

第10章

对文件的输入输出

C文件的有关基本知识

什么是文件

文件有不同的类型，在程序设计中，主要用到两种文件：

- (1) **程序文件**。包括源程序文件(后缀为.c)、目标文件(后缀为.obj)、可执行文件(后缀为.exe)等。这种文件的内容是程序代码。
- (2) **数据文件**。文件的内容不是程序，而是供程序运行时读写的数据，如在程序运行过程中输出到磁盘(或其他外部设备)的数据，或在程序运行过程中供读入的数据。如一批学生的成绩数据、货物交易的数据等。

为了简化用户对输入输出设备的操作，使用户不必去区分各种输入输出设备之间的区别，**操作系统把各种设备都统一作为文件来处理**。从操作系统的角度看，每一个与主机相连的输入输出设备都看作一个文件。例如，终端键盘是输入文件，显示屏和打印机是输出文件。

什么是文件

文件 (file) 一般指**存储在外部介质上数据的集合**。操作系统是以文件为单位对数据进行管理的。

输入输出是数据传送的过程，数据如流水一样从一处流向另一处，因此常将输入输出形象地称为**流** (stream)，即**数据流**。流表示了信息从源到目的端的流动。在输入操作时，数据从文件流向计算机内存，在输出操作时，数据从计算机流向文件(如打印机、磁盘文件)。

C语言把文件看作一个字符(或字节)的序列，即由一个一个字符（或字节）的数据顺序组成。一个输入输出流就是一个字符流或字节(内容为二进制数据)流。

C的数据文件由一连串的字符（或字节）组成，而不考虑行的界限，两行数据间不会自动加分隔符，对文件的存取是以字符（字节）为单位的。输入输出数据流的开始和结束仅受程序控制而不受物理符号（如回车换行符）控制，这就增加了处理的灵活性。这种文件称为流式文件。

文件名

一个文件要有一个唯一的文件标识，以便用户识别和引用。

文件标识包括3部分：(1)文件路径；(2)文件名主干；(3)文件后缀。

D:\CC\temp\file1.dat

↓ ↓ ↓

文件路径 文件主干名 文件后缀

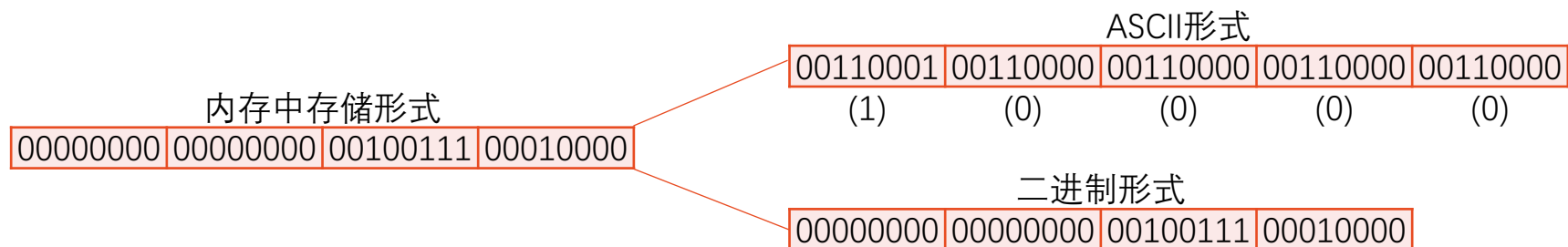
- 文件路径表示文件在外部存储设备中的位置。
- 文件名主干的命名规则遵循标识符的命名规则。
- 文件后缀用来表示文件的性质。

为方便起见，文件标识常被称为文件名，但应了解此时所称的文件名，实际上包括以上3部分内容，而不仅是文件名主干。

文件的分类

根据数据的组织形式，数据文件可分为**ASCII文件**和**二进制文件**。数据在内存中是以二进制形式存储的，如果不加转换地输出到外存，就是二进制文件，可以认为它就是存储在内存的数据的映像，所以也称之为**映像文件**(image file)。如果要求在外存上以ASCII代码形式存储，则需要在存储前进行转换。ASCII文件又称**文本文件** (text file)，每一个字节存放一个字符的ASCII代码。

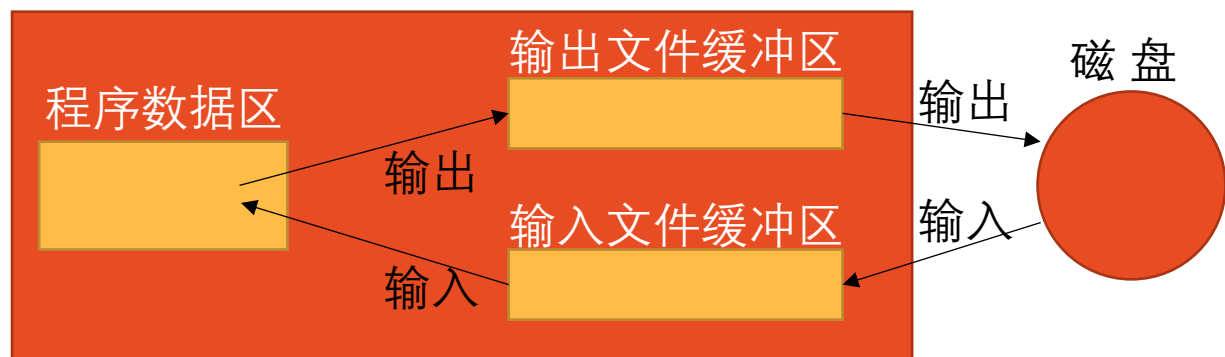
字符一律以ASCII形式存储，数值型数据既可以用ASCII形式存储，也可以用二进制形式存储。



