# 第八章　数据文件处理技术

C文件概述

文件类型和文件类型指针变量
几个常用的数据库函数
文件处理程序结构和常用文件库函数
文件程序设计实例

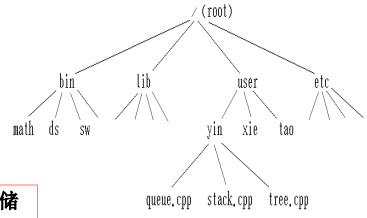
●要求：

1）熟悉文件及文件类型指针的概念；
2）熟练掌握文件的打开、关闭、读写与检测等针对文件的基本操作；

# C文件概述

## 一、文件定义与分类

### 1、文件




一个文件目录结构示意图

文件是存储在外部存储介质上的信息的集合。

每个文件有唯一的文件名（主名.后缀）来标识。计算机实现按名对文件进行读、写等有关操作。

广义上，操作系统将每一个与主机相联的输入输出设备都看作是文件。（显示器、打印机是输出文件，键盘是输入文件）。

## 2、文件的分类

（1）按存储介质：
磁盘文件、磁带文件等。

（2）按文件的内容：
源程序、目标文件、数据文件等。

（3）按文件的编码方式（存储形式）：
文本文件、二进制文件。

（4）按文件操作可划分为输入文件、输出文件、输入输出文件。

## 二、数据文件的存储形式

1、字符文件（正文文件）

文件中数据是字符，每个字符以ASCII 代码存储，占一个字节。存储数值数据要占较多的存储空间，输入输出时因内存和外存的存储形式不一致，还要进行内外表示形式之间的转换。正文文件数据流是字符，能让程序对文件作逐个字符处理和文件中的数据能供人阅读。

2、二进制文件：

文件中的数据按其在内存中的存储形式存储在文件中。存储数值数据只占其内部表示所需的字节数；用二进制形式存储数值数据可以节省外存空间，并免去数据内外表示形式之间的转换；用于程序与程序或程序与设备之间传递成批数据信息。

如：一个任意整数（short型），
其二进制文件存储方式都是占2个字节；
而字符文件存储方式所占字节数＝该数的数字个数。

| short型数 | 二进制文件 | 字符文件 |
|---|---|---|
| 126 | 2字节 | 3 字节 |
| 1268 | 2字节 | 4 字节 |

例如：整数1268

文本文件形式

| 31 | 32 | 36 | 38 |
|---|---|---|---|
| 00110001 | 00110010 | 00110110 | 00111000 |
| '1'的ASCII码 | '2'的ASCII码 | '6'的ASCII码 | '8'的ASCII码 |

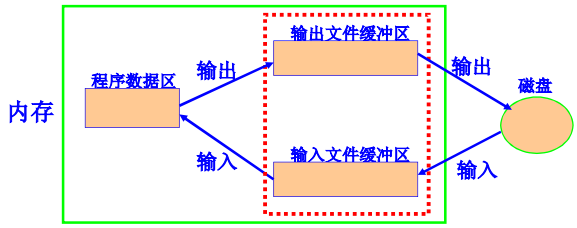
二进制文件形式

| 4 | F4 |
|---|---|
| 00000100 | 11110100 |

1268的二进制数

## 三. 标准文件系统、非标准文件系统

标准c:采用缓冲文件系统，其原理图如下：




缓冲文件系统的特点是：系统在内存为正在使用的每一个文件开辟一个固定容量的“缓冲区”，当执行读文件的操作时，从磁盘上将指定的文件数据先读入缓冲区，装满后再从缓冲区逐个将数据送到程序数据区；当执行写文件的操作时，先将程序数据写入缓冲区，待缓冲区装满后再送到磁盘。

非缓冲文件系统的特点是：

系统不自动为正在使用的每一个文件开辟一个固定容量“缓冲区”，而由程序根据自身的需要及系统的存储资源情况来为每一个文件设定缓冲区。目前仍有许多C版本支持非缓冲文件系统，但1983年ANSI C标准决定不采用非缓冲区文件系统，因此建议不要采用不符合ANSI C标准的那些部分，以免降低程序的可移植性。

