

第10章 对文件的输入输出（1）



几则通知

- 本周三（今天）最后一次上机课；
- OJ服务器本周日晚上关闭，做题历史记录会被清空，以备下周的上机考试；
- 期末上机考试，下周三（1月3日）晚19:10-21:40，3道题；题目分值为1，2，2；满分5分（总评分）
- 最终成绩：平时30%+上机5%+期末65%

复习回顾

➤ 上次课的内容：

- ◆ 有序链表的合并
- ◆ 查找链表节点
- ◆ 链表节点的冒泡排序
- ◆ 共用体 (union)
- ◆ 枚举类型 (enum)
- ◆ Typedef
- ◆ 圣诞歌的两个版本

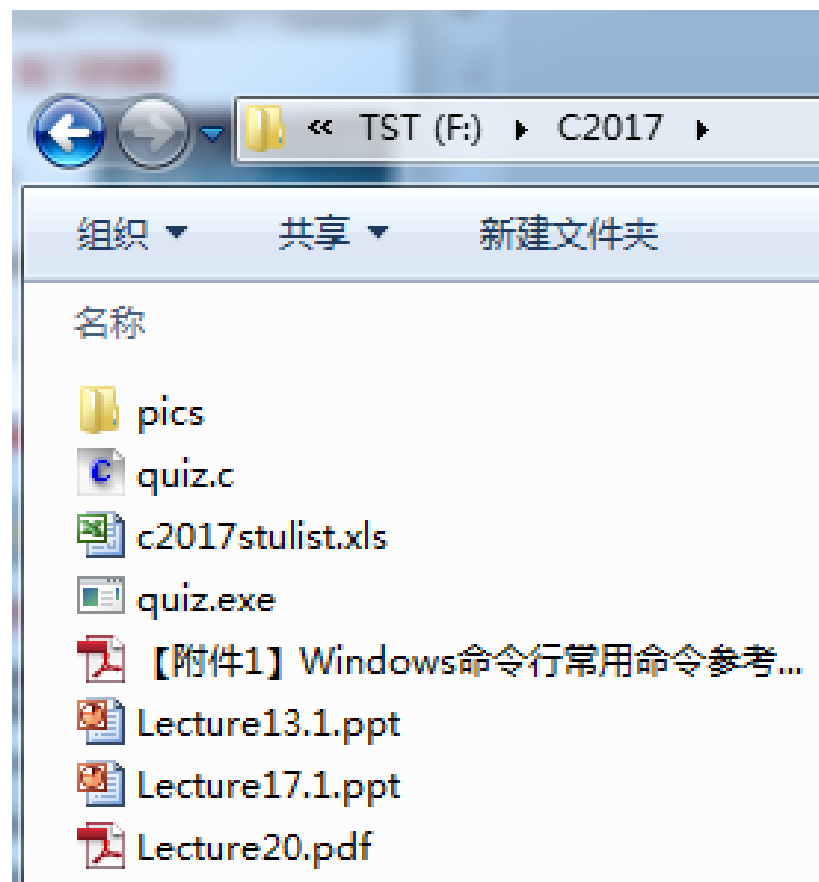


```
Union Xmas_Song
{
    char ver1[100]="jingle bells,
                    jingle bells,
                    jingle all the way...";

    char ver2[100]="single boys,
                    single boys,
                    single all the way...";
}
```

什么是文件

- “文件”（file）就是指存储在外部存储器上的数据集合。
- 程序文件和相关数据文件通常都存储在外部存储器上，只有在执行时，会被载入内存。
- 每个文件都有一个符号化的指代，这个符号化的指代就是“文件名”。