用ASCII码形式输出时字节与字符一一对应，一个字节代表一个字符，因而便于对字符进行逐个处理，也便于输出字符。但一般占存储空间较多，而且要花费转换时间（二进制形式与ASCII码间的转换）。用二进制形式输出数值，可以节省外存空间和转换时间，把内存中的存储单元中的内容原封不动地输出到磁盘(或其他外部介质)上，此时每一个字节并不一定代表一个字符。

文件缓冲区

ANSI C标准采用“**缓冲文件系统**”处理数据文件，所谓缓冲文件系统是指系统自动地在内存区为程序中每一个正在使用的文件开辟一个文件缓冲区。从内存向磁盘输出数据必须先送到内存中的缓冲区，装满缓冲区后才一起送到磁盘去。如果从磁盘向计算机读入数据，则一次从磁盘文件将一批数据输入到内存缓冲区（充满缓冲区），然后再从缓冲区逐个地将数据送到程序数据区（给程序变量）。这样做是为了节省存取时间，提高效率，缓冲区的大小由各个具体的C编译系统确定。



说明：每一个文件在内存中只有一个缓冲区，在向文件输出数据时，它就作为输出缓冲区，在从文件输入数据时，它就作为输入缓冲区。

文件类型指针

缓冲文件系统中，关键的概念是“**文件类型指针**”，简称“**文件指针**”。每个被使用的文件都在内存中开辟一个相应的文件信息区，用来存放文件的有关信息（如文件的名字、文件状态及文件当前位置等）。这些信息是保存在一个结构体变量中的。该结构体类型是由系统声明的，取名为**FILE**。

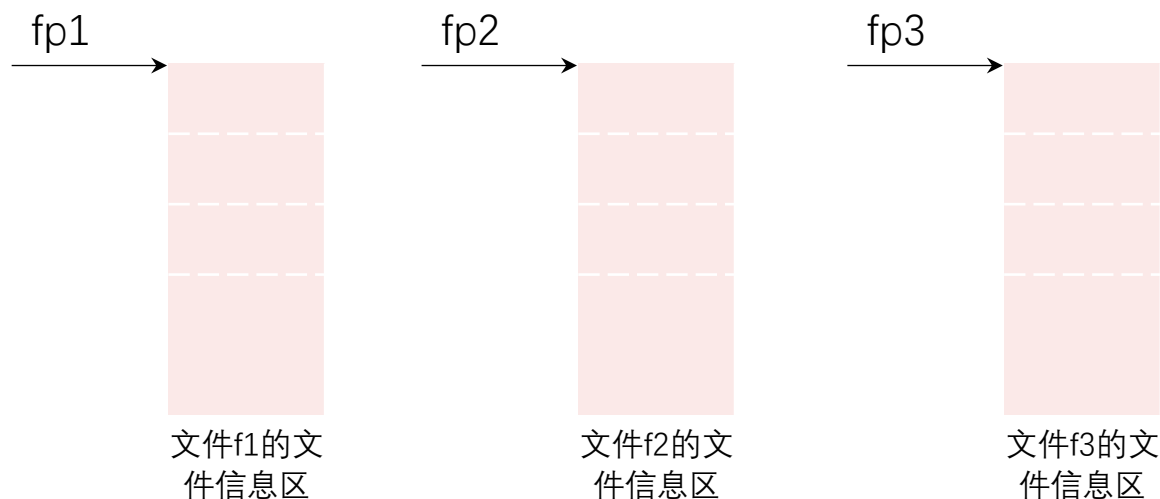
```
typedef struct
{   short level;           //缓冲区“满”或“空”的程度
    unsigned flags;       //文件状态标志
    char fd;              //文件描述符
    unsigned char hold;    //如缓冲区无内容不读取字符
    short bsize;          //缓冲区的大小
    unsigned char*buffer;  //数据缓冲区的位置
    unsigned char*curp;    //文件位置标记指针当前的指向
    unsigned istemp;       //临时文件指示器
    short token;           //用于有效性检查
}FILE;
```

一种C编译环境提供的stdio.h头文件中有以下的文件类型声明

文件类型指针

```
FILE *fp;  
//定义一个指向FILE类型数据的指针变量
```

可以使fp指向某一个文件的文件信息区(是一个结构体变量), 通过该文件信息区中的信息就能够访问该文件。也就是说, **通过文件指针变量能够找到与它关联的文件**。如果有n个文件, 应设n个指针变量, 分别指向n个FILE类型变量, 以实现对n个文件的访问。为方便起见, 通常将这种指向文件信息区的指针变量简称为**指向文件的指针变量**。



注意

- 指向文件的指针变量并不是指向外部介质上的数据文件的开头, 而是指向内存中的文件信息区的开头。

打开与关闭文件

打开与关闭文件

对文件读写之前应该“打开”该文件，在使用结束之后应“关闭”该文件。

所谓“打开”是指为文件建立相应的信息区(用来存放有关文件的信息)和文件缓冲区(用来暂时存放输入输出的数据)。

在编写程序时，在打开文件的同时，一般都指定一个指针变量指向该文件，也就是建立起指针变量与文件之间的联系，这样，就可以通过该指针变量对文件进行读写了。

所谓“关闭”是指撤销文件信息区和文件缓冲区，使文件指针变量不再指向该文件，显然就无法进行对文件的读写了。

用fopen函数打开数据文件

fopen(文件名, 使用文件方式);

```
FILE*fp;           //定义一个指向文件的指针变量fp  
fp=fopen("a1","r"); //将fopen函数的返回值赋给指针变量fp
```

