# 附录1：文件常用函数说明

**在实际应用中，我们往往需要对文件进行操作，下面我将介绍C语言的一些关于操作文件的函数。**

****一、计算机文件****

    计算机文件是以计算机硬盘为载体存储在计算机上的信息集合，是存储在某种长期储存设备上的一段数据流。在C语言中用一个指针变量指向一个文件，这个指针称为文件指针。通过文件指针就可对它所指的文件进行各种操作，C语言规定该文件指针类型为FILE型。

FILE文件指针结构体定义：

typedef struct

{

short level; // 缓冲区“满”或“空”的程度

unsigned flags; // 文件状态标志

char fd; // 文件描述符

unsigned char hold; // 如缓冲区无内容不读取字符

short bsize; // 缓冲区的大小

unsigned char \*buffer; // 数据缓冲区的位置

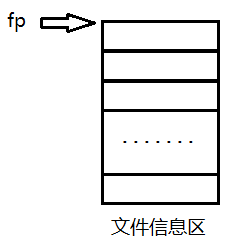
unsigned ar \*curp; // 指针当前的指向

unsigned istemp; // 临时文件指示器

short token; // 用于有效性检查

}FILE;

例如我们定义一个文件指针：FILE \*fp;    fp指向某一个文件的文件信息区（是一个结构体变量），通过该文件信息区就能够访问该文件。如下图所示：



文件缓冲区：缓冲文件系统是指系统自动地在内存区为程序中每一个正在使用的文件开辟一个文件缓冲区。从内存向磁盘输出数据必须先送到内存中的缓冲区，装满缓冲区后才一起送到磁盘中去。

****二、打开与关闭文件****

* **fopen 函数说明：**

**作用**：用来打开一个文件

**格式**：FILE \* fopen(const char \* path,const char \* mode);

**返回值**：打开文件成功返回一个文件指针，若打开文件失败则返回NULL

**参数说明**：

    \*path：字符串包含欲打开的文件路径及文件名(例如：D:\\A.txt)，注意在C语言中欲输出一个"\"则需要输两个"\\"

    \*mode：代表着流形态，mode有下列几种形态字符串：

                r      打开只读文件，该文件必须存在。

                r+    打开可读写的文件，该文件必须存在。

                rb+  读写打开一个二进制文件，只允许读写数据。

                rt+   读写打开一个文本文件，允许读和写。

                w      打开只写文件，若文件存在则文件长度清为0，即该文件内容会消失。若文件不存在则建立该文件。

                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，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    只写打开或新建一个二进制文件；只允许写数据。

                wb+  读写打开或建立一个二进制文件，允许读和写。

                wt+   读写打开或着建立一个文本文件；允许读写。

                at+    读写打开一个文本文件，允许读或在文本末追加数据。

                ab+   读写打开一个二进制文件，允许读或在文件末追加数据。

                上述的形态字符串都可以再加一个b字符，如rb、w+b或ab＋等组合，加入b 字符用来告诉函数库打开的文件为二进制文件，而非纯文字文件。

* **fclose 函数说明：**

**作用**：关闭一个文件流，释放文件指针

**格式**：int fclose( FILE \*fp );

**返回值**：如果流成功关闭，fclose 返回 0，否则返回EOF

**参数说明**：

    \*fp：需要关闭的文件指针

注：在文件操作完成后我们应该调用该函数来关闭文件，如果不关闭文件将可能会丢失数据。因为在向文件写入数据时会先将数据输出到缓冲区，待缓冲区充满后才正式输出给文件。另外，不关闭文件造成文件被占用，其它程序无法打开和调用。

**三、顺序读写数据文件**

* **fgetc 函数说明：**

**作用**：从文件指针指向的文件流中读取一个字符，读取一个字节后，光标位置后移一个字节

**格式**：int fgetc(FILE \*stream);

**返回值**：返回所读取的一个字节，如果读到文件末尾或者读取出错时返回EOF(EOF是文件结束标识符,一般值为-1)

**参数说明**：

    \*stream：文件指针，从该文件指针指向的文件中读取一个字符，然后将光标后移一个字节

* **fputc 函数说明：**

**作用**：将指定字符写到文件指针所指向的文件的当前写指针位置上

**格式**：int fputc (char c, File \*fp);

**返回值**：在正常调用情况下，函数返回写入文件的字符的ASCII码值，出错时，返回EOF

**参数说明**：

    c：需要写入的字符

    \*fp：文件指针，在当前文件指针所指向的文件的当前写指针位置上写入一个字符c，然后文件内部写指针会自动后移一个字节位置

* **fgets 函数说明：**

**作用**：从文件结构体指针stream中读取数据，每次读取一行。读取的数据保存在buf指向的字符数组中，每次最多读取bufsize-1个字符（第bufsize个字符赋'\0'），如果文件中的该行，不足bufsize个字符，则读完该行就结束。如若该行（包括最后一个换行符）的字符数超过bufsize-1，则fgets只返回一个不完整的行，但是，缓冲区总是以NULL字符结尾，对fgets的下一次调用会继续读该行。

**格式**：char \*fgets(char \*buf, int bufsize, FILE \*stream);

返回值：函数成功将返回buf，失败或读到文件结尾返回NULL。因此我们不能直接通过fgets的返回值来判断函数是否是出错而终止的，应该借助feof函数或者ferror函数来判断。

