C语言标准库函数每操作一个文件，都为了这个文件建立了一个FILE型变量

FILE * file

这个fp指针代表一个文件，对文件的任何操作都离不开这个文件类型的指针。

打开文件

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

filename是一个字符串，表示要打开的文件名，文件名前面可以加上该文件所在的磁盘路径

mode也是一个字符串，表示打开文件的方式。

返回值 打开成功返回文件指针

失败返回NULL

打开文件方式 含义

r 读文件不能写

w 写文件不能读

a 打开已存在的文件在文件尾部追加数据

r+，w+，a+ 可读写

t 打开文本文件

b 打开二进制文件

对于文件操作的库函数，函数原型都在头文件stdio.h中。

```
FILE *fp;
fp = fopen("wang.txt","r");
if (fp == NULL) {
    printf("the file:wang.txt not found!");
    exit(-1);
}
```

关闭文件

```
int *fclose(FILE * filepointer)
```

filepointer:文件指针，通常这个参数就是fopen函数的返回值。

返回值：如果正常关闭了文件，则函数返回值为0;否则返回值为非0.

```
FILE *fp;
fp = fopen("wang.txt","r+");
if(fp == NULL){
    printf("the file:wang.txt not found");
    exit(-1);
}
...                 //读取和加工数据
fclose(fp);         //关闭该文件
```

文件的读写

字符读写函数：fgetc和fputc

字符串读写函数：fgets和fputs

数据块读写函数：fread和fwrite

格式化读写函数: fscanf和fprintf

字符读写函数fgetc和fputc是以字符为单位进行文件读写的函数。每次可从文件读出或向文件写入一个字符。

fgetc函数

```
int fgetc(FILE * filepointer);
```

功能：从文件指针filepointer所指向的文件中，读取一个字节数据，同时将读写位置指针向前移动一个字节。

返回值：如果读取正常，返回读到字节值，结尾或者出错位置返回EOF(即-1)

fputc函数

```
int fputc(int c, FILE *filepointer)
```

功能：将形参c表示的字符数据输出到文件指向的文件中去，同时将读写位置指针向前移动1个字节

返回值：成功返回及时输出的字符数据c；否则返回EOF

将键盘输入的一个字符串(以'@'作为结束符)，以ASCII码形式存储到一个磁盘文件中，然后从该磁盘文件中读取其字符串并显示出来。

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

void main(int argc,char *argv[])
{
    FILE *fp1,*fp2;
    char ch;
```

```
    if(argc != 2){          //参数个数不对
        printf("the number of arguments not correct\n\n");
        printf("Usage:可执行文件名 filename\n");
        exit(0);
    }

    if ((fp1 = fopen(argv[1],"wt")) == NULL)     //打开文件失败
    {
        printf("can not open this file\n");
        exit(0);
    }

    //输入字符，并存储到指定文件中
    for(;(ch = getchar()) != '@';){
        fputc(ch,fp1);  //输入字符并存储到文件中
    }
    fclose(fp1);        //关闭文件

    //顺序输出文件的内容
    fp2 = fopen(argv[1],"rt");
    for(;(ch=fgetc(fp2)) != EOF;){
        putchar(ch);    //顺序读入并显示
    }
    fclose(fp2);        //关闭打开的文件
}
```

example wang.txt

How are you?@

How are you

利用字符读写函数实现文件拷贝

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

void main(int argc, char *argv[]){
    FILE *input,*output;         //input：源文件指针，output：目标文件指针

    if(argc ! = 3)      //参数个数不对
    {
        printf("the number of arguments not correct\n");
        printf("\n Usage:可执行文件名 source-file dest-file");
        exit(0);
    }

    if((input=fopen(argv[1],"r"))==NULL)  //打开源文件失败
    {
```

```
        printf("can not open source file\n");
        exit(0);
    }

    if((output=fopen(argv[2],"w"))==NULL)  //创建目标文件失败
    {
        printf("can not create destination file\n");
        exit(0);
    }

    //复制源文件到目标文件中
    for(;(!feof(input));){              //feof判断文件是否结束的库函数 结束返回1否则返回0
        fputc(fgetc(input),output);
    }
    fclose(input);              //关闭源文件
    fclose(output);             //关闭目标文件
}
```

字符串读写函数fgets和fputs

字符串读写函数fgets和fputs是以字符串的形式对文件进行读写的函数。每次可以从文件读出（指定长度）或向文件中写入一个字符串。

```
char *fgets(char *s,int n,FILE * filepoionter)
```

