# 文件操作

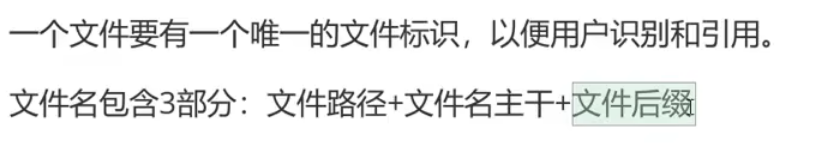
终于能文件操作辣！！

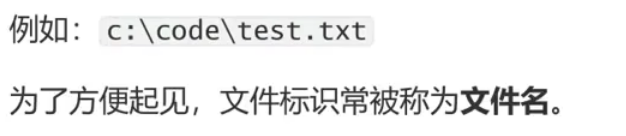
数据的持久化！

文件分为两类：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环境后缀为.exe)。

2、数据文件: 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或者输出内容的文件。

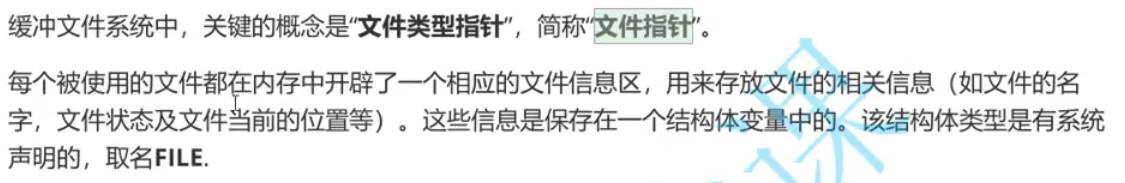
## 文件名



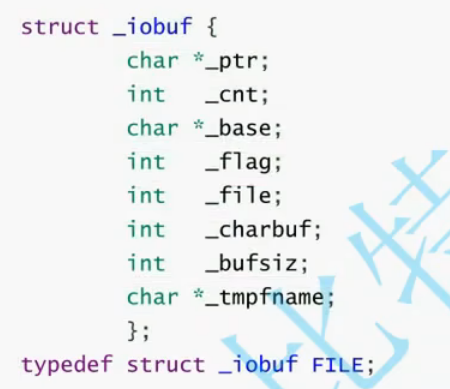


## 文件的打开和关闭

### 文件指针

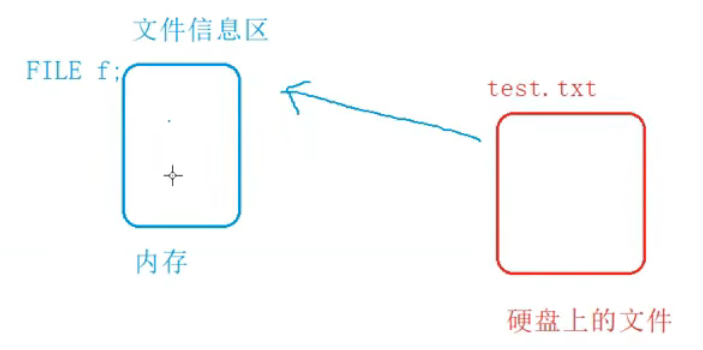


被使用的文件存放在内存中的文件信息区，文件的相关信息存放在一个结构体变量中，这个结构体在stdio.h中有定义

