内容，请在 Microsoft Developer Network 中查看有关 Visual C++ 的文档。通过为指定唯一的字符串，也可以创建自己的格式。

数据格式*	nFormat | cFormat	说明
CF_TEXT	1	文本格式。.
CF_OEMTEXT	7	包含 OEM 字符集中字符的文本格式。
CF_UNICODETEXT	13	Unicode 文本格式，只在 Windows NT 下可用。

续表

CF_FILES or CF_HDROP	15	一个标识一组文件的句柄，例如从 Windows 资源管理器拖来的一组文件。
CFSTR_OLEVARIANTARRAY	“OLE Variant Array”	一个 Visual FoxPro 数组。使用这个格式在一次拖放中可以传送多个值。例如，*这个格式可以用于将列表框中的一些项拖动到另一个列表框中*。
CFSTR_OLEVARIANT	“OLE Variant”	一个 Visual FoxPro 变量。Visual FoxPro 中的所有数据类型都可以用变量代表。这个格式可以用于拖放 Visual FoxPro 数据，并且不丢失数据类型。
CFSTR_VFPSOURCEOBJECT	“VFP Source Object”	对一个 Visual FoxPro 对象的引用。

*在 FOXPRO.H 中定义。

说明

如果 DataObject 包含在 nFormat 或 cFormat 中指定格式的数据，则 GetFormat 方法返回“真” (.T.)；否则返回“假” (.T.)。

应用于

DataObject 对象

请参阅

ClearData 方法, GetData 方法, OLE drag-and-drop 概览, SetData 方法, SetFormat 方法

GETHOST() 函数

返回对一个 Active Document 容器的对象引用。

语法

GETHOST()

返值类型

对象

说明

使用 GETHOST() 确定一个 Active Document 的容器。该对象引用的 Name 属性可以用

来确定容器的名称（例如 Microsoft Internet Explorer）。

如果不能确定该容器，或者 Active Document 没有在容器中运行，则 GETHOST() 返回 null 值（例如，Active Document 在 Visual FoxPro 的一次交互工作期或在 Visual FoxPro 运行时刻运行）。

请参阅

ISHOSTED(), Name 属性