# 西文方式下挂接 UCDOS 5.0 中的万能汉字输入法

湖北省浠水师范学校（436200） 江龙

本刊 9 7 年第 8 期刊发了赵亦朋先生的《面向对象汉字输入法的实现》一文。此文以常见的拼音输入法为蓝本，向读者介绍了怎样在西文方式下挂接输入法。其实，自 UCDOS 5.0 推出后，任何一种汉字输入法，只要经 LMDMNG.exe 程序制成 IMD 文件，便可在其操作环境下，通过 LIMD.exe 程序挂接起来直接使用。这的确让人心中有种说不出的激动，因此，若我们了解了 UCDOS 5.0 输入法驱动程序 IMD 文件格式，便可随心所欲挂接我们喜爱的“输入法”，而不必象赵先生那样对单一输入法文件逐个分析、再编程。

以下笔者便向您介绍“万能输入法”的挂接方法。

## 一 输入方法驱动（Input Method Drive）文件结构分析

UCDOS 5.0 中的 IMD 文件（Input Method Drive）可分为“文件头部信息”、“汉字编码信息”、“索引表信息”、“编码对应的汉字内容”四个部分。

### 1.1 文件头部信息

文件头部信息共占用 0x80 个字节，其结构如下:

```
typedef struct IMD_H
  {  BYTE ImdId[0x10];     /*IMD 文件标志*/
     BYTE space1[0x10];    /*保留，均设置为 0*/
     BYTE CodingName[9];   /*编码名称*/
     BYTE HotKey;          /*功能键号*/
     BYTE CodeTable[0x41]; /*码元表*/
     BYTE CodeTableSize;    //实际编码大小
     BYTE OmniptentKey;     /*万能码长*/
     BYTE MaxCodeLenght;    //最大码长
     BYTE AutoChoiceInput;   /*是否自动输入*/
     BYTE RunSearch;         /*是否执行模糊搜索*/
     BYTE UseDefineWord;      /*是否使用自定义词组*/
     BYTE DefineWordsMode;   //自定义词组方案
     BYTE space2;             /*保留*/
     BYTE PreWordCodeLength; /*单字编码最大长度*/
     BYTE RecodeId;           /*重码标志*/
     WORD SearchWordNumber;  /*单字编码个数*/
     WORD InderCodeSize;     /*单字编码大小*/
```

```
    WORD CwordNumber;       /*汉字个数*/
    WORD InderTableOffset;  /*查询索引表首地址*/
    WORD InderTableNumber;  /*索引表数目*/
    BYTE space3;            /*保留*/
} ;
```

IMD 文件标志为"UCDOS IMD FILE"加文件结束符 0x1a，我们可以以此来认别其为 IMD 文件。

编码名称存放此文件编码的名称，最多用四个汉字说明。如“【五笔】”、“【全拼】”、“【简拼】”等等。

功能键号表示该编码激活的热键，一般将其设置成 2－10 之间的一个数。如：2 表示 Alt+F2（全拼），5 表示 Alt+F5（五笔）。

码元表：存放该编码所用字符列表。如五笔字型中，用到了 abcde...y 这 25 个码元它的长度由实际码元个数决定.

万能键，也称通配符，如五笔字型输入法中′z′即是，用户在输入编码时，若一时不知对应的按键，则可用此字符代替。

1.2 汉字编码信息和索引表信息

汉字编码信息存放的是汉字对应的编码。个数由文件头中的“单字编码个数”决定，一共占用“单字编码大小”个字节。我们可以用此来反查汉字对应的编码。因篇幅有限，本文对此部分不作详细说明。

“编码对应的汉字地址表”中，每个地址占用 3 个字节， 一共占用“编码序号数目”（文件头部信息中的“索引表数目”）＊3 个字节空间。在文件中的存放形式符合 long 数据的存法（将其最高字节设置为 0）。如 45 23 00(十六进制）表示数据为 0x002345。编码对应的汉字信息大小实际上即等于两个编码对应的汉字偏移地址内容之差。在这里值得注意的是：序号为 0 的编码对应的汉字偏移地址内容需要我们计算出来，它等于 0x80+ “编码对应的汉字偏移地址”的偏移地址＋编码索引表数目＊3 。

