

## 一、文件的概念

### 1.1 文件的定义

### 1.2 磁盘文件的分类

## 二、文件指针

## 三、打开文件fopen

## 四、关闭文件fclose

## 五、一次读写一个字符

### 5.1 fgetc

### 5.2 fputc

## 六、一次读写一个字符串

### 6.1 fgets

### 6.2 fputs

## 七、读文件fread

## 八、写文件fwrite

## 九、格式化读写文件函数

## 十、随机读写

### 10.1 rewind

### 10.2 ftell

### 10.3 fseek

# 一、文件的概念

文件用来存放程序、文档、音频、视频数据、图片等数据的。

文件就是存放在磁盘上的，一些数据的集合。

在windows下可以通过写字板或记事本打开文本文件对文件进行编辑保存。写字板和记事本是微软程序员写的程序，对文件进行打开、显示、读写、关闭。  
作为一个程序员，必须掌握编程实现创建、写入、读取文件等操作

对文件的操作是经常要用到的知识，比如：写飞秋软件传送文件等

## 1.1 文件的定义

**磁盘文件：**（我们通常认识的文件）

指一组相关数据的有序集合,通常存储在外部介质(如磁盘)上，使用时才调入内存。

**设备文件：**

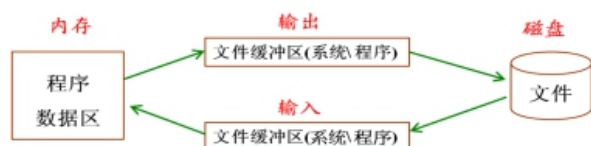
在操作系统中把每一个与主机相连的输入、输出设备看作是一个文件，把它们的输入、输出等同于对磁盘文件的读和写。

键盘：标准输入文件      屏幕：标准输出文件

其它设备：打印机、触摸屏、摄像头、音箱等

在Linux操作系统中，每一个外部设备都在/dev目录下对应着一个设备文件，咱们在程序中要想操作设备，就必须对与其对应的/dev下的设备文件进行操作。

### 标准io库函数对磁盘文件的读取特点



文件缓冲区是库函数申请的一段内存，由库函数对其进行操作，程序员没有必要知道存放在哪里，只需要知道对文件操作的时候的一些缓冲特点即可。

### 1、行缓冲

**标准io库函数，往标准输出（屏幕）输出东西的时候是行缓冲的**  
所谓的行缓冲就是缓冲区碰到换行符的时候才刷新缓冲区

如果不刷新缓冲区，无法对文件执行读写操作

行缓冲的刷新条件：

```
1 #include <stdio.h>
```

```

2
3 int main(int argc, char const *argv[])
4 {
5     //由于printf函数是一个标准io，所以只有刷新缓冲区才可以将数据输出到终端
6     //printf("hello world");
7
8     //刷新缓冲区方法1：使用\n
9     //printf("hello world\n");
10
11     //刷新缓冲区方法2：程序正常结束
12     //printf("hello world");
13     //return 0;
14
15     //刷新缓冲区方法3：使用fflush函数刷新缓冲区
16     //printf("hello world");
17     //fflush：刷新函数。可以刷新指定的缓冲区
18     //stdout：标准输出，就是对终端进行写操作
19     //fflush(stdout);
20
21     //刷新缓冲区方法4：当缓冲区满的时候自动刷新
22     //默认行缓冲的大小为1024个字节
23     int i;
24     for(i = 1; i < 300; i++)
25     {
26         printf("%03d ", i);
27     }
28
29     while(1)
30     ;
31
32     return 0;
33 }

```

## 2、全缓冲

标准io库函数，往普通文件读写数据的，是全缓冲的，碰到换行符也不刷新缓冲区，即缓冲区满了，才刷新缓冲区

### 刷新缓冲区的情况

- 1.缓冲区满了，刷新缓冲区
- 2.人为刷新缓冲区 fflush(文件指针)

