FILE operation

> 在<stdio.h>中

1. fopen()

```
FILE *a = fopen("***.txt","w");//后需附加代码 验证是否打开成功
```

- 1. 打开当前目录下的文件 只用输入文件名 否则要输入文件的绝对路径

> 输入的是字符串

2. 打开方式 常见的有“r”,“w”,“a”,“rb”--后有详解

> "r"(只读),"w"(只写),"a"(文件末尾追加),"rb"(二进制件打开，只读),"wb"(二进制件打开，只读),"ab"(二进制件打开，追加)等。

> **用"r"方式打开文件，若文件不存在，则返回一个空指针表示错误。**若用"w"或"a"打开文件，则若文件不存在，都会创建一个新文件，即使文件存在，写入时也会把原有内容先进行覆盖

---

2. fclose()

> * 在对文件操作后需要即时关闭文件以保存与释放资源

```
FILE *a = fopen("test.txt","r");
//.................
fclose(a);
```

- *若fopen函数打开失败，则返回空指针，且设置全局变量 errno 来标识错误，因此打开文件后应该做必要的判断。对于flose函数，如果成功返回值为零。如果失败，则返回 EOF*

3.字符输入函数-fgetc()

- fgetc函数返回文件指针指向的字符，并使指针向下一个字符偏移。若想用此函数打印文件全部内容，可用while((ch=fgetc(pf))!=EOF)循环来控制，这是因为当指针指向文件结尾，函数返回值为-1

```
int main ()
{
    FILE *pf = fopen("test.txt","r");
    if(!pf) return 0;
    else {
        char ch = fgetc(pf);//fgetc一次只读取一个字符;
        printf("%c\n",ch);
        ch = fgetc(pf);
        printf("%c\n",ch);
    }
    fclose(pf);//关闭文件
    return 0;
}// test.txt -> "abcd"
 //输出 >> a '\n' b
```

4. 字符输出函数-fputc() / 文本行输入函数-fputs()

> 向第二个参数输出

- fputc() 的用法为：
  int fputc (int ch, FILE *fp);
  ch 为要写入的字符，fp 为文件指针。fputc() 写入成功时返回写入的字符，失败时返回 EOF，返回值类型为 int 也是为了容纳这个负数。
  例如：fputc('a', fp); 或者：char ch = 'a'; fputc(ch, fp);
  表示把字符 'a' 写入fp所指向的文件中。
- .fputs() 函数用来向指定的文件写入一个字符串，
  它的用法为:
  **int fputs(char *str, FILE fp);*
  str 为要写入的字符串，fp 为文件指针。写入成功返回非负数，失败返回 EOF

```c
int main()//fputc
{
    FILE *pf = fopen("test.txt","w");//因为要向文本输出数据 所以“w“打开
    if(!pf) return 0;
    else {
        fputc('c',pf);//向pf指向的文件输出字符‘c’
    }
    fclose(pf);
    return 0;
}//运行后 文件“test.txt“ >> c
```

```c
int main()//fputs
{
    //.........//
    char *str = "http://c.biancheng.net";
    FILE *fp = fopen("D:\\demo.txt", "at+");
    fputs(str, fp);
    //.........//
}///表示把把字符串 str 写入到 D:\\demo.txt 文件中
```

5.文本行输入函数-fgets()

> 题外话
> 在输入时通常会使用gets()函数 **但gets函数不检查预留出存区是否能够容纳实际输入的数据**
> 如果输入的字符数目大于数组的长度,就会发生内存越界，*所以编程时建议使用 fgets()*

- fgets(char *s, int size, FILE *stream)
    - s 代表要保存到的内存空间的首地址，可以是字符数组名，也可以是指向字符数组的字符指针变量名
    - size 代表的是读取字符串的长度
      fgets的中间size参数直接赋值字符数组长度即可，不是n-1
    - 可以是文件指针 也可以是标准输入流stdin /* 键盘 */
    - 注意！fgets函数从缓冲区/文件读入的‘\n’会存入 s 中
      不会想gets函数一样丢弃

```
    # include <stdio.h>
    int main(void)
    {
    char str[20]; /*定义一个最大长度为19, 末尾是'\0'的字符数组来存储字符串*/
    printf("请输入一个字符串:");
    FILE *a = fopen("test.txt","w");
    fgets(str, 7, a); /*从输入流 a(test.txt文件)即输入缓冲区中读取7个字符到字符数组str中*/
    printf("%s\n", str);
    return 0;
    }
//输出结果是:
// 请输入一个字符串:i love you
// i love
```