表示以“读入”方式打开名字为a1的文件

在打开一个文件时，通知编译系统以下3个信息：

- ① 需要打开文件的名称，也就是准备访问的文件的名称
- ② 使用文件的方式（“读”还是“写”等）
- ③ 让哪一个指针变量指向被打开的文件

用fopen函数打开数据文件

fopen(文件名，使用文件方式)；

使用文件方式

文件使用方式	含义	如果指定的文件不存在
“r” （只读）	为了输入数据，打开一个已存在的文本文件	出错
“w” （只写）	为了输出数据，打开一个文本文件	建立新文件
“a” （追加）	向文本文件尾添加数据	出错
“rb” （只读）	为了输入数据，打开一个二进制文件	出错
“wb” （只写）	为了输出数据，打开一个二进制文件	建立新文件
“ab” （追加）	向二进制文件尾添加数据	出错
“r+” （读写）	为了读和写，打开一个文本文件	出错
“w+” （读写）	为了读和写，建立一个新的文本文件	建立新文件
“a+” （读写）	为了读和写，打开一个文本文件	出错
“rb+” （读写）	为了读和写，打开一个二进制文件	出错
“wb+” （读写）	为了读和写，建立一个新的二进制文件	建立新文件
“ab+” （读写）	为读写打开一个二进制文件	出错

用fopen函数打开数据文件

fopen(文件名, 使用文件方式);

-
- (1) 用 “r” 方式打开的文件只能用于向计算机输入而不能用作向该文件输出数据，而且该文件应该已经存在，并存有数据，这样程序才能从文件中读数据。不能用 “r” 方式打开一个并不存在的文件，否则出错。
 - (2) 用 “w” 方式打开的文件只能用于向该文件写数据（即输出文件），而不能用来向计算机输入。如果原来不存在该文件，则在打开文件前新建一个以指定的名字命名的文件。如果原来已存在一个以该文件名命名的文件，则在打开文件前先将该文件删去，然后重新建立一个新文件。
 - (3) 如果希望向文件末尾添加新的数据（不希望删除原有数据），则应该用 “a” 方式打开。但此时应保证该文件已存在；否则将得到出错信息。在每个数据文件中自动设置了一个隐式的“**文件读写位置标记**”，它指向的位置就是当前进行读写的位置。如果“文件读写位置标记”在文件开头，则下一次的读写就是文件开头的的数据。然后“文件读写位置标记”自动移到下一个读写位置，以便读写下一个数据。以添加方式打开文件时，文件读写位置标记移到文件末尾。
 - (4) 用 “r+” “w+” “a+” 方式打开的文件既可用于输入数据，也可用于输出数据。
-

用fopen函数打开数据文件

fopen(文件名, 使用文件方式);

(5) 如果不能实现“打开”的任务，fopen函数将会带回一个空指针值NULL。
(6) C标准建议用表10.1列出的文件使用方式打开文本文件或二进制文件，但目前使用的有些C编译系统可能不完全提供所有这些功能，需要注意所用系统的规定。

```
if ((fp=fopen("file1","r"))==NULL)
{
    printf("cannot open this file\n");
    exit(0);
}
```

打开一个文件的常用方法

(7) 有12种文件使用方式，其中有6种是在第一个字母后面加了字母b的(如rb,wb,ab,rb+,wb+,ab+)，b表示二进制方式。其实，带b和不带b只有一个区别，即对换行的处理。由于在C语言用一个'\n'即可实现换行，而在Windows系统中为实现换行必须要用“回车”和“换行”两个字符，即'\r'和'\n'。因此，如果使用的是文本文件并且用“w”方式打开，在向文件输出时，遇到换行符'\n'时，系统就把它转换为'\r'和'\n'两个字符，否则在Windows系统中查看文件时，各行连成一片，无法阅读。同样，如果有文本文件且用“r”方式打开，从文件读入时，遇到'\r'和'\n'两个连续的字符，就把它转换为'\n'一个字符。如果使用的是二进制文件，在向文件读写时，不需要这种转换。加b表示使用的是二进制文件，系统就不进行转换。

用fopen函数打开数据文件

fopen(文件名, 使用文件方式);

(8) 如果用“wb”的文件使用方式，并不意味着在文件输出时把内存中按ASCII形式保存的数据自动转换成二进制形式存储。输出的数据形式是由程序中采用什么读写语句决定的。例如，用fscanf和fprintf函数是按ASCII方式进行输入输出，而fread和fwrite函数是按二进制进行输入输出。

(9) 程序中可以使用3个标准的流文件——标准输入流、标准输出流和标准出错输出流。系统已对这3个文件指定了与终端的对应关系。标准输入流是从终端的输入，标准输出流是向终端的输出，标准出错输出流是当程序出错时将出错信息发送到终端。程序开始运行时系统自动打开这3个标准流文件。

用fclose函数关闭数据文件

```
fclose(文件指针) ; ;
```

```
fclose(fp);
```

在使用完一个文件后应该关闭它，以防止它再被误用。“关闭”就是撤销文件信息区和文件缓冲区，使文件指针变量不再指向该文件，也就是文件指针变量与文件“脱钩”，此后不能再通过该指针对原来与其相联系的文件进行读写操作，除非再次打开，使该指针变量重新指向该文件。

