# 第一部分　计算机系统的应用与控制技术

## 1.1　怎样获取五笔字型的词组表

众所周知，金山汉字操作系统的五笔字型汉字输入法中有词组功能。那么，到底有多少条词组？又有哪些词组？

词组表存于文件 WBX.COM 中，现以长度 47695 字节的 WBX.COM 为例，具体说明获取词组表的方法：

(1)用 DEBUG 或 PCTOOLS 等调试工具(最好是已汉化的，直观一些)找到词组表的开头(菜场)和结尾(为人民服务)的位置，并生成新的词组表文件为 CZB.BAK。

```
C)debug WBX.COM↙
-D 60F4↙                          ;观察开头
-D B2D0↙                          ;观察结尾
```

由于“菜”字位于 CS:60F4，“务”字位于 CS:B2D9 处，因此，词组表的总长度＝B2D9－60F4＋1＝51E6 字节。

```
-M 60F4,L51E6 100↙                ;将字词组表移至内存 CS:100 处
-N CZB.BAK↙
-R CX↙
CX BA4F
: 51E6↙
-W↙
-Q↙
```

至此，一个词组表文件 CZB.BAK 已生成。

(2)用 BASIC 程序将词组外的信息滤去，生成文本文件 CZB.TXT。程序清单如下：

```
10 CLS
20 OPEN "R",1,"czb.bak",2
30 FIELD #1,2 AS R$
40 OPEN "R",2,"czb.txt",2
50 FIELD #2,2 AS W$
60 I=1
70 IF EOF(1) THEN GOTO 170
80 GET #1,I
90 I=I+1
100 IF ASC(LEFT$(R$,1))<176 OR ASC(LEFT$(R$,1))>247 THEN PRINT" ";:GOTO 140
110 LSET W$=R$
120 PRINT R$;
130 GOTO 150
140 LSET W$=CHR$(32)
150 PUT #2,I-1
160 GOTO 70
170 CLOSE
180 END
```

运行以上程序，将生成一张“五笔字型”词组表，可供任意编辑使用。顺便提一句，其他类似的编码表也可以如此获取。至此，有多少条词组、有哪些词组就一目了然了。还可以知道，在词组表中，最长的词组并不是“中华人民共和国”，而是“五笔字型计算机汉字输入技术”。

(张毅)

## 1.2 应用OOP技术实现汉字显示控制

在进行一项专业软件系统开发(如结构设计CAD)时,常常需要有汉字显示的控制器。但该系统的核心部分(如有限元分析)不仅对内存需求庞大(包括了Expanded & Extented Memory),而且有用字相对集中的特点。此时若采用商品汉字系统方案就有些不适宜。因为这些商品化的汉字处理系统的设计目标是通用型的,在费用、速度和容量上无法取得平衡。笔者在实践中用OOP技术实现了汉字显示控制器CCOC(Chinese Character Output Controller),初步解决了这个问题。

### 1. CCOC原理

CCOC可以划分成读取字模和显示汉字两大部分。在此主要讨论读取字模。

读取字模设三级字模库(过程见图1.2.1):CACHE库(CL)、弹性缓存库(FL)和汉字库(CCLIB)。当一个汉字2字节的机内码进入CCOC时,系统首先根据这个码在CL中查找。CL是一个顺序表,它存放着特定专业系统中使用高频字的机内码及其字模,但其规模比较小,一般为5～20字。若在CL中没有找到,系统即转入第二级,在FL中查找。FL是CCOC中的主要部分,它为一个链式结构(见图1.2.2),每一次CCOC被激活后,FL均进行一次调整:将汉字按LRU算法(Least Recently Used)排列,即把所选中的汉字(CL中除外)插入FL的最上端,若此时FL发生溢出,则淘汰最下端汉字。经过一定时间的运行后,在FL高区的汉字属高频字。因此查找效率会不断上升。如果待显字机内码在CL、FL中均未找到,则转到最后一级,在CCLIB中读取字模,同时按LRU算法将字模插入FL。

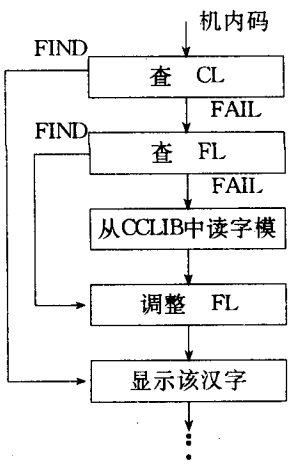


图1.2.1 CCOC主要逻辑图

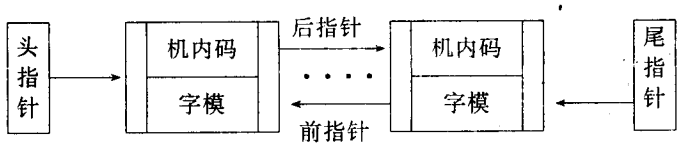


图1.2.2 弹性缓存库(FL)结构

### 2. 程序实现及说明

CCOC是在TURBO C++ 1.0上实现的。由于利用了面向对象程序设计语言(OOPL)的封装、抽象特性,使得CCOC能独立于系统其他部分,工作不受外界影响,具有很高的可靠性和透明度。

程序一和程序二分别为CCOC的头文件与主程序,使用时,首先引用"CCOC.H"中的类定义:CCOCOBJ,然后定义一个控制器对象(ccocobj),在定义对象的同时,控制器将自动读入一个名为

charcode.lib 的自带库(见图 1.2.3),此文件有两种形式:第一种是内码集合,一般由用户最初定义,前 MAXCACHE(在 CCOC.H 中定义)个汉字即为 CL 中的内容,这个文件可以由中文字处理软件按"程序编辑"方式建立,控制器将根据其机内码从 CCLIB 中读入相应字模;第二种是字典形式,这种形式的库是由 CCOC 建立的,它可使整个系统完全脱离 CCLIB 工作(当全部所需汉字均在 charcode.lib 中时)。在控制器对象初始化时这两种形式会被自动识别读入。

