

对文件的输入输出

程 淼

E-mail: mew_cheng@outlook.com

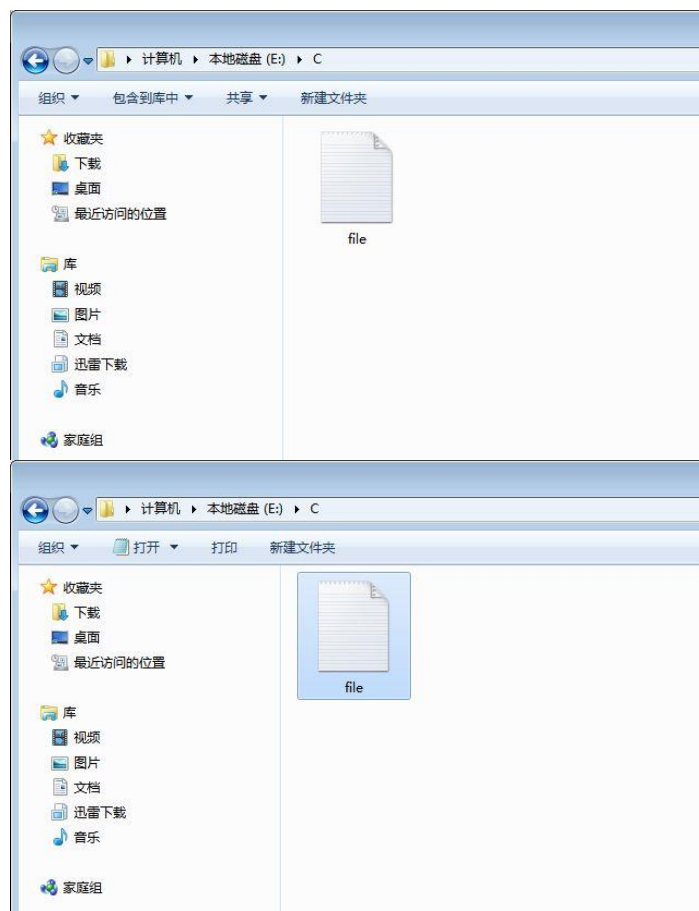
这一章学什么？

◎ 如何访问和修改文件

◎ 一个例子：如何打开文件？

■ 解题思路：

- 将鼠标移动到目标文件上方
- 双击鼠标左键



这一章学什么？

◎ 理想与现实的差别

请打开柜子



请打开柜子



了解文件

◎ 什么是文件

- 程序文件
- 数据文件

◎ 每一个与主机相连的输入输出设备都看作一个文件

- 操作系统把各种设备都统一作为文件来处理

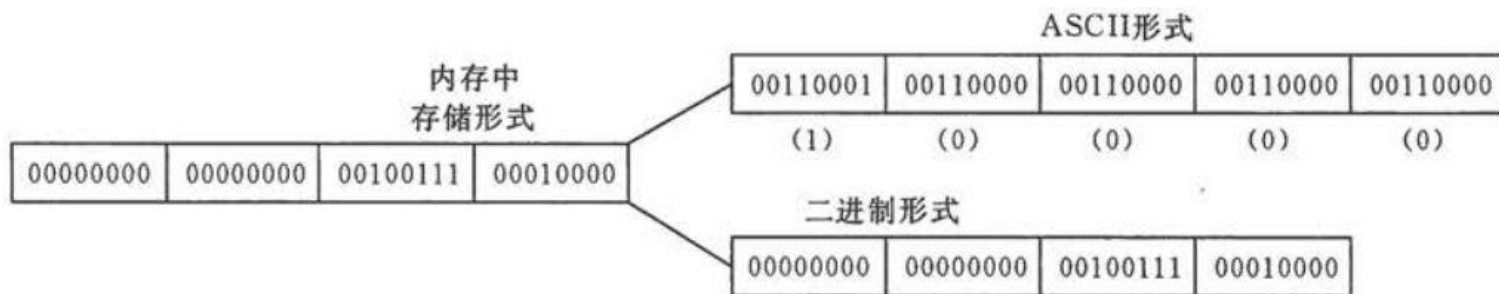


了解文件

- ◎ 所谓“文件”一般指存储在外部介质上数据的集合
- ◎ 操作系统以文件为单位对数据进行管理
- ◎ 数据输入输出的传送过程，如流水一样，因此常将输入输出形象地称为流，即数据流
 - “流”是一个传输通道，数据可以从运行环境流入程序中，或从程序流至运行环境

