计算机程序设计基础（1）

14 文件

清华大学电子工程系

杨昉

E-mail: fangyang@tsinghua.edu.cn

课程回顾

- **结构体变量与数组**
  - 结构体变量：定义、嵌套、初始化方法、作为函数参数
  - 结构体数组：定义与引用、作为函数参数
- **结构体与指针**
  - 结构体指针变量与指针数组：理解指针的含义、函数参数、引用
- **链表**
  - 概念和基本运算：查找、元素插入/删除、打印链表、逆转链表
- **联合体与枚举类型**
  - 联合体与结构体的辨析、枚举类型定义和使用

课程回顾：结构体

类型	定义	说明
结构体	struct 结构体类型名 {成员表}; struct 结构体类型名 变量表; struct 结构体类型名 {成员表} 变量表; struct {成员表} 变量表;	不同类型成员的**复合类型**，结构体类型名定义的是**类型**，**struct**不可省略；**结构体变量名.成员名**；结构体可嵌套；可定义结构体数组,与函数结合参考数组
结构体指针	p=&st1: st1.num, (*p).num, p->num, p[0].num	指向结构体类型**变量**或**数组**(或**数组元素**)的起始地址，"->"被称为**指向运算符**
联合体	union 联合体名 {成员表}; 引用：联合体变量名.成员名	各种不同数据**共用同一段存储空间**，按**最大成员**分存储空间，**成员首地址相同**
枚举类型	enum 枚举类型名{枚举元素列表}; enum 枚举类型名{枚举元素列表}变量表; enum {枚举元素列表}变量表;	变量的值**只限于列举出来的值的范围**内；可以给枚举量**赋值**,还可**整型值强制转换**

自定义类型：**typedef 原类型名 新类型名;** 增加了一个**类型别名**,而没有定义新的类型

链表：每个存储结点包含**数据域**和**指针域**,可进行链表**查找**、**插入**、**删除**、**逆转**

第14讲课程目标

14.1 文件的基本概念

- 文本文件与二进制文件
- 缓冲文件系统
- 文件类型指针

14.2 文件的基本操作

- 文件打开与关闭
- 文件读写
- 文件定位

14.1 文件的基本概念

- 文本文件与二进制文件
  - ✓ 文本文件
  - ✓ 二进制文件
- 缓冲文件系统
  - ✓ 高级文件系统
  - ✓ 低级文件系统
- 文件类型指针
  - ✓ 文件操作

文件的基本概念：文本与二进制文件

- 文件
  - 文件是指存储在**外存储器**上的**数据的集合**，可永久保存、复制、传输等
  - 每个文件都有一个名字，称为**文件名**。文件名的命名规则比较随意，且与操作系统密切相关，Windows操作系统**中文件名一般不能出现*，?等**
  - 一般来说，**同一个目录下**的不同的文件**有不同的文件名**，计算机操作系统就是根据文件名对各种文件进行存取，并进行处理

文件的基本概念：文本与二进制文件

- 文本文件
  - 文本文件又称为ASCII文件。在这种文件中，每个字节存放一个字符的ASCII码值。每个整数或实数是由每位数字的ASCII字符组成
    - 举例：.c、.cpp等源程序文件是文本文件
- 二进制文件
  - 二进制文件中的数据与该数据在内存中的二进制形式是一致的，是把整数的补码、浮点数的IEEE 754表示直接写入文件中，其中一个字节并不代表一个字符。可执行文件都是二进制文件
    - 举例：.jpg、.avi、.docx等都是二进制文件

文件的基本概念：文本与二进制文件

- 例14-1：文本文件和二进制文件写入

```
1  #include <stdio.h>                          //包含头文件"stdio.h"
2  void main(){
3    int a=12345,b=567890; float f=97.6875f; double f2=97.6875;
4    FILE *fp;
5    fp=fopen("data1.txt","w");                //文本文件
6    fprintf(fp,"%d %d %f %f\n", a, b, f, f2);
7    fclose(fp);                               //关闭文件
8    fp=fopen("data2.txt","wb");               //二进制文件
9    fwrite(&a,sizeof(int),1,fp);              //块写入
10   fwrite(&b,sizeof(int),1,fp);
11   fwrite(&f,sizeof(float),1,fp);
12   fwrite(&f2,sizeof(double),1,fp);
13   fclose(fp);                               //关闭文件
14 }
```

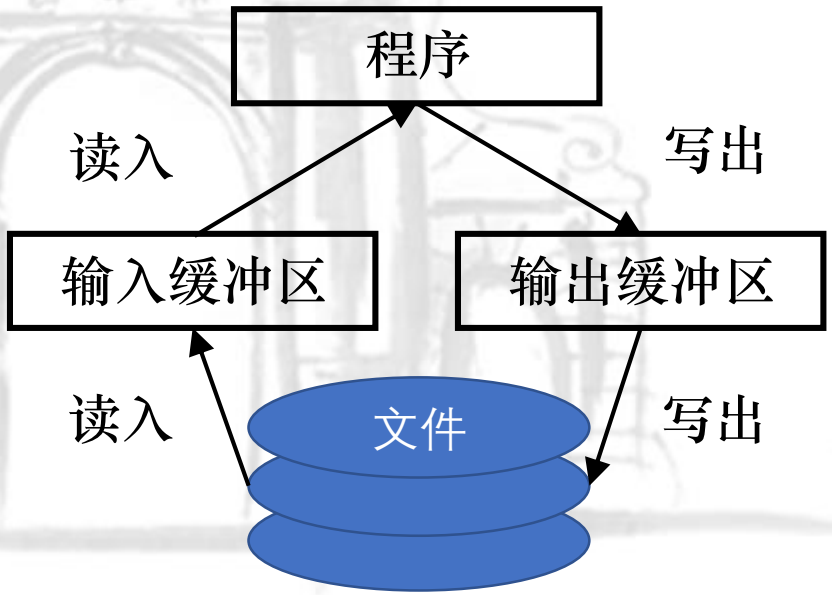
文件的基本概念：文本与二进制文件

- 例14-1写入结果
  - data1.txt结果

  - data2.txt结果

文件的基本概念：缓冲文件系统

- 文件系统的定义
  - 在C语言中，对**文件的操作**都是通过**库函数**来实现的
  - 文件系统可以分为**高级文件系统**和**低级文件系统**，这是按照是否有**缓冲文件系统**来划分的
- 高级文件系统示意图
  - 输入输出都引入缓冲区
  - 读入和写出都经过缓冲区



