第八章 文件

【教学目的】

掌握文件的基本操作，正确使用文件操作函数。

【教学内容】

文件的概念；
ASCII 码文件和二进制文件的差别；
文件数据类型和文件指针的概念；
文件读写方式的概念；
基本的文件操作函数；
文件数据块读写的概念及操作方法。

【教学重点和难点】

文件指针的使用方法及文件的读写。
文件数据块读写。

【问题的提出】

在前面的程序设计中，我们介绍了输入和输出，即从标准输入设备—键盘输入，由标准输出设备—显示器输出。不仅如此，我们也常把磁盘作为信息载体，用于保存中间结果或最终数据，会利用打开一个文件来将磁盘的信息输入到内存，通过关闭一个文件来实现将内存数据输出到磁盘。这时的输入和输出是针对文件系统，故文件系统也是输入和输出的对象。

按文件系统中数据组织形式分：文件可分为字符文件和二进制文件。字符文件通常又称为 ASCII 码文件或文本文件，按字符存储，具有可读性；而二进制文件是以二进制存储，不具备可读性，但从存储空间的利用来看，二进制文件存储数据比用文本格式节省存储空间，并且读写效率高。

【教学要点】

1. 按字符方式读写函数 写一个字符到磁盘文件 函数格式：fputc(ch,fp)

【例 8.1】从键盘输入一行字符，写入到文本文件 string.txt 中。

```
#include <stdio.h>
main()
{
   FILE *fp;
   char ch;
   if((fp=fopen("string.txt","w"))==NULL)   /*以方式打开文 string.txt */
   {
      printf("can't open file\n");exit(1);
   }
```

```
    do          /* 不断从键盘读字符并写入文件，直到遇到换行符 */
    {
      ch=getchar();                          /* 从键盘读取字符 */
      fputc(ch,fp);                          /* 将字符写入文件 */
    }while(ch!='\n');              /*当从键盘输入回车时，输入结束 */
    fclose(fp);                              /* 关闭文件 */
}
```

说明：

1. fopen 函数用于打开文件, 其调用格式为:
   FILE *fp
   fp=fopen(文件名，使用方式);
   本例中 fopen 函数返回指向文件 string.txt 的文件指针，然后赋值给 fp。
2. 文件使用完毕后必须关闭，以避免数据丢失。
   格式：fclose(文件指针);
3. 函数 fputc(ch,fp)
   功能：将字符 ch（可以是字符表达式，字符常量、变量等）写入 fp 所指向的文件。
   返回：输出成功返回值-输出的字符 ch；输出失败返回 EOF。
   注意：每次写入一个字符，文件位置指针自动指向下一个字节。

2. 按字符方式读写函数 写一个字符到磁盘文件　函数格式：ch=fgetc(fp)

【例 8.2】将磁盘上一个文本文件的内容复制到另一个文件中。

```
#include <stdio.h>
main()
{
  FILE *fp_in,*fp_out;
  char infile[20],outfile[20];
  printf("Enter the infile   name:");
  scanf("%s",infile);                    /* 输入欲拷贝源文件的文件名*/

printf("Enter the outfile name:");
  scanf("%s",outfile);                   /* 输入欲拷贝目标文件的文件名*/
  if((fp_in=fopen(infile,"r"))==NULL)
  {                                      /*打开源文件*/
    printf("can't open file:%s",infile);   exit(1);
  }
  if((fp_out=fopen(outfile,"w"))==NULL)
  {                                      /*打开目标文件*/
    printf("can't open file:%s",outfile);   exit(1);
  }
  while(!feof(fp_in))                    /* 若源文件未结束 */
  {
    fputc(fgetc(fp_in),fp_out);   /* 从源文件读一个字符，写入目标文件 */
  }
  fclose(fp_in);
  fclose(fp_out);                        /* 关闭源、目标文件 */
```

```
}
```
结果：

Enter the infile name:string.txt

Enter the outfile name:string1.txt

说明：

1. 本例使用了 feof()判断文件结束。feof()函数检测文件指针是否到达了文件结尾，若是则返回一个非 0 值，否则返回 0。
   feof 函数既适合文本文件，也适合二进制文件文件结束的判断。
2. 函数 fgetc(ch,fp)
   功能：从 fp 所指向的文件读一个字符，字符由函数返回。返回的字符可以赋值给 ch，也可以直接参与表达式运算。
   返回：输入成功返回值-输入的字符；遇到文件结束返回 EOF。
   注意：每次读入一个字符，文件位置指针自动指向下一个字节。

3. 字符串读写函数　从磁盘文件读一个字符串　函数格式：char *fgets(char *str,int n,FILE *fp)

