高级语言程序设计

第11章　文件

内容提要

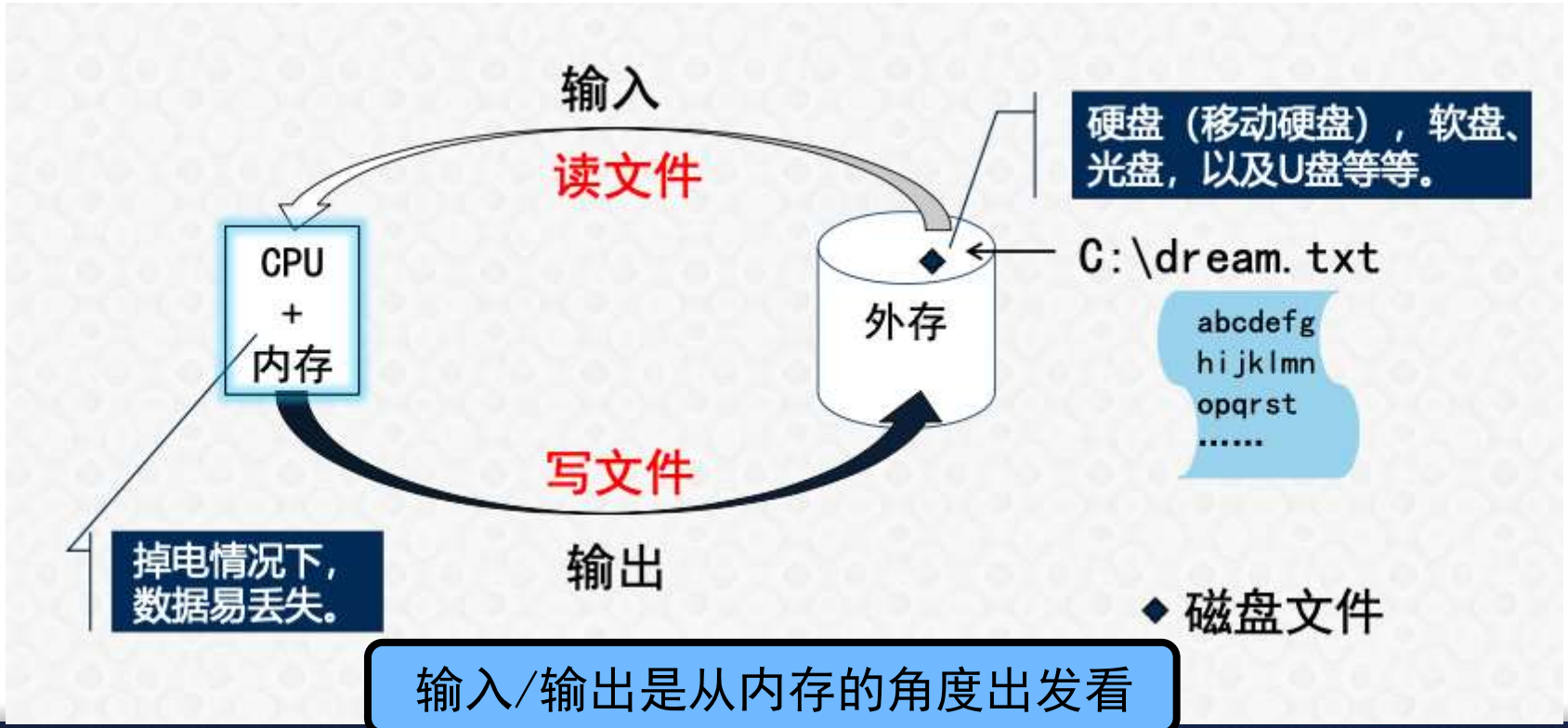
- 文件与文件指针
- 文件的打开和关闭
- 文件读写（4对函数）
- 文件的随机读写与位置指针的定位
- 应用举例——文件的综合操作

文件与文件指针

- 计算机以文件形式管理所有的软硬件资源
- 不同类型的文件其格式不同，后缀名不同，作用、特征、打开方式也不相同，例如：
  - 音频文件： wav、mp3、wma…
  - 视频文件： rm、mp4…
  - 图形文件：bmp、gif、 jpg、tif
  - 可执行文件：exe、com
  - 源代码文件：.c、.cpp、.pas、java、py…
  - 压缩文件：rar、zip、 arj…
  - ……