CL 与 FL 的大小由 MAXCACHE 和 MAXHZ 来控制,其数值由具体的应用条件所决定,一般 MAXCACHE 控制在 5～20 字,MAXHZ 控制在 200～600 字,太大了会降低查码效率。

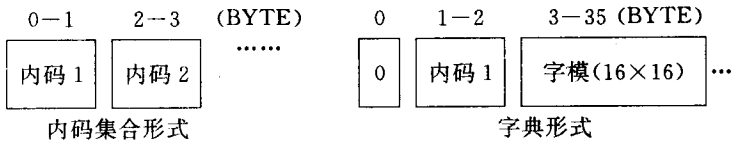


图 1.2.3 自带库的两种形式

在需要显示汉字串的地方可使用函数:

```
控制器对象名, couttextxy(X 坐标,Y 坐标,"字串");
```

字串可中英文混合,输出的汉字具有窗口字裁剪功能,在程序结束处,可使用函数:

```
控制器对象名,savelib(mode);        mode=CODE   以内码方式存库
                                    mode=LIB    以字典方式存库
```

这个函数用来保留内存字库(CL、FL),可供系统下次使用。程序三是一示范程序,使用时需要先建立工程文件(project)后编译。

本文所介绍的汉字显示控制器(CCOC)在一些专用的环境能发挥很大的作用。表 1.2.1 列出了作者利用程序三中间循环显示部分作的各种情况的耗时比较。

**表 1.2.1 时间测定比较**①

| | 循环显示 10 次(110 个汉字)耗时 |
|---|---|
| 假定所用字已在自带库 | 3″62 (最好情况) |
| 假定所显字总不在内存 | 13″40 (最差情况) |
| 所显字初始不在自带库 | 4″95 (一般情况) |
| 直接读硬盘字库 | 12″81 |

①使用 386SX/16 主机,CONNER CP30104 硬盘。

由上表可知,CCOC 就是在最坏情况下(110 个汉字都不一样,且又不在自带库中时),也只比普通直读 CCLIB 慢不到 1 秒,但在更一般情况下,CCOC 则比它快近 2.5 倍。因此,CCOC 集中了直读字库的节省内存与读内存字库的高速度,做到了同时兼顾时间与空间上的要求。当系统所需汉字的重复率越大时,CCOC 的效率也就越高。

当然,本文所列的程序并非是最优的,读者完全可以根据需要加入其他功能,以适应更高的系统要求。此外,虽然 CCOC 的语言背景是 C++,但可通过混合编程来达到悬挂至其他高级语言上去的目的。

## 程序一 （CCOC. H 头文件）

```
//Head of CCOC(Chinese Character Output Controller)program
#ifndef CCOCH
#define CCOCH
#include <dos. h>
const int MAXHZ=400;//        弹性库最大汉字数
const int MAXCACHE=10;//      CACHE 库最大汉字数
const int LIB=1,CODE=0;//     存库模式
const int CHARGAP=2;//        字距
struct CCOC {
  char *matrix;//             字模
  char internalcode[2];//     内码
  CCOC *backptr;//            后指针
  CCOC *nextptr;//            前指针
};
//定义类
class ccocobj {
  CCOC *head, *tail,CACHE[MAXCACHE];
  int number;//       当前弹性库字数
  int intc;
public:
  ccocobj(void);//    初始化汉字显示控制器
  void couttextxy(int xloc,int yloc,char *string);////显示 汉字/字符 字串
  int savelib(int mode=LIB);//                    保存当前内存中汉字库
private:
  void coutxy(int xloc,int yloc,char *m);//       按当前色构成一个汉字
  char *readlib (char *m); //                     从 CCLIB 中读字模
  char *readqlib (FILE *fmp);//                  从自带库中读字模
  char *getmatrix (char *m);//                   取得汉字字模
};
#endif ]
```