如果不关闭文件就结束程序运行将会丢失数据。因为，在向文件写数据时，是先将数据输出到缓冲区，待缓冲区充满后才正式输出给文件。如果当数据未充满缓冲区时程序结束运行，就有可能使缓冲区中的数据丢失。用fclose函数关闭文件时，先把缓冲区中的数据输出到磁盘文件，然后才撤销文件信息区。有的编译系统在程序结束前会自动先将缓冲区中的数据写到文件，从而避免了这个问题，但还是应当养成在程序终止之前关闭所有文件的习惯。

fclose函数也带回一个值，当成功地执行了关闭操作，则返回值为0；否则返回EOF(-1)。

顺序读写数据文件

怎样向文件读写字符

读写一个字符的函数：

函数名	调用形式	功能	返回值
fgetc	fgetc(fp)	从fp指向的文件读入一个字符	读成功，带回所读的字符，失败则返回文件结束标志EOF(即-1)
fputc	fputc(ch,fp)	把字符ch写到文件指针变量fp所指向的文件中	输出成功，返回值就是输出的字符；输出失败，则返回EOF（即-1）

fgetc的第1个字母f代表文件(file)，中间的get表示“获取”，最后一个字母c表示字符(character)，fgetc的含义很清楚：从文件读取一个字符。fputc也类似。

怎样向文件读写字符

【例10.1】从键盘输入一些字符，并逐个把它们送到磁盘上去，直到用户输入一个“#”为止。

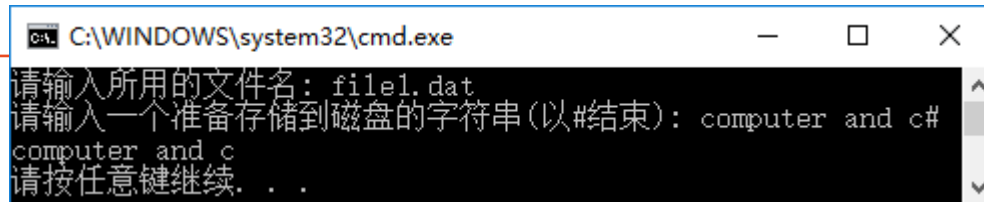


用来存储数据的文件名可以在fopen函数中直接写成字符串常量形式，也可以在程序运行时由用户临时指定。

用fopen函数打开一个“只写”的文件（“w”表示只能写入不能从中读数据），若成功，函数返回该文件所建立的信息区的起始地址给文件指针变量fp。若失败，则显示“无法打开此文件”，用exit函数终止程序运行，此函数在stdlib.h头文件中。

用getchar函数接收用户从键盘输入的字符。注意每次只能接收一个字符。

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    FILE *fp; //定义文件指针fp
    char ch,filename[10];
    printf("请输入所用的文件名:");
    scanf("%s",filename); //输入文件名
    getchar(); //用来消化最后输入的回车符
    if((fp=fopen(filename,"w"))==NULL) //打开输出文件并使fp指向此文件
    {
        printf("cannot open file\n"); //如果打开出错就输出“打不开”
        exit(0); //终止程序
    }
    printf("请输入一个准备存储到磁盘的字符串(以#结束):");
    ch=getchar(); //接收从键盘输入的第一个字符
    while(ch!='#') //当输入'#'时结束循环
    {
        fputc(ch,fp); //向磁盘文件输出一个字符
        putchar(ch); //将输出的字符显示在屏幕上
        ch=getchar(); //再接收从键盘输入的一个字符
    }
    fclose(fp); //关闭文件
    putchar(10); //向屏幕输出一个换行符
    return 0;
}
```



怎样向文件读写字符

【例10.2】将一个磁盘文件中的信息复制到另一个磁盘文件中。今要求将上例建立的file1.dat文件中的内容复制到另一个磁盘文件file2.dat中。



在访问磁盘文件时，是逐个字符(字节)进行的，为了知道当前访问到第几个字节，系统用“文件读写位置标记”来表示当前所访问的位置。开始时“文件读写位置标记”指向第1个字节，每访问完一个字节后，当前读写位置就指向下一个字节，即当前读写位置自动后移。

为了知道对文件的读写是否完成，只须看文件读写位置是否移到文件的末尾。

```
#include <stdio.h>
#include <stdlib.h>
int main()
```

```
{    FILE *in,*out;
    char ch,infile[10],outfile[10];
    printf("输入读入文件的名字:");
    scanf("%s",infile);
    printf("输入输出文件的名字:");
    scanf("%s",outfile);
    if((in=fopen(infile,"r"))==NULL)
    {    printf("无法打开此文件\n"); exit(0);    }
    if((out=fopen(outfile,"w"))==NULL)
    {    printf("无法打开此文件\n"); exit(0);    }
    ch=fgetc(in);
    while(!feof(in))
    {    fputc(ch,out);
        putchar(ch);
        ch=fgetc(in);
    }
    putchar(10);
    fclose(in);
    fclose(out);
    return 0;
}
```

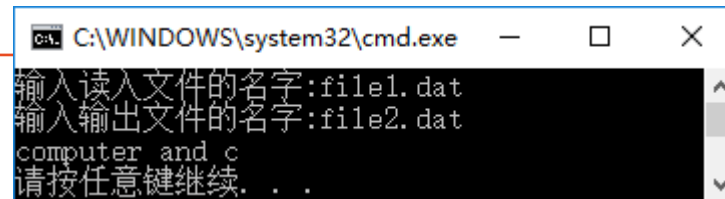
```
C:\WINDOWS\system32\cmd.exe
输入读入文件的名字:file1.dat
输入输出文件的名字:file2.dat
computer and c
请按任意键继续. . .