文件的基本概念：缓冲文件系统

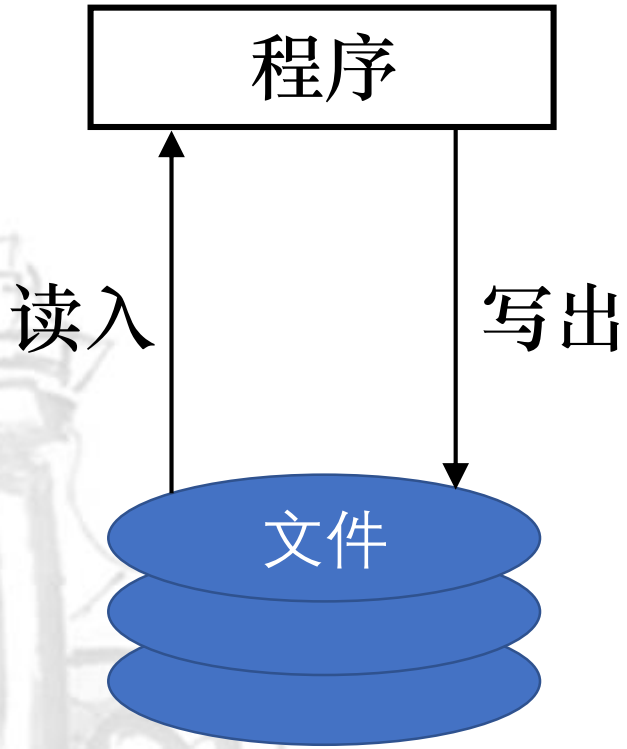
- 高级文件系统（缓冲文件系统）
  - 指系统自动地为正在被使用的文件在内存中**开辟一个缓冲区**
  - 当需要向外存储器中的文件输出数据时，必须先将数据送到为**该文件开辟的缓冲区**，缓冲区满以后才一起送到外存储器
  - 当需要从外存储器中的文件读入数据进行处理时，也首先一次从外存储器将一批数据**读入缓冲区**中（将缓冲区充满），然后再从缓冲区中将数据逐个**读入进行处理**
  - 由此可以看出，在缓冲文件系统中，对文件的**输入输出**是通过为该文件开辟的缓冲区进行的，对文件中**数据的处理**也是在该缓冲区中进行的

文件的基本概念：缓冲文件系统

- 设置文件缓冲区优势
  - 因为I/O操作可能是机械操作，不可能每秒钟操作上万次，因此过多的I/O操作会影响整个程序的**执行效率**
  - 设置文件缓冲区可减少程序直接进行**I/O操作的次数**，将每次读写一个数进行一次I/O操作，合并为多次读写仅仅进行一次I/O操作，从而提高效率和整个程序的执行速度
- 设置文件缓冲区缺点
  - 由于多次读写合并为一次I/O操作，如果要写出的数据因为**先写进缓冲区**，还没有真正写到磁盘U盘等介质上，此时程序非正常终止，会出现缓冲区中的**数据丢失的现象**

文件的基本概念：非缓冲文件系统

- 低级文件系统（非缓冲文件系统）
  - 是指系统**不自动**为文件开辟缓冲区，而是由用户程序自己为文件设定缓冲区
  - 这种情况，是程序的每次I/O读写，都**直接访问**磁盘、U盘等介质
  - 实例就是每次开机时，加载操作系统的过程，就是**非缓冲文件**操作
  - 示意图如右图，直接读入和写出



非缓冲文件系统示意图

文件的基本概念：文件类型指针

- 定义
  - 在C语言的缓冲文件系统中，用**文件类型指针**来标识打开的文件
  - 形式

```
FILE *指针变量名;
```

  - 系统将该结构体类型定义为FILE（注意：是**英文大写字母**），简称**文件类型**
  - 其中指针变量名用于指向一个文件，实际上是指向用于存放文件数据**缓冲区的首地址**

文件的基本概念：文件类型指针

- 文件操作步骤
  - 打开文件：在计算机内存中开辟一个**缓冲区**，用于存放被打开文件的有关信息
  - 文件处理：包括在缓冲区中**读写数据以及定位**等操作
  - 关闭文件：将缓冲区中的内容**写回到外存**（磁盘，固盘，U盘等），并**释放**缓冲区
- 例14-2：FILE的定义

FILE的定义

```
1  struct _iobuf {
2    char *_ptr;
3    int _cnt;
4    char *_base;
5    int _flag;
6    int _file;
7    int _charbuf;
8    int _bufsiz;
9    char *_tmpfname;
10 };
11 typedef struct _iobuf FILE;
```

14.2 文件的基本操作

- 文件打开与关闭
  - ✓ 文件的打开
  - ✓ 文件的关闭
- 文件的读写
  - ✓ 文本文件：字符/字符串读写
  - ✓ 二进制文件：块读写
  - ✓ 文件的终止判断
- 文件的定位