## 程序二 （CCOC. CPP 主程序）

```
//Program of Chinese Character Output Controller (CCOC)
#include <graphics. h>
#include <alloc. h>
#include <dos. h>
#include <stdio. h>
#include "ccoc. h"
const char *cclibname="HZK16";
const char *qlibname="CHARCODE. LIB";
ccocobj::ccocobj(void){//       初始化汉字显示控制器
  int i;
  int max; //                   盘上自带字库汉字总数
  int length;
char flag;//                   自带库的类型标志.   1:带内码和字模
      //                                          0:仅带内码
FILE *qlib;
CCOC *last, *now;
```

```
head=NULL;
number=0;
intc=sizeof(char);
if (coreleft()<(MAXHZ+MAXCACHE) * 42||(qlib=fopen(qlibname,"rb"))==NULL)
{   number=-1;return; }
//确定盘库中汉字总数
fread(&flag,intc,1,qlib);
fseek(qlib,01,SEEK_END);
if (flag&0x80) { flag=0;
                 max=ftell(qlib)/2;}
    else        { flag=1;
                 max=(ftell(qlib)-1)/34;}
fseek(qlib,(long)flag,SEEK_SET);
if (max<MAXCACHE+1)   { fclose(qlib); return;}
//读入 CACHE 库
for (i=0;i<MAXCACHE;i++)  {
   fread(CACHE[i].internalcode,intc,2,qlib);
   if ( ! flag) CACHE[i].matrix=readlib(CACHE[i].internalcode);//Read from CCLIB
     else       CACHE[i].matrix=readqlib(qlib);//Read from QLIB
   }
//读入弹性缓存库 FL
head=new CCOC;
last=NULL;
now=head;
length=(max-MAXCACHE)<=MAXHZ ? (max-MAXCACHE):MAXHZ;
for(i=0;i<length;i++) {
  fread(now->internalcode,intc,2,qlib);
  if ( ! flag)   now->matrix=readlib(now->internalcode);
    else         now->matrix=readqlib(qlib);
  now->backptr=last;
  now->nextptr=new CCOC;
  last=now;
  now=now->nextptr;
  number++;
  }
delete(last->nextptr);
last->nextptr=NULL;
tail=last;
fclose(qlib);
}
char * ccocobj::getmatrix (char *m)//  取得汉字字模
{
  char *temp,buffer[2];
  int i;
  CCOC *ptr;
  //在 CACHE 中查找
  for (i=0;i<MAXCACHE;i++)
    if (CACHE[i].internalcode[0]==m[0]&&
        CACHE[i].internalcode[1]==m[1]) return(CACHE[i].matrix);
  //在弹性缓存库中查找
  if (head==NULL) {
                        head=new CCOC;
```

```
head->nextptr=NULL;
tail=head;
head->internalcode[0]=0;
}
ptr=head;
while(ptr!=NULL) {
  if (ptr->internalcode[0]==m[0] && ptr->internalcode[1]==m[1]) {
                if (ptr!=head)     {  //找到.调整 FL
                  if (ptr!=tail) {
                            ptr->nextptr->backptr=ptr->backptr;
                            ptr->backptr->nextptr=ptr->nextptr;}
                  else {
                            tail=ptr->backptr;
                            tail->nextptr=NULL;}
                  ptr->backptr=NULL;
                  ptr->nextptr=head;
                  head->backptr=ptr;
                  head=ptr;
                  }
                return( ptr->matrix);
                }
ptr=ptr->nextptr;
}
//内存中没有,从盘上 CCLIB 中装载
temp=readlib(m);
//加入 FL 中
ptr=new CCOC;
ptr->matrix=temp;
ptr->internalcode[0]=m[0];
ptr->internalcode[1]=m[1];
ptr->backptr=NULL;
ptr->nextptr=head;
head->backptr=ptr;
head=ptr;
if (number)=MAXHZ) {
                          tail=tail->backptr;
                          delete(tail->nextptr);
                          tail->nextptr=NULL;
                          }
    else
          number++;
return(temp);
}
char * ccocobj::readlib(char *m)// 从 CCLIB 中读字模
{
  long order;
  char *buffer;
  FILE *filename;
  order=(long)32*( ((m[1]) &0x007f)-0x21 +94*((m[0]&0x007f)-0x30)+15*94);
  if ((filename=fopen(cclibname,"rb"))==NULL) return(NULL);
  fseek(filename,(long)order,SEEK_SET);
  buffer=(char *)malloc(32*intc);
```

```
  fread(buffer,321,1,filename);
  fclose(filename);
  return(buffer);
}
char *ccocobj::readqlib(FILE *fmp)//从自带库中读字模
{
  char *buffer;
  buffer=(char *)malloc(32*intc);
  fread(buffer,321,1,fmp);
  return(buffer);
}
void ccocobj::couttextxy(int xloc,int yloc,char *string)
//显示 汉字/字符 字串
{
  char *c;
  int maxX,maxY;
  struct viewporttype view;
  char *sptr,m[2];
  if (number<0 || coreleft()<(MAXHZ-number)*42)  return;
  getviewsettings(&view);
  maxX=view.right-view.left;
  maxY=view.bottom-view.top;
  sptr=string;
  if (yloc+16>maxY) return;
  while(*sptr!='\x0') {
    if (*sptr&0x80) {
        m[0]=*sptr; m[1]=*(sptr+1);
        c=getmatrix(m);
        if (xloc+16<maxX) coutxy(xloc,yloc,c);
          else return;
        sptr+=2;
        xloc+=(CHARGAP+16);
        }
      else  {
        m[0]=*sptr;  m[1]='\0';
        if (xloc+8<maxX) outtextxy(xloc,yloc+8,m);
          else  return;
        sptr++;
        xloc+=(CHARGAP+8);
        }
  }
}
void ccocobj::coutxy(int xloc,int yloc,char *m)//按当前色构成一个汉字
{
  long b;
  int color,i,j,l,k=0;
  color=getcolor();
    for(i=0;i<32;i+=2,k+=2) {
      b=((unsigned char) m[k]<<8)+(unsigned char) m[k+1];
      for (j=0;j<16;j++,b<<=1)
        if (b&0x8000) putpixel(xloc+j,yloc+i/2,color);
      }
```

```
}
int ccocobj::savelib(int mode)//保存当前内存中汉字库
{
  int i=0;
  FILE *qlib;
  CCOC *ptr;
  if (number<0 ||!(qlib=fopen(qlibname,"wb"))) return(0);
  if (mode==LIB) fwrite(&i,intc,1,qlib);
  for  (i=0;i<MAXCACHE;i++) {
    fwrite(CACHE[i].internalcode,intc,2,qlib);
    if (mode==LIB) fwrite(CACHE[i].matrix,intc,32,qlib);
    }
  ptr=head;
  while(ptr) {
     fwrite(ptr->internalcode,intc,2,qlib);
     if (mode==LIB) fwrite(ptr->matrix,intc,32,qlib);
     ptr=ptr->nextptr;
     }
  fclose(qlib);
  return(1);
}
```

## 程序三 （DEMO.CPP 示范程序）

```
//An example of the usage of CCOC
//本程序的编译模式为大模式
//需链接 DEMO.CPP  CCOC.CPP  EGAVGAF.OBJ  CGAF.OBJ
#include <graphics.h>
#include <stdio.h>
#include <conio.h>
#include <time.h>
#include "ccoc.h"
ccocobj cchar;//定义控制器对象
main()
{
  int i,j,color=1;
  int gdriver=DETECT,gmode,gerror;
  char *string="CCOC 汉字显示控制器
  struct time test1,test2;
  float test;
//进入图形状态
if (registerfarbgidriver(CGA_driver_far)<0 ||
   registerfarbgidriver(EGAVGA_driver_far)<0)
                  return(1);
initgraph(&gdriver,&gmode,"\\tc\\bgi");
if ((gerror=graphresult())<0) {
  printf("%s\n",grapherrormsg(gerror));
  return(1);
  }
//显示字串
gettime(&test1);
for (j=0;j<10;j++)
```

```
  for (i=0;i<4;i++) {
      setcolor(color++);
      cchar.couttextxy(45,i*20+70,string);
      if (color>15) color=1;
      }
gettime(&test2);
cchar.savelib();//字库存盘
closegraph();
printf("Began at %d:%02d and ended at %d:%02d\n",
        test1.ti_sec,test1.ti_hund,test2.ti_sec,test2.ti_hund);
getch();
}
```