## 4. 文件存取方式

在C语言中,文件的存取有两种方式：

（1）顺序存取

（2）随机存取,利用文件位置指针

## 8.1 文件类型和文件类型指针变量

### 1. 文件类型指针概述

文件类型指针是缓冲文件系统中最重要的概念。对缓冲文件系统来说，ANSI C为每个被使用的文件在内存开辟一小块固定大小的区域，用于存放文件的属性状态(如文件的名字、文件的性质、文件的当前状态等信息)，该区域利用一个结构类型变量存放。该变量的结构体类型是由系统定义的，取名为FILE，其定义包含在头文件stdio.h中，不同系统的文件类型所含内容也不全相同。格式如下：

```
typedef struct
 { short  level; /* 缓冲区“满”或“空”的标志 */
  unsigned flags; /* 文件状态标志 */
  char fd; /* 文件描述符 */
  unsigned char hold; /* 若无缓冲区不读取字符 */
  short bsize; /* 缓冲区的大小 */
  unsigned char *buffer; /* 数据缓冲区的位置 */
  unsigned char  *curp; /* 当前活动指针 */
  unsigned istemp; /* 临时文件描述符 */
  short token; /* 用于有效性检查 */
}FILE;
```

### 2. 文件型指针变量的定义

在操作文件以前，应先定义文件变量指针。FILE类型允许定义若干FILE类型指针变量，以便存放若干待操作文件的信息，定义格式为：

FILE *指针变量标识符;

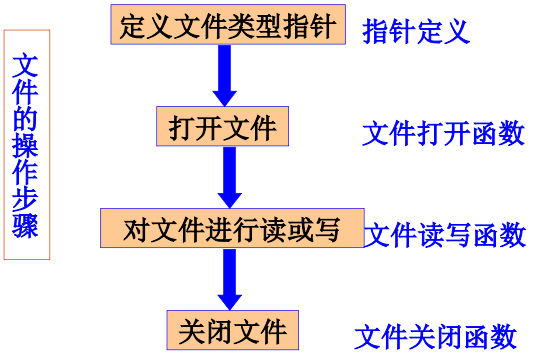
如 FILE *fp1, *fp2;

按照上面的定义，fp1和fp2均为指向FILE类型的指针变量，允许系统打开2个可操作的文件。换句话说，一个文件有一个文件变量指针，今后对文件的访问，会转化为针对文件变量指针的操作。

如果有N个文件，一般应设N个指针变量，使它们分别指向N个文件（确切地说，指向该文件的信息结构体），以实现对文件的访问。

标准文件操作的四个基本步骤：

1. 文件类型指针的定义
2. 打开标准文件
3. 标准文件的读或写的操作
4. 标准文件的关闭操作




## 8.2 几个常用的数据库函数

打开文件的含义

将文件信息从磁盘装入计算机内存，建立文件的各种有关信息，并使文件指针指向该文件，即建立文件类型指针与文件名之间的关联。因此在读写文件之前，先得打开文件。

1. 文件的打开函数：fopen()

