根据数据的组织形式，数据⽂件被称为⽂本⽂件或者⼆进制⽂件。

数据在内存中以⼆进制的形式存储，如果不加转换的输出到外存的⽂件中，就是⼆进制⽂件。

如果要求在外存上以ASCII码的形式存储，则需要在存储前转换。以ASCII字符的形式存储的⽂件就是文本文件。

⼀个数据在⽂件中是怎么存储的呢？ 字符⼀律以ASCII形式存储，数值型数据既可以⽤ASCII形式存储，也可以使⽤⼆进制形式存储。

如有整数10000，如果以ASCII码的形式输出到磁盘，则磁盘中占⽤5个字节（每个字符⼀个字节），而二进制形式输出，则在磁盘上只占4个字节。

而当我们在文件中打开记事本查看时，对于文本文件显示如下：

而对于二进制文件则是乱码，人的肉眼无法从中直接获取信息：

二、文件的打开与关闭

2.1流与标准流

2.1.1流

我们程序的数据需要输出到各种外部设备，也需要从外部设备获取数据，不同的外部设备的输⼊输出 操作各不相同，为了⽅便程序员对各种设备进⾏⽅便的操作，我们抽象出了流的概念，我们可以把流 想象成流淌着字符的河。

C程序针对⽂件、画⾯、键盘等的数据输⼊输出操作都是通过流操作的。 ⼀般情况下，我们要想向流⾥写数据，或者从流中读取数据，都是要打开流，然后操作。

其实简单来想，通俗一点的来说流就像我们用的水桶，你要给他装水时把盖子打开，装满或不用时将其关闭（水桶没盖子的情况不讨论）

2.1.2标准流

C语⾔程序在启动的时候，默认打开了3个流：

• stdin - 标准输⼊流，在⼤多数的环境中从键盘输⼊，scanf函数就是从标准输⼊流中读取数据。

• stdout - 标准输出流，⼤多数的环境中输出⾄显⽰器界⾯，printf函数就是将信息输出到标准输出 流中。

• stderr - 标准错误流，⼤多数环境中输出到显⽰器界⾯。 这是默认打开了这三个流，我们使⽤scanf、printf等函数就可以直接进⾏输⼊输出操作的。 stdin、stdout、stderr 三个流的类型是： FILE \* ，通常称为⽂件指针。

所以，在c语言中，我们需要使用FILE\*的指针来对流进行各种维护操作。

2.2文件的打开与关闭

在了解文件的打开与关闭之前，让我们先来了解一下只读与只写：

只读与只写的定义是：

只读：这种状态下的文件或对象允许用户读取其内容，但不允许用户对其进行修改或删除。这意味着用户可以查看文件的内容，但不能对其进行任何更改。在文件系统中，只读权限通常通过设置文件权限或修改文件属性来实现。这种权限的目的是保护文件内容不被意外修改，确保数据的完整性和安全性。

只写：与只读相反，只写状态的文件或对象允许用户对其进行修改或删除，但不允许用户读取其内容。这表示用户可以更改或删除文件，但不能查看文件的具体内容。这种权限通常用于需要保密的修改操作，例如在特定情况下允许用户对文件进行编辑，但不允许查看其原始内容。

以博主自己的了解，在我们的c语言编程过程中，只读其实就是给你一个文件，我们可以通过程序来观察文件里面的内容，而不可以对文件进行修改。其次，只写恰好相反，它允许我们读取文件中的内容，但不允许我们对文件的内容进行修改。

那么在c语言中，我们有以下几种方式改变对文件的打开方式，具体如下图所示：

其中与w有关的打开方式如果在源文件中有内容时，它会将文件中的内容清空在执行程序中的操作）

使用他们的代码实例将会在三中与顺序读写函数结合到一起介绍（比较重要的为前三个，所以在三中我们会着重讲解前三个的使用，其实前三个弄明白了，后面的自己鼓捣一会也就能掌握有半八九十了）

三、顺序读写主要函数介绍以及使用

主要的顺序读写函数如下：

上⾯说的适⽤于所有输⼊流⼀般指适⽤于标准输⼊流和其他输⼊流（如⽂件输⼊流）；所有输出流⼀ 般指适⽤于标准输出流和其他输出流（如⽂件输出流）。接下来，我们将结合2.2中的文件打开方式来介绍以上函数。

3.1fgetc函数（与r打开方式结合介绍）

在c++官网上，它的定义如下：

int fgetc ( FILE \* stream );

它返回指定流的内部文件位置指示器当前指向的字符。然后，内部文件位置指示器将前进到下一个字符。这里的流就是我们要打开的文件，若它在读取时已经到了文件的末尾，将返回EOF和文件的末尾指示器（feof),如果读取的文件中没有内容，它返回的也是EOF，但会设置其错误指示器（ferror）。

