文件操作

笔记本:	My Notebook		
创建时间:	2023/11/4 22:04	**更新时间:**	2023/11/7 20:48
作者:	lwb		
URL:	mk:@MSITStore:D:\bo\c语言学习\MSDN\vccore.chm::/html/_crt_sscanf.2c_.swsca...		

使用文件我们可以将数据直接存放在电脑的硬盘上，可以做到数据的持久化

1 文件
磁盘上的文件是文件，在程序设计中，文件有两种：程序文件、数据文件

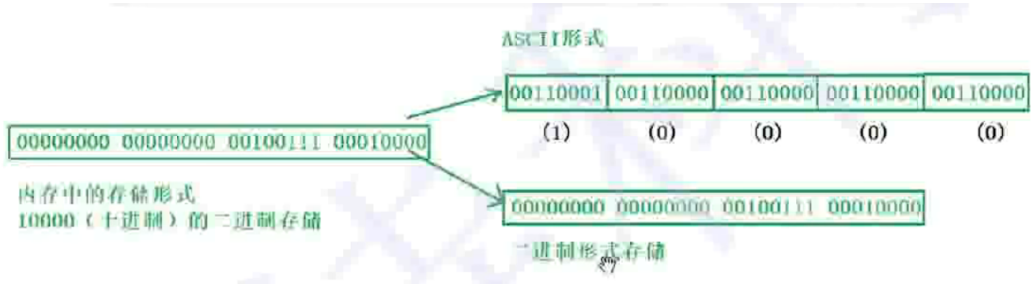
1.1 程序文件
包括源程序文件（后缀为.c），目标文件（windows环境后缀为.obj），可执行程序（windows环境后为.exe）。

1.2 数据文件
程序运行时读写的数据，比如程序运行需要从中读取数据的文件，或者输出内容的文件。
有时候我们会把信息输出到磁盘上，当需要的时候再从磁盘上把数据读取到内存中使用，这里处理的就是磁盘上的文件

1.3 文件名
一个文件要有一个唯一的文件标识，以便用户识别和引用
文件名包含三部分：文件路径+文件名主干+文件后缀
例如：c:\code\test.txt

1.4 文件类型
根据数据的组织形式，数据文件被称为文本文件或者二进制文件
数据在内存中以二进制的形式存储，如果不加转换的输出到外存，就是二进制文件，
如果要求在外存上以ASCII码的形式存储，则需要在存储前转换，以ASCII码字符的形式存储的文件就是文本文件

一个数据在内存中怎么存储?
字符一律以ASCII形式存储，数值型数据既可以用ASCII形式存储，也可以使用二进制形式存储。
如有整数10000，如果以ASCII码的形式输出到磁盘，则磁盘中占有5个字节，每个字符一个字节，而以二进制形式输出，在磁盘上只占4个字节。



文件缓冲区
ANSIC标准采用"缓冲文件系统"处理的数据文件，所谓缓存文件系统是指系统自动地在内存中为程序中每一个正在使用的文件开辟一块"文件缓冲区"。从内存向磁盘输出数据会先送到内存中的缓冲区，装满缓冲区后才一起送到磁盘上。如果从磁盘向计算机读入数据，则从磁盘文件中读取数据输入到内存缓冲区，充满缓冲区后再从缓冲区逐个地将数据送到程序数据区中，缓冲区的大小根据C编译系统决定的。

2 文件的使用

2.1文件指针

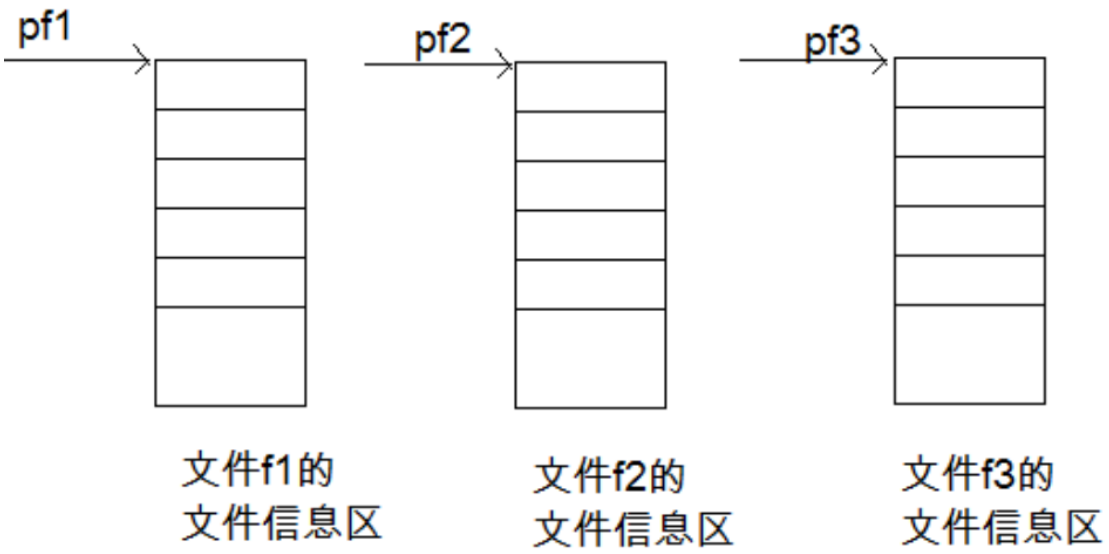
每个被使用的文件都在内存中开辟了一个相应的文件信息区，用来存放文件的相关信息(如文件的名字，文件状态及文件当前的位置等)。这些信息是保存再一个结构体变量中。该结构体类型是有系统声明的，取名为FILE

```
struct _iobuf {
       char *_ptr;
       int   _cnt;
       char *_base;
       int   _flag;
       int   _file;
       int   _charbuf;
       int   _bufsiz;
       char *_tmpfname;
      };
typedef struct _iobuf FILE;
```