“编码序号数目”由“码元表的数目”决定，为“实际码元个数”的平方。如五笔字型的“实际码元个数”为 25 个，则“编码序号数目”为 25*25=625 个，其中编码（设其为 A1A2...）对应的“编码序号”为“A1 在码元表中的序号*实际码元表大小＋A2 在码元表中的序号”。如：五笔编码 ab...，对应的“编码序号”为 0*25+1=1。

1.3 编码对应的汉字信息

该信息由若干个“编码＋编码对应的汉字或词组”组成。若有重码或结束，则将该编码对应汉字或词组的最后一个汉字的位码高位置成 0 。若编码对应的某一汉字位码（高位置 0 后），后面一字符的高位依然为 1，则表示有重码，否则表示结束。

如:72 73 68 69 B6 D9 CA 31 B6-D8 CA 35 64 75 6F B6 60 70 ....则表示 r(72)s(73)h(68)i(69)对应的汉字或词组为(此处用国标值表示） b6d9 cad1(词组），因为 31 后的 b6 高位为 1， 故它有重码，其后一个为 b6d8 cab5 因 35 后为 64，再无重码。对此处理可以参考 GetWord()函数。

## 二 编码对应汉字查找方法

查询某编码对应的汉字，其算法步骤为:

1、算出编码序号。

编码序号 = 第一个码元在码元表中的序号*码元表长度 + 第二个码元在码元表中的序号

2、根据编码序号，算出它在索引表中的偏移地址。

索引地址 = 索引表首地址 + 编码序号*3

3、求出查找汉字信息块的长度

长度 = 下一个编码序号的索引地址内容 - 该编码序号索引地址内容

4、将“汉字信息块”读入内存，在汉字信息块中查找与编码一致的内容，其后的汉字信息即是对应的汉字或词组。

具体查找过程，请参阅 GetWord()和 GetInputWord()两个函数。

## 三、程序实现

想在西文方式下挂接 UCDOS 5.0 中的万能输入方法，同时可通过不同的按键选择不同的输入法，则我们要先将这些编码文件打开，然后根据需要选择一个文件进行读取即可。用户可以通过 LIMD()函数装入 IMD 文件，通过 SetIMD()设置当前的输入方法。

为了提高查寻速度问题，本程序先将“编码对应汉字地址”索引表读入内存（因它们经常用到），同时采用了直接写屏显示汉字方法。另外，为方便用户，此程序界面仿 UCDOS 设计而成， 一些按键均等同崐 UCDOS 操作，Ctrl+F9 用于设置全／半角，Alt+Fn 用于选择某一输入方法，有重码时，可通过“+”，“-”键翻页，数码键选择。