【例 8.3】编制一个将文本文件中全部信息显示到屏幕的程序(类似于 dos 的 type 命令)。使用 fgets 函数实现。

```
#include <stdio.h>
main(int argc,char *argv[])
{
  FILE *fp;
  char string[81]; /* 最多保存 80 个字符，外加一个字符串结束标志 */
  if(argc!=2||(fp=fopen(argv[1],"r"))==NULL)          /* 打开文件 */
  {
    printf("can't open file"); exit(1);
  }
  while(fgets(string,81,fp)!=NULL)
/* 如果未读到文件末尾(EOF),函数不会返回 NULL,继续循环（执行循环体） */
          /* 从文件一次读 80 个字符，遇换行或 EOF，提前带回字符串 */
    printf("%s",string);                                  /* 打印串 */
    fclose(fp);                                           /* 关闭文件 */
}
```

说明：

函数 char *fgets(char *str,int n,FILE *fp)

功能：从 fp 所指向的文件读 n-1 个字符，并将这些字符放到以 str 为起始地址的单元中。如果在读入 n-1 个字符结束前遇到换行符或 EOF，读入结束。字符串读入后最后加一个‘\0’字符。

返回：输入成功返回值-输入串的首地址；遇到文件结束或出错返回 NULL。

4. 字符串读写函数　写一个字符串到磁盘文件　函数格式：fputs(char *str,FILE *fp)

【例 8.4】在文本文件 string.txt 末尾添加若干行字符。使用 fputs 函数实现。

```
#include <stdio.h>
main()
{
   FILE *fp;
   char s[81];
   if((fp=fopen("string.txt","a"))==NULL)          /*打开文件*/
   {
      printf("can't open file\n"); exit(1);
   }
   while(strlen(gets(s))>0)
      /*从键盘读入一个字符串，遇到空行（strlen=0）结束*/
   {
      fputs(s,fp);                          /*  将字符串写进文件  */
      fputs("\n",fp);                             /*  补一个换行符  */
   }
   fclose(fp);                                    /*  关闭文件  */
}
```

说明：

函数 fputs(char *str,FILE *fp)

功能：向 fp 所指向的文件写入以 str 为首地址的字符串。

返回：输入成功返回值 0；出错返回非 0 值。

5. 格式化读写函数　函数 fprintf(fp,格式字符串,输出表列)

fscanf(fp,格式字符串,输入表列)

【例 8.5】将一些格式化的数据写入文本文件，再从该文件中以格式化方法读出显示到屏幕上，其格式化数据是两个学生记录，包括姓名、学号、两科成绩。

```
#include <stdio.h>
main()
{
FILE *fp;
int i;
struct stu{                                        /*定义结构体类型*/
char name[15];
char num[6];
float score[2];
}student;                                          /*说明结构体变量*/
if ((fp=fopen("test1.txt","w"))==NULL)
{                                          /*以文本只写方式打开文件*/
printf("cannot open file");
exit(0);
}
printf("input data:\n");
for( i=0;i<2;i++)
```

```
{
scanf("%s %s %f %f",student.name,student.num,&student.score[0],
&student.score[1]);                                  /*从键盘输入*/
fprintf(fp,"%s %s %7.2f %7.2f\n",student.name,student.num,
student.score[0],student.score[1]);                  /*写入文件*/
}
fclose(fp);                                          /*关闭文件*/
if ((fp=fopen("test1.txt","r"))==NULL)
{ /*以文本只读方式重新打开文件*/
printf("cannot open file");
exit(0);
}
printf("output from file:\n");
while (fscanf(fp,"%s %s %f %f ",student.name,student.num,
&student.score[0],&student.score[1])!=EOF）      /*从文件读入*/
printf("%s %s %7.2f %7.2f\n",student.name,student.num,
student.score[0],student.score[1]);                  /*显示到屏幕*/
fclose(fp);                                          /*关闭文件*/
}
```

说明：

程序设计一个文件变量指针，两次以不同方式打开同一文件，写入和读出格式化数据，有一点很重要，那就是用什么格式写入文件，就一定用什么格式从文件读，否则，读出的数据与格式控制符不一致，就造成数据出错。

格式化文件读写函数 **fprintf,fscanf** 与函数 **printf,scanf** 作用基本相同，区别在于 **fprintf,fscanf** 读写的对象是磁盘文件，**printf,scanf** 读写的对象是终端。

格式：**fprintf(fp,格式字符串,输出表列);**

fscanf(fp,格式字符串,输入表列);

其中：**fp** 是文件指针。

6. 数据块读写函数

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

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

【例 8.6.】从键盘输入一批学生的数据，然后把它们存到磁盘文件 **stud.dat** 中

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
struct student
{
   int num;
   char name[20];
   char sex;
   int age;
```