每当打开一个文件的时候，系统会根据文件的情况自动创建一个FILE结构的变量，并填充其中的信息，使用者不必关心细节。
一般都是通过一个FILE的指针来委会这个FILE结构的变量，这样使用起来更加方便

```
FILE*  pf; // 文件指针变量
```

定义pf指向某个文件的文件信息区(是一个结构体变量)。通过该文件信息区中的信息就能够访问该文件；也就是说通过文件指针变量就能找到与它关联的文件



2.2 文件的打开和关闭

文件在读写之前应该先打开文件，在使用结束之后应该关闭文件
在打开文件的同时，都会返回一个FILE*的指针变量指向该文件，也相当于建立了指针和文件的关系

```
//打开文件
FILE * fopen ( const char * filename, const char * mode );
//关闭文件
int fclose ( FILE * stream );
```

文件使用方式	含义	如果指定文件不存在	
“r” (只读)	为了输入数据，打开一个已经存在的文本文件	出错	

“w”（只写）	为了输出数据，打开一个文本文件	建立一个新的文件	
“a”（追加）	向文本文件尾添加数据	建立一个新的文件	
“rb”（只读）	为了输入数据，打开一个二进制文件	出错	
wb”（只写）	为了输出数据，打开一个二进制文件	建立一个新的文件	
“ab”（追加）	向一个二进制文件尾添加数据	建立一个新的文件	
“r+”（读写）	为了读和写，打开一个文本文件	出错	
“w+”（读写）	为了读和写，建立一个新的文件	建立一个新的文件	
“a+”（读写）	打开一个文件，在文件尾进行读写	建立一个新的文件	
“rb+”（读写）	为了读和写打开一个二进制文件	出错	
“wb+”（读写）	为了读和写，新建一个新的二进制文件	建立一个新的文件	
“ab+”（读写）	打开一个二进制文件，在文件尾进行读和写	建立一个新的文件	

2.2文件的顺序读写

功能	**函数名**	**适用于**
字符输入函数	fgetc	所有输入流
字符输出函数	fputc	所有输出流
文本行输入函数	fgets	所有输入流
文本行输出函数	fputs	所有输出流
格式化输入函数	fscanf	所有输入流
格式化输出函数	fprintf	所有输出流
二进制输入	fread	文件
二进制输出	fwrite	文件

int scanf(const char * *format* [**,** *argument*]*...* **);**
int printf(const char * *format* [**,** *argument*]*...* **);**
scanf/printf 是针对标准输入流/标准输出流的格式化输入/输出语句
int fscanf(FILE * *stream*, **const char *** *format* [, *argument*]... **);**
int fprintf(FILE * *stream*, **const char *** *format* [, *argument*]...**);**

fscanf/fprintf 是针对所有输入流/所有输出流的格式化输入/输出语句
int sscanf(const char *_buffer_, const char *_format_ [, _argument_] ...);
int sprintf(char *_buffer_, const char *_format_ [, _argument_] ...);
sscanf/sprintf sscanf是从字符串中读取格式化的数据 sprintf是格式化数据输出到（存储到）字符串中

2.3 文件的随机读写
根据文件指针的位置和偏移量来定位文件指针
int fseek(FILE *_stream_, long _offset_, int _origin_);

- offset： 偏移量
- origin：SEEK_CUR - 文件指针的当前位置 SEEK_END - 文件的末尾位置 SEEK_SET- 文件起始位置

返回文件指针相对于起始位置的偏移量
long ftell(FILE *_stream_);

让文件指针的位置回到文件的起始位置
void rewind(FILE *_stream_);

2.4 文件读取结束的判定
被错误使用的feof
牢记：在文件读取过程中，不能用feof函数的返回值直接用来判断文件的是否结束
而是用于当文件读取结束的时候，判断是否是遇到文件尾结束的。
1.文本文件读取是否结束，判断返回值是否为EOF（fgetc），或者NULL（fgets）；
2.二进制文件的读取结束判断，判断返回值是否小于实际要读的个数

strerror - 把错误码对应的错误信息的字符串地址返回
perror
void perror(const char *_string_); - 会把输入的字符串和错误码对应的信息一起打印