第一部分　DOS 命令扩充

一、硬盘分区表的保存与恢复

1. 原理

硬盘上的分区表位于硬盘的第 1 个扇区，它的完整与否直接关系到机器能否正常运行。计算机病毒经常破坏硬盘分区表，导致系统瘫痪。因此，有必要保存硬盘分区表的内容，以便在硬盘分区表出现故障时，把保存的内容写回到硬盘。具体实现是通过 biosdisk()函数先读出硬盘分区表的内容到缓冲区，然后把缓冲区中的内容写到用 fopen()函数打开的文件中。

2. 程序清单

源程序清单如下：

```
#include <stdio.h>
#include <bios.h>
#include <fcntl.h>
#include <sys\types.h>
#include <sys\stat.h>
void helpmsg(void);
int main(int argc,char *argv[])
{
  int result;
  char buffer[512];
  FILE *fp;
  if(argc==1) helpmsg();
  if(*argv[1]=='b'||*argv[1]=='B')
  {
    result=biosdisk(2,0x80,0,0,1,1,buffer);
   if(! result){
     printf("读硬盘分区表成功\n");
     if((fp=fopen("b:part.doc","wb+"))==NULL)
     {
       fprintf(stderr,"不能创建文件：b:\part.doc\n");
      exit(1);
     }
     fwrite(buffer,1,512,fp);
     fclose(fp);
     printf("硬盘分区表保存成功\n");
     return 0;
   }
```

```
    else {
      fprintf(stderr,"读硬盘分区表失败");
      exit(1);
     }
    }
    if( *argv[1]=='c'||*argv[1]=='C')
    {
       if((fp=fopen("b:\part.doc","rb+"))==NULL)
       {
          fprintf(stderr,"文件打开失败");
         exit(1);
       }
       fread(buffer,1,512,fp);
       result=biosdisk(3,0x80,0,0,1,1,buffer);
       if(! result){
          printf("硬盘分区表恢复成功");
          fclose(fp);
          return 0;
       }
       else{
          fprintf(stderr,"硬盘分区表恢复失败");
          fclose(fp);
          exit(1);
       }
     }
return 0;
}
void helpmsg(void)
{
  puts("程序使用的正确格式为: SAVEPART [B] 或 SAVEPART [C] ");
  puts("其中参数: B-----保存硬盘分区表到 B 盘 ");
  puts("其中参数: C-----从 B 盘恢复硬盘分区表 ");
  exit(0);
}
```

二、给硬盘加软锁

1. 原理

给硬盘加锁的方法很多,但最简单的方法是改变主引导记录或者改变分区表中某些关键字,达到不能使用硬盘的目的。硬盘主引导扇区内容如下:

000H	主引导记录(240字节)
0F0H	全0(206字节)
1BEH	第一分区表(16字节)
1CEH	第二分区表(16字节)
1DEH	第三分区表(16字节)
1EEH	第四分区表(16字节)
	55H AAH

主引导扇区最后两个字节55H和AAH是硬盘自举记录的有效标志。每个分区表为16个字节,内容如下:

	0	1	2	3	
	Boot ind	H	S	CYL	
4	SYS ind	H	S	CYL	7
	Rel seet				11
	# of sects				15

其中Boot ind是自举标志字节,其值为80H时,表示可自举分区;其值为00H时,表示不可自举分区。SYS ind是DOS系统标志字节,其值为01时,表示DOS分区;为00H时,表示未知。H.S.CYL表示分区的起止地址;H是磁头号;S是扇区号;CYL指柱面号的低8位;高2位在S字节的高2位;Rel seet表示该分区的相对扇区号;# of sects表示该分区实用的扇区数。例如,某硬盘的一个分区表为80 00 02 00 01 03 51 30 01 00 00 00 03 51 00 00,第一个字节80H是自举分区标志,第五个字节01是DOS分区标志,若改变01H为00H,可达到加密硬盘效果。下面的程序仅仅修改主引导扇区最后两个字节55H和AAH,修改后若用硬盘启动,则系统提示:Disk Boot Failure,Insert system Disk AND Press Enter。