文件的命名

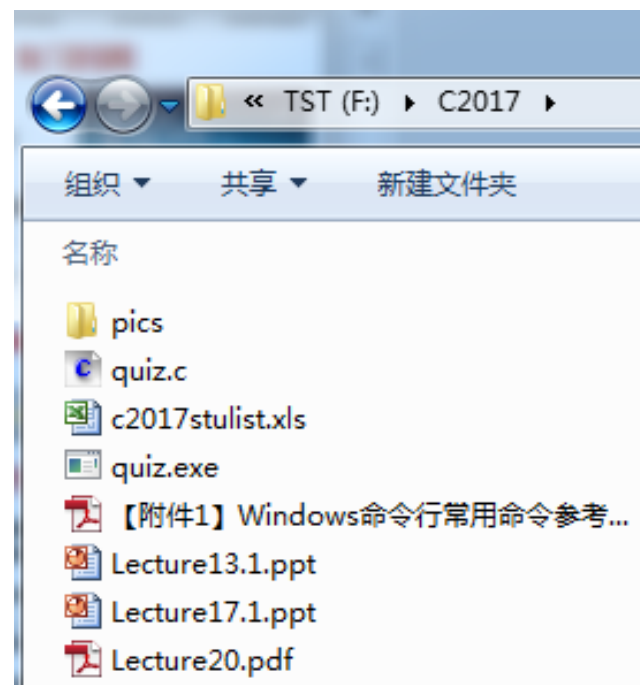
➤ 文件要有一个唯一的**文件标识**，以便用户识别和引用。

◆ 文件标识包括三部分：

(1) 文件**路径**

(2) 文件名**主干**

(3) 文件**后缀**



文件名举例

➤ 文件路径表示文件在外部存储设备中的位置。如：

文件路径

文件名主干

文件后缀

D: \CC\temp\file1.dat

◆ 表示file1.dat文件存放在D盘中的CC目录下的temp子目录下面

文件名举例

- 文件路径表示文件在外部存储设备中的位置。如：

文件名

D: \CC\temp\file1.dat

- ◆ 表示file1.dat文件存放在D盘中的CC目录下的temp子目录下面

文件名举例

- 文件路径表示文件在外部存储设备中的位置。如：

命名规则遵循标识符的命名规则

D: \CC\temp\file1.dat

- ◆ 表示file1.dat文件存放在D盘中的CC目录下的temp子目录下面

文件名举例

➤ 文件路径表示文件在外部存储设备中的位置。如：

一般不超过3个字母（doc、txt、dat、c、cpp、obj、exe、ppt、bmp等）

D: \CC\temp\file1.dat

◆ 表示file1.dat文件存放在D盘中的CC目录下的temp子目录下面

五花八门的计算机文件



广义的文件分类

➤ 从操作系统的角度来看，可分为**普通文件**和**设备文件**

◆ **普通文件**：驻留在磁盘或其他外部介质的一个有序数据集

◆ **设备文件**：与主机相连的各种外部设备，如显示器、打印机、键盘等。这是一种逻辑上的文件。



设备文件

➤ 设备只是**逻辑上**的文件，操作系统把各种设备都统一作为文件处理

◆ 每一个与主机相联的输入输出设备都看作是文件



● 键盘被定义为**标准输入文件**，在键盘上键入数据就意味着从标准输入文件接收数据



● 显示屏被定义为**标准输出文件**，在屏幕上显示信息就是向标准输出文件输出信息

普通文件

- 在以前各章中所处理的数据的输入和输出都是面向设备文件，从终端的键盘输入数据，运行结果输出到终端显示器上
- 常常需要将一些数据输出到磁盘上保存起来，以便以后使用
- 这就要用到**磁盘文件**（此后所出现的文件专指除设备文件外的普通文件）



程序设计中的文件类型

➤ 从程序设计的角度来看，主要用到两种文件：

(1) **程序文件。**

◆ **源程序文件**(后缀为.c)

◆ **目标文件**(后缀为.obj)

◆ **可执行文件**(后缀为.exe)

这种文件的内容是程序代码。

程序设计中的文件类型

➤ 从程序设计的角度来看，主要用到两种文件：

(2) **数据文件**。文件的内容不是程序，而是供程序运行时读写的数据，如在程序运行过程中输出到磁盘(或其他外部设备)的数据，或在程序运行过程中供读入的数据。

● 如一批学生的成绩数据，或货物交易的数据等。

➤ 本章主要讨论的是**数据文件**

数据文件的分类

- 根据数据的组织形式，数据文件可分为**文本文件**和**二进制文件**。
 - ◆ 数据在内存中是以二进制形式存储的，如果不加转换地输出到外存，就是**二进制文件**
 - ◆ **文本文件**又称**ASCII文件**，每一个字节放一个字符的ASCII代码

数据的存储方式

- 字符一律以**ASCII形式存储**
- 数值型数据既可以用ASCII形式存储，也可以用**二进制形式存储**，如果二进制数据要求在外存上以ASCII代码形式存储，则需要在存储前进行转换
 - ◆ 如有整数123，如果用ASCII码形式输出到磁盘，则在磁盘中占3个字节(每一个字符占一个字节)，而用二进制形式short类型输出，则在磁盘上只占2个字节

数值型数据的存储方式

整数
123

ASCII形式

| | | |
|----------|----------|----------|
| 00110001 | 00110010 | 00110011 |
|----------|----------|----------|

('1')

('2')

('3')

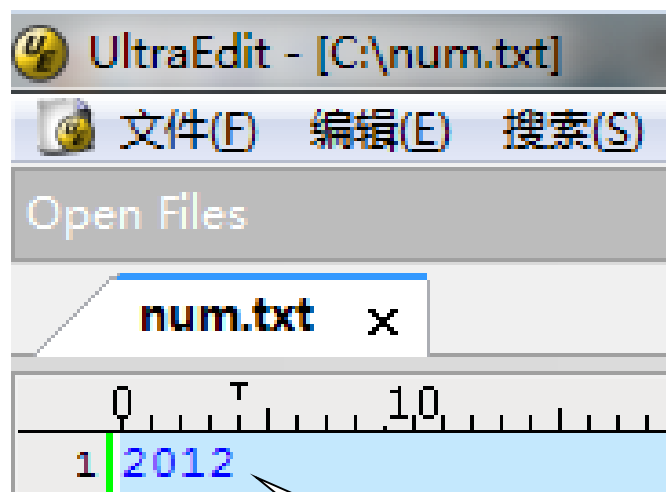
二进制形式 (short类型)

| | |
|----------|----------|
| 00000000 | 01111011 |
|----------|----------|

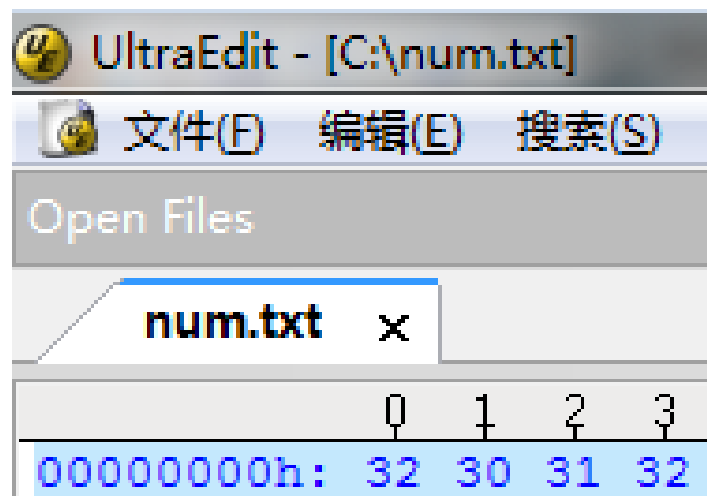
(123)