文件与文件指针

- 文件的操作只有两种：
  - 读文件：将数据从磁盘文件读取至内存
  - 写文件：将数据从内存存放至磁盘文件上



输入/输出是从内存的角度出发看

文件与文件指针

- 各种文件**本质上**一致
- 都可看作是**一系列的数据**按照某种次序组织起来的**数据流**

➢ 根据数据的组成形式，C语言将文件分为两种：

文本文件：

- 字符序列
- 以ASCII字符的形式存放

二进制文件：

- 字节序列
- 以在内存中的形式（即二进制形式）存放

文件与文件指针

短整型 127：

- **文本文件存储：**

0	0	1	1	0	0	0	1

'1' 49

0	0	1	1	0	0	1	0

'2' 50

0	0	1	1	0	1	1	1

'7' 55

- **二进制文件存储：**

0	0	0	0	0	0	0	0

0	1	1	1	1	1	1	1

127

数字比较多，选择用二进制存储
字符比较多，选择用文本文件存储

文件与文件指针

- 对一个文件进行操作时，计算机会将该文件的相关信息，如文件状态、文件在内存中的缓冲区大小等，保存在一个**结构体类型的变量**中。
- 该结构体类型是系统预先定义的，名为FILE，其类型定义如下页所示。

文件与文件指针

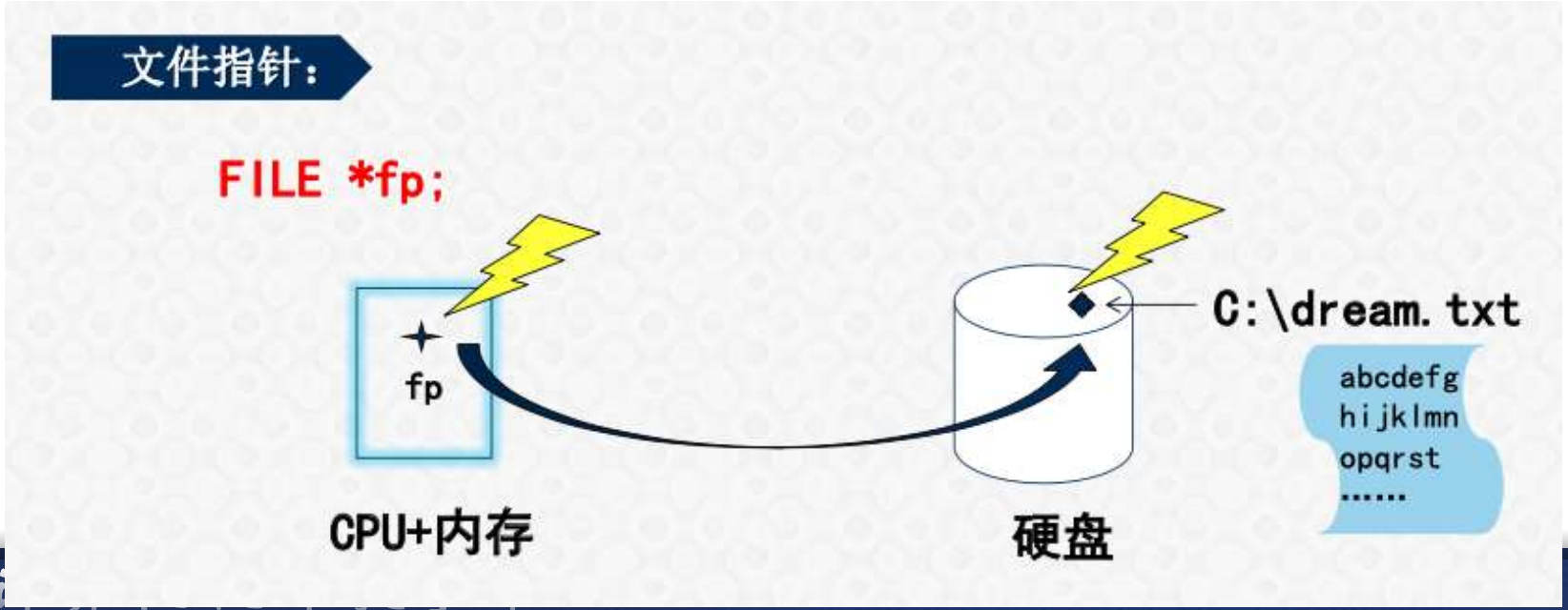

```
struct FILE
{   short level;                /* 缓冲区使用程度 */
    unsigned  flags;            /* 文件状态标志 */
    char      fd;               /* 文件描述符 */
    unsigned char  hold;        /* 若无文件缓冲区,
                                   则不读取数据 */
    short     bsize;            /* 缓冲区大小 */
    unsigned char  *buffer;     /* 缓冲区位置 */
    unsigned char  *curp;       /* 指向缓冲区当前
                                   数据的指针 */
    unsigned  istemp;           /* 临时文件指示器 */
    short  token;               /* 用于有效性检验 */
};
```