（沈　激）

## 1.3　2.13E 汉字系统使用技巧

CCBIOS 2.13E 汉字系统的功能比较强，该系统有一 CXMB.EXE 程序，它是用来查询、修改拼音码首尾码表的。笔者在实际工作中发现，该程序还可用作字典来查询汉字的读音，尤其是比较生僻的汉字。

如果你的2.13E 系统的硬盘\213子目录中没有 CC11.EXE 或 CC16.EXE 或 CC21.EXE 或 CC25.EXE 四个文件之一时，若在 C>下运行 CXMB 程序，并选择“1”，则会出现类似于“C:\213\CC11.EXE 文件没找到”的错误信息。此时，可用如下步骤修正上述错误。

第一步：先找一张空软盘（已格式化的），并插入 A 驱动器；

第二步：执行命令 C>COPY C:\213\CCCC.EXE A:(ENTER)；

第三步：执行命令 C>REN A:CCCC.EXE A:CC11.EXE(ENTER)；

第四步：执行命令 C>COPY A:CC11.EXE C:\213(ENTER)之后，再次执行 C>CXMB(ENTER)，并且选择“1”后，屏幕上会出现下述提示信息：“拼音查询方式（单个——1；连续——回车；查询首尾码——－；结束——E）”。这时，可以选择1（单个），又出现提示信息“汉字：”，此时，可以用区位码方式或五笔字型方式输入你要查询的不知其读音的汉字。例如“荨”字，利用区位码方式输入此字时，已知其区位码为6101；若利用五笔字型方式输入此字时，可以把该字分解为“艹”、“彐”和“寸”三个笔划。无论用上述哪种方式输入该字，均可查得其拼音码为“XUN”。接着，又提示“改成：”，此时，必须敲回车键（即不修改）；之后，又提示“优先：”，此时，敲空格键（即取消优先）。当返回到 CXMB 的二级菜单时，选 E，即可返回到 DOS 状态。

另外，用以上查询生疏汉字读音的前提条件是：你所使用的2.13E 汉字系统未被任何人修改过，并且每个汉字只能查到一个读音。

（石国忠）

## 1.4　多种中文操作系统的共存与拼接

目前国内流行多种中文操作系统，比如：浪潮记忆联想，EASYIN，CCDOS 2.13系列，CCDOS 4.0，金山 DOS 等等，它们各具所长，也各有不足之处。大多数用户都习惯把常用的一些中文操作系统分装在硬盘不同子目录中，使用时总是先启动硬盘进入西文状态，然后根据需要进入某子目录或运行根目录下建立的各种批处理文件来启动相应的汉字操作系统。显然，这样做非常麻烦，而且各

种中文系统的显示、输入及打印等功能又各具特色。浪潮记忆联想系统，其汉字输入具有很方便的联想记忆功能及GRD图形显示功能，但却缺乏五笔字型输入功能，打印驱动模块功能也很弱，不能压缩行距，字体变化极少；EASYIN汉字系统有极强的汉字拼音连输功能，速度几乎可以与五笔字型媲美；CCDOS 2.13系列是目前用户较普遍接受的汉字系统，它有一套自己的拼音联想、五笔字型汉字输入及屏幕特殊显示功能，并具有丰富的汉字打印功能与排版打印功能；金山DOS除了支持其配套的桌面排版系统WPS及图文排版系统SPT外，其最大特点是，它的键盘显示模块能自动识别大多数显示卡而进入中文操作系统，并可以挂接拼音、五笔字型、表形码、电报明码等多种汉字输入模块，而且具有丰富打印排版功能的16、24、40点阵高级打印驱动模块等等。

然而，用户由于习惯或工作需要，常常需要进入某一种操作系统或者不同操作系统的组合。因此，是否可以通过制做一个自动批处理文件，由菜单项进行选择，实现多种中文操作系统的共存与拼接呢？

笔者制作的批处理文件AUTOEXEC.BAT巧妙地利用了DOS命令中的批处理，实现了8种操作系统的连接或拼接。启动时由MENUHH.COM文件显示出功能选择菜单，这时，只需敲一个键字，就可以进入你所希望的操作系统。例如：敲"4"就会进入2.13H汉字操作系统，敲"6"就会进入浪潮记忆联想汉字输入及2.13打印的组合中文操作系统等等。菜单文件可用EDLIN、WORDSTAR等常用编辑软件直接编辑修改2.13H操作系统提供的菜单选择文件MENUHH.COM。值得注意的是切不可破坏任何不能识别的字符。以上均在IBM XT/AT及其兼容机上实现通过。下面仅是几种操作系统连结与组合的实例，通过编辑或修改AUTOEXEC.BAT和MENUHH.COM即能实现其他系统的共存与拼接。

