第 10 讲

磁盘数据组织：文件

主要内容

10.1 文件概述

10.2 文件的打开与关闭

10.3 文件的操作

10.4 文件的随机读写与状态跟踪

10.1 文件概述

文件的概念

- 程序中的常量及变量的值只有在程序运行时有效，程序结束后即消失。要永久保存这些数据，只能通过文件的形式
- 文件是一组相关数据的有序集合，通常存放在外部介质（磁盘等）上，需要时被调入内存中

- 文件是一个逻辑概念，除了磁盘文件之外，操作系统将显示器、打印机、键盘等外部设备都当做文件处理（**设备文件**），把对它们的输入/出操作等同于对磁盘文件的操作
- 高级计算机语言中，把不同对象，如键盘、文件、显示器、打印机、网络连接等的输入/出，抽象表述为“**流**”（stream）。通过流的方式，使得程序能以相同的方式完成不同对象的输入/出操作

- 流是一种抽象，它负责数据源（数据的生产者）和数据的目的地（数据的消费者）之间建立联系、并管理数据的流动
- C程序开始运行时，系统自动打开3个标准流
  - ✓ 标准输入流（stdin）：scanf、getchar、gets等从这个流中读取数据
  - ✓ 标准输出流（stdout）：printf、putchar、puts等将输出数据写入这个流
  - ✓ 标准错误输出流（stderr）：程序出错时将出错信息输出到终端

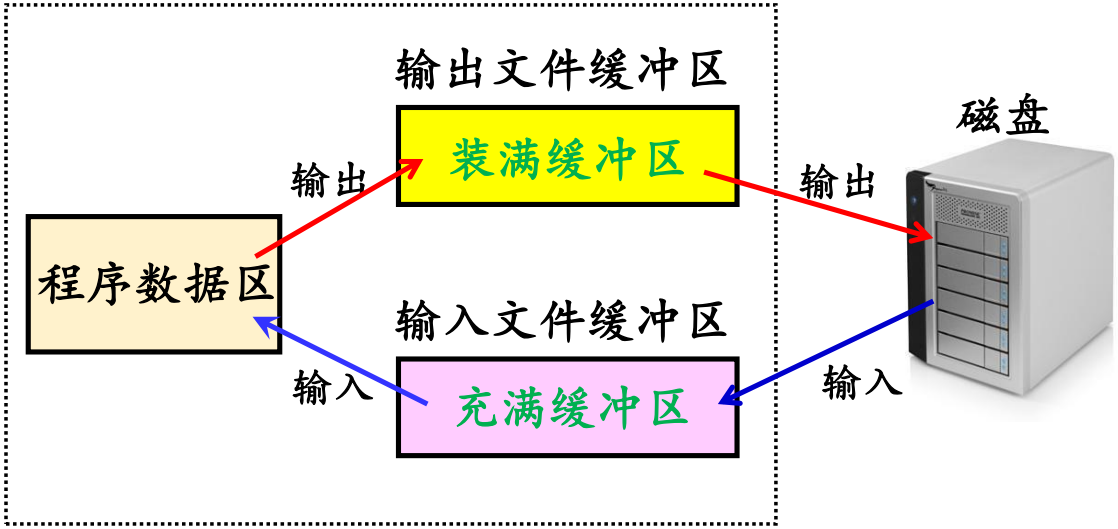
文件的分类

- 按逻辑结构：记录文件、流式文件
- 按存储介质：普通文件、设备文件
- 按数据存储：文本文件、二进制文件
- 按数据组织：顺序文件、随机文件

缓冲文件系统

- 缓冲文件系统：系统自动在内存区为每一个正在使用的文件开辟一个文件缓冲区，用于进行文件读写时数据的暂存
- 非缓冲文件系统：不由系统自动设置缓冲区，而由用户自己根据需要设置
- 一般把缓冲文件系统的输入输出称为**标准输入输出**（标准I/O），非缓冲文件系统的输入输出称为**系统输入输出**（系统I/O）

- ANSI C采用缓冲文件系统
- 从磁盘向内存读入数据时，一次从磁盘文件将一些数据输入到文件缓冲区（充满缓冲区），然后再从缓冲区逐个地将数据送给接收变量
- 向磁盘文件输出数据时，先将数据送到文件缓冲区，装满缓冲区后才一起送到磁盘去


输出文件缓冲区
装满缓冲区
输出
输出
磁盘
程序数据区
输入文件缓冲区
输入
充满缓冲区
输入

文件指针

- 缓冲文件系统为每个打开的文件在内存中开辟了一个“**文件信息区**”，用来存放文件名、文件操作方式、文件状态和当前读写位置等信息
- 为对文件进行操作，需定义一个指针变量（文件指针），通过文件指针可以找到该文件的文件信息区，从而实施对文件的操作

