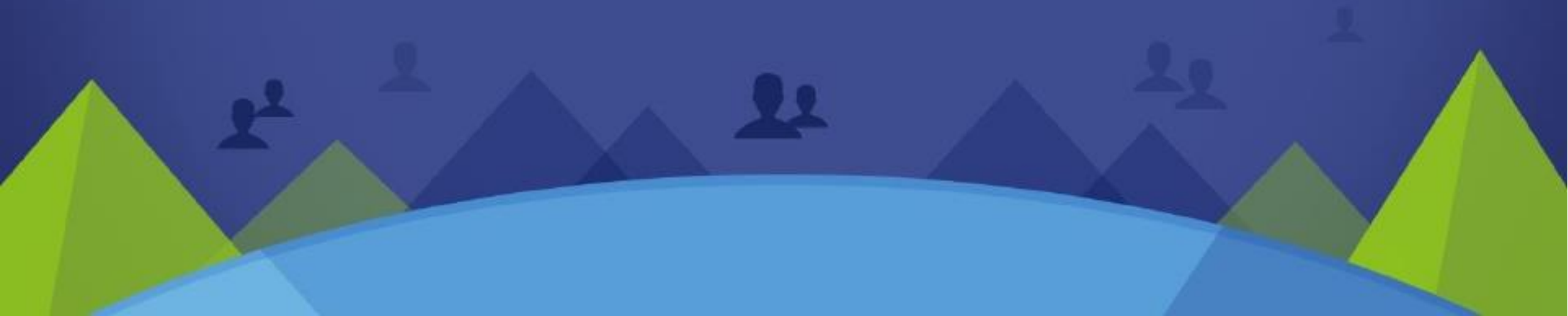


## 第9章

# 文件操作

## 《C语言程序设计新编教程》



# 能力要求

CAPACITY



理解C语言中文件的概念



掌握C语言的文件的库函数



掌握C语言文件的打开与关闭



掌握顺序和随机访问文件



# 内容导航

CONTENTS

## 文件操作

- 文件的基本概念
- 打开与关闭
- 常用文件读/写函数
- 文件的随机读/写
- 操作的错误处理
- 文件操作综合应用

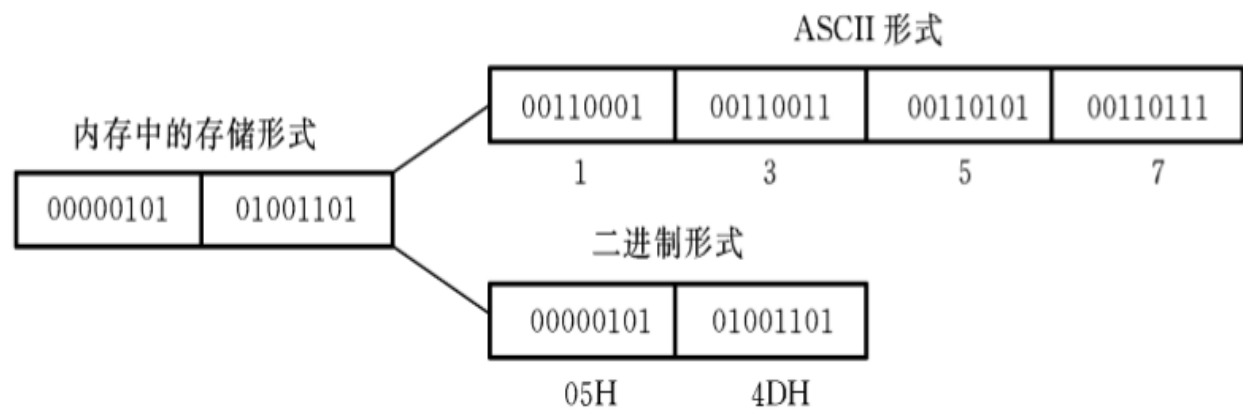
## 课程导入

文件对于今天的计算机系统至关重要。它们用于存储程序，文档，数据，通信，表单，图形和无数的其他信息。

❖ C语言操作文件时，使用C的标准 I/O 函数系统处理文件。本章主要介绍C语言中文件的概念，操作文件的库函数fopen(), getc(), putc(), fread(), fwrite()等；如何使用 C 的标准 I/O 函数系统处理文件；文件的打开与关闭；顺序和随机访问文件的功能等内容。