本程序在配有 VGA 显示卡的 X86 系列微机中，经 Borland C++ 3.1 编译通过，运行良好。

```
#include <bios.h>
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#define ShowWordSize 38       //重码每页字符数目
#define ShowCodeSize 8        //最长编码个数
#define WORD unsigned int
#define BYTE unsigned char
#define BottomY               29 /*提示信息显示行数*/
#define AscWordX               1   /*全半角显示列数*/
```

```
#define IMNameX   6   /*输入方法显示列数*/
#define CodingX              15    /*编码显示列数*/
#define MsgX                 23    /*重码汉字提示显示列数*/
#define ShowCodingColor   RED   //显示编码的前景色
#define ShowIMDNameColor BLUE //显示输入方法名称的前景色
#define ShowMsgColor       BLACK      //显示汉字提示的前景色
#define ShowAscWordColor BLACK   //显示全半角前景色
#define MsgBkColor           WHITE   //提示行前景色
#define HZK             "C:\\ucdos\\hzk16"            //汉字库文件
#define ASCFILE       "C:\\ucdos\\asc16"            //ASC 文件
#define WB              "c:\\ucdos\\drv\\wb.imd"   //五笔字型
#define PY              "C:\\ucdos\\drv\\py.imd"   //全拼
#define SP              "C:\\ucdos\\drv\\sp.imd"   //双拼
#define JP              "c:\\ucdos\\drv\\jp.imd"   //简拼
#define MaxNumber     5      //挂接汉字输入法的最多个数
#define RET(n)          { ErrorCode=n;return(ErrorCode);}
#define BEEP    printf("\7")     //响铃
enum KeyVal
    { Alt_F1      = 0x6800,       Alt_F10     = 0x7100,
      Ctrl_F9     = 0x6600,       ESC           = 0x011b,
      BACKSPACE = 0x0e08,       ENTER       = 0x1c0d};
enum IMD_ErrorCode     //类的错误代码
     {   OK,                          //无错误
         HzkFileNotOpen,          //汉字库不能打开
         AscFileNotOpen,          //asc 字库不能打开
         FileNotOpen,             //编码文件不能打开
         MallocError,             //内存分配错误
         NotImdFile,              //不是 IMD 文件
         IMTooMany,     //编码文件太多
         Exist,                   //此代码键已经用上
         NoThisIM       //没有这种输入法
       };
typedef struct IMD_H; //参阅文章“头部信息”说明部分
class IMD
{private:
   FILE *LibFp,*AscFp;       //汉字库,ASCII 点阵信息文件
   FILE *ImdFp[MaxNumber]; /*编码文件指针*/
   struct IMD_H *ImdHeadMsg[MaxNumber]; /*IMD 文件头部信息*/
   BYTE     *InderTable[MaxNumber]; /*IMD 查询索引表指针*/
```

```
   BYTE IM;     /*当前使用的输入方法*/
   BYTE ErrorCode;       /*错误代码*/
   BYTE WordId;   /*全半角标志*/
public:
   BYTE GetColor(BYTE    data, BYTE    forecolor,
                    BYTE backcolor,    BYTE i);
   void ShowString(int x,int y,BYTE *s,int color,int bkclor);
   void Bar(int x,int y,int w,int h,int bkcolor);//清一矩形
   WORD EditString(int x,int y,int color ,int bkcolor,
                   WORD *RetKey,BYTE *buf);
   int    LIMD(BYTE *file,int i);//装入汉字输入法 IMD 文件
   WORD GetInputWord(BYTE *buf);//取用户输入的汉字或 ASCII 字符
   void GetWord(BYTE *,BYTE *);
   int    IsInTable(BYTE c);