10.2 文件的打开与关闭

文件指针

- C语言中，文件的相关信息保存在**FILE类型**的结构体变量中，FILE类型的定义包含在stdio.h中

```
typedef struct
{
  short  level;  //缓冲区“满”或“空”的程度
  unsigned  flags;  //文件状态标志
  char  fd;   //文件号
  unsigned char  hold;  //如无缓冲区不读取字符
  short  bsize;    //缓冲区的大小
  unsigned char  *baffer;   //数据缓冲区的读写位置
  unsigned char  *curp;  //指针指向的当前文件的读写位置
  unsigned  istemp;   //临时的文件指示器
  short  token;    //用于有效性检查
}FILE;
```

FILE *fp;

- fp是FILE类型的指针变量，通过fp可以找到存放某个文件信息的结构体变量，按该结构体变量提供的信息找到该文件，实施对文件的操作
- 文件中数据流的结束标志为EOF，其定义包含在stdio.h

#define EOF -1

文件的打开和关闭

- C中对文件的操作通过函数实现，具体过程为：打开文件、文件操作、关闭文件
- 文件打开：系统在内存中开辟一个文件信息区，并使文件指针指向该区域

格式：文件指针名=fopen(文件名, 打开方式);

如：**FILE *fp;**

fp=fopen(“D:\\CAI\\myfile.dat”, “rb”);

使用方式	含义	如果文件存在	如果文件不存在
r	只读	仅允许从文件中读取数据	出错
w	只写	仅允许向文件写数据（清空原数据）	创建新文件
a	追加	仅允许向文件尾部追加数据	创建新文件
r+	读写	允许读写数据（从文件头开始）	出错
w+	读写	允许读写数据（清空原数据）	创建新文件
a+	读写	允许读写数据（追加）	创建新文件
rb	只读	仅允许从文件中读取数据	出错
wb	只写	仅允许向文件写数据（清空原数据）	创建新文件
ab	追加	仅允许向文件尾部追加数据	创建新文件
rb+	读写	允许读写数据（从文件头开始）	出错
wb+	读写	允许读写数据（清空原数据）	创建新文件
ab+	读写	允许读写数据（追加）	创建新文件

- 文件关闭：确保所有数据进行正确的操作，并删除与当前文件相关的内存空间

  格式：`int fclose(文件指针);`

  如：`fclose(fp);`

  说明：

  （1）如果不正确关闭，则可能会造成数据的丢失

  （2）关闭成功，返回0；不成功，返回EOF(-1)

10.3 文件的操作

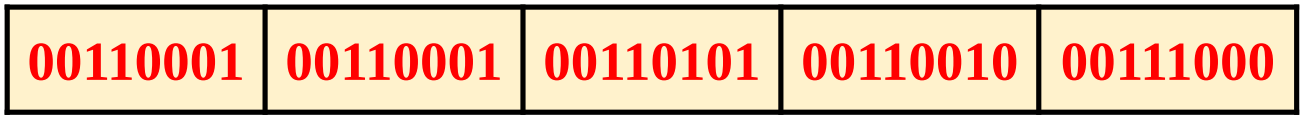
文本文件 与 二进制文件

- **文本文件**
  - ✓直接以字符形式进行存储的文件
  - ✓每个字符一个字节，存储对应的ASCII码值
  - ✓便于阅读和编辑，但占用的存储空间较大，且由于需要进行字符与二进制的转换，效率较低

二进制文件

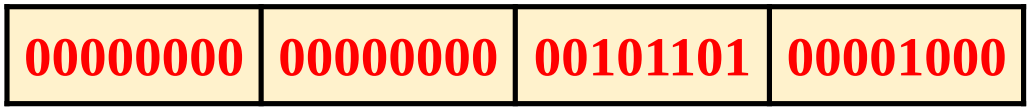
- ✓将内存中的数据（二进制）不进行任何转换，严格按照其在内存中的形式保存的文件
- ✓占用的存储空间较小，输入输出速度快
- ✓无法直接读懂，主要用于暂存程序的中间结果，以供另一个程序读取
- ✓C语言在处理文本文件及二进制文件时，并不区分类型，都看成字符流，按字节进行处理

文本文件



‘1’ ‘1’ ‘5’ ‘2’ ‘8’

二进制文件



1 1 5 2 8

文件的顺序读写

- 将文件从头到尾逐个数据读出或写入
  - ✓ 字符读写：**fgetc、fputc**
  - ✓ 字符串读写：**fgets、fputs**
  - ✓ 格式化读写：**fscanf、fprintf**
  - ✓ 数据块读写：**fread、fwrite**

