C 输入 & 输出 C 预处理器

C 文件读写

上一章我们讲解了 C 语言处理的标准输入和输出设备。本章我们将介绍 C 程序员如何创建、打开、关闭文本文件或二进制文件。

一个文件，无论它是文本文件还是二进制文件，都是代表了一系列的字节。C 语言不仅提供了访问顶层的函数，也提供了底层（OS）调用来处理存储设备上的文件。本章将讲解文件管理的重要调用。

打开文件

您可以使用 **fopen( )** 函数来创建一个新的文件或者打开一个已有的文件，这个调用会初始化类型 **FILE** 的一个对象，类型 **FILE** 包含了所有用来控制流的必要的信息。下面是这个函数调用的原型：

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

在这里，**filename** 是字符串，用来命名文件，访问模式 **mode** 的值可以是下列值中的一个：

模式	描述
r	打开一个已有的文本文件，允许读取文件。
w	打开一个文本文件，允许写入文件。如果文件不存在，则会创建一个新文件。在这里，您的程序会从文件的开头写入内容。如果文件存在，则该会被截断为零长度，重新写入。
a	打开一个文本文件，以追加模式写入文件。如果文件不存在，则会创建一个新文件。在这里，您的程序会在已有的文件内容中追加内容。
r+	打开一个文本文件，允许读写文件。
w+	打开一个文本文件，允许读写文件。如果文件已存在，则文件会被截断为零长度，如果文件不存在，则会创建一个新文件。
a+	打开一个文本文件，允许读写文件。如果文件不存在，则会创建一个新文件。读取会从文件的开头开始，写入则只能是追加模式。

如果处理的是二进制文件，则需使用下面的访问模式来取代上面的访问模式：

```
"rb", "wb", "ab", "rb+", "r+b", "wb+", "w+b", "ab+", "a+b"
```

关闭文件

为了关闭文件，请使用 fclose( ) 函数。函数的原型如下：

```
int fclose( FILE *fp );
```

如果成功关闭文件，**fclose( )** 函数返回零，如果关闭文件时发生错误，函数返回 **EOF**。这个函数实际上，会清空缓冲区中的数据，关闭文件，并释放用于该文件的所有内存。EOF 是一个定义在头文件 **stdio.h** 中的常量。

C 标准库提供了各种函数来按字符或者以固定长度字符串的形式读写文件。

写入文件

下面是把字符写入到流中的最简单的函数：

```
int fputc( int c, FILE *fp );
```

函数 **fputc()** 把参数 c 的字符值写入到 fp 所指向的输出流中。如果写入成功，它会返回写入的字符，如果发生错误，则会返回 **EOF**。您可以使用下面的函数来把一个以 null 结尾的字符串写入到流中：

```
int fputs( const char *s, FILE *fp );
```

函数 **fputs()** 把字符串 **s** 写入到 fp 所指向的输出流中。如果写入成功，它会返回一个非负值，如果发生错误，则会返回 **EOF**。您也可以使用 **int fprintf(FILE *fp,const char *format, ...)** 函数来写把一个字符串写入到文件中。尝试下面的实例：

> ***注意：****请确保您有可用的* ***tmp*** *目录，如果不存在该目录，则需要在您的计算机上先创建该目录。*
>
> `/tmp` *一般是 Linux 系统上的临时目录，如果你在 Windows 系统上运行，则需要修改为本地环境中已存在的目录，例如:* `C:\tmp`、`D:\tmp` *等。*

实例

```
#include <stdio.h>
int main()
{
FILE *fp = NULL;
fp = fopen("/tmp/test.txt", "w+");
fprintf(fp, "This is testing for fprintf...\n");
fputs("This is testing for fputs...\n", fp);
fclose(fp);
}
```

当上面的代码被编译和执行时，它会在 /tmp 目录中创建一个新的文件 **test.txt**，并使用两个不同的函数写入两行。接下来让我们来读取这个文件。

读取文件

下面是从文件读取单个字符的最简单的函数：

```
int fgetc( FILE * fp );
```

fgetc() 函数从 fp 所指向的输入文件中读取一个字符。返回值是读取的字符，如果发生错误则返回 **EOF**。下面的函数允许您从流中读取一个字符串：

```
char *fgets( char *buf, int n, FILE *fp );
```

