编译 FoxBASE 程序状态开关的变换机理及实现程序

葛晓滨 （合肥市统计局计算站）

摘要:本文全面系统地剖析了编译 *FoxBASE* 程序状态开关语句目标码的形成机理。并在此基础上，探讨了重新设置编译 *FoxBASE* 程序状态开关的方法。

一、引言

普通反编译 FoxBASE 程序方法是通过设置 FoxBASE 程序的状态开关，将 FOX 程序缓冲区内的源程序解释语句转向输出到某外部文件中，由此获取源程序代码。但在实际运用中，这种方示往往难以奏效，这是因为多数的 FOX 程序都在其程序内部设置了关闭上述 FoxBASE 状态开关的开关语句，使得在外部环境中设置好的开关状态在进入应用程序时又重新关闭。

解决这个问题的途径之一，便是重新设置编译 FoxBASE 程序的内部状态开关，使其按照用户的需要进行工作。

二、基本原理和方法

FoxBASE 状态开关的设置是由 FoxBASE 的 SET 命令完成的，命令的基本格式是:

SET <变量名>[<ON> <OFF>]

在增强型 FoxBASE2.0 / 2.1 版软件中，可由 SET 命令设置完成的 FoxBASE 状态开关共有 30 个。为实现重新设置 FOX 程序状态开关，就需要先从 SET 命令目标码形成过程的分析入手。

根据 FoxBASE 用户手册，FOXPCOMP.EXE 在编译生成 FOX 程序的目标码时，提供给用户两种编译方式:一种是不加"-E"参数编译形成的 FOX 程序，可称之为"普通型 FOX 程序"；另一种是加"-E"参数编译形成的加密的 FOX 程序，可称之为"加密型 FOX 程序"。

通过对 FOX 程序的内部结构进行分析，可以探知普通型与编译型 FOX 程序的区别方法:提取 FOX 程序开始的两个字内容，若为 FBH、2BH 则为普通型 FOX 程序，若为 FBH、2AH 则为编译型 FOX 程序。

普通型 FOX 程序 SET 命令关键码表

SET 变量名	关键码	SET 变量名	关键码
ALTERNATE	16H	EXCLUSIVE	CEH
BELL	1BH	FIELD	3BH
CARRY	1FH	FIXED	86H
CENTURY	90H	HEADING	43H
CLEAR	79H	HELP	97H
COLOR	25H	HISTORY	BDH
CONFIRM	27H	INTENSITY	46H
CONSOLE	28H	MENU	9BH
DEBUG	2CH	PRINT	57H
DELETE	2EH	SAFETY	8BH
DELIMITER	2FH	SCOREBOARD	8CH
DOHISTORY	C4H	STATUS	65H
ECHO	33H	STEP	66H
ESCAPE	36H	TALK	6AH
EXACT	37H	UNIQUE	8DH

1.普通型 FOX 程序开关命令目标码的形成机理

对于普通型 FOX 程序，FOXPCOMP 产生的 SET 语句的目标码是一个具有 5 个字节长度的二进制代码，它的组成可以表述为如下的表达式:

<SET 命令目标码>=<起始码>+<关键码>+<开关码>+<结束码>

<起始码>:是 SET 语句的起始标志码，占 2 个字节，其内容固定为 04H 和 47H。

<关键码>:是 SET 语句的环境变量标志码，占 1

个字节，其内容为下表各环境变量对应的＜关键码＞值。

＜开关码＞:是 SET 语句的开关状态标志码，占 1 个字节，其内容为 51H 时表示开关状态为"OFF"，其内容为 52H 时表示开关状态为"ON"。

＜结束码＞:是 SET 语句结束标志码，占 1 个字节，其内容固定为 FEH。

举例说明如下:

对于 SET TALK ON 语句，查上述表格的"TALK"项可得其对应的＜关键码＞为 6AH，同时其状态开关为"ON"，对应的＜开关码＞为 52H，则编译后的 SET TALK ON 语句的目标码是 04H、47H、6AH、52H、FEH。同理，对于 SET TALK OFF 语句而言，其开关状态为"OFF"，对应的＜开关码＞为 51H，则其产生的目标码是 04H、47H、6AH、51H、FEH。

根据普通型 FOX 程序 SET 语句目标码这种形成特点，只要在普通型 FOX 程序中查找到 SET 语句的目标码，重新设置其中的＜开关码＞值，即可直接改变这种 FOX 程序中的状态开关。

2.加密型 FOX 程序开关命令的目标码形成机理

对于加"-E"参数编译的 FoxBASE 程序，FOXPCOMP 的工作实际上是先对其按照普通型 FOX 程序的编译方法进行编译。在此基础上，再对编译好的 FOX 程序进行加密。加密型 FOX 程序的加密工作只是在普通型 FOX 程序的目标码上进行的，因此，只要能够加密型 FOX 程序代码还原为普通型 FOX 程序的目标码，再按照普通型 FOX 程序开关状态的设置方法将其开关状态码设置为新的状态码，再将该新设定值按加密算法进行运算，产生编译型 FOX 程序的代码，并将之写回加密型 FOX 程序中，由此即可改变加密型 FOX 程序的开关状态。