   void ShowCodeMsg(BYTE *s); //在提示行显示汉字编码字符
   void SetIMD(int i); //设置当前所用的汉字输入方法
   void SetWordMode() ;   //    设置全半角(热键为 Ctrl+F9)
   void ShowCwordMsg(BYTE *s); //在提示显示输入的汉字
   void SetCursorXY(int x,int y); /*设置当前光标显示的位置*/
   void ClearShowCword(); //清显示输入重码汉字提示区
   void ClearShowCode(); //清显示编码提示区
   WORD Init_IMD(void); //初始化函数
    ~IMD();//析构函数
   BYTE GetErrorCode(){return(ErrorCode);}/*取错误代码*/
};
void IMD::ShowString(int x,int y,BYTE   *  s,
                int color,int bkcolor)
{ BYTE buf[32],bitcolor;int i,j,k,num; long offset;
   while(*s)
    { if((*s)&0x80)
       {offset=((*s-0xa1)*94L+*(s+1)-0xa1)*32L;
         fseek(LibFp,offset,SEEK_SET); //取字模
         fread(buf,32,1,LibFp);num=2;s+=2;}
       else{ fseek(AscFp,(*s)*16,SEEK_SET); //取 ASCII 字符点阵
          fread(buf,16,1,AscFp);s++; num=1;}
       for(k=0;k<num;k++) //显示汉字可 ASC 字符
           for(i=0;i<16;i++)
       for(j=0;j<4;j++)
         {bitcolor=GetColor(buf[num*i+k],color,bkcolor,j);
```

```
            outportb(0x3c4,2);outportb(0x3c5,1<<j);
            pokeb(0xa000,80*16*y+80*i+x+k,bitcolor);}
        x+=num;}}
BYTE IMD::GetColor( BYTE data,BYTE    forecolor,
                    BYTE backcolor,BYTE i)
   {int color;
    if(forecolor>>i&1)
              if (backcolor>>i&1) color=0xff;
              else                color=data;
    else
              if (backcolor>>i&1)    color=~data;
              else                color=0;
   return(color);}
void    IMD::ShowCodeMsg(BYTE *s)
//在提示行显示汉字编码字符
{ ClearShowCode(); //清编码显示区域
 ShowString(CodingX,BottomY,s,ShowCodingColor,MsgBkColor);}
void IMD::SetIMD(int n)
 //在提示行显示当前所用的汉字输入方法
{BYTE *msg="【英文】";
 if(n>=1 && n<=MaxNumber)
   { if(ImdFp[n-1]!=NULL) //所选择的输入方法已挂接
      {msg=ImdHeadMsg[n-1]->CodingName;IM=n;}}
 else //没有挂接,则设置为英文输入方法
{IM=0;ShowCwordMsg("西文方式下挂接 UCDOS 5.0 万能汉字输入法");}
 ClearShowCode();
 ShowString(IMNameX,BottomY,msg,ShowIMDNameColor,MsgBkColor);
}
void IMD::SetWordMode() //    设置全半角(热键为 Ctrl+F9)
{ BYTE *msg[2]={"半角","全角"};WordId=(WordId+1)%2;
   ShowString(AscWordX,BottomY,msg[WordId],  ShowAscWordColor 崐,MsgBkColor);
}
void IMD::ShowCwordMsg(BYTE *s) //在提示显示输入的汉字
{ ClearShowCword();//清汉字重码提示区域
   ShowString(MsgX,BottomY,s,ShowMsgColor,MsgBkColor);
}
WORD IMD::EditString(int x,int y,int color,int bkcolor,
      WORD *ReturnKey,BYTE *text)
{ WORD key;int i,num;BYTE s[80]=""; *text=0;
```

```
   while(1)
{SetCursorXY(x,y); //显示光标
 do {key=GetInputWord(s); //取当前输入的字串
    }while(key==0 && *s==0);
 SetCursorXY(x,y); //隐含光标
 if(key==0) //字串有效
 {strcpy(&text[strlen(text)],s); //将输入的字串送入缓冲区中
  ShowString(x,y,s,color,bkcolor);x+=strlen(s);}
  else  // 为不可显示按键
  {if(key==BACKSPACE)//前删除键
