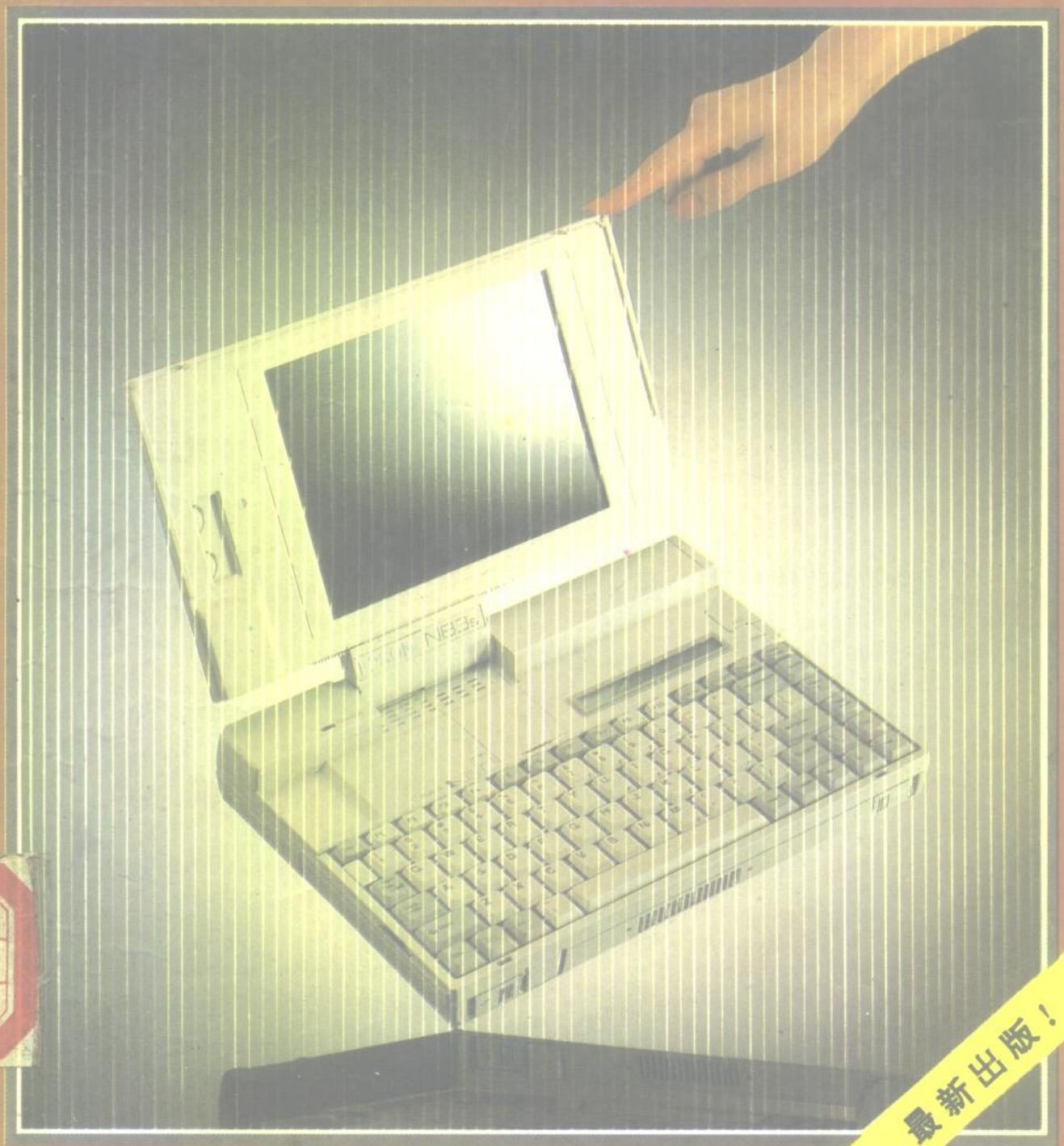
C语言
编程技巧程序集

袁征　杨仁树　严建新　编著



最新出版！

電子工業出版社

C 语言编程技巧程序集

袁征　杨仁树　严建新　编著

电子工业出版社

(京)新登字 055 号

内容提要

本书是一本用C语言开发应用软件的工具书，从七个方面介绍了有关C语言的编程经验和技巧。主要内容有：如何用C语言扩充DOS命令，怎样编制屏幕界面程序，内存驻留技术的实现，怎样在WINDOWS环境下开发C语言程序，C语言与数据库、汇编语言接口以及C语言调试过程中常见错误等。书中介绍的经验和技巧可以直接被软件开发者使用，对软件开发者有启迪和帮助作用。

本书适合于从事计算机软件开发的工程技术人员及大专院校师生阅读。

C语言编程技巧程序集

袁征　杨仁树　严建新编著

责任编辑　张丽华

*

电子工业出版社(北京万寿路)

电子工业出版社发行　各地新华书店经销

北京市怀柔县东晓印刷厂印刷

*

开本：787×1092 毫米 1/16　印张：18.5　字数：460 千字

1993 年 5 月第 1 版　1993 年 5 月第 1 次印刷

印数 10100 册　定价：12.50 元

ISBN7--5053--1950--7/TP・470

前　言

C 语言由于其数据类型丰富、语句精炼灵活、效率高、表达力强、以及可移植性好等诸多优点，倍受程序员喜爱。目前，绝大多数程序员在开发应用软件时，都采用 C 语言来编程。因此，出版有关这方面的图书，提高 C 语言的实际编程能力意义重大。

本书从七个方面介绍 C 语言编程经验和应用技巧。书中大量的实例不仅适合 C 语言初学者学习使用，还适合具体开发应用软件的工程技术人员使用，作为他们开发软件的基础。书中每个子例程都可以作为一个独立的子模块，在开发具体应用软件时直接调用。

书中所有实例程序都收集在一张高密度磁盘内。

在编写本书的过程中，收进了一些近几年来中外计算机刊物上的典型实例，作为讲解范例，对这些工程技术人员表示衷心的感谢。同时，也得到了曹悦工程师、刘景华硕士、孟贵武工程师以及张宏宇女士的大力帮助，他们做了一定的工作，提出很多宝贵的意见，对此也表示衷心的感谢。

由于本人学识水平所限，书中有不妥之处，敬请朋友们批评指正。

编者

1993 年元月

目　录

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	
4	Boot ind	H	S	CYL	7
	SYS ind	H	S	CYL	
	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);
 }
```