文件与文件指针

- 因此，如果编程对一个文件进行操作，用户需要定义1个FILE类型的指针，也称为文件指针：

  FILE * fp;

- 将fp指向文件对应的结构体变量，就可以获取该文件的相关信息，进而进行文件的读写操作



文件的打开和关闭

- 文件与文件指针
- 文件操作的完整过程
- 文件的读写操作

文件的打开和关闭

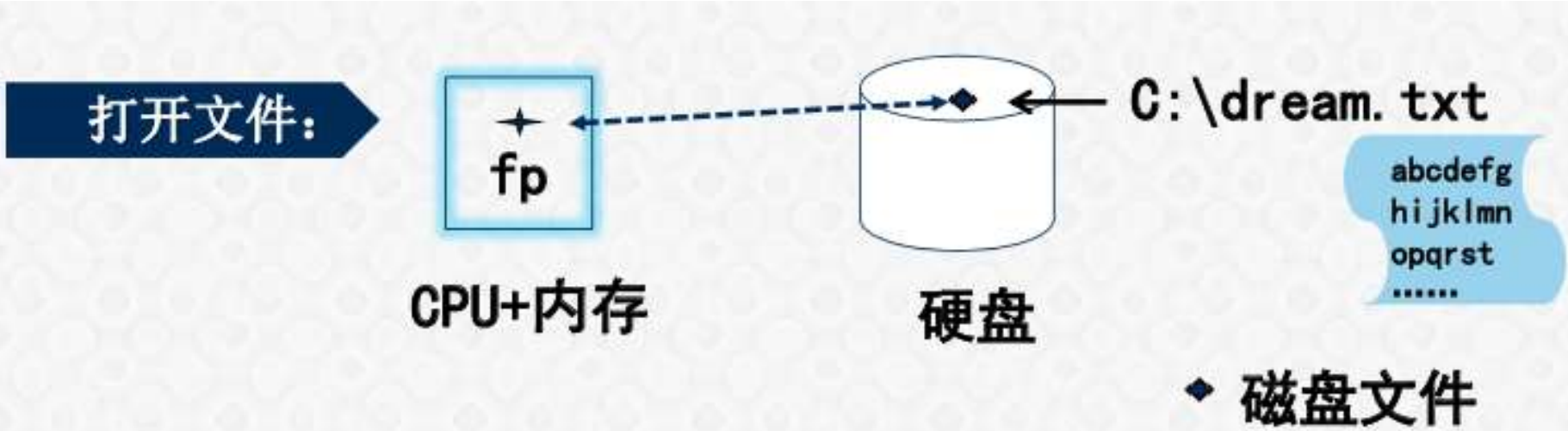
步骤：

FILE *fp;

1. 定义文件指针
2. 打开文件 —— 指针与文件关联
3. 对文件进行读写操作
4. 关闭文件 —— 断开关联

文件的打开和关闭

- **打开文件**就是将文件指针和待处理文件相关联



打开文件：

表明函数返回的是文件指针类型；

```
FILE * fopen(char *filename, char *mode);
```

字符指针，实参两个字符串传递

文件的打开和关闭

用一个字符串常量作为实参传递给filename和mode

说明是指针变量

- FILE * fopen(char *filename, char *mode);

  - filemame为需要打开的文件名称，可以包含文件路径。
  - 文件路径
    - 绝对路径 “C:\\temp\\abc.txt”
    - 相对路径 “abc.txt”

文件的打开和关闭

- FILE * fopen(char *filename, char *mode);
  - mode为文件使用方式（打开方式）

明白打开这个文件为了完成什么样的工作

文件的打开和关闭

文件打开方式	含义
r	以输入方式打开1个文本文件
w	以输出方式打开1个文本文件
a	以输出追加方式打开1个文本文件
r+	以读/写方式打开1个文本文件
w+	以读/写方式建立1个新的文本文件
a+	以读/写追加方式打开1个文本文件

- 从内存角度看：
- r是read为了读取，打开一个已经存在的文件，将文件内容读入/输入到内存。
- r+是为了更新文件内容，以读/写两种方式，打开一个已经存在的文件。并不删除已有内容和新建文件

文件的打开和关闭

文件打开方式	含义
r	以输入方式打开1个文本文件
w	以输出方式打开1个文本文件
a	以输出追加方式打开1个文本文件
r+	以读/写方式打开1个文本文件
w+	以读/写方式建立1个新的文本文件
a+	以读/写追加方式打开1个文本文件