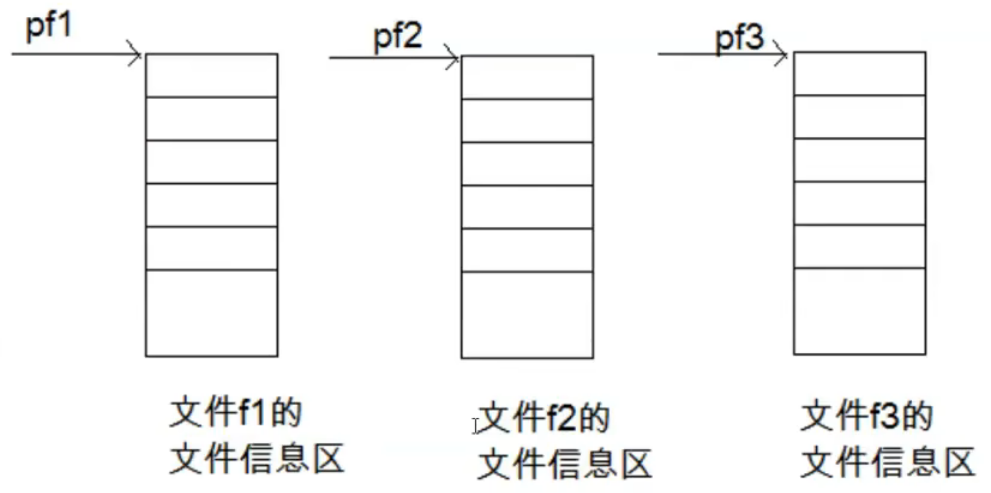


重新定义这个结构体类型为FILE

程序要去读取文件时，会在内存中开辟一个类型为自定义结构体FILE的文件信息区，用于存放该文件的信息；该结构体与改文件是高度绑定的。



那么我们用fopen打开一个文件的时候，fopen会返回一个FILE\*类型的指针，这个指针就叫做文件指针。我们就可以通过这个文件指针来对文件进行维护辣！



文件的打开与关闭是相统一的。

在编写程序的时候，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。

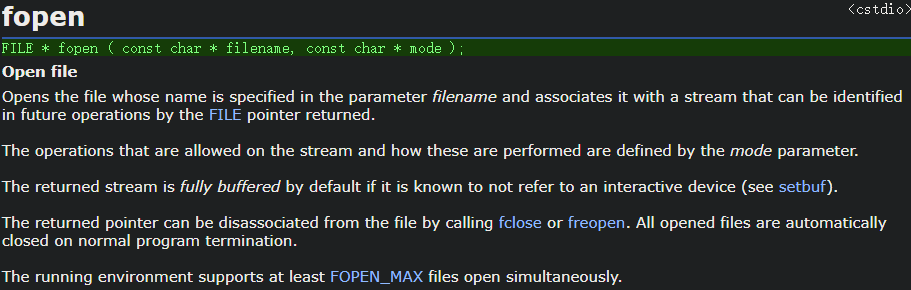
ANSIC规定使用fopen函数来打开文件，fclose来关闭文件。

以下的输入输出，都是站在程序的角度对文件而言。

fopen

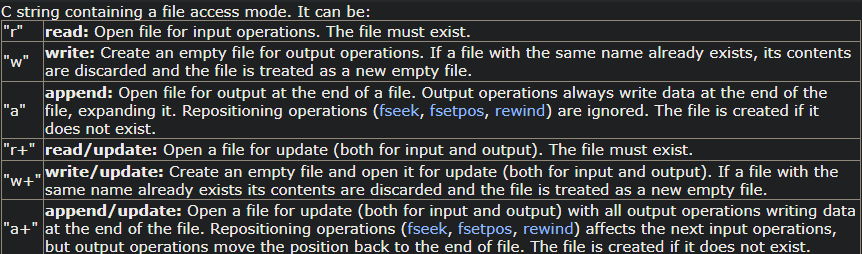
FILE \* fopen ( const char \* filename, const char \* mode )