3.程序正常结束 会刷新缓冲区

### 3.无缓冲

在读写文件的时候通过系统调用io ( read write ) ,对文件进行读写数据  
这个时候是无缓冲的，即写数据会立马进入文件，读数据会立马进入内存

### 写文件的流程

应用程序空间-à内核空间 -à驱动程序--à硬盘上

应用程序和内核程序运行在不同的空间里，目的是为了保护内核。

### 设置缓冲区的目的

通过缓冲可以减少进出内核的次数，以提高效率。

## 1.2 磁盘文件的分类

一个文件通常是磁盘上一段命名的存储区

计算机的存储在物理上是二进制的，

所以物理上所有的磁盘文件本质上都是一样的：以字节为单位进行顺序存储

从用户或者操作系统使用的角度（逻辑上）把文件分为：

**文本文件**：基于字符编码的文件

**二进制文件**：基于值编码的文件

### 文本文件

基于字符编码，常见编码有ASCII、UNICODE等

一般可以使用文本编辑器直接打开

例如：数5678的以ASCII存储形式为：

ASCII码：00110101 00110110 00110111 00111000

歌词文件(lrc):文本文件

### 二进制码文件

基于值编码,自己根据具体应用,指定某个值是什么意思

把内存中的数据按其内存中的存储形式原样输出到磁盘上

一般需要自己判断或使用特定软件分析数据格式

例如：数5678的存储形式为：

二进制码：00010110 00101110

音频文件(mp3):二进制文件

图片文件（bmp）文件，一个像素点由两个字节来描述\*\*\*\*\*&&&&&

\*代表红色的值

#代表绿色的值

&代表蓝色的值

二进制文件以位来表示一个意思。

### 文本文件、二进制文件对比：

#### 译码：

文本文件编码基于字符定长，译码容易些；

二进制文件编码是变长的，译码难一些（不同的二进制文件格式，有不同的译码方式）。

#### 空间利用率：

二进制文件用一个比特来代表一个意思(位操作)；

而文本文件任何一个意思至少是一个字符。

二进制文件，空间利用率高。

#### 可读性：

文本文件用通用的记事本工具就几乎可以浏览所有文本文件

二进制文件需要一个具体的文件解码器，比如读BMP文件，必须用读图软件。

总结一下：

文件在硬盘上存储的时候，物理上都是用二进制来存储的。

咱们的标准io库函数，对文件操作的时候，不管文件的编码格式（字符编码、或二进制），而是按字节对文件进行读写，所以咱们管文件又叫流式文件，即把文件看成一个字节流。

## 二、文件指针

文件指针就是用于标识一个文件的，所有对文件的操作都是用对文件指针进行操作的

### 定义文件指针的一般形式为:

FILE \* 指针变量标识符；

本质上文件指针是一个结构体指针，结构体中包含了当前文件的很多信息，但是在实际编程时，

不需要关系结构体中的成员，只需要使用文件指针即可

## 对文件操作的步骤：

- 1、对文件进行读写等操作之前要打开文件得到文件指针
- 2、可以通过文件指针对文件进行读写等操作
- 3、读写等操作完毕后，要关闭文件，关闭文件后，就不能再通过此文件指针操作文件了

