FoxPRO 事件驱动接口的实现

李智慧 陈继进 (哈尔滨工程大学)

摘要:*FoxPRO* 事件驱动接口是一种新型的接口。与传统的接口相比,它具有快速灵活的特点,其概念和结构也更为复杂。本文较详细地介绍了事件驱动接口的结构、特点及其设计方法。

FoxPRO 是继 dBASEIII 与 FoxPRO 以后出现的又一个大众化数据库管理系统,它继承了 dBASEIII 和 ForPRO 的许多特点,如易使用、易维护、速度快等,同时又增加了许多新的功能,并且引入了一些新的程序设计方法,这些新的设计方法因其复杂而且难于理解,所以很少被采用。大多数人还是用传统的方法设计 FoxPRO 程序,因而没有能最大限度地发挥其优越性。事件驱动接口是 FoxPRO 引入的一个新型的接口,它类似于 FoxPRO 本身的接口形式。与以往类型的接口相比,事件驱动接口的使用效率更高。

一、事件驱动接口的特点

事件驱动接口与传统的接口相比,加强了程序启动的随机性。从外部形式来看,使用了这种接口,只要在系统等待用户输入时都可以对菜单进行选择,不必等到一个模块运行完毕返回菜单。例如,当用户在输入数据时可以选择统计模块,对数据进行统计,然后再返回输入模块继续输入数据。显然,这样类型的接口更加方便用户。

从内部结构来说,事件驱动接口不同于传统的接口结构。传统的接口结构是在循环过程中等待用户输入。所以必须在一个进程运行完毕之后返回菜单才能运行另一个进程。而事件驱动接口是由菜单程序或屏幕程序中的一系列过程或函数来控制的,这些过程或函数在系统或模块启动时被驻留内存,在适当的时机被激活,开始运行。这些过程或函数可以根据条件决定终止运行或是返回断点,而且在其运行过程中可以被其它过程中断或终止。因此,这种程序结构更加灵活。

二、事件驱动接口的设计基础

设计事件驱动接口的必要条件是首先需要设计出事件驱动的菜单和屏幕程序,然后是对各个模块进行协调的进程。这种协调的功能是由"基本 READ"来实现的。上述程序的设计都不同于传统的程序设计方法。下面对此进行简单的介绍。

1.菜单设计

FoxPRO 提供了一种具有中断能力的弹出式菜单。这种菜单只要在有 READ 命令执行(或有 BROWSE 等命令执行)的情况下都可以对其进行选择并执行相应模块。执行完毕后再返回现在运行的程序。

设计 FoxPRO 菜单最简单的方法是使用菜单生成器,设计方法在很多参考书上都有介绍,这里就不再说明。值得一提的是,设计事件驱动接口最好使用 FoxPRO 的系统菜单棒。使用系统菜单棒与普通菜单的区别是设计系统菜单棒首先必须执行下述命令:

```
SET SYSMENU TO    &&显示系统菜单棒中的所有子菜单
SET SYSMENU AUTOMATIC    &&显示系统菜单棒
```

此外,在定义菜单中各项时要使用系统主菜单名"_MSYSTEM"。

2.屏幕设计

FoxPRO 的屏幕程序不再是由循环加 READ 的传结构来实现,它的过程控制是由 READ 命令后一系列函数来实现的。例如,VALID 函数可以根据用户最后所按的键来控制是退出 READ 命令还是移动记录指针;SHOW 函数可以用于在改变内容的情况下改变屏

幕的一些提示信息;ACTIVATE 和 DEACTIVATE 函数可以用于协调同一 READ 中的多个窗口等。

事件驱动的屏幕程序一般包含以下四个部分。

(1)设置代码:包括数据库及索引文件的打开,各种开关及参数的设置等命令。

(2)目标设置:包括各个窗口的定义、激活命令以及与窗口对应的 GET 命令。

(3)READ 命令:包括各种控制屏幕程序运行过程的函数。

(4)清除代码:包括对文件关闭,窗口撤消等命令。

由此可见,FoxPRO 屏幕程序最主要的功能是由 READ 命令来完成的,而且采用的是一种随机触发的方法,因此更加快速灵活。这种程序在被其它进程中断时既可以被清除从而终止运行,也可以被保护然后继续运行,所以适用于事件驱动接口。

3.基本 READ

基本 READ 就是不包含 GET 目标的 READ 命令,它是 FoxPRO 新增的内容。基本 READ 命令中包含一个 VALID 函数,系统根据它的返回值来确定是否终止基本 READ 的运行。

基本 READ 命令的 VALID 函数在以下两种情况下被触发。

(1)有键或鼠标被按下,而且所按的键和鼠标与菜单无关。在基本 READ 行过程中选择菜单不能触发其 VALID 函数。

(2)基本 READ 下一级的 READ 被终止也会导致基本 READ 的 VALID 函数被激活。这一点正是协调各个普通 READ 程序的关键。在应用系统的主控程序中的菜单定义命令后加入包括 VALID 函数的基本 READ 命令,基本 READ 的 VALID 函数可用作各个 READ 程序之间转换的枢纽。

三、事件驱动接口的设计原理

1.主控模块设计

事件驱动接口的主控模块主要由以下部分组成。

(1)菜单定义命令事件驱动接口的模块调用不是直接的,而是通过两个过程来实现的。一个过程用于调用 READ,另一个过程用于调用非 READ 程序。这两个过程都有一个参数,就是被调用的程序名。菜单定义命令中应包含以下命令。

```
on selection bar n do mspr with name1
on selection bar n do mprg with name2
```