使用的主要函数如下:

char biosdisk(int cmd,int drive,int head,int track,int sector,int nsects void * buffer)

该函数使用中断ox13,把磁盘操作直接转给BIOS。cmd指示待执行的操作。其中cmd为2时是读盘操作;为3时是写盘操作。drive为0时代表第一个软驱;为1表示第二个软驱,ox80表示第一个硬驱,返回值00时,表示操作成功。

2.程序清单

源程序清单如下:

```
#include <bios.h>
#include <stdio.h>
#include <conio.h>
int main(void)
{
  int result;
```

```
  char buffer[512];
  result=biosdisk(2,0x80,0,0,1,1,buffer);
  if(! result){
    buffer[510]=0x0;
    buffer[511]=0x0;
    printf("读主引导扇区失败! \n");
  }
  if(! result) result=biosdisk(3,0x80,0,0,1,1,buffer);
  (! result) ? (printf("写主引导扇区成功! \n")):(printf("写主引导扇区失败! \n"));
  return 0;
}
```

三、外设的软锁和解除

1. 原理

当 IBM PC 系列机启动后，ROM BIOS 进行自诊断测试，硬件设备配置情况和操作结果存放在 ROM BIOS 的系统参数区 0040：0000 开始的 256 个 RAM 单元中，这些数据非常重要，控制着机器的许多操作。虽然这些数据在设计时仅供 ROM BIOS 使用，但是用户程序也可读取这些数据来了解机器状态，某些数据也可以修改，以便对机器进行控制。如 0040：0017 单元为换档状态字节 RAM 区，其中：

位	值	意义
0	置位	右边的 shift 键按下
1	置位	左边的 shift 键按下
2	置位	ctrl 键按下
3	置位	AlT 键按下
4	置位	scroll 键开关处于开状态
5	置位	NumLock 开关处于开
6	置位	CapsLock 处于开
7	置位	Ins 键按下

又如 0040：0013 单元给出了计算机系统的实际内存容量(病毒程序往往把 280H(640K)变更为 27EH(638K)或 27DH(637K))。0040：0008 单元为 1 号打印机基地址，0040：0075 为硬盘驱动器号。键盘数据缓冲区的地址为 0040：001E—0040：003D，改变该数据区内容，可以达到程序的自动演示。ROM BIOS 参数区的具体内容可参见附录(二)。

2. 应用实例

(1)每次启动 AT 或 386 计算机，Numlock 模式打开，需要人工关闭，这比较麻烦，因此通过改写 0040：0017 单元的值来达到软关闭 Numlock 目的。程序清单如下：

```
#include <stdio.h>
#include <string.h>
```

```
#include <dos.h>
char far *ptr;
void main(int argc,char *argv[])
{
  ptr=(char far *)MK_FP(0x0040,0x0017); /*指针 ptr 指向 0040:0017 单元*/
  printf(" 改写 ROMBIOS 参数区来开关 NUMLOCK 键\n ");
  if(argc==1){
    printf(" 程序正确用法为:\n ");
    printf("Numlock 01 打开 NUMLOCK 键\n ");
    printf("Numlock 00 关闭 NUMLOCK 键\n ");
  }
  if(! strcmp(argv[1],"01")) *ptr|=0x20;
  if(! strcmp(argv[1],"00")) *ptr&=0xdf;
}
```

(2)改写 0040：0790 单元的内容可软加锁 A 驱动器;改写 0040：07E2 单元的内容可软加锁 B 驱动器;改写 0040：0075 单元的内容可软加锁 C 驱动器。软加锁后,该驱动器的读写命令指示灯不亮,且显示:General Failure error reading drive A(B,C)。

