《现化电子技术》增刊
第 219 号

FoxBASE 编程技巧实例汇编

张　达　菲

陕西电子杂志社

FoxBASE 编程技巧实例汇编

张 达 菲

陕西电子杂志社

目　录

在Foxbase应用中需要注意的几个问题

〖问题的提出〗 在Foxbase和dBASEⅢ应用中经常碰到一些事先没有预料到的问题，而这些问题往往又是书本和资料上没有介绍的。为了解决这些问题，需要花比较多的时间和精力，笔者在这里把在程序设计和应用中碰到的几个问题的处理方法介绍给大家。

一、记录指针对READ语句的影响

在当前工作区打开了一个数据库，并且记录指针已指向库尾（即EOF( )函数的值为真），此时无论是用READ语句对字段名变量还是对内存变量进行输入或修改操作都会出现End of file encountered错误。例如有一数据库HT.DBF，其结构和内容如下：

Field	Name	Type	Width	Dec	Record#	X	Y	Z
1	X	Numeric	3		1	100	100	100
2	Y	Numeric	3		2	150	150	100
3	Z	Numeric	3		3	80	80	40
** Total **			10		4	200	200	100

针对数据库HT编写如下程序：

```
set talk off                          jx=" "
use ht                                @10, 20 say "输入JX:" get jx
do while .not. eof( )                 read
  ? x, y, z                           use
  skip                                set talk on
enddo                                 return
```

程序运行到READ语句就出现End of file encountered错误。解决办法是在执行READ语句之前增加USE语句（关闭数据库）或增加SKIP-1语句（把指针退回去）。

二、使用同名的字段变量与内存变量

同时定义了同名的字段变量与内存变量，两种变量都存在，但对内存变量的操作需加前缀M->，关闭数据库后就不需加前缀了。例如：

```
.use ht          .x=x+100          .?M->x          .?x
.?x                .?x                  200                200
100                100              .use
```

注意x=x+100语句中，前一个x是内存变量，后一个x是字段变量。在程序设计中要特别注意同名内存变量和字段变量的使用，STORE和"="只能给内存变量赋值，在其它语句中，若不加前缀M->则只对字段变量操作，而@…GET READ语句在有同名的字段变量与内存变量时，只对字段变量操作；没有同名的字段变量时，才对内存变量操作。因此，最好不要使用同名的内存变量和字段变量。

三、关于SUM语句和AVERAGE语句

使用SUM求和或用AVERAGE语句求平均值时，可以把结果赋给内存变量，但必须注意内存变量的个数要与求和或求平均值的数值型表达式个数一样多，否则出现语法错

误。若在 SUM（或 AVERAGE）语句中没有指定数值表达式，表示对库中所有数值字段求和，如果希望把部分计算结果赋给内存变量，也必须在语句中指定与数值字段个数一样多的内存变量。

四、Fox 系统中的冗余文件

运行 Fox 系统程序进入点状态，便根据当前时间生成一个以"0"或"1"打头、文件长度为 0 的冗余文件，该文件在用 QUIT 命令退出 Fox 系统时被自动删除。如果不是用 QUIT 退出 Fox 而直接关机或复位，则冗余文件不被删除而驻留在磁盘上了。

五、对打开了索引文件的数据库的插入

对一个打开了索引文件的数据库实施插入操作时，不管插入前记录指针在什么位置，新记录插入后的逻辑位置根据索引关键字的值排列，而其物理位置总是在库的最后一条记录.(即其记录号是原最大记录号加 1)，等同于 APPEND 命令。

六、用 READ 语句输入数据不能归位的处理

在某些系统环境下用@…GET 和 READ 语句输入或修改数据时，往往数据输入以后不归位，屏幕上的显示给人一种错觉。下面程序是用@…GET 和 READ 语句向数据库 HT 输入数据:

```
set talk off
use ht
appe blank
@10, 10 say "输入 X:" get x
@10, 30 say "输入 Y:" get y
@10, 50 say "输入 Z:" get z
read
?x, y, z
use
set talk on
retu
```

运行程序，分别给 X，Y，Z 输入数据 10，但在屏幕上却显示为:

输入 X: 100　　输入 Y: 100　　输入 Z: 100

10　10　10

数据输入且回车以后，数据没有归位，所以输入的是 10，接收的也是 10，但显示的是 100，容易造成输入了 100 的错觉。解决办法是在 READ 语句之后增加@…SAY 语句重新显示，比如上例应在 READ 语句增加下面 3 个语句:

```
@10, 16 say x
@10, 36 say y
@10, 56 say z
```