函数 **fgets()** 从 fp 所指向的输入流中读取 n - 1 个字符。它会把读取的字符串复制到缓冲区 **buf**，并在最后追加一个 **null** 字符来终止字符串。

如果这个函数在读取最后一个字符之前就遇到一个换行符 '\n' 或文件的末尾 EOF，则只会返回读取到的字符，包括换行符。您也可以使用 **int fscanf(FILE *fp, const char *format, ...)** 函数来从文件中读取字符串，但是在遇到第一个空格字符时，它会停止读取。

实例

```
#include <stdio.h>
int main()
{
FILE *fp = NULL;
char buff[255];
fp = fopen("/tmp/test.txt", "r");
fscanf(fp, "%s", buff);
printf("1: %s\n", buff );
fgets(buff, 255, (FILE*)fp);
printf("2: %s\n", buff );
fgets(buff, 255, (FILE*)fp);
printf("3: %s\n", buff );
fclose(fp);
}
```

当上面的代码被编译和执行时，它会读取上一部分创建的文件，产生下列结果：

```
1: This
2: is testing for fprintf...

3: This is testing for fputs...
```

首先，**fscanf()** 方法只读取了 **This**，因为它在后边遇到了一个空格。其次，调用 **fgets()** 读取剩余的部分，直到行尾。最后，调用 **fgets()** 完整地读取第二行。

二进制 I/O 函数

下面两个函数用于二进制输入和输出：

```
size_t fread(void *ptr, size_t size_of_elements,
size_t number_of_elements, FILE *a_file);
size_t fwrite(const void *ptr, size_t size_of_elements,
size_t number_of_elements, FILE *a_file);
```

这两个函数都是用于存储块的读写 - 通常是数组或结构体。

C 输入 & 输出 C 预处理器

2 篇笔记 写笔记

fseek 可以移动文件指针到指定位置读,或插入写:

```
int fseek(FILE *stream, long offset, int whence);
```

fseek 设置当前读写点到 offset 处, whence 可以是 SEEK_SET,SEEK_CUR,SEEK_END 这些值决定是从文件头、当前点和文件尾计算偏移量 offset。

你可以定义一个文件指针 `FILE *fp`,当你打开一个文件时，文件指针指向开头，你要指到多少个字节，只要控制偏移量就好，例如, 相对当前位置往后移动一个字节：`fseek(fp,1,SEEK_CUR);` 中间的值就是偏移量。 如果你要往前移动一个字节，直接改为负值就可以：`fseek(fp,-1,SEEK_CUR)`。

执行以下实例前，确保当前目录下 `test.txt` 文件已创建：

```
#include <stdio.h>

int main(){
    FILE *fp = NULL;
    fp = fopen("test.txt", "r+");  // 确保 test.txt 文件已创建
    fprintf(fp, "This is testing for fprintf...\n");
    fseek(fp, 10, SEEK_SET);
    if (fputc(65,fp) == EOF) {
        printf("fputc fail");
    }
    fclose(fp);
}
```

执行结束后，打开 test.txt 文件：

```
This is teAting for fprintf...
```

注意： 只有用 `r+` 模式打开文件才能插入内容，`w` 或 `w+` 模式都会清空掉原来文件的内容再来写，`a` 或 `a+` 模式即总会在文件最尾添加内容，哪怕用 fseek() 移动了文件指针位置。

GGLLCC 2年前 (2017-08-22)

在 Visual Studio 2015 开发环境中运行下列代码, 会提示 fopen 函数不安全:

```
#include <stdio.h>
int main() {
    // 申明文件指针
    FILE *filePointer = NULL;
    // 以追加模式打开文件 test.txt, 其中路径参数用正斜杠("/")
    filePointer = fopen("C:/Users/Administrator/Desktop/test.txt", "a+");
    // 在test.txt文件末尾追加写入
```

```
    fputs("The text was added to the file by executing these codes.", filepointer);
    // 关闭文件
    fclose(filepointer);
    return 0;
}
```

错误提示:

错误 C4996 'fopen': This function or variable may be unsafe. Consider using fopen_s instead. To disable deprecation, use _CRT_SECURE_NO_WARNINGS. See online help for details.

解决办法，在代码首行写上:

```
#define _CRT_SECURE_NO_WARNINGS
```

定义宏忽略警告, 即可忽略安全警告, 生成运行。

Kazin 2周前 (03-05)