//定义指向FILE类型文件的指针变量
//定义两个字符数组，分别存放两个数据文件名

//输入一个输入文件的名字

//输入一个输出文件的名字
//打开输入文件
//打开输出文件
//从输入文件读入一个字符，赋给变量ch
//如果未遇到输入文件的结束标志
//将ch写到输出文件
//将ch显示到屏幕上
//再从输入文件读入一个字符，赋给变量ch

//显示完全部字符后换行
//关闭输入文件
//关闭输出文件
```



怎样向文件读写一个字符串

读写一个字符的函数：

函数名	调用形式	功能	返回值
fgets	fgets(str,n,fp)	从fp指向的文件读入一个长度为(n-1)的字符串， 存放到字符串数组str中	读成功， 返回地址str， 失败则返回NULL
fputs	fputs(str,fp)	把str所指向的字符串写到文件指针变量fp所指向的文件中	输出成功， 返回0； 否则返回非0值

fgets中最后一个字母s表示字符串(string)。见名知义， fgets的含义是： 从文件读取一个字符串。

怎样向文件读写一个字符串

fgets函数的函数原型为

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

其作用是从文件读入一个字符串。调用时可以写成下面的形式:

```
fgets(str,n,fp);
```

其中,n是要求得到的字符个数,但实际上只从fp所指向的文件中读入n-1个字符,然后在最后加一个'\0'字符,这样得到的字符串共有n个字符,把它们放到字符数组str中。如果在读完n-1个字符之前遇到换行符“\n”或文件结束符EOF,读入即结束,但将所遇到的换行符“\n”也作为一个字符读入。若执行fgets函数成功,则返回值为str数组首元素的地址,如果一开始就遇到文件尾或读数据出错,则返回NULL。

怎样向文件读写一个字符串

fputs函数的函数原型为

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

其作用是将str所指向的字符串输出到fp所指向的文件中。调用时可以写成:

```
fputs("China",fp);
```

把字符串"China"输出到fp指向的文件中。fputs函数中第一个参数可以是字符串常量、字符数组名或字符型指针。字符串末尾的'\0'不输出。若输出成功，函数值为0;失败时，函数值为EOF(即-1)。

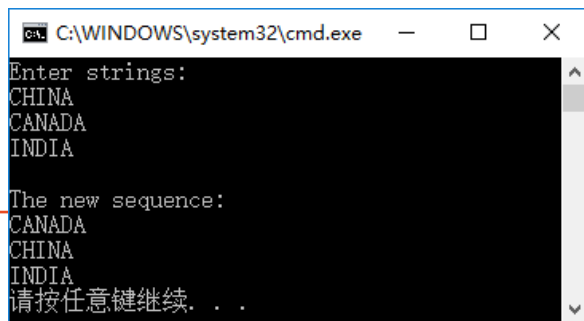
fgets和fputs这两个函数的功能类似于gets和puts函数，只是gets和puts以终端为读写对象，而fgets和fputs函数以指定的文件作为读写对象。

怎样向文件读写一个字符串

【例10.3】从键盘读入若干个字符串，对它们按字母大小的顺序排序，然后把排好序的字符串送到磁盘文件中保存。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{   FILE*fp;
    char str[3][10],temp[10];
    //str是用来存放字符串的二维数组，temp是临时数组
    int i,j,k,n=3;
    printf("Enter strings:\n");    //提示输入字符串
    for(i=0;i<n;i++)
        gets(str[i]);            //输入字符串
    for(i=0;i<n-1;i++)            //用选择法对字符串排序
    {   k=i;
        for(j=i+1;j<n;j++)
            if(strcmp(str[k],str[j])>0) k=j;
        if(k!=i)
        {   strcpy(temp,str[i]);
```

```
            strcpy(str[i],str[k]);
            strcpy(str[k],temp);}
    if((fp=fopen("D:\\CC\\string.dat","w"))==NULL) //打开磁盘文件
    //'\为转义字符的标志，因此在字符串中要表示'\'用'\\'
    {
        printf("can't open file!\n");
        exit(0);
    }
    printf("\nThe new sequence:\n");
    for(i=0;i<n;i++)
    {   fputs(str[i],fp);fputs("\n",fp);
        //向磁盘文件写一个字符串，然后输出一个换行符
        printf("%s\n",str[i]);    //在屏幕上显示
    }
    return 0;
}
```



```
C:\WINDOWS\system32\cmd.exe
Enter strings:
CHINA
CANADA
INDIA
The new sequence:
CANADA
CHINA
INDIA
请按任意键继续. . .
```



【例10.3】从键盘读入若干个字符串，对它们按字母大小的顺序排序，然后把排好序的字符串送到磁盘文件中保存。

可以编写出以下的程序，从文件string.dat中读回字符串，并在屏幕上显示。

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    FILE*fp;
    char str[3][10];
    int i=0;
    if((fp=fopen("D:\\CC\\string.dat","r"))==NULL) //注意文件路径必须与前相同
    {
        printf("can't open file!\n");
        exit(0);
    }
    while(fgets(str[i],10,fp)!=NULL)
    {
        printf("%s",str[i]);
        i++;
    }
    fclose(fp);
    return 0;
}
```

用格式化的方式读写文本文件

可以对文件进行格式化输入输出，这时就要用fprintf函数和fscanf函数，从函数名可以看到，它们只是在printf和scanf的前面加了一个字母f。它们的作用与printf函数和scanf函数相仿，都是格式化读写函数。只有一点不同：fprintf和fscanf函数的读写对象不是终端而是文件。它们的一般调用方式为：