文件是其实是磁盘上一个命名的存储区。在C语言中文件可分为两类：**文本文件**和**二进制文件**。文本文件又称为ASCII文件，它的每一个字节放一个ASCII代码，代表一个字符。有一个整数1357，如果按二进制文件存放，则需2个字节；按文本文件形式存放，则需2个字节，如图所示。



## 文件类型指针

**FILE** 是关键字，

FILE \* 指针变量标识符；

例如：FILE \*fp;

- 1) 其中：FILE应为大写，它实际上是由系统定义的一个结构，在编写源程序时不必关心FILE结构的细节。
- 2) 表示fp是指向FILE结构的指针变量。通过fp即可先找到存放某个文件信息的结构变量，然后按结构变量提供的信息找到该文件，再实施对文件的操作。





# 内容导航

CONTENTS

## 文件操作

- 文件的基本概念
- 打开与关闭
- 常用文件读/写函数
- 文件的随机读/写
- 操作的错误处理
- 文件操作综合应用

## 文件打开函数 fopen( )

C程序使用fopen()打开文件，这一函数再stdio.h中声明。他的第一个参数是要打开的文件名；第二个参数是用于指定文件打开模式的一个字符串。C库提供了一些可能的模式，如下表

| 模式   | 含义   |
|------|--|
| "r"  | 打开一个文本文件，可以读取文件。   |
| "w"  | 打开一个文本文件，可以写入文件，先将文件的长度截为0，如果该文件不存在则先创建。                   |
| "a"  | 打开一个文本文件，可以写入文件，向已有文件尾部追加内容，如果该文件不存在则创建。                   |
| "r+" | 打开一个文本文件，可以进行更新，也即可以读取和写入文件。                               |
| "w+" | 打开一个文本文件，可以进行更新，如果该文件部存在则首先将其长度截为0，如果不存在则先创建               |
| "a+" | 打开一个文本文件，可以进行更新，向已有文件的尾部追加内容，如果不存在则先创建；可以读取整个文件，但写入时只能追加内容 |



## 文件打开函数 fopen( )

常用以下程序段打开文件：

```
if((fp=fopen("c:\myfile.dat","rb"))==NULL) //检查是否打开myfile.dat文件
{
    printf("\n error on open c:\myfile.dat \n ")
    exit(0)           //退出
}
```

在打开一个文件时，如果出错，fopen将返回一个空指针NULL。

在程序中可以用这一信息来判别是否完成打开文件的工作，并作相应的处理。

## 文件关闭函数 fclose( )

文件一旦使用完毕，就应该用关闭文件函数把文件关闭，以避免发生文件的数据丢失等 错误。  
函数fclose调用的一般形式是

```
fclose (文件指针) ;  
例如： fclose(fp);
```

其中：fp是已有确定指向的文件指针。该函数在关闭前清除与文件有关的所有缓冲区，正常 完成关闭文件操作时，函数fclose返回值为0；如返回非零值，则表示有错误发生。一般来说，函数fopen和fclose是成对出现的。



# 内容导航

CONTENTS

## 文件操作

- 文件的基本概念
- 打开与关闭
- 常用文件读/写函数
- 文件的随机读/写
- 操作的错误处理
- 文件操作综合应用

## 字符读/写函数putc和getc

最简单的文件I/O函数是getc和putc。他们类似与getchar和putchar函数,一次处理一个字符。假设文件以w方式打开,文件指针为fp1,那么语句:

**putc(c,fp1);**

把字符变量c包含的字符写入fp1指向的文件中。同样,getc函数用于从以读取方式打开的文件中读取一个字符。例如,语句:

**c=getc(fp2);**

从文件指针fp2指向的文件中读取一个字符。

每次进行getc和putc操作后,文件指针就移动一个字符的位置。当到达文件末尾时,getc函数将返回文件末尾标记符EOF。因此,当遇到EOF标记符时,读取工作将停止。

## 字符读/写函数putc和getc

例9.1:

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

```
main()
```

```
{
```

```
    FILE *f1;
```

```
    char c;
```

```
    printf("Data INPUT\n\n");
```

```
    /*打开INPUT文件*/
```

```
    f1=fopen("INPUT","w");
```

```
    /*从键盘获取一个字符*/
```

```
    while((c=getchar())!='0')
```

```
        /*把一个字符写入INPUT文件
```

```
        */
```

```
        putc(c,f1);
```

```
        fclose(f1);
```

```
        printf("\nData Output\n\n");
```

```
        /*再次打开INPUT文件*/
```

```
        f1=fopen("INPUT","r");
```

```
        /*从文件读取一个字符*/
```

```
        while((c=getc(f1))!=EOF)
```

```
            printf("%c",c);
```

```
        fclose(f1);
```

```
        return 0;
```

```
}
```

