计算机实用软件工具系列丛书

TURBO C
实用工具程序设计
原理与技术

李春葆
马玉琳 编著

计算机实用软件工具系列丛书

TURBO C
实用工具程序设计原理与技术

李春葆
马玉琳 编著
希 望 审校

学苑出版社
1993

(京)新登字 151 号

内 容 提 要

本书讲述了如何使用 Turbo C 开发实用工具程序的原理和方法。书中列举了 102 个工具程序，每个程序给出使用方法、设计原理和源程序清单。本书有这样几个特点：一是实用性强；二是技术性高；三是适用面广。本书适用于从事计算机程序设计的同行们参考，也可作为广大师生的教材。

需要本书的用户，请直接与北京 8721 信箱联系，电话 2562329，邮编 100080。

计算机实用软件工具系列丛书

TURBO C 实用工具程序设计原理与技术

编　　著：李春葆　马玉琳
审　　校：希　望
责任编辑：徐建军
出版发行：学苑出版社　　**邮政编码**：100032
社　　址：北京市西城区成方街 33 号
印　　刷：太和印刷厂
开　　本：787×1092　1/16
印　　张：28.875　　**字　　数**：671 千字
印　　数：1～5000 册
版　　次：1993 年 12 月北京第 1 版第 1 次
ISBN7－5077－0757－1/TP・6
本册定价：35.00 元

学苑版图书印、装错误可随时退换

前　言

本书讨论了如何使用 Turbo C 开发实用工具程序的原理和方法。书中列举了 102 个工具程序，每个程序给出使用方法、设计原理和源程序清单，这些程序包括的内容很广，有文件处理的，如连接多个文件、查询和设置文件属性等；有目录处理的，如在计算机系统的各个驱动器的所有路径中查找文件；有加密解密的，如采用磁盘加密法加密文件；有用于打印的，如打印文件部分内容，自动加上行号和页号；有数据库操作的，如读取. DBF 数据库文件的结构和数据；还有其它一些实用工具，如计算器和万年历程序等。

本书有这样几个特点，一是实用性强，书中的程序具有极好的实用性，例如删除目录程序，很多计算机用户在安装一个系统，中途失败退出，不得不花很长时间逐一删除此次安装中所建的各级目录，如果使用本书中的 DELDIR 程序就十分方便，它自动删除用户指定目录中的所有文件与子目录，再例子拷贝空间测定程序 CHECKSPACE，在我们把一批文件从硬盘复制到软盘上时，往往凭借文件尺寸的总和来确定是否放得下，而结果却常常事与愿违，如果用户使用一下 CHECKSPACE，就很方便地确定是否真正拷贝得下。二是技术性高，书中的每个程序，都从原理到设计过程给予了较详细的讨论，并附加了一些例子。使读者清楚地体会到程序设计的技术性，书中的例子并非一般性的学习例子。而是采用 C 语言中难度较高的设计技术，特别适合于有一定 C 语言程序设计基础的读者阅读。三是适用面广，对于不关心 C 语言程序设计的读者，可以不必阅读设计原理部分，只需知道程序的使用方法即可。而对于希望通过这些程序提高程序设计能力的读者可以结合设计原理。使用方法和程序清单，深入领会其中的技巧和方法。