Abort,Retrl,Fail? 解锁可在相应单元处赋予该单元原有的值即可。具体程序如下:

```
#include <stdio.h>
#include <dos.h>
char far *ptr;
void main(void)
{
  int j;
  ptr=(char far *)MK_FP(0x0040,0x0790); /*指针 ptr 指向 0040:0790 单元*/
  printf(" 改写 ROMBIOS 参数区来软加锁\n ");
  printf("1－－－－－－软加锁 A 驱动器－－－－－－\n");
  printf("2－－－－－－软加锁 B 驱动器－－－－－－\n");
  printf("3－－－－－－软加锁硬盘 －－－－－－\n");
  printf("4－－－－－－开锁 B 驱动器 －－－－－－\n");
  printf("5－－－－－－开锁 C 驱动器 －－－－－－\n");
  printf("－－－－－－－其它键退出－－－－－－－－－－－－\n");
  scanf("%d",&j);
  switch(j){
    case 1:
      ptr=(char far *)MK_FP(0x0040,0x0790);
      *ptr++=0x0;
      *ptr=0x0; /* *ptr=0168 可解锁 360K 软盘*/
      break; /*ptr=01d0 可解锁 1.2M 软盘*/
    case 4:
```

```
        ptr=(char far *)MK_FP(0x0040,0x07e2);
        *ptr++=0x68;
        *ptr=0x01; /* *ptr=0168 可解锁 360K 软盘 */
        break;
      case 2:
        ptr=(char far *)MK_FP(0x0040,0x07e2);
        *ptr++=0x0;
        *ptr=0x0; /* *ptr=0168 可解锁 */
        break;
      case 3:
        ptr=(char far *)MK_FP(0x0040,0x0075);
        *ptr=0x0; /* *ptr=8001 可解锁 */
        break;
      case 5:
        ptr=(char far *)MK_FP(0x0040,0x0075);
        *ptr++=0x01;
        *ptr=0x80; /* *ptr=8001 可解锁 */
        break;
      default:
        break;
    }
}
```

(3)改变 0040:0008 单元的内容可软加锁打印机,软加锁后,打印机将不能打印。无论是否联机或有无纸,系统在进行打印操作时,显示:

No paper error writing devic PRN Abort,Retrh,Fail?。

解锁时可把 0040:0008 单元的原有值写回(该值对于不同的系统可能不同)。

程序清单如下:

```
#include <stdio.h>
#include <string.h>
#include <dos.h>
char far *ptr;
void main(int argc,char *argv[])
{
  ptr=(char far *)MK_FP(0x0040,0x0008); /*指针 ptr 指向 0040:0008 单元*/
    printf(" 改写 ROMBIOS 参数区来软加锁打印机\n ");
    if(argc==1){
      printf(" 程序正确用法为:\n ");
      printf("Printlock 01 开锁打印机\n ");
      printf("Printlock 00 加锁打印机\n ");
    }
```

```
    if(! strcmp(argv[1],"01")){
       *ptr++=0x78;
       *ptr=0x03;
    }
    if(! strcmp(argv[1],"00")){
       *ptr++=0x0;
       *ptr=0x0;
    }
}
```

四、DIR 功能扩充

1. 原理

在 DOS 系统中，DIR 命令仅能显示当前目录下的可见文件，不能显示隐含文件或子目录下的文件，因而使用不方便。下面的程序利用 findfirst()和 findnext()函数连续查找各级子目录和各子目录下的文件，并能显示出隐含文件和子目录。使用方法如下：

①键入"DIRD　盘符:"时，递归显示指定磁盘的各级子目录和各子目录下文件名，并在隐含文件名后显示"*"号，与普通文件匹别，在子目录后则显示一个"\"号。

②键入"DIRD　盘符:文件名"时，将作为查询该文件的全路径名使用，当查找到与指定文件名匹配的文件后，将显示该文件的全路径名和其它属性。

③键入"DIRD　盘符:*·扩展名"时，将递归显示各级子目录下指定扩展名的文件的全路径名和其它属性。

程序使用的主要函数如下

- int findnext(struct ffblk *ffblk)