功能：从文件指针filepointer所指向文件中读取长度最大为n-1个字符的字符串，并在字符串的末尾加上串结束标志'\0'，然后将字符串存放到s中。同时将读写位置指针向前移动实际读取的字符长度（<=n-1）个字节。当从文件中读取第n-1个字符后或读取数据过程中遇到换行符'\n'后，函数返回。

返回值：成功返回读取的字符串指针；如果读到文件尾或出错，则返回null

```
int fputs(chars *s,FILE *filepointer);
```

功能：将s所表示的字符串写到文件指针filepointer所指向的文件中去，同时将读写位置指针向前移动字符串长度个字节。注意fputs函数不会将字符串结尾符'\0'写入文件，也不会自动向文件写入换行符。

返回值：如果操作成功,则函数返回值就是最后写入文件的字符值；否则返回EOF。

向文件wang.txt中写入两行文本，然后分三次读出其内容

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

void main(){
    FILE *fp1,*fp2;
    char str[] = "123456789";
```

```
    fp1 = fopen("wang.txt","w");       //创建文本文件wang.txt
    if(fp1 == NULL){                   //创建文件失败
        printf("can not open file:wang.txt\n");
        exit(0);
    }

    fputs(str,fp1);//将字符串"123456789"写入文件
    fputs("\nabcd",fp1);      //写入第一行文本的换行符和下一行文本
    fclose(fp1);    //关闭文件

    fp2 = fopen("wang.txt","rt");     //以只读方式打开wang.txt文件
    fgets(str,8,fp2);       //读取字符串，最大长度是7，将是"1234567"
    printf("%s\n",str);
    fgets(str,8,fp2);       //读取字符串，最大长度是7，实际上将是"89\n"
    printf("%s\n",str);
    fgets(str,8,fp2);       //读取字符串，最大长度是7，实际上将是"abcd"
    printf("%s\n",str);

    fclose(fp2);            //关闭打开文件
}
```

运行结果：

1234567

89

abcd

数据块读写函数fread和fwrite

C语言提供用于整块数据读写函数fread和fwrite

```
unsigned fread(void *ptr,unsigned size,unsigned n,FILE *filepointer);
```

功能：从filepointer所指向的文件中读取n个数据项，每个数据项的大小是size个字节，这些数据将被存放到ptr所指的内存中，同时，将读写位置指针向前移动n*size个字节。

返回值：如果成功返回读取的数据项个数，如果出错或遇到文件尾部返回0

将一个整型数组存放到文件中，然后从文件中读取数据到数组中并显示出来

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

void main() {
```

```
    FILE *p;
    short i,a[10] = {0,1,2,3,4,5,6,7,8,9};

    fp = fopen("wang.dat","wb");  //创建二进制文件wang.dat
    if(fp == NULL){
        printf("can not create file: wang.txt\n");
        exit(0);
    }
    fwrite(a,sizeof(short),10,fp);        //将数组a的10个整型数写入到文件中
    fclose(fp);                           //关闭文件

    fp = fopen("wang.dat","rb");          //打开二进制文件wang.dat
    if(fp == NULL){                       //打开失败
        printf("can not open file:wang.dat\n");
        exit(0);
    }
    memset(a,0,10*sizeof(short));         //将数组a的10个元素清0
    fread(a,sizeof(short),10,fp);         //从文件中读取10个整型数据到数组a
    fclose(fp);

    for(i = 0; i<10 ; i++){           //显示数组a的元素
        printf("%d",a[i]);
    }
}
```

运行结果:0 1 2 3 4 5 6 7 8 9

二进制文件wang.dat中存放的信息如下(十六进制形式)

(00 01) 0 (01 00)2 (02 00) 2 (03 00)3 (04 00)4 ... (09 00)9

格式化读写函数fscanf和fprintf

C语言提供了对文件进行格式化读写函数fscanf和fprintf。fscanf和fprintf函数的操作对象是指定的文件。

```
//从filepointer所指向的文件中读取数据。
int fscanf(FILE *filepointer,const char*format[,address,...]);
//将表达式输出到filepointer所指向的文件中。
int printf(FILE *filepointer,const char*format[,address,...]);
```

例子:

```
fprint(fp,"%d,%6.2f",i,t);          //将i和t按%d,%6.2f格式输出到fp文件
fscanf(fp,"%d,%f",&i,&t);           //若文件中有3,4.5,则将3送入i,4.5送入t
```

前面的对文件读写方式都是顺序读写，即读写文件只能从头开始，顺序读写文件中各个数据。实现文件的随机读写是要按要求移动位置指针，这称为文件定位。移动文件读写位置指针主要有rewind,fseek等。

```
void rewind(FILE *filepointer);
```