 {if(*text) //若当前缓冲区中有输入的数据
    {if(text[strlen(text)-1]>0xa0)num=2;//前为汉字删除两个字符
     else   num=1;
    text[strlen(text)-num]=0;x-=num;
    Bar(x,y,num,1,bkcolor);} //清删除的字符
  else BEEP;}
else{ for(i=0;ReturnKey[i];i++)
       if(key==ReturnKey[i])return(key);
      BEEP;} //否则响铃
 }}} //function end
WORD IMD::Init_IMD(void)
 { int i;WordId=1;IM=0;   //英文
for(i=0;i<MaxNumber;i++)
{ImdFp[i]=NULL;ImdHeadMsg[i]=NULL;InderTable[i]=NULL;}
LibFp=fopen(HZK,"rb"); //打开汉字库
if(LibFp==NULL)   RET(HzkFileNotOpen);
AscFp=fopen(ASCFILE,"rb"); //打开 ASC 字库
if (AscFp==NULL) RET(AscFileNotOpen);
Bar(0,BottomY,80,1,MsgBkColor);
SetWordMode();/*半角*/ SetIMD(0);//英文方式
RET(OK); }
void IMD::SetCursorXY(int x,int y) //设置光标位置
{int i; //以下以反像方式显示一下画线光标
for(i=0;i<4;i++)
{outportb(0x3ce,4); outportb(0x3cf,i);
 outportb(0x3c4,2); outportb(0x3c5,1<<i);
pokeb(0xa000,y*16*80+17*80+x,~peekb(0xa000,y*16*80+x+17*80));
}}
void IMD::Bar(int x,int y,int w,int h,int bkcolor)
```

```
//从左上角(x,y)高为 h,宽为 w,前景色为 bkcolor 清一矩形( 坐标以文崛本方式给出)
{ int i,j;
   bkcolor=bkcolor&0x0f;
   for(j=0;j<h*16;j++)
      for (i=0;i<w;i++)
      {outportb(0x3c4,2); outportb(0x3c5,bkcolor);
       pokeb(0xa000,y*16*80+j*80+i+x,0xff);
       outportb(0x3c4,2); outportb(0x3c5,~bkcolor);
       pokeb(0xa000,y*16*80+j*80+i+x,0);
   }
}
IMD::~IMD() //析构函数
{int i;
   for (i=0;i<MaxNumber;i++) //关闭所有打开的文件
if(ImdFp[i]!=NULL)
 {free(ImdHeadMsg[i]);free(InderTable[i]);fclose(ImdFp[i]);}
if (AscFp!=NULL) fclose(AscFp); //关闭打开的汉字库和 ASC 文件
if (LibFp!=NULL) fclose(LibFp);}
int IMD::LIMD(BYTE *file,int i)
 /*   作      用:装入汉字输入法 IMD 文件
      入口参数:file   输入方法文件名
                  i       表示对应的功能键
                          0   英文          1   alt+F1
                2   alt+F2          .....
      说      明:程序运行后,将 ErrorCode 设置事下:
           OK      装入成功   其它   表示错误*/
{ int j;   long size;
   if(i>MaxNumber||i<=0) RET(IMTooMany);
   /*若当前挂接的 IMD 超过规定的最多个数*/
   i--;if(ImdFp[i]) RET(Exist);//代码为 i 的键已经接接了输入法
   if((ImdFp[i]=fopen(file,"rb"))==NULL) RET(FileNotOpen);
   ImdHeadMsg[i]=(IMD_H *) malloc(sizeof(struct IMD_H));
   if(ImdHeadMsg[i]==NULL) /*为 IMD 文件头部信息分配内存*/
    { fclose(ImdFp[i]);RET(MallocError);}
   fread(ImdHeadMsg[i],sizeof(struct IMD_H),1,ImdFp[i]);
   if(strcmp(ImdHeadMsg[i]->ImdId,"UCDOS IMD FILE\x1a"))
    {/*比较是否为 UCDOS IMD FllE*/
   free(ImdHeadMsg[i]);/*释放 IMD 文件头部信息占用内存*/
      fclose(ImdFp[i]);RET(NotImdFile);}
```

```
 size=ImdHeadMsg[i]->InderTableNumber*3; //计算索引表大小
 InderTable[i]=(BYTE *)malloc(size+1);/*为索引内容分配内存*/
 if (InderTable[i]==NULL)
  { free(ImdHeadMsg[i]); fclose(ImdFp[i]);RET(MallocError);}
fseek(ImdFp[i],0x80+ImdHeadMsg[i]->InderTableOffset,SEEK_SET);
