第10章 对文件的输入输出

10.3 顺序读写数据文件

- 在顺序写时，先写入的数据存放在文件中前面，后写入的数据存放在文件中后面
- 在顺序读时，先读文件中前面的数据，后读文件中后面的数据
- 对顺序读写来说，对文件读写数据的顺序和数据在文件中的物理顺序是一致的
- 顺序读写需要用库函数实现

10.3 顺序读写数据文件

10.3.1 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	写成功，返回值就是输出的字符；输出失败，则返回ＥＯＦ（即**-1**）
feof	**feof(fp)**	判断文件结尾	结尾返回**true(1)**

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

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   FILE *fp;
   char ch, filename[10];
   printf("Please input file name: ");
   gets(filename);
   if((fp=fopen(filename,"w"))==NULL) {
        printf("Open file error!\n");   exit(0);
   }
   while((ch=getchar( )) != '#'){
       fputc(ch, fp);
    }
    fclose(fp);
    return 0;
}
```

用exit函数时加

输入文件名

只写打开

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。

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   FILE *in, *out;
   char c;
   if((in=fopen("ac.txt", "r"))==NULL ||
   (out=fopen("ad.txt", "w"))==NULL){
        printf("Open file error!\n"); exit(0);
    } c = fgetc(in);
    while(!feof(in)){
       fputc(c, out);
       c = fgetc(in);
    }
   fclose(in);fclose(out);
   return 0;
}
```

10.3.2 怎样向文件读写一个字符串

- 读写一个字符串的函数

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

- 说明：

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：

  fgets(str,n,fp);

- ◆**fgets(str,n,fp);**中**n**是要求得到的字符个数，但实际上只读**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
- ◆执行**fgets**成功，返回**str**数组首地址，如果一开始就遇到文件尾或读数据错，返回**NULL**

- 说明：

fputs函数的函数原型为：

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

- ◆**str**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);**
- ◆**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
- ◆不自动换行
- ◆输出成功，函数值为0；失败，函数值为**EOF**

例**10.3** 从键盘读入若干人名，对它们按字母顺序排序，然后把排好序的人名送到磁盘文件中保存。

- 解题思路：为解决问题，可分为三个步骤：
  - ◆从键盘读入**n**个人名，存放在一个二维字符数组中，每一个一维数组存放一个人名；
  - ◆对字符数组中的**n**个字符串按字母顺序排序，排好序的字符串仍存放在字符数组中；
  - ◆将字符数组中的字符串顺序输出到文件。

```
void sort(char (*p)[], int n)
{
    int i, j;
    for(i=0; i<n-1; i++){
        for(j=0; j<n-1-i; j++){
                ……
        }
    }
}
```

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main()
{
   char names[5][20];
   int i, j;
   FILE *fp;
   if((fp=fopen("names.data", "w") )== NULL){
        printf("open file error!"); exit(0);
   }
   for(i=0; i<5; i++)
        gets(names[i]);
   sort(names, 5);
   for(i=0; i<5; i++){
        fputs(names[i], fp);
        fputc('\n', fp);
   }
   return 0;
}
```

显式地向文件中输出一个'\n'

- 思考：
  - ◆反过来，如何从文件**names.data**中读回字符串，显示并在屏幕上呢**?**

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   char name[20];
   FILE *fp;
   if((fp=fopen("names.data", "r") )== NULL){
        printf("open file error!"); exit(0);
   }
   fgets(name, 20, fp);
   while(!feof(fp)){
      printf("%s", name); //puts(name);
      fgets(name, 20, fp);
   }
   return 0;
}
```

10.3.3用格式化的方式读写文件

- 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**,**输入表列**);**

  ◆用法和**printf**、**scanf**类似，按照指定的格式，对文件指针所关联的文件进行格式化输入和输出。

  ◆**printf**、**scanf**是**fprintf**和**fscanf**的特例

  ◆**fprintf(stdout, 格式字符串, 输出表列);**

  ◆**fscanf(stdin, 格式字符串, 输入表列);**

10.3.4 用二进制方式向文件读写一组数据

- 函数原型:

```
int fread(void * _buff,          //用来存储读取的数据
          int  element_size, //一个数据的大小
          int count,             //读取的数据的个数
          FILE *fp               //读取的文件
); //从文件中读数据，返回成功读取的数据个数

int fwrite(const void * _buff,  //用来存储待写的数据
           int element_size,    //一个数据的大小
           int count,           //写入文件的数据个数
           FILE *fp             //待写的文件
); //向文件中写数据，返回成功写入的数据个数
```

- **buff**是一个地址，经常使用数组名(用来一次读写多个数据)，或数据元素(变量)的地址(用来一次读写**1**个数据)