- w是write为了写入，清空文件或者新建一个文件，将内存中数据写入/输出到磁盘文件。
- w+是为了更新，可以读一个文件，删除原先文件中内容，然后写入新内容，文件不存在则新建

文件的打开和关闭

文件打开方式	含义
r	以输入方式打开1个文本文件
w	以输出方式打开1个文本文件
a	以输出追加方式打开1个文本文件
r+	以读/写方式打开1个文本文件
w+	以读/写方式建立1个新的文本文件
a+	以读/写追加方式打开1个文本文件

- 以w方式打开文件，文件内容首先会被清空，然后再将新的数据写入；如果保留原来的数据，用a是append，新的数据是增加的内容，以追加方式打开文件。
- a+是可读可写的追加打开方式。

文件打开方式	含义
r	以输入方式打开1个文本文件
w	以输出方式打开1个文本文件
a	以输出追加方式打开1个文本文件
r+	以读/写方式打开1个文本文件
w+	以读/写方式建立1个新的文本文件
a+	以读/写追加方式打开1个文本文件

Mode	Read	Write	Create New File*	Truncate
r	Yes	No	No	No
w	No	Yes	Yes	Yes
a	No	Yes	Yes	No
r+	Yes	Yes	No	No
w+	Yes	Yes	Yes	Yes
a+	Yes	Yes	Yes	No
*Creates a new file if it doesn't exist.				

文件的打开和关闭

文件打开方式	含义
r	以输入方式打开1个文本文件
w	以输出方式打开1个文本文件
a	以输出追加方式打开1个文本文件
r+	以读/写方式打开1个文本文件
w+	以读/写方式建立1个新的文本文件
a+	以读/写追加方式打开1个文本文件
rb	以输入方式打开1个二进制文件
wb	以输出方式打开1个二进制文件
ab	以输出追加方式打开1个二进制文件
rb+	以读/写方式打开1个二进制文件
wb+	以读/写方式建立1个新的二进制文件
ab+	以读/写追加方式打开1个二进制文件

文件的打开和关闭

- FILE * fopen(char *filename, char *mode);
  - 函数的返回值为FILE类型的地址，如果文件打开失败，则返回值为NULL。
- 由于文件不存在、磁盘空间满、磁盘写保护等各种原因，文件打开可能会失败。因此，打开文件时最好进行一定的判别。

```
if((fp = fopen( “D:\\data\\file1.txt” ,  “r” ))==NULL)
{
      printf("can not open file\n");
      exit(1);
}
```