## 字符读/写函数putc和getc

### 程序运行结果

```
Data INPUT
hello ,this is a getc and putc example 0
Data Output
hello ,this is a getc and putc example
```

### 程序说明:

首先定义文件指针f1,指向文件INPUT,通过while循环putc函数逐个地把字符写入文件INPUT。输入0,并回车表示数据输入结束。getc然后逐个读取文件内容,并显示再屏幕上。当getc遇到文件结束符EOF时,读取工作终止。

## 整数读/写函数getw和putw

---

类似于getc和putc函数，getw和putw是基于整数的函数，用于读取和写入整数值，当只处理整数数据时，这些函数很有用。getw和putw的一般形式如下：

```
putw(integer,fp);  
getw(fp);
```

## 整数读/写函数getw和putw

例9.2:

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

```
main()
```

```
{
    FILE *f1,*f2;
    int number,i;
    printf("input integer\n\n");
    f1=fopen("data.txt","w");
    //创建data.txt文件
    for(i=1;i<=30;i++)
    {
        scanf("%d",&number);
```

```
        if(number == -1) break;
        putw(number,f1);
        //写入data.txt文件
    }
    fclose(f1);
    f2=fopen("data.txt","r");
    printf("Contents of data file\n\n");
    while((number=getw(f2))!=EOF)
    //读取文件，并判断是否到文件尾
        printf("%d",number);
    return 0;
}
```



## 整数读/写函数getw和putw

---

### 程序运行结果

```
input integer
123123-1
Contents of data file
123123
```

### 程序说明:

首先通过fopen（）函数创建data.txt文件，从键盘获取整数值，并使用putw()函数写入，当输入-1时，写入终止，关闭文件。利用getw()函数读取文件内容后输出。

## 二进制读/写函数fread和fwrite

---

函数fread和函数fwrite是ANSI C文件系统提供的用于二进制方式读/写的函数。可用来读/写一组数据，如一个数组元素，一个结构变量的值等。

函数fread和fwrite都有返回值。函数fread返回读入的项数，如果出错或者达到文件的尾部，则返回值可能会小于n；函数fwrite返回写出的项数，如果出错，则该值将等于n。

例如：**fread (fa,4,5,fp) ;**

其含义是从fp所指的文件中，每次读4个字节送入fa所指向的内存空间中，连续读5次。

## 二进制读/写函数fread和fwrite

例9.3:

```
#include "stdio.h"
#include <stdlib.h>
void main()
{FILE *fp;
    char c='a',c1;int i=123,i1;
    long l=2004184001L,l1;
    double d=4.5678,d1;
    //检查是否以读/写方式打开或建立文本
    文件text1.txt
    if((fp=fopen("test1.txt","wt+"))==NULL)
    {printf("不能打开文件");
    exit(1); }
    //通过fwrite将几个变量所存放的数据写
    入文件
    fwrite(&c,sizeof(char),1,fp);
    fwrite(&i,sizeof(int),1,fp);
```

```
    fwrite(&l,sizeof(long),1,fp);
    fwrite(&d,sizeof(double),1,fp);
    //重新定位指针到文件首部
    rewind(fp);
    //通过函数fread将数据读出文件
    fread(&c1,sizeof(char),1,fp);
    fread(&i1,sizeof(int),1,fp);
    fread(&l1,sizeof(long),1,fp);
    fread(&d1,sizeof(double),1,fp);
    //输出
    printf("c1=%c\n",c1);
    printf("i1=%d\n",i1);
    printf("l1=%ld\n",l1);
    printf("d1=%f\n",d1);
    fclose(fp);
}
```

## 二进制读/写函数fread和fwrite

### 程序运行结果

```
c1=a  
i1=123  
11=2004184001  
d1=4.567800
```

### 程序说明:

定义四个不同类型的变量，通过fwrite(), fread()函数写入和读取文件。  
fread(&c1,sizeof(char),1,fp): c1为读入文件数据的内存存储地址，fp为指定要读取的文件。Sizeof(char),1表示读取一个char值。

## 二进制读/写函数fread和fwrite

### 例9.4

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
struct stu{
    char name[10];
    int num;
    int age;
    char addr[15] ;
};
main()
{FILE *fp;
struct stu boya[2],boyb[2],*pp,*qq;
    char ch;
    int i;
    pp=boya;
```

