

实验十 文件

韦璐 PB16000702

学习重点

文件的概念

文件类型指针

文件的打开和关闭 (fopen())函数和 fclose () 函数)

格式化的读写

数据块的读写

字符的读写

字符串的读写

Feof, rewind, fseek 函数的基本使用方法

一. 要点综述

1. 文件的概念

- (1) 一个 C 文件是一个字节或二进制流, 它把数据看作是一连串的字符序列 (字节), 而不考虑记录的界限, 即 C 文件并不是由记录组成的
- (2) C 语言中的文件, 按照数据存放的形式可分为两类: 文本文件和二进制文件:
 1. 文本文件由一个一个字符组成, 每个字符用一个 ASCII 码表示, 其特点是可以通过显示器显示, 占用内存比较大。
 2. 二进制文件是指以数据在内存中的存储形式原样输出存储到磁盘上的文件, 它按照数据值的二进制代码形式存放。如 123.45 在内存中以浮点数形式存储, 占 4 个字节, 若以二进制形式存储到磁盘, 则仍是 4 个字节按原来在内存中的存储形式一样。其特点是节省存储空间, 输入输出无需转换, 速度快, 但不能在显示器直接输出, 所以一般用于程序与程序之间或者程序与设备之间数据的传递。

(3) 对文件的操作要与各种外部设备发生联系, 因此对文件的输入输出 (读写), 所以一般用于程序与程序之间或者程序与设备之间数据的传递。

(4) 根据计算机的系统是否可以自动设置缓冲区, 可以分缓冲文件系统和非缓冲文件系统。1. 缓冲文件系统的特点: 对程序中的每一个文件都在内存中开辟一个缓冲区。从磁盘文件输出数据先送到输入缓冲区中, 然后再从缓冲区中依次将数据送给接收变量。在向此盘文件输出数据时, 先将程序数据区中变量或表达式送到输出文件缓冲区中, 待装满后才一起输出给磁盘文件。目的是减少对磁盘的读写次数, 即一次可以读入一批数据或输出一批数据。2. 非缓冲文件系统的特点: 不由系统自动设置所需缓冲区, 而由用户自己根据需要设置。

原来的系统用缓冲文件系统处理文本文件, 用非缓冲文件系统处理二进制。新的标准只建议使用缓冲系统, 并对缓冲文件系统的共那个进行了扩充, 使之既能用于处理字符代码文件, 也能处理二进制文件。

2. 文件类型的指针

在 C 程序中对文件的操作都是通过问及文件指针变量试下的。缓冲文件系统中关键的概念是文件指针。

- (1) 调用一个文件至少需要有一下的信息:
 - 文件当前读写的位置
 - 与该文件对应的内存缓冲区的地址
 - 缓冲区未被处理的字符数
 - 文件的操作方式

批注 [管1]: 9.5 分

为此 C 语言在头文件中定义了一个 FILE 文件结构类型。

- (2) 文件结构体类型的指针变量定义形式;

FILE *指针变量名

- (3) 文件指针变量的复制操作是通过打开文件函数 fopen 实现的

3.文件必须先打开后使用, 必须关闭

文件打开的方式;

FILE *fp;

```
If ((fp=fopen("文件名","文件使用方式"))==NULL) {
```

```
Printf("cannot open this file. \n");
```

```
Exit(0);
```

```
}
```

- (1) 文件名可以是包含文件标识符(盘符, 路径, 文件名和扩展名)的字符串常量, 字符数组名或指向字符串的字符指针。
- (2) 使用文件方式
- (3) 如果按指定使用文件方式失败, 则将返回一个空指针 NULL。此时一般应该结束程序的运行并返回系统, 查找原因进行出错处理。
- (4) 读文件需要先确认此文件是否已存在, 并将读写当前位置设定为文件的开头
- (5) 写文件前先检查原来是否有同名文件, 如有则将该文件原有文件删除, 如无同名文件就建立一个新文件, 然后将写当前位置设定于文件的开头, 以便从文件开头写入数据。

4. 文件的关闭

C 语言使用完后, 应该及时关闭。以防止由于误操作等原因破坏已经打开的文件。

文件的关闭通过库函数进行;

功能: 将文件指针 fp 所指文件关闭

关闭文件意味着:

1. 释放文件有关信息区
2. 将输出文件缓冲区的内容(无论缓冲区是否为满)都输出写入文件, 然后关闭文件, 这样可以防止丢失本来应写入文件的数据。

5. 常用读写文件的函数

- (1) 格式化的读(输入)写(输出)文件函数要求文件的输入合适与输出格式对应统一。

1. 格式化的读文件函数一般调用格式串读入数据。函数执行成功返回读的数据项个数; 若在读第一项前已到达文件结束处, 则返回 EOF。

2. 格式化的写文件函数一般调用形式:

如果写入成功, 返回写入文件的输出表列的数目。

- (3) 数据块的读文件函数。

1. 数据块的读文件函数。

数据块的读文件函数一般调用形式

返回所指文件中的数据个数。

数据块的读写文件函数每次都需要指定读写的长度, 通常用于二进制文件的读写。

- (4) 读写字符文件函数

1. 独自付函数 fgetc () 一餐调用形式:

Ch=fgetc (文件型指针变量);

Ch 为字符变量, 用以接收函数带回的字符

功能：从指定的文件读入一个字符，该文件必须是以读或者写的格式打开的。如果在执行的时候遇到结束的字符，返回一个文件结束标识符。

2. 读函数的一般调用形式：

fputc (ch, 文件型指针变量)

其中 ch 是一个字符常量，也可以是一个字符变量

功能：把一个字符写到指定文件中。函数的返回值是输出的字符，失败就返回 EOF (-1)

3. 读字符串函数 fgets () 一般调用形式：

Fgets (字符数组, 字符数 n, 文件型指针变量)

功能：从 fp 指定的文件中读取一个串，若满足以下条件读取结束：已读取 n-1 个字符；读取到回车符；读取到 EOF。读取成功后函数返回值为字符数组的首地址。

读取的字符个数至多 n-1 个，剩下的字符空间为结束符的。读取回车符的时候，作为第一个字符送到缓冲区。然后再加一个字符串结束标志。

4. 写字符串函数 fputs () 一般调用形式：

注意这个函数会自动加上字符串结束标志。

5. 文件的其他常用函数

1, Feof 函数调用形式

Feof (文件型指针变量)

功能：判断文件型指针变量所指向的文件是否到达尾端。如果文件型指针变量所指向的文件的当前位置为文件最后的结束标志 EOF，则函数返回一个非零值，否则返回 0 值。

Rewind 函数调用形式：rewind (文件型指针变量)

功能：是将指向文件的读写指针重新设置为该文件的开头

每个打开的文件都有一个文件位置指针，它指示对应文件进行读写操作的具体位置，所以也称作文件读写指针。读写位置总是表示为距文件开头的字节数。

Fseek (文件型指针变量, 偏移量, 起始位置)

功能：以文件的起始位置为标准，根据偏移量往前或往后移动读写指针。其中偏移量是一个长整型数，表示从起始位置移动的字节数，正数表示指针往后移动，负数表示指针往前移动。其实为遏制用数字 0, 1, 2 或者符号常量代表文件开始文件当前为遏制和文件结束为止。拖文件指针设置成功，返回 0；否则是非零值。重置文件指针的时候，可以超出文件尾部，但是不能指向文件开头之前。

Ftell (文件型指针变量)

功能：给出文件中读写指针的当前值。返回值是一个长整型，如果测试成功，则返回读写指针距离文件开头的字节数。否则返回 -1L。

6. 文件的顺序存取和随机存取

文件按照读写方式，可分为顺序文件和随机文件。顺序文件的读写只能从头开始顺序读取，随机文件的读写过程可以随机进行。C 语言生成的文件既可以顺序存取也可以随机存取，并且以上介绍的文件的读写函数都适用于这两种读写，fseek 函数可以用来实现稳健的随机存取。

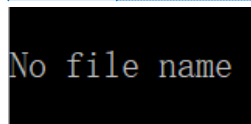
范例二

题目编程实现分页打印 ASCII 文件，并加上行号。要求用命令行方式。

实验代码：

```
#include "stdio.h"
#include <cstdlib>
#define TLINE 10
#define LINEP 2
main(int argc, char *argv[]){
    int flag, page, line, i;
    char buf[100];
    FILE * fp;
    if(argc < 2){
        printf("\nNo file name\n"); exit(0);
    }
    if((fp = fopen(argv[1], "r")) == 0){
        printf("\nNo file name\n"); exit(0);
    }
    flag = page = line = 1;
    while(flag){
        for(i = 0; i < LINEP; i++) printf("\n");
        printf("%s\n", argv[1]);
        for(i = 0; i < TLINE; i++){
            if((fgets(buf, 80, fp)) == 0)
                flag = 0;
            if(flag){
                printf("%6d %s", line++, buf);
                if(buf[78] != 0) printf("\n");
            }
            else
                for(i += 2; i < TLINE; i++) printf("\n");
        }
        printf("\n%6s%d.\n", ".", page++);
        for(i = 0; i < LINEP; i++) printf("\n");
    }
    fclose(fp);
}
```

