文件操作

1 为什么使用文件

是为了将有用的信息保存下来，实现数据的持久化。比如我们以前写的程序，是在内存中运行的，程序结束后相关信息就被释放了，不能长久保存；而类似电脑的硬盘上的文件可以长期保存，即使关机也不会丢失。

2 什么是文件

一般说的文件是指磁盘上的文件。

但是在程序设计中，我们一般谈的文件有两种：程序文件和数据文件（从文件功能角度进行分类的）。

2.1 程序文件

程序文件包括源文件（后缀为.c）、目标文件（windows环境后缀.obj）和可执行文件（windows环境后缀.exe）。

2.2 数据文件

数据文件的内容不一定是程序，而是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（如test.txt），或者程序输出内容的文件（本章重点）。

以前处理数据的输入输出都是以终端为对象的，即从终端的键盘输入数据，运行结果显示到显示器上。今天我们主要讨论文件的读取和输出。

2.3 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。

文件名包括三个部分：文件路径+文件名主干+文件后缀。

如：D:\code\test.txt绝对路径

3 文件的打开与关闭

3.1 文件指针

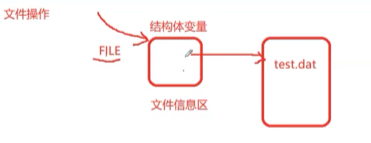
缓冲文件系统中，关键的概念是文件类型指针，简称文件指针。

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（如文件名字，文件状态及文件当前的位置等）。这些信息一直是保存在一个结构体变量中的，这个结构体变量的类型是由系统声明的，取名FILE。

例如VS2013编译环境提供的stdio.h头文件中有以下类型声明：



当我们打开一个文件时，会产生相应的文件信息区，我们一般通过文件类型的指针来维护文件信息区，就有了文件类型指针FILE\*的概念。可以知道，文件类型指针指向文件信息区，而文件信息区记录了文件的信息，也就是说，可以通过文件指针变量找到与它相关联的文件。



3.2 文件的打开与关闭

**FILE** **\*fopen(** **const** **char** **\****filename***,** **const** **char** **\****mode* **);//传入文件名和文件操作类型（读、写），系统创建文件信息区，返回FILE\*指针（返回文件信息区的地址），打开失败返回空指针。例如：**

FILE\* fp = fopen("test.txt", "r");

**文件在使用之前一定要打开文件**

#include<stdio.h>

int main()

{

//打开文件

FILE\* fp = fopen("tet.txt", "r");

//判断是否打开成功

if (fp == NULL)

{

perror("fopen");

return 1;

}

//使用文件

//关闭文件

fclose(fp);

//指针置空

fp = NULL;

return 0;

}

**文件操作模式：**



相对路径：文件和工程在同一文件夹中，只需要文件名和后缀即可访问。

绝对路径：文件的真实路径，需要文件的路径+文件名+后缀才能访问成功，注意C语言中\具有转义含义，由此在书写绝对路径时要用\\才行。

FILE\* fp = fopen("D:\\Visual Studio2019\\code

\\c\_code\\test\_10\_07\\tet.txt", "r");

4 文件的顺序读写

文件的使用：顺序读写





**注意：**文件相对于程序的输入（读取）输出（写入）的含义。

**（1）int** **fputc(** **int** *c***,** **FILE** **\****stream* **);//写/输出一个字符，到流中去。**

#include<stdio.h>

int main()

{

//打开文件

FILE\* fp = fopen("test.txt", "w");

if (fp == NULL)

{

perror("fopen");

return 1;

}

//使用文件 写入一个字符 顺序写入

fputc('l', fp);

fputc('o', fp);

fputc('v', fp);

fputc('e', fp);

//关闭文件

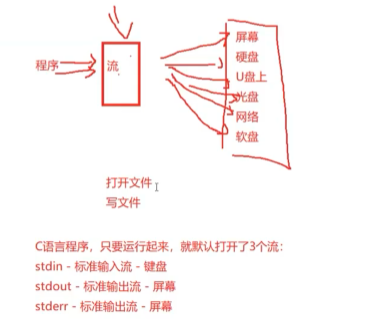
fclose(fp);

fp = NULL;

return 0;

}

**流：高度抽象的概念，程序到各种外部设备的转换机制。**



#include<stdio.h>

int main()

{

fputc('l', stdout);//标准输出流-屏幕

fputc('o', stdout);

fputc('v', stdout);

fputc('e', stdout);

fputc('\n', stdout);

return 0;

}

**（2）int** **fgetc(** **FILE** **\****stream* **);//从流中读取/输入字符，返回ASCII码**

int main()

{

fputc(fgetc(stdin), stdout);//stdin标准输入流-键盘，stdout标准输出流-屏幕

fputc(fgetc(stdin), stdout);//输入到流中，从流中获取输出

fputc(fgetc(stdin), stdout);

fputc(fgetc(stdin), stdout);

return 0;

}

**EOF文件结束标志，整型-1。**

流的字符串写入：

**（3）int** **fputs(** **const** **char** **\****string***,** **FILE** **\****stream* **);//写入一行字符串，到流信息**

#include<stdio.h>

int main()