例**10.4** 现在开始做一个班级的学生**(**包含信息：学号、姓名、性别、年龄、专业、班级**)**信息管理系统：前期功能要求如下：

(1)从键盘录入学生信息

(2)将学生信息保存到文件，已备后续使用。

- 解题思路：
  - ◆声明表示学生信息的结构体数据类型
  - ◆定义包含整个班级学生的结构体数组
  - ◆数据录入：用一个函数
  - ◆数据写入文件：用一个
    - ●在次函数中调用**fw**

```
typedef struct Student
{
   int num;
   char name[10];
   char sex;
   int age;
   char major[20];
   int class;
} STU;
```

```
#include <stdio.h>
#include <stdlib.h>
void input(STU *p, int n)
{
   int i;
   for(i=0; i<n; i++)
      scanf("%d %s %c %d %s %d", &p[i].num,
   p[i].name, &p[i].sex, &p[i].age, p[i].major,
   &p[i].class);
}
void save(const char *file, const STU *p, int n)
{
   FILE *fp; int i;
   if((fp=fopen(file, "wb")) == NULL){
     printf("Open file error!\n"); exit(0);
   }

   fwrite(p, sizeof(STU), n, fp);


   fclose(fp);
}
```

```
int main()
{
   STU s[5];
   input(s, 5);
   save("stu.data", s, 5);
   return 0;
}
```

如何验证**stu.data**文件中的数据呢？

例**10.4**续 在上例**10.4**的基础上，将文件中的数据读出来显示在屏幕上。

- 解题思路:
  - ◆怎么读？
    - ●使用**fread**函数实现
  - ◆读出来的数据放哪儿**?**
    - ●使用数组来存储
  - ◆如何输出这些数据？
    - ●定义一个**output**函数实现数据输出

```
#include <stdio.h>
#include <stdlib.h>
void output(STU *p, int n)
{
   int i;
   for(i=0; i<n; i++)
      printff("%4d%10s%4c%4d%20s%4d\n",
   p[i].num,  p[i].name, p[i].sex, p[i].age, p[i].major,
   p[i].class);
}
void load(const char *file, const STU *p, int n)
{
   FILE *fp; int i;
   if((fp=fopen(file, "rb")) == NULL){
     printf("Open file error!\n"); exit(0);
   }

   fread(p, sizeof(STU), n, fp);


   fclose(fp);
}
```

```
int main()
{
  STU s[5];
  load("stu.data", s, 5);
  output(s, 5);
  return 0;
}
```

如果不确定文件中存储的学生信息个数呢？

```
int load(const char *file, const STU *p)
{
   FILE *fp; int total=0;
   if((fp=fopen(file, "rb")) == NULL){
     printf("Open file error!\n"); exit(0);
   }
   while(!feof(fp)){
      if(1 == fread(&p[total], sizeof(int), 1, fp)))
          total++;
   }
   fclose(fp);
}
```

10.4 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
- 如，上例中，如果需要读取指定学号的学生信息或要修改该学生的信息，顺序读取需要读取每个数据，造成效率很低的后果
- 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

10.4 随机读写数据文件

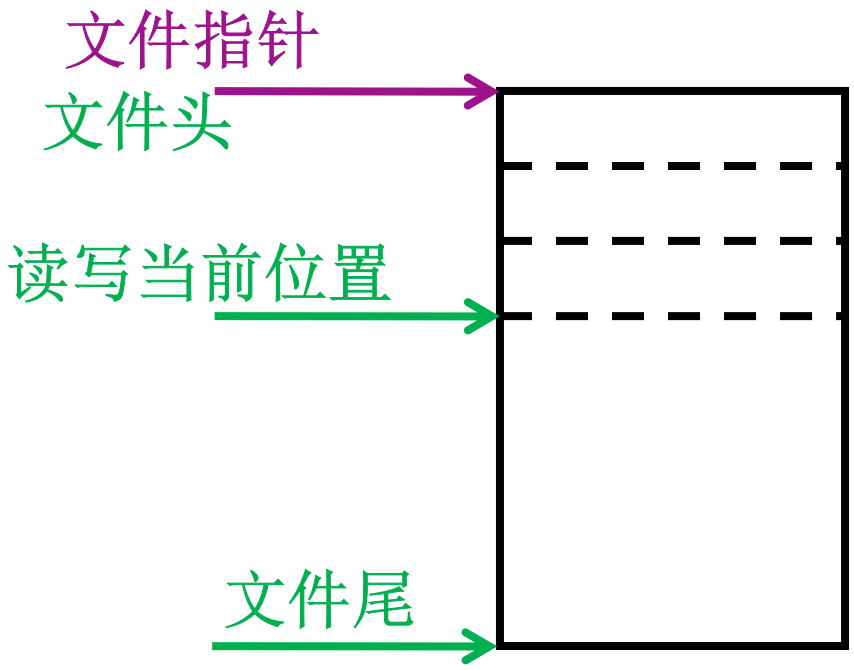
10.4.1 文件位置标记及其定位

10.4.2 随机读写

10.4.1 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(简称文件标记)，用来指示“接下来要读写的下一个字符的位置”



