



第十三章 文件 (1)



在程序运行时，程序本身和数据一般都存放在内存。当程序运行结束后，存放在内存中的数据被释放。

如果需要长期保存程序运行所需的原始数据，或程序运行产生的结果，就必须将数据以文件形式存储到外部存储介质（如磁盘）上。

1、数据的层次结构

2、文件概述

3、文件的打开和关闭

4、位置指针与文件定位

5、文件的读写操作

6、顺序文件的操作

7、随机文件的操作



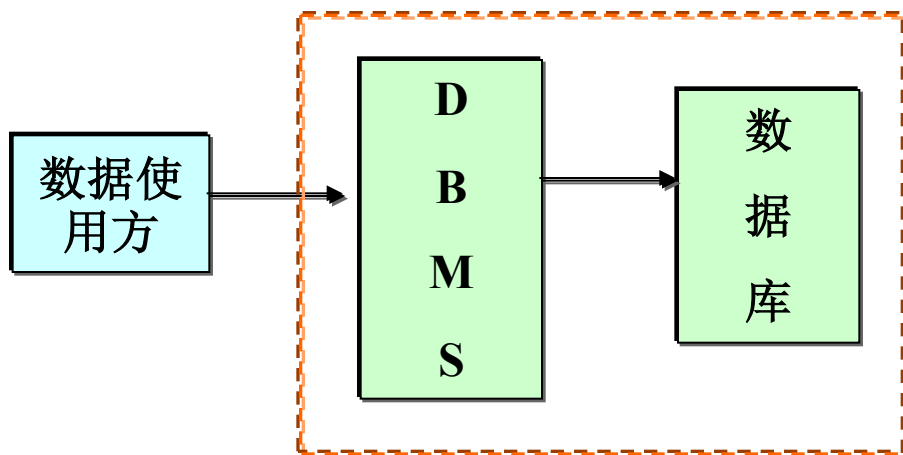
数据的层次结构



数据 (data)：描述事物的符号记录。

数据库 (database)：一组相关的文件集合。
如结构化数据库，非结构化数据库。

数据库管理系统：DataBase Management System(DBMS)，方便对数据的检索、更新、备份等操作



数据的层次结构



北京邮电大学
Beijing University of Posts and Telecommunications

数据：

- **结构化数据**：即行数据，有结构，可以存储在关系数据库中。如学生的信息是结构化数据，包括姓名、年龄、籍贯、身高等。
- **非结构化数据**：数据没有明确的结构，如全文文本、图像、声音、视频等。
- **半结构化数据**：介于完全结构化和完全无结构化数据之间的数据，如**HTML**文档。

数据的层次结构



结构化数据示例

001	Sally	34	
002	Judy	29	
003	Rose	30	

文件(file)

001	Judy	34	
-----	------	----	--

记录(record)

Judy

字段(field)

01001010

字节(Byte)

1

位(bit)

为了检索文件中的记录，
记录中至少要选出一个字
段作为记录关键字。

1、数据的层次结构

2、文件概述

3、文件的打开和关闭

4、位置指针与文件定位

5、文件的读写操作

6、顺序文件的操作

7、随机文件的操作



程序设计语言中的文件



- 文件是为了某种目的系统地把数据组织起来而构成的数据集合体。
- 为了处理的统一和概念的简化，操作系统把外部数据（磁盘文件）、外部设备（输入输出设备）一律作为文件来进行管理。
- 程序设计语言中的文件，指的是这些外部数据和外部设备。

13.2 文件概述



- 文件分类
- 文件的作用
- 读文件与写文件
- 文件和流
- ANSI C的缓冲文件系统
- 文件类型FILE
- 对文件的操作



13.2 文件概述

1. 普通文件（相对于设备文件）与文件名
普通文件是指存放在外部存储介质上的数据集合，外部存储介质有：硬盘、光盘、U盘等。为标识一个文件，每个文件都必须有一个文件名，其一般结构为：主文件名[. 扩展名]。文件命名规则需遵循操作系统的约定，扩展名代表了文件的类型。
 - 数据以二进制形式存储。
 - 程序读取数据时，首先要知道数据的类型，才能正确解析出数据。
 - 如果以类型**A**的二进制形式存入，却当作类型**B**读出，就乱套了。所以，读文件前，必须确切知道文件每一字节的确切类型和含义，所以才有标准的mp3、bmp、jpg等文件格式。