函数原型：

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

**fp=fopen(文件名，使用文件方式）；**

返回值：FILE类型指针----成功
　　NULL ----失败

fopen()函数的返回值应赋给一个FILE指针变量，否则，此函数的返回值就会丢失，导致无法对此文件进行操作。

例如：
```
FILE   *fp;
fp=fopen("A1", "r");
```

以上语句表示：

要打开名字为A1的文件，使用文件方式为“读入”，fp指向A1文件。

另外文件名(可能还包括目录路径)为字符串表达式。使用方式也是一个字符串，用来指明文件的读写方式(见表8-1)。如语句

第1个\为转义字符

```
fp = fopen("\\usr4\\smp.dat", "r");
```

以读方式打开根目录下的usr4子目录中的 smp.dat 文件。

调用函数fopen()时，可能会不能打开。如读方式下打开不存在的文件；在写方式下，外部存贮介质已无剩余的自由空间，或外设故障，或超过系统能同时打开的文件数。文件不能打开时，函数fopen()将返回一个空指针值NULL。考虑到文件不能正常打开的极端情况，所以常用以下形式的C代码描述打开文件：

```
if ((fp = fopen(filename, "r")) == NULL)
{
    printf("Can not open %s file.\n", filename);
    return;
}
```

## 文件名

- 文件名是一个字符串，可以是字符串常量，也可以是字符数组、字符指针等。
  - fopen("ascii.txt","r");
  - fopen("d:\\joe\\ascii.txt","r");
  - char *file_name="ascii.txt";
  - fopen(file_name,"r");
  - char filename[20];
  - strcpy(file_name,"ascii.txt");
  - fopen(file_name,"r");

## 文件打开函数的调用给编译系统的三个信息：

1. 需要打开的文件名；
2. 使用文件的操作方式；
3 . 让指针变量指向被打开的文件。

C语言文件操作方式　　表8-1

| 文件使用方式 | 意义 |
|---|---|
| "r" | 只读，为读打开正文文件 |
| "w" | 只写，为写打开正文文件 |
| "a" | 追加，从正文文件尾开始写 |
| "rb" | 只读，为读打开二进制文件 |
| "wb" | 只写，为写打开二进制文件 |
| "ab" | 追加，从二进制文件尾开始写 |
| "r+" | 读写，为读/写打开正文文件 |
| "w+" | 读写，为读/写建立并打开新的正文文件 |
| "a+" | 读写，为读/写打开正文文件 |
| “rb+"或“r+b" | 读写，为读/写打开二进制文件 |
| “wb+"或“w+b" | 读写，为读/写建立并打开新的二进制文件 |
| "ab+"或"a+b" | 读写，为读/写打开二进制文件 |

## 关于fopen()使用方式参数的几点说明以下：

(1)用"r"方式打开的文件只能用于向计算机输入数据，而且该文件应该已经存在；

(2)用 "w" 方式打开的文件只能用于向文件输出数据。如打开时，文件不存在，则新建一个以指定名字命名的文件；如原文件已存在，则原文件上的数据被全部删除。

(3)打开文件用于写，不删除原文件中的数据，从原文件的末尾开始添加新数据，用 "a" 方式。

(4)用 "r+"、 "w+"、 "a+" 方式，可以输入和输出。用 "r+" 方式只允许打开已存在的文件；用 "w+" 方式可新建一个文件；用"a+"方式打开一个已存在的文件，位置指针先移到文件的末尾，准备添加数据，也可以输入数据。

(5)要打开二进制文件，只要在对应正文件文件打开方式中接上字符b即可，如”rb"表示以输入方式打开二进制文件。

正文文件与二进制文件在使用时，还有一点不同。对于正文文件，输入时，回车符和换行符合成为一个换行符输入；输出时，换行符('\n')转换成为回车符和换行符两个字符一起输出。对于二进制文件，不进行上述这种转换，二进制文件中的数据形式与在内存中的数据形式是完全一致的。

(6) 标准输入文件（键盘）,标准输出文件（显示器），标准出错输出（显示器）是在程序运行时由系统自动打开的，可直接使用而无需打开与它们相连的终端文件。

### 2. 文件关闭库函数:fclose()

在使用完一个文件后，程序应该立即关闭它。关闭文件可调用库函数 fclose() 来实现。

函数原型： int　fclose(FILE *fp);

功能：关闭文件指针fp所指的文件,释放相应的文件信息区. 正常关闭文件时,函数返回值为0.在使用完一个文件后应该关闭它，以防止它再被误用。

调用函数 fclose() 的一般形式为

　　fclose(文件指针);