{

FILE\* fp = fopen("test.txt", "w");

if (fp == NULL)

{

perror("fopen");

return 1;

}

fputs("hello world\n", fp);

fputs("hello", fp);

fclose(fp);

fp = NULL;

return 0;

}

**（4）char** **\*fgets(** **char** **\****string***,** **int** *n***,** **FILE** **\****stream* **);// char\*指针string指向读到的流中的n-1个字符（默认添加结束标识符\0）。**

#include<stdio.h>

int main()

{

char str[10] = { 0 };

FILE\* fp = fopen("test.txt", "r");

if (fp == NULL)

{

perror("fopen");

return 1;

}

fgets(str, 6, fp);

printf("%s\n", str);

fgets(str, 6, fp);

printf("%s\n", str);

fgets(str, 6, fp);

printf("%s\n", str);

fgets(str, 6, fp);

printf("%s\n", str);

fclose(fp);

fp = NULL;

return 0;

}

**（5）int** **fprintf(** **FILE** **\****stream***,** **const** **char** **\****format* [, *argument* ]...**);//文件格式化输出/写入函数 格式控制字符，可变参数。fscanf()函数类似**

#include<stdio.h>

int main()

{

FILE\* fp = fopen("test.txt", "w");

if (fp == NULL)

{

perror("fopen");

return 1;

}

fprintf(fp, "%s\n", "hello world");

fprintf(fp, "%s", "hello");

fclose(fp);

fp = NULL;

return 0;

}

**注：以上的函数都是以文本形式的读和写的。**

**（6）二进制形式的数据读和写**

**size\_t** **fwrite(** **const** **void** **\****buffer***,** **size\_t** *size***,** **size\_t** *count***,** **FILE** **\****stream* **);**

**size\_t** **fread(** **void** **\****buffer***,** **size\_t** *size***,** **size\_t** *count***,** **FILE** **\****stream* **);//** *buffer是一个内存地址，size是空间大小，count空间个数，stream流。*

#include<stdio.h>

struct S

{

char str[10];

int num;

float n;

};

int main()

{

struct S s = { "abcdef", 10, 5.5f };

FILE\* fp = fopen("test.txt", "w");

if (fp == NULL)

{

perror("fopen");

return 1;

}

fwrite(&s, sizeof(s), 1, fp);

fclose(fp);

fp = NULL;

return 0;

}

int main()

{

struct S s = { 0 };

FILE\* fp = fopen("test.txt", "r");

if (fp == NULL)

{

perror("fopen");

return 1;

}

fread(&s, sizeof(s), 1, fp);

printf("%s %d %f\n", s.str, s.num, s.n);

fclose(fp);

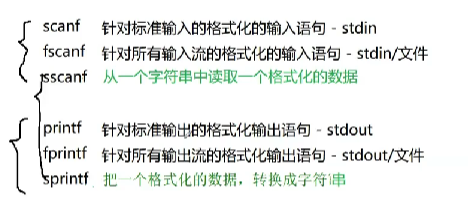
fp = NULL;

return 0;

}

**注：文件的读写学会了，修改通讯录为文件的读写形式。**

（7）对比几组函数的作用



**int** **sprintf(** **char** **\****buffer***,** **const** **char** **\****format* [**,** *argument*] ... **);//将一个格式化的数据写入到buffer内存中。**

#include<stdio.h>

struct S

{

char str[10];

int a;

float f;

};

int main()

{

struct S s = { "hello", 19, 5.5f };

char buf[100] = { 0 };

sprintf(buf, "%s %d %f", s.str, s.a, s.f);//将格式化信息写入到一块空间中

printf("%s\n", buf);

struct S tmp = { 0 };

sscanf(buf, "%s %d %f", tmp.str, &(tmp.a), &(tmp.f));//将从一块空间中读取格式化信息

printf("%s %d %f\n", tmp.str, tmp.a, tmp.f);

return 0;

}

5 文件的随机读写

5.1 fseek()函数

**int** **fseek(** **FILE** **\****stream***,** **long** *offset***,** **int** *origin* **);// 调整文件指针的位置。参数：文件流，相对于文件指针的偏移量可以为正或为负，设置文件指针的位置（起始位置、当前位置和末尾位置），整个函数会改变文件指针的位置。**

#include<stdio.h>

int main()

{

FILE\* fp = fopen("test.txt", "r");//假如文件中为abcdef

if (fp == NULL)

{

perror("fopen");

return 1;

}

char ch = fgetc(fp);//a

printf("%c\n", ch);

//调整文件指针的位置

fseek(fp, 2, SEEK\_CUR);//当前b的后面两个偏移，则指向d

ch = fgetc(fp);

printf("%c\n", ch);//d

ch = fgetc(fp);

printf("%c\n", ch);//e

//调整文件指针的位置

fseek(fp, -2, SEEK\_END);//从文件末尾向前偏移两个，则指向e

ch = fgetc(fp);

printf("%c\n", ch);//e

ch = fgetc(fp);

printf("%c\n", ch);//f

fclose(fp);

fp = NULL;

return 0;

