



第十三章 文件（2）



- 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





13.4 位置指针与文件定位

文件中有一个“**文件位置指针**”，指向下一次读写操作所在字节的整数值。每次读写1个（或1组）数据后，系统自动将文件位置指针移动到下一个读写位置上。

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

↑ ↑

打开文件时位置指针的起始位置与打开文件的方式有关

—以“**r**”、“**w**”方式打开文件，位置指针指向文件头

—以“**a**”方式打开文件，位置指针指向文件尾
所以文件位置指针并不是真正意义上的指针！

13.4 位置指针与文件定位



文件位置指针相关的3个函数：

`rewind()`：文件位置指针指向文件头。

`fseek()`：修改文件位置指针使其指向任一字节处。

`ftell()`：返回文件当前位置的函数

13.4 位置指针与文件定位




一、位置指针复位函数rewind()

1. 用法: **void rewind(FILE * stream);**

2. 功能: 把“文件位置指针”重新定位到文件的起始位置(即0字节处).

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



13.4 位置指针与文件定位



二、随机读写与fseek()函数

文件既可以顺序读写，也可以随机读写，关键在于控制文件的位置指针。

所谓顺序读写是指，读写完当前数据后，系统自动将文件的位置指针移动到下一个读写位置上。

所谓随机读写是指，读写完当前数据后，可通过调用**fseek()**函数，将位置指针移动到文件中任何一个地方。



13.4 位置指针与文件定位

1. 原型: `int fseek(FILE *stream, long offset, int whence);`
2. 用法: `int fseek(文件指针, 位移量, 参照点);`
3. 功能: 将指定文件的位置指针, 从参照点开始, 移动指定的字节数 (位移量)。

(1) 参照点 (3种取值):

SEEK_SET —值为0, 表示文件头

SEEK_CUR —值为1, 表示当前位置

SEEK_END —值为2, 表示文件尾

(2) 位移量: 以参照点为起点, 向文件尾方向 (位移量 >0) 或文件头方向 (位移量 <0) 移动的字节数。在**ANSI C**标准中, 要求位移量为**long int**型数据。如:

`fseek(fPtr, 10, SEEK_SET)`

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



13.4 位置指针与文件定位



三、 返回文件当前位置的函数ftell()

由于文件的位置指针可以任意移动，也经常移动，往往容易迷失当前位置，**ftell()**就可以解决这个问题。

1. 用法: **long ftell(FILE *stream);**

2. 功能: 返回文件位置指针的当前位置（用相对于文件头的位移量表示）。

如果返回值为-1L，则表明调用出错。例如:

```
offset=ftell(fp);
```

```
if(offset== -1L)
```

```
    printf( "ftell() error\n" );
```




函数使用示例

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

```
int main()
```

```
{
```

```
    FILE *fPtr; /*声明指向文件的指针*/
```

```
    if ( (fPtr=fopen("Exam1_1.in","a"))==NULL ) /*打开文件*/
```

```
        printf("File could not be opened\n");
```

```
    else
```

```
    {
```

```
        printf("文件位置指针是: %ld\n", ftell(fPtr));
```

```
        rewind(fPtr);/*把文件位置指针重新定位到文件的起始位置*/
```

```
        printf("文件位置指针是: %ld\n", ftell(fPtr));
```

```
        fseek(fPtr,10,SEEK_CUR);/*文件位置指针重定位*/
```

```
        printf("文件位置指针是: %ld\n", ftell(fPtr));
```

```
        fclose(fPtr);/*关闭文件*/
```

```
    }
```

```
    return 0;
```

```
}
```