文件的基本操作：文件打开

- 打开文件
  - 形式：

```
FILE *fp;
...
fp=fopen("文件名","文件打开方式");
```

  - fopen() 函数：文件名是要打开的文件名字；文件打开方式是指出以何种方式打开文件。两者**均为字符串**
  - 主要功能是为需要打开的文件**分配一个缓冲区**，并返回该缓冲区的**首地址**

文件的基本操作：文件打开

- 文本文件打开方式

标志	含义	详细说明
r	只读	为读打开一个文件。若指定的文件不存在，则返回空指针值NULL
w	只写	为写打开一个新文件。若指定的文件已存在，则其中原有内容被删去；否则创建一个新文件
a	追加写	向文件尾增加数据。若指定的文件不存在，则创建一个新文件
r+	读写	为读写打开一个文件。若指定的文件不存在，则返回空指针值NULL
w+	读写	为读写打开一个新文件。若指定的文件已存在，则其中原有内容被删去；否则创建一个新文件
a+	读与追加写	为读写向文件尾增加数据打开一个文件。若指定的文件不存在，则创建一个新文件

文件的基本操作：文件打开

- 二进制文件打开方式：**符号后面加b**

标志	含义	详细说明
rb	只读	为读打开一个二进制文件。若指定的文件不存在，则返回空指针值NULL
wb	只写	为写打开一个新二进制文件。若指定的文件已存在，则其中原有内容被删去；否则创建一个新文件
ab	追加写	向文件尾增加数据。若指定的文件不存在，则创建一个新二进制文件
r+b	读写	为读写打开一个二进制文件。若指定的文件不存在，则返回空指针值NULL
w+b	读写	为读写打开一个新二进制文件。若指定的文件已存在，则其中原有内容被删去；否则创建一个新文件
a+b	读与追加写	为读写向文件尾增加数据打开一个二进制文件。若指定的文件不存在，则创建一个新文件

文件的基本操作：文件打开

- 打开文件报错
  - 在一般的C程序中，常采用以下方式来打开文件：

```
1  FILE *fp;
2  if ( (fp=fopen("文件名","文件打开方式"))==NULL ){
3    printf("Cannot open this file!\n");
4    exit(0);                                    //终止调用过程
5  }
```

  - 打开一个文件有时会出错，例如"r"方式打开一个不存在文件
  - 在以上述方式打开文件时，如果出现“打开”错误，fopen()函数**返回空指针值**，程序就显示以下信息，并退出当前的调用执行过程

```
Cannot open this file!
请按任意键继续……
```

文件的基本操作：文件打关闭

- 关闭文件
  - 形式：

```
fclose(fp);
```

  - fclose( )函数的主要功能是将**由fp指向的缓冲区**中的数据存放到外部设备的文件中，然后**释放该缓冲区**
  - 如果文件关闭成功,fclose()**返回0**，否则**返回EOF**(即-1)
  - 文件**被关闭后**，如果再想对该文件进行操作，则**必须重新打开**
  - 如果不关闭已经处理完的文件，当打开的文件个数很多时，会**影响对其他文件**的打开操作

建议当一个文件使用完后应立即关闭它！

文件的基本操作：文件的读写

- 字符读函数：主要适用于**文本文件的字符读操作**
  - 读操作：从指定的文件向程序输入数据
  - fgetc()函数：

```
char fgetc(FILE *fp);
```

其中fp为**文件类型的指针**，指向已打开的文件

  - 该函数的功能是，从指定的文件读入一个字符并返回，**读取到文件末尾**或**读取失败**时**返回EOF**

```
1 char ch;
2 FILE *fp=fopen("文件名","文件打开方式");
3 ch = fgetc(fp);
```

文件的基本操作：文件的读写

- 字符写函数：主要适用于**文本文件的字符写操作**
  - 写操作：将程序中处理好的数据写到指定的文件中
  - fputc()函数：

```
int fputc(char ch, FILE *fp);
```

其中fp为文件类型的指针，指向已打开的文件；这里的ch可以是字符型变量，也可以是字符型常量或字符型表达式

  - 该函数的功能是：将一个字符写到指定的文件中。若写成功，则返回已输出的字符，否则返回文件结束标志EOF

文件的基本操作：文件的读写

- 例14-3：从文件a.dat中顺序**读入字符**并在屏幕上显示

```
1  #include <stdio.h>
2  #include <stdlib.h>   //使用exit()函数必须引用
3  void main() {
4    char c;
5    FILE *fin;
6    if ( (fin=fopen("a.dat","r"))==NULL ) {
7      printf("cannot open this file!\n");
8      exit(0);
9    }
10   c=fgetc(fin);          //从文件读取一个字符
11   while (c!=EOF) {       //EOF为文件的结束标志，值为-1
12     putchar(c);          //在屏幕上显示字符
13     c=fgetc(fin);        //继续从文件读取一个字符
14   }
15   fclose(fin);
16 }
```

a.dat的内容为:
abcdefg
12345

```
abcdefg
12345请按任意键继续……
```

文件的基本操作：文件的读写


- 例14-4：键盘输入文本写到abc.txt文件中，#作为键盘输入结束

```
1  #include <stdio.h>
2  #include <stdlib.h>   //使用exit()函数必须引用
3  void main() {
4    char c;
5    FILE *fout;
6    if( (fout=fopen("abc.txt","w"))==NULL ){
7      printf("cannot open this file!\n");
8      exit(0);
9    }
10   c=getchar();          //从键盘读取一个字符
11   while(c!= '#' ){      //#为输入结束标志
12     fputc(c, fout);     //向文件写入一个字符
13     c=getchar();        //继续从键盘读取一个字符
14   }
15   fclose(fout);
16 }
```

```
akddndn
dldlld
asss#
请按任意键继续……
```

abc.txt的内容为:
akddndn
dldlld
asss

文件的基本操作：文件的读写

- 字符串读函数：主要适用于**文本文件的字符串读操作**
  - fgets()函数：`char *fgets(char*string, int n, FILE *fp);`
    - 其中fp为文件类型的指针，指向已打开的文件
    - string是一个字符串指针；n是一个整型变量或整型常量或整型表达式
  - 从指定的文件读入一行字符（不超过n-1个字符）存放到由string指向的存储空间中，结束后在字符串最后加字符串结束符'\0'
    - 当读入正确时，函数返回值是与string相同的指针值，函数返回值为NULL则表示读入不成功
    - 如果在未读满n-1个字符时，就已经读到一个换行符或文件结束标志EOF，则将结束本次读操作，但包括回车符也被读入到字符串string中