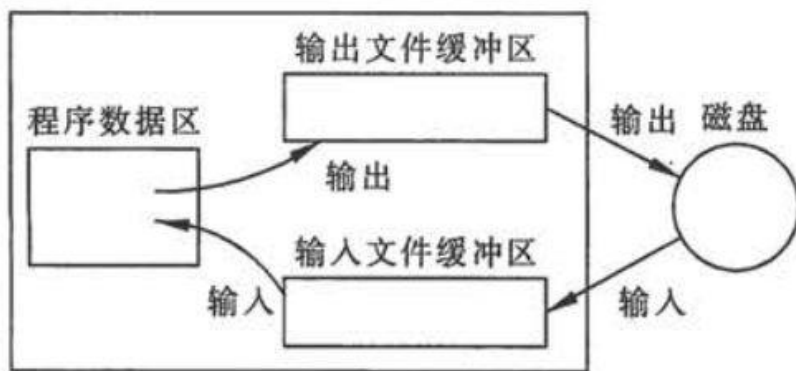
了解文件

- ◎ 一个文件有一个唯一的文件标识
- ◎ 文件的分类
 - ASCII文件：文本文件
 - 二进制文件：映像文件



了解文件

- 文件缓冲区：ANSI C标准采用“缓冲文件系统”处理数据文件
- 缓冲文件系统**是指系统自动地在内存区为程序中每一个正在使用的文件开辟一个文件缓冲区



了解文件

- ◎ 缓冲文件系统中，关键的概念是“**文件类型指针**”，简称“**文件指针**”
- ◎ 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
- ◎ 这些信息是保存在一个系统声明的结构体变量中，取名为FILE
 - 例如，C编译环境提供的stdio.h头文件

```
typedef struct
{
    short level;
    unsigned flags;
    char fd;
    unsigned char
    hold;
    short bsize;
    unsigned char *
    buffer;
    unsigned char *
    curp;
    unsigned istemp;
    short token;
}FILE;
```


了解文件

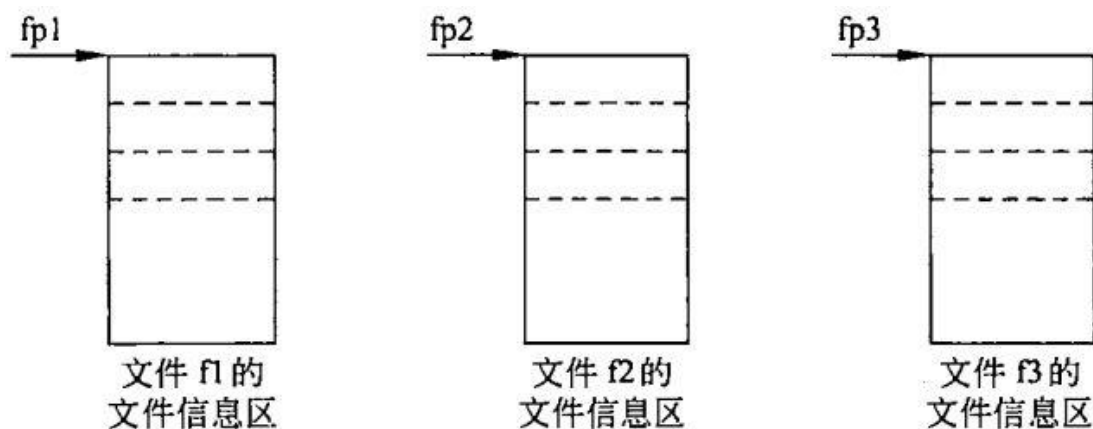
- ◎ 在程序中可以直接用FILE类型名定义变量
 - 声明FILE结构体类型的信息包含在“stdio.h”中
- ◎ 每一个FILE类型变量对应一个文件的信息区，其中存放该文件的有关信息
 - 例如，FILE f1;