}

5.2 ftell()函数

**long** **ftell(** **FILE** **\****stream* **);//获取文件的文件指针相对于起始位置的偏移量，返回值为偏移量。**

int ret = ftell(fp);

5.3 rewind()函数

**void** **rewind(** **FILE** **\****stream* **);//让文件指针回到起始位置**

#include<stdio.h>

int main()

{

FILE\* fp = fopen("test.txt", "r");//假如文件中为abcdef

if (fp == NULL)

{

perror("fopen");

return 1;

}

char ch = fgetc(fp);//a

printf("%c\n", ch);

//调整文件指针的位置

fseek(fp, 2, SEEK\_CUR);//当前b的后面两个偏移，则指向d

ch = fgetc(fp);

printf("%c\n", ch);//d

ch = fgetc(fp);

printf("%c\n", ch);//e

//调整文件指针的位置

fseek(fp, -2, SEEK\_END);//从文件末尾向前偏移两个，则指向e

ch = fgetc(fp);

printf("%c\n", ch);//e

ch = fgetc(fp);

printf("%c\n", ch);//f

int ret = ftell(fp);//获取文件指针相对于起始位置的偏移量

printf("%d\n", ret);//6

//让文件指针回到起始位置

rewind(fp);

ch = fgetc(fp);//a

printf("%c\n", ch);

fclose(fp);

fp = NULL;

return 0;

}

6 文本文件和二进制文件

根据数据的组织形式，数据文件被称为文本文件和二进制文件。

数据在内存中本来是以二进制形式存储的，如果不加转换的输出到外存中，则就是二进制文件。

如果要求在外存以ASCII码的形式存储，则需要在存储前进行转换，以ASCII码形式存储的文件就是文本文件。

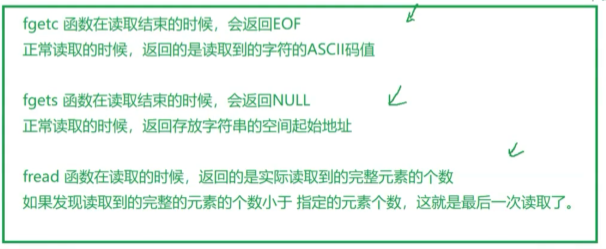
7 文件读取结束的判定

7.1 被错误使用的feof()函数

**注：**在文件读取过程中，不能用feof()函数的返回值来直接判断文件是否结束。

feof()函数的作用是：当文件读取已经结束的时候，判断文件是正常读取结束返回非零还是读取失败造成的结束，文件读取结束的原因。ferror()函数如果文件异常结束返回非零。

判断文件读取结束的方式：



例：文件内容拷贝，将一个文件读取到的内容写入到另一个文件中。

#include<stdio.h>

//把一个文件中的内容读取后写入到另一个文件中 文件拷贝

int main()

{

//打开读取文件

FILE\* pfread = fopen("test.txt", "r");

if (pfread == NULL)

{

perror("pfread");

return 1;

}

//打开写入文件

FILE\* pfwrite = fopen("test1.txt", "w");

if (pfwrite == NULL)

{

perror("pfwrite");

fclose(pfread);

pfread = NULL;

return 2;

}

char ch = 0;

//判断文件读取结束标志

while ((ch = fgetc(pfread)) != EOF)

{

fputc(ch, pfwrite);

}

if (feof(pfread))

{

printf("文件正常读取结束\n");

}

else if (ferror(pfread))

{

printf("文件读取失败结束\n");

}

//关闭文件

fclose(pfread);

pfread = NULL;

fclose(pfwrite);

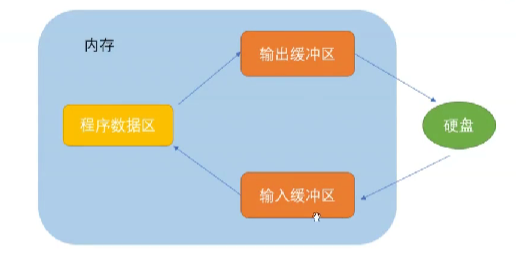
pfwrite = NULL;

return 0;

}

8 文件缓冲区

程序到硬盘，或者硬盘到程序都有一个输入/输出缓冲区的概念，为了防止操作系统不断被打断的情况。



例：下面的程序证明了缓冲区的存在

#include<stdio.h>

#include<Windows.h>

int main()

{

FILE\* pf = fopen("test.txt", "w");

if (pf == NULL)

{

perror("fopen");

return 1;

}

//写入数据，先写入缓冲区

fputs("abcdef", pf);

printf("写入数据，先写入缓冲区，睡眠十秒，可以看到文件中没有数据\n");

Sleep(10000);

fflush(pf);

printf("刷新缓冲区，则数据被写入到文件中，文件中可以看到数据，睡眠十秒关闭文件");

Sleep(10000);

fclose(pf);//关闭文件会自动刷新缓冲区，所以为了展示刷新的效果所以睡眠十秒

pf = NULL;

return 0;

}