文件位置指针是: 115
文件位置指针是: 0
文件位置指针是: 10

- 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





13.5 文件的读写操作

文件打开之后，就可以对它进行读与写的操作了。

12.3.1 读 / 写文件中的一个字符（文本文件）

12.3.2 读 / 写一个字符串（文本文件）

12.3.3 对文件进行格式化读 / 写（文本文件）

12.3.4 读 / 写一个数据块（二进制文件）



13.5 文件的读写操作

一、读 / 写文件中的一个字符

1、库函数fputc(): 将一个字符写入指定文件

1) 用法: `int fputc(int c, FILE *stream);`

2) 功能: 将c所指定字符写入stream所指向的输出流中。与流相关的文件位置指针将向前移动1个字节（即指向下一个写入位置）。

如果输出成功，则函数返回值就是输出的字符数据；否则，返回一个符号常量EOF（其值在头文件stdio.h中，被定义为-1）。



13.5 文件的读写操作

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



`fputc('A',fPtr);` /*在0字节处写入'A',文件位置指针后移1个字节*/



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



`fputc(c,stdout)`等价于`putchar(c)` ,将变量`c`输出到标准输出设备上（一般为屏幕）。



13.5 文件的读写操作

2、库函数fgetc()：从指定文件读取一个字符

1) 用法：**int fgetc(FILE *stream);**

2) 功能：从**stream**所指向的输入流中获取下一个**unsigned char**类型的字符(如果有的话)并转换成**int**类型，同时将文件位置指针向前移动1个字节。函数返回值为读取的字符，如果到文件尾，则设置该流的文件结束指示符，并返回**EOF**；如果出现读错误，则设置该流的错误指示符，并返回**EOF**。

例如，**ch=fgetc(fPtr)**，从文件**fPtr**中读一个字符到**ch**中，同时将**fPtr**的文件位置指针向前移动到下一个字符。



13.5 文件的读写操作

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



`c=fgetc(fPtr);` /*赋值后c的值为'A'，文件位置指针后移一个字节*/

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



`ch=fgetc(stdin)`等价于`ch=getchar()`,从标准输入流（通常是键盘）中读取一个字符



13.5 文件的读写操作

3、读文件时如何判断文件是否读到文件尾

在对ASCII码文件执行读入操作时，如果遇到文件尾，则读操作函数返回一个文件结束标志EOF（其值在头文件stdio.h中被定义为-1）。

在对二进制文件执行读入操作时，必须使用库函数feof()来判断是否遇到文件尾。

4、库函数feof()：

1) 用法： **int feof(FILE *stream);**

2) 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，则返回逻辑假（0）。feof()函数同时适用于ASCII码文件和二进制文件。

例如，!feof(fPtr))表示源文件（用于输入）未结束，循环继续。

[例1]设计函数，将键盘上输入的字符串（以ctrl+Z作为结束字符）逐个字符存储到指定文本文件中



```
void writeFileFromKeyboard(char *filename)
```

```
{
```

```
    char ch;
```

```
    FILE * fPtr=NULL;
```

```
    if((fPtr=fopen(filename,"w"))!=NULL){; //打开文件
```

```
        while((ch=getchar())!=EOF) //从键盘读取字符
```

```
            fputc(ch, fPtr);
```

```
        fclose(fPtr); //关闭文件。执行本语句后，数据将
```

//如果缓冲区未满足且不关闭文件，则数据不会写入文件中。

```
    }
```

```
}
```

This is an example

To read and write

^Z

```
ch=getchar();
```

```
while(ch!=EOF) {
```

```
    fputc(ch, fPtr);
```

```
    ch=getchar();
```

```
}
```

```
ch=getchar();
```

```
while(!feof(stdin)) {
```

```
    fputc(ch, fPtr);
```

```
    ch=getchar();
```

```
}
```

[例2] 设计函数：将一个文本文件逐个字符复制
制到另一文本文件中。



Beijing University of Posts and Telecommunications

```
void copyFile(char * sourceFileName,char * destFileName)
```

```
{
```

函数调用：

```
copyFile( "G:\\temp\\source.c" , " G:\\temp\\dest.c" );
```

```
printf("can't open the source file\n");
```

```
else if( (destfPtr=fopen(destFileName,"w"))==NULL)
```

```
printf("can't open the dest file\n");
```

```
else{
```

```
ch=fgetc(sourcefPtr); //从源文件读取一个字符
```

```
while(!feof(sourcefPtr)){//逐字符复制
```

```
fputc(ch,destfPtr); //将字符写入目标文件
```

```
ch=fgetc(sourcefPtr);
```

```
}
```

```
fclose(sourcefPtr);
```

```
fclose(destfPtr);
```

```
}
```

```
}
```