了解文件

◎ **指向文件的指针变量**：通常通过设置一个指向FILE类型变量的指针变量来引用这些FILE类型变量

■ 例如，FILE * fp;

■ 通过指针变量能够访问**文件信息区**，进而通过信息区的信息找到与它关联的文件，进行访问



打开与关闭文件

打开文件

◎ 一个例子：把一只大象放进冰箱需要几个步骤？

■ 解题思路：

- 打开冰箱
- 放入大象
- 关上冰箱

打开文件

- ◎ 所谓“打开”是指为文件建立相应的信息区(用来存放有关文件的信息)和文件缓冲区(用来暂时存放输入输出的数据)
- ◎ 用标准输入输出函数fopen来实现打开文件
 - fopen(文件名, 使用文件方式);
 - 例如, fopen(“a1”, “r”);
- ◎ 通常将fopen函数的返回值赋给一个指向文件的指针变量
 - 例如,

```
FILE * fp;  
fp = fopen(“a1”, “r”);
```

打开文件

◎ 使用文件的方式

文件使用方式	含义	如果指定的文件不存在
“r”(只读)	为了输入数据，打开一个已存在的文本文件	出错
“w”(只写)	为了输出数据，打开一个文本文件	建立新文件
“a”(追加)	向文本文件尾添加数据	出错
“rb”(只读)	为了输出数据，打开一个二进制文件	出错
“wb”(只写)	为了输出数据，打开一个二进制文件	建立新文件
“ab”(追加)	向二进制文件尾添加数据	出错

打开文件

◎ 使用文件方式

文件使用方式	含义	如果指定的文件不存在
“r+”(只读)	为了读和写，打开一个文本文件	出错
“w+”(只写)	为了读和写，建立一个新的文本文件	建立新文件
“a+”(追加)	为了读和写，打开一个文本文件	出错
“rb+”(只读)	为了读和写，打开一个二进制文件	出错
“wb+”(只写)	为了读和写，建立一个新的二进制文件	建立新文件
“ab+”(追加)	为读写打开一个二进制文件	出错

关闭文件

- ◎ 为防止文件被误用，使用完一个文件后应该关闭它
 - “关闭”就是撤销文件信息区和文件缓冲区
 - 使文件指针变量不再指向该文件
- ◎ 一般形式为：`fclose(文件指针);`
 - 例如，`fclose(fp);`

关闭文件

- ◎ 如果不关闭文件将会丢失数据
 - 数据未充满缓冲区而程序结束允许
- ◎ 使用fclose函数，先把缓冲区中的数据输出到磁盘文件，然后才撤销文件信息区
- ◎ fclose函数也带回一个值，当成功地执行了关闭操作，则返回值0；否则返回EOF(-1)

顺序读写数据文件

顺序读写数据文件

- ◎ 文件打开之后，就可以对它进行**顺序读写**了
- ◎ 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

顺序读写数据文件

◎ 读写一个字符的函数

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EOF(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	输出成功，返回值就是输出的字符，输出失败，则返回EOF(即-1)

顺序读写数据文件

- ◎ 一个例子：从键盘输入一些字符，逐个把它们送到磁盘上去，直到用户输入一个“#”为止
 - 解题思路：用fgetc函数从键盘逐个输入字符，然后用fputc函数写到磁盘文件