13.2 文件概述



2. 文件分类

可以从不同的角度对文件进行分类：

- (1) 根据文件的内容，可分为程序文件和数据文件，程序文件又可分为源文件、目标文件和可执行文件。



13.2 文件概述-文件分类

(2) 根据文件编码方式，可分为ASCII码文件和二进制文件

一.ASCII码文件

也称文本文件、**TEXT**文件，如**.txt**，**.c**，**.h**等文件内容是可打印字符。用一个字节来存储一个字符，存放的是对应字符的**ASCII**码。

假设内存中有整数**234**，占用**4**个内存字节。若用**ASCII**码文件来存放该整数，则计算机实际是先将该整数转换成字符串“**234**”（字符**2**、**3**、**4**的**ASCII**码分别为十进制的**50**、**51**、**52**），再存储到文件中，其存储形式为：

内存中用
4个字节
存放整数
234

00000000
00000000
00000000
11101010

00110010	00110011	00110100
----------	----------	----------

‘2’

‘3’

‘4’

ASCII码文件中用3个字节存放字符串“234”



13.2 文件概述-文件分类

- 缺点：将数据从内存存储到**ASCII**文件中，需要做转换（耗费时间）。
- 优点：**ASCII**文件可以在屏幕上按字符显示，人能读懂其内容（需要将二进制数据转换成对应的字符）

二. 二进制文件

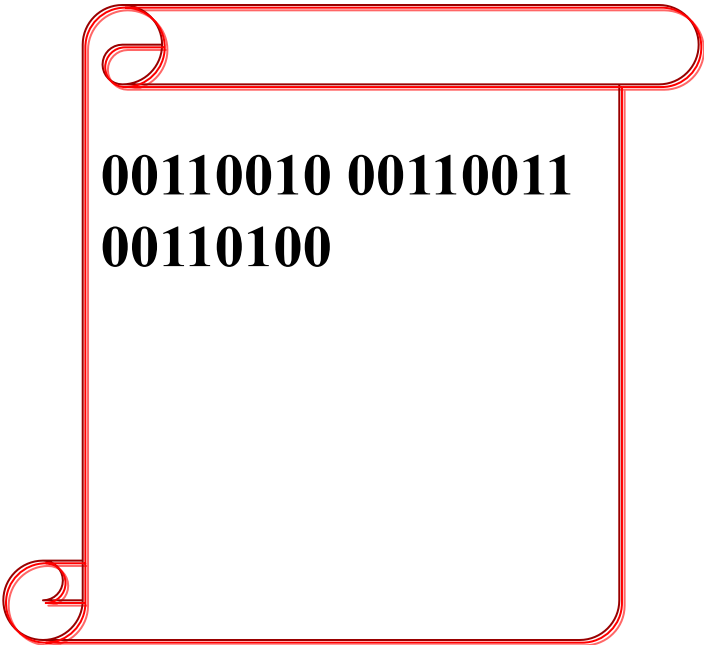
- 如**.exe**, **.doc**, **.mp3**, **.jpg**等文件。
- 优点：**1)** 二进制文件是把内存中的数据，原样输出到磁盘文件中，中间不做任何处理，这样可以节省转换时间。**2)** 比文本文件小很多。**3)** 有些数据不容易被表示为字符
- 缺点：二进制文件虽然也可在屏幕上显示，但其内容人无法读懂。必须使用专用的软件来打开。