  取得下一个匹配 findfirst 模式的文件

- char *getcwd (char *buf,int n

  取当前的工作目录的完整路径名，最长为 n 个字节，并把它存于缓冲区的 buf 中

- void getdfree (int drive,struct dfree *dfreep)

  该函数接受磁盘驱动器号 drive，并把磁盘特性填入由 dfreep 所指的 dfree 结构中。dfree 结构定义如下：

```
struct dfree{
unsigned      df_avail      /*可用簇*/
unsigned      df_total      /*全部簇*/
unsigned      df_base       /*每扇区字节*/
unsigned      df_sclus      /*每簇扇区*/
```

- int chdir (char *path)

该函数使得 path 指定的目录变为当前工作目录。

2. 程序清单

源程序清单如下：

```
#include <dos.h>
#include <dir.h>
```

```
#include <string.h>
#include <conio.h>
void help_message(void);
void main(int argc,char *argv[])
{
  struct ffblk ffblk;
  int do1,d2;
  char path[80]="\0",path1[80][80],*p;
  int level=0,d1=0,d3=0;
  path[0]='A'+getdisk();
  if(argc==1) help_message();
  if(argv[1][1]==':'){ /*处理盘符:*/
    path[0]=argv[1][0];
    argv[1]+=2;
    }
  if((p=strchr(argv[1],'*'))==argv[1]){ /*处理盘符:*.扩展名*/
    argv[1]+=2;
    strupr(argv[1]);
   }
  strcat(path,":");
  do{
    if(! (strlen(argv[1])>0)) printf("\n%-s\n",path);
    d1=0;
    strcat(path,"\\");
    strcpy(path1[d3],path);
    strcat(path,"*.*");
    do1=findfirst(path,&ffblk,23);
    while(! do1){
      if(stricmp(ffblk.ff_name,".") && stricmp(ffblk.ff_name,"..")) {
        if(argv[1]==NULL) /*处理盘符:*/
          printf("\n%s%s %1d\n",path1[d3],ffblk.ff_name,ffblk.ff_fsize);
        else /*处理盘符:文件名*/
          if(! stricmp(ffblk.ff_name,argv[1]))
          printf("\n%s%s %1d\n",path1[d3],ffblk.ff_name,ffblk.ff_fsize);
          /*处理盘符:*.扩展名*/
        if((p=strstr(ffblk.ff_name,argv[1])) && (*(--p)=='.'))
          printf("\n %s%s %1d",path1[d3],ffblk.ff_name,ffblk.ff_fsize);
        if((ffblk.ff_attrib & 0x10)==FA_DIREC) {
          strcpy(path1[d3+1],path1[d3]);
          strcat(path1[d3++],ffblk.ff_name);
```

```
            }
            if(! strlen(argv[1])) {
              printf("%-s",ffblk.ff_name);
              if((ffblk.ff_attrib&0x10)==FA_DIREC)
                printf("\\");
              else ((ffblk.ff_attrib & 0x02)==FA_HIDDEN) ? printf("*");printf
("");
            for(d2=1;d2<(13-strlen(ffblk.ff_name));++d2)
              printf("");
            if(++d1>5) {
              printf("\n");
              d1=0;
            }
            }
          }
          do1=findnext(&ffblk);
        }
        strcpy(path,path1[level++]);
      }while(level<=d3);
    }
    void help_message(void)
    {   printf("\n ---- 程序的正确用法如下----");
        printf("\n ---- DIRD 盘符: ----");
        printf("\n ---- DIRD 盘符:文件名 ----");
        printf("\n ---- DIRD 盘符: *.扩展名 ----");
        exit(1);
    }
```

五、type 命令扩充

1. 原理

DOS 提供的 type 命令不能分屏显示，若分屏显示，必须使用大多数用户不太熟悉的管道操作，即 type 文件名|more 命令。下面的程序通过调用 BIOS 的中断功能(int 10H)来实现分屏显示。

int int86(int intr—num,union REGS * inregs,union REGS * outregs)