## c语言中有三个特殊的文件指针无需定义，在程序中可以直接使用

**stdin**：标准输入 默认为当前终端（键盘）

我们使用的scanf、getchar函数默认从此终端获得数据

**stdout**：标准输出 默认为当前终端（屏幕）

我们使用的printf、puts函数默认输出信息到此终端

**stderr**：标准错误输出设备文件 默认为当前终端（屏幕）

当我们程序出错使用perror函数时信息打印在此终端

## 三、打开文件fopen

```
1 #include <stdio.h>
2 FILE *fopen(const char *path, const char *mode);
3 功能：创建或者打开一个文件
4 参数：
5   path: 文件名，如果只写文件名，默认就是当前路径，也可以添加路径
6   mode: 文件权限
7   r 只读，如果文件不存在则报错
8   r+ 读写，如果文件不存在则报错
9   w 只写，如果文件不存在则创建，如果文件存在则清空
10  w+ 读写，如果文件不存在则创建，如果文件存在则清空
11  a 只写，如果文件不存在则创建，如果文件存在则追加
12  a+ 读写，如果文件不存在则创建，如果文件存在则追加
13 返回值：
14  成功：文件指针
15  失败：NULL
```

```
1 #include <stdio.h>
2
3 int main(int argc, char *argv[])
4 {
5     //使用fopen函数打开或者创建文件，返回文件指针
```

```

6  FILE *fp;
7  //以只读的方式打开文件，如果文件不存在则报错
8  //fp = fopen("C:/Users/lzx/Desktop/file.txt", "r");
9
10 //以只写的方式打开文件，如果文件不存在则创建，如果文件存在清空
11 //fp = fopen("C:/Users/lzx/Desktop/file.txt", "w");
12
13 //以只写的方式打开文件，如果文件不存在则创建，入股哦文件存在则追加
14 fp = fopen("C:/Users/lzx/Desktop/file.txt", "a");
15 if(fp == NULL)
16 {
17     printf("fail to fopen\n");
18     return -1;
19 }
20
21 return 0;
22 }

```

## 四、关闭文件fclose

```

1  #include <stdio.h>
2  int fclose(FILE *stream);
3  功能：关闭一个文件指针，无法在对当前文件进行操作
4  参数：
5      stream: 指定的文件指针，fopen函数的返回值
6  返回值：
7      成功: 0
8      失败: EOF
9  注意：注意一个文件只能关闭一次，不能多次关闭。
10  关闭文件之后就不能再文件指针对文件进行读写等操作了。

```

```

1  #include <stdio.h>
2
3  int main(int argc, char *argv[])
4  {
5      //使用fopen函数打开或者创建文件，返回文件指针
6      FILE *fp;
7
8      //以只写的方式打开文件，如果文件不存在则创建，入股哦文件存在则追加
9      fp = fopen("C:/Users/lzx/Desktop/file.txt", "a");

```

```

10  if(fp == NULL)
11  {
12  printf("fail to fopen\n");
13  return -1;
14  }
15
16  //使用fclose关闭文件
17  fclose(fp);
18
19  return 0;
20 }

```

## 五、一次读写一个字符

### 5.1 fgetc

```

1  #include <stdio.h>
2  int fgetc(FILE *stream);
3  功能：从文件指针标识的文件中读取一个字符
4  参数：
5    stream: 指定的文件指针
6  返回值：
7    成功：读取的字符
8    失败：EOF
9    如果文件读取完毕，也会返回EOF

```



```

1  #include <stdio.h>
2
3  int main(int argc, char *argv[])
4  {
5    FILE *fp;
6    fp = fopen("C:/Users/lzx/Desktop/file.txt", "r");
7    if(fp == NULL)
8    {
9      printf("fail to fopen\n");
10     return -1;

```



```

11  }
12
13  //使用fgetc从文件中读取一个字符
14  // int c = fgetc(fp);
15  // printf("c = [%c] - %d\n", c, c);
16
17  // c = fgetc(fp);
18  // printf("c = [%c] - %d\n", c, c);
19
20  //文件的每一行结束的位置都有一个标识，是一个换行符，称之为行结束符
21  //fgetc可以读取到行结束符
22  int c;
23  while((c = fgetc(fp)) != EOF)
24  {
25      printf("c = [%c] - %d\n", c, c);
26  }
27
28  return 0;
29  }

```

## 执行结果

```

Starting C:\Users\lzx\Desktop\
\03_fgetc.exe...
c = [h] - 104
c = [e] - 101
c = [l] - 108
c = [l] - 108
c = [o] - 111
c = [
] - 10
c = [6] - 54
c = [6] - 54
c = [6] - 54
C:\Users\lzx\Desktop\src\buil