13.2 文件概述-文件分类



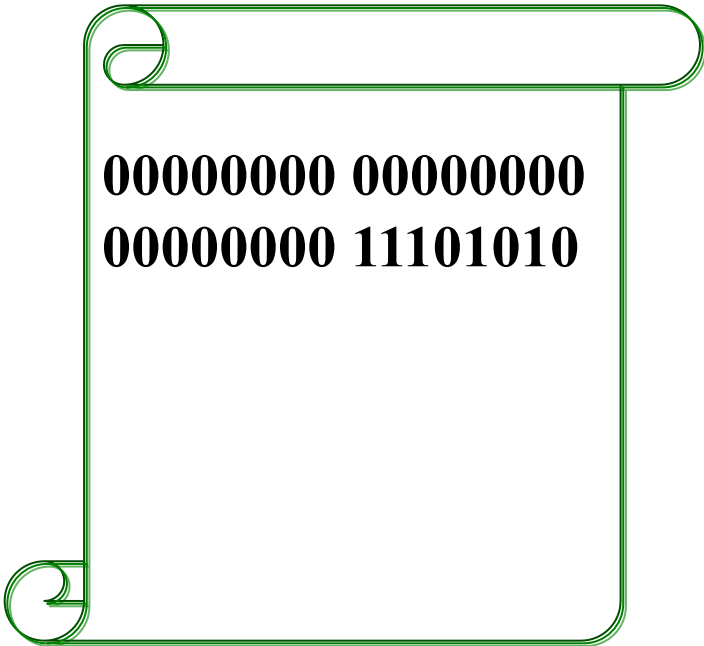
北京邮电大学
Beijing University of Posts and Telecommunications

- 整数234的存储



00110010 00110011
00110100

ASCII文件



00000000 00000000
00000000 11101010

二进制文件

13.2 文件概述-文件分类



(3) 根据文件的组织形式，可分为顺序存取文件和随机存取文件。

顺序存取文件 (Sequential Access File)

简称“顺序文件”，数据写入文件的方式是后输入的数据放在以前输入数据的后面，按照数据的先后次序一个接一个的放。若要读取数据，也是由第一条记录开始读取。

随机存取文件 (Random Access File)

简称“随机文件”，数据不必按顺序读写，可以直接访问任意位置。

13.2 文件概述-文件分类



(4) 从用户角度，可分为普通文件和设备文件。

普通文件是指存放在外部存储介质上的数据集合。

设备文件是指与主机相连的各种外部设备，如显示器、键盘、打印机等。操作系统中为了处理的统一和概念的简化，把外部设备也作为文件来进行管理，对他们的输入输出等同于对普通文件的读和写。通常，显示器定义为标准输出文件，键盘定义为标准输入文件。



13.2 文件概述

3. 文件的作用

- (1) 同时处理大量的数据；
- (2) 文件中的数据可以多次重复使用；
- (3) 文件可以永久保存，其中的数据不会因为应用程序的结束或关机而消失；
- (4) 文件中的数据可以为多个应用程序所共享；
- (5) 便于网络传输。

4. 读文件与写文件（数据在内存和文件之间的传送）

读文件：将文件中的数据传送到计算机内存。

写文件：将计算机内存中的数据传送到文件。



13.2 文件概述

5. 文件和流

不论是普通文件还是设备文件，C语言把每个文件一律都看作是一个有序的**字节流**，以字节为单位进行操作处理。

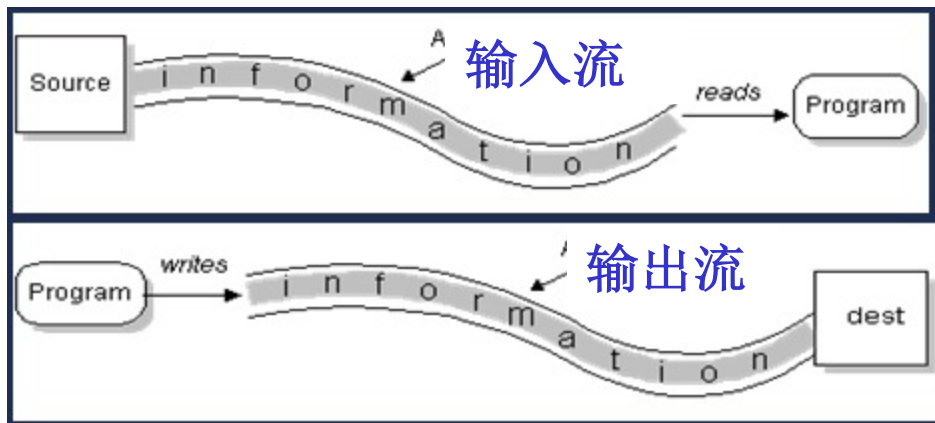
0	1	2	3	4	5	6	7	8	9	10	...	n-1	
											...		end of file marker