```
qq=boyb;
//以读/写方式打开二进制文件
if((fp=fopen("stu_list.dat","wb+"))==NULL)
{
    printf("不能打开文件，按任意键推出\n");
    getch();
    exit(1);
}
//输入两个学生数据
printf("\n input data\n");
for(i=0;i<2;i++,pp++)
    scanf("%s%d%d%s",pp->name,&pp->num,&pp->age,&pp->addr);
    //写数据到文件
    pp=boya;
    fwrite(pp,sizeof(struct stu),2,fp);
    fclose(fp);
```

## 二进制读/写函数fread和fwrite

```
//再次以只读形式打开文件
if((fp=fopen("stu_list.dat","rb"))==NULL)
{
    printf("不能打开文件，按任意键推出");
    getch();
    exit(1);
}
//把文件内部位置指针移到文件首，读出两个学生数据后，在屏幕上显示
rewind(fp);
fread(qq,sizeof(struct stu),2,fp);
printf("\n\nname\tnumber age addr\n");
for(i=0;i<2;i++,qq++)
{
    printf("%s\t%5d\t",qq->name,qq->num);
    printf("%7d\t%s\n",qq->age,qq->addr);
}
fclose(fp);
return 0;
}
```

## 二进制读/写函数fread和fwrite

### 程序运行结果

```
input data
frank 1001 16 suzhou
bob   1002 18 beijing

name      number age addr
frank     1001    16 suzhou
bob       1002    18 beijing
```

### 程序说明:

本程序定义了一个结构stu，两个指针分别指向boya和boyb。程序以读/写方式打开二进制”stu\_list.dat”，输入两个学生数据后，写入文件中，然后把文件内部指针移动到文件首，读出两个学生数据后，在屏幕上显示。

## 格式化读/写函数fscanf和fprintf

---

函数fscanf和fprintf与前面使用的函数scanf和printf的功能相似，都是格式化读/写函数。二者的区别在于函数fscanf和fprintf的读/写对象不是键盘和显示器，而是磁盘文件。



## 格式化读/写函数fscanf和fprintf

```
#include<stdio.h>
#include<process.h>
main()
{
    FILE *fp;
    int i=88;
    char filename[30];/*定义一个字符型数组*/
    printf("please input filename:\n");
    scanf("%s",filename);/*输入文件名*/
    fp=fopen(filename,"w+")判断文件是否打开失败*/
    fprintf(fp,"%c",i);/*将88以字符形式写入fp所指的磁盘文件中*/
    fclose(fp);
}
```

### 程序运行结果

```
please input filename:
1.txt
```

### 程序说明:

使用fopen以w+方式打开文件，如文件不存在，则创建，然后用fprintf(fp,"%c",i)把整型值88，以字符形式写入文件。

## 二进制读/写函数fread和fwrite

### 例9.6

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 40
int main (void)
{
    FILE *fp;
    char words[MAX];

    if ((fp = fopen ("words", "a+")) ==
    NULL)
    {
        fprintf (stdout, "Can't open \"words\"
file.\n");
        exit(1);
    }
```

```
    puts ("Enter words to add to the file; press the Enter");
    while (gets(words) != NULL && words[0] != '\0')
        fprintf (fp, "%s", words); //把内容输出到文件
    puts ("file contents :");
    rewind(fp);
    while (fscanf (fp, "%s", words) == 1) //从文件获取内容
        puts (words);
    if (fclose(fp) != 0)
        fprintf (stderr, "Error closing file\n");
    return 0;
}
```

## 二进制读/写函数fread和fwrite

程序运行结果:输入” hello”

```
Enter words to add to the file; press the Enter  
hello  
  
file contents :  
hello
```

程序说明:

通过fprintf以%s的格式向文件words.txt已追加的方式添加内容，fscanf读取文件内容。while (gets(words) != NULL && words[0] != '\0')如果你键入一个空行，程序终止循环。

## fgets( )和fputs函数

---

fgets用来从文件中读入字符串。

fgets()的用法如下： fgets(buf,MAX,fp);