**参数说明**：

    \*buf: 字符型指针，指向用来存储所得数据的地址。

    bufsize: 整型数据，指明存储数据的大小。

    \*stream: 文件指针，将要读取的文件流。

* **fputs 函数说明：**

**作用**：向指定的文件写入一个字符串（不自动写入字符串结束标记符‘\0’）

**格式**：int fputs(char \*str, FILE \*fp);

**返回值**：若成功返回0，失败返回EOF

**参数说明**：

    \*str: 需要写入的字符串

    \*fp: 文件指针，将要写入的文件流

* **fprintf 函数说明：**

**作用**：格式化后输出到文件中

**格式**：int fprintf (FILE\*  stream, const char\*  format, [argument]);

**返回值**：若成功返回值是输出的字符数，当发生错误时返回一个负值

**参数说明**：

    \*stream：文件指针

    \*format：输出格式

    [argument]：附加参数列表

注：用法与printf函数类似，这里只是参数多了一个文件指针，将格式后的结果输出到文件中

* **fscanf 函数说明：**

**作用**：从一个流中执行格式化输入

**格式**：int fscanf(FILE\* stream,constchar\* format,[argument]);

**返回值**：成功返回读入的参数的个数，失败返回EOF。

**参数说明**：

    \*stream：文件指针

    \*format：格式字符串

    [argument]：输入列表

注：用法与scanf函数类似

* **fread 函数说明：**

**作用**：从一个文件流中读数据，最多读取count个元素，每个元素size字节

**格式**：size\_t fread ( void \*buffer, size\_t  size, size\_t  count, FILE \*stream) ;

**返回值**：如果调用成功返回实际读取到的元素个数，如果不成功或读到文件末尾返回0

**参数说明**：

    \*buffer：用于接收数据的内存地址

    size：要读的每个数据项的字节数，单位是字节

    count：要读count个数据项，每个数据项size个字节

    \*stream：文件指针  
注：这个函数以二进制形式对文件进行操作，不局限于文本文件

* **fwrite 函数说明：**

**作用**：向文件写入一个数据块

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

**返回值**：返回实际写入的数据块数目

**参数说明**：

    \*buffer：待写入的数据的内存地址

    size：要写入内容的单字节数

    count：要进行写入size字节的数据项的个数

    \*stream：文件指针

注：这个函数以二进制形式对文件进行操作，不局限于文本文件

**四、随机读写数据文件**

* **rewind 函数说明：**

**作用**：将文件内部的位置指针重新指向一个流（数据流/文件）的开头

**格式**：void rewind(FILE \*stream);

**返回值**：无

**参数说明**：

    \*stream：文件指针

* **ftell 函数说明：**

**作用**：得到文件位置指针当前位置相对于文件首的偏移字节数（测定文件位置标记的当前位置）

**格式**：long ftell(FILE \*stream);

**返回值**：成功返回当前文件位置，失败返回-1L

**参数说明**：

    \*stream：文件指针

注：因为ftell返回long型，根据long型的取值范围-2^31~2^31-1（-2147483648～2147483647），故对大于2.1G的文件进行操作时出错。

* **fseek 函数说明：**

**作用**：重定位流(数据流/文件)上的文件内部位置指针

**格式**：int fseek(FILE \*stream, long offset, int origin);

**返回值**：成功返回0，失败返回非0值

**参数说明**：

    \*stream：文件指针

    offset：偏移量，正数表示正向偏移，负数表示负向偏移。因为是long型数据，所以应在数字后面加一个字母L

    origin：设定从文件的哪里开始偏移,可取值为：SEEK\_CUR、 SEEK\_END 或 SEEK\_SET，详见下表：

|  |  |  |
| --- | --- | --- |
| 名字 | 起始点 | 用数字代表 |
| SEEK\_SET | 文件开始位置 | 0 |
| SEEK\_CUR | 文件当前位置 | 1 |
| SEEK\_END | 文件末尾位置 | 2 |

示例：

fseek(fp,100L,0); // 把位置指针移动到离文件开头100字节处

fseek(fp,100L,1); // 把位置指针移动到离文件当前位置100字节处

fseek(fp,-100L,2); // 把位置指针退回到离文件结尾100字节处

**五、文件读写的出错检测**

* **ferror 函数说明：**

**作用**：在调用各种输入输出函数（如 putc、getc、fread、fwrite等）时，如果出现错误，除了函数返回值有所反映外，还可以用ferror函数检查

**格式**：int ferror(FILE \*stream);

**返回值**：返回0表示未出错，返回非0值表示出错

**参数说明**：

    \*stream：文件指针

* **clearerr 函数说明：**
* **作用**：使文件错误标志和文件结束标志置为0。假设在调用一个输入输出函数时出现了错误，ferror函数值为一个非零值。在调用clearerr（fp）后，ferror（fp）的值变为0。只要出现错误标志，就一直保留，直到对同一文件调用clearerr函数或rewind函数，或任何其他一个输入输出函数。

**格式**：void clearerr(FILE \*stream);

**返回值**：无

**参数说明**：

    \*stream：文件指针