文件的结束：以文件结束标志（end-of-file marker）结束，
或者在特定字节号处结束（字节号记录在系统维护和管理的数据结构中）

13.2 文件概述



流是文件和程序之间通讯的通道。当打开一个文件时，该文件就和某个流关联起来了,当关闭文件时，将会取消流和文件的连接。



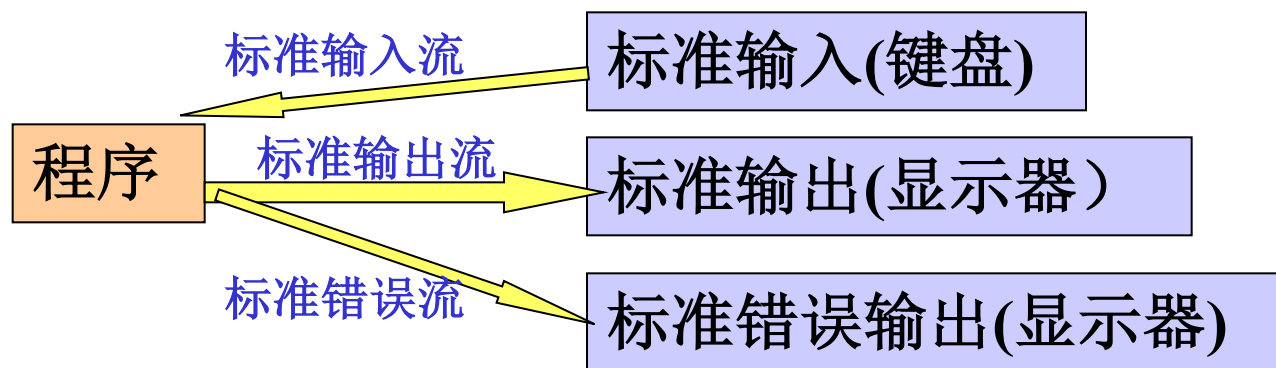
执行程序时会自动打开三种文件以及和这三种文件关联的流——标准输入流、标准输出流和标准错误流。

13.2 文件概述

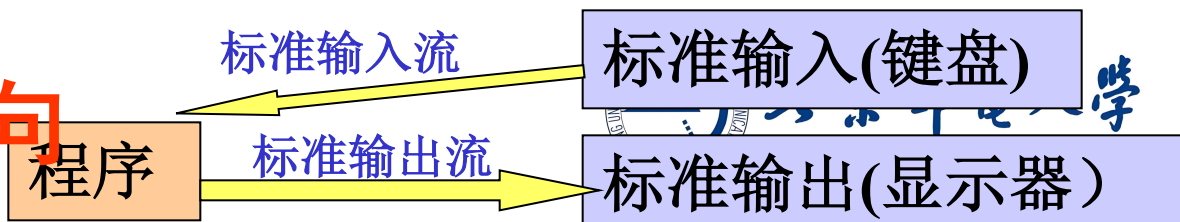
操作系统把键盘、显示器也抽象为文件。

在程序开始运行时，系统自动打开三个流：

1. **标准输入流**：指向终端输入（一般为键盘，可重定向）。
该流能够使程序读取来自终端输入的数据。
2. **标准输出流**：指向终端输出（一般为显示器，可重定向）。
该流能够使程序把数据打印到终端输出上。
3. **标准错误流**：指向终端标准错误输出（一般为显示器）。



输入输出重定向



```
#include<stdio.h>
int main()
{
    int i;
    int a[5];
    printf("input 5 numbers:\n");
    for(i=0;i<5;i++)
        scanf("%d",&a[i]);

    printf("the 5 numbers are:\n");
    for(i=0;i<5;i++)
        printf("%d,",a[i]);

    return 0;
}
```

运行结果

```
input 5 numbers:
10 20 30 40 50
the 5 numbers are:
10,20,30,40,50,请按任意
键继续...
```