这里，buf 是一个 char 数组的名称，MAX 是字符串的最大长度，fp 是一个 FILE 指针。

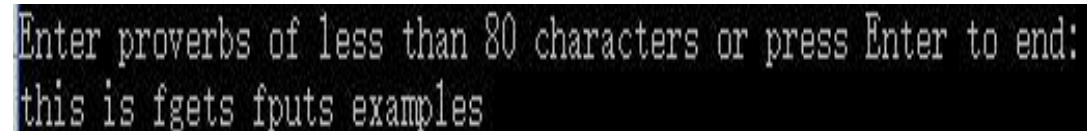
fputs()函数接受两个参数，它们依次是一个字符串的地址和一个文件指针。它把字符串地址指针所指的字符串写入指定文件。

## fgets( )和fputs函数

### 例9.7

```
#include <stdio.h>
const int LENGTH = 80;
int main(void)
{ char more[LENGTH];
  FILE *pfile = NULL;
  char *filename = "d:\\myfile.txt";
  pfile = fopen(filename, "a+");
  printf("Enter proverbs of less than 80
characters or press Enter to end:\n");
  fgets(more, LENGTH, stdin);      /* 从
键盘读取字符串到数组          */
  fputs(more, pfile);              /* 写入文件
*/
  fclose(pfile);
  return 0;
}
```

### 程序运行结果



```
Enter proverbs of less than 80 characters or press Enter to end:
this is fgets fputs examples
```

通过键盘输入内容后，在d的盘myfile.txt文件中就有相应的内容



```
myfile.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
this is fgets fputs examples
```

### 程序说明：

通过fgets(more, LENGTH, stdin) 其中stdin表示从键盘输入，存在数组中，使用fputs写入相应的文件。



# 内容导航

CONTENTS

## 文件操作

- 文件的基本概念
- 打开与关闭
- 常用文件读/写函数
- 文件的随机读/写
- 操作的错误处理
- 文件操作综合应用

## 文件定位函数rewind()和fseek()

实现随机读/写的关键是按要求移动位置指针，这称为文件的定位。文件定位时，移动文件内部位置指针的函数主要有两个，即函数rewind和fseek。前面已多次使用过函数rewind，其调用形式为

**rewind**（文件指针）；

它的功能是把文件内部的位置指针移到文件首。下面主要介绍函数fseek。函数fseek用来移动文件内部位置指针，其调用形式为

**fseek**（文件指针，位移量，起始点）；

例如： **fseek**（fp, 100L, 0）；

其含义是把位置指针移到离文件首100个字节处。

## 文件定位函数rewind()和fseek()

例9.8

```
#include <stdio.h>
#include <stdlib.h>
#define ARSIZE 1000

int main ()
{
    double numbers[ARSIZE];
    double value;
    const char * file = "numbers.dat";
    int i;
    long pos;
    FILE *iofile;
    /* 创建一组 double 类型的值 */
    for (i = 0; i < ARSIZE; i++)
        numbers[i] = 100.0 * i + 1.0 / (i + 1);
    /* 尝试打开文件 */
```

```
if ((iofile = fopen (file, "wb")) == NULL)
{
    fprintf (stderr, "Could not open %s
for output \n", file);
    exit(1);
}
/* 将数组中的数据以二进制写入文件 */
fwrite (numbers, sizeof (double),
ARSIZE, iofile);
fclose(iofile);
if ((iofile = fopen (file, "rb")) == NULL)
{
    fprintf (stderr, "Coule not open %s for
random access \n",file);
    exit(1);
}
```



## 文件定位函数rewind()和fseek()

```
/* 仅文件中读取所限项目 */
printf ("Enter an index in the range 0-
%d\n", ARSIZE - 1);
scanf ("%d", &i);
while (i >= 0 && i < ARSIZE)
{
    pos = (long) i * sizeof(double); // 计
    算偏移量
    fseek (iofile, pos, SEEK_SET);    //
    在文件中定位到那里
    fread (&value, sizeof(double), 1 ,
    iofile);
    printf ("The value there is %f \n",
    value);
```

```
printf ("Next index (out of range to quit):
\n", scanf ("%d", &i));
    }
    fclose (iofile);
    puts ("Bye !");
    return 0;
}
```

## 文件定位函数rewind()和fseek()

### 程序运行结果

```
Enter an index in the range 0-999
80
The value there is 8000.012346
0
Next index (out of range to quit):
The value there is 1.000000
-9
Next index (out of range to quit):
Bye !
```