  以上函数的原型在**stdio.h**中

函数原型	**int fgetc(FILE *fp);**
函数功能	从fp所指向的文件中读取一个字符
参数	fp—要读取的文件
返回值	成　功：所读取的字符 不成功：EOF
头文件	#include<stdio.h>
说明：	• 读取的文件应以只读或读写方式打开 • 读取成功后，位置标记向后移动一个字符 应用举例：ch=fgetc(fp); //从指向的文件中读取一个字符赋给变量ch

函数原型	**int fputc(int ch, FILE *fp);**
函数功能	将字符ch输出到fp所指向的文件中
参数	ch—要写入的字符；fp—要写入的文件
返回值	成　功：所写入的字符 不成功：EOF
头文件	#include<stdio.h>
说明：	• 被写入的文件应以只写、读写或追加方式打开 • 写入成功后，位置标记向后移动一个字符 应用举例：fputc('A', fp); // 向fp指向的文件中写入一个字符A

```
#include“stdio.h"     // 例10.1
#include"stdlib.h"
void main( )
{       FILE *fpin, *fpout;
        char ch;
        int i=100;
        if((fpin=fopen("F:\\CAI\\计算机程序设计\\file_source.dat", "r+"))==NULL)
          { printf("文件打开失败！ \n"); exit(0);}
        if((fpout=fopen("F:\\CAI\\计算机程序设计\\file_dest.dat", "w+"))==NULL)
          { printf("文件打开失败！ \n"); exit(0);}
        fprintf(fpout, "%-4d", i*100);
        while((ch=fgetc(fpin))!=EOF)
        {  fputc(ch, fpout);
           if(ch=='\n'||ch=='\r')  fprintf(fpout, "%-4d", 100+(i++)*10);
        }
        fclose(fpout);   fclose(fpin);
}
```

函数原型	**char *fgets(char *str, int n, FILE *fp);**
函数功能	从fp所指向的文件中读取n-1个字符，存入起始地址为str的空间
参数	str—拟存入空间的地址；fp—要读取的文件； n—读取的字符数
返回值	成功:字符指针str的值;文件结束或出错:NULL
头文件	#include<stdio.h>
说明：	• 读取的文件应以只读或读写方式打开 • 读取n-1个字符，提前遇到‘\0’，或提前遇到EOF时读取结束 应用举例：char str[10]; fgets(str, 8, fp);

函数原型	**int fputs(const char *str, FILE *fp);**
函数功能	将str所指向的字符串写入到fp所指向的文件
参数	str—字符串常量、字符数组名或字符指针 fp—要写入的文件
返回值	成功：0 不成功：EOF
头文件	#include<stdio.h>
说明：	• 被写入的文件应以只写、读写或追加方式打开 • 将字符串写入到fp指向的文件的指定位置，文件位置标记向后移动到写入字符串之后的位置 应用举例：fputs(“Ecust”, fp);

函数原型	**int fscanf(FILE *fp, const char *format, args, …);**
函数功能	**从fp所指向的文件中按指定的格式读取数据送到args所指向的内存单元**
参数	**fp—要读取的文件；format—格式字符串； args—要存放的位置地址**
返回值	**成功：已读取的数据个数；不成功：EOF**
头文件	**#include<stdio.h>**
说明：	• **读取的文件应以只读或读写方式打开** • **读取成功后，文件位置标记向后移动到最后读取的数据所处位置之后的位置** **应用举例：fscanf(fp, “%d,%c,%f”,&i,&ch,&f);**

函数原型	**int fprintf(FILE *fp, const char *format, args, …);**
函数功能	**将args的值以format指定格式写入fp指向的文件**
参数	**fp—要写入的文件；format—格式字符串； args—要输出的数据**
返回值	**成功：实际写入的数据个数；不成功：负值**
头文件	**#include<stdio.h>**
说明：	• **被写入的文件应以只写、读写或追加方式打开** • **写入成功后，文件位置标记向后移动到最后读写入的数据所处位置之后的位置** **应用举例：fprintf(fp, "%d,%c,%f", i, ch, f);**

函数原型	**unsigned fread(void *ptr, unsigned size, unsigned count, FILE *fp);**
函数功能	从fp所指向的文件中读取长度为size的最多count个数据项，送到ptr所指向的内存区
参数	fp—要读取的文件；count—数据项的个数； size—每个数据项的长度；ptr—要存放的位置地址
返回值	成功：实际读取的数据项个数；不成功：0
头文件	#include<stdio.h>
说明：	• 读取的文件应以只读或读写方式打开 • 实际读取的数据项个数可能小于count • 读取成功后，文件位置标记向后移动实际读取数据项个数*size字节。该函数以二进制方式进行读取 应用举例：float buf[10]; fread(buf, 4, 10, fp);