编译型 FOX 程序的目标码还原可描述为四步:

(1)取文件头开始的第 18 到 33 字节的内容:D_i(i＝1～16)。

(2)用算法 $K=\sum_{i=1}^{16}(D_i\&C_o)\times 2^{i-1}$ 取得密钥匙 K。

(其中常量 C_o＝01H，运算符"&"代表与"运算")

(3)产生密码组:其迭代算法为

$S_o=K$

$S_i=S_{i-1}\times C_1+C_2$　(i＝1～508)

其中常量 C_1＝0C45H、C_2＝3619H。通过这个迭代运算，可产生系列数 S_i，它们均是双字节的长整数，取 S_i 系列数的高字节的 8 位数组成两个密码组:

S_1～S_{251} 为第一序列:M_i(i＝1～251)

S_{252}～S_{508} 为第二序列:N_j(j＝1～257)

由此，我们获得了两个密码组 M_i 和 N_j。

(4)还原算法

对编译型 FOX 程序内的每个字节 F_k，采用算法:

$G_k=F_k\hat{}M_i\hat{}N_j$(i＝1～251,j＝1～257)

可以将加密的 FOX 代码 Fk 还原为普通型 FOX 程序代码 G_k(其中运算符"^"代表"异或"运算)。

再采用算法:

$F_k=G_k\hat{}M_i\hat{}N_j$(i＝1～251,j＝1～257)

又可以将普通型 FOX 程序代码 Gk 加密为加密型 FOX 程序代码 F_k。

三、实现程序

据此，笔者设计了一个实用程序 SETFOX，它可以直接修改编译后的 FoxBASE 程序的状态开关。有关说明如下:

1.程序的使用环境

硬件:IBM PC、PC／XT、286、386 等微机

软件:①操作系统:DOS2.10、DOS3.00、DOS3.30 等

②编译环境:Turbo C 2.0

2.程序运行格式:SETFOX[＜INPUT＞,＜OUTPUT＞]

其中:＜INPUT＞是待设置状态开关 FOX 程序名

＜OUTPUT＞是设置后输出 FOX 程序名

这两项参数可以在启动 SETFOX 时作为命令行参数使用，也可在 SETFOX 运行后输入其内容。

3.SETFOX 程序可以自动识别 FOX 程序的类型，它既可对普通型的 FOX 程序进行状态开关设置，也可对加密型 FOX 程序状态开关进行设置。其中，对程序中开关语句目标码的识别使用了 TEMP 数组进行模式匹配运算，具有一定的灵活性。有关程序的其它设计技巧，请读者阅后体会，这里不再赘述。

4.源程序清单见附录，其中有一定辅助说明。

四、应用举例

通过设置编译 FOX 程序状态开关，可为提供多方面的服务：

1.无需对源程序重新组织和编译，就能够切换已编译 FoxBASE 程序的运行状态开关。

2.通过开启 TALK、ECHO 等开关，可以为用户分析程序运行故障提供方便。

3.通过开启 ALTERNATE 等开关并在应用程序外部设置转向输出文件，可以产生松散型的源程序清单，由此还可以进一步获取 FOX 程序的密码体制，为分析已编译 FoxBASE 程序提供方便。

4.打开程序的 ESCAPE 开关，并通过外部环境中设置的"ON ESCAPE DO……"语句，可以使已编译 FOX 程序在运行时，由＜EXCAPE＞键激活转向到用户设定的附加程序中运行，由此可行分析程序运行机制以及增加对程序的调试功能。