本书的所有程序均已上机通过，其中大部分程序是作者编制的，另外一部分来自同行的经验，作者在此向这些同行们表示衷心感谢。

本书的出版得到北京希望电脑公司秦人华高级工程师的大力支持，作者在此表示深切谢意。

由于时间仓促，书中难免存在错误和不足之处，敬请读者指正。

编　者

目　录

§0　Turbo C 工具程序设计的一些技术和方法 …… (1)
§1　文件属性查询程序 QUATTR …… (12)
§2　文件属性设置程序 FATTRIB …… (16)
§3　文件属性修改程序 ALTER …… (21)
§4　文件属性操作程序 ATTD …… (26)
§5　查找文件程序 FINDFILE …… (29)
§6　查找文件尺寸程序 FILESIZE …… (35)
§7　按照日期查找文件程序 FFDATE …… (39)
§8　按照时间查找文件程序 FFTIME …… (42)
§9　按照尺寸查找文件程序 FFSIZE …… (45)
§10　全盘模糊查找文件程序 SEEK …… (48)
§11　全盘文件查找程序 DISPAF …… (52)
§12　按日期全盘显示文件程序 DISPDF …… (56)
§13　按时间全盘查找文件程序 DISPTF …… (61)
§14　连接两个文件程序 FILEAPP …… (66)
§15　文件复制程序 VCOPY …… (70)
§16　批文件自动分盘拷贝程序 MCOPY …… (75)
§17　复制文件部分内容程序 EXTRACT …… (81)
§18　全盘文件复制程序 COPYAF …… (84)
§19　按日期全盘复制文件程序 COPYDF …… (90)
§20　按时间全盘复制文件程序 COPYTF …… (96)
§21　文件移动程序 MOVE …… (102)
§22　批文件移动程序 VMOVE …… (105)
§23　驱动器之间移动文件程序 FMOVE …… (109)
§24　文件删除程序 VDEL …… (113)
§25　删除文件部分内容程序 REMOVE …… (117)
§26　不可恢复的文件删除程序 FDEL …… (120)
§27　文件不可恢复地删除程序 WIPEFILE …… (123)
§28　多个文件不可恢复地删除程序 VWIPE …… (127)
§29　全盘文件删除程序 DELAF …… (132)
§30　按日期全盘删除文件程序 DELDF …… (136)
§31　按时间全盘删除文件程序 DELTF …… (141)
§32　全面删除备份文件程序 CLEAN …… (146)
§33　多个文件合并程序 MAPPEND …… (150)

§ 34 文件分割和恢复程序 BACKREST …… (155)
§ 35 大型文件的分解与合成程序 LCOPY …… (161)
§ 36 文件分割程序 BKFILE …… (165)
§ 37 灵活的文件分割程序 CUT …… (169)
§ 38 文件个数和尺寸统计程序 FCOUNT …… (172)
§ 39 文件行数统计程序 FNUM …… (175)
§ 40 给文件加行号程序 NUMLINE …… (179)
§ 41 文件大小写转换程序 TRANSCHAR …… (186)
§ 42 设置文件日期和时间标志程序 STAMP …… (189)
§ 43 文件比较程序 COMPARE …… (194)
§ 44 显示 DOS 环境信息程序 DISPDOS …… (198)
§ 45 显示磁盘信息程序 DISPDISK …… (201)
§ 46 剩余内存空间检测程序 CHECKMEMORY …… (203)
§ 47 检测文件占用实际磁盘拷贝空间程序 CHECKSPACE …… (206)
§ 48 存贮和恢复 CMOS 程序 SRCMOS …… (210)
§ 49 硬盘维护程序 DISKTOOL …… (214)
§ 50 内存文本恢复程序 RESTFILE …… (223)
§ 51 快速改变当前目录程序 LCD …… (226)
§ 52 快速列目录程序 LISTDIR …… (229)
§ 53 双列目录显示程序 DDIR …… (234)
§ 54 显示目录树程序 VTREE …… (238)
§ 55 修改目录名程序 RENDIR …… (243)
§ 56 目录复制程序 COPYDIR …… (245)
§ 57 目录快速删除程序 DELDIR …… (249)
§ 58 读取 WPS 文件密码程序 PASSWORD …… (254)
§ 59 WPS 文件解密程序 JMWPS …… (257)
§ 60 文件安全性保密程序 SECURE …… (260)
§ 61 文件加密程序 VAULT …… (263)
§ 62 文件加密和解密程序 CRYPT …… (267)
§ 63 应用程序加密解密程序 JJM …… (270)
§ 64 磁盘法加密程序 DISKJM …… (274)
§ 65 文件深加密程序 JM …… (277)
§ 66 查阅文件部分内容程序 TALK …… (281)
§ 67 阅读文件程序 TTYPE …… (285)
§ 68 文件内容查阅程序 READ …… (289)
§ 69 文件内容显示程序 SHOW …… (295)
§ 70 多文件内容显示程序 CAT …… (298)
§ 71 文件内容查找与删除程序 FDFILE …… (302)
§ 72 多功能文件输出程序 PRNT …… (306)