近距离观察文本文件

在C盘根目录创建num.txt文本文件，如下：



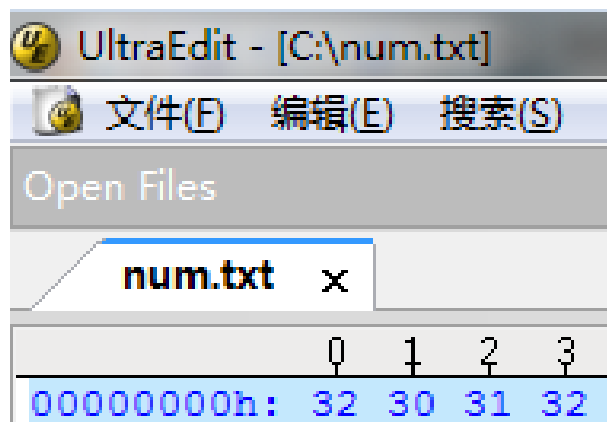
文本模式



16进制模式

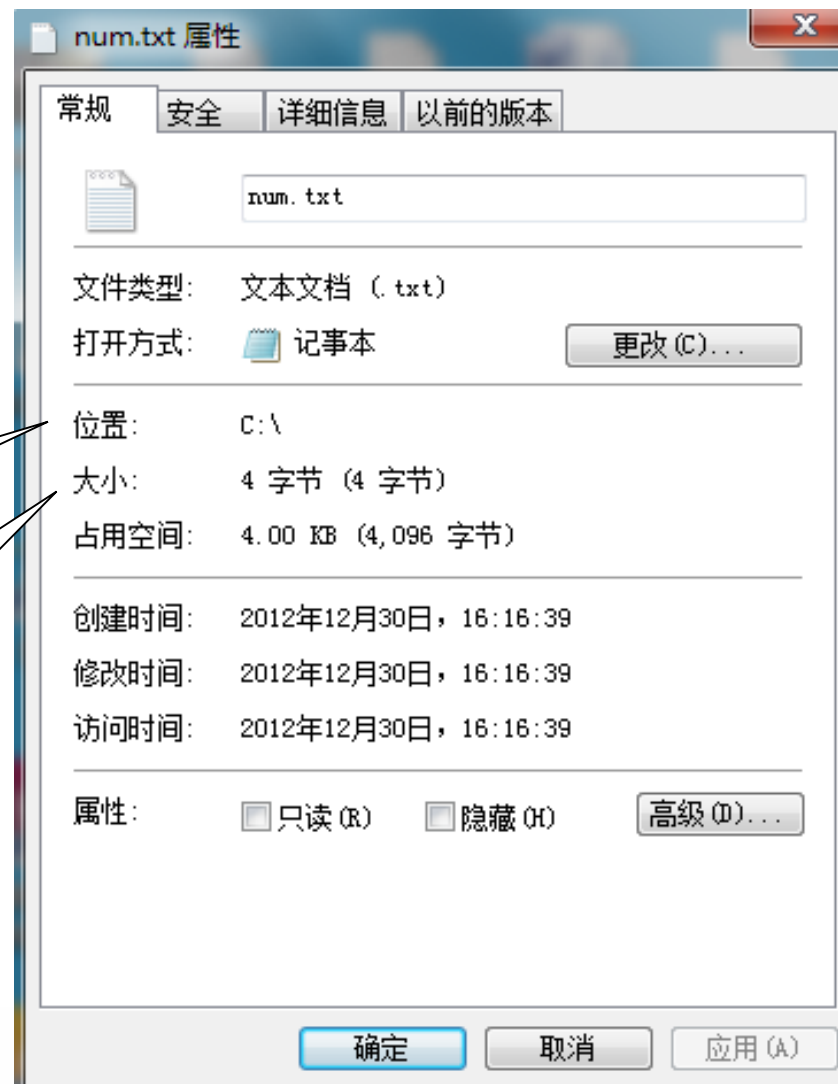
提示：'2'的ASCII码 = $(50)_{10} = (3 \times 16 + 2)_{10} = (32)_{16}$