在这些系统环境下用@…GET 和 READ 语句输入或修改数据时，在"插入"状态下，数据不显示，应将状态改为"覆盖"方式。

七、几个命令对指针的影响

PACK 命令执行以后，记录指针移到数据库的顶部，若打开了索引文件，记录指针移到逻辑排序的第一条记录。

在索引文件打开的情况下，GO TOP 将记录指针移到逻辑排序的第一条记录，GO BOTT 将记录指针移到逻辑排序的最后一条记录，GO n 将记录指针移到第 n 条记录。而 SKIP 命令则是根据索引文件向前或向后相对移动 n 条记录。

多用户下 FoxBASE⁺出现的问题及解决办法

〖问题的提出〗 我公司一台 LC386／25C 微机（4M 内存，200 硬盘），使用 XENIX 2.3.1 带 5 台汉字终端。我们用 FoxBASE⁺1.0 自己开发了一个管理程序，为 8 个用户管理各自的文件（文件在各个用户目录下，各文件名相同），每个用户要打开 8 个数据库及 8 个索引文件，当第 4 个用户进入时，总会出现其他用户正在使用的出错信息。少于 4 个用户时，不出现这种情况，其实其他用户并没有使用文件，不知何故？

据我分析，问题是对于 SCO-XENIX 系统下，SCO-FoxBASE⁺V1.00 关系数据库的使用不当所致。本人以前也曾遇到过类似情况，其解决办法有两种，供参考。

(1) 在你的管理程序（.PRG）中（尤其在同时使用多个工作区，打开多个库文件或索引文件的地方），要首先将库文件的属性设置为共享状态，否则就会出现"其他用户正在使用"现象。具体作法是，在打开库文件之前加上一条设置：SET EXCLUSIVE OFF，然后再运行程序，错误提示消失。

(2) 多用户 FoxBASE⁺V1.00 运行环境的好坏取决于：①系统安装时交换区大小的设置；②用户目录下 confiy.fx 的设置。

根据本人经验，用户交换区设置为 12 000 块（每块为 1K）为好。

confiy.fx 建立为 confiy.215 为好，在用户目录中，输入建立环境的设置命令：

$ cp／usr／lib／foxplus／config.215.／config.fx(CR)

使用 Foxbase 故障解决一例

〖问题的提出〗 我单位在使用用 Foxbase 语言编制的管理程序时，当数据库进行数据统计后突然屏幕出现"Invalid buffpoint call"错误信息，机器锁死。

1. 故障分析

从错误指示看是不正确的缓冲区指针调用，在 DOS 系统中无此错误信息，可基本断言这不是操作系统问题。程序执行时机器锁死，不外乎由两种因素造成：一是程序死循环；二是程序走不通而阻塞。由于是缓冲区指针指向不正确，因而可排除死循环，因为死循环指针调用仍是正确。

Foxbase 中记录的物理结构是根据稠密索引组成的非顺序文件。要处理记录首先要将文件的索引读进内存，然后查找需要的关键字确定其记录的柱面和磁道位置之后将记录读进内存。文件的关键字是放在磁盘的一个索引区内，而记录放在另一个对应区内，因此"Invalid buffpoint call"是指为处理记录数据将索引调入内存，指针在索引文件中查找关键字时走不通。我们无法观察索引文件。利用 use 文件名 INDEX 索引文件名打开所有索引文件，在点状态下用 REINDEX 重建索引文件，再运行程序正确，解决了死锁的问题。

2. 故障原因

Foxbase 是严格的数据库系统，由于操作不当，特别是打开数据库文件时未关闭而强

行回到 DOS 状态下，或者突然掉电引起索引文件损失（这时数据文件幸免）。由于 Foxbase 程序编译执行，.Fox 文件不能打开，错误不易发现，Foxbase 用户在使用时要注意这一问题。

快速编制软件使用说明书的技巧

〖问题的提出〗 众多 dBASEⅢ软件开发者在完成软件的设计后，都想为用户编制一份使用说明书，附在其系统盘上，为用户了解和使用软件提供方便。但因为编码在工作结束后，程序中已经存有编辑好的菜单等内容，如果再在说明书中重新编辑输入这些内容，无疑这部分的工作是重复的。如何利用现有的程序，简单快速地生成使用说明书呢?