从键盘缓冲区
取数据



将键盘输入数据写入磁盘文件过程:

```
while((ch=getchar()) != EOF) /*输入字符，并存储到指定文件中*/
```

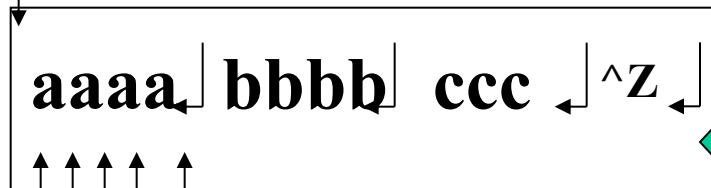
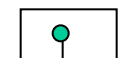
```
fputc(ch,fPtr);
```

```
/*输入字符并存储到文件中*/
```

stdin

缓冲区1

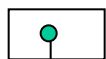
先写入文件
缓冲区



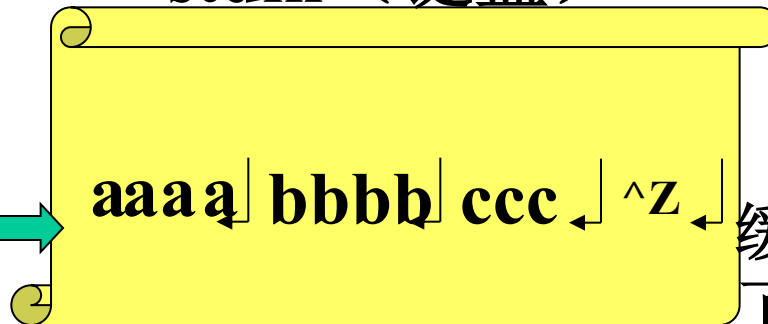
文件位置指针

fPtr

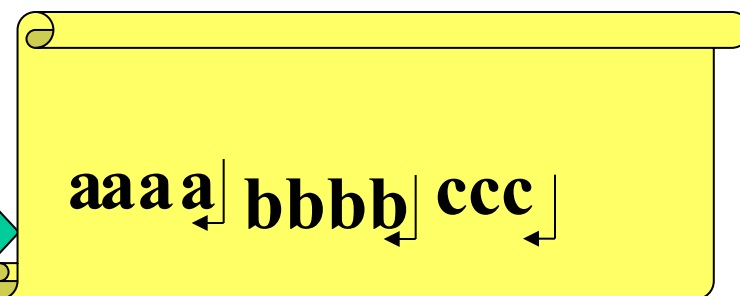
缓冲区2



stdin (键盘)



client.dat



缓冲区满了或者关闭文件时，缓冲区中的数据将批量写入磁盘文件。

总结：文本文件复制的通用处理



```
void copyFile(char * sourceFileName,char * destFileName)
```

```
{  
    //待增变量定义：存放从源文件读取的数据。  
    FILE * sourcefPtr,*destfPtr;  
    if((sourcefPtr=fopen(sourceFileName,"r"))==NULL)  
        printf("can't open the source file\n");  
    else if((destfPtr=fopen(destFileName,"w"))==NULL)  
        printf("can't open the dest file\n");  
    else{  
        //待增代码：从sourcefPtr指向的源文件读取一个（组）数据；  
        while(!feof(sourcefPtr)){  
            //待增代码：将读取的数据写入destfPtr指向的文件  
            //待增代码：从sourcefPtr指向的源文件读取一个（组）数据；  
        }  
        fclose(sourcefPtr);  
        fclose(destfPtr);  
    }  
}
```

数据的读取：逐字符、
逐行、逐记录



```
#include <stdio.h>
#include <stdlib.h> // for exit()
#include <string.h> // for strcpy(), strcat()
#define LEN 40
int main(int argc, char *argv[])
{
    FILE *in, *out;
    int ch;
    char name[LEN];
    int count = 0;
    // 检查命令行参数
    if (argc < 2)
    {
        fprintf(stderr, "Usage: %s filename\n", argv[0]);
        exit(1);
    }
```

命令行参数：第一个参数argv[0]是文件名（包括路径）。多个参数之间用空格分开。

要运行本程序，可在命令行下，输入下述命令：

reducto eddy.txt



// set up input

```
if ((in = fopen(argv[1], "r")) == NULL)
{
    fprintf(stderr, "I couldn't open the file \"%s\"\\n",
        argv[1]);
    exit(2);
}
```