首先，该函数是读取文件，所以我们要读取文件中的内容，这里我们用文本文件，所以选用r示例代码如下：

int main()

{

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

if (fp == NULL)

{

perror("fopen");

return 1;

}

char c = fgetc(fp);

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

fclose(fp);

fp = NULL;

return 0;

}

（文件中的内容为abcdefghijklmnopqrstuvwxyz）

他的运行结果为：

如果我们用for循环：

int main()

{

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

if (fp == NULL)

{

perror("fopen");

return 1;

}

char c = '0';

for (; c != EOF;)

{

c = fgetc(fp);

printf("%c ", c);

}

fclose(fp);

fp = NULL;

return 0;

}

它的运行结果如下：

可以看到每次执行fgets函数时，它在文件中的指针就会向后移动一格。

那如果文件读取失败呢（这里我们用perror来输出错误信息）：

3.2fputc函数(与w,a打开方式结合介绍）

在c++官网上，它的定义如下：

int fputc ( int character, FILE \* stream );

这里是对文件的写入，如果写入成功则返回写入的字符如果写入失败则返回EOF并设置错误提示器（ferror)既然是要写入，那么这里我们应使用w的方式来打开文件：

int main()

{

FILE\* fp = fopen("tex1.txt", "w");

if (fp == NULL)

{

perror("fopen");

return 1;

}

char c = '0';

for (c = 'a';c <= 'z';c++)

{

fputc(c,fp);

printf("%c ", c);

}

fclose(fp);

fp = NULL;

return 0;

}

这时让我们来看文件中的内容：

那如果我们想要继续向文件中输入内容（就是在打开之前文件中已经有内容时而我们不想删除它里面原有的内容）这时我们的打开方式应为a,让我们在次运行上面代码（将其里面的w改为a)此时文件中的内容如下：

3.3fgets函数与fputs函数

其实在经过上面fputc和fgets函数的介绍后，我们从函数的字面意思上其实就不难猜出他们的作用---从文件中获取字符，向文件中输入字符，所以这里我们把他们放到一起介绍。

首先让我们来看c++官网给出的他们的定义：

fgets函数：

char \* fgets ( char \* str, int num, FILE \* stream );

官网的定义是从流中读取字符，并将它们作为 C 字符串存储到 str 中，直到读取 （num-1） 个字符或到达换行符或文件末尾，以先发生者为准。

需要特别注意换行符使 fgets 停止读取，但它被函数视为有效字符，并包含在复制到 str 的字符串中。

终止 null（'\0')字符会自动附加到 str 的字符之后。

同时fgets 与 gets 有很大不同：fgets 不仅接受流参数，还允许指定 str 的最大大小（这里就是告诉fgets函数我的字符串函数最大的容量为多少）并在字符串中包含任何结束换行符。

此外fgets函数如果读取成功，返回的是str首元素的地址，如果读取错误，返回空指针并设置错误提示符（需要注意的是此时的str里面的内容可能已经被更改）。

这里文件的内容为：

让我们来用一串代码来解释：

int main()

{

FILE\* fp = fopen("tex1.txt", "r");

if (fp == NULL)

{

perror("fopen");

return 1;

}

char str[50] = { 0 };

fgets(str, 50, fp);

puts(str);

fclose(fp);

fp = NULL;

return 0;

}

输出的结果为：

可以见得，当文件内容读取完的话，它会自动在字符串末尾加上'\0'（即便没有初始化结果也一样）

fputs函数：

C++官网给出的定义为：

int fputs ( const char \* str, FILE \* stream );

将 str 指向的c字符串写入流。该函数从指定的地址 （str） 开始复制，直到到达终止 null 字符 （'\0'）。此终止 null 字符不会复制到流中。

fputs不仅与puts的不同之处在于可以指定目标流，而且fputs不会写入其他字符，而puts会自动在末尾附加一个换行符。

除此之外，如果该函数输入成功，返回的是一个非负值，如果输入失败则返回EOF并设置错误提示符。

多说无益，直接上代码解释：

int main()

{

FILE\* fp = fopen("tex2.txt", "r+");

if (fp == NULL)

{

perror("fopen");

return 1;

}

int c = fputs("dsadsa", fp);

fclose(fp);

fp = NULL;

return 0;

}

运行的结果如下：

文件中的内容为：

可见他是直接输入，不需要借用第三者字符串函数来协助输入字符到文件中。

我们下篇文章见。

————————————————

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

原文链接：https://blog.csdn.net/xiuwoaiailixiya/article/details/139327239