课堂练习1

- 读入一行文本，写出输出结果
  - 第1行不足19个字符，连回车符也读入到字符串中并输出到了屏幕
  - 第2行超过19个字符，因此fgets只读入前19 个字符，fgets下一次读入了第2行剩余的字符连同回车符

```
1  #include <stdio.h>
2  #include <string.h>
3  void main() {
4    char a[20]; int n;  FILE *fp;
5    fp=fopen("d:\\1.txt","r");
6    for (n=0; n<3; n++){
7      fgets(a,20,fp);
8      printf("%s%d\n",a,strlen(a));
9    }
10   fclose(fp);
11 }
```

1.txt的内容为:
abcdefghijklmnopqr
abcdefghijklmnopqrstuvwxyz

```
abcdefghijklmnopqr
19
abcdefghijklmnopqrs 19
tuvwxyz
8
请按任意键继续……
```

文件的基本操作：文件的读写

- 字符串写函数：主要适用于**文本文件的字符串写操作**
  - fputs()函数：`int fputs(char *string, FILE *fp);`
    - 其中fp为文件类型的指针，指向已打开的文件
    - string可以是一个字符串常量，也可以是一个指向字符串的指针，还可以是存放字符串的数组名
  - 将指定的字符串写到文件fp中，若写成功，则返回非负值；若写不成功，则返回EOF
    - 字符串中最后的结束符'\0'并不写到文件中，也不自动加换行符'\n'
    - 因此在写字符串到文件时，必要时可以人为加入如'\n'这样的字符

文件的基本操作：文件的读写

- 判断文件结束
  - 在C语言中，文本文件是以**读入EOF**作为文件结束标志的
  - 二进制文件**不是以EOF**作为文件结束标志的
  - C语言提供了一个**feof()函数**，专门用来判断文件是否结束
- feof()函数
  - 功能：在读fp指向的文件时判断是否遇到文件结束。如果遇到文件结束，则函数feof(fp)的**返回非0值**；否则**返回值为0**
  - 形式：

```
int feof(fp)
```

文件的基本操作：文件的读写

- 例14-5：用feof重写例14-3

```
1  #include <stdio.h>
2  #include <stdlib.h>   //使用exit()函数必须引用
3  void main() {
4    char c;
5    FILE *fin;
6    if ( (fin=fopen("a.txt","r+"))==NULL ){
7      printf("cannot open this file!\n");
8      exit(0);
9    }
10   c=fgetc(fin);          //从文件读取一个字符
11   while (!feof(fin)){ //未到文件尾继续循环
12     putchar(c);          //在屏幕上显示字符
13     c=fgetc(fin);        //继续从文件读取一个字符
14   }
15   fclose(fin);
16 }
```

a.txt的内容为:
abc

```
abc
请按任意键继续……
```

文件的基本操作：文件的读写

- 例14-6：用feof实现字符串的读出和写入

```
1  #include <stdio.h>
2  #include <stdlib.h>   //使用exit()函数必须引用
3  void main() {
4    char a[80];
5    FILE *fin, *fout;
6    if ((fin=fopen("a.dat","r"))==NULL || (fout=fopen("b.dat","w"))==NULL){
7      printf("cannot open this file!\n");
8      exit(0);
9    }
10   while (!feof(fin)){ //判断文件是否结束
11     fgets(a,80,fin);  //从文件读取一行字符
12     fputs(a,fout);    //写到另一个文件中
13   }
14   fclose(fin);fclose(fout);
15 }
```

文件的基本操作：文件的读写

- 例14-6中用feof实现字符串的读出和写入

若a.dat的内容为:
abcdefg
12345
998e929292
ddlldldldldldlldldldldl

则b.dat的内容为:
abcdefg
12345
998e929292
ddlldldldldldlldldldldl
ddlldldldldldlldldldldl

- 因为feof(fin)只有当读不成功才**置为1**，因此老是慢一拍!
- 上面循环最后一次读不成功，把**上次读入的字符串又写了一次**
- 可以通过fgets()函数的**返回值是否为NULL**来确定是否成功读入

文件的基本操作：文件的读写

- 例14-7：避免出现最后一行重复写的问题，修改为

```
1  #include <stdio.h>
2  #include <stdlib.h>   //使用exit()函数必须引用
3  void main() {
4    char a[80];
5    FILE *fin, *fout;
6    if ((fin=fopen("a.dat","r"))==NULL || (fout=fopen("b.dat","w"))==NULL){
7      printf("cannot open this file!\n");
8      exit(0);
9    }
10   while (!feof(fin)){                    //判断文件是否结束
11     if (fgets(a,80,fin)==NULL)           //若读不成功,立刻终止
12       break;
13     fputs(a,fout);                       //写到另一个文件中
14   }
15   fclose(fin);fclose(fout);
16 }
```

b.dat的内容为:
abcdefg
12345
998e929292
ddlldldldldldlldldldldl

文件的基本操作：文件的读写

- 数据块读函数：主要适用于**二进制文件的块读操作**
  - 从指定的文件中以二进制格式读入一组数据到指定的内存区
    - 若成功读入，则函数返回值是读入的项数
    - 若函数返回值是小于等于0的数，则表示读入不成功
  - fread ( )函数：int fread(void *buffer, int size, int count, FILE *fp);
    - buffer表示存放读入数据的内存首地址
    - size表示每个数据项的字节数
    - count表示数据项个数
    - fp为文件类型的指针，指向已打开的二进制文件