实验结果：



No file name

实验第一题

题目：有 5 个学生，每个学生有 3 门课程的成绩，从键盘输入学生数据（包括学号，姓名，3 门课程成绩），并计算出每人的平均成绩。将原有数据和计算出的平均成绩存放在磁盘文件中。

实验代码：

批注 [管2]: 这个需要结合命令行参数进行，结果是实验书上类似的情况

```
#include <stdio.h>

#define ID 11

typedef struct _stu
{
    char id[ID];
    int sco1;
    int sco2;
    int sco3;
}stu;

int main()
{
    int i;
    stu s[5];
    FILE *fp;
    float ave;
    printf("请输入五个学生的学号和三门成绩: \n");
    for(i=0;i<5;i++)
    {
        scanf("%s%d%d%d",s[i].id,&s[i].sco1,&s[i].sco2,&s[i].sco3);
    }

    if((fp=fopen("stud_dat.txt","w"))==NULL)
    {
        printf("文件打开失败\n");
        return(0);
    }

    for(i=0;i<5;i++)
    {
        ave=(s[i].sco1+s[i].sco2+s[i].sco3)/3.0;
        fprintf(fp,"%s %d %d %d %.2f\n",s[i].id,s[i].sco1,s[i].sco2,s[i].sco3,ave);
    }

    fclose(fp);
    return 0;
}

实验结果:
```

请输入五个学生的学号和三门成绩:

boa 1 2 3

our 4 5 6

nboeis 6 7 8

gnire 5 6 8

hiosd 4 7 3

boa 1 2 3 2.00

our 4 5 6 5.00

nboeis 6 7 8 7.00

gnire 5 6 8 6.33

hiosd 4 7 3 4.67

实验第二题

实验题目: 将上述第一题生成的文件中的学生数据, 按平均成绩进行排序处理, 将已排序的学生数据存入一个新文件中。

实验代码:

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

```
#include <stdlib.h>
```

```
#include <string.h>
```

```
#define ID 11
```

```
typedef struct _stu
```

```
{
```

```
char id[ID];
```

```
int sco1;
```

```
int sco2;
```

```
int sco3;
```

```
float ave;
```

```
}stu;
```

```
void swap(struct _stu* a, struct _stu* b) {
```

```
    struct _stu tmp;
```

```
    tmp.sco1 = a->sco1;
```

```
    tmp.sco2 = a->sco2;
```

```
    tmp.sco3 = a->sco3;
```

```
    tmp.ave = a->ave;
```

```
    strcpy(tmp.id, a->id);
```

```
    a->sco1 = b->sco1;
```

```
    a->sco2 = b->sco2;
```

```
    a->sco3 = b->sco3;
```

```
    a->ave = b->ave;
```

```

        strcpy(a->id, b->id);

        b->sco1 = tmp.sco1;
        b->sco2 = tmp.sco2;
        b->sco3 = tmp.sco3;
        b->ave = tmp.ave;
        strcpy(b->id, tmp.id);
    }

void deal(struct _stu *a)
{
    int i,j;
    for(i=0;i<5-1;i++)
        for(j=0;j<4-i;j++)
            if((a[j].ave)>(a[j+1].ave))
            {
                swap(&a[j], &a[j + 1]);
            }
}

int main()
{
    int i;
    stu s[5];
    FILE *fp;
    float ave;
    printf("请输入五个学生的学号和三门成绩: \n");
    for(i=0;i<5;i++)
    {
        scanf("%s%d%d%d",s[i].id,&s[i].sco1,&s[i].sco2,&s[i].sco3);
    }

    if((fp=fopen("stud_dat.txt","w"))==NULL)
    {
        printf("文件打开失败\n");
        return(0);
    }

    for(i=0;i<5;i++)
    {
        s[i].ave=(s[i].sco1+s[i].sco2+s[i].sco3)/3.0;
    }

    deal(s);

```

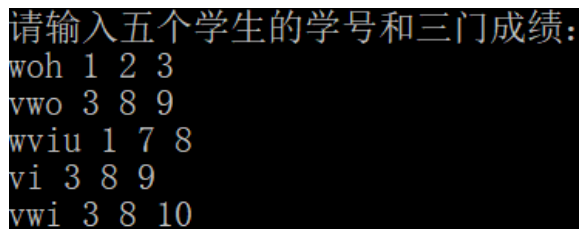
```

for(i=0;i<5;i++)
{
s[i].ave=(s[i].sco1+s[i].sco2+s[i].sco3)/3.0;
fprintf(fp,"%s %d %d %d %.2f\n",s[i].id,s[i].sco1,s[i].sco2,s[i].sco3,s[i].ave);
}

fclose(fp);
return 0;
}