例如　fclose(fp);

调用函数fclose()的作用是使文件指针变量终止原先调用函数fopen()时所建立的它与文件的联系。调用函数fclose()之后，不能再通过该文件指针变量对其原先相连的文件进行读写操作，除非被再次打开。文件被关闭后，原文件指针变量又可用来打开文件，或与别的文件相联系，或重新与原先文件建立新的联系。若关闭文件操作成功，fclose()函数返回值为0；否则返回EOF（－1）。

## 文件处理

当文件按指定的工作方式打开以后，就可以执行对文件的读和写。针对文本文件和二进制文件的不同性质，对文本文件来说，可按字符读写或按字符串读写；对二进制文件来说，可进行成块的读写或格式化的读写。C语言提供多种文本文件读写函数这些函数原型都包含在头文件stdio.h中。

C语言提供的主要文件读写函数

- ◆字符读/写函数：fgetc(),fputc()
- ◆字符串读/写函数：fgets(),fputs()
- ◆格式读/写函数：fscanf(),fprintf()
- ◆成批读/写函数：fread(),fwrite()

文件定位

- ◆rewind(),fseek(),ftell()

文件测试　◆feof()

### 3. 读文件字符函数:fgetc

fp是指向所读文件指针变量

函数原型: int fgetc(FILE *fp) ;

函数功能:
从文件指针fp指向的文件当前位置(位置指针)读出一个字符,然后文件位置指针自动后移,指向文件中的下一个字符,返回值为读入的字符.
若遇到文件结束符,则返回结束符EOF(-1).

例:ch=fgetc(fp);

早先介绍的getchar()函数就是从fputc函数派生出来的.用函数fgetc()定义的宏:
#define getchar() fgetc(stdin)

【例1】: 将c磁盘文件"data.txt"的信息读出并显示到屏幕上.(假设文件已经存在,若不存在,可以用记事本或其它文本编辑器编辑一个)

```
#include <stdio.h>
void main()
{
    FILE *fp;
    char c;
  if ((fp=fopen("c:\\data.txt","r" ))==NULL)
  {
    printf("\n File notexist!");return;
  }
  while((c=fgetc(fp))!=EOF)//文本文件结束标志:EOF (-1)
    putchar(c);
  fclose(fp);
}
```

### 4 .文件写入字符函数: fputc

函数原型: int fputc(int ch, FILE *fp);

格式: fputc(ch, fp)

函数功能:将ch中的字符输出到文件指针fp指向的文件中(类似于putchar函数)。
函数返回值:输出成功,返回输出的代码;输出失败则返回EOF(在stdio.h文件中定义为 -1)。

早先介绍的putchar()就是用函数fputc()定义的宏:
#define putchar(ch) fputc(ch, stdout)

【例2】从键盘上输入字符,依次送入指定的文件,直到输入一个"#"为止。

```
#include <stdio.h>
void main()
{ FILE *fp;   char ch, fname[40];
  printf("输入文件名: "); scanf("%s", fname);
  fp = fopen(fname, "w");
  fflush (stdin);   /* 清输入缓冲区 */
  printf ("开始输入字符串: \n");
  ch = fgetc(stdin);
  while (ch != '#') {
   fputc (ch, fp);  ch = fgetc(stdin);
  }
  fclose (fp);  /* 关闭文件 */
}
```

【例3】将文件dataa.txt的内容复制到文件datab.txt中。

```
#include <stdio.h>
void main()
 {   FILE *f1,*f2;
     char c;
    if ((f1=fopen( "c:\dataa.txt" , "r" ))==NULL)
    {
      printf(" \n File1 cannot open! ");return ; }
    if ((f2=fopen( "c:\\datab.txt" , "w" ))==NULL)
    {printf("\n File2 cannot creat!");return ; }
    while((c=fgetc(f1 ))!=EOF )
          fputc ( c,f2 );
    fclose(f1); fclose(f2);
}
```