该函数执行由参数 intr—num 指定的 8086 软中断。执行前，把 inregs 中的寄存器值拷贝到各寄存器中；返回时，把当前寄存器的值拷贝到 outregs 中。

2. 程序清单

源程序清单如下：

```
#include <stdio.h>
#include <dos.h>
```

```
#include <process.h>
void main(int argc,char *argv[])
{
   FILE *fp;
  char ch, *filename;
  int row;
  union REGS in,out;
  if(argc! =2){
    printf("\nUsage:TYPE filename\n");
    exit(1);  }
  filename=argv[1];
  fp=fopen(filename,"r");
  clrscr();
  gotoxy(1,1);
  row=0;
  while((ch=fgetc(fp))! =EOF){
    in.h.ah=3;
    in.h.bh=0;
    int86(0x10,&in,&out);
    if(out.h.dh! =row){
      if(out.h.dh<=23){
        row=out.h.dh;
        gotoxy(1,row+1);
      }
      else{
      gotoxy(37,25);
      printf("--more--");
      getch();
      clrscr();
      gotoxy(1,1);
      }
    }
    in.h.ah=14;
    in.h.al=ch;
    in.h.bh=0;
    in.h.bl=7;
    int86(0x10,&in,&out);
  }
  fclose(fp);
}
```

六、修改子目录名

1. 原理

DOS 提供给用户目录操作的命令有:建立目录,进入目录,删除目录等。而要给子目录改名必须使用工具软件(如 PCTOOLS 等),下面程序可实现子目录改名。具体算法是:首先确定要改名的文件是子目录(通过—chmod()函数),再通过 rename()函数重命名文件。

使用的主要函数如下

- int rename(char * oldname,char * newname)

  该函数把老文件名 oldname 改为新文件名 newname。

- int _chmod (char * filename,int func[,int attrib]) 该函数改变文件的存取方式。若 func 为 0,本函数返回文件的当前 MS DOS 属性;若 func 为 1,则把文件属性置为 attrib 值。attrib 可以为下列符号常量之一(在 dos.h 中定义):

FA_RDONLY	只读
FA_HIDDEN	隐含
FA_SYSTEM	系统

2. 程序清单

源程序清单如下:

```
#include <dos.h>
#include <stdio.h>
#include <io.h>
void main(int argc,char *argv[])
{
  int state;
  if(argc! =3){
    printf("%s\n","error command");
    printf("%s\n","rendir oldname newname");
    exit(1);
  }
  state=_chmod(argv[1],0);
  if(state! =FA_DIREC)
    {
    printf("%s\n","not is directory");
    exit(1);
  }
  state=rename(argv[1],argv[2]);
  if(state==-1) /*Changing sucessfully*/
    puts("Changing error");
  else /*Changing badly*/
    puts("Changing ok!");
}
```

七、子目录删除

1. 原理

在DOS系统中，每一个子目录下往往有很多子目录及文件，在用RD命令删除目录时，必须首先把该子目录下的所有文件及目录删除后，才能返回该子目录的上级目录，再删除它，这样非常麻烦。因此，编写了下面的RD命令，实用性更大。具体办法为：首先从根结点开始往下查找，遇到文件就删除该文件，遇到目录就进入该目录，当找到树叶时则删除它。之后，再从根结点查找，依次下去，直至再也没有结点为止。然后进入该目录的上级目录并删除此目录。使用的主要函数如下

- int getcurdir(int drive,char * direc)

  该函数取指定驱动器drive的当前工作目录。

- int rmdir(char * pathname)

  该函数删除pathname所指定的目录。

- int chmod(char * filename,int permiss)

  该函数改变文件的存取方式，其中pemiss的值定义如下：

S—IWRITE	允许写
S—IREAD	允许读
S—IWRITE |S—IREAD	允许读和写

- int unlink(char * filename)

  该函数删除由filename所指定的文件。

- char * strupr(char * str)