文件的基本操作：文件的读写

- 例14-8：从二进制文件b.dat中读入4个整数存放到数组x中

```
1  #include <stdio.h>
2  #include <stdlib.h>   //使用exit()函数必须引用
3  void main() {
4    int x[4];
5    FILE *fp;
6    if ( (fp=fopen("b.dat","rb"))==NULL ){
7      printf("cannot open this file!\n");
8      exit(0);
9    }
10   fread(x,sizeof(int),4,fp);
11   fclose(fp);
12 }
```

文件的基本操作：文件的读写

- 数据块写函数：主要适用于**二进制文件的块写操作**
  - 将一组数据以二进制格式写到指定的文件中
    - 若成功读入，则函数返回值是写出的项数
    - 若函数返回值是小于等于0的数，则表示写入不成功
  - fwrite( )函数：int fwrite(void *buffer, int size, int count, FILE *fp);
    - buffer表示输出数据的内存首地址
    - size表示每个数据项的字节数
    - count表示数据项个数
    - fp为文件类型的指针，指向已打开的二进制文件

文件的基本操作：文件的读写

- 例14-9：下列C程序将一维数组中的元素写到二进制文件c.dat中

```
1  #include <stdio.h>
2  #include <stdlib.h>   //使用exit()函数必须引用
3  void main() {
4    double x[5]={1.1, 2.3, 4.5, -3.6, 9.5};
5    FILE *fp;
6    if ( (fp=fopen("b.dat","wb"))==NULL ){
7      printf("cannot open this file!\n");
8      exit(0);
9    }
10   fwrite(x,sizeof(double),5,fp);
11   fclose(fp);
12 }
```

文件的基本操作：文件的读写

- 格式读函数：主要适用于**文本文件的格式化读操作**
  - 从指定的文本文件中格式化读入数据
    - 若成功，则返回值为读入并进行格式**转换且赋值的项数**；
    - 返回的值中不包括读入但没有成功进行格式转换并赋值的项数
    - 若没有任何项被正确读入，函数值**返回0**；若读到文件尾，**返回EOF**（即-1）
  - fscanf( )函数：

```
int fscanf(文件指针, 格式控制, 地址表);
```

文件的基本操作：文件的读写

- fscanf()函数与格式输入函数scanf()很相似，区别在于：
  - scanf()函数是从**键盘输入数据**，而fscanf()函数是从**文件读入数据**
  - 在fscanf()函数参数中多了一个**文件指针**，用于指出从哪个文件读入数据
  - 由于标准输入的设备名为**stdin**，因此fscanf(stdin,格式控制,地址表); 等价于 scanf(格式控制，地址表);

文件的基本操作：文件的读写

- 例14-10：下面C程序从文本文件ABC.txt中按格式读入两个整数，分别赋给整型变量a与b

```
1  #include <stdio.h>
2  #include <stdlib.h>   //使用exit()函数必须引用
3  void main() {
4    int a, b;
5    FILE *fp;
6    if( (fp=fopen("ABC.txt","r"))==NULL ){
7      printf("cannot open this file!\n");
8      exit(0);
9    }
10   fscanf(fp,"%d%d", &a, &b);
11   printf("%d%d\n", a, b);
12   fclose(fp);
13 }
```

ABC.txt的内容为:
12
34

```
12 34
请按任意键继续……
```

文件的基本操作：文件的读写

- 格式写函数：主要适用于**文本文件的格式化写操作**
  - 格式化写数据到指定的文本文件中
    - 若成功写出，则返回值为写出的项数
    - 返回负数表示写失败
  - fprintf( )函数：

```
int fprintf(文件指针, 格式控制, 输出表);
```

  - fprintf( )函数与fscanf( )函数是对应的
    - 在使用fscanf()函数从文件读数据时，其格式应与用fprintf()函数将数据写到文件时的格式一致，否则将会导致读写错误

文件的基本操作：文件的读写

- 对比fprintf()函数和printf()函数
  - printf( )函数是将数据输出到**显示屏幕**上，而fprintf( )函数是将**数据写到文件**中
  - 因此在fprintf( )函数参数中多了一个**文件指针**，用于指出将数据写到哪个文件中
  - 标准输出的设备名为**stdout**，因此fprintf(stdout, 格式控制, 地址表); 完全等价于 printf(格式控制, 地址表);

文件的基本操作：文件的读写

- 例14-11：下列C程序将一个整数与一个双精度实数按格式存放到文本文件AB.txt中

```
1  #include <stdio.h>
2  #include <stdlib.h>   //使用exit()函数必须引用
3  void main() {
4    int a=10; double x=11.4;
5    FILE *fp;
6    if ( (fp=fopen("AB.txt","w+"))==NULL ){
7      printf("cannot open this file!\n");
8      exit(0);
9    }
10   fprintf(fp,"%d,%lf", a, x);
11   fclose(fp);
12 }
```