文件的属性



文件位置：c:\

文件大小：4个字节



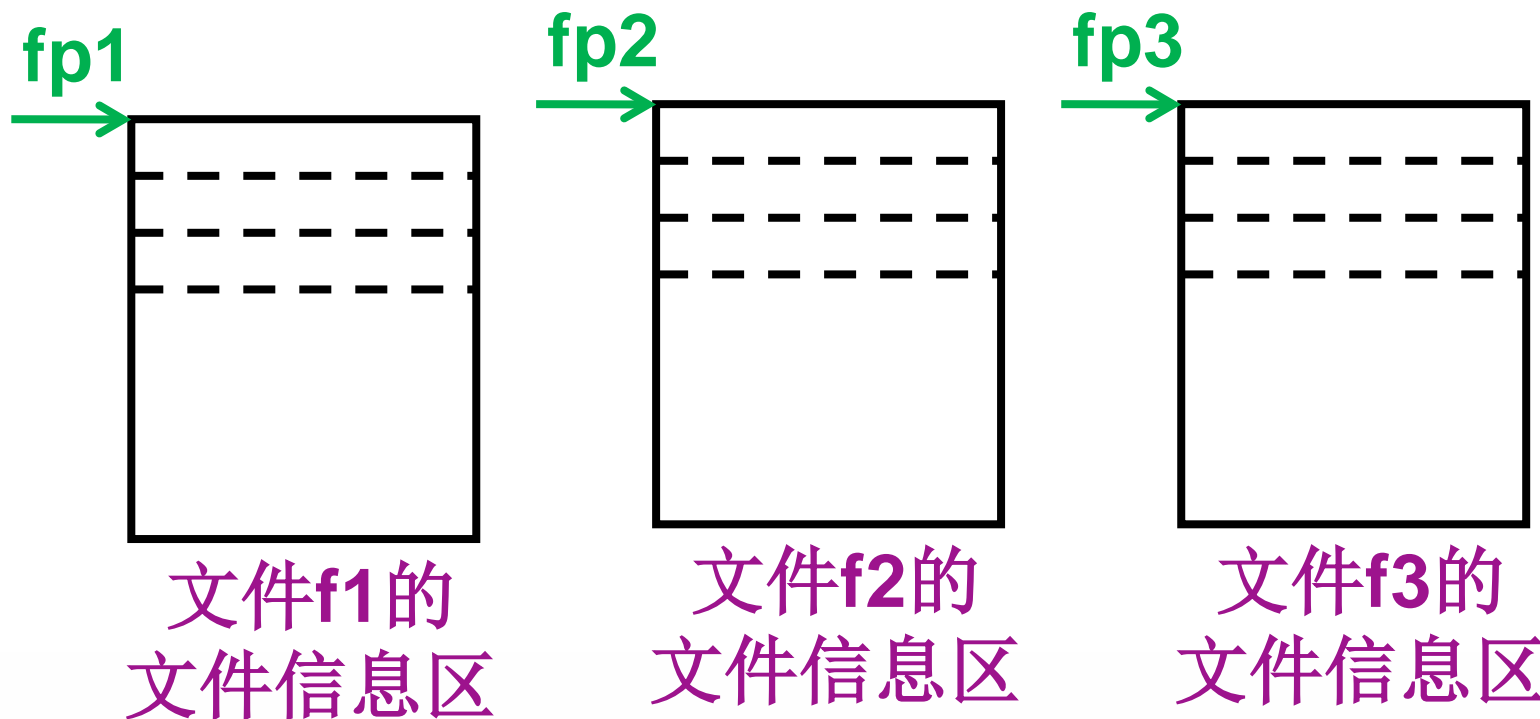
文件类型指针

- 每个被使用的文件都在内存中开辟一个相应的**文件信息区**，用来存放文件的有关信息（如文件的**名字**、**文件状态**及**文件当前位置**等），指向这片内存的指针就是**文件类型指针**
- 这些信息是保存在一个结构体变量中的。**该结构体类型是由系统声明的**，取名为FILE，相应的指针类型就是FILE *
- 声明FILE结构体类型的信息包含在头文件“stdio.h”中

文件类型指针

- 一般设置一个指向FILE类型变量的指针变量，然后通过它来引用这些FILE类型变量，

例如，`FILE f1, f2, f3, *fp1=&f1, *fp2=&f2, *fp3=&f3;`



C语言中的文件操作

- 在C语言中，对于文件的操作分为几个部分
 1. 文件**打开**
 2. 文件数据**读取**
 3. 文件数据**写入**
 4. 文件**关闭**
- **文件的打开和关闭是对文件操作的前提！**
- **文件数据的读/写是文件操作的核心部分**

fopen函数：用于打开一个文件

➤ fopen函数的调用方式为：

fopen(文件名, 打开文件的方式);

◆ 例如：

fopen("a1", "r");

- 表示要打开名为“a1”的文件，打开文件的方式为“读入”
- 当打开的文件和执行程序在同一个目录时，文件名的路径可以省略，否则，应该提供完整路径！
- fopen函数的返回值是指向a1文件的指针

fopen函数的使用方式

➤ 通常将fopen函数的返回值赋给一个指向文件的指针变量。

◆ 如：

文件名

打开文件的方式

```
FILE *fp;
```

```
fp=fopen("a1", "r");
```

// 此处把文件指针fp和文件a1相联系，即fp指向了a1文件

打开文件方式中不同字符的含义

➤ 一共六种字符：

◆ **r** (read) : 读

◆ **w** (write) : 写

◆ **b** (binary) : 二进制文件

◆ **t** (text) : 文本文件，默认情况，可省略不写

◆ **a** (append) : 追加

◆ **+** : 读和写

➤ 注意，某些字符可以组合！

◆ 例如，“**rb**”表示读一个二进制文件，“**wb**”表示写一个二进制文件，“r”和“w”则分别表示读写文本文件

使用fopen的注意事项

- 如果**文件打开失败**，fopen函数将会带回一个出错信息。fopen函数将带回一个空指针值NULL。故常用if(fp==NULL)判断操作是否成功。

◆常用下面的方法打开一个文件：

```
if ((fp=fopen("file1","r"))==NULL)
{
    printf("cannot open this file\n");
    exit(0);
}
```

终止正在执行的程序

用fclose函数关闭数据文件

- 关闭文件用fclose函数。fclose函数调用的一般形式为

fclose(文件指针);

例如：fclose (fp); 其中fp是一个FILE类的指针，指向一个已打开的文件

- 所谓“**关闭**”是指撤销文件信息区和文件缓冲区，若成功则fclose函数返回0，否则返回EOF(-1)。如果不关闭文件将会**丢失**数据

顺序读写数据文件

- 在**顺序写**时，先写入的数据存放在文件中前面，后写入的数据存放在文件中后面
- 在**顺序读**时，先读文件中前面的数据，后读文件中后面的数据
- 对顺序读写来说，对文件读写**数据的顺序**和数据在文件中的**物理顺序**是一致的
- 顺序读写需要用**库函数**实现

怎样向文件读写字符

➤ 读写一个字符的两个函数：fgetc与fputc

| 函数名 | 调用形式 | 功能 | 返回值 |
|--------------|---------------------|------------------------|--|
| fgetc | fgetc(fp) | 从fp指向的文件读入一个字符 | 读成功，带回所读的字符，失败则返回文件结束标志 E O F (即-1) |
| fputc | fputc(ch,fp) | 把字符ch写到文件指针变量fp所指向的文件中 | 写成功，返回值就是输出的字符；输出失败，则返回 E O F (即-1) |