// set up output

```
strcpy(name,argv[1]); // copy filename
strcat(name, ".red"); // append .red
if ((out = fopen(name, "w")) == NULL)
{
    // open file for writing
    fprintf(stderr, "Can't create output file.\\n");
    exit(3);
}
```



getc and fgetc are equivalent, except that getc may be implemented as a macro in some libraries

```
// copy data
while ((ch = getc(in)) != EOF)
    if (count++ % 3 == 0)
        putc(ch, out); // print every 3rd char

// clean up
if (fclose(in) != 0 || fclose(out) != 0)
    fprintf(stderr, "Error in closing files\n");
return 0;

}
```

读 / 写一个字符串——`fgets()`和`fputs()`



1、库函数`fputs()`——向文本文件输出一个字符串

1) 用法: `int fputs(const char *s, FILE * stream);`

2) 功能: 将s所指向的字符串写入stream所指向的流中(‘\0’不被写入)。同时将文件位置指针向前移动字符串长度个字节,指向下一写入位置。如果发生写错误,则函数返回EOF; 否则, 返回一个非负值。

Hello
↑
●

HelloWorld
↑
●

`fputs(s, stdout)`等价于`puts(s)`, 将s所指向的字符串写入到标准输出流。

读 / 写一个字符串——fgets()和fputs()



2、库函数fgets()——从文本文件读一个字符串

1) 用法:

char * fgets(char * s, int n, FILE * stream);

2) 功能: 从**stream**指向的流中读取**最多****n-1**个字符并放到**s**所指向的数组中, 遇到下面情况不再往后读: **1) 读到**新行符 ‘\n’ 或者文件结束符(新行符会被读入到s)**;2) 或虽未遇新行符和文件结束符、但已读入n-1个字符。**

最后一个字符读入数组后接着写入结束标志 ‘\0’, 并将文件位置指针向前移动**n-1** (字符串长度) 个字节。如果遇到文件结束符并且没有字符读入数组, 则数组内容不变。

fgets(s, sizeof(s), stdin)等价于**gets(s)**吗?

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

```
int main(void)
```

```
{
```

```
    char s1[5], s2[100];
```

```
    FILE * fPtr;
```

```
    if((fPtr=fopen("data.txt","r"))==NULL,
```

```
        printf("open file error\n");
```

```
    else{
```

```
        fgets(s1,5,fPtr);
```

```
        printf("%s\n",s1);
```

```
        fgets(s2,100,fPtr);
```

```
        printf("%s",s2);
```

```
        rewind(fPtr);
```

```
        fgets(s2,100,fPtr);
```

```
        printf("%s",s2);
```

```
        fclose(fPtr);
```

```
    }
```

```
    return 0;
```



Where there is life,
There is hope.

data.txt

Where
there is life,
Where there is life,
请按任意键继续...

[例3]设计函数，将键盘上输入的若干字符串（以CTRL+Z作为结束字符），存储到指定文本文件中



执行fgets语句时，程序请求键盘输入。如果用户输入CTRL+Z，则文件结束指示符被设置,此时feof函数返回1。

```
FILE * fPtr=NULL;
```

```
if( (fPtr=fopen(filename,"w"))!=NULL){//打开文件
```

```
    fgets(s,sizeof(s),stdin); // 从键盘读取一行字符
```

```
    while (!feof(stdin)){
```

```
        fputs(s, fPtr); //将一行字符写入文件中
```

```
        fgets(s,sizeof(s),stdin);
```

```
    }
```

```
    fclose(fPtr);
```

```
} //end of if
```

```
}
```