功能：将filepointer所指向的文件的读写位置指针重新置回到文件首部。

返回值：无

```
void fseek(FILE *filepointer,long offset,int whence)
```

功能：将filepointer所指向的文件的读写位置指针移动到特定的位置。这个特定的位置由whence和offset决定，即将位置指针移动到距离whence的offset字节处。如果offset是正值，表明新的位置在whence的后面，如果offset是负值，表明新的位置在whence的前面。

返回值：如果操作成功，返回值为0，否则返回一个非0值。

whence的常量值 数值 含义

SEEK_SET 0 文件的开始处

SEEK_CUR 1 文件位置指针的当前位置

SEEK_END 2 文件的末尾

```
long ftell(FILE *filepointer)
```

功能：返回filepointer所指向的文件的当前读写位置指针的值（用相对文件开头的位移量表示）

返回值：如果操作成功，返回当前位置指针的值，如果出错，否则返回-1L。

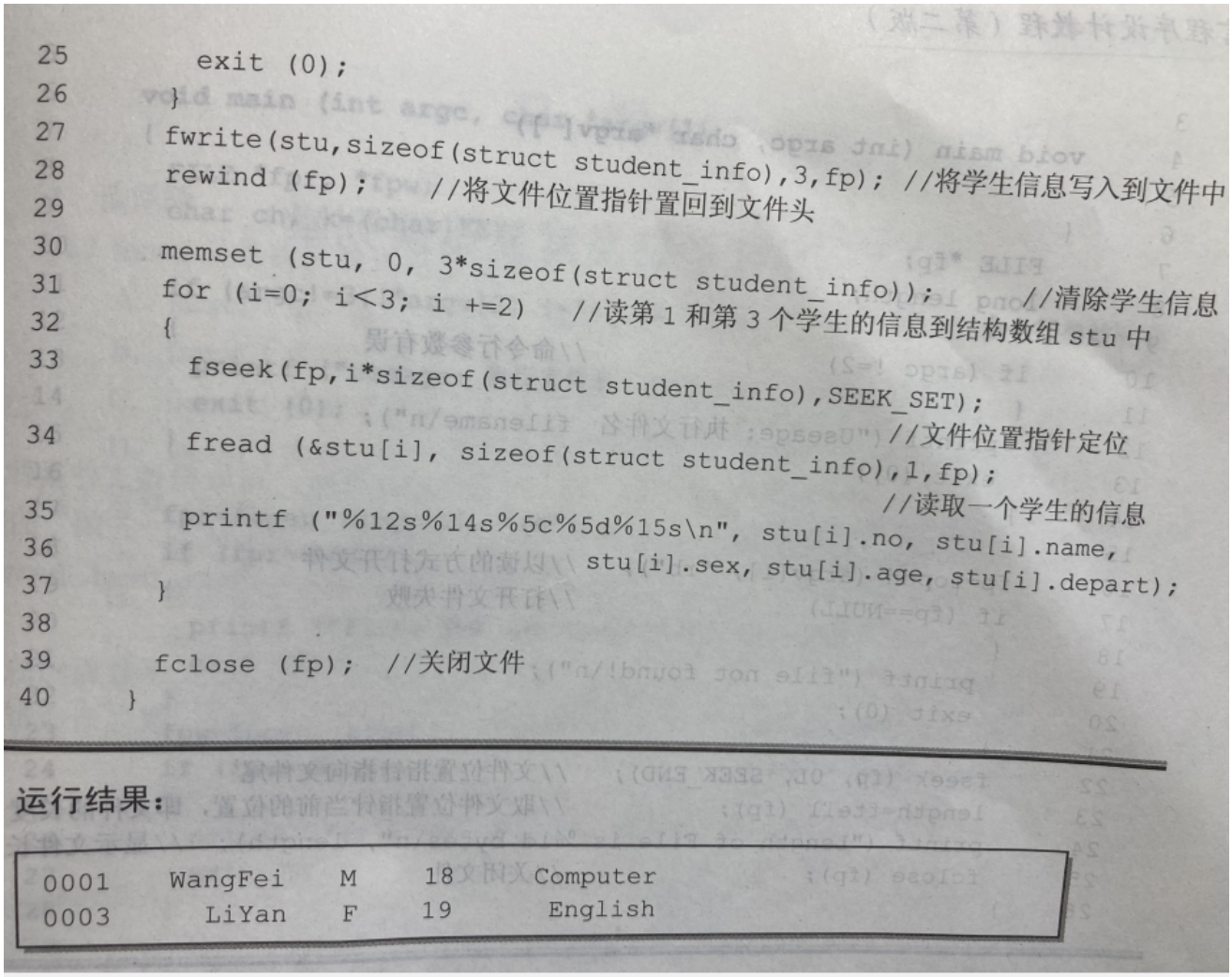
【例 12-8】 磁盘文件上有 3 个学生数据，要求读入第 1，3 学生数据并显示。