我们发现输入的是“i love you”，而输出只有“i love”。原因是 fgets() 只指定了读取 7 个字符放到字符数组 st

---

6.二进制输入-fread()

> C 库函数 size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream) 从给定流 stream 读取数据到 ptr 所指向的数组中。

- fread(void *ptr, size_t size, size_t nmemb, FILE *stream)
    - ptr – 这是指向带有最小尺寸size*nmemb 字节的内存块的指针。
    - size – 这是要读取的每个元素的大小，以字节为单位。
    - nmemb – 这是元素的个数，每个元素的大小为 size 字节。
    - stream – 这是指向 FILE 对象的指针，该 FILE 对象指定了一个输入流。
- int nRead = fread(buf, sizeof(char), size, fp);
  int nRead = fread(buf, sizeof(char)*size, 1, fp);
  两种写法均可

```
# include <stdio.h>
<p class="mume-header " id="include-stdioh"></p>

# include <string.h>
<p class="mume-header " id="include-stringh"></p>


int main()
{
   FILE *fp;
   char c[] = "This is runoob";
   char buffer[20];

   /* 打开文件用于读写 */
   fp = fopen("file.txt", "w+");
   /* 写入数据到文件 */
   fwrite(c, strlen(c) + 1, sizeof(char), fp);
   /* 查找文件的开头 */
   fseek(fp, 0, SEEK_SET);
   /* 读取并显示数据 */
   fread(buffer, strlen(c)+1, sizeof(char), fp);    <-------
   printf("%s\n", buffer);
   fclose(fp);

   return(0);
}
```

7.格式化输入函数-fscanf()

> fscanf()与scanf有相同的用法
> fscanf()函数是格式化读写函数 读取的对象是磁盘文件

fscanf()函数

fscanf(FILE * fp,char * format,...);

> - fp为文件指针
> - format为C字符串
> - ...为参数列表
> - 返回值为成功写入的字符的个

- fsanf()函数会从文件输入流中读入数据 储存到format中 遇到空格和换行时结束
    - 如果要忽略换行的影响 则使用___ **fscanf(fp,"%[^\n]")**
      其中的“_ **%[\n]** _”表示读取字符直到遇到\n为止.
    - 如果要将换行符读取掉，但不储存到变量中，可以采用 **fscanf(fp, “%[^\n]%c”, test);*

```
/* fscanf example */
# include <stdio.h>
<p class="mume-header " id="include-stdioh-1"></p>


int main ()
{
  char str [80];
  float f;
  FILE * pFile;

  pFile = fopen ("myfile.txt","w+");
  fprintf (pFile, "%f %s", 3.1416, "PI");
  rewind (pFile);//将文件指针重新指向文件开头
  fscanf (pFile, "%s", str);
  fclose (pFile);
  printf ("I have read: %f and %s \n",f,str);
  return 0;
}
此示例代码创建一个名为myfile.txt的文件，并向其写入一个浮点数和一个字符串。 然后，流被重新卷绕，并且两个值都
I have read: 3.141600 and PI
```

8.格式化输出函数-fprintf()

> fprintf() 和 printf()十分相像
> 都是格式化输出 但fprintf比printf()函数多一个FILE参数

- **fprintf(FILE *fp,char * format,...);***

    - fp为文件指针
    - format为C字符串
    - ...为参数列表

```
# include<stdio.h>
<p class="mume-header " id="includestdioh"></p>

# define N 2
<p class="mume-header " id="define-n-2"></p>

struct stu{
char name[10];
int num;
int age;
float score;
} boya[N], boyb[N], *pa, *pb;
int main(){
FILE *fp;
int i;
pa=boya;
pb=boyb;
if( (fp=fopen("D:\\demo.txt","wt+")) == NULL ){
puts("Fail to open file!");
exit(0);
}
//从键盘读入数据，保存到boya
printf("Input data:\n");
for(i=0; i<N; i++,pa++){
scanf("%s %d %d %f", pa->name, &pa->num, &pa->age, &pa->score);
}
pa = boya;
//将boya中的数据写入到文件
for(i=0; i<N; i++,pa++){
fprintf(fp,"%s %d %d %f\n", pa->name, pa->num, pa->age, pa->score);
}
//重置文件指针
rewind(fp);
//从文件中读取数据，保存到boyb
for(i=0; i<N; i++,pb++){
fscanf(fp, "%s %d %d %f\n", pb->name, &pb->num, &pb->age, &pb->score);
}
pb=boyb;
//将boyb中的数据输出到显示器
for(i=0; i<N; i++,pb++){
printf("%s %d %d %f\n", pb->name, pb->num, pb->age, pb->score);
}
fclose(fp);
return 0;
}
```

9.fwrite()函数//输出函数?