### 程序说明:

程序首先创建了一个数组，然后在其中存放一些值。接着它以二进制模式创建了一个名为number.dat的文件，接着使用fwrite()把数组的内容复制到文件中，每个double值的64位模式从内存复制到文件中，不能通过文本编辑器来读取结果的二进制文件。程序的第二部分为了读取打开文件，请求用户输入一个值的索引。通过索引和double值占用的字节数相乘就可以得到文件中的位置，通过fseek()定位到该位置，利用fread()读取该位置的数据值。最后显示value的值。当输入不在0-999范围内时，程序结束。



# 内容导航

CONTENTS

## 文件操作

- 文件的基本概念
- 打开与关闭
- 常用文件读/写函数
- 文件的随机读/写
- 操作的错误处理
- 文件操作综合应用

在进行文件I/O操作时可能会发生错误。如果不能检查读写错误，当错误发生时，程序将不能正常运行。未检测出的错误可能导致程序的提前终止或不正确输出。幸运的是，C语言有两个状态查询库函数`feof`和`ferror`，可用来检测文件的I/O错误。

`feof`函数用来检测是否达到文件末尾。该函数以`FILE`指针为唯一参数，如果指定文件的所有数据都已读取，返回非零整数；否则返回零。如果`fp`为指向已打开用于读取数据的文件的指针，那么如果到达文件末尾，下面语句：

```
if(feof(fp))
    printf("End of data.\n");
```

`ferror`函数用于报告指定函数的状态。该函数也是以`FILE`指针为参数，如果检测出错误，就返回一个非零值；否则返回零。如果读取工作不成功，那么下面的语句：

```
if(ferror(fp)!=0)
    printf("An error has occurred.\n");
```

将显示一条错误信息。

我们知道，当使用`fopen`函数打开文件时，将返回一个文件指针。如果因为某些原因不能打开文件，那么函数返回`NULL`指针。这可以用来测试文件是否已打开。例如：

```
if(fp==NULL)
    printf("File could not be opened.\n");
```

### 例9.8

```
#include <stdio.h>
void main()
{
    char filename[10];
    FILE *fp1,*fp2;
    int i,number;
    fp1=fopen("TEST","w");
    //写方式打开文件,
    for(i=10;i<=100;i+=10)
        putw(i,fp1);
    //写入文件
    fclose(fp1);
    printf("\nInput filename\n");
    open_file:
    scanf("%s",filename);
```

```
if((fp2=fopen(filename,"r"))==NULL)
//如果文件不存在
{ printf("Cannot open the file.\n");
  printf("Type filename again.\n\n");
  goto open_file;
}
else
for(i=1;i<=20;i++)
{ number=getw(fp2);
  iffeof(fp2))
  //查看是否读取到文件末尾
  { printf("\nRan out of data.\n");
    break;}
  else
    printf("%d\n",number); }
fclose(fp2);
}
```

## 程序运行结果

```
Input filename
TEAT
Cannot open the file.
Type filename again.

TEST
10
20
30
40
50
60
70
80
90
100

Ran out of data.
```

## 程序说明：

当输入TEAT时，fopen函数返回NULL指针，因为文件TEAT不存在，因此将显示” Can’t open the file”。同样当所有数据都已读取时，函数调用feof(fp2)将返回一个非零整数，因此程序将显示 “Ran out of data”消息，并停止进一步的读取工作。