fread(&InderTable[i][3],size,1,ImdFp[i]);
size=ImdHeadMsg[i]->InderTableOffset+ImdHeadMsg[i]->InderTableNumber
     *3+0x80;
memmove(InderTable[i],&size,3);RET(OK);}
WORD IMD::GetInputWord(BYTE *buf)
/* 作      用:取当前用户输入的汉字或 ASCII 字符
   入口参数:buf  汉字可 ASCII 字符保存处
   说      明:若当前输入了汉字或 ASCII,则将其内容置于 buf,且返回 0;
     否则将 buf 置成空,返回用户的按键值*/
 { union    {WORD Key;BYTE ch,s;}KEY;
   long ll=0,length=0,offset; int i,n;
   int   size=ImdHeadMsg[IM-1]->PreWordCodeLength;
   BYTE   *CodeBuf=(BYTE *)malloc(size+1);
   BYTE   *WordBuf=0,find,keyn=0;*buf=0;
 while(keyn<size) //读当前按键
  {KEY.Key=bioskey(0);
   if(!(KEY.ch>=' ' && KEY.ch<='~'))//不是可显示字符
   {switch(KEY.Key) {
     case BACKSPACE: //前删除键
          if (keyn==0) {*buf='\0';    free(  CodeBuf)  ;
         return(BACKSPACE);} /*无字符输入,则返回前删除键*/
          else {keyn--;CodeBuf[keyn]='\0';}
               /*有字符输入,则删除一个字符*/
          break;
    case   Ctrl_F9: SetWordMode();continue;
    case ENTER://编码缓冲区无编码,返回 ENTER,否则清编码缓冲区
     if (keyn==0) { *buf='\0';free(CodeBuf);return(ENTER);}
         else {keyn=0;*CodeBuf='\0';}
         break;
    default:   /*其它字符视为无效,给予响铃警告*/
     if(KEY.Key>=Alt_F1 && KEY.Key<=Alt_F10)//为输入方法切换键?
     {if (ImdFp[(KEY.Key-Alt_F1)>>8]!=NULL)
          SetIMD(((KEY.Key-Alt_F1)>>8)+1);
           else SetIMD(0); /*否则,设置成英文方式*/
```

```
            *CodeBuf=0;keyn=0;}
          else { *buf='\0';free(CodeBuf);return(KEY.Key);}
            break;}   }
  else   { if (IM==0 || IsInTable(KEY.ch)==0 && keyn==0)
 /*当前处于英文输入方式或者当前可显示字符不在编码列表中*/
 { if(WordId==1 && KEY.ch!=' ') /*当前处于全角英文方式*/
     { /*将半角 ASCII 字符转换成全角字符,并将其存入 buf 处;*/
    *buf=(KEY.ch-' ')/94+0xa3; *(buf+1)=(KEY.ch-' ')%94+0xa0;
     *(buf+2)='\0';}
     else { *(buf++)=KEY.ch;*buf='\0';}
     free(CodeBuf);return(0);}
    else if(IsInTable(KEY.ch)>0)
        {CodeBuf[keyn]=KEY.ch;keyn++;CodeBuf[keyn]='\0';}
           else if(KEY.ch==' '&&keyn>0) break;
             //按下空格键,表示编码输入结束
          else BEEP;} //无无效的字符,给予响铃警告
 if(IM!=0){ShowCodeMsg(CodeBuf);ShowCwordMsg("");}
}   //While End
CodeBuf[keyn]=0;offset=IsInTable(CodeBuf[1]);
if (offset==0) offset=1; //若第 2 个字符为空格,表示一级简码
offset=(IsInTable(*CodeBuf)-1)*ImdHeadMsg[IM-1]->CodeTableSize
        +offset-1;
memmove(&ll,&InderTable[IM-1][3*offset],3);
memmove(&length,&InderTable[IM-1][3*(offset+1)],3);
length-=ll;WordBuf=(BYTE *) malloc(length+2);
if (WordBuf==NULL) return(0);
WordBuf[0]=0xa1;WordBuf[1]='a';
fseek(ImdFp[IM-1],ll,SEEK_SET); /*将文件指针移到索引内容处*/
fread(WordBuf+2,length,1,ImdFp[IM-1]); /*读取内容*/
WordBuf[length]=0;GetWord(WordBuf,CodeBuf);
if(*WordBuf==0)BEEP;/*对应的编码无汉字 * /
else strcpy(buf,WordBuf);
free(CodeBuf);free(WordBuf);return(0);}