AB.txt的内容为:
10,11.400000

文件的基本操作：文件的定位

- rewind()函数
  - 将文件的**读写指针**移动到**文件的开头**
  - 形式：`void rewind(FILE *fp);`

  其中fp是已经打开的**文件指针**，此函数**没有返回值**
- fseek()函数
  - 将文件的**读写指针**移动到**指定的位置**
  - 形式：`int fseek(FILE *stream, long offset, int origin);`

  stream为**文件指针**，offset为**偏移量**，origin为**起始位置**

文件的基本操作：文件的定位

- fseek()函数参数含义
  - 起始位置是指移动文件读写指针的**参考位置**，它有以下三个值：
    - SEEK_SET 或0 表示从文件首（开头）
    - SEEK_CUR 或1 表示从当前读写的位置
    - SEEK_END 或2 表示从文件尾
  - 偏移量offset是指以**“起始位置”为基点**，文件指针向后或向前移动的字节数
    - 正整数表示向后移动，负整数表示向前移动
    - 偏移量的类型要求为长整型

文件的基本操作：文件的定位


- 例14-12：fseek()函数的使用

```
1.1 2.3 2.3 4.5 4.5 -3.6请按任意键继续……
```

```
#include <stdio.h>
#include <stdlib.h>   //使用exit()函数必须引用
void main() {         //定义main函数，这是程序的主体
  double x[5]={1.1,2.3,4.5,-3.6,9.5}, y[6]={0};
  FILE *fp; int k;
  if((fp=fopen("cc.dat","w+b"))==NULL){printf("cannot open this file!\n"); exit(0);}
  fwrite(x,sizeof(double),5,fp);      //块写入5个二进制double数
  fseek(fp,0L,SEEK_SET);              //回到文件首
  fread(&y[0],sizeof(double),2,fp);   //从y[0]开始连续读入两个double数，即1.1和2.3
  fseek(fp,-4L*(long)sizeof(double),SEEK_END);//从文件尾向前移动4个数，即2.3位置
  fread(&y[2],sizeof(double),2,fp);   //从y[2]开始连续读入两个double数，即2.3和4.5
  fseek(fp,-(long)sizeof(double),SEEK_CUR); //从当前位置向前移动1个double数，即4.5
  fread(&y[4],sizeof(double),2,fp);   //从y[4]开始连续读入两个double数，即4.5和-3.6
  fclose(fp);
  for(k=0; k<6; k++) printf("%4.1f",y[k]);
}
```

文件的基本操作：文件的定位

- fseek()函数定位方法
  - 返回值：如果成功定位，fseek()**返回0**，否则**返回一个非零的值**
    - 对于那些不能重定位的设备，fseek的返回值是不确定的
  - 可以在一个文件中用fseek()把指针重定位在**任何地方**，甚至文件指针可以定位到**文件结束符之外**
    - fseek()将清除文件结束符，忽略先前ungetc调用对输入输出流的作用
  - 若文件是以**追加方式**被打开的，那么当前文件指针的位置是由**上一次I/O 操作决定**的，而不是由下一次写操作在那里决定的
    - 若一个文件是以追加方式打开的，且没有进行I/O操作，那么文件指针是在文件的开始之处

文件的基本操作：文件的定位

- fseek()函数定位方法（续）
  - 对于以文本方式打开的文件，fseek的用途是**有限的**
  - 文本方式下carriage return – linefeed 转换，会使得fseek产生意想不到的结果
    - Windows中对于以文本方式打开的文件，若向文件中写入回车符\r，其ASCII值是0x0D，Windows系统会自动加一个换行符\n(0x0A)
    - 但二进制方式下如果向文件写回车符\r(0x0D),Windows系统不会自动加换行符\n(0x0A)

文件的基本操作：文件的定位

- fseek()函数定位方法（续）
  - 在文本方式下，CTRL+Z 在输入时被当作文件结束符
    - 对于打开进行读写的文件，fopen和所有相关的函数都在文件尾部检测CTRL+Z字符，如果可能就删除
    - 文件结束符EOF不是字符串"EOF"也不是数值-1

https://blog.csdn.net/chouchijiao8952/article/details/101007087

    - 用fseek和ftell在某个文件中移动文件指针时，若这个文件是用CTRL+Z标记结束的，可能会使得fseek在靠近文件尾部时行为失常(定位变得不确切)

文件的基本操作：文件的定位

- fseek()函数定位方法（续）
  - 在文本方式下，fseek操作结果如下
    - 相对于任何起始位置，重定位的位移值是0(offset=0)
    - 从文件起始位置(SEEK_SET)进行重定位，位移的值是用ftell函数返回的值
  - 向文件中写数据就像向磁带中录歌曲一样，同一个位置上，后写入的数据将覆盖并抹去以前的数据
    - 不可能在原来的数据前插入数据
    - 并且新老数据也不可能在同一个位置上同时存在

文件的基本操作：文件的定位

- 例14-13：fseek()函数的使用

```
1  #include <stdio.h>
2  #include <stdlib.h>   //使用exit()函数必须引用
3  void main() {
4    double x[5]={1.1, 2.3, 4.5, -3.6, 9.5}, y[5]={0};
5    FILE *fp; int k;
6    if((fp=fopen("cc.dat","w+b"))==NULL){printf("cannot open this file!\n"); exit(0);}
7    fwrite(x,sizeof(double),5,fp);       //块写入5个二进制double数
8    fseek(fp,0L,SEEK_SET);               //回到文件首
9    fwrite(&x[2],sizeof(double),3,fp); //写入三个double数，用4.5，-3.6和9.5覆盖了1.1，2.3和4.5
10   fseek(fp,0L,SEEK_SET);               //回到文件首
11   fread(y,sizeof(double),5,fp);        //将5个新的double数读入y中
12   fclose(fp);
13   for(k=0; k<5; k++) printf("%4.1f",y[k]);
14 }
```

```
4.5 -3.6 9.5 -3.6 9.5请按任意键继续……
```

文件的基本操作：文件的定位

- ftell()函数
  - 返回文件的当前读写位置（出错返回-1）
  - 形式：`long ftell(FILE *fp);`
  - 其中fp是已经打开的文件指针，此函数返回值为长整型变量