其中"MSPR"为 READ 程序的调用过程,"NAME1"为所调用的 READTKYC;"PRG"为非 READ 程序的调用过程,"NAME2"为所调用的非 READ 程序。

(2)包括 VALID 函数的基本 READ 命令。

事件驱动接口的控制主要是通过以下几个过程与函数来实现的。

2.READ 程序调用过程

该过程的功能是判断是否有 READ 程序正在运行,如果有则清除现行程序,激活主控程序中基本 READ 命令从而执行新调用的 READ 程序。其程序如下所示。

```
parameter name_read   &&被调用的程序名
if rdlevel()>1   &&是否有 READ 程序运行
  tobedone=name read   &&被调用的程序名赋给全局变量
  clear read   &&清除现行 READ,激活基本 READ
else   &&没有 READ 程序运行
  do (name read)   &&直接调用 READ 程序
endif
```

3.非 READ 程序调用过程

该过程的功能是调用新的非 READ 过程,不管现在是否有程序正在运行。但在调用非 READ 过程之前必须保存程序运行现场,包括当前工作区、数据库指针和其它参数。在调用后对其进行恢复,防止出现错误。

4.退出过程

在主控模块中必须定义一个全局变量作为是否退出系统的标志。在进入退出过程中首先将设置退出标志,然后清除所有 READ 程序,激活基本 READ。在基本 READ 的 VALID 函数中根据退出标志终止基本 READ,从而退出系统。其程序如下所示:

```
fexit=.t.   &&置退出标志
clear read all   &&清除所有 READ,激活基本 READ 的 VALID 函数
```

5.基本 READ 的 VALLD 函数

在基本 READ 的 VALID 函数中应包含以下内容。

(1)判断是否终止系统运行。是则返回"T"退出系统,否则继续执行。

(2)是否有新的 READ 程序被调用。如有则调用该程序,返回"F"退出 VALID 函数。否则继续执行。

(3)使用 DO CASE 命令根据 WONTOP 函数的返回值确定最后一个被中断的 READ 程序,并执行这个程序。返回"F"退出 VALID 函数。

6.事件驱动接口的运行过程

事件驱动接口的运行过程与循环结构的运行过程相比要复杂得多,它在不同的条件下有不同的运行过程。下面举一个简单的例子来说明。

当进入应用系统后,首先进行的是对菜单的设置,然后执行基本 READ 命令,系统处于等待状态。这时如果用户在菜单中选择了 READ1,进入 READ1 程序。在 READ1 的运行过程中用户又在菜单中选择了非 READ 程序 PRG1,然后选择了 READ2。其运行过程如图 1 所示:

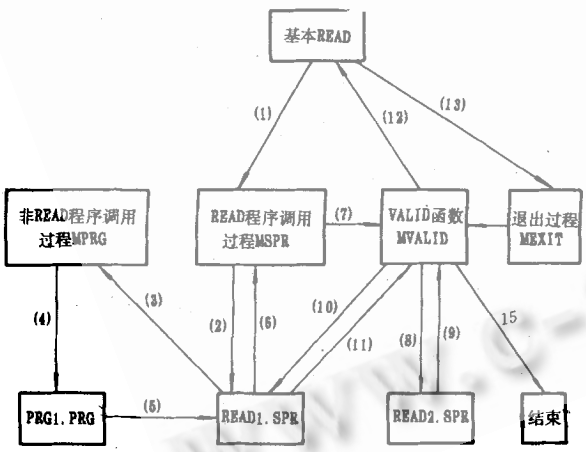

图 1 事件驱动接口运行过程示意图

图 1 中各步骤的说明如下:

(1)用户在菜单中选择 READ1,激活 READ 程序调用过程 MSPR。

(2)MSPR 调用 READ1.SPR。

(3)用户在菜单中选择非 READ 程序 PRG1,激活非 READ 调用过程 MPRG。

(4)MPRG 调用 PRG1.PRG。

(5)PRG1.PRG 运行结束,返回 READ1.SPR

(6)用户在菜单中选择了 READ 程序 READ2,又激活了 MSPR。

(7)在 MSPR 中执行 CLEAR READ 命令激活了基本 READ 命令的 VALID 函数 MVALID。

(8)MVALID 调用 READ2.SPR。

(9)READ2.SPR 运行正常结束。又激活了 MVALID。

(10)MVALID 重新调用 READ1.SPR。

(11)READ1.SPR 正常运行结束。MVALID 再次被激活。

(12)没有未执行的 READ 程序,基本 READ 继续执行。

(13)用户选择退出模块,激活退出过程 MEXIT。

(14)MEXIT 清除所有 READ,激活 MVALID。

(15)MVALID 返回"T",终止基本 READ,从而退出系统。

7.设计事件驱动接口的注意事项

由上可知,事件驱动接口不同于传统的循环结构的接口,各个模块不再是独立的。在任何一个模块运行的过程中,都可能被中断或中断其它模块的运行。因此,在各个模块的设计过程中,都要考虑到与其它模块的协调问题,主要有以下几点:

(1)运行状态的恢复。在模块的运行过程中,必然要对某些状态进行改变,如对数据库文件的打开和关闭、指针的移动、参数的改变等。如果这些状态被改变而没有恢复,就有可能引起其它模块的运行错误。因此在程序运行结束时,应对改变的状态进行恢复。

(2)变量的设置。在事件驱动接口中,各个模块中用到的局部变量和全局变量都有可能被其它模块修改。因此在设计过程也要考虑到这一点,应该了解全局变量的各种变化。对于局部变量,应采取措施尽量避免其名字出现重复。