C语言文件操作

2025年2月11日　　13:36

一、C语言中的文件是什么？

我们对文件的概念已经非常熟悉了，比如常见的 Word 文档、txt 文件、源文件等。文件是数据源的一种，最主要的作用是保存数据。

在操作系统中，为了统一对各种硬件的操作，简化接口，不同的硬件设备也都被看成一个文件。对这些文件的操作，等同于对磁盘上普通文件的操作。例如：

- 通常把显示器称为标准输出文件，printf 就是向这个文件输出数据；
- 通常把键盘称为标准输入文件，scanf 就是从这个文件读取数据。

文件	硬件设备
stdin	标准输入文件，一般指键盘；scanf()、getchar() 等函数默认从 stdin 获取输入。
stdout	标准输出文件，一般指显示器；printf()、putchar() 等函数默认向 stdout 输出数据。
stderr	标准错误文件，一般指显示器；perror() 等函数默认向 stderr 输出数据（后续会讲到）。
stdprn	标准打印文件，一般指打印机。

常见硬件设备所对应的文件

我们不去探讨硬件设备是如何被映射成文件的，大家只需要记住，在C语言中硬件设备可以看成文件，有些输入输出函数不需要你指明到底读写哪个文件，系统已经为它们设置了默认的文件，当然你也可以更改，例如让 printf 向磁盘上的文件输出数据。

操作文件的正确流程为：打开文件 --> 读写文件 --> 关闭文件。文件在进行读写操作之前要先打开，使用完毕要关闭。

所谓打开文件，就是获取文件的有关信息，例如文件名、文件状态、当前读写位置等，这些信息会被保存到一个 FILE 类型的结构体变量中。关闭文件就是断开与文件之间的联系，释放结构体变量，同时禁止再对该文件进行操作。

在C语言中，文件有多种读写方式，可以一个字符一个字符地读取，也可以读取一整行，还可以读取若干个字节。文件的读写位置也非常灵活，可以从文件开头读取，也可以从中间位置读取。

文件流

在《载入内存，让程序运行起来》一文中提到，所有的文件（保存在磁盘）都要载入内存才能处理，所有的数据必须写入文件（磁盘）才不会丢失。数据在文件和内存之间传递的过程叫做文件流，类似水从一个地方流动到另一个地方。数据从文件复制到内存的过程叫做输入流，从内存保存到文件的过程叫做输出流。

文件是数据源的一种，除了文件，还有数据库、网络、键盘等；数据传递到内存也就是保存到C语言的变量（例如整数、字符串、数组、缓冲区等）。我们把数据在数据源和程序（内存）之间传递的过程叫做数据流(Data Stream)。相应的，数据从数据源到程序（内存）的过程叫做输入流(Input Stream)，从程序（内存）到数据源的过程叫做输出流(Output Stream)。

输入输出（Input output，IO）是指程序（内存）与外部设备（键盘、显示器、磁盘、其他计算机等）进行交互的操作。几乎所有的程序都有输入与输出操作，如从键盘上读取数据，从本地或网络上的文件读取数据或写入数据等。通过输入和输出操作可以从外界接收信息，或者是把信息传递给外界。

我们可以说，打开文件就是打开了一个流。

来自 <https://c.biancheng.net/view/2053.html>

二、C语言fopen函数的用法，C语言打开文件详解

在C语言中，操作文件之前必须先打开文件；所谓“打开文件”，就是让程序和文件建立连接的过程。

打开文件之后，程序可以得到文件的相关信息，例如大小、类型、权限、创建者、更新时间等。在后续读写文件的过程中，程序还可以记录当前读写到了哪个位置，下次可以在此基础上继续操作。

标准输入文件 stdin（表示键盘）、标准输出文件 stdout（表示显示器）、标准错误文件 stderr（表示显示器）是由系统打开的，可直接使用。

使用 <stdio.h> 头文件中的 fopen() 函数即可打开文件，它的用法为：

```
FILE *fopen(char *filename, char *mode);
```

filename为文件名（包括文件路径），mode为打开方式，它们都是字符串。

fopen() 函数的返回值

fopen() 会获取文件信息，包括文件名、文件状态、当前读写位置等，并将这些信息保存到一个 FILE 类型的结构体变量中，然后将该变量的地址返回。

FILE 是 <stdio.h> 头文件中的一个结构体，它专门用来保存文件信息。我们不用关心 FILE 的具体结构，只需要知道它的用法就行。

如果希望接收 fopen() 的返回值，就需要定义一个 FILE 类型的指针。例如：

```
FILE *fp = fopen("demo.txt", "r");
```

表示以“只读”方式打开当前目录下的 demo.txt 文件，并使 fp 指向该文件，这样就可以通过 fp 来操作 demo.txt 了。fp 通常被称为文件指针。