文件的打开和关闭

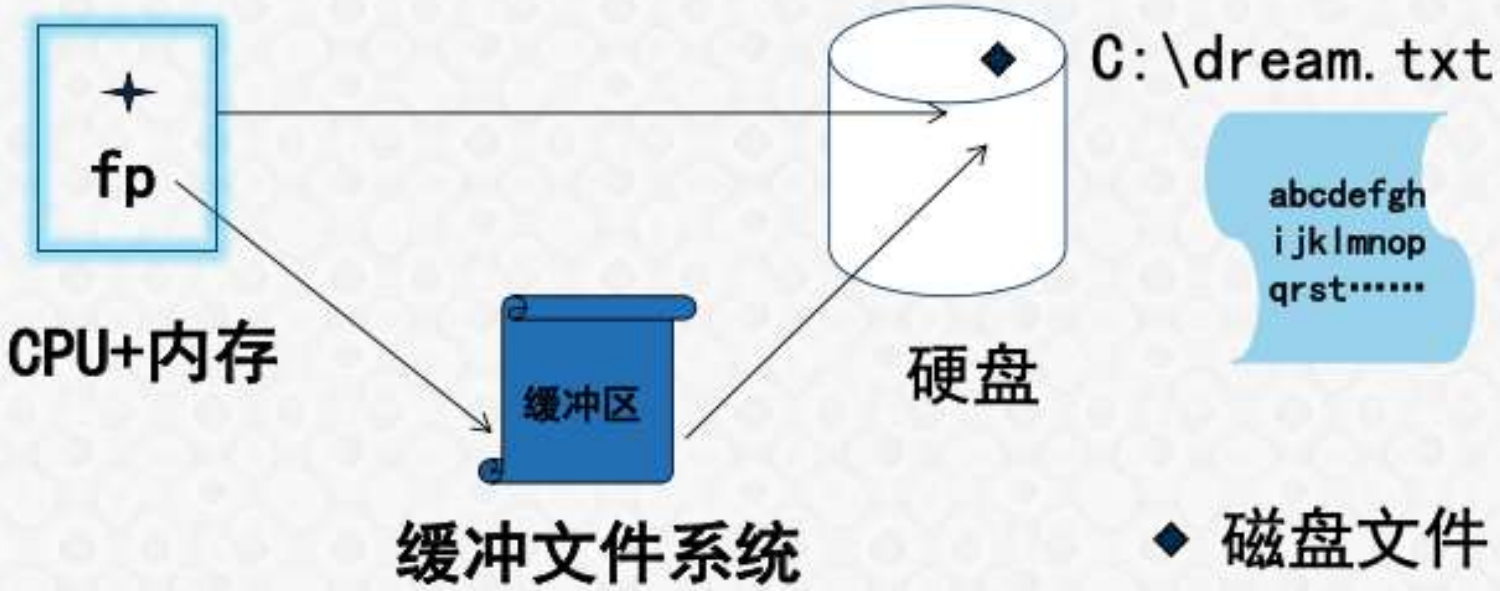
步骤：

FILE *fp;

1. 定义文件指针
2. 打开文件
3. 对文件进行读写操作
4. 关闭文件

文件的打开和关闭

- 养成在程序终止前，关闭所有文件的习惯。否则有可能造成部分数据的丢失。



文件的打开和关闭

- 使用fclose函数

  将文件指针fp与前面相关联的磁盘文件，断开关联。

- fclose函数原型是：

```
int fclose(FILE *fp);
```

其中fp是指向待关闭文件的指针。如果文件关闭成功，函数返回值为0，否则返回一个EOF（EOF是一个定义在stdio.h中的常量，一般为-1）。

- 例如，要关闭上页例子中的**fp**指针指向的文件 **D:\\data\\file1.txt**，可使用：

```
fclose(fp );
```

设置

1、关于文件，下列说法中正确的是_____。

A C语言中，根据数据的存放形式，文件可分为文本文件和二进制文件

B C语言只能读写二进制文件

C C语言中的文件由记录序列组成

D C语言只能读写文本文件

提交

2、如果要对E盘myfile目录下的文本文件abc.txt进行读、写操作，文件打开方式应为_____。

A fopen("e:\\myfile\\abc.txt", "wb");

B fopen("e:\\myfile\\abc.txt", "r+");

C fopen("e:\myfile\abc.txt", "r");

D fopen("e:\\myfile\\abc.txt", "rb");

3、若执行fopen函数时发生错误，则函数的返回值是_____。

A 地址值

B 0

C 1

D EOF

```
FILE *fp=fopen( “xxx.txt", “w");
if(!fp)
if(fp==0)
if(fp==NULL)
```

提交

文件读写

- 文件读写主要有四对函数控制：
  - 字符读写 ： fgetc、fputc
  - 字符串读写： fgets、fputs
  - 格式化读写： fscanf、fpirntf

  } 文本文件

  - 数据块读写： fread、fwrite ⟶ 二进制文件或文本文件

输入	一个字符	字符串	格式化
从控制台	getchar	gets	scanf
从文件	fgetc	fgets	fscanf

输出	一个字符	字符串	格式化
向控制台	putchar	puts	printf
向文件	fputc	fputs	fprintf

设置

4、如果要对E盘myfile目录下的二进制文件abc.dat进行读、写操作，应使用的读写函数为_____。

A fgetc和fputc

B fgets和fputs

C fscanf和fprintf

D fread和fwrite

提交

文件读写

- （1）fputc——将字符写入文件
  - 函数原型：int fputc(int c, FILE *fp);
  - 第一形参c：是要写入文件的字符的ASCII值，它虽被定义为整型，但只使用最低位的一个字节
  - 第二形参fp ： 文件指针
  - 函数功能：将字符c写入fp所指向的文件
  - 函数返回值：如果成功，位置指针自动后移一个字节的位置，并且返回c；否则返回EOF
  - 例11_1： 将个人收货信息从键盘输入，以'#'结尾，内容存至文本文件D:\CV.txt中

例11-1主要代码


```
。。。。。。                                    /*文件包含*/
int main( )
  {
        FILE *fp;        /* 首先定义文件指针 */
        char ch;
        fp = fopen( "D:\\CV.txt", "w" );  /* 以"W"方式打开文本文件 */
        if ( fp == 0 )      /* 文件打开后需判断，如果失败 则结束程序*/
            ......
        printf( "Enter a text ( end with '#' ):\n");
         ch = getchar( );
        while( ch != '#' )              /* 写文件操作 */
        {   fputc(ch, fp);             /* 调用fputc将刚读的字符写到文件*/
            ch = getchar( );
        }
         fclose(fp);                   /* 关闭文件 */
        return 0;     }
```