```

实验结果：



```

请输入五个学生的学号和三门成绩:
woh 1 2 3
vwo 3 8 9
wviu 1 7 8
vi 3 8 9
vwi 3 8 10

```

```

woh 1 2 3 2.00
wviu 1 7 8 5.33
vwo 3 8 9 6.67
vi 3 8 9 6.67
vwi 3 8 10 7.00

```

调试分析：

实验总结：

在调换结构体的内容的时候可以将 temp 设置为一个新的结构体，然后用结构体中的内容进行交换，然后用结构体中的元素进行交换，在交换的时候如果遇到了字符串常量可以用 c 自带的系统函数进行操作，然后我对排序的方法还是不熟悉，还有就是 printf 函数中后面的部分要求的是地址，地址可以赋值给指针变量但是地址之间不能相互赋值不然会引起混乱，然后注意在串结构体的指针的时候要引用结构体的前面的部分当做地址传入函数中，然后要注意在定义新的变量进行交换的时候要注意自己想要交换的变量的类型，还有就是指向结构体中的元素的时候要区分这个到底是指针还是元素的具体指

第三题

题目：读取"stu_sort"文件内容，并打印输出（输出到屏幕）

实验代码：

```

#include <stdio.h>
int main(void)
{
    char line[1024];
    FILE *fp_read = NULL; /* 定义一个文件指针 */

```



```

if (!(fp_read = fopen("stud_dat.txt", "r"))) /* 打开文件 */
{
    printf("打开文件错误! \n");
    return 1;
}
do
{
    fgets(line, 1024, fp_read);
    printf("%s", line);
}while(!feof(fp_read));
fclose(fp_read); /* 解除文件指针和文件的关联 */
return 0;
}

```

实验结果:

```

woh 1 2 3 2.00
wviu 1 7 8 5.33
vwo 3 8 9 6.67
vi 3 8 9 6.67
vwi 3 8 10 7.00
vwi 3 8 10 7.00

```

调试分析: 无

实验总结: 奇怪

第五题

题目: 编写程序统计某给定 ASCII 文件中各字母的出现频率。

实验代码:

```

#include<stdio.h>
void sort(int a[],int n)
{int i,j,t;
  for(i=0;i<n-1;i++)
    for(j=0;j<n-1-i;j++)
      if(a[j]<a[j+1])
        {t=a[j];a[j]=a[j+1];a[j+1]=t;}
}
int main()
{
  FILE *fp;
  int i,a[26]={0},b[26]={0}; char c;
  fp=fopen("stud_dat1.txt","r");
  if(fp!=NULL)
  do
  {c=fgetc(fp);
   if(c>='a'&&c<='z')b[c-'a']++;
   else if(c>='A'&&c<='Z')a[c-'A']++;
  }while(!feof(fp));
}

```

```

}while(!feof(fp));
sort(a,26);
sort(b,26);
for(i=0;a[i];i++)
    printf("%4c:%3d",'A'+i,a[i]);
printf("\n");
for(i=0;b[i];i++)
    printf("%4c:%3d",'a'+i,b[i]);
printf("\n");
fclose(fp);
return 0;
}

```

实验结果:

```

A:  3  B:  1  C:  1  D:  1
a:  6  b:  6  c:  5  d:  4  e:  4  f:  2  g:  2  h:  2  i:  2  j:  2  k:  1  l:  1  m
:  1  n:  1  o:  1

```

批注 [管3]: 统计频率不是频数

调试分析: 无

实验总结: 无

第六题

题目: 编写程序, 分别求出给定整数文件中等于大于某给定整数值的元素个数。

实验代码:

```

#include<stdio.h>
main(){
    int a[20]={},j=0;
    FILE *fp;
    fp=fopen("data.txt","r");
    for(int i=0;i<20;i++){
        fscanf(fp,"%d",&a[i]);
        printf("%d ",a[i]);
    }
    for(int i=0;i<20;i++){
        // if(a[i]==1)
        if((a[i]==10)||a[i]>10)
            ++j;
    }
    printf("%d",j);
    fclose(fp);
}

```

实验结果:

```

1 2 3 4 5 6 7 8 9 8 6 4 23 56 76 87 7 99 0 55 6

```

调试分析: 无

实验总结: 无