```

## 5.2 fputc

```

1  #include <stdio.h>
2  int fputc(int c, FILE *stream);
3  功能：向文件指针标识的文件中写入一个字符
4  参数：
5      c：要写入的字符
6      stream：指定的文件指针
7  返回值：
8      成功：要写入的字符
9      失败：EOF


```

```

1 #include <stdio.h>
2
3 int main(int argc, char *argv[])
4 {
5     FILE *fp;
6     fp = fopen("C:/Users/lzx/Desktop/file.txt", "w");
7     if(fp == NULL)
8     {
9         printf("fail to fopen\n");
10        return -1;
11    }
12
13    //通过fputc函数向文件写入一个字符
14    fputc('w', fp);
15    fputc('h', fp);
16    fputc('a', fp);
17    fputc('t', fp);
18    fputc('\n', fp);
19    fputc('o', fp);
20
21    return 0;
22 }

```

## 执行结果

 file.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

what

o

# 六、一次读写一个字符串

## 6.1 fgets

```

1 #include <stdio.h>
2 char *fgets(char *s, int size, FILE *stream);
3 功能：从文件中读取内容
4 参数：
5     s: 保存读取到的内容
6     size: 每次读取的最大个数

```

7 **stream**: 文件指针  
8 返回值:  
9 成功: 读取的数据的首地址  
10 失败: **NULL**  
11 如果文件内容读取完毕, 也返回**NULL**  
12 注意: 从**stream**所指的文件中读取字符, 在读取的时候碰到换行符或者是碰  
13 到文件的末尾停止读取, 或者是读取了**size-1**个字节停止读取, 在读取  
14 的内容后面会加一个**\0**, 作为字符串的结尾

 file.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
hello world
nihao beijing
```

```
1 #include <stdio.h>
2
3 int main(int argc, char *argv[])
4 {
5     FILE *fp;
6     fp = fopen("C:/Users/lzx/Desktop/file.txt", "r");
7     if(fp == NULL)
8     {
9         printf("fail to fopen\n");
10        return -1;
11    }
12
13    //使用fgets读取文件内容
14    //fgets每次读取时最多读取文件一行内容, 只要遇到行结束符就立即返回
15    //如果想要读取的字节数小于一行内容, 则只会读取第二个参数-1个字节,
16    //最后位置补\0
17    char buf[32] = "";
18    //fgets(buf, 8, fp);
19    fgets(buf, 32, fp);
20    printf("buf = %s\n", buf);
21
22    return 0;
23 }
```

执行结果

```
Starting C:\Users\lzx\Desktop\src\05_fgets.exe...
buf = hello world
```


```
C:\Users\lzx\Desktop\src\05_fgets.exe
exited with code 0
```

## 6.2 fputs

```
1 #include <stdio.h>
2 int fputs(const char *s, FILE *stream);
3 功能：向文件写入数据
4 参数：
5   s：要写入的内容
6   stream：文件指针
7 返回值：
8   成功：写入文件内容的字节数
9   失败：EOF
```

```
1 #include <stdio.h>
2
3 int main(int argc, char *argv[])
4 {
5     FILE *fp;
6     fp = fopen("C:/Users/lzx/Desktop/file.txt", "w");
7     if(fp == NULL)
8     {
9         printf("fail to fopen\n");
10        return -1;
11    }
12
13    //通过fputs函数向文件写入数据
14    fputs("6666666666666666\n", fp);
15    fputs("nihao", fp);
16
17    return 0;
18 }
```

### 执行结果

 file.txt - 记事本


文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
6666666666666666
nihao
```

## 七、读文件fread

```
1 #include <stdio.h>
2 size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);
3 功能：从文件中读取数据
4 参数：
5     ptr：保存读取的数据
6     size：每次读取的字节数
7     nmemb：一共读取的次数
8     stream：文件指针
9 返回值：
10    成功：实际读取的次数（对象数、块数）
11    失败：0
12    如果文件内容读取完毕，返回0
```

```
1 例1：
2     int num;
3     num=fread(str,100,3,fp);
4     从fp所代表的文件中读取内容存放到str指向的内存中，读取的字节数为 ，每块100个字节，3块。
5     返回值num，
6     如果读到300个字节返回值num为3
7     如果读到了大于等于200个字节小于300个字节 返回值为2
8     读到的字节数，大于等于100个字节小于200个字节 返回1
9     不到100个字节返回0
```

 file.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
6666666666666666
nihao
```