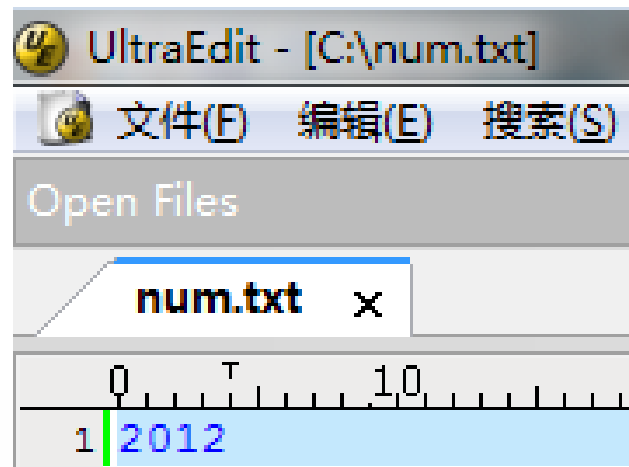
打开文本文件并读取一个字符

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

3. int main()
4. {
5.     FILE * fp;
6.     char c;
7.     fp = fopen("c:\\num.txt", "r"); //“文本只读” 方式打开文件
8.     if (fp==NULL)
9.     {
10.         printf("error on opening file!\n");
11.         exit(0);
12.     }
13.     c = fgetc(fp); //读取一个字符
14.     printf("%c\n", c);
15.     fclose(fp); //关闭文件
16.
17.     return 0;
18. }
```

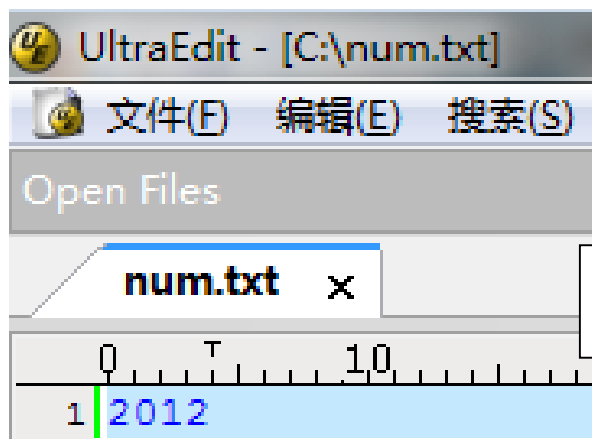
2

Press any key to continue



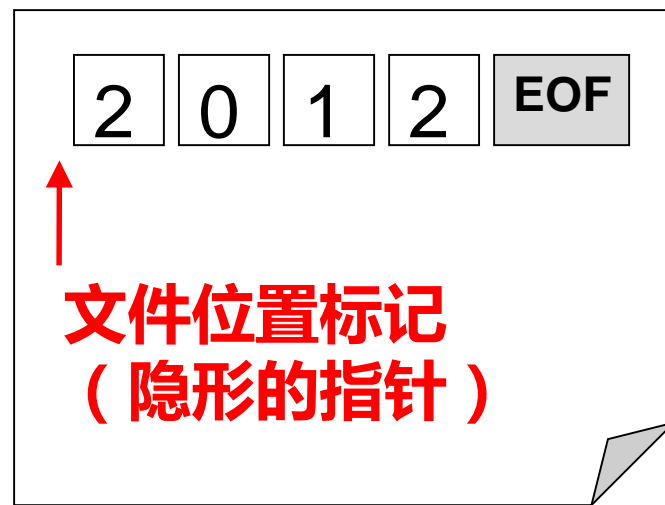
“r”模式打开文件时发生了什么

- 文件从硬盘载入内存，占据一片内存空间
- 文件指针（即fp）指代整个文件，始终指向文件头
- 文件位置指针（即位置标记）暂时指向文件头



载入内存

fp

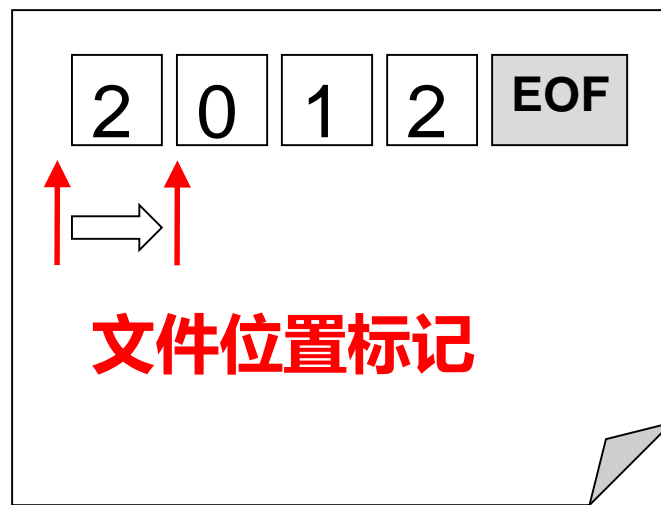


执行一次fgetc()后发生了什么

- 文件里文件位置标记当前位置的字符被读取
- 文件位置标记往后移动一个字节，下次读/写将从这个位置开始

```
c = fgetc(fp);
```

→
fp



用fgetc读取整个文本文件内容

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

3. int main()
4. {
5.     FILE * fp;
6.     char c;
7.     fp=fopen("c:\\num.txt", "r"); //“文本只读” 方式打开文件
8.     if (fp==NULL)
9.         exit(0);
10.    while (!feof(fp))
11.    {
12.        c = fgetc(fp);
13.        putchar(c);
14.    }
15.    putchar(10);
16.    fclose(fp); //关闭文件
17.    return 0;
18. }
```

2012
Press any key to continue

fp →

文件位置标记
此时feof(fp)=1

文件末尾的检测：feof函数

➤ 一般调用形式为 `feof(fp)`;

◆ 其中`fp`必须是一个有效的文件指针，而且该文件必须已经成功打开。

◆ **作用**是在对文件读/写操作的过程中，判断文件是否已经读/写结束，即测试文件位置指针是否达到文件尾。

◆ 若达到文件尾或发生错误则返回`true`（非0），其他情形返回`false`（0）

使用fgetc函数的注意事项

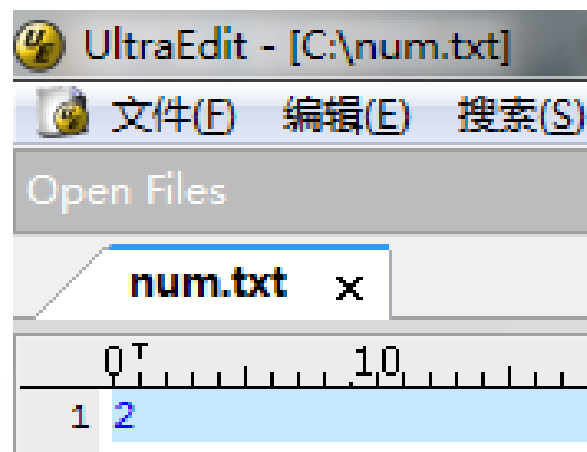
- 在fgetc()函数中，读取的文件必须是以读或者读/写方式打开的
- 读取字符的结果也可以不向字符变量赋值，但在这种情况下，读出的字符将不能被保存，如fgetc(pf);
- 文件内部在进行读/写操作时，文件位置指针是移动的。在打开文件时，该指针总是指向文件的第一个字节。在使用fgetc()函数后，该位置指针往后移动一个字节。因此，在对文件进行读/写操作时，可以使用fgetc()连续多次读取字符。