```
    float score;
};                                          /* 共 5 个成员，占用 29 bytes */
main()
{
    struct student stud;
    char numstr[20],ch;
/* numstr-临时字符串，保存学号/年龄/成绩，然后转换为相应类型; ch-Y/N */
    FILE *fp;
    if((fp=fopen("stud.dat","wb"))==NULL) /*以二进制、写方式打开文件*/
    {
        printf("can't open file stud.dat\n");
        exit(1);
    }

    do
    {
        printf("enter number:"); gets(numstr); stud.num=atoi(numstr);
        printf("enter name:"); gets(stud.name);
        printf("enter sex:"); stud.sex=getchar(); getchar();
        printf("enter age:"); gets(numstr); stud.age=atoi(numstr);
printf("enter score:"); gets(numstr); stud.score=atof(numstr);
/* 每次将一个准备好的结构体变量的所有内容写入文件（写一个记录） */
fwrite(&stud,sizeof(struct student),1,fp);
        printf("have another student record(y/n)?");
ch=getchar(); getchar();
    }while(toupper(ch)=='Y');   /* 循环 读数据/写记录*/
    fclose(fp);   /* 关闭文件 */
}
```

键盘输入数据，保存在结构体变量中

空读（有读取动作，但不保存读取的值-‘\n’）

说明：

从文件（特别是二进制文件）读写一块数据（如一个数组元素，一个结构体变量的数据-记录）使用数据块读写函数非常方便。

数据块读写函数的调用形式为：

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

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

其中：**buffer** 是指针，对 **fread** 用于存放读入数据的首地址；对 **fwrite** 是要输出数据的首地址。

size 是一个数据块的字节数(每块大小)，**count** 是要读写的数据块块数。

fp 文件指针

fread、**fwrite** 返回 读取/写入 的数据块块数。（正常情况=**count**）

以数据块方式读写，文件通常以二进制方式打开。

7. 文件的定位 rewind-重返文件头函数

【例 8.7】有一个文本文件，第一次使它显示在屏幕上，第二次把它复制到另外一个文件中。

```
#include <stdio.h>
```

```
main()
{
   FILE *fp1,*fp2;
   fp1=fopen("string.txt","r");   /* 打开文件 */
   fp2=fopen("string2.txt","w");
   /* 从文件 string.txt 读出，写向屏幕 */
   while(!feof(fp1))putchar(getc(fp1));
   /* 重返文件头 */
   rewind(fp1);
   /* 从文件 string.txt 读出，写向文件 string2.txt */
   while(!feof(fp1))putc(getc(fp1),fp2);
   fclose(fp1);   fclose(fp2);   /* 关闭文件 */
}
```

说明:

rewind-重返文件头函数

功能：使文件位置指针重返文件的开头。该函数无返回值

8. 文件的定位 fseek-位置指针移动函数

【例 8.8】编程读出文件 stu.dat 中第三个学生的数据。

```
#include <stdio.h>
struct student
{
   int num;
   char name[20];
   char sex;
   int age;
   float score;
};
main()
{
   struct student stud;
   FILE *fp;
   int i=2;
   if((fp=fopen("stud.dat","rb"))==NULL)
   {
      printf("can't open file stud.dat\n");
      exit(1);
   }
fseek(fp,i*sizeof(struct student),SEEK_SET);        /*定位第 3 个记录*/
   if(fread(&stud,sizeof(struct student),1,fp)==1)       /*将 1 记录读出*/
   {
printf("%d,%s,%c,%d,%f\n",stud.num,stud.name,stud.sex,
stud.age,stud.score);                                  /* 打印此记录 */
   }
   else
```

```
        printf("record 3 does not presented.\n");
    fclose(fp);
}
```

说明：

移动文件读写位置指针，以便文件的随机读写。

格式：**fseek(FILE *fp,long offset,int whence)**；

参数：

fp-文件指针.

whence-计算起始点（计算基准）.计算基准可以是下面符号常量：

符号常量	符号常量的值	含义
SEEK_SET	**0**	从文件开头计算
SEEK_CUR	**1**	从文件指针当前位置计算
SEEK_END	**2**	从文件末尾计算

offset-偏移量（单位：字节）.从计算起始点开始再偏移 **offset**，得到新的文件指针位置。**offset** 为正，向后偏移；**offset** 为负，向前偏移。

【小结】

在前面的章节中,我们了解了许多有关 C 语言的数据类型和结构的内容,本章则介绍另一种数据类型———文件类型。通过文件,可以将数据输送到外部介质如磁盘上永久地保存起来,或者将数据发送到其他设备上。回顾前述的程序设计过程不难发现,在我们编写的程序中,所有的数据或来源于用户输入,或被写在源程序代码中。对于前者,在数据量较大时,通过输入得到这些数据显然会给用户带来很大的负担。对于后者,在数据需要变化时,通常又需要修改程序代码。此外,在以往的程序中,程序运行后所产生的大量数据结果也不能保留下来。如果使用文件,上述问题将会得到很好的解决。