默认情况下，标准输入设备是键盘，标准输出设备是显示器

10 20 30 40 50

data1.txt

data2.txt

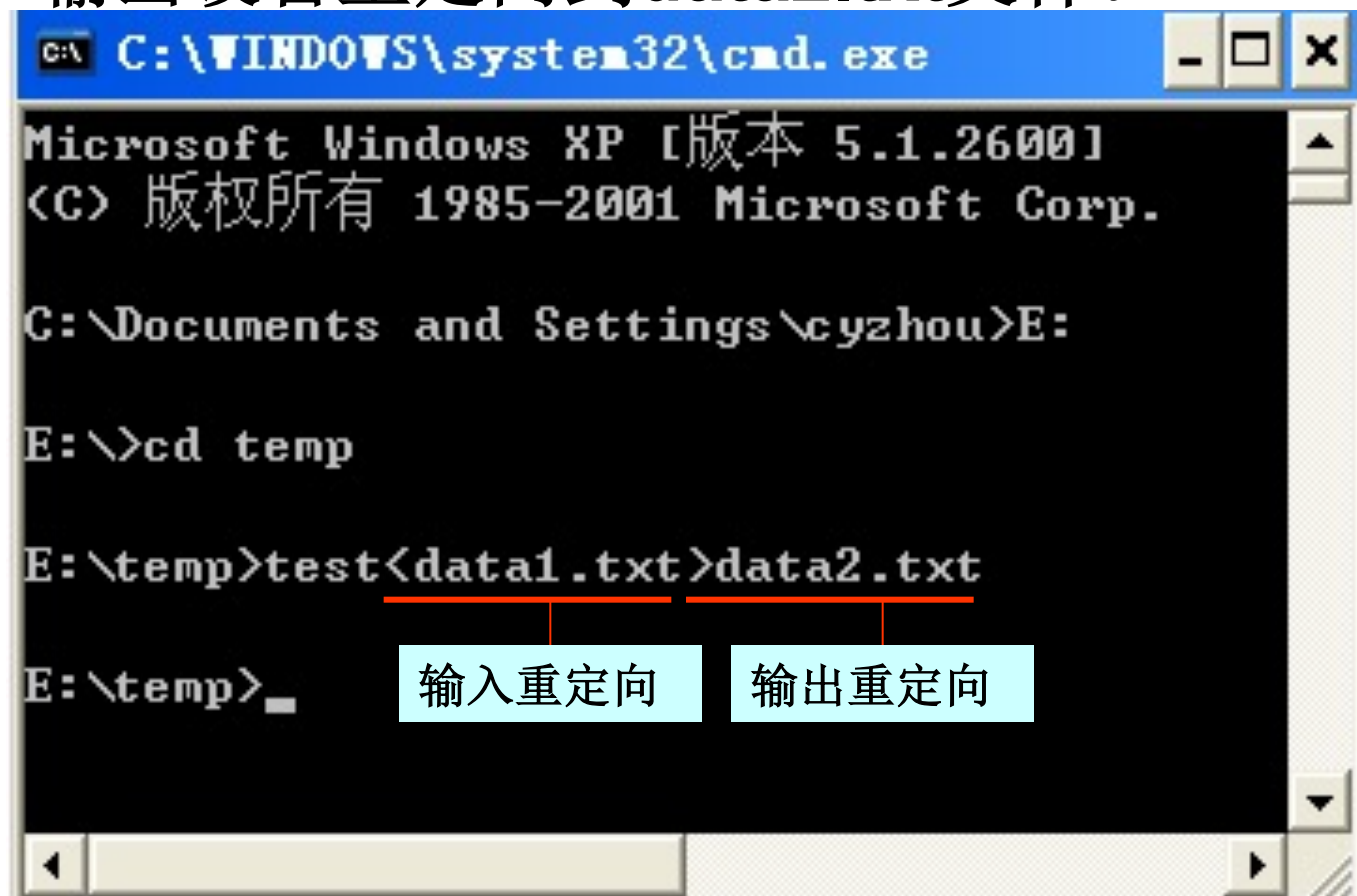
请按任意键继续...

input 5 numbers:

the 5 numbers are:

11,21,31,41,51,

将标准输入设备重定向到**data1.txt**文件，标准输出设备重定向到**data2.txt**文件。



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [版本 5.1.2600]
(C) 版权所有 1985-2001 Microsoft Corp.