> 向指定的文件中写入若干数据块

- **fwrite(const void* buffer, size_t size, size_t count, FILE* stream);**

> - buffer是一个指针 指向fwirte()获取数据的地址
> - size 要写入内容的单节字节数;
> - count 要写入块的个数;
> - stream 文件指针

> - ***返回实际写入数据块的个数***

fseek对此函数有作用，但是fwrite 函数写到用户空间缓冲区，并未同步到文件中，所以修改后要将内存与文件同步可以用fflush（FILE *fp）函数同步。

10. fflush()函数

> 更新缓存区。

- **flush(FILE * stream);**
  调用fflush()会将缓冲区中的内容写到stream所指的文件中去.若stream为NULL，则会将所有打开的文件进行数据更新

> fflush(stdin)：刷新缓冲区,将缓冲区内的数据清空并丢弃。
> fflush(stdout)：刷新缓冲区,将缓冲区内的数据输出到设备。

9.ftell()函数

- 返回文件位置指针相对启始位置的偏移量
  - 单位（/字节）

```
{
    /*................*/
    FILE *pf= fopen("test.txt","r");
    char ch ;
    ch = fgetc(pf);
    printf("%ld",ftell(pf));//1
    ch = fgetc(pf);
    printf("%ld",ftell(pf));//2
    ch = fgetc(pf);
    printf("%ld",ftell(pf));//3
    /*................*/
}
```
以"r"的方式打开文件，位置指针初始时在文首，偏移量为0，而每进行一次字符读写，偏移量便加一。 ftell函数能反映

10.rewind()

- 使文件指针回到文件开头

```
{
    /*................*/
    FILE *pf= fopen("test.txt","r");
    char ch ;
    ch = fgetc(pf);
    printf("%ld",ftell(pf));//1
    ch = fgetc(pf);
    printf("%ld",ftell(pf));//2
    rewind(pf);
    printf("%ld",ftell(pf));//0
    ch = fgetc(pf);
    printf("%ld",ftell(pf));//1
    /*................*/
}
```
使用rewind函数后，偏移量归零

11.fseek()

- 根据文件指针的位置和偏移量来定位文件指针。

int fseek(FILE *stream, long offset, int fromwhere);

- stream 为文件指针
- offset 为偏移量 正数是正向偏移量 负数逆向偏移量
- 第三个参数设定从文件的哪里开始偏移
- ◦ SEEK_SET : 文件开头
  SEEK_CUR : 当前位置
  SEEK_END : 文件结尾
  SEEK_SET , SEEK_CUR和SEEK_END和依次为0，1和2

如果执行失败(比如offset超过文件自身大小)，则不改变stream指向的位置。返回值: 成功，返回0，否则返回其他值。

- 简言之：
  fseek(fp,100L,0);把fp指针移动到离文件开头100字节处；
  fseek(fp,100L,1);把fp指针移动到离文件当前位置100字节处
  ffseek(fp,-100L,2)；把fp指针退回到离文件结尾100字节处;

```
# include <stdio.h>
<p class="mume-header " id="include-stdioh-2"></p>

# define N 5
<p class="mume-header " id="define-n-5"></p>

typedef struct student {
 long sno;
 char name[10];
 float score[3];
} STU;
void fun(char *filename, STU n)
{
 FILE *fp;
 fp = fopen("filename", "rb+");
 fseek(fp, -1L*sizeof(STU),SEEK_END);
fwrite(&n, sizeof(STU), 1, fp);
fclose(fp);
}
void main()
{
  STU t[N]={
    {10001,"MaChao", 91, 92, 77},
    {10002,"CaoKai", 75, 60, 88},
    {10003,"LiSi", 85, 70, 78},
    {10004,"FangFang", 90, 82, 87},
    {10005,"ZhangSan", 95, 80, 88}
    };
  STU n={10006,"ZhaoSi", 55, 70, 68}, ss[N];
  int i,j; FILE *fp;
  fp = fopen("student.dat", "wb");
  fwrite(t, sizeof(STU), N, fp);
  fclose(fp);
  fp = fopen("student.dat", "rb");
  fread(ss, sizeof(STU), N, fp);
  fclose(fp);
  printf("\nThe original data :\n\n");
  for (j=0; j<N; j++)
  {
   printf("\nNo: %ld Name: %-8s Scores: ",ss[j].sno, ss[j].name);
   for (i=0; i<3; i++)
   printf("%6.2f ", ss[j].score[i]);
   printf("\n");
}
fun("student.dat", n);
printf("\nThe data after modifing :\n\n");
```

```
fp = fopen("student.dat", "rb");
fread(ss, sizeof(STU), N, fp);
fclose(fp);
for (j=0; j<N; j++)
{
 printf("\nNo: %ld Name: %-8s Scores: ",ss[j].sno, ss[j].name);
 for (i=0; i<3; i++)
 printf("%6.2f ", ss[j].score[i]);
 printf("\n");
}
```