**AUTOEXEC.BAT 程序清单**

```
@ECHO OFF                    关闭命令显示
CLS                          清屏
CD\213                       进入2.13H系统子目录
MENUHH                       启动系统菜单选择画面
IF ERRORLEVEL 56 GOTO H      按回车键转H段
IF ERRORLEVEL 55 GOTO G      选7转G段
IF ERRORLEVEL 54 GOTO F      选6转F段
IF ERRORLEVEL 53 GOTO E      选5转E段
IF ERRORLEVEL 52 GOTO D      选4转D段
IF ERRORLEVEL 51 GOTO C      选3转C段
IF ERRORLEVEL 50 GOTO B      选2转B段
IF ERRORLEVEL 49 GOTO A      选1转A段
:H                           启动金山DOS及WPS桌面排版系统段
CD\WPS                       进入WPS子目录
SPLIB/1                      调金山DOS一级字库至内存
SPDOS                        启动金山DOS汉字输入及显示系统
WBX                          调五笔字形输入模块
WPS                          启动WPS桌面排版系统
GOTO I                       转结束段I
:G                           启动金山DOS及2.13H打印系统段
CD\WPS                       进入WPS子目录
SPLIB/1                      调金山DOS一级字库至内存
SPDOS                        启动金山DOS汉字输入及显示系统
WBX                          调五笔字形输入模块
CD\213                       进入2.13H子目录
PRTA                         调2.13H系统汉字打印驱动模块
```

```
FILE16D                    调16点阵打印字库至内存
FILE24A 4SFHK              调24点阵宋仿黑楷字库至内存
FILE40A 1SFHK              调40点阵宋仿黑楷字库至内存
ZF24 3                     选调24点阵3#字符库模块
GOTO B                     转B段
:F                         启动浪潮记忆联想系统及2.13H打印系统段
CD\LCJYLX                  进入LCJYLX子目录
LCLX16                     启动浪潮记忆联想汉字输入及显示系统
CD\213                     进入2.13H子目录
PRTA                       调2.13H系统汉字打印驱动模块
FILE16D                    调16点阵打印字库至内存
FILE24A 4SFHK              调24点阵宋仿黑楷字库至内存
FILE40A 1SFHK              调40点阵宋仿黑楷字库至内存
ZF24 3                     选调24点阵3#字符库模块
GOTO B                     转B段
:E                         启动EASYIN汉字系统及2.13H打印系统段
CD\EASYIN                  进入EASYIN子目录
LC                         启动EASYIN汉字输入及显示系统
CD\213                     进入2.13H子目录
PRTA                       调2.13H系统汉字打印驱动模块
FILE16D                    调16点阵打印字库至内存
FILE24A 4SFHK              调24点阵宋仿黑楷字库至内存
FILE40A 1SFHK              调40点阵宋仿黑楷字库至内存
ZF24 3                     选调24点阵3#字符库模块
GOTO B                     转B段
:D                         启动2.13H汉字系统及2.13H打印系统段
CD\213                     进入2.13H子目录
GWINT16h 21                调2.13H汉字输入及显示模块
INT10D                     调特显功能扩展模块
YX1                        装入预选字表
PRTA                       调2.13H系统汉字打印驱动模块
FILE16D                    调16点阵显示字库至内存
FILE24A 4SFHK              调24点阵宋仿黑楷字库至内存
FILE40A 1SFHK              调40点阵宋仿黑楷字库至内存
ZF24 3                     选调24点阵3#字符库模块
KEY                        定义功能键ALT-A～W程序
GWB                        五笔字形接口模块
WBZX                       调五笔字形模块
GOTO B                     转B段
:C                         进入西文DOS系统段
GOTO B                     转B段
:B                         仅启动EASYIN汉字系统段
CD\EASYIN                  进入EASYIN子目录
LC                         启动EASYIN汉字输入及显示系统
GOTO B                     转B段
:A                         仅启动浪潮记忆联想系统及2.13H打印系统段
CD\LCJYLX                  进入LCJYLX子目录
LCLX16                     启动浪潮记忆联想汉字输入及显示系统
:B                         命令执行路径搜寻设置段
PATH C:\;D:\DOS;C:\213
APPEND /E
APPEND D:\DOS
CD\
```

| | |
|---|---|
| :I | 结束段 |

**由 MENUHH.COM 显示的结果**

| | |
|---|---|
| 1—LCDOS | 启动浪潮记忆联想汉字输入系统 |
| 2—EASYIN | 启动 EASYIN 汉字输入系统 |
| 3—DOS x.xx | 进入西文 DOS 系统 |
| 4—2.13H | 启动2.13H 汉字输入系统及打印系统 |
| 5—EASYIN+2.13H | 启动 EASYIN 汉字输入系统及2.13H 打印系统 |
| 6—LCDOS+2.13H | 启动浪潮记忆联想汉字输入系统及2.13H 打印系统 |
| 7—SPDOS+2.13H | 启动金山 DOS 汉字输入系统及2.13H 打印系统 |
| CR—SPDOS+WPS | 启动金山 DOS 汉字输入系统及 WPS 桌面排版系统 |
| Qing Xuanze : | 选择提示 |

（吴晨辉　李　焰）

## 1.5 文件磁盘的抢救恢复

微机软件一般都存放在软磁盘和硬磁盘上，一旦磁盘出问题，迅速修复和抢救软件中的文件就显得比什么都重要。现以大量使用的360KB、1.2MB 和1.44MB 软磁盘为例，阐述迅速抢救的技术方法。

360KB 磁盘分为0磁面和1磁面，每面由外向内分为40个同心圆，即40个磁道，每磁道分为9个扇区，所以共有9×40×2=720个扇区，文件就存放在一个个扇区中。1.2MB 和1.44MB 磁盘与此结构相同，仅磁道数和扇区数不同。

整个磁盘结构空间的安排见表1.5.1。

由此表清楚地看到，磁盘的关键信息：BIOS 信息，FDT 表，它们均在0磁道区，若该区稍有问题，整个文件，甚至使整个磁盘的所有文件都无用。再者，只要磁盘插入计算机，读写任何文件都必须先读0磁道的信息，0磁道区使用频率最高，也就较易受伤损坏。