打开文本文件并写入一个字符

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

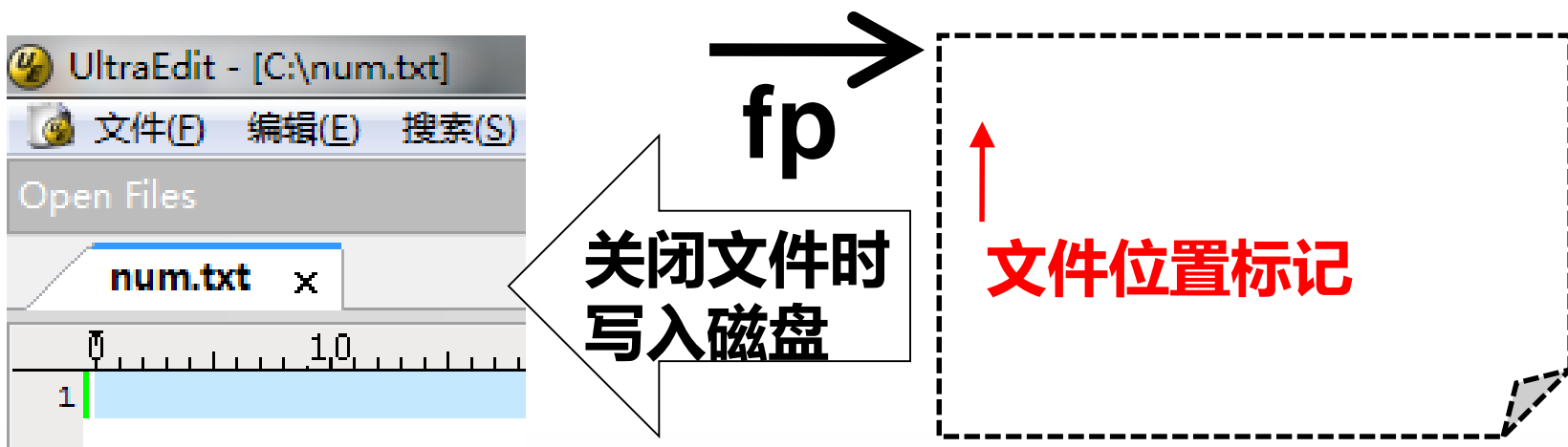
3. int main()
4. {
5.     FILE * fp;
6.     fp = fopen("c:\\num.txt", "w"); //“文本只写” 方式打开文件
7.     if (fp==NULL)
8.     {
9.         printf("error on opening file!\n");
10.        exit(0);
11.    }
12.    fputc(50, fp); //写入一个字符
13.    fclose(fp); //关闭文件
14.
15.    return 0;
16. }
```

Press any key to continue.



“w”模式打开文件时发生了什么

- 若原文件存在，内容清空；
- 文件指针（即fp）指代整个文件，始终指向文件头
- 文件位置指针（即位置标记）暂时指向文件头



使用fputc函数的注意事项

- 被写入的文件可以用**写、读/写、追加方式**打开
 - ◆ 用**写或读/写方式**打开一个已存在的文件时将清除原有的文件内容，写入的字符从文件首开始。
 - ◆ 如需保留原有文件内容，希望写入的字符以文件末开始存放，必须以**追加方式**打开文件。被写入的文件若不存在，则创建该文件
- 每写入一个字符，**文件位置指针向后移动一个字节**
- fputc()函数有一个**返回值**，如写入成功则返回写入的字符，否则返回EOF，可用此来判断写入是否成功

实现文本文件复制的例子

例：将一个文本文件中的信息复制到另一个文本文件中。今要求将上例建立的num.txt文件中的内容复制到另一个文本文件num.copy中。

➤ **解题思路：**处理此问题的算法是：从num.txt文件中逐个读入字符，然后逐个输出到num.copy中。

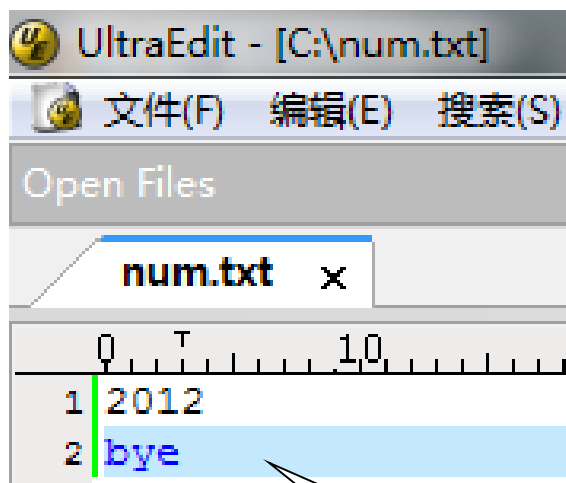

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. int main( )
4. {
5.     FILE *in,*out;
6.     char  ch,infile[10],outfile[10];
7.     printf("输入读入文件的名字:");
8.     scanf("%s",infile);
9.     printf("输入输出文件的名字:");
10.    scanf("%s",outfile);
11.    if ((in=fopen(infile, "r"))==NULL)
12.    {
13.        printf("无法打开此文件\n");
14.        exit(0);
15.    }
16.    if ((out=fopen(outfile, "w"))==NULL)
17.    {
18.        printf("无法打开此文件\n");
19.        exit(0);
20.    } //未完待续
```