This is an example

To read and write

^Z

This is an example

To read and write

文件

键盘输入

[例4] 设计函数：将一个文本文件逐行复制到另一文本文件中。



Beijing University of Posts and Telecommunications

```
void copyFile(char * sourceFileName, char * destFileName)
{
    char s[100];
    FILE * sourcefPtr,*destfPtr;
    if((sourcefPtr=fopen(sourceFileName,"r"))==NULL)
        printf("can't open the source file\n");
    else if((destfPtr=fopen(destFileName,"w"))==NULL)
        printf("can't open the dest file\n");
    else{
        fgets(s,sizeof(s),sourcefPtr); // 从源文件读取一行字符到数组s
        while(!feof(sourcefPtr)){
            fputs(s,destfPtr); //将s中的字符串写入文件中
            fgets(s,sizeof(s),sourcefPtr);
        }
        fclose(sourcefPtr);
        fclose(destfPtr);
    }
}
```

读 / 写一个字符串——fgets()和fputs()



思考

```
fgets(s,sizeof(s),stdin);  
while(!feof(stdin)){  
    fputs(s,stdout);  
    fgets(s,sizeof(s),stdin);  
}
```

输入:

This is an example

^Z

输出:

This is an example

This is an example

将以上语句改写成以下语句会两次(如右图所示)！为什么？

```
while(!feof(stdin)){  
    fgets(s,sizeof(s),stdin);  
    fputs(s,stdout);  
}
```

1. 只有当读取了文件结束符后，feof才返回1。
2. fgets如果遇到文件结束符并且没有字符读入数组，则数组内容不变

读 / 写一个字符串——fgets()和fputs()



北京邮电大学
Beijing University of Posts and Telecommunications

总结读文件操作:

.....

打开文件;

读取一个(组) 数据;

```
while(!feof(文件指针)){
```

```
    处理读取的数据;
```

```
    读取下一个(组) 数据;
```

```
}
```

关闭文件;

.....

对文件进行格式化读 / 写——**fscanf()** 和 **fprintf()** 函数



Beijing University of Posts and Telecommunications

一、再谈**scanf**函数：

int scanf(const char * format, 输入变量首地址表)

scanf函数从**stdin**指向的标准输入流中读取字符序列，按照**format**中的格式规格说明对字符序列进行解释，并把结果存储在变量中。

当**scanf**函数使用完了格式输入串或当一些输入无法与控制说明相匹配时，它就停止运行，并返回成功匹配和赋值的输入项的个数。遇到文件的结尾，**EOF**被返回（**EOF**一般是-1）。下一次对这个函数的调用将从上一次转换的最后一个字符的下一个字符开始继续搜索。

对文件进行格式化读 / 写——**fscanf()** 和 **fprintf()** 函数



Beijing University of Posts and Telecommunications

***fscanf()**和**fprintf()**函数，用于ASCII文件的处理。*

其功能与**scanf()**和**printf()**函数相似，区别在于：**fscanf()**和**fprintf()**函数的操作对象是指定文件，而**scanf()**和**printf()**函数的操作对象是标准输入输出文件。

1、**int fscanf(FILE * stream, const char * format, 输入变量首地址表);/*读文件*/**

如：**fscanf(fPtr,"%d,%f", &i, &f);**

fscanf (stdin, const char * format, 输入变量首地址表)
等价于

scanf (const char * format, 输入变量首地址表)

对文件进行格式化读 / 写——**fscanf()** 和 **fprintf()** 函数



2、**int fprintf(FILE * stream, const char * format , 输出参量表); /*写文件*/**

例如,

```
int i=3; float f=9.80;
```

```
fprintf(fPtr,"%2d,%6.2f", i, f);
```

.....

fprintf()函数的作用是, 将变量**i**按**%2d**格式、变量**f**按**%6.2f**格式, 以逗号作分隔符, 输出到**fPtr**所指向的文件中: □3,□□9.80 (□表示1个空格)。

fprintf (stdout, const char * format, 输出变量首地址表)
等价于

printf (const char * format, 输出变量首地址表)

读 / 写一个数据块——fread()和fwrite()



实际应用中，常常要求1次读 / 写1个数据块。为此，**ANSI C** 标准设置了 **fread()** 和**fwrite()**函数，用于读写二进制文件。

1. 用法：

int fread(void *buffer, int size, int count, FILE * stream);

从**stream**所指向的流中读取数据到**buffer**所指向的数组中，**size**表示单个数组元素的大小，最多读取**count**个数组元素，流的文件位置指针根据成功写入的字节数递增。函数返回成功读入的元素个数。如果发生读错误，则返回值可能小于**count**。