```
fprintf(文件指针, 格式字符串, 输出表列);
```

```
fscanf(文件指针, 格式字符串, 输出表列);
```

```
fprintf (fp,"%d,%6.2f",i,f);    //将int型变量i和float型变量f的值按%d和%6.2f的格式输出到fp指向的文件中
```

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

```
//磁盘文件上如果有字符 “3,4.5” ， 则从中读取整数3送给整型变量i， 读取实数4.5送给float型变量f
```

用二进制方式向文件读写一组数据

C语言允许用fread函数从文件中读一个数据块，用fwrite函数向文件写一个数据块。在读写时是以二进制形式进行的。在向磁盘写数据时，直接将内存中一组数据原封不动、不加转换地复制到磁盘文件上，在读入时也是将磁盘文件中若干字节的内容一批读入内存。

```
fread(buffer, size, count, fp);
```

```
fwrite(buffer, size, count, fp);
```

buffer : 是一个地址。对fread，它是用来存放从文件读入的数据的存储区的地址。对fwrite，是要把此地址开始的存储区中的数据向文件输出（以上指的是起始地址）。

size : 要读写的字节数。

count : 要读写多少个数据项(每个数据项长度为size)。

fp : FILE类型指针。

```
float f[10];
```

```
fread(f,4,10,fp); //从fp所指向的文件读入10个4个字节的数据，存储到数组f中
```

怎样向文件读写一个字符串

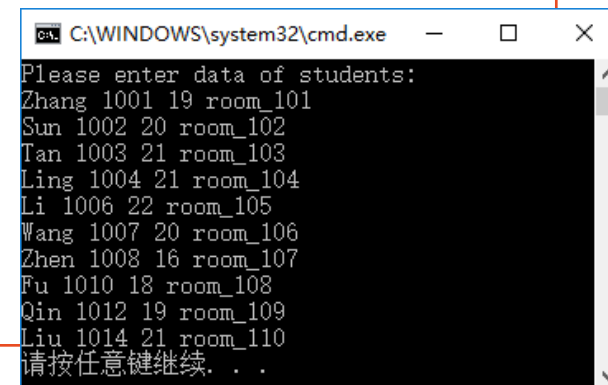
【例10.4】从键盘输入10个学生的有关数据，然后把它们转存到磁盘文件上去。

```
#include <stdio.h>
#define SIZE 10
struct Student_type
{   char name[10];
    int num;
    int age;
    char addr[15];
}stud[SIZE]; //定义全局结构体数组stud, 包含10个学生数据

void save() //定义函数save, 向文件输出SIZE个学生的数据
{   FILE *fp;
    int i;
    if((fp=fopen("stu.dat","wb"))==NULL) //打开输出文件stu.dat
    {   printf("cannot open file\n");
        return;
    }
    for(i=0;i<SIZE;i++)
```

```
        if(fwrite(&stud[i],sizeof(struct Student_type),1,fp)!=1)
            printf("file write error\n");
    fclose(fp);
}

int main()
{   int i;
    printf("Please enter data of students:\n");
    for(i=0;i<SIZE;i++)
        //输入SIZE个学生的数据, 存放在数组stud中
        scanf("%s%d%d%s",stud[i].name,&stud[i].num,
            &stud[i].age,stud[i].addr);
    save();
    return 0;
}
```



```
C:\WINDOWS\system32\cmd.exe
Please enter data of students:
Zhang 1001 19 room_101
Sun 1002 20 room_102
Tan 1003 21 room_103
Ling 1004 21 room_104
Li 1006 22 room_105
Wang 1007 20 room_106
Zhen 1008 16 room_107
Fu 1010 18 room_108
Qin 1012 19 room_109
Liu 1014 21 room_110
请按任意键继续. . .
```



【例10.4】从键盘输入10个学生的有关数据，然后把它们转存到磁盘文件上去。

为了验证在磁盘文件stu.dat中是否已存在此数据，可以用以下程序从stu.dat文件中读入数据，然后在屏幕上输出。

```
C:\WINDOWS\system32\cmd.exe
Zhang    1001    19 room_101
Sun      1002    20 room_102
Tan      1003    21 room_103
Ling     1004    21 room_104
Li       1006    22 room_105
Wang     1007    20 room_106
Zhen     1008    16 room_107
Fu       1010    18 room_108
Qin      1012    19 room_109
Liu      1014    21 room_110
请按任意键继续. . .
```

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 10
struct Student_type
{
    char name[10];
    int num;
    int age;
    char addr[15];
}stud[SIZE];

int main()
{
    int i;
    FILE *fp;
    if((fp=fopen("stu.dat","rb"))==NULL) //打开输入文件stu.dat
    {
        printf("cannot open file\n");
        exit(0);
    }
    for(i=0;i<SIZE;i++)
    {
        fread(&stud[i],sizeof(struct Student_type),1,fp); //从fp指向的文件读入一组数据
        printf("%-10s %4d %4d %-15s\n",stud[i].name,stud[i].num,stud[i].age,stud[i].addr);
        //在屏幕上输出这组数据
    }
    fclose(fp); //关闭文件stu_list
    return 0;
}
```