再来看一个例子：

```
FILE *fp = fopen("D:\\demo.txt","rb+");
```

表示以二进制方式打开 D 盘下的 demo.txt 文件，允许读和写。

判断文件是否打开成功

打开文件出错时，fopen() 将返回一个空指针，也就是 NULL，我们可以利用这一点来判断文件是否打开成功，请看下面的代码：

```
1.  FILE *fp;
2.  if( (fp=fopen("D:\\demo.txt","rb")) == NULL ){
3.    printf("Fail to open file!\n");
4.    exit(0);  //退出程序（结束程序）
5.  }
```

我们通过判断 fopen() 的返回值是否和 NULL 相等来判断是否打开失败：如果 fopen() 的返回值为 NULL，那么 fp 的值也为 NULL，此时 if 的判断条件成立，表示文件打开失败。

以上代码是文件操作的规范写法，读者在打开文件时一定要判断文件是否打开成功，因为一旦打开失败，后续操作就都没法进行了，往往以“结束程序”告终。

fopen() 函数的打开方式

不同的操作需要不同的文件权限。例如，只想读取文件中的数据的话，“只读”权限就够了；既想读取又想写入数据的话，“读写”权限就是必须的了。

另外，文件也有不同的类型，按照数据的存储方式可以分为二进制文件和文本文件，它们的操作细节是不同的。

在调用 fopen() 函数时，这些信息都必须提供，称为“文件打开方式”。最基本的文件打开方式有以下几种：

控制读写权限的字符串（必须指明）

打开方式	说明
"r"	以“只读”方式打开文件。只允许读取，不允许写入。文件必须存在，否则打开失败。
"w"	以“写入”方式打开文件。如果文件不存在，那么创建一个新文件；如果文件存在，那么清空文件内容（相当于删除原文件，再创建一个新文件）。
"a"	以“追加”方式打开文件。如果文件不存在，那么创建一个新文件；如果文件存在，那么将写入的数据追加到文件的末尾（文件原有的内容保留）。
"r+"	以“读写”方式打开文件。既可以读取也可以写入，也就是随意更新文件。文件必须存在，否则打开失败。
"w+"	以“写入/更新”方式打开文件，相当于w和r+叠加的效果。既可以读取也可以写入，也就是随意更新文件。如果文件不存在，那么创建一个新文件；如果文件存在，那么清空文件内容（相当于删除原文件，再创建一个新文件）。
"a+"	以“追加/更新”方式打开文件，相当于a和r+叠加的效果。既可以读取也可以写入，也就是随意更新文件。如果文件不存在，那么创建一个新文件；如果文件存在，那么将写入的数据追加到文件的末尾（文件原有的内容保留）。

控制读写方式的字符串（可以不写）

打开方式	说明
"t"	文本文件。如果不写，默认为"t"。
"b"	二进制文件。

调用 fopen() 函数时必须指明读写权限，但是可以不指明读写方式（此时默认为"t"）。

读写权限和读写方式可以组合使用，但是必须将读写方式放在读写权限的中间或者尾部（换句话说，不能将读写方式放在读写权限的开头）。例如：

- 将读写方式放在读写权限的末尾："rb"、"wt"、"ab"、"r+b"、"w+t"、"a+t"
- 将读写方式放在读写权限的中间："rb+"、"wt+"、"ab+"

整体来说，文件打开方式由 r、w、a、t、b、+ 六个字符拼成，各字符的含义是：

- r(read)：读
- w(write)：写

- a(append)：追加
- t(text)：文本文件
- b(binary)：二进制文件
- +：读和写

关闭文件

文件一旦使用完毕，应该用 fclose() 函数把文件关闭，以释放相关资源，避免数据丢失。

fclose() 的用法为：int fclose(FILE *fp);

fp 为文件指针。例如：`fclose(fp);`

文件正常关闭时，fclose() 的返回值为0，如果返回非零值则表示有错误发生。

实例演示

最后，我们通过一段完整的代码来演示 fopen 函数的用法，这个例子会一行一行地读取文本文件的所有内容：

```
#include <stdio.h>
#include <stdlib.h>
#define N 100
int main() {
    FILE *fp;
    char str[N + 1];
    //判断文件是否打开失败
    if ( (fp = fopen("d:\\demo.txt", "rt")) == NULL ) {
        puts("Fail to open file!");
        exit(0);
    }
    //循环读取文件的每一行数据
    while( fgets(str, N, fp) != NULL ) {
        printf("%s", str);
    }
    //操作结束后关闭文件
    fclose(fp);
    return 0;
}
```

来自 <https://c.biancheng.net/view/2054.html>