  该函数将串str中的小写字母转换为大写字母。

2. 程序清单

源程序清单如下：

```
#include <stdio.h>
#include <dir.h>
#include <dos.h>
#include <io.h>
#include <sys\stat.h>
#include <string.h>
#include <stdlib.h>
void pushdir(void);
struct ffblk f;
int done;
int attrib;
char deldir[MAXDIR], rootdir[MAXDIR], del[MAXDIR], dir[MAXDIR], mid
[MAXDIR];
void main(int argc,char *argv[])
{
  if(argc! =2){
```

```
        printf("This rountine can remove a directory ! \n");
        printf("example:\n"); /*正确用法如下:*/
        printf(" rmd dirname\n"); /*rmd 子目录名*/
        exit(1);
        }
      strcpy(deldir,argv[1]);
      strupr(deldir);
      getcwd(rootdir,MAXDIR); /*取当前根目录名*/
      if(chdir(deldir)) {
        printf("No this path\n");
        exit(0);
        }
      pushdir();
    }
    void pushdir(void)
    {
      done=findfirst("*.*",&f,FA_RDONLY|FA_HIDDEN|FA_DIREC|FA_SYS-
TEM|FA_ARCH);
      while(! done){
        if(f.ff_attrib! =16){/*删除文件*/
          chmod(f.ff_name,S_IREAD|S_IWRITE);
          unlink(f.ff_name);
          printf("%-12s is deleted.\n",f.ff_name);
        }
      if(f.ff_attrib==16 && strcmp(f.ff_name,".") && strcmp(f.ff_name,".."))
      { /*是不是子目录*/
        getcwd(dir,MAXDIR);
        if(! chdir(f.ff_name)) strcpy(del,f.ff_name); /*进入子目录*/
        else{
        printf("exit\n");
        exit(0);
          }
        pushdir();
        }
        done=findnext(&f); /*找下一个匹配文件*/
      }
      getcurdir(0,mid); /*取当前目录名*/
      if(! strcmp(mid,deldir)){ /*判断是不是删除目录*/
        chdir(rootdir); /*进入根目录*/
        rmdir(deldir); /*删除目录*/
```

```
    printf("* * * The End ! * * * \n");
    exit(0);
  }
  chdir(dir);
  if(! rmdir(del)) /* 删除子目录 */
    printf("Remove %s ! \n",del);
  else
    printf("No remove %s! \n",del);
    chdir(rootdir); /* 进入根目录 */
    chdir(deldir); /* 进入删除目录 */
    pushdir(); /* 从根目录重新开始 */
}
```

八、修改文件名(或子目录名)的属性

1. 原理

修改文件(或子目录)的属性,往往使用 pctools 工具软件,但有时隐含了子目录名(目录项最后两字节改为非零值),pctools 也不能发现,更谈不上修改。本程序利用 findfirst 和 findnext ()两函数连续查找匹配的文件(子目录),再利用_chmod()函数修改文件(子目录)的属性。