## 二进制文件的读处理

- 基本结构

```
FILE *fp;     /* 文件指针 */
char c; /* 用来存放读出的字符 */
fp = fopen(文件名, "rb"); /* 读方式 */
while(!feof(fp)) {
   c = fgetc(fp); ...
}
fclose(fp);/* 关闭文件 */
```

- 二进制文件结束判断函数: feof

## 二进制文件的写处理

- 基本结构

```
FILE *fp;
char c;
fp = fopen(文件名, "wb"); /* 写方式 */
while(还有字节) {
   ...
   fputc(c, fp); /* 输出字符到文件 */
}
fclose(fp);
```

## 文件结束的判别方式

1. 文件结束检测函数feof

函数原型：int feof (FILE *fp)；

函数功能：判断fp指向的文件是否处于文件结束位置，如文件结束，则返回值为1，否则为0。

- 文本文件

  (c=getc(fp)) == EOF

  feof(fp)

- 二进制文件

  feof(fp)

  第一种方法不能用是因为-1（0xFF）可能是二进制数值

## 标准文件

- 标准输入文件
  - stdin，默认为键盘。
- 标准输出文件
  - stdout，默认为屏幕显示。
- 标准出错输出文件
  - stderr，默认也是屏幕显示。

## 5. 格式化写读函数:fprintf和fscanf

fprintf一般调用方式为：

fprintf（文件指针，格式字符串，输出表列)；
按照指定格式，向文件写

功能：
作用类似于printf函数。
函数返回值：
输出成功，返回实际输出的字符数；输出失败则返回EOF。

例如：
fprintf (wp, "i = %d r = %6.4f\n", i, r);
表示：将整型变量 i 和实型变量 r 的值按格式输出到 wp 指向的文件中。

fscanf一般调用方式为：

fscanf（文件指针，格式字符串，输入表列)；
按照指定格式，从文件读

功能：
作用类似于scanf函数。
函数返回值：
读取成功,返回实际读入数据的个数,读入失败则返回EOF。
例如： fscanf(rp, "%d %f", &i, &r);
表示：从 rp 指向的文件上为整型变量 i 和浮点型变量 r 读入数据。

注意：fprintf和fscanf函数对磁盘文件读写
printf和scanf函数对显示器或键盘操作。

由于引入了这两个格式化输入/输出函数，使得文件在存储格式上满足某种指定格式。例如：

```
fscanf(fp,"%d%s",&i,s);
fprintf(fp,"%d%s",i,s);
```

注意：一般情况下是用什么格式写入文件，就用什么格式从文件读入数据，否则，读出的数据就会与写入的数据不一致。

【程序8.1】从键盘上输入整数序列，并按输入顺序输出到指定的文件。然后再读出到显示器.设文件名在程序启动时由输入指定。

```
#include <stdio.h>
int main()
{       FILE *fp;
        char fname[40];int x,k;
        printf("输入文件名!\n");
        scanf("%s%*c", fname); /* 读入文件名和名后的回车符 */
        fp = fopen(fname, "w");
         k=1;
        printf("Please input data\n");
        while(scanf("%d",&x)==1){ printf(fp,"%d\t",x);
             if(k++%5==0) fprintf(fp,"\n");}
        printf("\n共输出了%d个整数到文件%s.\n",k-1,fname);
        fclose(fp);
        fp = fopen(fname, "r");
        while(fscanf(fp,"%d",&x)==1)printf("%d\t",x);
        printf("\n");

}
```

## 6. 读写,文件字符串函数:fgets和fputs( )

```
char  *fgets( char *buf,int n,FILE  *fp)  ;
```

函数功能：

从指针fp指向的文件中读取 n – 1 个字符，把它送到由指针buf 指向的字符数组中。

说明：

1.在读入n-1个字符结束之前，遇到换行符或EOF，读字符过程结束。

2.字符串读入后，在最后自动加一个‘\0’。

3.存储读入的换行符。而函数gets()不存储读入的换行符。

函数返回值：str 的首地址。

例如（将文件内容一行一行地输入）：

```
char a[80]; FILE *in;
while ( !feof(in))
   printf ("%s", fgets (a, 80, in));
```

```
int  * fputs ( char  * buf, FILE  * fp ) ;
```