```
1 #include <stdio.h>
2
3 int main(int argc, char *argv[])
4 {
5     FILE *fp;
```

```

6  fp = fopen("C:/Users/lzx/Desktop/file.txt", "r");
7  if(fp == NULL)
8  {
9      printf("fail to fopen\n");
10     return -1;
11 }
12
13 //使用fread函数读取文件内容
14 int num;
15 char buf[128] = "";
16 num = fread(buf, 5, 4, fp);
17
18 printf("buf = %s\n", buf);
19 printf("num = %d\n", num);
20
21 return 0;
22 }

```

## 执行结果

```

Starting C:\Users\lzx\Desktop\
\07_fread.exe...
buf = 6666666666666666
nihao
num = 4
C:\Users\lzx\Desktop\src\build
exited with code 0

```

# 八、写文件fwrite

```

1  #include <stdio.h>
2  size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);
3  功能：向文件中写入数据
4  参数：
5      ptr: 要写入的数据
6      size: 一次写入的字节数
7      nmemb: 一共写入的次数
8      stream: 文件指针
9  返回值：
10     成功：实际写入的次数
11     失败：0

```

```

1  #include <stdio.h>
2

```

```

3 typedef struct{
4     int a;
5     int b;
6     char c;
7 }MSG;
8
9 int main(int argc, char *argv[])
10 {
11     FILE *fp;
12     fp = fopen("C:/Users/lzx/Desktop/file.txt", "w+");
13     if(fp == NULL)
14     {
15         printf("fail to fopen\n");
16         return -1;
17     }
18
19     //使用fwrite向文件写入一个结构体
20     MSG msg[4] = {1, 2, 'a', 3, 4, 'b', 5, 6, 'c', \
21     7, 8, 'd'};
22
23     fwrite(msg, sizeof(MSG), 4, fp);
24
25     //将文件的偏移量设置为文件的起始位置
26     rewind(fp);
27
28     MSG rcv[4];
29     fread(rcv, sizeof(MSG), 4, fp);
30     int i;
31     for(i = 0; i < 4; i++)
32     {
33         printf("%d - %d - %c\n", rcv[i].a, rcv[i].b, rcv[i].c);
34     }
35
36
37     return 0;
38 }

```

## 执行结果

file.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

☐ a☐@ ☐ ☐ b ☐ ☐ c ) ☐ ☐ d

```
Starting C:\Users\lzx\Desktop\src\08_fwrite.exe...
1 - 2 - a
3 - 4 - b
5 - 6 - c
7 - 8 - d
C:\Users\lzx\Desktop\src\l
```

## 九、格式化读写文件函数

```
1  函数调用：
2  fprintf ( 文件指针，格式字符串，输出表列)；
3  fscanf  ( 文件指针，格式字符串，输入表列)；
4
5  函数功能：
6  从磁盘文件中读入或输出字符
7
8  fprintf 和printf函数类似：
9  printf是将数据输出到屏幕上（标准输出），
10 fprintf函数是将数据输出到文件指针所指定的文件中。
11
12 fscanf和scanf 函数类似：
13 scanf是从键盘（标准输入）获取输入，
14 fscanf是从文件指针所标示的文件中获取输入。
```