//紧接上页

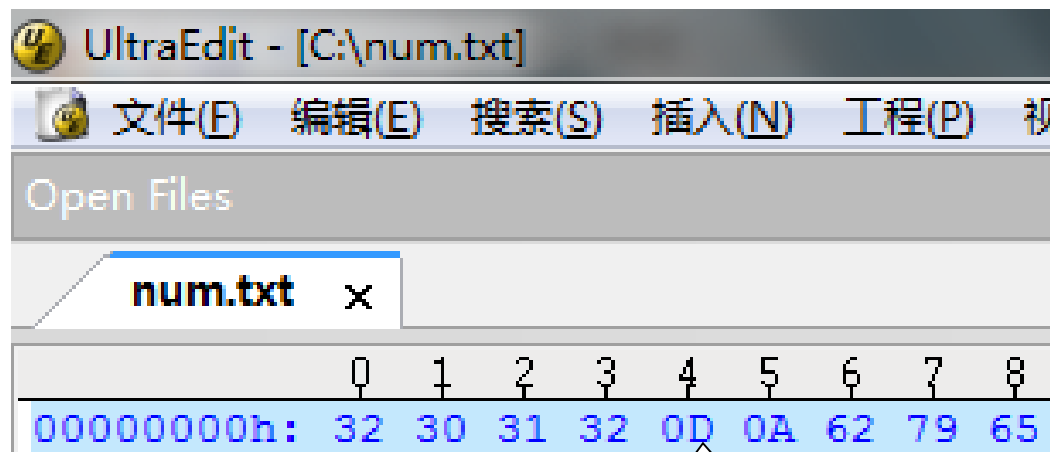
```
21. while (!feof(in))
22. {
23.     ch = fgetc(in);
24.     fputc(ch, out);
25.     putchar(ch);
26. }
27. putchar(10);
28. fclose(in);
29. fclose(out);
30. return 0;
31. }
```

检查当前读写位置
是否移到文件末尾

多行文本文件有什么不同



文本模式



16进制模式

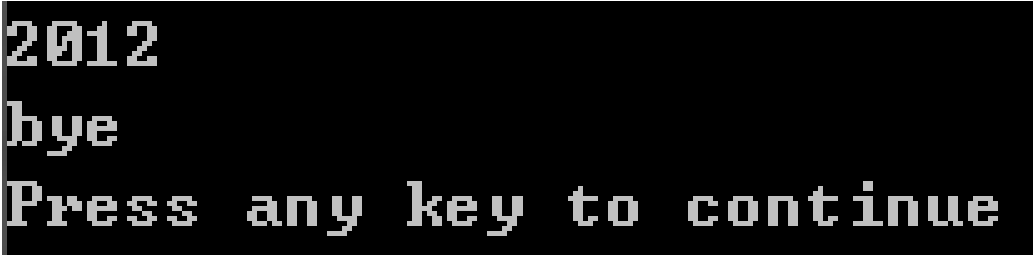
提示： $(0D)_{16}$ 和 $(0A)_{16}$ 分别是回车符和换行符的ASCII码

注意： windows和unix情况不同，unix下只有换行符

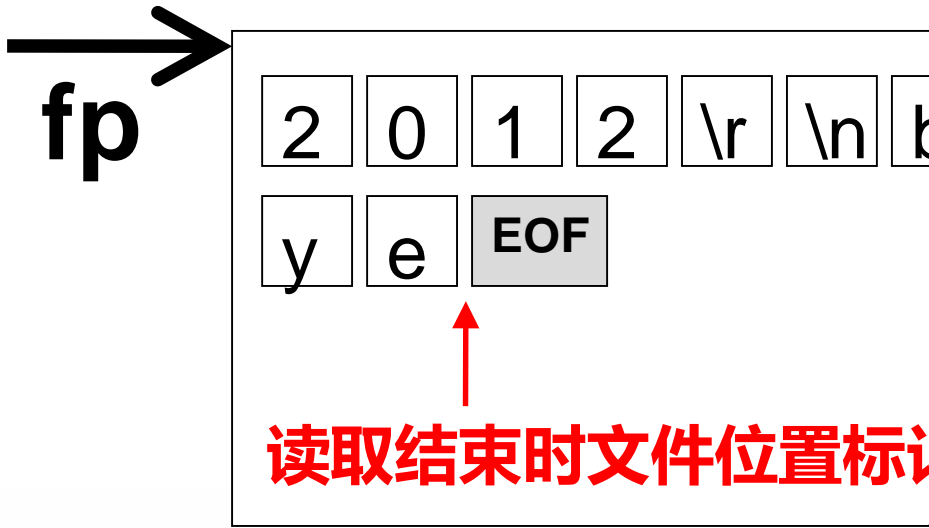
用fgetc读取多行文本文件内容

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

3. int main()
4. {
5.     FILE * fp;
6.     char c;
7.     fp = fopen("c:\\num.txt", "r"); //“文本只读” 方式打开文件
8.     if (fp==NULL)
9.         exit(0);
10.    while (!feof(fp))
11.    {
12.        c = fgetc(fp);
13.        putchar(c);
14.    }
15.    putchar(10);
16.    fclose(fp); //关闭文件
17.    return 0;
18. }
```



2012
bye
Press any key to continue



fp →

| | | | | | | |
|---|---|-----|---|----|----|---|
| 2 | 0 | 1 | 2 | \r | \n | b |
| y | e | EOF | | | | |

读取结束时文件位置标记

怎样向文件读写一个字符串

➤ 读写一个字符串的函数fgets与fputs

| 函数名 | 调用形式 | 功能 | 返回值 |
|--------------|------------------------|------------------------------------|------------------------|
| fgets | fgets(str,n,fp) | 从fp指向的文件读入长度为(n-1)的字符串，存放到字符数组str中 | 读成功，返回地址str，失败则返回NULL) |
| fputs | fputs(str,fp) | str所指向的字符串写到文件指针变量fp所指向的文件中 | 写成功，返回0；否则返回非0值 |

关于fgets函数的说明

fgets函数的函数原型为：

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

◆其作用是从文件读入一个字符串

◆调用时可以写成：`fgets(str,n,fp);`

◆`fgets(str,n,fp);`中n是要求得到的字符个数，但实际上只读n-1个字符，然后在最后加一个'\0'字符，这样得到的字符串共有n个字符，把它们放到字符数组str中

关于fgets函数的说明

- 如果在读完 $n-1$ 个字符之前遇到换行符“`\n`”或文件结束符EOF，读入即结束，但将所遇到的换行符“`\n`”也作为一个字符读入
- 执行fgets成功，返回str数组首地址，如果一开始就遇到文件尾或读数据错，返回NULL

关于fputs函数的说明

fputs函数的函数原型为：