2. 程序清单

源程序清单如下:

```
#include <stdio.h>
#include <dir.h>
#include <conio.h>
void helpmsg(void);
void disp(void);
void modi(char *s);
void str(char *s,int att);
int at;
char s1[30],s2[30];
struct ffblk f;
char *a[6]={"只读","隐型","系统","卷标","目录","读写"};
void main(int argc,char *argv[])
{
  if(argc==1) helpmsg();
  else if(findfirst(argv[1],&f,31))
    printf("%s 文件(目录)没找到! \n",argv[1]);
  else if(argc>2 && argv[2][1]=='\0' && (at=argv[2][0]-'0')>=0 && at<8)
    modi(argv[1]);
  else
    disp();
```

```
}
void str(char *s,int att) /*生成属性字符串*/
{
  int i,j;
  *s='\0';
  for(i=j=0;i<5;i++)
    if(att&(1<<i)){
      if(j&&i!=4) strcat(s,"+");
      strcat(s,a[i]);
      j++;
    }
  if(!j) strcat(s,a[i]);
}
void modi(char *s) /*修改属性*/
{
  if(_chmod(s,1,f.ff_attrib&32|at)==-1)
    printf("%s 文件(目录)属性修改失败!\n",s);
  else{
    str(s1,f.ff_attrib);
    str(s2,f.ff_attrib&16|at);
    printf("\n%s 属性已由 %s 改为 %s!\n",s,s1,s2);
  }
}
void disp(void) /*显示属性*/
{
  printf("%-16s%-24s","文件名---","属性---");
  printf("%-16s%-24s","文件名---","属性---");
  do {
      str(s1,f.ff_attrib);
      printf("%-16s%-24s",f.ff_name,s1);
  }while(!findnext(&f));
  printf("\n");
}
void helpmsg(void) /*帮助信息*/
{
  printf("\n 修改或显示子目录和文件属性");
  printf("\n 程序使用格式 :ATTD[文件(目录)名][属性编号]");
  printf("\n [属性编号]取值 [0-7] 时,将文件(目录)属性改为指定属性.");
  printf("\n 否则显示文件(目录)的属性,这时,[文件(目录)名]可用统配符.");
  printf("\n [属性编号]含义如下:");
```

```
    printf("\n\t 0--读写 1--只读 2--隐型 3--只读+隐型");
    printf("\n\t 4--系统 5--只读+系统 6--隐型+系统 7--只读+隐型+系统");
}
```

九、文件移动

1. 原理

DOS 操作系统没有提供给用户一个类似 XENIX 系统中移动文件的命令，因而树形文件系统中在移动一个文件时，必须首先拷贝文件到指定目录，再删除该目录下的原文件，使用较麻烦。因此，编写一条移动文件的命令，作为 DOS 外部命令使用。

使用的主要函数如下：

- int remove(const char * filename)

  该函数删掉由 filename 所指定的文件，它是一个宏。

- int fnsplit(char * path,char * drive,char * dir,char * name,char * ext)

  把一个完整路径名分解为各组成成份。

用法示例：如把 bin 子目录下所有以.bak 为扩展名的文件移到 CCED 子目录下，可以键入：move c:\borLandc\bin*.bak　　c:\CCED*.bak

2. 程序清单

源程序清单如下：

```
#include<dir.h>
#include<string.h>
#include<stdio.h>
#include<alloc.h>
void getname(char str[30]);
void getcd(char arg1[],char arg2[]);
int fnum,value;
char fname[1000][30];
char *subspath[2];
char drive[MAXDRIVE];
char dir[MAXDIR];
char file[MAXFILE];
char ext[MAXEXT];
int flag;
void getname(char str[30])
{
  struct ffblk ff;
  int j=0,done=findfirst(str,&ff,0);
  while(! done){
    j++;
    strcpy(fname[j],ff.ff_name);
```

```
        done=findnext(&ff);
      }
      fnum=j;
    }
    void getcd(char arg1[],char arg2[])
    {
      int i;
      for(i=0;i<2;i++)
        subspath[i]=(char malloc(sizeof(char)*80);
        fnsplit(arg1,drive,dir,file,ext);
        strcpy(subspath[0],drive);
        strcat(subspath[0],dir);
        strcat(subspath[0],"\0");
        if(arg2){
          flag=fnsplit(arg2,drive,dir,file,ext);
          strcpy(subspath[1],drive);
          strcat(subspath[1],dir);
        }
        else
          getcwd(subspath[1],80);
        if(subspath[1][strlen(subspath[1])-1]!='\\')
          strcat(subspath[1],"\\");
        strcat(subspath[1],"\0");
    }
    void main(int argc,char *argv[])
    {
      int k;
      char source[40],dest[40];
      if(argc<2){
        printf("\nUsage: [d:][path]MV [[path]oldfilename [path]newfilename]");
        return;
      }
      else{
        getname(argv[1]);
        getcd(argv[1],argv[2]);
      }
      for(k=1;k<fnum+1;k++){
        strcpy(source,subspath[0]);
        strcat(source,fname[k]);
        strcat(source,"\0");
```