其实两个参数就是字符串嘛。



steam 流 指的就是文件

mode：





如果文件打开失败，会返回一个空指针。、

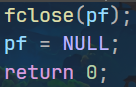


如果不给\，那就是相对路径，只会在程序当前文件夹地下去找，



此为绝对路径。

关闭文件



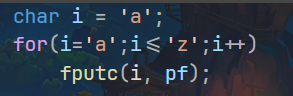
和free极为相似。

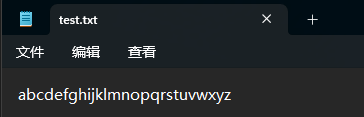
## 文件的顺序读写、



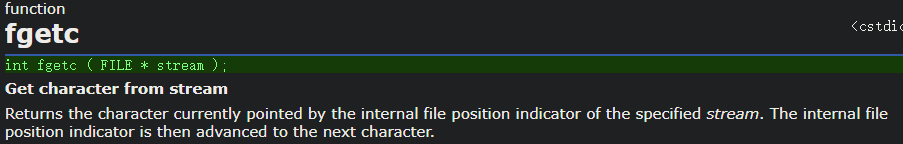
先来用fputc输出个字符

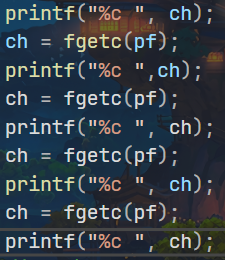






现在来读文件







我们发现，这个fgetc是会记录下读取的位置，不难想出他是记录在FILE结构体中的，



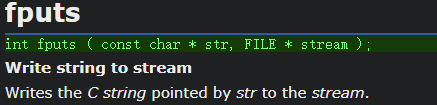
当文件读完时，fgetc会返回EOF，因此，我们可以给出循环

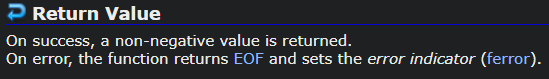


就能将文件内的字符全部读出了。

读写一行数据

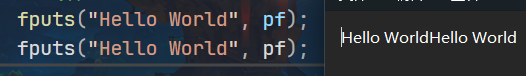
fputs





成功返回非负数 失败返回EOF

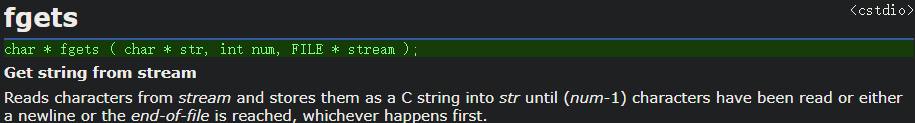
模式为w 的时候，每次写文件时都会格式化文件后再写



写两行的话并不会给我们换行，所以要加\n



**fgets**

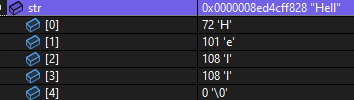




第一个参数是我们的要存放读取字符串的指针。



我们发现当我们读5个字符的时候，只能打印出四个字符。



我们在读的时候，fgets会自己算上一个\0 实际上只读了4个

perror:相当于是printf("%s\n", strerror(errno));

打印错误函数<stdio.h>



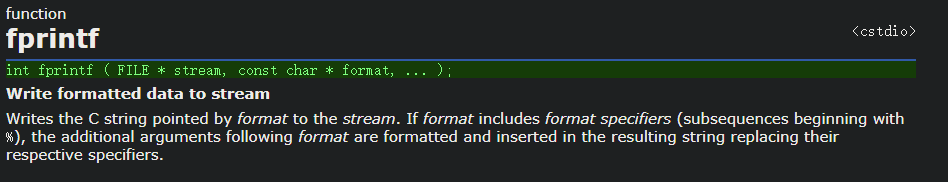
如果传入的不是空指针，那这个函数会自动给你在后面追加一个:和一个空格



但是当我们不想打印的时候，就可以只用strerror就行。

上述的输入输出函数不是作用于字符就是字符串，那我如果想输入输入各种格式呢？那我们就以结构体的形式进行写入。

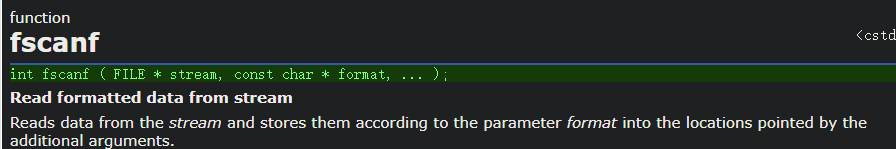
fprintf



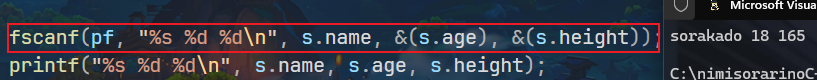
类printf，只是多了一个FILE\* 参数，相当于原来输出在终端的信息现在输出到了文件里。



fscanf



类scanf，同样需要传入FILE\* ,相当于将文件内的数据读入程序中。



stream流

我们在读写文件的时候，并不是直接就把数据和外部设备进行读写的，而是写到中间的一个中转站：流。因为不同的外部设备的读写方式是不同的，而流能够帮我们很好的与外部设备进行交流。