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

struct student_info
{
  char no[9];
  char name[10];
  char sex;
  int  age;
  char depart[15];
}stu[3]={ {"0001", "WangFei", 'M', 18, "Computer"},
          {"0002", "ZhangMin", 'M', 19, "Math"},
          {"0003", "LiYan", 'F', 19, "English"} };

void main ( )
{
  int i;
  FILE *fp;

  fp=fopen ("student.dat", "wb+");     //以读写方式打开二进制文件
  if (fp==NULL)                        //打开失败
  {
    printf ("can't create file: student.dat\n");
```



```
25        exit (0);
26      }
27      fwrite(stu,sizeof(struct student_info),3,fp); //将学生信息写入到文件中
28      rewind (fp);      //将文件位置指针置回到文件头
29
30      memset (stu, 0, 3*sizeof(struct student_info));      //清除学生信息
31      for (i=0; i<3; i +=2)   //读第 1 和第 3 个学生的信息到结构数组 stu 中
32      {
33        fseek(fp,i*sizeof(struct student_info),SEEK_SET);
                                                    //文件位置指针定位
34        fread (&stu[i], sizeof(struct student_info),1,fp);
                                                    //读取一个学生的信息
35        printf ("%12s%14s%5c%5d%15s\n", stu[i].no, stu[i].name,
36                stu[i].sex, stu[i].age, stu[i].depart);
37      }
38
39      fclose (fp);  //关闭文件
40    }
```

运行结果：

```
0001    WangFei    M    18    Computer
0003      LiYan    F    19     English
```

判断文件是否结束的库函数feof

```
int feof(FILE * filepointer);
```

功能：在执行读文件操作时，如果遇到文件尾，则函数返回逻辑真（1），否则返回逻辑假（0）

带参数的main函数形式为：

```
void main(int args,char *argv[]){

   ...

}
```

其中:argc用于存放命令行中参数的个数（字符串的个数）；argv是一字符型指针数组，数组中的元素（即字符串指针）指向命令行中各字符串的首地址；

命令名(argv[0]) 参数1(argv[1]) ... 参数n(argv[n])

==========参数个数n+1==========

```
void main(int argc,char *argv[]){
    while(argc-->0)
        printf("%s\n",*argv++);
}
```

上诉假设程序名为test.cpp，对其执行，在 DOS下运行，输入命令行假设为C:\test hello world

运行结果为：

test

hello

world

统计候选人选票。

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

struct person{
    char name[20];    //候选人姓名
    int count;        //得票数
} leader[3]={"LI", 0, "Zhang", 0, "Wang", 0};

void main(){
    int i,j;
    char leader_name[20];

    while(1)          //统计候选人得票数
    {
        scanf("%s",leader_name);     //输入候选人姓名
        if(strcmp(leader_name,"0") == 0){    //输入为"0"结束
            break;
        }
        for(j = 0;j<3;j++){                  //比较是否为合法人
            if(strcmp(leader_name,leader[j].name) == 0){//合法
                leader[j].count++;//得票数加一
            }
        }
    }

    for(i = 0; i<3;i++){
        printf("%5s :%d\n",leader[i].name,leader[i].count);
    }
}
```

一维数组在程序中赋值（memset和memcpy包含在string.h头文件中）

memset函数可以实现对某内存块的各字节单元整体赋同样的值。

```
void *memset(void *s,char ch,unsigned n)
```

其功能就是将s为首地址的一片连续的n个字节内存单元都赋值ch。注意：是对内存的每个字节单元都赋值为ch。

```
char str[10];
memset(str,'a',10);
//将数组str的每个数据单元赋值为'a'
```

memcpy函数可以实现数组间的赋值。

```
void *memcpy(void *d,void *s,unsigned n)
```

其功能就是将以s为首地址的一片连续的n字节内存单元的值拷贝到以d为首地址的一片连续的内存单元中。

字符及字符串操作的常用函数

1.字符串输入

```
gets(字符数组变量名)          //头stdio.h
```

功能：接受键盘的输入，将输入的字符串放在字符数组中，直到遇到回车符时返回。回车换行符'\n'不会作为有效字符存储到字符数组中，而是转换为字符结束标志'\0'来存储。

gets scanf

输入的字符串中可含空格 不可含空格，遇到空格或回车结束

只能输入一个字符串 可连续输入多个字符串

2.字符串输出

```
puts(字符串的地址);    //应包含的.h文件为stdio.h
```

功能：将字符串中的所有字符输入到终端上，输出时将字符串结束标志'\0'转换成换行符。使用puts无法进行格式控制

```
char str[] = "I love china!"
puts(str);
puts("I love wuhan!");
```

输出结果：

I love china!