C:\Documents and Settings\cyzhou>E:

E:\>cd temp

E:\temp>test<data1.txt>data2.txt

E:\temp>_
```

输入重定向

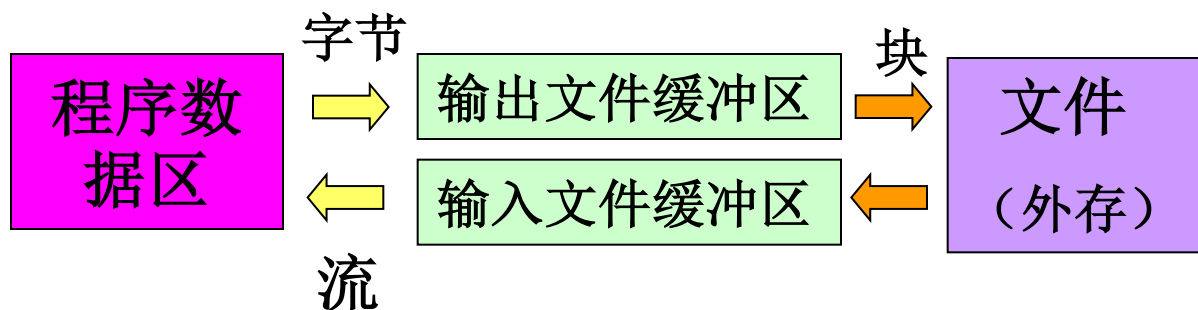
输出重定向

13.2 文件概述-ANSI C的缓冲文件系统

6. ANSI C的缓冲文件系统

缓冲文件系统：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。

缓冲区在面向字节的流和面向块的磁盘之间扮演着接口的角色。



写文件时，数据首先被写到内存中的文件缓冲区，当缓冲区被写满时，缓冲区中的全部内容将作为一个数据块被写入磁盘。读文件时，首先会到缓冲区中读，如果要读的数据在缓冲区中便读取出来，如果缓冲区中没有，就从磁盘上将该数据所在的块整个读到缓冲区中。

缓冲区操作函数



- **int fflush(FILE *fPtr);**
 - 清理一个文件的缓冲区。
 - 参数**fPtr**是打开文件时由**fopen()**函数返回的**FILE**指针。
 - 对于以写方式打开的文件，函数将缓冲区中的数据写入磁盘文件。
 - 对于以读方式打开的文件，函数将缓冲区清空。
 - 函数如果运行成功则返回**0**，如果失败则返回**EOF**（也就是**-1**）。
- **int flushall(void);**
 - 用于清理所有文件的缓冲区。
 - 返回打开着的文件个数。



13.2 文件概述

7. 文件类型FILE

C系统为了处理文件，给每个打开的文件都在内存中开辟一片区域，用于存放文件的有关信息（如缓冲区位置、缓冲区大小、缓冲区中当前所指向的字节位置、文件状态--读还是写、是否已经到达文件结尾等等）。

为了得到这片内存，需要在程序中定义一个类型为**FILE**的结构变量。**FILE**类型由系统定义（见**<stdio.h>**）。

注意：结构类型名“**FILE**”必须大写。




FILE声明示例

不同编译器中FILE类型包含的内容不完全相同，但大同小异。

FILE记录