具体实施步骤如下:

(1) 在dBASEⅢ状态下，进入软件所在驱动器及其目录。

(2) 用 set alte to＜使用说明书名＞语句将以后的所有屏幕输出转向到使用说明书中。

(3) 用 set alte on 语句打开屏幕输出转向开关。

(4) 运行应用软件。这里需要注意软件的使用顺序，要按菜单的安排逐一运行，并且对菜单的每个功能均需"面面俱到"。

(5) 关开关。运行结束后，要用 set alte off 语句关闭屏幕输出的转向开关，以形成使用说明书。

(6) 对软件说明书进行编辑。由于此时使用说明书只含有菜单等屏幕信息，因此还需使用文本编辑软件如 WS 对其加注解。该工作完成后，你即可获得一份完整的软件使用说明书。

如何快速启动 dBASEⅢ

〖问题的提出〗 在使用 dBASEⅢ数据库管理系统时，在 C＞提示符下硬盘运行 dBASEⅢ，当键入 dBASE 后，由于 dBASEⅢ系统的内部控制，系统先要到 A 驱动器查找系统文件，直至在 A 盘查找失败后才转移到硬盘读取系统文件，然后进入到 dBASEⅢ系统管理状态。在 dBASEⅢ的启动过程中，对于读 A 盘的操作是我们所不希望的，特别是当工作中需要频繁进入、退出 dBASEⅢ状态时，文件的启动就占用相当长时间。怎样才能够绕过 A 驱动器，而直接由硬盘启动 dBASEⅢ系统，从而缩短启动时间呢?

本文介绍一种快速方法。我们知道在 DOS3.X 以上版本的操作系统中新增有一个 DOS 外部命令 SUBST，其意是将子目录定义为逻辑盘。命令格式:

SUBST [＜盘符＞＜盘符＞＜路径＞]

SUBST＜盘符＞／D

可以使用这一命令直接在硬盘启动 dBASEⅢ。具体操作过程如下:（假定硬盘为 C 盘，用户的子目录名是 USER)。

(1) 在硬盘 DOS 操作系统的根目录下拷入 SUBST.EXE 文件。

(2) 在硬盘 USER 子目录名下拷入 dBASEⅢ的所有系统文件 dBASE.EXE、dBASE.OVR 等。

(3) 在硬盘 USER 子目录名下编辑一个名为 dBASE.BAT 的批处理文件。内容是:

```
TYPE dBASE.BAT
C:\SUBST A: C: USER
dBASE
C:\SUBST A:/D
```

使用: 当要启动 dBASEⅢ时, 只要在用户自己的子目录名下键入 dBASE 即可由批处理文件自动执行把当前子目录定义为逻辑盘 A, 并自动从硬盘启动运行 dBASEⅢ, 进入 dBASEⅢ管理状态。当退出 dBASEⅢ时, 由批处理文件自动完成取消子目录和逻辑盘 A 的逻辑替换关系, 使用非常方便。

批处理文件内容命令解释:

① C:\SUBST A: C:/USER

定义 C 盘子目录名 USER 为逻辑盘 A, 执行这条命令后, 以后所有对 A 盘的读写操作都转到 C 盘子目录名 USER 之下进行。

② C:\SRBST A:\D

解除 C 盘子目录和逻辑盘 A 的替换关系。

注意: 批处理文件执行完后, A 盘已经被子目录名所代替, 故 A 盘逻辑上已等价于子目录名 USER, 即 A: =USER。如 DIR A: 实际上执行的是 DIR C:\USER 操作。DEL A: *.* 实际上执行的是 DEL C:\USER *.* 操作。这种逻辑关系上发生的变化务必要注意到, 避免引起误操作。

实际上不仅 dBASEⅢ可以这样启动运行, 其它的商品化软件在启动运行时由程序内部控制, 需先读取 A 盘文件, 然后再转到硬盘继续运行象五笔字形(WBZX), 印刷线路板制作(Smtwork)等皆可仿此方式启动运行, 从而可以缩短文件启动时间, 提高工作效率。

高效全自动型 dBASEⅢ程序调试环境的设计

〖问题的提出〗 程序的设计思想如下: 本软件首先将 F2 功能键模拟出 dBASEⅢ的"MODI COMM"编辑程序环境, 用户可以输入某一被调试程序名, 接着用户可以进入 dBASEⅢ字处理器编辑修改该程序。用户编辑修改完毕后, 本软件直接进入 dBASEⅢ"DO"运行环境去运行当前被调试程序, 同时把当前被调试程序的特征记录于自身的参数文件中, 从而实现记忆功能, 以便下次用户按 F2 功能键重新进入对最近一次被调试程序的编辑。由此本软件就构造出调试指定程序的高效全自动环境。