I love wuhan!

3.求字符串长度

```
strlen(字符串的地址);                          //头文件string.h
```

功能：返回字符串中包含的字符个数（不包含'\0'）,即字符串的长度。注意：字符串的长度是从给定字符串的起始地址开始到第一个'\0'为止。

```
char str[] = "0123456789";
printf("%d",strlen(str));                    //输出结果为10
printf("%d",strlen(&str[5]));                //输出结果为5
```

4.字符串的复制

字符串的复制不能使用赋值运算符"="，而必须使用strcpy,strncpy或memcpy函数。

```
strcpy(字符数组1，字符串2);                     //应包含的.h文件为string.h
```

功能：将字符串2复制到字符数组1中去（包括字符串结尾符'\0'）。

```
char str1[20],str2[20]
scanf("%s",str2);
strcpy(str1,str2);
```

```
strncpy(字符数组1，字符串2，长度n);        //应包含的.h文件为string.h
```

功能：将字符串2的前n个字符复制到字符数组1中去，并在末尾加'\0'。因此strncpy函数可实现字符串的部分复制。

```
char str[20];
strncpy(str,"0123456789",5)      //将"0123456789"的前5个字符复制到str中，并加'\0'
printf("%s",str);                //将输出01234
```

5.字符串的比较

两字符串的比较不能用">","<"或"=="来进行，必须使用strcmp,stricmp,strncmp,strnicmp等库函数来完成。

```
strcmp(字符串1，字符串2);              //应包含的.h文件为string.h（区分大小写）
```

功能：比较两个字符串的大小。如果字符串1大于字符串2，则返回一个正整数；如果字符串1小于字符串2，则返回一个负整数；如果字符串相等，则返回0。

```
strcmp("abcd","abCD") //返回一个正整数
strcmp("1234","12345") //返回一个负整数
strcmp("hello","hello") //返回0
```

```
stricmp(字符串1, 字符串2);          //应包含的.h文件为string.h （不区分大小写）
```

功能：也是比较两个字符串的大小，只不过stricmp在比较两个字符串时不区分大小写，而strcmp区分大小写。

```
int i;
i = strcmp("abcd","ABCD"); //i的值大于0
i = strcmp("abcd","ABCD"); //i的值等于0
```

```
strncmp(字符串1, 字符串2, 长度n);      //应包含的.h文件为string.h
```

功能：将字符串1前n个字符的子串与字符串2前n个字符的子串进行比较，返回值规则同strcmp。

```
int i;
i = strncmp("abcd","abcDEF",3);          //i的值等于0
i = strncmp("abcd","abcDEF",5);          //i的值大于0
```

---

strnicmp函数的调用格式与strncmp相同，其功能与strncmp相比只是strncmp区分大小写，而strnicmp不区分。

6.字符串连接

如果想要将两个字符串连接起来构成一个新的字符串，则可以调用strcat函数。其调用格式为:

```
strcat(字符数组1, 字符串2);     //应包含的.h文件为string.h
```

功能：将字符串2连接到字符数组1的后面（包括结尾符'\0'）。其中串2没变，而字符数组1中的字符将增加了。

```
char str1[20] = "12345",str2[] = "6789";
strcat(str1,str2);
printf("%S",str1); //将输出123456789
```

```
void main(int argc, char *argv[])
{
    while(argc-->0)
        printf("%s\n",argv++);
}
```

c:>test hello world其内存映射方式如图所示

其内存映射方式如图 9-33 所示。

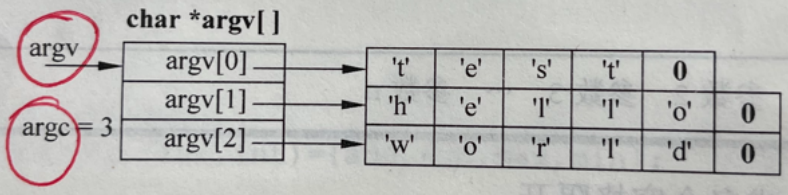


图 9-33　命令行参数传递实例