```
typedef struct {  
    short  level; /* fill/empty level of buffer */  
    unsigned  flags; /* File status flags */  
    char  fd; /* File descriptor */  
    unsigned char  hold; /* Ungetc char if no buffer */  
    short  bsize; /* Buffer size */  
    unsigned char  *buffer; /* Data transfer buffer */  
    unsigned char  *curp; /* Current active pointer */  
    unsigned  istemp; /* Temporary file indicator */  
    short  token; /* Used for validity checking */  
} FILE;
```

A vertical stack of six pink rectangular boxes of varying heights, representing the memory layout of the FILE structure. From top to bottom, the boxes correspond to: 'level' (short), 'flags' (unsigned), 'fd' (char), 'hold' (unsigned char), 'buffer' (unsigned char *), and 'curp' (unsigned char *). The 'buffer' and 'curp' boxes are the tallest, while 'fd' and 'hold' are the shortest.



13.2 文件概述

8、对文件的操作

任何高级语言对文件的操作都应该遵循下列过程：

打开文件—>读或写文件—>关闭文件

FILE * fPtr;

fPtr = fopen (“clients.dat” , “w”);

用**fPtr**指向要操作的文件

标准输入流指针：**stdin**

标准输出流指针：**stdout**

标准错误流指针：**stderr**



- 重要概念回顾：
 - 文本文件、二进制文件
 - 顺序存取文件、随机存取文件
 - 普通文件、设备文件
 - 流
 - 文件缓冲区
 - 文件类型**FILE**

- 1、数据的层次结构
- 2、文件概述
- 3、文件的打开和关闭
- 4、位置指针与文件定位
- 5、文件的读写操作
- 6、顺序文件的操作
- 7、随机文件的操作





13.3 文件的打开与关闭

对文件进行操作之前，必须先打开该文件；使用结束后，应立即关闭，以免数据丢失。

C语言规定了标准输入输出函数库，用**fopen()**函数打开一个文件，用**fclose()**函数关闭一个文件。

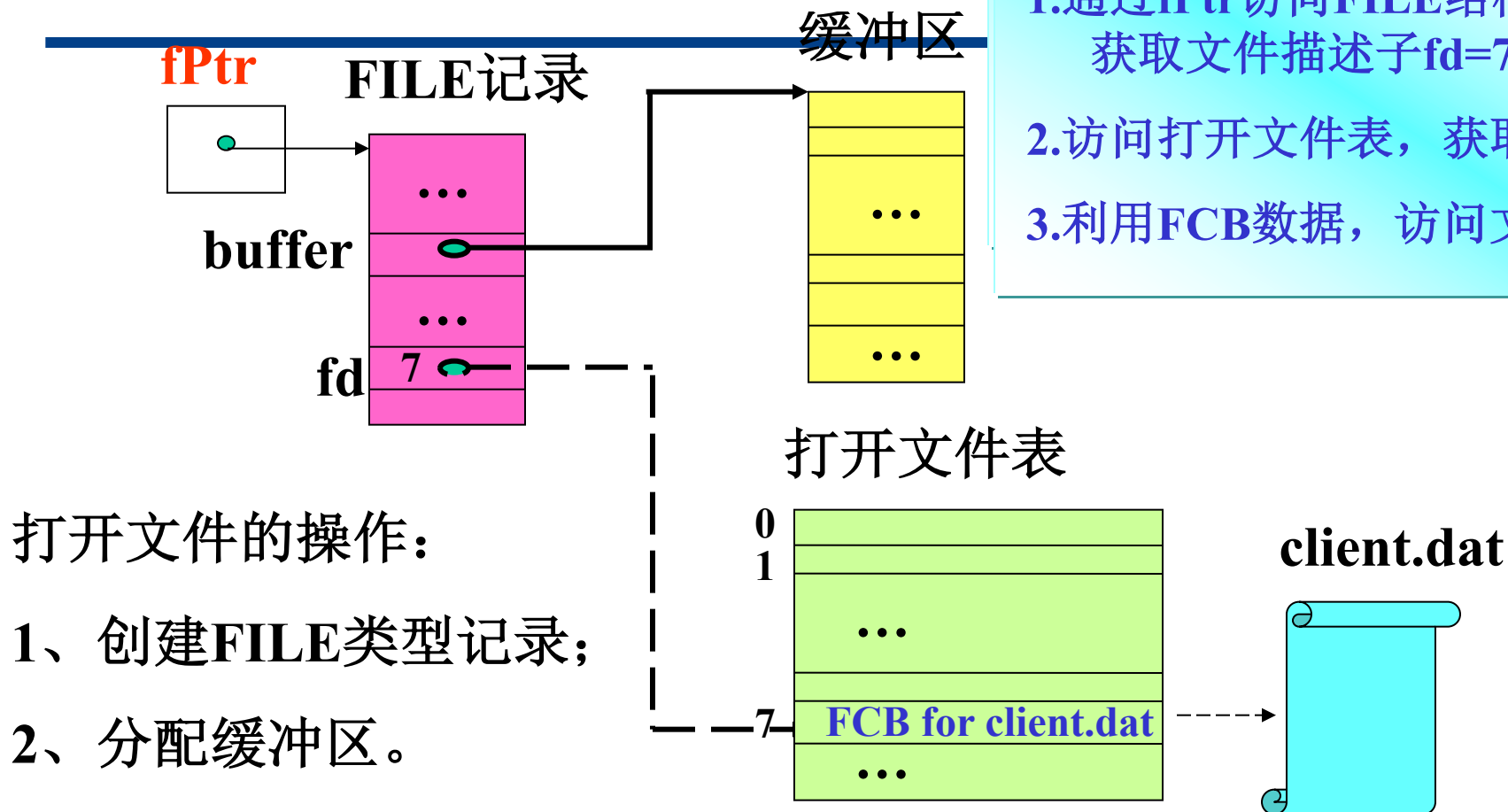
一、文件的打开——fopen()函数

1. 用法： **FILE *fopen(const char * filename, const char * mode);**
2. 功能：打开以**filename**所指向的字符串为文件名的文件，使之与一个流关联。返回一个指向**FILE**类型结构变量的指针（后称**文件指针**）。若打开失败，返回一个空指针**NULL**。
3. 函数原型：**stdio.h** 。