顺序读写数据文件

◎ 编写程序：

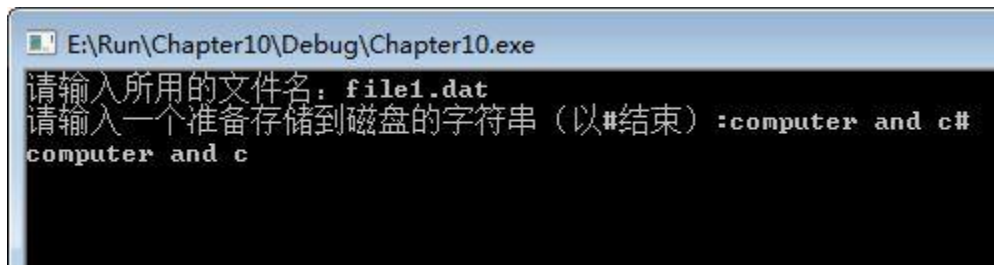
```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    FILE * fp;
    char ch, filename[10];
    printf("请输入所用的文件名：");
    scanf("%s", filename);
    if ((fp = fopen(filename, "w")) == NULL)
    {
        printf("无法打开此文件\n");
        exit(0);
    }

    ch = getchar();
    printf("请输入一个准备存储到磁盘的字符串（以#结束）：");
    ch = getchar();

    while(ch != '#')
    {
        fputc(ch, fp);
        putchar(ch);
        ch = getchar();
    }

    fclose(fp);
    putchar(10);
    return 0;
}
```



```
E:\Run\Chapter10\Debug\Chapter10.exe
请输入所用的文件名：file1.dat
请输入一个准备存储到磁盘的字符串（以#结束）：computer and c#
computer and c
```

顺序读写数据文件

- ◎ 例子：将一个磁盘文件中的信息复制到另一个磁盘文件中。将上例建立的file1.dat文件中的内容复制到另一个磁盘文件file2.dat中
 - 解题思路：从file1.dat文件中逐个读入字符，然后逐个输出到file2.dat中

顺序读写数据文件

◎ 编写程序：

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

int main()
{
    FILE * in, * out;
    char ch, infile[10], outfile[10];
    printf("输入读入文件的名字：");
    scanf("%s", infile);
    printf("输入输出文件的名字：");
    scanf("%s", outfile);
    if ((in = fopen(infile, "r")) == NULL)
    {
        printf("无法打开此文件\n");
        exit(0);
    }
    if ((out = fopen(outfile, "w")) == NULL)
    {
        printf("无法打开此文件\n");
        exit(0);
    }

    while(!feof(in))
    {
        ch = fgetc(in);
        fputc(ch, out);
        putchar(ch);
    }

    putchar(10);
    fclose(in);
    fclose(out);
    return 0;
}
```



顺序读写数据文件

- ◎ C语言允许通过函数fgets和fputs一次读写一个字符串，而非一个字符
 - 例如，fgets(str, n, fp);
 - 从fp所指向的文件中读入一个长度为n-1的字符串，并在最后加一个‘\0’字符，然后把这n个字符存放到字符数组str中

顺序读写数据文件

◎ 读写一个字符串的函数

函数名	调用形式	功能	返回值
fgets	fgets(str, n, fp)	从fp指向的文件读入一个长度为(n-1)的字符串，存放 to 字符数组str中	读成功，返回地址str，失败则返回NULL
fputs	fputs(str, fp)	把str所指向的字符串写到文件指针变量fp所指向的文件中	输出成功，返回0；否则返回非0值

顺序读写数据文件

◎ fgets函数的函数原型为

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

◎ fputs函数的函数原型为

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

顺序读写数据文件

- ◎ 一个例子：从键盘读入若干个字符串，对它们按字母大小的顺序排序，然后把排好序的字符串送到磁盘文件中保存

■ 解题思路：

- 从键盘读入 n 个字符串，存放在一个二维字符数组中，每个一维数组存放一个字符串
- 对字符数组中的 n 个字符串按字母顺序排序，排好序的字符串仍存放在字符数组中
- 将字符数组中的字符串顺序输出

◎ 编写程序

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    FILE * fp;
    char str[3][10], temp[10];
    int i, j, k, n = 3;
    printf("Enter strings:\n");

    for (i = 0; i < n; i++)
        gets(str[i]);

    for (i = 0; i < n-1; i++)
    {
        k = i;
        for (j = i+1; j < n; j++)
            if (strcmp(str[k], str[j]) > 0)
                k = j;

        if (k != i)
        {
            strcpy(temp, str[i]);
            strcpy(str[i], str[k]);
            strcpy(str[k], temp);
        }
    }

    if ((fp = fopen("E:\\Run\\Chapter10\\Chapter10\\10-3.dat", "w")) == NULL)
    {
        printf("can't open file!\n");
        exit(0);
    }

    printf("\nThe new sequence:\n");
    for (i = 0; i < n; i++)
    {
        fputs(str[i], fp);
        fputs("\n", fp);
        printf("%s\n", str[i]);
    }

    return 0;
}
```