§73 文件阅读程序 FLRD …… (312)
§74 文件中单词替换程序 REPLACE …… (317)
§75 功能强大的编辑器程序 EDITOR …… (323)
§76 获取数据库结构程序 GETSTRU …… (341)
§77 获取数据库数据程序 GETDATA …… (345)
§78 数据库文件数据拷贝程序 GETDBF …… (349)
§79 数据库结构拷贝程序 DBCOPY …… (353)
§80 数据库压缩程序 COMPRESS …… (356)
§81 数据库恢复程序 EXPAND …… (361)
§82 过程文件分解程序 FOXSPLIT …… (365)
§83 规范化 FOXBASE+程序文件的程序 FORMPRG …… (370)
§84 打印针检测程序 CHECKPRINT …… (376)
§85 打印机单针检测程序 TESTPRINT …… (380)
§86 设置打印机方式程序 PMODE …… (383)
§87 屏幕显示和打印功能程序 DISPRINT …… (387)
§88 文件部分内容打印程序 FLPRINT …… (392)
§89 文件打印程序 FPRINT …… (396)
§90 文件内容打印程序 PRINTOOL …… (400)
§91 多功能文件打印程序 PRINFILE …… (403)
§92 自动联机打印程序 PRINTFLIE …… (409)
§93 统计和过滤控制字符程序 FILTER …… (413)
§94 WPS 文件回车键转换程序 TRANSWPS …… (416)
§95 WS 文件转换成 CCED 文件程序 WTOC …… (419)
§96 WS 文件转换成 ASCII 文件程序 WSASCII …… (422)
§97 删除文件中 TAB 键程序 DETAB …… (425)
§98 WPS 文件显示与删除程序 DISPWPS …… (428)
§99 文件压缩与恢复程序 COMPREST …… (431)
§100 小型计算器程序 COMPUTE …… (436)
§101 显示按键字符程序 DISPCHAR …… (441)
§102 万年历程序 CANDER …… (447)
参考文献…… (453)

§0 Turbo C 工具程序设计的一些技术和方法

Turbo C 工具程序设计是充分利用该系统提供的功能设计出具有通用功能的程序;工具程序有这样几个特点,一是用户界面友好,通常以命令行方式使用;二是程序具有很好的通用性,这种程序不是解决某个特殊问题而是解决一类问题,在设计时应尽量考虑到问题的一般性;三是程序具有很高的可靠性,因为工具程序是面向用户的,应保证在很多情况都是正确的。正因为如此,设计工具程序具有较高的难度,需要充分掌握 Turbo C 的各种程序设计手段,下面归纳一下我们在开发工具程序中使用的一些技术。

1. 使用命令行参数

我们提供的工具程序均以命令行方式使用。把用户在命令行输入的参数转入程序有两种方法,现讨论如下:

1.1 直接获取参数

在主函数中使用 argc 和 argv 参数,前者为命令行中参数的个数,其中程序名计入以内;后者存贮参数值,参数个数的区别以空格作为分隔符,例如,如下命令行:

```
program—name par1 par2... parn
```

则有如下对应关系:

```
argc =n+1
argv[1]=par1
arg[2]=par2
   ⋮
arg[n]=parn
```

我们看一个获取命令行参数的例子程序,其程序清单如下:

```
/* filename:getparam.c */
#include <stdio.h>
main(argc,argv)
int argc;
char *argv[ ];
{
 int i;
 if(argc<=1)
 {
  printf ("\n No parameter ! \n");
  exit(1);
 }
```

```
for(i=1; i<argc;i++)
  printf("parameter %d:%s\n",i,argv[i]);
exit(0);
}
```

输入如下命令：

```
getparam abc 123 xyz
```

显示结果：

```
parameter 1:abc
parameter 2:123
parameter 3:xyz
```

我们便可以在程序中直接处理 argv[ ]中的参数，值得注意的是，这些参数均以字符串方式存贮的。

1.2 利用 PSP 获取参数

我们知道 DOS 把用户的命令行参数均存放在 PSP(程序段前缀)的 0x80 偏移地址处，它们以连续字符串方式存放，即一个程序只有一个参数，我们可以获取该参数后进行相应的处理，例如以下程序便采用了这种方法获取用户在命令行输入的参数，并显示参数的字符数目(空格计入在内)和参数值。

```
/* filename:commline.c */
#include <stdio.h>
#include <string.h>
#include <mem.h>
#include <dos.h>
char comline[128];
main(argc,argv)
int arg

char *argv[ ];

{
 unsigned int comlinelength;
 int i;
 if(argc<=1)
 {
  printf("\n No parameter ! \n");
  exit(1);
 }

 comlinelength=peekb(getpsp(),0x0080);
 memcpy(comline,MK_FP(getpsp(),0x0081),comlinelength);
 comline[comlinelength]=0;

 for(i=0; i<comlinelength; i++)
      if(comline[i]==0)
        comline[i]=32;
        comline[i]=0;
```

```
printf{\n Parameter Length=%u\n",comlinelength);
printf("\n Commline Parameter =%s\n",comline);
exit(0);
```