- 例14-14：使用ftell函数获取当前位置

```
1  #include <stdio.h>
2  void main() {
3    FILE *fp; char *p="Hello!";
4    fp=fopen("11.txt","a");
5    printf("ftell=%d\n",ftell(fp));
6    fprintf(fp,"%s\n",p);
7    printf("ftell=%d\n",ftell(fp));
8    fclose(fp);
9  }
```

文件的基本操作：文件的定位

- 例14-14每次运行结果不尽相同
  - 每次运行都用**追加方式**向文件11.txt中写入字符串“Hello!”以及回车符和换行符
  - 每次追加写完后文件指针所在的位置**都是文件尾**
  - 文件11.txt会变得**越来越长**
  - 指针位置的**数值各不相同**

第1次运行结果是：

```
ftell=0
ftell=8
请按任意键继续……
```

第2次运行结果是：

```
ftell=0
ftell=16
请按任意键继续……
```

文件的基本操作：文件的定位

- 例14-15：类似例14-13，在每一步打印出来位置信息

```
#include <stdio.h>
#include <stdlib.h>   //使用exit()函数必须引用
void main() {
  double x[5]={1,2,3,4,5}, y[5]={0};
  FILE *fp; int k;
  if((fp=fopen("a.dat","a+b"))==NULL){printf("cannot open this file!\n"); exit(0); }
  printf("ftell=%d\n",ftell(fp));    //打印位置
  fwrite(x,sizeof(double),5,fp);     printf("ftell=%d\n",ftell(fp));//打印位置
  fseek(fp,0L,SEEK_SET);             //回到文件首
  fread(&y[0],sizeof(double),2,fp); printf("ftell=%d\n",ftell(fp));//读入1、2,打印位置
  fwrite(x,sizeof(double),5,fp);     printf("ftell=%d\n",ftell(fp));//打印位置
  fseek(fp,-4L*(long)sizeof(double),SEEK_END); //从文件尾向前移动4个double数位置
  fread(&y[2],sizeof(double),2,fp); printf("ftell=%d\n",ftell(fp));//读入2、3,打印位置
  fclose(fp);                        //关闭文件
  for(k=0; k<5; k++) printf("%f",y[k]);
}
```

文件的基本操作：文件的定位

- 例14-15输出，第1/2次运行后文件长度为40/80个字节

第1次运行结果是：

```
ftell=0
ftell=40
ftell=16
ftell=40
ftell=24
1.000000 2.000000 2.000000 3.000000 0.000000请按任意键继续……
```

第2次运行结果是：

```
ftell=0
ftell=80
ftell=16
ftell=56
ftell=64
1.000000 2.000000 2.000000 3.000000 0.000000请按任意键继续……
```

文件的基本操作：文件的定位

- 例14-15每次运行时，第二个fwrite(x,sizeof(double),5,fp);看起来没起任何作用，原因如下
  - 文件打开方式是“a+b”，这种方式可在文件尾部**追加写也可读入**
  - 当写入数据后执行9-10行代码；并且在读入操作后，此时立即**执行写操作**（第11行代码）
  - 由于**没有刷新**使得输入输出缓冲区同步，因此这5个double数**没有写入到文件**中，或者说文件写入不成功，文件长度仍然是40个字节
  - 若文件打开方式是“a+b”,在执行读操作后，如果想成功写入，需要先用**fseek**或**rewind移动文件指针**（**位置不限**，可以原地不动）

文件的基本操作：文件的定位

- 例14-16：修正例14-15如下

```
1  #include <stdio.h>
2  #include <stdlib.h>   //使用exit()函数必须引用
3  void main() {
4    double x[5]={1,2,3,4,5}, y[5]={0};
5    FILE *fp; int k;
6    if((fp=fopen("a.dat","a+b"))==NULL){printf("cannot open this file!\n"); exit(0);}
7    printf("ftell=%d\n",ftell(fp));
8    fwrite(x,sizeof(double),5,fp);      printf("ftell=%d\n",ftell(fp));
9    fseek(fp,0L,SEEK_SET);
10   fread(&y[0],sizeof(double),2,fp);   printf("ftell=%d\n",ftell(fp));
11   fseek(fp,0L,SEEK_CUR);      //文件指针保持原地不动，用于清空缓冲区
12   fwrite(x,sizeof(double),5,fp);      printf("ftell=%d\n",ftell(fp));
13   fseek(fp,-4L*(long)sizeof(double),SEEK_END);
14   fread(&y[2],sizeof(double),2,fp);   printf("ftell=%d\n",ftell(fp));   fclose(fp);
15   for(k=0; k<5; k++) printf("%f",y[k]);
16 }
```

文件的基本操作：文件的定位

- 例14-16输出，第1/2次运行后文件长度为80/160个字节

第1次运行结果是：

```
ftell=0
ftell=40
ftell=16
ftell=80
ftell=64
1.000000 2.000000 2.000000 3.000000 0.000000请按任意键继续……
```

第2次运行结果是：

```
ftell=0
ftell=120
ftell=16
ftell=160
ftell=144
1.000000 2.000000 2.000000 3.000000 0.000000请按任意键继续……
```

思维培养——找到定位

- 不论是在工作还是日常生活中，找到自己的 **定位** 都是十分重要的
- 一如在文件的使用中，各项操作的基础也需要基于正确的定位
- 定位是否正确，是**能否正常读/写文件**的基础

找准在服务和融入构建**新发展格局中的定位**

文件的基本操作：文件缓冲区的清除

- fllush()函数
  - 刷新函数fllush()功能是清空文件的输入输出缓冲区，使文件的输入输出缓冲区中的内容**立刻输出**到实际的文件中
  - 形式

```
int fflush(FILE *fp);
```