文件读写

- （2）fgetc——从文件中读出一个字符
  - 函数原型：int fgetc(FILE *fp);
  - 形参fp ： 文件指针
  - 函数功能：从fp所指向的文件中读取位置指针所指向的一个字符
  - 函数返回值：如果成功则返回读取的字符，位置指针自动后移一个字节的位置；否则返回EOF
  - 例11_2：读取收货信息文件D:\CV.txt的内容，并原样输出至屏幕。

例11-2主要代码

```
。。。。。。                                    /*文件包含*/
int main( )
  {
        FILE *fp;        /* 首先定义文件指针 */
        char ch;
        fp = fopen( "D:\\CV.txt", "r" );   /* 以"r"方式打开文本文件 */
        if ( fp == 0 )      /* 文件打开后需判断，如果失败 则结束程序*/
            ......
        while( ( ch = fgetc(fp) ) != EOF )          /* 读文件操作 */
       {        putchar(ch);            /*屏幕输出刚刚读出的字符内容*/
        }
        putchar( '\n' );
        fclose(fp);                  /* 关闭文件 */
       return 0;
  }
```

小结

1. `fopen` - 打开文件：
   - `fopen` 用于打开文件，它返回一个文件指针（`FILE*`），该指针用于后续对文件的读写操作。
   - 原型：`FILE *fopen(const char *filename, const char *mode);`
   - `filename` 是文件的路径和名称，`mode` 是打开文件的模式，例如 `"r"` 表示读取模式，`"w"` 表示写入模式等。
   - 如果打开成功，`fopen` 返回一个指向文件的指针；如果失败，返回 `NULL`。

c Copy code

```c
FILE *fp = fopen("example.txt", "r");
if (fp != NULL) {
    // 文件打开成功，可以进行读取或写入操作
    fclose(fp); // 记得在使用完文件后关闭文件
} else {
    // 文件打开失败
}
```

小结

2. `fgetc` - 从文件中读取一个字符：
   - `fgetc` 用于从文件中读取一个字符。
   - 原型：`int fgetc(FILE *stream);`
   - `stream` 是指向文件流的指针。
   - 如果成功读取字符，返回实际读取的字符；如果到达文件末尾或发生错误，返回 `EOF`。

c Copy code

```c
FILE *filePointer = fopen("example.txt", "r");
if (filePointer != NULL) {
    int ch = fgetc(filePointer);
    while (ch != EOF) {
        // 处理读取到的字符
        ch = fgetc(filePointer);
    }
    // 关闭文件
    fclose(filePointer);
}
```

小结

3. `feof` - 检查文件结束标志：
   - `feof` 用于检查文件流的文件结束标志。
   - 原型：`int feof(FILE *stream);`
   - `stream` 是指向文件流的指针。
   - 如果文件流已经到达文件末尾，`feof` 返回非零值（true），否则返回零值（false）。

c Copy code

```c
FILE *fp = fopen("example.txt", "r");
if (fp != NULL) {
    while (!feof(fp)) {
        // 循环读取文件内容
        int ch = fgetc(fp);
        // 处理文件内容
    }
    fclose(fp);
}
```

文件读写

- 文件读写主要有四对函数控制：
  - 字符读写 ： fgetc、fputc
  - 字符串读写：fgets、fputs
  - 格式化读写：fscanf、fpirntf
  - 数据块读写：fread、fwrite

文件读写

- （3）fputs——将字符串写入文件
  - 函数原型：int fputs(const char *s, FILE *fp);
  - 第一形参s：要写入的文件的字符串
  - 第二形参fp：文件指针
  - 函数功能：将字符串s输出至fp所指向的文件（不含结尾标志'\0'）
  - 函数返回值：如果成功，位置指针自动后移，函数返回一个非负整数；否则返回EOF

文件读写

- （4） fgets——从文件读取字符串
  - 函数原型：

    char *fgets(char *s, int n, FILE *fp);
  - 第一形参s ： 指向待赋值字符串的首地址
  - 第二形参n: 控制读取串的字符个数， 最多n-1个
  - 第三形参fp ： 文件指针
  - 函数功能：从位置指针开始读取一行或n-1个字符，并存入s，存储时自动在字符串结尾加上'\0'
  - 函数返回值：如果函数执行成功，位置指针自动后移，并返回s的值，否则返回NULL