```
int fputs (char *str, FILE *fp);
```

- ◆str指向的字符串输出到fp所指向的文件中
- ◆调用时可以写成：fputs("China",fp);
- ◆fputs函数中第一个参数可以是字符串常量、字符数组名或字符型指针
- ◆字符串末尾的'\0'不输出
- ◆输出成功，函数值为0；失败，函数值为EOF

用fgets读取一行文本文件内容

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

3. int main()
4. {
5.     FILE * fp;
6.     char c[10];
7.     fp = fopen("c:\\num.txt", "r"); //“文本只读” 方式打开文件
8.     if (fp==NULL)
9.         exit(0);
10.
11.     fgets(c, 10, fp);
12.     printf("%s", c);

13.     fclose(fp); //关闭文件
14.
15.     return 0;
16. }
```

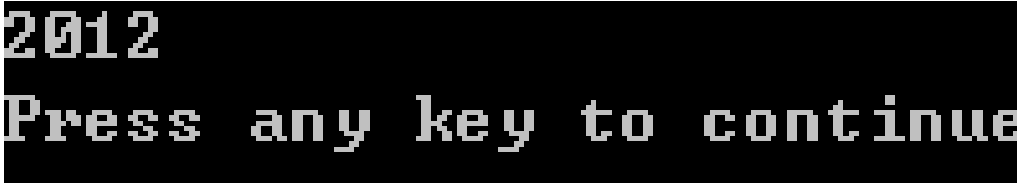


Diagram illustrating the state of the file pointer `fp` after reading a line from the file:

| | | | | | | |
|---|---|-----|---|----|----|---|
| 2 | 0 | 1 | 2 | \r | \n | b |
| y | e | EOF | | | | |

Red text: 读取结束时文件位置标记 (Read end file position marker)

用fgets读取少于一整行的内容

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

3. int main()
4. {
5.     FILE * fp;
6.     char c[10];
7.     fp = fopen("c:\\num.txt", "r"); //“文本只读” 方式打开文件
8.     if (fp==NULL)
9.         exit(0);
10.
11.     fgets(c, 4, fp);
12.     printf("%s", c);

13.     fclose(fp); //关闭文件
14.
15.     return 0;
16. }
```

201Press any key to continue

fp →

| | | | | | | |
|---|---|-----|---|----|----|---|
| 2 | 0 | 1 | 2 | \r | \n | b |
| y | e | EOF | | | | |

↑ 读取结束时文件位置标记

用fgets读取文本文件全部内容

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

3. int main()
4. {
5.     FILE * fp;
6.     char c[10];
7.     fp = fopen("c:\\num.txt", "r"); //“文本只读” 方式打开文件
8.     if (fp==NULL)
9.         exit(0);
10.    while (!feof(fp))
11.    {
12.        fgets(c, 10, fp);
13.        printf("%s", c);
14.    }
15.    fclose(fp); //关闭文件
16.
17.    return 0;
18. }
```

2012
byePress any key to continue

fp →

| | | | | | | |
|---|---|-----|---|----|----|---|
| 2 | 0 | 1 | 2 | \r | \n | b |
| y | e | EOF | | | | |

文件位置标记

写入一个字符再写入字符串

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

3. int main()
4. {
5.     FILE * fp;
6.     fp = fopen("c:\\num.txt", "w"); //“文本只写”方式打开文件
7.     if (fp==NULL)
8.         exit(0);

9.     fputc(50, fp);
10.    fputs("012", fp);

11.    fclose(fp); //关闭文件
12.
13.    return 0;
14. }
```

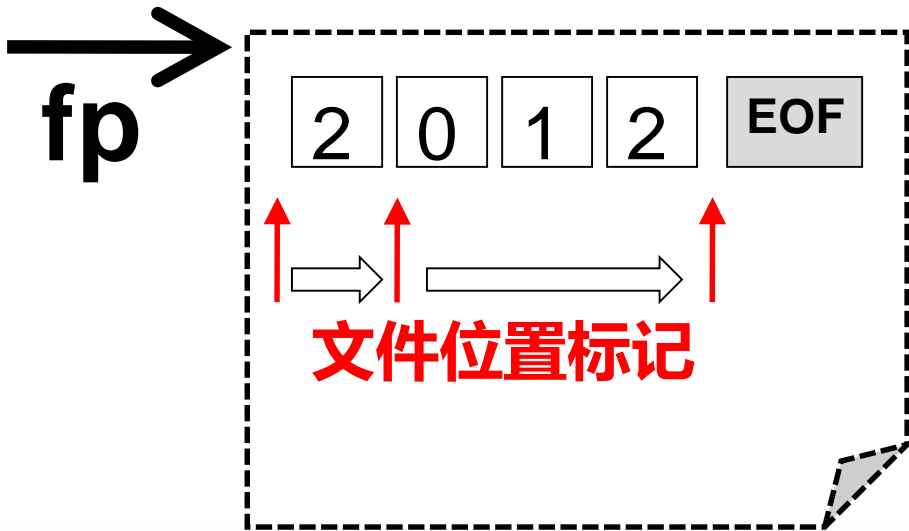


Diagram illustrating file position markers. A large arrow labeled **fp** points to a dashed box containing a sequence of boxes: **2**, **0**, **1**, **2**, and **EOF**. Below these boxes are three red upward-pointing arrows, with a horizontal arrow pointing from the first to the last, labeled **文件位置标记** (File Position Marker).

| num.txt | | x |
|---------|------|----|
| 0 | 1 | 10 |
| 1 | 2012 | |

作业 2017/12/27

➤ 按下列要求编写程序，提交手写源代码

1. 打造自己的姓名有序链表。分别把自己的姓和名按字符递增顺序排序。然后把排好序的姓和名分别添加到两个字符链表中，再把两个链表合并，要求合并后的链表字符还是保持递增顺序。

输入格式：

第一行输入自己的姓的汉语拼音全拼（大写）

第二行输入自己的名字的汉语拼音全拼（大写）

输出格式：输出自己的姓名有序链表结果

样例输入： ZHENG

WEI

样例输出： EEGHINWZ

【注意事项】 程序必须添加必要的注释！