# 内容导航

CONTENTS

## 文件操作

- 文件的基本概念
- 打开与关闭
- 常用文件读/写函数
- 文件的随机读/写
- 操作的错误处理
- 文件操作综合应用

### 案例一

---

**【例9.10】** 使用学习到的文件处理函数，创建家庭成员数据的文件，输入成员数据（本人输入名字、出生日期，父母输入名字），读取文件，并输出家庭成员信息，最后删除文件。使用结构体来表示家庭成员信息。



## 例9.10

```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
/* 全局变量 */
/* 结构体 */
struct
{ char *filename;          /* 物
理文件名      */
  FILE *pfile;             /* 文件
指针          */
} global = {"D:\\myfile.bin", NULL };
struct Date                /* 日期结
构题      */
{
  int day;
  int month;
```

```
int year;
};
typedef struct family
/* 家庭成员结构体 */
{
  struct Date dob;
  char name[20];
  char pa_name[20];
  char ma_name[20];
}Family; /* Function prototypes */
bool get_person(Family *pfamily);
/* 输入家庭成员函数，具体函数定义在main函数
后      */
void getname(char *name);
/* 获取名字函数，具体函数定义在main 函数
后      */
void show_person_data(void);
/*输出函数，具体函数定义在main函数后
*/
```

```
void get_parent_dob(Family *pfamily);  
/* 查找家庭成员函数，具体函数定义在  
main函数后 */
```

```
int main(void)  
{  
    Family member;  
    /* 结构体变量member */  
    global.pfile = fopen(global.filename,  
        "wb");  
    while(get_person(&member))  
        /* 输入家庭成员信息 */  
        fwrite(&member, sizeof member, 1,  
            global.pfile); /* 写入文件 */  
    fclose(global.pfile);  
    /*关闭文件*/  
    show_person_data();  
    /* 输出函数 */
```

```
if(remove(global.filename))  
    /* 删除文件 */  
    printf("\nUnable to delete %s.\n", global.filename);  
else  
    printf("\nDeleted %s OK.\n", global.filename);  
return 0;  
}  
/* 输入家庭成员函数 */  
bool get_person( Family *temp)  
{  
    static char more = '\0';  
    /* 输入回车表示结束 */  
    printf("\nDo you want to enter details of a%s  
person (Y or N)? ", more != '\0'? "nother " : "" );  
    scanf(" %c", &more);  
    if(tolower(more) == 'n')  
        return false;  
    printf("\nEnter the name of the person: ");  
    getname(temp->name);
```

```
printf("\nEnter %s's date of birth (day
month year); ", temp->name);
scanf(" %d %d %d", &temp->dob.day,
&temp->dob.month, &temp->dob.year);
printf("\nWho is %s's father? ", temp-
>name);
getname(temp->pa_name);
/* 获得爸爸的名字 */
printf("\nWho is %s's mother? ", temp-
>name);
getname(temp->ma_name);
/* 获得妈妈的名字 */
return true;
} /* 从键盘获取名字 */
void getname(char *name)
{
    fflush(stdin);
    /* 忽略空格 */
    fgets(name, 20, stdin);
```

```
int len = strlen(name);
if(name[len-1] == '\n')          /* 如果最
后时空行 */
    name[len-1] = '\0';
}

/* 输出数据函数 */
void show_person_data(void)
{
    Family member;               /* 结构体变量 */
    fpos_t current = 0;          /* 文件位置
*/
    /* 以二进制方式只读打开文件 */
    if(!(global.pfile = fopen(global.filename, "rb")))
    {
        printf("\nUnable to open %s for reading.\n",
global.filename);
        exit(1);
    }
```

```
/* 一个一个读数据 */
while(fread(&member, sizeof member, 1,
global.pfile))
{
    fgetpos(global.pfile, &current); /* 保
存当前位置 */
    printf("\n\n%s's father is %s, and mother
is %s.",
        member.name, member.pa_name,
member.ma_name);
    //get_parent_dob(&member); /*
获取父母数据 */
    fsetpos(global.pfile, &current); /* 设置
下一个数据位置, */
}
fclose(global.pfile); /* 关闭文
件 */
}
```

## 程序运行结果

```
Do you want to enter details of a person (Y or N)? Y
Enter the name of the person: cindy
Enter cindy's date of birth (day month year); 1 3 2010
Who is cindy's father? bob
Who is cindy's mother? kate
Do you want to enter details of another person (Y or N)? N

cindy's father is bob, and mother is kate.
Deleted D:\myfile.bin OK.
```

## 程序说明：

在#include语句后，有一些结构定义和函数原型声明。

在get\_person（）函数中一次获得一个人的输入信息，将得到的数据存储在member结构对象中。接收数据后，将每个结构写入文件中，当get\_person()函数返回0是，就停止输入处理。

While循环结构结束后，关闭输入文件，调用show\_person\_data()函数。在这个函数中要使用文件位置的获取和设定函数。最后，使用remove()函数从磁盘删除文件。

## 案例二

---

【例9.11】编写一个程序，从键盘读入姓名和电话号码，将它们写入文件。如果这个文件不存在，就写入一个新文件。如果文件已存在，就将它们写入该文件。最后读取显示所有数据。

### 例9.11

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <ctype.h>

#define FIRST_NAME_LENGTH 31
#define SECOND_NAME_LENGTH 51
#define NUMBER_LENGTH 21

/* 定义名字结构体 */
typedef struct Nname
{ char firstname[FIRST_NAME_LENGTH];
  char secondname[SECOND_NAME_LENGTH];
} Name;
```