文件读写

- fgets和fputs的应用举例：
  - 例11_3 某购物群购物，要求在文件D:\CV.txt的基础上追加几行，每行是本次购买的每种产品的名称及数量，从键盘输入以“$”结束并存入文件，最后输出统计的文件总行数以及文件的完整内容。
  - 打开方式分析：追加内容所以用a方式打开，又因为既要写又要读文件，所以是a+方式打开
  - 程序控制：首先用一层循环负责从键盘读入字符串并调用fputs写内容入文件，然后调用rewind函数使指针回到文件开头（此步很关键），再用一层循环统计文件行数并读取内容。

例11-3主要代码


```
。。。。。。                                    /*文件包含*/
int main( )
  {
        FILE *fp;        /* 首先定义文件指针 */
        char str[N];       int lines=0;
        fp = fopen( "D:\\CV.txt", "a+" ); /*以"追加"及可读写方式打开文件*/
        if ( fp == 0 )      /* 文件打开后需判断，如果失败 则结束程序*/
            ......
        gets(str) ;                    /*从键盘上读入一个字符串代表一种商品*/
        while( strcmp(str,"$")!= 0 )      /*如果读入的不是"$"串则写入文件*/
        {
            fputs(str,fp) ;             /*将读入的字符串写入文件*/
            fputc('\n',fp);             /*在该串末尾写入一个换行符*/
            gets(str);                 /*从键盘上继续读入一个字符串*/
        }
```

例11-3主要代码


```
rewind(fp) ;                    /*让文件指针重新定位到文件开始位置*/
while (fgets(str,N,fp)!=NULL)  /*从文件读取一行内容放到str串中*/
{
    printf("%s",str);          /*原样输出该字符串*/
    lines++;                   /*行数加1*/
}
fclose(fp);                              /* 关闭文件 */
printf("the file has %d lines.\n",lines); /*输出统计的行数*/
return 0;
}
```

文件读写

- 文件读写主要有四对函数控制：
  - 字符读写 ： fgetc、fputc
  - 字符串读写：fgets、fputs
  - 格式化读写：fscanf、fpirntf
  - 数据块读写：fread、fwrite

文件读写

- （5）fprintf——将内容按格式写入文件
  - 函数原型：int fprintf(FILE *fp, const char* format, 输出参数1, 输出参数2...);
  - 第一形参fp ： 文件指针
  - 第二形参format ： 格式控制字符串
  - 其余参数：输出参数表列为待输出的数据
  - 函数功能：根据指定的格式（format参数）发送数据（输出参数）到fp打开的文件
  - 函数返回值：正常写入情况下返回写入文件的数据个数，出错返回EOF

文件读写

- fprintf与printf的关系：
  - printf是fprintf的特殊形式
  - 语句printf("There are %d cats here.\n",2);与语句fprintf(stdout,"There are %d cats here.\n",2);完全等效
  - stdout就是默认的对应显示器的文件指针,无需定义
  - 使用fprintf的方式与printf几乎是一样的，向显示器按何种格式输出什么内容，到了fprintf函数中控制方式不变，只需要在最前面加一个文件指针参数，这些内容就写到文件指针打开的对应文件中而不是输出到显示器上

文件读写

- （6）fscanf——从文件按格式读取内容给变量
  - 函数原型：int fscanf(FILE *fp,
    const char* format, 地址1, 地址2…);
  - 第一形参fp ： 文件指针
  - 第二形参format ： 格式控制字符串
  - 其余参数：地址表列为输入数据的存放地址
  - 函数功能：根据format参数指定的格式从fp打开的文件读取数据存到地址参数指定的内存中
  - 函数返回值： 返回实际读取的数据个数,出错或者到结尾返回EOF

文件读写

- fscanf与scanf的关系：
  - scanf是fscanf的特殊形式
  - 若有变量定义：int a; 则语句scanf("%d ",&a);与语句fscanf(stdin, "%d ",&a); 完全等效
  - stdin是默认的对应键盘的文件指针,无需定义
  - 使用fscanf的方式与scanf几乎是一样的，原先从键盘输入内容到变量，到了fscanf函数中控制方式不变，只需要在最前面加一个文件指针参数，则这些内容就通过指针打开的文件自动输入到变量

设置

5、有变量定义int x; 则语句scanf（“%d”,&x);
与下列哪条语句等效---------

A scanf(stdin, “%d”,&x);

B fscanf(stdin, “%d”,&x);

C printf(stdout, “%d”,&x);

D fprintf(stdout, “%d”,&x);

提交

文件读写

- 文件读写主要有四对函数控制：
  - 字符读写 ： fgetc、fputc
  - 字符串读写：fgets、fputs
  - 格式化读写：fscanf、fpirntf
  - 数据块读写：fread、fwrite