2. 开辟缓冲区

为了提高程序处理的效率,经常需要开辟缓冲区,一般的方法是,如果开辟缓冲区的尺寸小于 64k,可以采用直接分配,如:

```
char *buffer;
buffer=(unsigned char *) calloc(32767,1);
```

该命令设置 buffer 缓冲区的尺寸为 32767 字节。如果尺寸大于 64k,就要采用远指针,例如如下程序利用这堆函数分配一片 65k 的内存空间,并重新分配该空间:

```
/* FILENAME:mfar.c */
#include <stdio.h>
#include <alloc.h>
main()
{
 char far *block;
 long size=65000;
 printf ("MEMORY ALLOCATE STATUS: \n\n");
 printf ("%lubytes free \n",farcoreleft());
 block=farmalloc(size);
 if (block==NULL)
 {
  printf ("\nfailed to allocated. \n")
  exit(0)
 }
 printf ("%lubytes allocated\n\n",size);
 printf ("%lubytes free\n",farcoreleft());
 size=size12
 farrwalloc(block,size);
 printf("block now reallocated to%lu bytes\n",size);
 printf("%lubytes free \n\n",farcoreleft()
 printf("FREE THE BLOCK:\n");
 farfree (block1);
 printf(" block now freed");
 printf("%lubytes free\n",farcoreleft());
}
```

该程序执行结果如下:

```
MEMORY ALLOCATE STATUS:
  527912 bytes free
  65000 bytes allocated
  462904 bytes free
```

```
block now reallocated to 32500 bytes
430392 bytes free
FREE THE BLOCK:
        block now freedd 4303392 bytes free
```

如果在上述程序中用 malloc()函数代替 farmalloc(),coreleft()函数代替 farcoreleft(),则会显示这样的出错信息:

```
failed to allocated
```

即表示不能分配 65KB 的内存空间。

3. 文件打开

Turbo C 提供了文件打开函数 fopen()和 open()等,它们的使用格式分别如下:

```
FILE *fopen(char *fname,char *mode);
```

mode 的合法值如下:

```
"r"    打开一个文本文件只读
"w"    打开一个文本文件只写
"a"    对一个文本文件添加
"rb"   打开一个二进制文件只读
"wb"   打开一个二进制文件只写
"ab"   对一个二进制文件添加
"r+"   打开一个文本文件读/写
"w+"   生成一个文本文件读/写
"a+"   打开一个文本文件读/写
"rb+"  打开一个二进制文件读/写
"wb+"  生成一个二进制文件读/写
"ab+"  打开一个二进制文件读/写
```

一个文件可以文本或二进制两种方式打开,在文本方式中,输入时回车换行输入被译为换行符,输出时反过来,即把换行符译为回车换行。而在二进制文件中没有这种转换,因此必须考虑编制的程序是处理哪一类文件的,然后采用相应的打开方式。

```
int open(char *fname,int access,int mode);
```

该函数与缓冲型 I/O 不同,不使用 FILE 型文件指针,而是使用 int 型文件说明符,它打开名称为 fname 的文件,并用 access 设置该文件的访问方式。access 取值如下:

```
O_RDONLY 打开文件只读
O_WRONLY 打开文件只写
O_RDWR 打开文件读写
```

在上述值选定一个后,可以把该值与如下一个或多个值以"或"方式一起使用:

```
O_NDELAY 未使用
O_APPEND 在每一写操作之前把文件指针移到文件尾部
O_CREAT 如果文件不存在,生成以 mode 值为属性的文件。
```

O_TRUNC 如果文件存在,把其长度缩短为 0。
O_BINARY 打开一个二进制文件
O_TEXT 打开一个文本文件

fopen()和 open()各有一套相配套的函数,例如使用 fopen()打开的文件必须用 fclose()关闭,而用 open()函数打开的文件需要用 close()关闭。对于有些函数如 read()等必须根据实际情况作相应处理,例如:

```
int fh;
fh=open(fn,O_RDWR);
read (fh,buffer,4096);
      ⋮
close(fh);
```

而如果使用 fopen()函数,上述过程应为:

```
FILE *fp;
fp=fopen(fn,"r+");
handle=fileno(fp);
read(handle,buffer,4096);
      ⋮
fclose(fp);
```

4. 文件复制和生成

文件复制和生成是工具程序中常用的操作。一般有字节复制与生成、行复制与生成和缓冲区复制与生成。下面分别讨论:

4.1 字节复制与生成

我们以两文件复制为例说明,即每次复制一个字节,这样的方法程序设计简单,但效率低,速度慢。

```
fp1=fopen(fname1,"r");
fp2=fopen(fname2,"w");
while((ch=fgetc(fp1))! =EOF)
fputc(ch,fp2);
```

4.2 行复制与生成

这时一次处理一行,如下是采用行复制方法实现两个文件的复制:

```
fp1=fopen(fname1,"r");
fp2=fopen(fname2,"w");
buffer=(unsigned char *) calloc (255,1);
while(! feof (fp1))
{
 if (fgets(buffer,255,fp1))
```

```
    fputs(buffer,fp2);
}
```

采用这种方法在程序设计技术和速度上都比较适中，适合于问题比较简单的领域。

4.3 缓冲区复制与生成

这里指的是采用大尺寸缓冲区的设计方法，上面例子的程序设计如下：

```
fp1=fopen(fname1,"r");
fp2=fopen(fname2,"w");
buffer=(unsigned char *) calloc (32767,1);
inhandle=fileno(fp1);
outhandle=fileno(fp2);
size=filelength(inhandle);
if(size<=32767)
   numtoget=size
else
   numtoget=32767;
numread=0;
do {
    filepos=ftell(fp1);
    fseek(fp1,numread,0);
    _read(inhandle,buffer,numtoget);
    numread+=numtoget;
    fseek(fp2,filepos,0);
    _write(outhandle,buffer,numtoget);
    if ((numread+numtoget)> size)
    numtoget=size-numread;
   }while (numtoget! =0);
```

5. 文件内容的修改

这里指的是程序对文件进行处理，如加密和压缩等，运行的结果反映在文件中，不必生成新的文件，这种情况下，采用的设计方法不用于前面讨论的文件生成，可以使用两个流操作，这里只使用一个流。

下面讨论的例子假设文件的字节发生变化，但文件的尺寸没有变化，这种情况的处理比较简单些，程序中采用了缓冲区。

```
fp=fopen(fname,"r+");
buffer=(unsigned char*) calloc (32767,1);
handle=fileno(fp);
size=filelength(handle);
if(size<32767)
    numtoget=size;
else
    numtoget=32767;
numread=0;
```

```
do {
    filepos=ftell(fp);
    fseek(fp,numread,o);
    _read(handle,buffer,numtoget);
    numread+=numtoget;
    process(buffer;      /* 处理 buffer 中字节,但不改变其尺寸*/
    fseek(fp,filepos,0);
    _write(handle,buffer,numtoget);
    if((numread+numtoget)>size)
        numtoget=size-humread;
}while(numtoget! =0);
```

如果文件内容的处理要改变其尺寸,那么设计起来就很复杂,一般可以使用一个临时文件,这样打开两个文件流,处理后生成临时文件,然后删除原文件,把临时文件换名成原文件名,其过程如下:

```
fp1=fopen(fname,"r");
fp2=fopen("abc.tmp","w");      /*abc.tmp 为一个临时文件*/
process();                     /*处理过程*/
fclose(fp1);
fclose(fp2);
unlink(fname);
rename("abc.tmp",fname);
exit(0);
```

6. 获取参数文件的路径名

如下例子程序就是获取用户指定参数文件的路径名并显示出来。

```
/* filename:getpath.c*/
#include <stdio.h>
#include <dir.h>
main(argc,argv)
int argc;
char *argv[ ];
{
    ⋮
    ⋮
 fnsplit(argv[1],drive,subdir,file,ext);
 sprintf(path,"%s%s",drive,subdir);
 printf("\n path:%s\n",path);
 exit(0);
}
```

7. 几个结构的使用

在设计工具程序时,经常使用 ffblk、dfree 等结构,下面分别进行讨论。

7.1 ffblk 结构

该结构如下：

```
struct ffblk
{
 char ff_reserved[2];        /* DOS 保留字 */
 char ff_attrib;             /* 文件属性 */
 int ff_ftime;               /* 创建的时间 */
 int ff_date;                /* 创建的日期 */
 long ff_fsize;              /* 文件尺寸 */
 char ff_name[13];           /* 文件名 */
};
```