本软件所构造的调试环境平常处于"挂起"休眠状态, 不影响用户其他操作, 不影响 dBASEⅢ的整体环境; 本软件的"激活"运行状态, 完全由用户控制 (按 F2 功能键)。本软件的安全性也较好, 当参数文件被破坏时, 程序能以"空"文件名作为缺省值, 迫使过程中止; 当用户回答文件名不正确时, 过程也自动中止。程序清单如下:

```
CLEA ALL
IF FILE(' L02.MEM ')
   SET SAFE ON
   REST FROM L02 MEM
   SET SAFE ON
   LENTH0=LEN(COMMAND0)
   COMMAND0=COMMAND0+SPACE(30-LENTH0)
ELSE
   COMMAND0=SPACE(30)
ENDI
   CLEA
   @1, 0 SAY' MODICOMM ' GET COMMAND0
   READ
   COMMAND0=TRIM(COMMAND0)
   MODI COMM &COMMAND0
   SET SAFE OFF
   SAVE TO L02.MEM ALL LIKE COMMAND0
   SET SAFE ON
   DO &COMMAND0
   RETU
```

Foxbase 2.1 命令执行效率分析

〖问题的提出〗 Foxbase2.1 系列数据库软件，以其兼容性好、功能强、速度快的优点，成为国内管理信息系统开发的主流软件，其应用技术逐渐为广大用户所掌握。作为该软件的命令执行效率（在这里效率是指执行时间和所占空间），由于使用手册与各种文章介绍较少，使用者还不是很清楚。

根据命令的执行效率，合理使用之，则会使编出的程序处于最佳的运行状态；否则，可能是事倍功半。作者根据这几年使用 Foxbase2.0、2.1 的体会，就 Foxbase2.1 中的三类命令：计数运算类、查找类、组织类分别作一番论述。

首先给出所做实验的环境条件：

机型：AST　P396C

硬盘：110M　时钟：20M　内存：2M

操作系统：DOS3.31　CCBIOS　2.13H

数据库：

记录数：80000 条　　容量：17.6M

硬盘所剩空间为：75M

一、计数、运算类

计数、运算类主要包括用于求和、计数、替换、删除等命令，它们只对当前使用的数据库有效，而不会产生其它数据库。

表1

命　令	ALL	FOR	WHILE EXP1	WHILE EXP2
COUNT	58″	1′15″	2′8″	0.11″
AVERAGE	1′33″	1′41″	2′31″	0.16″
SUM	1′33″	1′43″	2′31″	0.17″
REPLACE	3′43″	3′47″	4′19″	0.16″
DELETE	3′19″	3′32″	4′3″	0.17″
RECALL	3′17″	3′30″	4′4″	0.11″
PACK	2′52″			

ALL：表示对所有记录进行处理

FOR：SEX＝'1'满足条件者52398条

WHILE EXP1：SEX＝'1'满足条件者52398条

WHILE EXP2：SEX＝'0'满足条件者3条

SEX——性别

由表1可以得出以下结论：

①计数语句执行最快，这是因为C中的累加++运行最快。

②用FOR语句会减慢运行速度。

③慎用WHILE条件，因为从表1可以看出当满足条件的记录数较大时，WHILE比FOR执行还慢。

④用RECCOUNT( )函数代替COUNT ALL语句，看表2即知。

表2

命　令	时　间
COUNT ALL	58″
RECCOUNT( )	0.6″

二、查找类

查找是指按某些条件在数据库中搜索满足的记录数，主要用于数据库的检索功能。

时间1是由NATION＝'01'得到的。满足条件者63548人。

时间1是由NATION＝'02'得到的。满足条件者1703人。

NATION——民族。

由表3可以得出以下结论：

①IF——ELSE——ENDIF和COUNT两语句与满足条件人数无关，即其查找时间是一定值。

②SEEK、FIND和LOCA、CONT则与满足条件人数有关，即其查找时间是一不定值。

③在满足条件记录数较少时应使用SEEK或FIND语句，否则用COUNT语句。

④用IIF函数代替IF... ELSE... ENDIF语句，看表4即知。

表3

命　令	时间1	时间2
IF...ELSE...ENDIF	20′20″	20′18″
COUNT	1′15″	1′15″
SEEK 或FIND	1′45″	15″
LOCATE CONT	16′41″	1′27″

表4

命　令	时　间
IF...ELSE...ENDIF	0.11″
IIF	0.5″

三、组织类

组织类命令是指索引、排序、拷贝等操作，它们在执行过程中都会产生附加的文件。

表5中的全部是指对所有字段进行处理，局部，只对各别字段进行处理。

时间下面是所形成文件的字节数。

表5

命　令	全　部	局　部	置唯一	条件 SEX＝1
INDEX	4′28″ 5.12M	2′56″ 460K	2′48″ 1K	3′31″ 3.3M
COPY	3′11″ 17.6M	1′24″ 1.28M		
TOTAL	6′19″ 13.2M	1′10″ 960K		
SORT	19′48″ 17.6M	4′10″ 1.28M		
APPEND	4′36″ 17.6M	1′21″ 1.28M		