【例10.4】从键盘输入10个学生的有关数据，然后把它们转存到磁盘文件上去。

从磁盘文件stu_list中读二进制数据，并存放在stud数组中。

```
##include <stdio.h>
#define SIZE 10
struct Student_type
{   char name[10];
    int num;
    int age;
    char addr[15];
}stud[SIZE]; //定义全局结构体数组stud，包含10个学生数据
void load()
{   FILE *fp;
    int i;
    if((fp=fopen("stu_list","rb"))==NULL) //打开输入文件stu_list
    {   printf("cannot open infile\n");
        return;
    }
    for(i=0;i<SIZE;i++)
        if(fread(&stud[i],sizeof(struct Student_type),1,fp)!=1)
            //从stu_list文件中读数据
            {   if(feof(fp))
                    {   fclose(fp);
                        return;
                    }
            }
```

```
        printf("file read error\n");
    }
    fclose(fp);
}
//定义函数save，向文件输出SIZE个学生的数据
void save()
{   FILE *fp;
    int i;
    if((fp=fopen("stu.dat","wb"))==NULL) //打开输出文件stu.dat
    {   printf("cannot open file\n");
        return;
    }
    for(i=0;i<SIZE;i++)
        if(fwrite(&stud[i],sizeof(struct Student_type),1,fp)!=1)
            printf("file write error\n");
    fclose(fp);
}
int main()
{   int i;
    load();
    save();
    return 0;
}
```

怎样向文件读写一个字符串

(1) 数据的存储方式

- 文本方式: 数据以字符方式(ASCII代码)存储到文件中。如整数12, 送到文件时占2个字节, 而不是4个字节。以文本方式保存的数据便于阅读。
- 二进制方式: 数据按在内存的存储状态原封不动地复制到文件。如整数12, 送到文件时和在内存中一样占4个字节。

(2) 文件的分类

- 文本文件(ASCII文件): 文件中全部为ASCII字符。
- 二进制文件: 按二进制方式把在内存中的数据复制到文件的, 称为二进制文件, 即映像文件。

(3) 文件的打开方式

- 文本方式: 不带b的方式, 读写文件时对换行符进行转换。
- 二进制方式: 带b的方式, 读写文件时对换行符不进行转换。

(4) 文件读写函数

- 文本读写函数: 用来向文本文件读写字符数据的函数 (如fgetc, fgets,fputc,fputs,fscanf,fprintf等) 。
 - 二进制读写函数: 用来向二进制文件读写二进制数据的函数 (如getw,putw,fread,fwrite等) 。
-

随机读写数据文件

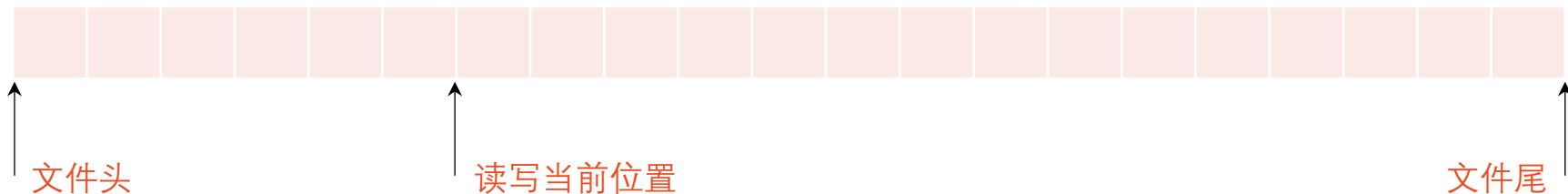
对文件进行顺序读写比较容易理解，也容易操作，但有时效率不高。而随机访问不是按数据在文件中的物理位置次序进行读写，而是可以对任何位置上的数据进行访问，显然这种方法比顺序访问效率高得多。

文件位置标记及其定位

1. 文件位置标记

为了对读写进行控制，系统为每个文件设置了一个**文件读写位置标记**(简称文件位置标记或文件标记)，用来指示“接下来要读写的下一个字符的位置”。

一般情况下，在对字符文件进行顺序读写时，文件位置标记指向文件开头，这时如果对文件进行读/写的操作，就读/写完第1个字符后，文件位置标记顺序向后移一个位置，在下一次执行读/写操作时，就将位置标记指向的第2个字符进行读出或写入。依此类推，直到遇文件尾，此时文件位置标记在最后一个数据之后。



对流式文件既可以进行顺序读写，也可以进行随机读写。关键在于控制文件的位置标记。如果文件位置标记是按字节位置顺序移动的，就是顺序读写。如果能将文件位置标记按需要移动到任意位置，就可以实现随机读写。所谓随机读写，是指读写完上一个字符（字节）后，并不一定要读写其后续的字符（字节），而可以读写文件中任意位置上所需要的字符（字节）。即对文件读写数据的顺序和数据在文件中的物理顺序一般是不一致的。可以在任何位置写入数据，在任何位置读取数据。

文件位置标记及其定位

2. 文件位置标记的定位

(1) 用rewind函数使文件位置标记指向文件开头

rewind(文件指针);

rewind函数的作用是使文件位置标记重新返回文件的开头，此函数没有返回值。

(2) 用fseek函数改变文件位置标记

fseek(文件类型指针, 位移量, 起始点);

“起始点”：用0, 1或2代替，0代表“文件开始位置”，1为“当前位置”，2为“文件末尾位置”