函数功能：将字符串输出到 fp 指向的文件。

说明：

1、buf 可以是字符串常量、字符数组或字符指针。

2、不在复制的字符序列之后另外再添加换行符。

函数返回值：输出成功为非负值；输出失败为EOF。

例如：fputs ("China", fp);

表示：将字符串 "China" 输出到 fp 指向的文件。

【例4】利用函数fgets,将文本文件data.txt中的内容全部读出并显示在屏幕上。

```
#include <stdio.h>
void  main()
{    FILE *fp;
     char str[81];
     if ((fp=fopen( "c:\\data.txt" , "r" ))==NULL)

     {
            printf(" \n File1 cannot open! ");
            return ;
       }
     while(fgets(str,81,fp)!=NULL )puts(str);
     fclose(fp);
}
```

【例5】从键盘输入若干行字符，将它们添加到磁盘文件data.txt中。

```
#include <stdio.h>
void  main()
{    FILE *fp;
     char str[81];
     if ((fp=fopen( "c:\\data.txt" , "a" ))==NULL)

     {
            printf(" \n File1 cannot open! ");
            return ;
       }
     while(fgets(str,81,stdin)!=NULL )fputs(str,fp);
     fclose(fp);
}
```

前面介绍的几种读写文件的方法，对于复杂的数据类型无法以整体形式向文件写入或从文件读出。为了解决这个问题，C语言提供成块的读写方式来操作文件，使数组或结构体等类型可以进行一次性读写。C语言提供的数据块读写函数fread()和fwrite()，可用来读写一组数据（一个数据块）。

## 7.成批读写函数:fread和fwrite

函数原型:

```
int fread(void *buffer,int size,int count,FILE *fp);

int fwirte(void *buffer,int size,int count,FILE  *fp);
```

buffer:是一个指针。

对于fread，它是读入数据的存放地址。

对于fwrite，是要输出数据的地址（起始地址）。

size:　要读写的字节数。

count:　要进行读写多少个size字节的数据项。

fp:　　文件型指针。

其功能是：

(1) fread()函数从打开的文件fp中读取n项数据，每一项数据的长度为size字节，放入指定的缓冲区buffer中，所读的字节长度为n×size。函数调用成功后，返回实际读的数据的数据项，若遇文件结束或出错则返回0。

(2) fwrite( )函数把buffer所指向的n个数据项（每个数据项的长度为size个字节）写入到已经打开的文件fp中。函数返回值为写到文件中的数据项个数。

```
typedef struct   {          // 应用实例
          char name[21];     /* 名字 */
          char phone[15];    /* 电话 */
          char  zip[10];     /* 邮编 */
          char addr[31];     /* 地址 */
 } STUTYPE;
```

利用类型STUTYPE定义的结构数组

```
STUTYPE  stud[30];
```

能存放30个通信录数据。则下面的两个函数调用能分别实现20个通信录数据从文件读出和写入文件：

```
fread(stud, sizeof(STUTYPE),  20, rfp);
fwrite(stud, sizeof(STUTYPE), 20, wfp);
```

如果要逐个读入或写出结构数组的每个元素，也可用循环实现:

```
for(i = 0; i < 20; i++)
 fread(&stud[i], sizeof(STUTYPE), 1, rfp);

for(i = 0; i < 20; i++)
fwrite(&stud[i], sizeof(STUTYPE), 1, wfp);
```