由表5可知:

①所形成文件的大小对运行时间起决定作用。文件越大，时间越长；文件越小，时间越短。

②用FOR有时更快。

另外需注意:

③SORT要求有三倍于原数据文件的空间。

④TOTAL中的FIELDS选择的是求和的数值字段，而不是选择应在TO文件中产生的字段子集。

解决 CLIPPER 不能编译子程序的问题

〖问题的提出〗 dBASEⅢ的CLIPPER编译器对于诸如"DO ZK&XZ"这样的命令语句，虽能正常通过，却无法将宏代换所指的ZK1、ZK2等子程序编译到主程序中，使编译出的程序无法使用，因而使dBASEⅢ的这种宏代换的用法受到了限制，解决该问题的方法有两种。

方法一：在主程序中用判断语句一一指定这些子程序的具体名称，如下面的程序:

```
IF XZ="1"                    IF XZ="2"
   DO ZK1                       DO ZK2
ENDIF                        ENDIF
.....
```

这样就可以让编译器能识别出这些子程序，并将之编译到主程序中。该方法的特点是程序结构直观，但语句繁琐。

方法二：仍采用"DO ZK&XZ"的使用方式，但在主程序最后加上如下的小程序即可:

```
DONE=.F.                        DO ZK2
IF DONE                         .....
   DO ZK1                       ENDIF
```

这种解决问题的技巧是制造一个逻辑值为假的判断条件，使该条件所对应的程序语句不能被执行，但却可使其中的 ZK1、ZK2、等子程序在编译时能被编译器所调用，并编译到所生成的文件中，从而解决了子程序不被编译的问题。该方法的特点是程序结构简单，并保持了宏代换的技巧。

如何解决 dBASEⅢ编译工具 Clipper 对 Total 命令的缺陷

〖问题的提出〗 Clipper 较 dBASEⅢ有很多优点，例如执行速度快，保密性强等；但也有不足之处，其主要表现在缺少摘要命令 Total。虽然，Clipper 提供了外部过程文件 Total.prg 速度特别慢，而且得到的摘要数据库不具有字符型字段。

下面举的一个小例子，不调用 Total.prg 而对这个问题做了很好的解决，在应用过程中只要根据你的数据库及要求，对这个例子稍加修改就可以达到预期的目的。

数据库结构--数据库名: C: hu.dbf
数据库中的数据记录个数: 6
数据库的最后更新日期: 01/04/91

字段	字段名	类型	宽度	小数位
1	BH	Character		2
2	C1	Character		2
3	N1	Numeric	5	2
4	N2	Numeric	5	2
5	N3	Numeric	5	2
* * 总计 * *			20	

Record#	BH	C1	N1	N2	N3
1	01	SS	1.00	2.00	3.00
2	01	EE	3.00	4.00	1.00
3	02	Q1	3.00	4.00	2.00
4	03	E3	3.00	3.00	3.00
5	02	WW	1.00	1.00	1.00
6	01	RT	3.00	3.00	3.00

```
SELE 1
USE HU
*建立摘要数据库结构 HU1.DBF 选择你需要的字段
COPY TO HU1 FIELDS BH,
C1, N1, N2 STRU
SELE 2
USE HU1
SELE 1
*按 BH 索引
INDE ON BH TO HU
GOTO TOP
DO WHILE .NOT. EOF( )
   *建立内存变量作累加用
   STORE 0 TO NN1, NN2
   BH1=BH
   CC1=C1
     DO WHILE .NOT. EOF( )
      *对数字型变量做累加
      NN1=NN1+N1
      NN2=NN2+N2
      SKIP
      *如果对某些记录摘要完毕则
      去 HU1.DBF 追加记录
      IF BH1#BH .OR. EOF( )
           EXIT
      ENDIF
     ENDDO
   SELE 2
   APPE BLANK
   REPL BH WITH BH1
   REPL C1 WITH CC1
```