该结构主要与 findfirst()和 findnext()函数对一起使用，找出相匹配的文件，把找到的文件信息存放在该结构中，如下例子程序找出当前目录下所有文件或子目录的名称和属性等。

```
/* filename:findf.c */
#include <stdio.h>
#include <dir.h>
main(argc,argv)
int argc;
char *argv[ ];
{
 int attrib,done;
 struct ffblk f;
 attrib =FA_RDONLY+FA_SYSTEM+FA_HIDDEN+FA_DIREC+FA_ARCH;
 done =findfirst(argv[1],&f,attrib);
 if (done)
 {
  printf("\n No match files in the current directory");
  exit(1);
 }
 while(! done)
 {
  printf("\n %s %c %ld",f.ff_name,f.ff_attrib,f.ff_fsize);
  done=findnext(&f);
 }
}
```

7.2 dfree 结构

该结构如下：

```
struct dfree;
{
 unsigned df_avail;     /* 没有使用的簇数 */
 unsigned df_total;     /* 总的簇数 */
 unsigned df_bsec;      /* 每个扇区的字节数 */
 unsigned df_sclus;     /* 每个簇的扇区数 */
```

```
}
```

该结构一般与 getdfree()函数一起使用，用于检测指定驱动器中的剩余磁盘空间，以 1 代表 A 驱动器，2 代表 B 驱动器等等。如果指定的驱动器无效，则 df_sclus 返回－1。如下例子程序显示所有硬盘驱动器的剩余空间数：

```
/* filename:getdiskfree.c */
#include <stdio.h>
#include <dos.h>
main()
{
 int i;
 struct dfree fspace;

 for (i=3,i<=26,i++)
 {
  getdfree (i,&fspace);
  if (fspace.df_sclus! =-1)
  {
   printf("Drive %c:",i+'A');
   printf("free space:%ld\n",512*fspace.df_avail*fspace.df_bsec);
  }
 }
 exi(0);
}
```

8. 利用 DOS、BIOS 系统资源

我们在程序中经常通过利用 DOS 和 BIOS 系统资源达到程序设计事半功倍的效果，这些资源调用的方法是相似，我们通过如下几个子程序说明，使用 intdos()函数表示调用 DOS 中断，使用 int86()函数表示调用 BIOS 中断。

• *ctrl_break_status*:

该子程序的功能是获取或设置 Ctrl_break 状态，如果 function 为 0，则返回当前 Ctrl_break 状态，若为 1 则设置当前 Ctrl_break 状态；如果 status 为 0，则不能检测 Ctrl_break 状态，为 1 则可以检测 Ctrl_break 状态。

```
#include <dos.h>
int Ctrl_break_status(int function,int state)
{
 union REGS inregs,outregs;
 inregs.h.ah=0x33;
 inregs.h.al=function;
 inregs.h.dl=state;
 intdos(&inregs,&outregs);
 return(outregs.h.dl);
}
```

get_disk_transfter_address:

该子程序的功能是返回 DOS 磁盘转换区的段地址和偏移地址。

```
#include <dos.h>
void get_disk_transfer_address(segment,offset)
int *segment,*offset;
{
 union REGS inregs,outregs;
 struct SREGS segregs;
 inregs.h.ah=0x2F;
 intdos (&inregs.&outregs,&segres);
 *segment=segregs.es;
 *offset=outregs.x.bx;
}
```

get_interrupt_vector:

该子程序的功能是返回一个特定中断的中断子程序的地址。

```
#include <dos.h>
void get_interrupt_vector(int interrupt_number,int *segment,int *offset)
{
 union REGS inregs,outregs;
 struct SREGS segregs;
 inregs.h.ah=0x35;
 inregs.h.al=interrupt_number;
 intdos(&inregs,&outregs,&segregs);
 *segment=segregs.es;
 *offset=outregs.x.bx;
}
```

scroll_up;

该子程序的功能是上滚某个屏幕区域的内容,输入参数有:numlines 为上滚的行数;attribute 是滚动的空行所用的属性,top_row 为左上角行号,left_column 为左边列号,bottom_row 为右下角行号,right_column 为右边列号。

```
#include <dos.h>
void scroll_up (int numlines,int attrbute,int top_row,int bottom_row,int left_column,int right_column)
{
 union REGS inregs,outregs;
 inregs.h.ah=6;
 inregs.h.al=numlines;
 inregs.h.bh=attribute;
 inregs.h.ch=top_row;
 inregs.h.dh=bottom_rom;
 inregs.h.cl=left_column;
 inregs.h.dl=right_column;
 int86(0x10,&inregs,&outregs);
```