调用函数fread()和fwrite()也有返回值，它的返回值是实际完成输入或输出的数据块的个数。

## 文件的定位

前面介绍的对文件的读写方式都是顺序读写，即读写文件只能从头开始，顺序读写各个数据。 在实际问题中常要求读写文件中某一指定的部分，为了解决这个问题，可以将文件内部的位置指针移动到需要读写的位置，再进行读写操作。这种读写操作称为随机读写。按要求移动文件内部的位置指针的操作称为文件的定位。

C语言提供的文件内部位置指针移动函数主要有rewind()函数和fseek()函数：rewind()函数的功能是把文件内部的位置指针移到文件首，这个函数在前面已多次使用过；fseek()函数用来移动文件内部位置指针到某一指定位置。

8. 函数:rewind(),fseek(),ftell()

文件随机存取是指读写完一个字符(或字节)之后，并不一定要读写其后继的字符(或字节)，可以改变当前位置去读写文件中任意位置上的字符(或字节)。

程序即刻能读当前读位置上的数据，即刻输出的数据存于当前写位置之后。对于顺序读写情况，每读写一个字符后，当前位置就会自动向后移一个字符位置。

允许随机读写文件，就得改变文件当前读写位置。改变当前位置可调用库函数rewind()和fseek()来实现。另设有函数ftell()用于查询文件当前读写位置。

rewind()函数原型：

int rewind (FILE *fp)；

函数功能：将fp指向的文件中的当前读写位置回到文件之首。该函数没有返回值。

fseek()函数原型：

int fseek(FILE *fp,long offset, int ptrname)；

函数功能：

将文件的当前位置任意移动，实现随机读写。

说明： ptname为文件读写的起始位置，只允许0、1或2。0表示文件首（或用SEEK_SET表示）； 1表示当前位置（或用SEEK_CUR表示）； 2表示文件尾（或用SEEK_END表示）。

offset 为long型的位移量，以ptname为基准，移动的字节数。

起始点表示方法

| 起始点 | 表示符号 | 数字表示 |
|---|---|---|
| 文件首 | SEEK_SET | 0 |
| 当前位置 | SEEK_CUR | 1 |
| 文件末尾 | SEEK_END | 2 |

若offset为负数，表示向文件首方向移动，否则是向文件尾方向移动；

调用函数 fseek() 的例子：

fseek(fp,40L, SEEK_SET);当前位置定于离文件头40个字节处

fseek(fp,20L, SEEK_CUR);文件当前位置定于离当前位置20个字节处

fseek(fp,-30L, SEEK_END);将文件的当前位置定于文件尾后退30个字节处

说明：fseek 一般用于二进制文件的随机读写。

ftell()函数原型：

Long fseek(FILE *fp)；

函数功能：

用于得到文件当前位置相对于文件首的偏移字节数。在随机方式存取文件时，由于文件位置频繁的前后移动，程序不容易确定文件的当前位置。调用函数ftell()能非常容易地确定文件的当前位置。利用函数 ftell()也能方便地知道一个文件的长：

fseek(fp, 0L, SEEK_END);当前位置移到文件的末尾

len = ftell(fp)；获得当前位置相对于文件首的位移

## 8.3 文件处理程序结构(自学)

文件处理程序，包含以下与文件有关的几项内容：

(1) 定义文件指针变量，和存储文件名的字符数组：

```
#include  <stdio.h>
FILE *fp; /* 定义文件指针变量fp */
char fname[40]; /* 存储文件目录路径和文件名的字符数组，最多可有39个字符 */
```

(2)输入文件的名。如以下代码所示：

```
printf("输入文件名(包括目录路径、扩展名)\n");
scanf("%s%*c",fname);/* 输入文件名及回车符 */
```

(3)先打开文件，这里只介绍两种打开方式：

若让程序从正文文件输入数据，用以下代码：

```
if ((fp = fopen(fname, "r")) == NULL){
    printf("%s文件不能打开\n", fname);
    return;
}//读打开时，要求被打开文件已存在;
```