文件读写

- （7） fwrite——将内容按规定字节写入文件
  - 函数原型： int fwrite(const void *buffer,
    int size, int n, FILE *fp);
  - 第一形参buffer ： 要输出数据在内存中的首地址
  - 第二形参size ： 一个数据块的字节数
  - 第三形参n：数据块的个数
  - 第四形参fp ： 文件指针
  - 函数功能：从内存的buffer地址开始，将连续n*size个字节的内容原样写入到fp打开的文件中
  - 函数返回值：实际写入的数据块个数

文件读写

- （8）fread——从文件读取规定字节内容给变量
  - 函数原型：int fread(void *buffer, int size,
    int n, FILE *fp);
  - 第一形参buffer ： 要输入数据在内存中的首地址
  - 第二形参size ： 一个数据块的字节数
  - 第三形参n：数据块的个数
  - 第四形参fp ： 文件指针
  - 函数功能：从fp打开的文件的当前位置开始，连续读取n*size个字节的内容，存入buffer作为首地址的内存空间里
  - 函数返回值：实际读入的数据块个数

文件读写

- fwrite和fread函数的应用举例：
  - 例11_5 有一批学生的数据，包括学号、姓名、考试成绩等信息，要求使用fwrite函数将其存入文件D:\computer.dat文件中。
  - 需要演示本例中fwrite函数的几种等效调用方法
  - 例11_6 使用fread函数读取文件 D:\computer.dat文件的内容并输出至屏幕。
  - 需要注意如何控制循环一条条读出记录的方法

例11-5主函数代码


```
int main( )
{
    FILE *fp;
    STU2 stu[3] = {{ 1001, "Tom", 77 }, { 1002, "Jack", 93 }, { 1003, "Lisa", 86} };
    fp = fopen( "D:\\computer.dat", "wb" );/* 打开文件, 写方式 */
    if ( fp == 0 )                          /* 文件打开失败 */
        ......
    fwrite( stu, sizeof(STU2), 3, fp);      /*将3条记录一次性写入文件*/
    fclose(fp);                             /*关闭文件*/
    return 0;
}
```

例11-6主函数代码


```
int main( )
{
        FILE *fp;
         STU2 stu[N];
        int i=0,num;
        fp = fopen( "D:\\computer.dat", "rb" );/* 打开文件,读方式 */
        if ( fp == 0 )                          /* 文件打开失败 */
            ......
         fread( &stu[i], sizeof(STU2), 1, fp );     /* 读取一个数据块 */
        while( !feof(fp) )                  /*将所有记录读出放在数组中*/
           fread( &stu[++i], sizeof(STU2), 1, fp );   /* 读取一个数据块 */
        num=i;                          /*用num记下记录条数*/
        for (i=0; i<num; i++)                   /*集中输出所有的元素*/
        printf( "%d %-8s \t%.2f\n", stu [i].ID, stu [i].name, stu[i].score);
        fclose(fp);                     /*关闭文件*/
        return 0;   }
```

6、对“fread(arr, 36, 3, fp)”解释正确的是（ ）

A 从fp中读出整数36，并存放至 arr 中

B 从fp中读出整数36和3，并存放至 arr 中

C 从fp中读出36个字节的内容，并存放至 arr 中

D 从fp中读出3个36 个字节的内容，并存放至 arr 中

提交

7、设已有一个结构体类型ST，并定义一个结构体数组如下：struct ST stu[30];

如果要将这个数组的内容全部写入文件fp，以下方法中不正确的是 _____。

A fwrite(stu, sizeof(struct ST), 30, fp);

B fwrite(stu, 30*sizeof(struct ST), 1, fp);

尝试一次性写入整个数组，而不是每个元素分别写入
应该使用 sizeof(struct ST) 代替 30*sizeof(struct ST)

C fwrite(stu, sizeof(stu), 30, fp);

D
```
for ( i=0 ; i<30 ; i++ )
    fwrite( stu +i, sizeof(struct ST), 1, fp );
```

提交

本章小结

- 文件打开模式

r	以输入方式打开1个文本文件
w	以输出方式打开1个文本文件
a	以输出追加方式打开1个文本文件
r+	以读/写方式打开1个文本文件
w+	以读/写方式建立1个新的文本文件
a+	以读/写追加方式打开1个文本文件

- 文件读写的四对函数及其使用例示
  - 字符读写 ：fgetc、fputc
  - 字符串读写：fgets、fputs
  - 格式化读写：fscanf、fpirntf
  - 数据块读写：fread、fwrite

输入理想的程序

输出快乐的人生