常见的磁盘损坏有：0磁道0扇区(BIOS)损坏；FAT 文件分配表损坏；FDT 文件目录表损坏。下面分别介绍抢救恢复的方法：

**表1.5.1　磁盘结构空间安排**

| 存放信息内容 | 360KB 软磁盘 | | | 1.2MB 软磁盘 | | |
|---|---|---|---|---|---|---|
| | 磁道 | 扇区范围 | 字节长度 | 磁道 | 扇区范围 | 字节长度 |
| BIOS 信息区 | 0面0道 | 00～00 | 512 | 0面0道 | 00～00 | 512 |
| FAT 文件分配表区 | 0面0道 | 01～02 | 1024 | 0面0道 | 01～07 | 3584 |
| FAT 文件分配表备份 | 0面0道 | 03～04 | 1024 | 0面0道 | 08～14 | 3584 |
| FDT 文件目录区 | 0面0道<br>1面0道 | 05～08<br>09～11 | 2048<br>1536 | 1面0道 | 15～28 | 7168 |
| 文件实体存放区 | ⋮<br>1面40道 | 12～719 | 362,496 | ⋮<br>1面80道 | 29～2399 | 1,213,952 |

### 1.0磁道0扇区损坏

0扇区存放着大量的磁盘信息，从绝对地址0000(即磁盘起头处)开始，存放的信息依次是(括号

内为地址)：JMP 转移指令(00～02)；DOS 格式化时的版本(03～0A)；每扇区字节数(0B～0C)；每簇扇区数(0D)；保留扇区数(0E～0F)；FAT 表的数目(10)；最大根目录存储容量(11～12)；磁盘总其扇区数(13～14)；磁盘介质说明标志(15)；FAT 表长度(16～17)；每磁道扇区数目(18～19)；磁头数(0A～0B)以及其他许多信息。

当磁盘驱动器读写该盘文件时，必先将这些 BIOS 信息读入，进行识别，若这些信息损坏，将会造成计算机不认此盘，此时显示：

```
C:\>a:
    General failure reading drive A
    Abort,Retry,Fail?
```

严重时 pctools 无法进入该盘，甚至连 FORMAT 格式化该盘也不行，任何计算机命令对此盘都无效，好像已彻底损坏，或者如有的文章介绍，将该盘从纸套中取出，翻个身，重新格式化。

实际上这种磁盘是完全可以抢救恢复的。解决的办法是：用一盘好的同类磁盘，将0磁道0扇区的 BIOS 信息覆盖到坏盘的0扇区即能完全恢复，而其他文件信息不受任何破坏。下面介绍两种具体操作方法。

(1)pctools 方法：先从 A 盘或 C 盘进入 pctools，取一张好的同类磁盘插入 A 驱中(以下划线的为需敲入的键)

```
ctrl+A                  ;读入 A 盘
F10  D  E               ;选择 Edit 功能
```

取出 A 磁盘，再将坏磁盘插入 A 驱中

```
F5  S                   ;把好盘的 BIOS 信息覆盖到坏盘的0扇区
ESC                     ;退出
```

(2)debug 法：若没有 pctools 软件时取一张好的同类磁盘插入 A 驱中，坏盘插入 B 驱内，键入：

```
A:\>DEBUG
-L 100 0 0 1
-W 100 1 0 1
-Q
```

这样0扇区即能修复，磁盘文件亦抢救好。抢救 BIOS 信息的方法很简便，但很实用。

### 2. FAT 文件分配表损坏

FAT 表的作用：文件在磁盘中是以“簇”为基本分配单元，360KB 软盘为1簇2扇区。文件在分配表中记有该文件的起始簇号，在本 FAT 表中的起始簇号登记项中记有与起始簇号相连的下一簇号，在相连的下一簇号中又记有再下相连的簇号，形成文件存放的簇号“链”，表明了文件的实际存放区。很明显，簇号链错了，有时虽能用 dir 列出文件名，但实际文件却全错了。

FAT 表的位置：由表1.5.1可知，它们都由01扇区开始，360KB 软盘占 01～02两个扇区。

FAT 表的起头为磁盘介质标志，表示本磁盘的介质性质。360KB 软盘为 FDFFFF 起头，1.2MB 软盘为 F9FFFF 起头，硬盘为 F8FFFF 起头。若介质标志损坏，也将出错，甚至会读不出任何文件。

根据以上原理，介绍修复方法如下：

(1)用 pctools 观察 FAT 表起首的磁盘标志是否正确。一般说，FAT 表损坏，pctools 软件还是

能进入该盘读写的。若发现磁盘标志丢失或变化，可将之复原。

具体操作：进入 pctools，将坏盘插入 A 驱中。

```
ctrl+A                          ;读 A 盘
F10  D  E                       ;进入 Edit 功能
F D F F F F                     ;修改360KB 软盘标志
FS  S                           ;写入，存盘
Esc                             ;退出
```

若没有 pctools 软件，亦可用 DEBUG。

```
C:\>DEBUG↙
-L 0100 0   1   1               ;读 A 盘
-E  100 FD FF FF                ;修改磁盘标志
-W  100 0   1   1               ;存盘，写入
-Q                              ;退出
```

(2)如果发现整个 FAT 表都出了问题怎么办?由表1.5.1可知，FAT 表还有一个备份，平常是不用它的，它存放在03～04扇区，可用此备份表覆盖修改第一个已损坏的 FAT 表。

具体操作如下：

```
C:\>DEBUG↙                      ;进入 DEBUG
-L ␣100␣0␣3␣2↙                 ;将3～4扇区值读出
-W␣100␣0␣1␣2↙                  ;写入1～2扇区
-Q↙                             ;退出
```

这样处理后，FAT 文件分配表就修复得完好如初了。

### 3. FDT 文件目录表损坏

由表1.5.1可知，不同磁盘种类，FDT 表长度不一，360KB 磁盘为7个扇区，1.2MB 磁盘为14个扇区，1.44MB 磁盘也为14扇区。但它们的格式都一样，如同一本书前的目录页一样，每一文件为一目录项，在 FDT 表中占有32字节的登记项，见表1.5.2的当中部分，每项分两行显示出，每两个16进制数为一字节。字节位置和目录项内容对应情况见表1.5.2。

表1.5.2 FAT 表内容

| 字节位置 | 0～7 | 8～10 | 11 | 12～21 | 22～23 | 24～25 | 26～27 | 28～31 |
|---|---|---|---|---|---|---|---|---|
| 内容 | 文件名 | 扩展名 | 属性 | 保留 | 文件写入时间<br>时·分·秒 | 文件写入日期<br>年·月·日 | 文件起始<br>簇号 | 文件长度 |

(1)文件数目变少。这类故障的现象是磁盘能读能写，但用 dir 或 pctools 列目录会发现文件数目变少了，这是什么原因呢?