若让程序向正文文件输出结果，用以下代码：

```
fp=fopen(fname,"w");  /* 为写打开文件*/
```

若被打开文件不存在，则建立一个新文件；若被打开文件已存在，则该文件中的数据被删除。

(4)文件使用结束后，要及时关闭，如以下代码所示：

```
fclose(fp);  /* 以后fp又可用于打开文件 */
```

(5)调用有关文件输入输出库函数。

调用函数fgetc（）从文件输入下一个字符，如：

```
ch=fgetc(fp); /* 将输入字符存于变量ch */
```

调用函数fscanf()从文件按指定格式输入数据，如：

```
fscanf(fp,"%d%d",&k,&j); /* 从文件输入整数 */
```

调用函数fputc（）向文件输出一个字符，如：

```
fputc(ch, fp); /* 将变量ch中的字符输出到文件*/
```

调用函数fprintf()向文件按指定格式输出数据，如：

```
fprintf(fp,"%d  %d\n", k, j);
```

正文文件逐个字符输入的程序结构

```
int c;  /* 不能为char类型 */
FILE *fp;
 …  /* 说明有关变量和设置初值等 */
if ((fp = fopen(文件名, "r")) == NULL) {
   printf("不能打开文件 %s。\n", 文件名);
   return;
}
while((c = fgetc(fp)) != EOF)  {
  … /* 对刚读入的字符信息 c 作某种处理  */
}
fclose(fp);
… /* 输出处理结果 */
```

二进制文件逐个字节输入的程序结构

```
       …
      char c;  /* 也可以是int类型 */
       FILE *fp;
       … /*这里插入说明有关变量和设置初值等的代码 */
       fp = fopen(文件名, " rb");
       while(!feof(fp)) {       c = fgetc(fp);
     …
   /*这里插入对刚读入的字节信息c作某种处理的代码  */
       }
… /* 这里插入输出处理结果的代码 */
```

字符(或字节)逐一输出形成新文件的程序结构：

```
int c;  /* 也可以是char类型 */
FILE *fp;
 …  /* 说明有关变量和设置初值等 */
fp = fopen(文件名,"w");
while(还有字符)  {
  …  /* 生成字符(或字节)存于变量c */
  fputc(c, fp); /* 将生成的字符输出 */
}
fclose(fp);
  … /* 输出程序结束报告 */
```

## 8.4 文件处理程序设计实例

【例8.4】输入一篇英文短文，统计文件中的行数、单词数和字符数的程序。
设全由英文字母组成的一段连续字符序列为一个英文单词。程序为统计单词数，需要识别一个单词的开始和结束，程序引入一个状态变量。如果程序遇到一个非英文字母字符，程序设置状态不在单词中；如果程序遇到一个英文字母字符，程序的原先状态又不在单词中，表示程序遇到一个新的单词，程序将单词计数器增1，并置状态在单词中。

```
#include  <stdio.h>
#include  <ctype.h>
#define  INWORD   1    /* 正在单词中 */
#define  OUTWORD  0    /* 当前不在单词中 */
FILE *fp;
int main()
{  int nl, nw, nc, ch, state /* 状态变量 */;
   char fname[40];  /* 存储文件名 */
   printf("输入文件名!\n");
   scanf("%s%*c", fname);
 if ((fp=fopen(fname,"r")) == NULL) {
       /* 以输入方式打开正文文件 */
       printf("不能打开文件 %s。\n",fname);return 0;
      }
     state = OUTWORD; nl = nw = nc = 0;
```

```
      while((ch = fgetc(fp)) != EOF) {
          /* 这里对刚输入的字符信息 ch 作某种处理  */
          ++nc;  if (ch == '\n') ++nl;
          if (!isalpha(ch))//ch中内容不是英文字母
                state = OUTWORD;
          else if (state == OUTWORD){/*从原先不在单词中，遇到了英文字母*/
                state = INWORD;//置状态在单词中
            ++nw; //单词计数器增1
          }
       }
```

nc:字符计数器
nl: 行计数器
nw:单词计数器