E:\Run\Chapter10\Debug\Chapter10.exe

Enter strings:

CHINA

CANADA

INDIA

The new sequence:

CANADA

CHINA

INDIA

用格式化的方式读写文件

◎ 对文件进行格式化输入输出，要用到fprintf函数和fscanf函数

- fprintf(文件指针, 格式字符串, 输出表列);

- fscanf(文件指针, 格式字符串, 输入表列);

◎ 例如,

- fprintf(fp, “%d, %6.2f”, i, f);

- fscanf(fp, “%d, %f”, &i, &f);

用二进制方式向文件读写一组数据

- ◎ 一次输入输出一组数据：C语言允许用fread函数从文件中读一个数据块，用fwrite函数向文件写一个数据块
- ◎ 一般调用形式
 - fread(buffer, size, count, fp);
 - fwrite(buffer, size, count, fp);

用二进制方式向文件读写一组数据

- ◎ 一个例子：从键盘输入10个学生的有关数据，然后把它们转存到磁盘文件上去
 - 解题思路：定义一个有10个元素的结构体数组，存放10个学生的数据。
 - 从main函数输入10个学生的数据
 - 用save函数实现向磁盘输出学生数据
 - 用fwrite函数输出一个学生的数据

◎ 编写程序：

```
#include <stdio.h>
#define SIZE 10

struct Student_type
{
    char name[10];
    int num;
    int age;
    char addr[15];
} stud[SIZE];

void save()
{
    FILE * fp;
    int i;
    if ((fp = fopen("stu.dat", "wb")) == NULL)
    {
        printf("cannot open file\n");
        return;
    }

    for (i = 0; i < SIZE; i++)
        if (fwrite(&stud[i], sizeof(struct Student_type), 1, fp) != 1)
            printf("file write error\n");

    fclose(fp);
}

int main()
{
    int i;
    printf("Please enter data of students:\n");
    for (i = 0; i < SIZE; i++)
        scanf("%s%d%d%s", stud[i].name, &stud[i].num, &stud[i].age, stud[i].addr);

    save();
    return 0;
}
```

E:\Run\Chapter10\Debug\Chapter10.exe

```
Please enter data of students:
Zhang 1001 19 room_101
Sun 1002 20 room_102
Tan 1003 21 room_103
Ling 1004 21 room_104
Li 1006 22 room_105
Wang 1007 20 room_106
Zhen 1008 16 room_107
Fu 1010 18 room_108
Qin 1012 19 room_109
Liu 1014 21 room_110
```

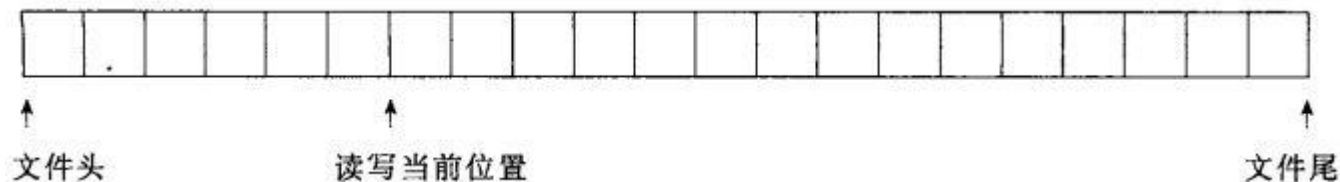
随机读写数据文件

随机读写数据文件

- ◎ 随机访问不是按数据在文件中的物理位置次序进行读写，而是可以对任何位置上的数据进行访问
 - 文件位置标记
 - 文件位置标记的定位

文件位置标记

- 为了对读写进行控制，系统为每个文件设置了一个**文件读写位置标记**(简称文件位置标记或文件标记)
 - 用来指示“接下来要读写的下一个字符的位置”



文件位置标记的定位

- ◎ 强制使文件位置标记指向人们指定的位置
 - 用rewind函数使文件位置标记指向文件开头

文件位置标记的定位

- ◎ 一个例子：有一个磁盘文件，内有一些信息。要求第1次将它的内容显示在屏幕上，第2次把它复制到另一文件上
 - 解题思路：在第1次读入完文件内容后，文件位置标记已到文件末尾。如果再接着读数据，就遇到文件结束标志
 - 必须使用rewind函数使位置指针返回文件的开头