磁盘对文件名有一定要求，即为英文大小写52个字母，0～9个数字，以及如!%({等为数不多的一些字符。如果损坏而使文件名变为不合法的 ASCII 码字符，或出现16进制的“00”等，就会使本文件名的文件“丢失”，甚至会造成本文件以下的所有文件全部“丢失”。从而使目录表中的文件数大大减少。

例如有一张 FDT 表损坏的磁盘，用 pctools 观察 FDT 表如下：

```
+                                    Disk View
+                  Absolute sector 0000005, System ROOT, Disk Abs Sec 0000005

0000(0000)  39 36 2D 30 31 20 20 20 46 49 4C 20 00 00 00 00 96-01      FIL
0016(0010)  00 00 00 00 00 00 B9 05 21 00 02 00 83 01 00 00            !
0032(0020)  00 87 00 00 00 20 20 20 46 49 4C 20 00 00 00 00            FIL
0048(0030)  00 00 00 00 00 00 CC 05 21 00 00 00 00 00 00 00            !
0064(0040)  39 37 2D 31 30 20 20 20 46 49 4C 20 00 00 00 00 97-10      FIL
0080(0050)  00 00 00 00 00 00 36 09 21 00 04 00 76 01 00 00            6 !     V
0096(0060)  39 37 2D 31 31 20 20 20 46 49 4C 20 00 00 00 00 97-11      FIL
0112(0070)  00 00 00 00 00 00 56 09 21 00 05 00 35 01 00 00            V !     5
0128(0080)  39 37 2D 31 38 20 20 20 46 49 4C 20 00 00 00 00 97-18      FIL
0144(0090)  00 00 00 00 00 00 99 09 21 00 06 00 35 01 00 00            !       5
0160(00A0)  4D 2D 31 37 32 34 20 20 45 58 45 20 00 00 00 00 M-1724     EXE
0176(00B0)  00 00 00 00 00 00 83 1B B0 16 09 00 8B 03 00 00
```

FDT 表左边括号前和括号内数字为地址，当中16列是目录项的16进制值，右边为目录项的相对应的 ASCII 码值。损坏在第三行地址为0032中文件名16进制值为0087000000202020(文件名为8个字符)，其中出现了"00"非法字符，这样造成该文件以下的文件如97－10.FIL，M－1724.EXE 等所有以下文件全都"丢失"，无法显示与读写。用 dir 或 pctools 列目录，该盘只有1个文件，其实该盘下面有几十个文件。

解决的方法：给"消失"或"卡"住的文件，虚拟一个文件名。具体操作：

先进入 pctools

| | |
|---|---|
| ctrl＋A | ；进入坏盘 |
| F10 D E F6 R | ；进入 FAT 区 |
| F7 | ；进入修改功能 |
| F8 | ；进入右边 ASCII 码区 |
| 移动光标到错误文件名处 | |
| A1 | ；虚拟一个文件名 |
| F5 S | ；存盘 |
| Esc | ；退出 |

重新读 A 盘后 FDT 表变为：

```
+                  Absolute sector 0000005, System ROOT, Disk Abs Sec 0000005

0000(0000)  39 36 2D 30 31 20 20 20 46 49 4C 20 00 00 00 00 96-01      FIL
0016(0010)  00 00 00 00 00 00 B9 05 21 00 02 00 83 01 00 00            !
0032(0020)  41 31 20 20 20 20 20 20 46 49 4C 20 00 00 00 00 A1         FIL
0048(0030)  00 00 00 00 00 00 CC 05 21 00 00 00 00 00 00 00            !
0064(0040)  39 37 2D 31 30 20 20 20 46 49 4C 20 00 00 00 00 97-10      FIL
0080(0050)  00 00 00 00 00 00 36 09 21 00 04 00 76 01 00 00            6 !     V
```

第三行文件名已虚拟为 A1.FIL，用 dir 列文件目录可见新增并恢复了的 A1.FIL 文件，以及增加了一大批文件。当然，若还有文件名错，可如法处理，直至所有被"卡"的文件全都恢复。

(2)FDT 表中其他信息的损坏处理。由于大部分文件都是按顺序存放的，所以一般来说是可以恢复的。由表1.5.2可知，最重要的是26～27字节的文件存放的起始簇号和28～31字节的文件长度。

失去了文件存放起始簇号，就丢失了文件存放区，必须先找出文件起始簇号，对于按顺序存放的文件，故障文件起始簇号＝上一文件起始簇号＋上一文件长度÷每簇字节数，并取其不小于该值的整数。这里每簇字节数对360KB磁盘为1024字节，1.2MB和1.44MB磁盘为512字节。

例如，前面显示的FDT表中，第一个文件为96－01.FIL，文件起始簇号为0002(26～27字节，低位在前，高位在后)，文件长度为16进制值00000183，则183化为十进制为387，除以1024为0.37，取不少于该值的整数，即为1，故故障文件A1.FIL的起始簇号应为0002＋1＝0003，填入26、27FAT表字节中(注意低位在前，高位在后)即可。

其次，恢复文件长度值时，若FAT表文件分配链已恢复好，方可进行该项工作。具体方法：用DOS中的磁盘插换命令加上下参数，这样该命令则具有了自动调整文件长度，并将其值自动填入FDT表中的功能。