“位移量”：指以“起始点”为基点，向前移动的字节数（长整型）

fseek函数一般用于二进制文件。

```
fseek (fp,100L,0);           //将文件位置标记向前移到离文件开头100个字节处
fseek (fp,50L,1);           //将文件位置标记向前移到离当前位置50个字节处
fseek (fp,-10L,2);          //将文件位置标记从文件末尾处向后退10个字节
```

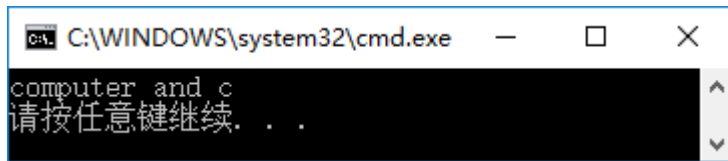
(3) 用ftell函数测定文件位置标记的当前位置

ftell函数的作用是得到流式文件中文件位置标记的当前位置，用相对于文件开头的位移量来表示。如果调用函数时出错（如不存在fp指向的文件），ftell函数返回值为-1L。

```
i=ftell(fp);                //变量i存放文件当前位置
if(i== -1L) printf("error\n"); //如果调用函数时出错，输出"error"
```

文件位置标记及其定位

【例10.5】有一个磁盘文件，内有一些信息。要求第1次将它的内容显示在屏幕上，第2次把它复制到另一文件上。

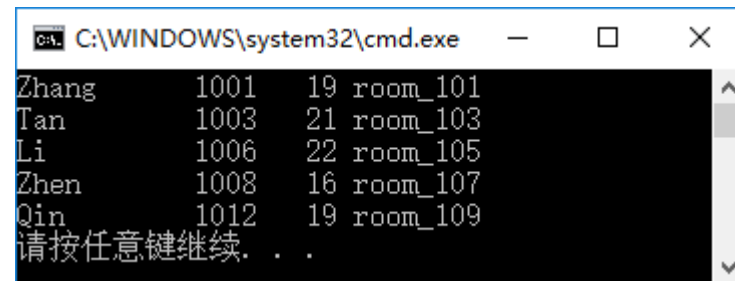


```
#include<stdio.h>
int main()
{   char ch;
    FILE *fp1,*fp2;
    fp1=fopen("file1.dat","r");    //打开输入文件
    fp2=fopen("file2.dat","w");    //打开输出文件
    ch=getc(fp1);                  //从file1.dat文件读入第一个字符
    while(!feof(fp1))              //当未读取文件尾标志
    {   putchar(ch);                //在屏幕输出一个字符
        ch=getc(fp1);              //再从file1.dat文件读入一个字符
    }
    putchar(10);                   //在屏幕执行换行
    rewind(fp1);                   //使文件位置标记返回文件开头
    ch=getc(fp1);                  //从file1.dat文件读入第一个字符
    while(!feof(fp1))              //当未读取文件尾标志
    {   fputc(ch,fp2);              //向file2.dat文件输出一个字符
        ch=getc(fp1);              //再从file1.dat文件读入一个字符
    }
    fclose(fp1);fclose(fp2);
    return 0;
}
```

随机读写

【例10.6】在磁盘文件上存有10个学生的数据。要求将第1,3,5,7,9个学生数据输入计算机，并在屏幕上显示出来。

```
#include<stdio.h>
#include <stdlib.h>
struct Student_type //学生数据类型
{
    char name[10];
    int num;
    int age;
    char addr[15];
}stud[10];
int main()
{
    int i;
    FILE *fp;
    if((fp=fopen("stu.dat","rb"))==NULL) //以只读方式打开二进制文件
    {
        printf("can not open file\n");
        exit(0);
    }
    for(i=0;i<10;i+=2)
    {
        fseek(fp,i*sizeof(struct Student_type),0); //移动文件位置标记
        fread(&stud[i],sizeof(struct Student_type),1,fp); //读一个数据块到结构体变量
        printf("%-10s %4d %4d %-15s\n",stud[i].name,stud[i].num,stud[i].age,stud[i].addr);
        //在屏幕输出
    }
    fclose(fp);
    return 0;
}
```



```
C:\WINDOWS\system32\cmd.exe
Zhang      1001      19 room_101
Tan        1003      21 room_103
Li         1006      22 room_105
Zhen       1008      16 room_107
Qin        1012      19 room_109
请按任意键继续...
```

文件读写的出错检测

文件读写的出错检测

1. `ferror`函数 `ferror(fp);`

在调用各种输入输出函数（如`putc`,`getc`,`fread`,`fwrite`等）时，如果出现错误，除了函数返回值有所反映外，还可以用`ferror`函数检查。

如果`ferror`返回值为0（假），表示未出错；
如果返回一个非零值，表示出错。

注意

对同一个文件每一次调用输入输出函数，都会产生一个新的`ferror`函数值，因此，应当在调用一个输入输出函数后立即检查`ferror`函数的值，否则信息会丢失。

在执行`fopen`函数时，`ferror`函数的初始值自动置为0。

2. `clearerr`函数

`clearerr`的作用是使文件出错标志和文件结束标志置为0。

假设在调用一个输入输出函数时出现错误，`ferror`函数值为一个非零值。应该立即调用`clearerr(fp)`，使`ferror(fp)`的值变成0，以便再进行下一次的检测。

只要出现文件读写出错标志，它就一直保留，直到对同一文件调用`clearerr`函数或`rewind`函数，或任何其他一个输入输出函数。