```
1  #include <stdio.h>
2
3  int main(int argc, char *argv[])
4  {
5      FILE *fp;
6      char ch1='a', ch2;
7      int num1=50, num2;
8      char string1[20]="hello", string2[20];
9      float score1 = 85.5, score2;
10
11     if((fp = fopen("C:/Users/lzx/Desktop/file.txt","w+"))==NULL)
12     {
13         printf("fail to fopen\n");
14         return -1;
15     }
16
17     //使用fprintf向文件写入字符串
```

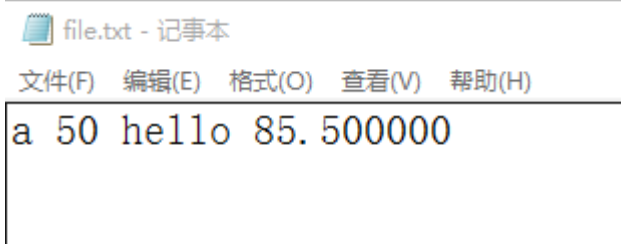


```

18  fprintf(fp, "%c %d %s %f\n", ch1, num1, string1, score1);
19
20  rewind(fp);
21
22  //使用fscanf获取文件内容
23  fscanf(fp, "%c %d %s %f\n", &ch2, &num2, &string2, &score2);
24  printf("%c %d %s %f\n", ch2, num2, string2, score2);
25
26  fclose(fp);
27  return 0;
28 }

```

## 执行结果



```

Starting C:\Users\lzx\Desktop
\debug\09_fprintf_fscanf.exe
a 50 hello 85.500000
C:\Users\lzx\Desktop\src\bu
\09_fprintf_fscanf.exe exit

```

# 十、随机读写

前面介绍的对文件的读写方式都是顺序读写，即读写文件只能从头开始，顺序读写各个数据;

但在实际问题中常要求只读写文件中某一指定的部分，例如：读取文件第200--300个字节

为了解决这个问题可以移动文件内部的位置指针到需要读写的位置，再进行读写，这种读写称为随机读写实现随机读写的关键是要按要求移动位置指针，这称为文件的定位.

## 10.1 rewind

```

1  #include <stdio.h>
2  void rewind(FILE *stream);
3  功能：将文件位置定位到起始位置
4  参数：
5    stream: 文件指针
6  返回值：无

```

## 10.2 ftell

```
1 #include <stdio.h>
2 long ftell(FILE *stream);
3 功能：获取当前文件的偏移量
4 参数：
5     stream: 文件指针
6 返回值：
7     获取当前文件的偏移量
```

## 10.3 fseek

```
1 #include <stdio.h>
2 int fseek(FILE *stream, long offset, int whence);
3 功能：设置文件位置指针的偏移量
4 参数：
5     stream: 文件指针
6     offset: 偏移量
7     可正可负也可0
8     whence: 相对位置
9     SEEK_SET 文件起始位置
10    SEEK_CUR 文件当前位置
11    SEEK_END 文件末尾位置（最后一个字符后面一个位置）
12 返回值：
13    成功：0
14    失败：-1
15
16 rewind(fp) <==> fseek(fp, 0, SEEK_SET);
```


```
1 #include <stdio.h>
2
3 int main(int argc, char *argv[])
4 {
5     FILE *fp;
6     if((fp = fopen("C:/Users/lzx/Desktop/file.txt", "w+"))==NULL)
7     {
8         printf("fail to fopen\n");
9         return -1;
10    }
11
12    fputs("123456789\n", fp);
```

```

13  fputs("abcdefghijklmn", fp);
14
15  //获取当前文件指针的读写位置
16  printf("offset = %ld\n", ftell(fp));
17
18  //将当前文件的读写文件设置到文件的起始位置
19  //rewind(fp);
20  //fseek(fp, 0, SEEK_SET);
21
22  //将当前文件的读写位置设置为倒数第五个位置
23  fseek(fp, -5, SEEK_END);
24
25  char buf[32] = "";
26  while(fgets(buf, 32, fp) != NULL)
27  {
28      printf("[%s]\n", buf);
29  }
30
31  return 0;
32 }

```

## 执行结果

 file.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

123456789  
abcdefghijklmn

starting C:\Users\lzx\I  
Debug\debug\10\_rewind\_  
offset = 25  
[jklmn]  
C:\Users\lzx\Desktop\s  
10\_rewind\_ftell\_fseek