  - 其中fp是已经打开的文件指针，此函数返回值为**整型变量**
  - 若成功执行，则函数返回值为0；若出错，则函数返回值为-1

文件的基本操作：文件缓冲区的清除

- 例14-17：体现fflush()函数清空输入输出缓冲区

```
1  #include <stdio.h>
2  #include <stdlib.h>   //使用exit()函数必须引用
3  void main() {
4    char c[21];
5    FILE *fp; int i;
6    if ( (fp=fopen("r.txt","w+"))==NULL )
7      { printf( "Cannot open this file!\n"); exit(0); }
8    for (i=0; i<2; i++) fputs("1234567890",fp);
9    fseek(fp,-18L,SEEK_CUR);
10   fputs(" ",fp);
11   fgets(c,20,fp);                          //读取字符串到c数组
12   c[20]='\0';
13   puts(c);                                 //输出字符串
14   fclose(fp);
15 }
```

文件的基本操作：文件缓冲区的清除

- 例14-17运行结果
  - 程序运行结果

```
第1次运行结果是:
234567890123456789O请按任意键继续……
```

  - 文件r.txt结果

```
r.txt的内容为:
12 234567890123456789O
```

  - **问题**：按照程序的操作，应该向文件r.txt输出了2次字符串"1234567890"，r.txt中应该有20个字符

文件的基本操作：文件缓冲区的清除

- 例14-17结果分析
  - fseek(fp,-18L, SEEK_CUR);应该使得文件指针指向第3个字符'3'的开始处
  - 随后fputs (" ", fp);写出一个空格字符，空格字符应该覆盖掉第3个字符'3'，此时文件指针指向第4个字符'4'
  - fgets(c,20, fp);应该读入字符串
    "4567890123456789 0 "
  - 但输出结果不是这样，同样文件r.txt的内容也不是我们期待的： 12 4567890123456789 0

文件的基本操作：文件缓冲区的清除

- 例14-17结果分析
  - 原因是fputs(" ", fp);向文件r.txt中写数据，并不是立刻写到磁盘上的文件中，而是停留在文件输出缓冲区中
  - 文件输出缓冲区中的数据只有当输出缓冲区满，或收到强制写出指令时，才会真正写到磁盘上的文件中
  - 如果此时立刻用fgets()从文件输入缓冲区读入，会造成输入输出缓冲区与磁盘文件的不一致，导致混乱
  - 因此需要在使用fgets()前，用fflush(fp)清空缓冲区，使得输出缓冲区的内容写到磁盘上，这样结果才会保证正确

文件的基本操作：文件缓冲区的清除

- 例14-18：修正例14-17如下

```
1  #include <stdio.h>
2  #include <stdlib.h>   //使用exit()函数必须引用
3  void main() {
4    char c[21];
5    FILE *fp; int i;
6    if((fp=fopen("r.txt","w+"))==NULL){printf("cannot open this file!\n"); exit(0);}
7    for(i=0; i<2; i++) fputs("1234567890",fp);
8    fseek(fp,-18L,SEEK_CUR);
9    fputs(" ",fp);
10   fflush(fp);          //关键是需要fflush清空缓冲区，让数据同步
11   fgets(c,20,fp);
12   c[20]='\0';
13   puts(c);
14   fclose(fp);
15 }
```

文件的基本操作：文件缓冲区的清除

- 例14-18结果
  - 程序运行结果

```
第1次运行结果是：
4567890123456789请按任意键继续……
```

  - 文件r.txt结果

```
r.txt的内容为:
12 4567890123456789
```

结果正确！

  - 除了fflush( )函数外，通常fseek( )、rewind( )、fclose( )也具备清空缓冲区的能力

文件的基本操作：文件缓冲区的清除

- 例14-18分析
  - 注意：第10行代码可以换为

```
fseek(fp, 0L, SEEK_CUR);
```

  - fseek( )函数并没有**真正移动指针位置**，但此操作把缓冲区的内容清空并同步了
  - 用scanf( )、getchar( )等从键盘输入时，也可以用fflush(stdin) **清空标准输入缓冲区**
  - 防止读入数据后遗留回车等字符对下一次读数据产生副作用

文件的基本操作：指针错误状态清除


- clearerr()函数
  - 清除由于读写操作失败引起文件输入输出缓冲区处于错误状态，以保证其后其他文件操作函数的正确使用
  - 形式

```
void clearerr(FILE *fp);
```

  - 例如键盘输入数字格式错误，无法正常读入；错误数据一直停留在缓冲区，下次再读还会遇到，继续出错
  - 通过clearerr(stdin);fflush(stdin);配合使用，不再出现死循环

本课总结

- **文件的基本概念**
  - 文本文件与二进制文件：定义、格式
  - 文件系统：缓冲文件系统和非缓冲文件系统
  - 文件类型指针
- **文件的基本操作**
  - 文件打开与关闭：文本文件和二进制文件打开/关闭方法
  - 文件的读写：文本文件字符(串)读写、二进制文件块读写
  - 文件的定位：各种文件操作函数

本课作业

- 第十四讲作业
  - 教材p.343-344习题3，6，7
  - 完成后将**word文档**或**拍照**提交到网络学堂

附加作业

- 二进制写入
  - 从键盘输入一个字符串，将其中小写字母全部转化为大写字母，并以二进制格式保存为test.dat文件
- 成绩录入
  - 有5名学生，每位学生有学号、姓名、及3门课程成绩，请将上述内容以文本格式保存为stu.txt文件
- 文件读写
  - 通过读stu.txt文件，计算出各学生的平均成绩，以及每门课程的平均成绩，将相关内容追加写入到stu.txt文件中


THANKS