10.4.1 文件位置标记及其定位

1.文件位置标记

- 一般情况下，在对字符文件进行顺序读写时，文件标记指向文件开头，进行读的操作时，就读第一个字符，然后文件标记向后移一个位置，在下一次读操作时，就将位置标记指向的第二个字符读入。依此类推，直到遇文件尾，结束
- 如果是顺序写文件，则每写完一个数据后，文件标记顺序向后移一个位置，然后在下一次执行写操作时把数据写入指针所指的位置。直到把全部数据写完，此时文件位置标记在最后一个数据之后

10.4.1 文件位置标记及其定位

1.文件位置标记

- 可以根据读写的需要，人为地移动了文件标记的位置。文件标记可以向前移、向后移，移到文件头或文件尾，然后对该位置进行读写——随机读写
- 随机读写可以在任何位置写入数据，在任何位置读取数据

2. 文件位置标记的定位

◆可以强制使文件位置标记指向指定的位置

◆可以用以下函数实现：

(1)用**rewind**函数使文件标记指向文件开头

rewind函数的作用是使文件标记重新返回文件的开头，此函数没有返回值。

例**10.5** 磁盘有一文件，内有一些文本信息。要求读取该文件内容两遍，第一次将它的内容显示在屏幕上，第二次把它复制到另一文件中。

- 解题思路：
  - ◆第一遍读去文件内容，顺序读取，没有任何问题，文件读取结束后，文件标记已指到文件的末尾，如果再接着读数据，就遇到文件结束标志，**feof**函数的值等于**1(**真**)**，无法再读数据
  - ◆所以，第二遍重新读取文件内容时，必须在程序中使用**rewind**函数使位置指针返回文件的开头
  - ◆**rewind**函数原型
    - ●**void rewind(FILE *);**

```
#include<stdio.h>
int main()
{
  FILE *fp1,*fp2;
  fp1=fopen("file1.data","r");
  fp2=fopen("file2.data","w");
  while(!feof(fp1))
      putchar(fgetc(fp1));
  putchar('\n');
  rewind(fp1);
  while(!feof(fp1))
        fputc(fgetc(fp1),fp2);
  fclose(fp1);   fclose(fp2);
  return 0;
}
```

10.4.1 文件位置标记及其定位

(2) 用**fseek**函数改变文件标记

int fseek(FILE *stream, long offset, int whence);

操作成功返回**0**，否则返回非**0**

fseek函数的调用形式为：

fseek(文件类型指针**,**位移量**,**起始点**)**

- C 标准指定的文件位置名字

起始点	名 字	用数字代表
文件开始位置	**SEEK_SET**	**0**
文件当前位置	**SEEK_CUR**	**1**
文件末尾位置	**SEEK_END**	**2**

- **fseek**函数一般用于二进制文件。下面是**fseek**函数调用的几个例子：
  - **fseek (fp,100L,0);**
  - **fseek (fp,50L,1);**
  - **fseek (fp,-10L,2);**
- **fseek**函数的经典用法
  - **fseek(fp, sizeof(T)*n, 0/1/2);**
    - 当起始位置为**2**时，偏移量为负，表示向前移动

  (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。

  - **ftell**函数原型
    - **long int ftell(FILE *stream);**
  - 执行成功返回文件当前位置，否则返回**-1L**

10.4.2 随机读写

例**10.6** 例**10.4**的程序生成一磁盘文件，上存有**5**个学生的数据。要求在屏幕上显示第**1,3,5**个学生的信息。

- 解题思路：
  - ◆按二进制只读方式打开文件
  - ◆将文件位置标记指向文件的开头，读入一个学生的信息，并把它显示在屏幕上
  - ◆再将文件标记指向文件中第**1,3,5**个学生的数据区的开头，读入相应学生的信息，并把它显示在屏幕上
  - ◆关闭文件

```
#include<stdio.h>
#include <stdlib.h>
int main()
{
  STU s[5];
  int i;  FILE *fp;
  if((fp=fopen(”stu.data”, ”rb”))==NULL) {
     printf("can not open file\n"); exit(0); }
   for(i=0;i<5;i+=2){
     fseek(fp,i*sizeof(STU),0);
     fread(&s[i], sizeof(STU),1,fp);
     printff(“%4d%10s%4c%4d%20s%4d\n”,
           s[i].num,  s[i].name, s[i].sex, s[i].age,
           s[i].major, s[i].class);
  }
  fclose(fp);
  return 0;
}
```

10.5 文件读写的出错检测

1.ferror函数

- **ferror函数的一般调用形式为**

  ferror(fp);

  - ◆如果返回值为**0**，表示未出错，否则表示出错
  - ◆每次调用输入输出函数，都产生新的**ferror**函数值，因此调用输入输出函数后立即检查
  - ◆调用**fopen**时，**ferror**的初始值自动置为**0**

2. clearerr函数

- **clearerr函数的一般调用形式为**

  clearerr(fp);

  - ◆作用是使文件错误标志和文件结束标志置为**0**
  - ◆调用一个输入输出函数时出现错误（**ferror**值为非零值），立即调用**clearerr(fp)**，使**ferror(fp)**值变**0**，以便再进行下一次检测
  - ◆只要出现文件读写错误标志，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