函数原型	**unsigned fwrite(const void *ptr, unsigned size, unsigned count, FILE *fp);**
函数功能	将ptr所指向的最多count*size个字节的数据输出到fp所指向的文件中
参数	fp—要写入的文件；count—数据项的个数； size—每个数据项的长度；ptr—要输出的数据项的地址
返回值	成功：实际写入的数据项个数；不成功：0
头文件	#include<stdio.h>
说明：	• 被写入的文件应以只写、读写或追加方式打开 • 实际输出的数据项个数可能小于count • 输出成功后，文件位置标记向后移动实际读取数据项个数*size字节。该函数以二进制方式进行写操作 应用举例：float buf[10]; fwrite(buf, 4, 10, fp);

例10.2 从键盘上输入以下10名学生的学号、姓名，以及数学、英语、物理课程成绩，写到文本文件score.dat，再从文件中取出数据，计算每个学生的总成绩和平均分，并将结果显示在屏幕上。

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
    char filename[30],stuName[10];
    int stuNum,Math,Eng,Phys,total,i;
    float aver;
    FILE *fp;
    printf("Input the name of file that you want to open:");
    gets(filename);
```

```
if((fp=fopen(filename,"w"))==NULL)
    { printf("File open error!\n");  exit(0);  }
printf("Input the information of each students (No.  Name  Math  Eng
            Phys):\n");
for(i=0;i<10;i++)
{
   scanf("%d%s%d%d%d",&stuNum,stuName,&Math,&Eng,&Phys);
   fprintf(fp,"%3d%8s%5d%5d%5d\n", stuNum, stuName, Math, Eng,
                Phys);
}
fclose(fp);
```

```
if((fp=fopen(filename,"r"))==NULL)
   { printf("Can't Open File!");  exit(0); }
printf("学号  姓名 数学 英语 物理 总分 平均分\n");
fscanf(fp,"%d%s%d%d%d",&stuNum,stuName,&Math,&Eng,&Phys);
while(!feof(fp)){
   total=Math+Eng+Phys;
   aver=total/3.0;printf("%3d%8s%5d%5d%5d%6d%7.1f\n",
                          stuNum,stuName,Math,Eng,Phys, total, aver);
    fscanf(fp,"%d%s%d%d%d",&stuNum,stuName,&Math,&Eng,&Phys);
  }
  fclose(fp);
}
```

```
Input the name of file that you want to open:score.dat
Input the information of each students (No.  Name  Math  Eng  Phys):
101    张三   81   75   82
102    李四   87   68   85
103    王五   73   84   80
104    赵六   76   81  100
105    孙七   83   75   71
106    陈八   89   78   91
107    钱九   82   80   62
108    方十   60   87   98
109    丁一   89   75   92
110    曹二   93   85  100
---------读取文件内容并计算输出---------
学号    姓名   数学 英语 物理   总分  平均分
101    张三   81   75   82   238   79.3
102    李四   87   68   85   240   80.0
103    王五   73   84   80   237   79.0
104    赵六   76   81  100   257   85.7
105    孙七   83   75   71   229   76.3
106    陈八   89   78   91   258   86.0
107    钱九   82   80   62   224   74.7
108    方十   60   87   98   245   81.7
109    丁一   89   75   92   256   85.3
110    曹二   93   85  100   278   92.7
```

10.4 文件的随机读写与状态跟踪

文件的定位

- **ftell**
  - ✓获取文件读写指针当前位置
  - ✓原型： **long ftell(FILE *fp);**

    如： **long n; n=ftell(fp);**

- **rewind**
  - ✓把文件的位置指针移到文件头
  - ✓原型：　**void rewind(FILE *fp);**
    如：　　**rewind(fp);**

fseek

- ✓用于将读写位置指针移到到需要的位置
- ✓原型：

```
int  fseek(FILE *fp, long offset, int base);
```

如：

```
fseek(fp, 10L, 0);  //0:文件头
fseek(fp, 20L, 1);  //1:当前位置
fseek(fp, -30L, 2);  //2:文件尾
```

文件的状态跟踪

- **int feof(FILE *fp);**
  - ✓ 判断文件是否处于结束位置，是返回非0值，否则0
- **int ferror(FILE *fp);**
  - ✓ 检查文件在用各种输入输出函数进行读写时是否出错，未出错返回0，否则非0
- **void clearerr(FILE *fp);**
  - ✓ 使文件错误标志和文件结束标志置为0。假设在调用一个输入输出函数时出现了错误，ferror函数值为一个非零值。在调用clearerr(fp)后，ferror(fp)值变为0

Homework

- 实践

实验7 文件程序设计