这大大减少了程序员的学习压力，我们只需关注流即可，而流如何将数据再写入外部设备，这是C底层帮我们实现好的。

实际上，我们要进行读写操作时，只说过打开文件、关闭文件，但是从来没说过打开屏幕，打开键盘，

这是因为，每一个C程序运行起来的时候，会默认打开三个流

FILE\* stdin:标准输入流（对应键盘）

FILE\* stdout：标准输出流（对应屏幕）

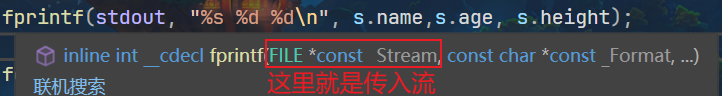
FILE\* stderr:标准错误流（对应屏幕）

这三个流都是FILE\*类型的指针

在看这张图



fprintf可以适用于所有输出流，那能不能用fprintf打印到屏幕上呢？



当然可以，只需要传入屏幕对应的流stdout！

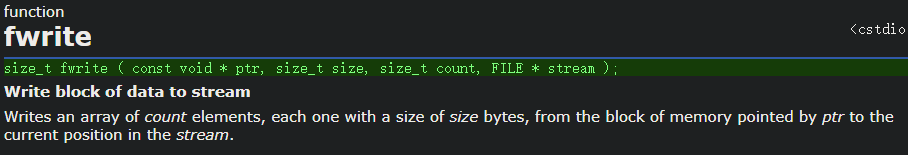
流其实就是对应的输入输出缓冲区

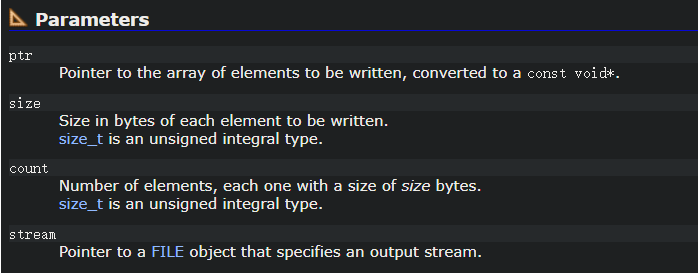
下面这两个只适用于文件流了



fwrite

二进制输入





第一个参数传的是指向要写入的元素的指针

第二个传的是写入的每个数据的大小

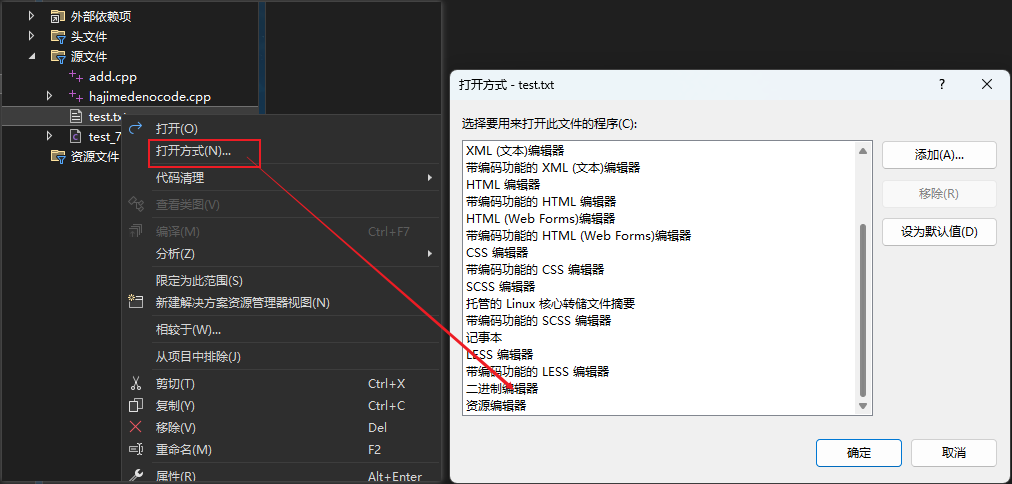
第三个传的是数据个数

第四个传的是文件流

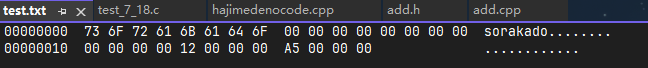


可以看到是乱码，是因为记事本的编码格式并不是二进制

我们将这个txt文件添加到源文件中，



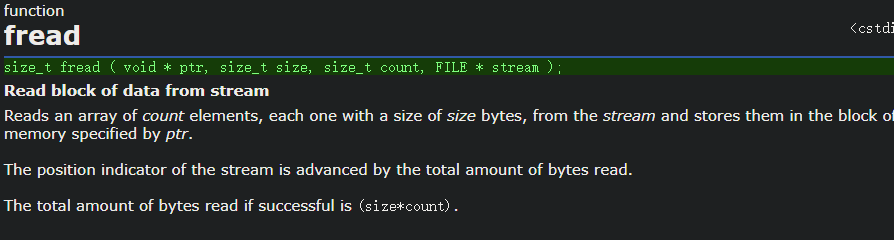
点击打开方式，选择二进制编码后



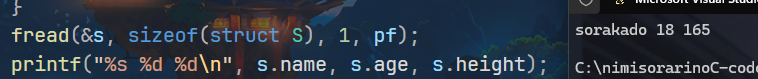
可能就能窥见其中的面貌……

实在看不懂也没关系，我们还有fread嘛~

fread



参数和fwrite一样，就是反过来从文件流读文件到ptr中



也是帮我们成功读出来了