具体操作(如上例A1.FIL文件长度)：

A:\> CHKDSK /F ↙

显示：A:\> A1.FIL

Allocation error,size adjusted

将自动计算该文件长度，自动填入FDT表中28～31字节。

最后，由于修复了的文件可读可写可执行，根据执行情况等可知其准确的文件名，再将它代替虚拟的文件名即可。

FDT表中其他一些信息如写盘时间、日期，并不影响文件的运行，无足轻重。若要修改也可用pctools直接写入时间、日期即可。

### 4. 其他文件实体的修复

(1)带有系统文件的软磁盘，特别是一些带商品性质的应用软件，带有本盘的启动系统文件，能独立工作。若插入A驱中，系统不能启动，大多是系统文件出问题。系统文件主要是三个：IBMBIO.COM、IBMDOS.COM和COMMAND.COM，可用一张DOS版本相同的盘，将系统中的三个文件重新装入即可修复。

(2)磁盘插入计算机即显示出错，无法读写，也就无法修复，此时，可用“冒名顶替”法。即将一盘好磁盘先插入A驱中，待机器读好此盘后，拔出，将坏盘插入，即可进行修复软件工作。

(3)pctools中有一Verify Disk命令，能迅速查出磁盘文件具体部位的损坏，再进入该位置校复，或者用相同的文件覆盖之，即可修复。

总之，用上述方法，磁盘文件都能迅速修复。由于硬盘和软盘结构基本相似，故本文所述方法对修复硬盘也完全适用。

以上所述是用pctools 5.5版本、6.0版本，在DOS 3.2以上版本，PC-XT机和286机上通过。

(严居济)

## 1.6 快速显示彩色图形汉字

笔者阅读了几篇有关在西文状态下显示图形汉字的文章，发现其都是采用描点法显示汉字的。描点即写图形象素，最简便的方法是调用高级语言的写图形象素语句或函数，也可以利用BIOS的写图形象素功能，还可以利用直接写显示存储器的方法。

在16点阵汉字库中，每个汉字字模占32字节，其点阵信息按行顺序存放，每行2个字节，共16行，

构成16×16的方阵。采用描点法显示彩色汉字的程序一般为两重循环结构，外层循环控制汉字字模的行，每循环一次处理1行字模；内层循环控制字模的列，每循环一次处理字模的1位。若字模的某一位为1，则用前景色写图形象素，否则用背景色写。程序的时间复杂度为 n(n=16)。因此，这种方法显示汉字的速度慢，效率低，效果差。为此，笔者分析了目前广泛应用的 EGA/VGA 显示适配器的组成和工作原理，根据16点阵汉字字模的结构，摸索出一种按照汉字字模的字节快速显示16点阵彩色汉字的方法。

EGA/VGA 是面向页面的显示设备，它含有256K 字节的显示存储器，显示存储器被分成4个均从 A000:0000H 开始的页面，分别是亮度、红、绿和蓝色页面(3、2、1、0)，每个页面控制一种彩色成分，在屏幕上最多可以同时显示16种颜色。对于显示模式10H 和12H，显示分辨率分别是640×350和640×480。显示存储器的1个字节对应屏幕上的8个图形象素，汉字显示的位置(x,y)在显示存储器中的字节偏移地址为：80×y+x/8。在 EGA/VGA 内部有一个32位的数据锁存器，对显示存储器的一次读操作，将页面上同一地址的4个字节装入锁存器，对显示存储器的一次写操作，可以将处理器或锁存器的数据写入4个页面，当图形控制器的位屏蔽寄存器某一位为1时，该位数据由处理器写入页面，否则由锁存器写入。通过对 EGA/VGA 寄存器的操作可以方便地实现按字节快速显示汉字。下面是显示16点阵彩色汉字的 C 函数 disp_16hz( )：

```
/* 按汉字字模的字节直接显示汉字函数 */
disp_16hz(unsigned char *hzmp, int x, int y, int color, int
bkcolor)
{
  char far *p;
  int i,j,k;
  p=(char far *)(0xa0000000+80*y+x/8);
  for (i=0;i<16;i++)
    for (j=0;j<2;j++) {
      outport(0x3ce,0x0205);
      k = *(p+80*i+j);
      outportb(0x3ce, 0x08);
      outportb(0x3cf,hzmp[2*i+j]);
      *(p+80*i+j) = color;
      k = *(p+80*i+j);
      outportb(0x3ce, 0x08);
      outportb(0x3cf, hzmp[2*i+j]);
      *(p+80*i+j) = bkcolor;
    }
  outport(0x3ce, 0x0005);
  outport(0x3ce, 0xff08);
}
```

该函数采用写模式2方式，使用了图形控制器(I/O 端口3CEH/3CFH)的模式寄存器(索引号5)和位屏蔽寄存器(索引号8)。其工作原理是：将处理器数据的低4位写入对应的四个页面，第0位写入页面0，第3位写入页面3，亦即处理器数据用作象素的颜色。对应于位屏蔽寄存器的值，一次可写1到8个象素。显示彩色汉字时是通过两次写入法实现的。首先将存储器数据读入锁存器，通过位屏蔽寄存器用字模前景数据屏蔽背景数据，并将前景色写入页面，即完成字模前景数据的显示。然后，再读显示存储器，将前景数据求反得背景数据，并用位屏蔽寄存器屏蔽前景数据，最后将背景色写入页面，即完成字模背景数据的显示。显然，该程序的时间复杂度为 m×n(其中 m=16,n=2)。与描点法相比，按字节显示汉字的方法速度快，效率高，效果好，经测试，显示速度比描点法快8倍。

最后给出一个调用 disp_16hz( )函数的程序实例，该程序在屏幕上显示三个不同色彩的汉字串，并且适用于中西文环境。

```
#include <io.h>
#include <fcntl.h>
#include <dos.h>
main()
{
  union REGS r;
  int col, row, f_handl, current_mode, id_mode=0;
  unsigned qh,wh;
  unsigned long offset;
  unsigned char *hz_p,hzm[32];
  unsigned char cs[]="春眠不觉晓　处处闻啼鸟　夜来风雨
声　花落知多少";
  /*　以二进制只读方式打开16点阵汉字库文件　*/
  f_handl=open("c:\\213\\hzk16",0_RDONLY|0_BINA-
RY);
  if (f_handl==-1){
```