4. **FILE * fPtr;**
fPtr = fopen (“clients.dat” , “w”); //文件和程序在同一目录
fPtr = fopen (“C:\\temp\\clients.dat” , “w”);
//文件绝对路径。注意：路径中必须是“\\”而不是“\”

文件控制块(FCB). 是操作系统通过fPtr操作文件的过程:

- 1.通过fPtr访问FILE结构变量, 获取文件描述子fd=7。
- 2.访问打开文件表, 获取FCB。
- 3.利用FCB数据, 访问文件。

fPtr = fopen ("clients.dat","w");
用fPtr来引用文件



打开文件的操作:

- 1、创建FILE类型记录;
- 2、分配缓冲区。
- 3、拷贝磁盘上的FCB到内存的打开文件表。
- 4、将缓冲区首地址、文件控制块(FCB)在打开文件表中的下标写入FILE记录

13.3 文件的打开与关闭



文件控制块(FCB)：是操作系统为管理文件而设置的数据结构，存放了为管理文件所需的所有有关信息（文件属性）。是文件存在的标志。操作系统要依靠FCB中的数据完成对文件的读或写操作。

包含内容：文件名，文件号，用户名，文件地址，文件长度，文件类型，文件属性，共享计数，文件的建立日期，保存期限，最后修改日期，最后访问日期，口令，文件逻辑结构，文件物理结构等。



13.3 文件的打开与关闭

打开文件的操作方式

模式	描述
r	打开一个文本文件，可以读取文件。
w	打开一个文本文件，可以写入文件（但会先删除文件原内容）。如果文件不存在，则创建。
a	打开一个文本文件，可以写入文件，向已有文件的尾部追加内容，如果该文件不存在则创建之
r+	打开一个文本文件，可以进行更新（即读和写）
w+	打开一个文本文件，可以进行更新（即读和写）（但会先删除文件原内容）。如果文件不存在，则创建之。
a+	打开一个文本文件，可以进行更新（即读和写），向已有文件的尾部增加内容）。如果文件不存在，则创建之。 可以读取整个文件，但写入时只能追加数据到文件末尾。
rb/wb/ab/r+ b/w+b/a+b	对二进制文件的操作。 r 、 w 、 a 、 + 的含义同ASCII文件操作。



13.3 文件的打开与关闭

(1)如果不能打开指定的文件，则**fopen()**函数返回一个空指针**NULL**（其值在头文件**stdio.h**中被定义为0）。

为增强程序的可靠性，常用下面的方法打开一个文件：

```
if( (fPtr=fopen( "clients.dat" , "w" ) )==NULL)  
    printf("File could not be opened\n");  
else
```

.....

(2) **C**程序使用一个**\n**表示行尾，**Windows**文本文件使用回车换行的组合**\r\n**来表示行尾。**C**程序读取文本文件时，会将**\r\n**转换为**\n**；而写入文件时，会将**\n**转换为**\r\n**。

(3)对文件操作的库函数，函数原型均在头文件**stdio.h**中。后续函数不再赘述。



13.3 文件的打开与关闭

二、文件的关闭——fclose()函数

1. 用法: `int fclose(FILE * stream);`
2. 功能: 关闭文件指针**stream**所指向的文件, 将所有未回写缓冲数据写入文件, 断开文件和流的关联, 释放文件占用的资源, 如缓冲区、**FILE**类型记录占用的内存等。如果正常关闭了文件, 则函数返回值为 0 ; 否则, 返回**EOF**。

例如, `fclose(fPtr); /*关闭fPtr所指向的文件*/`