```
/* 定义电话结构体*/
typedef struct PPhoneRecord
{
    Name name;
    char number[NUMBER_LENGTH];
} PhoneRecord;
/* 声明函数原型 */
PhoneRecord read_phonerecord();
/* 从键盘中读取电话信息的函数，具体函数定义在main()函数后 */
Name read_name();
/* 从键盘中读取名字的函数，具体函数定义在main()函数后 */
void list_records(char *filename);
/* 显示文件中的内容，具体函数定义在main()函数后 */
void show_record(PhoneRecord record);
/* 输出函数，具体函数定义在main()函数后*/
```

```
int main(void)
{
    FILE *pFile = NULL;           /*输出文件
指针      */
    char *filename = "d:\\records.bin"; /* 文件名
*/
    char answer = 'n';
    PhoneRecord record;
    bool file_empty = true;
    printf("Do you want to enter some phone
records(y or n)? ");
    scanf(" %c", &answer);
    if(tolower(answer) == 'y')
    {
        pFile = fopen(filename, "a+"); /* 打开/创
建 文件，并可追加 */
        do
        {
```

```
record = read_phonerecord();
/* 获取名字和电话 */
    fwrite(&record, sizeof record, 1, pFile);
    printf("Do you want to enter another(y or n)? ");
    scanf(" %c", &answer);
    }while(tolower(answer) == 'y');
    fclose(pFile);
/* 关闭文件 */
    printf("\nFile write complete.");
    }
    printf("\nDo you want to list the records in the
file(y or n)? ");
    scanf(" %c", &answer);
    if(tolower(answer) == 'y')
        list_records(filename);
return 0;
}
```



```
/*从键盘获取名字和电话，并创建结构体
PhoneRecord */
PhoneRecord read_phonerecord()
{
    PhoneRecord record;
    record.name = read_name();
    printf("Enter the number: ");
    scanf(" %[0123456789]", record.number); /* 读
取数字，包括空格 */
    return record;
}
/* 从键盘获取名字并存入结构体 */
Name read_name()
{
    Name name;
    printf("Enter a first name: ");
    scanf(" %s", &name.firstname);
    printf("Enter a second name: ");
```

```
    scanf(" %s", &name.secondname);
    return name;
}
/* 列出文件内容 */
void list_records(char *filename)
{
    FILE *pFile;
    PhoneRecord record;
    bool file_empty = true;
    /* 空文件标记 */
    pFile = fopen(filename, "r");
    for(;;)
    {
        fread(&record, sizeof record, 1, pFile);
        if(feof(pFile))
            break;
        file_empty = false;
        /* 能读取记录，所有空文件标记为false */
```

```
show_record(record);    /* 输出记录 */
}
fclose(pFile);          /* 关闭文件 */
/* 检查是否有记录 */
if(file_empty)
    printf("The file contains no records.\n");
else
    printf("\n");
}

/* 输出结构体，包括名字和电话 */
void show_record(PhoneRecord record)
{
    printf("\n%s %s  %s", record.name.firstname,
record.name.secondname, record.number);
}
```

### 程序运行结果

```
Do you want to enter some phone records(y or n)? y
Enter a first name: li
Enter a second name: lei
Enter the number: 1
Do you want to enter another(y or n)? y
Enter a first name: wang
Enter a second name: bing
Enter the number: 2
Do you want to enter another(y or n)? n

File write complete.
Do you want to list the records in the file(y or n)? y

li lei  1
wang bing  2
```

## 程序说明:

创建结构体PhoneRecord，用来存储名字和电话号码。其中  
PhoneRecord read\_phonerecord()函数用来从键盘中读取电话信息。  
Name read\_name()函数用来从键盘中读取并返回名字的函数。  
list\_records(char \*filename) 函数用来列出文件中的内容  
show\_record(PhoneRecord record) 为输出函数。

## 本章小结

C语言中文件的处理过程通常要经历“打开文件→文件的读 / 写→关闭文件”等3个步骤，C标准函数库为此都配备有相应的操作函数。按文件内的数据组织形式，可把文件分为文本流文件和二进制流文件，编写程序时应注意这两种流文件在完成上述3个存取操作步骤的不同之处。文件可按只读、只写、读 / 写、追加4种操作方式打开，同时还必须指定文件的类型是二进制文件还是文本文件。学习 C语言文件读 / 写函数，文件定位函数，重点需要掌握 `fopen`，`fclose`，`fread`，`fwrite`，`fscanf`，`fprint`，`rewind`，`fseek`，`fgetpos`和`fsetpos`等函数。

---

# THANKS

---