文件位置标记的定位

◎ 编写程序

```
#include <stdio.h>

int main()
{
    FILE * fp1, * fp2;
    fp1 = fopen("file1.dat", "r");
    fp2 = fopen("file2.dat", "w");

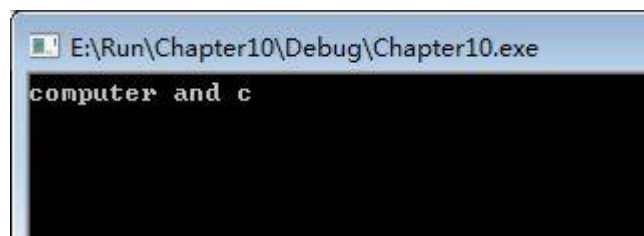
    while(!feof(fp1))    putchar(getc(fp1));

    putchar(10);
    rewind(fp1);

    while(!feof(fp1))    putc(getc(fp1), fp2);

    fclose(fp1);
    fclose(fp2);

    return 0;
}
```



文件位置标记的定位

◎ 用fseek函数改变文件位置标记

- 调用形式为:fseek(文件类型指针, 位移量, 起始点)
- 起始点用0、1或2代替, 0代表“文件开始位置”, 1为“当前位置”, 2为“文件末尾位置”

起始点	名字	用数字代表
文件开始位置	SEEK_SET	0
文件当前位置	SEEK_CUR	1
文件末尾位置	SEEK_END	2

文件位置标记的定位

- ◎ 用ftell函数测定文件位置标记的当前位置
 - 得到流式文件中文件位置标记的当前位置
 - 例如, `i = ftell(fp);` `// 变量i存放文件当前位置`

利用FSEEK函数的随机读写

- ◎ 一个例子：在磁盘文件上存有10个学生的数据。要求将第1，3，5，7，9个学生数据输入计算机，并在屏幕上显示出来

◎ 编写程序

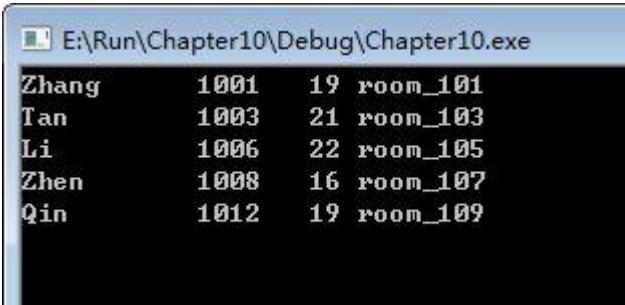
```
#include <stdio.h>
#include <stdlib.h>

struct student_type
{
    char name[10];
    int num;
    int age;
    char addr[15];
} stud[10];

int main()
{
    int i;
    FILE * fp;
    if ((fp = fopen("stu.dat", "rb")) == NULL)
    {
        printf("can not open file\n");
        exit(0);
    }

    for (i = 0; i < 10; i += 2)
    {
        fseek(fp, i*sizeof(struct student_type), 0);
        fread(&stud[i], sizeof(struct student_type), 1, fp);
        printf("%-10s %4d %4d %-15s\n", stud[i].name, stud[i].num, stud[i].age, stud[i].addr);
    }
    fclose(fp);

    return 0;
}
```



Zhang	1001	19	room_101
Tan	1003	21	room_103
Li	1006	22	room_105
Zhen	1008	16	room_107
Qin	1012	19	room_109

文件读写的出错检测

文件读写的出错检测

- ◎ 检查输入输出函数调用时肯能出现的错误
 - `ferror`函数：初始值自动置为0
 - 一般形式：`ferror(fp)`;
 - 如果`ferror`返回值为0，表示未出错；如果返回一个非零值，表示出错
 - `clearerr`函数
 - 使文件错误标志和文件结束标志置为0
 - 调用`clearerr(fp)`，使出现错误时`ferror(fp)`的非零值变成0，以便进行下一次的检测

