d BASE / FoxBASE+过程文件的装订

宁波市机械工业局 朱孟海

摘要: *d BASE / FoxBASE+过程文件的建立是提高采用 d BASE / FoxBASE+程序设计语言所开发的应用软件运行速度的一种重要途径，本文介绍了 dBASE / FoxBASE+过程文件的创建方法，内容包括 FoxBASE+的使用实用程序 FoxBIND 存在问题、解决方法和采用 C 语言编制一个 FoxBIND 的仿真程序的过程。*

在 FoxBASE+中有一个实用工具程序 FoxBIAN，使用它可以极方便地将若干 PRG 文件集中到一个过程文件中，进行自动、快速、高效地装订。

笔者在使用 FoxBIND 装订过程文件时，发现有一二点不足之处：(1)对于不同标准的汉字编码系统(例如汉字编码方法不是简单地以国标码的区码加 160 作为汉字机内码高字节。国标码的位在码加 160 作为汉字机内码的低字节，而是采用其它编码办法)时，使用该程序来装订过程文件，装订完成后在该过程文件中某些汉字被修改成其它字符；(2)在现行目录下执行 FoxBIND 命令，假设被装订的 PRG 文件在其它目录下，即输入的。PRG 文件带有路径名操作时，在装订成的过程文件中语句 PROCEDURE 后面的过程名中也加上了路径名，这使得应用程序有可能无法正常使用。

由于这些原因，笔者在长城机上用 C 语言编制一了个 FoxBIND 的仿真程序，该程序成功地解决了 FoxBIND 存在的缺陷，程序的执行方式完全与 FoxBIND 格式一样，在功能上完全能够代替 FoxBIND；由于考虑到操作者在使用时经常忘记输入后缀名，故该程序能自动地为你加上后缀名 PRG。

一、程序运行

在 DOS 提示符下，输入格式

PRGBIND < outfile > < infile__1 [< infile__2 > ···[< fnfile__n >]

参数说明：

1.各参数之间以空格分隔，参数可以带通配符“ * ”或“?”，参数可以带后缀名 PRG。也可以不带后缀名；

2. < outfile > ———想要生成的带路径名的过程文件名；

3. < infile__1 > 、< infile__2 > 、···、< infile) n > ———带路径名的各 PRG 命令文件名。

4.本程序在建立过程文件时，只对后缀名为PRG的文件进行处理。

二、程序运行环境

该程序适用于支持 Turbo C1 ·5 或 2 ·0 版本的计算机，具体来说有如下要求：

适用机型：IBM PC、 IBM PC/ XT、 IBM PC / AT、长城机及其它兼容机；

操作系统：DOS2 ·10或CCDOS2 ·10及以上；

该程序在长城 286 EX 及 IBM5550 计算机上调试通过，效果很好，并且只要在一种机器上编译、连接(当然采用 8088 / 8086 指令集)，就可以把该程序的可运行程序(不作任何修改)到其它机型上正确运行。

原程序见附录：PrgBIND · C

附录：PrgBIND · C

```
/ * d BASE__ Ⅲ / FoxBASE+ 过程文件装订程序
        Prgbind * /
#define str __long100
#include < dos · h >
#include < dic · h >
#include < string · h >
```

```
#include <stdio.h>
struct file_path_strings
{
    char path[80];
    char drive[3];
    char dir[66];
    char fname[9];
    char ext[5];
};
main(int argc, char *argv[])
{
 int i1, J, index, find_flag, wild_flag;
 int  file_wild(); *判断文件名中是否有通配符"*"或
      "?",若有返回值1,否则为0 */
char user_string[str_long];
static char *run_message = "PrgBind Utility Rev2.
      00(c)1990 d BASE/Fox Software";
static char *use_help = "Usage: PBGBIND
      <Outfile> <Ijfile>[<Infile>…[<Infile
      >]";
static char *procedure = PROCEDURE";
static char *user_return = "/r/n";
struct file_path_strings  outfile_name, infile_name;
struct file_path_strings *outfile, *infile;
struct ffblk fff;
FILE *in, *out;
void writefile();?  *从输入文件中读信息,然后写到输
      出文件*/
void write_str(); /*向输出文件中写字符串/*
outfile = &outfile_name;
infile = &infile_name;
prinft("%s/n", run_message);
if(argc<3)
{
 printf("/n %s/n", use_help);
 exit(1);
}
strcpy(ouftie->path, argv[1]);
wide_flag = file_wild(outfile);
if(wild_flag)
{
 prinft("Error: %s ---had file name./n", outfile
      ->path);
 exit(1);
}
find_flag = findfirst(outfile->path, &fff, 0);
if(find_flag==0)out = fopen(outfil->path, "a");
else out = fopen(outfile->path, "w");
if(out==NULL)
{
 printf("Error: %s file can 'topen for oufput, /n",
      outfile->path);
 exit(3);
}
for(il=2; ii<argc; il++)
{
strcpy(infile->path, argv[ii]);
wild_fiag = file_wild(infile);
find_flag = findfirst(infile->; ath, &fff, 0);
while(find_flag==0)

  jk=0
  index=0
  while(fff.ff_name[index]!='.')
    {
    infile->fname[index] = fff.ff—name[index];
    ++index;
    }
  infile->fname[index]='/0';
  while(fff.ff_name[index]!='/0')
    {
    infile-ext[jk] = fff.ff_name[index];
    ++jk;
    ++index;
    }
    infile->ext[jk]='/n';
    fnmerge(infile->path, infile->drive, infile->dir
, infile->fname,
```

```
            infile->ext);
        if((in=fopen (infile->paht, infile->path, "r"))
==NULL)
        {
        prinft(Error: %s file can't topen for input。 /n",
infile->path);
        find_flag=findnext(&fff);
        continue;
    }
    printf("binding %s /n,fff·ff_name);
    strcpy(user_string,"******");
    strcat(user_string, infile->path);
    strcat(user_string,"******");
    write_str (user_string·out);
    write_str (user_return·out);
    strcpy(user_string, procedure);
    strcat(user_string, infile->fname);
    write_str (user_string·out);
    write_str (user_return·out);
    writefile (in·out);
    strcpy(user_string, "RETURN/r/n/");
    write_str (user_string·out);
    write_str (user_rsturn·out);
    find_flag=findnext(&fff);
    continue;
    }
    }
    }
    inrfile _wild(struct file_path_strings *tempfile)
    {
    int split_flag;
    int result;
    split_flag=fnsplit (tempfile->path, tempfile->drive,
tempfile->dir, tempfile->fname, tempfile-ext);
    strcpy(tempfile-ext, "·prg");
    if(split_flag&WILDCARDS)resrlt=1;
    else result=0;
    fnmerge (tempfile->path, tempfile->drive
tempfile->dir, tempfile->fname, tempfile->ext);
    return(result);
    }
    void write_str (char*str_pointer, FILE*out)
    {
    while (*str_pointer!='/n')
        {
        putc(*str_pointer·out);
        if(ferror(out))
        {
        printf("Warning: write user_strings error。");
        return;
        ++str_pointer;
        }
    }
    void Writefile(FILE*in, FILE*out)
    }
    char ch;
    while (!feof(in))
      {
      ch=getc(in);
      if(ferror(in))
      {
      printf("Warning: read file-information error!");
      return;  }
      if(!feof(in))putc(ch·out);
      if(ferror(out))
      {
      printf("Warning; write file_information error。)");
      return;
      }
    }
    fclose(in);4
    return;
    }
```