附:源程序清单

```
/ * * * * * * * * * * * * * * * * * * * * *
  * FILE NAME:SETFOX.C                    *
  * ENVIRONMENT:Turbo c 2.0               *
  * * * * * * * * * * * * * * * * * * * * /
#include "stdio.h"
#include "conio.h"
#include "string.h"
char * table1[]={"ALTENATER","BELL","CARRY","CENTURY",
  "CLEAR","COLOR","CONFIRM","CONSOLE",
  "DEBUG","DELETED","DELIMITERS",
  "DOHISTORY","ECHO","ESCAPE","EXACT",
  "EXCLUSIVE","FLELDS","FIXED","HEADING",
  "HELP","HISTORY","INTENSITY","MENU",
  "PRINT","SAFETY","SCOREBOARD",
  "STATUS","STEP","TALK","UNIQUE"};
  int   table2[]={0x16,0x1b,0x1f,0x90,0x79,
                  0x25,0x27,0x28,0x2c,0x2e,
                  0x2f,0xc4,0x33,0x36,0x37,
                  0xce,0x3b,0x86,0x43,0x97,
                  0xbd,0x46,0x9b,0x57,0x8b,
                  0x8c,0x65,0x66,0x6a,0x8d};
  main (int argc,char * argv[]){
   FILE * in, * out;
   char sfile [33],ofile[33];
   int i,j,k,h,x,y,v1,v2,meet,result;
   int c,d,m,n,s1,s2,setnum,setswitch,encode;
   int encrypt=0;
   int startcode[]={0x04,0x47};
   int switchcode[]={0x51,0x52};
   int fox[2],setcode[4],temp[4],a[252],b[258];
   / * Display program's title * /
   clrscr();
   gotoxy(22,1);
   printf(" < < < SETFOX > > > ");
   gotoxy(22,2);
   printf("Author:X.B.Ge");
   gotoxy(22,3);
   printf("Usage:SSETFOX[<input>,<output>]");
 / * Define input output file * /
   if (argc > =3){
    strcpy(sfile,argv[1]);
    strcpy(ofile,argv[2]);
   }
   if (argc = =1){
     gotoxy(20,4);
     printf("Please input source file name:");
     scanf("%s",sfile);
   }
   if (argc = =2      argc = =1){
     gotoxy(20,5);
     printf("Please input object file name:");
     scanf("%s",ofile);
   }
   if (strchr(sfile,'.') = =NULL)
     strcat(sfile,".FOX");
   if (strchr(ofile,'.') = =NULL)
     strcat(ofile,".FOX");
   gotoxy(24,8);
   if ((in=fopen(sfile,"rb")) = =NULL){
    printf("Cannot open source file:%s", sfile);
    exit(1);
   }
   if ((out =fopen(ofile,"wb")) = =NULL){
    printf("Cannot open object file:%s",ofile);
    exit(1);
   }
   / * Check input file * /
  for (h=0;h<2;h++)fox[h]=getc(in);
if(!(fox[0] = =0xfb(fox[1] = =0x2a
fox[1] = =0x2b))){
    printf("Source file isn't fox file!");
    exit(1);
  }
   / * If input file is an encrype file * /
  if (fox [1] = =0x2a)encrypt=1;
```

```
    /* Display items on screen */
    gotoxy(22,7);
    printf("*** FOXBASE SET ITEMS ***");
    h=0;
    for(i=5;i<=60;i=i+20){
     for (j=8;j<=17;j++){
      gptpxu(i,j);
      printf("%2d:%-12s",h,table1[h]);
      h++;
     }
    }
/* Select item switch */
    again1:gotoxy(5,19);
      printf("Please select item by its number:");
  scanf ("%d, setnum);
      if (setnum<0      setnum>29)goto again1;
    again2:gotoxy(5,20);
      ptintf("Set 0:off->on,1:on->off you reply:");
  scanf ("%d", setswitch);
      if (setswitch<0      setswitch>1)goto again2;
    /* Define option code of selected item */
    if (setswitch==0)k=1;
      else k=0;
    encode=switchcode[k];
    setcode[0]=startcode[0];
    setcode[1]=startcode[1];
    setcode[2]=table1[setnum];
    setcode[3]=switchcode[setswitch];
    /* Obtain the encrypt code */
    if (encrypt ){
      x=1;c=0;
      fseek(in, 18L, SEEK__SET);
      for (i=1, i<=16;i++){
       d=getc(in);
   c+=(dOx01)*x;
       x=x*2;
    }
    for (i=0; i<251; i++){
      m=c*Oxc45+Ox3619;
  c=mOxffff;
  a[i]=(c>>8)Oxff;
    }
    for (i=0;i<257;i++){
      n=c*Oxc45+Ox3619;
  c=nOxffff;
  b[i]=(c>>8)Oxff;
     }
    }
    /* Read code of input file */
    fseek(in, OL, SEEK__SET);
    v1=v2=0;
    for (i=0;i<=3;i++){
     temp[i]=getc(in);
     putc(temp[i],out);
     v1++;
     V2++;
    }
    while(1){
     /* Nove the temp dimension */
     for (h=0;h<2;h++){
      temp[h]=temp[h+1];
     }
     temp[3]=result=getc(in);
     if (encrypt){
       temp[3]^=a[v1];
       temp[3]^=b[v2];
       s1=v1;
       s2=v2;
       v1++;
       v2++;
       if (v1==251)v1=0;
       if (v2==257)v2=0;
     }
     meet=1;
     if (!feof(in)){
      /* compare code */
      for (j=0;j<=3;j++){
       if (temp[j]!=setcode[j])
        meet=0;
     }
     /* If find option code in input file */
     if (meet){
       if (encrypt){
        encode=switchcode[k]^a[s1];
        encode^=b[s2];
       }
       putc(encode,out);
      }
     else
      putc(result,out );
    }
    else
      break;
    }
    gotoxy(14,23);
    printf("Succeed SETFOX! Press any key to exit...");
    getch();
    fclose(in);
    fcloxh(out);
    }
```