void IMD::ClearShowCword() // 清显示重码提示区域
{   Bar(MsgX,BottomY,ShowWordSize,1,MsgBkColor);}
void IMD::ClearShowCode() //清显示编码区域
 {Bar(CodingX,BottomY,ShowCodeSize,1,MsgBkColor); }
void IMD::GetWord(BYTE *buf,BYTE *s)
 { /*作用:取编码 s 对应的汉字
```

```
    入口参数:buf 汉字索引表内容   s    编码
    出       口: buf 编码 s 对应的汉字可词组
             为空时 表示无*/
{   int x, y,i,page,next,num,l;
    BYTE *ss,*PreS,msg[20][41],ch;PreS=buf;
    while(1)
        { ss=strstr(PreS,s);
      if (ss==NULL){*buf=0;return;} //编码对应无汉字
      if(*(ss+strlen(s))>0x80&&*(ss-2)>0x80)break;
      PreS=ss+strlen(s);}
   while(*ss<0x80) ss++; //直接指向其对应的汉字
   i=0;page=0;next=1;num=0;msg[page][0]=0;
while(*ss && next) //将重码汉字或词组读入 msg 处
   { PreS=ss; while(*ss>0x80) ss++; /*查询该汉字*/
      *ss|=0x80;l=strlen(msg[page]);ch=*(ss+1);
      //其汉字后对应的是 ASCII 表示没有汉字或词组
     if(ch>0x80) next=1;else next=0;
      num++;   *(ss+1)=0;
      if (38-l-(ss+1-PreS)>0) i++; //当前页中,还可容纳该汉字
      else     //容纳不下,则送给下页
          { page++; i=1; l=0; msg[page][0]=0;}
      sprintf(&msg[page][l],"%1d:%s",i,PreS);
      ss++;*ss=ch;}
i=0;ShowCwordMsg(msg[0]); //显示编码对应的汉字
if (num==1)   //无重码字,直接将该汉字或词组信息返回
   { memmove(buf,&msg[0][2],strlen(msg[0])-1); return;}
while(1)
   { ShowCwordMsg(msg[i]); //显示重码汉字,让用户选择
      ch=getch();
      switch(ch)
        {case '=':   if (i==page) BEEP; else i++; break;
         case '-':   if (i<=0) BEEP; else i--;   break;
         default: if (ch>='0' && ch<='9')
                { ss=strchr(msg[i],ch);
            if (ss!=NULL)
                        { ss+=2;
                  while(*ss>0x80) *(buf++)=*(ss++);
                        *buf=0; return;}
                  else BEEP;}
```

```
                else {ss=&msg[i][2];
            while(*ss>0x80) *(buf++)=*(ss++);
            *buf=0; return;}
                break;
    }}}}
int IMD::IsInTable(BYTE key) //key 是否在码元表中
   { BYTE *s;     int l;
     s=strchr(ImdHeadMsg[IM-1]->CodeTable,key);
     if (*s==0 || s==NULL) return(0);
     else     return(s-ImdHeadMsg[IM-1]->CodeTable+1);
   }
void   SetShowMode(int mode) /*设置显示模式*/
{union REGS r;r.h.al=mode;r.h.ah=0;int86(0x10,&r,&r);}
void main()
  { IMD i; int ii;BYTE code[10];   BYTE msg[80];
    WORD RetKey[]={ENTER,ESC,0};//定义返回键
    SetShowMode(0x12);   //设置成 640*480
   if(i.Init_IMD()!=OK) //初始不成功
   {SetShowMode(3);printf("init Error.");exit(0);}
    i.Bar(0,0,80,29,BLUE); //将屏幕清成 BLUE 色
    i.SetIMD(0); //英文方式
    i.LIMD(WB,4); //挂接五笔输入法 热键为 Alt+F4
    i.LIMD(SP,3);//挂接双拼 热键为 Alt+F3
    i.LIMD(PY,2);//挂接全拼 热键为 Alt+F2
    i.LIMD(JP,1);//挂接简拼 热键为 Alt+F1
    //接受用户输入的字串
    i.EditString(2,1,WHITE,BLUE,RetKey,msg);
    SetShowMode(3);
}
```