文件读取结束的判定

1. ferror()函数-判断是否由于发生错误而结束读取
2. feof函数-判断是否遇到文件结束而结束读取

```
{
FILE * pf = fopen("asdf.txt","w");
if(!pf)
{
    printf("%s",strerror(errno));
    return 0;
}
else{
    char ch;
    while((ch = fgetc(pf))!=EOF)
    {
        /*............*/;
    }
}
if(ferror(pf))
    printf("I/O error when reading");
if(feof(pf)) printf("End of file readed successfully");
}
```

ferror()若读取错误而结束，该函数返回一个非零值，否则返回一个零值。
如图以"w"(只写)方式进行读取操作，显然由于错误而结束读取文件。

> 补充 EOF 在文章中的位置

> C和C ++提供了不同的文件操作功能。 我们可以使用EOF值来检查文件的结尾，该文件可以用来检查不同函数的返回值。 EOF存储-1 ，到达文件末尾时文件操作函数将返回-1 。

```
# include <stdio.h>
<p class="mume-header " id="include-stdioh-3"></p>

int main()
{
   FILE *fp = fopen("myfile.txt", "r");
   int ch = getc(fp);

   //Check enf of file and if not end execute while block
   //File EOF reached end while loop
   while (ch != EOF)
   {
    /* Print the file content with ch */
    putchar(ch);
        /* Read one character from file */
    ch = getc(fp);
   }

   if (feof(fp))
    printf("\n End of file reached.");
   else
    printf("\n Something went wrong.");
   fclose(fp);

   getchar();

   return 0;
}
```

下面是一个文件读入程序代码

```
/*************************************************************************
概要：图片读取与存储测试
参考：https://ask.csdn.net/questions/206408
          https://blog.csdn.net/yhl_leo/article/details/50782792
目的：将一张图片用二进制格式读取出来，然后传输，并存储在一个新文件
日期：2019.9.26
作者：maohuifei
*************************************************************************/

# include<stdio.h>
<p class="mume-header " id="includestdioh-1"></p>

# include<string.h>
<p class="mume-header " id="includestringh"></p>


/*************************************************************************
* 摘    要：将缓冲区数据写入文件
* 参    数：
                buf：
                filename：
                size：长度
* 当前版本：1.0.0
* 作    者：maoge
* 日    期：2019.09.07
* 备    注：
*************************************************************************/
void WriteFromStream(char * buf, char * filename, unsigned int  size)
{
        FILE * f = fopen(filename, "wb+");
        if (f)
        {
                fwrite(buf, 1, size, f);
                fclose(f);
        }
}

/*************************************************************************
* 摘    要：读取文件到缓冲区
* 参    数：
                buf：
                filename：
                size：
* 当前版本：1.0.0
* 作    者：maoge
* 日    期：2019.09.07
```

```
* 备    注:
*********************************************************************/
void ReadToStream(char * filename)
{
        char new_file[30] = "";//文件名
        FILE * f = fopen(filename, "rb");
        if (f)
        {
                //下面
                fseek(f, 0, SEEK_END);
                int size = ftell(f);//返回给定流stream的当前文件位置。一般作用是获取文件大小，以定义
                char * buf = new char[size];
                fseek(f, 0, SEEK_SET);
                memset(buf, 0, size);//赋初值0
                int nRead = fread(buf, sizeof(char), size, f);
                printf("size=%d\n ", size);
                printf("nRead=%d\n ",nRead);
                //int nRead = fread(buf, 1, size, f);//sizeof(char)值就是1
                fclose(f);
                if (nRead > 0)
                {
                        //将二进制流打印成16进制字符串
                        for (unsigned int i = 0; i < nRead; i++)
                        {
                                printf("%02X ", (unsigned char)buf[i]);
                                if (i % 16 == 15)
                                {
                                        printf("\n");
                                }
                        }
                }


                sprintf(new_file, "new3_%s", filename);
                WriteFromStream(buf, new_file, size);
                delete buf;
        }
}


int main()
{
        ReadToStream("03.jpg");

        return 0;
```

```
}
```