**C语言的fopen函数（文件操作/读写）**



[意念586](https://blog.csdn.net/u011392772) 2016-07-13 10:13:54  2843  收藏 8

分类专栏： [文件系统](https://blog.csdn.net/u011392772/category_6300788.html)

头文件：#include <stdio.h>  
  
fopen()是一个常用的函数，用来以指定的方式打开文件，其原型为：  
    FILE \* fopen(const char \* path, const char \* mode);  
  
【参数】path为包含了路径的文件名，mode为文件打开方式。  
  
mode有以下几种方式：

|  |  |
| --- | --- |
| **打开方式** | **说明** |
| r | 以只读方式打开文件，该文件必须存在。 |
| r+ | 以读/写方式打开文件，该文件必须存在。 |
| rb+ | 以读/写方式打开一个二进制文件，只允许读/写数据。 |
| rt+ | 以读/写方式打开一个文本文件，允许读和写。 |
| w | 打开只写文件，若文件存在则长度清为0，即该文件内容消失，若不存在则创建该文件。 |
| w+ | 打开可读/写文件，若文件存在则文件长度清为零，即该文件内容会消失。若文件不存在则建立该文件。 |
| a | 以附加的方式打开只写文件。若文件不存在，则会建立该文件，如果文件存在，写入的数据会被加到文件尾，即文件原先的内容会被保留（EOF符保留)。 |
| a+ | 以附加方式打开可读/写的文件。若文件不存在，则会建立该文件，如果文件存在，则写入的数据会被加到文件尾后，即文件原先的内容会被保留（原来的EOF符 不保留)。 |
| wb | 以只写方式打开或新建一个二进制文件，只允许写数据。 |
| wb+ | 以读/写方式打开或建立一个二进制文件，允许读和写。 |
| wt+ | 以读/写方式打开或建立一个文本文件，允许读写。 |
| at+ | 以读/写方式打开一个文本文件，允许读或在文本末追加数据。 |
| ab+ | 以读/写方式打开一个二进制文件，允许读或在文件末追加数据。 |

在POSIX 系统，包含Linux 下都会忽略 b 字符。由fopen()所建立的新文件会具有S\_IRUSR|S\_IWUSR|S\_IRGRP|S\_IWGRP|S\_IROTH|S\_IWOTH(0666)权限，此文件权限也会参考umask 值。  
  
二进制和文本模式的区别：

* 在windows系统中，文本模式下，文件以"\r\n"代表换行。若以文本模式打开文件，并用fputs等函数写入换行符"\n"时，函数会自动在"\n"前面加上"\r"。即实际写入文件的是"\r\n" 。
* 在类Unix/Linux系统中文本模式下，文件以"\n"代表换行。所以Linux系统中在文本模式和二进制模式下并无区别。

更多信息请查看：[C语言fopen()打开文本文件与二进制文件的区别](http://c.biancheng.net/cpp/html/2423.html)  
  
有些C编译系统可能不完全提供所有这些功能，有的C版本不用"r+","w+","a+",而用"rw","wr","ar"等，读者注意所用系统的规定。  
  
【返回值】文件顺利打开后，指向该流的文件指针就会被返回。若果文件打开失败则返回NULL，并把错误代码存在errno 中。  
  
注意：一般而言，开文件后会作一些文件读取或写入的动作，若开文件失败，接下来的读写动作也无法顺利进行，所以在fopen()后请作错误判断及处理。  
  
文件操作完成后，需要将文件关闭，一定要注意，否则会造成文件所占用内存泄露和在下次访问文件时出现问题。  
  
文件关闭后，需要将文件指针指向空，这样做会防止出现游离指针，而对整个工程造成不必要的麻烦，如fp = NULL。  
  
【实例】打开一个文件然后关闭该文件。

* 1. #include<stdio.h>
  2. #include<string.h>
  3. #include<stdlib.h>
  4. int main()
  5. {
  6. FILE\* fstream;
  7. char msg[100] = "Hello!I have read this file.";
  8. fstream=fopen("test.txt","at+");
  9. if(fstream==NULL)
  10. {
  11. printf("open file test.txt failed!\n");
  12. exit(1);
  13. }
  14. else
  15. {
  16. printf("open file test.txt succeed!\n");
  17. }
  18. fclose(fstream);
  19. return 0;
  20. }