四、读 / 写一个数据块——fread()和fwrite()

读 / 写一个数据块——**fread()**和**fwrite()**



```
int fread(void *buffer, int size, int count,  
FILE * stream);
```

```
struct student stu[3], aStu;
```

```
// 从文件读取3条记录写入数组stu中
```

```
fread(stu, sizeof(struct student ), 3 , stream);
```

```
//一条记录写入结构变量aStu中
```

```
fread(& aStu, sizeof(struct student ), 1, stream);
```

```
//读取一个字节写入字符变量ch中
```

```
fread(& ch, sizeof(char), 1, stream);
```

读 / 写一个数据块——**fread()**和**fwrite()**



北京邮电大学
Beijing University of Posts and Telecommunications

```
int fwrite(void *buffer, int size, int count,  
FILE * stream);
```

fwrite()——将 **buffer** 所指向的数组的内容写入 **stream** 所指向的流中。**size** 表示单个数组元素的大小，最多写入 **count** 个数组元素。流的文件位置指针根据成功写入的字节数递增。函数返回成功写入的元素个数，如果遇到写错误，返回值可能小于 **count**。

fread() 和 **fwrite()** 函数，用于二进制文件的处理（因为二进制文件存储记录时每一个记录是等长的）。

读 / 写一个数据块——**fread()**和**fwrite()**



```
int fwrite(void *buffer, int size, int count, FILE  
* stream);
```

```
struct student stu[3], aStu;
```

```
//把数组stu中所有元素一次性写入文件中
```

```
fwrite(stu, sizeof(struct student ), 3 , stream);
```

```
//把变量aStu中内容写入文件中
```

```
fwrite(& aStu, sizeof(struct student ), 1, stream);
```

```
//读取一个字节写入字符变量ch中
```

```
fwrite(& ch, sizeof(char), 1, stream);
```

例1：用文本文件保存不同类型的数据



```
#include<stdio.h>
int main(void)
{
    char ch='a',ch1;
    short a=10,a1;
    int b=100,b1;
    long c=1000,c1;
    float d=1.0,d1;
    double e=100.10,e1;
    int arr1[5]={1,2,3,4,5},arr[5];
    FILE * fptr1;
    int i;
```

例1：用文本文件保存不同类型的数据



北京邮电大学
Beijing University of Posts and Telecommunications

//将上述变量写入文本文件types.txt中

```
if( (fptr1=fopen("types.txt","w"))==NULL)
```

```
    printf("can't open file1\n");
```

```
else{
```

```
    fprintf(fptr1,"%c\n",ch);
```

```
    fprintf(fptr1,"%d\n",a);
```

```
    fprintf(fptr1,"%d\n",b);
```

```
    fprintf(fptr1,"%ld\n",c);
```

```
    fprintf(fptr1,"%f\n",d);
```

```
    fprintf(fptr1,"%lf\n",e);
```

```
    for(i=0;i<5;i++)
```

```
        fprintf(fptr1,"%3d",arr1[i]);
```

```
    fclose(fptr1);
```

```
}
```

例1：用文本文件保存不同类型的数据



```
//从文本文件types.txt中读取不同类型的数据到变量
if( (fptr1=fopen("types.txt","r"))==NULL)
    printf("can't open file1\n");
else{
    fscanf(fptr1,"%c",&ch1);
    fscanf(fptr1,"%d",&a1);
    fscanf(fptr1,"%d",&b1);
    fscanf(fptr1,"%ld",&c1);
    fscanf(fptr1,"%f",&d1);
    fscanf(fptr1,"%lf",&e1);
    for(i=0;i<5;i++)
        fscanf(fptr1,"%d",&arr[i]);
    fclose(fptr1);
}
```


例1：用文本文件保存不同类型的数据



//输出变量到显示器

```
printf("ch1=%c\n",ch1);
```

```
printf("a1=%d\n",a1);
```

```
printf("b1=%d\n",b1);
```

```
printf("c1=%ld\n",c1);
```

```
printf("d1=%f\n",d1);
```

```
printf("e1=%lf\n",e1);
```

```
for(i=0;i<5;i++)
```

```
    printf("a[%d]=%d\n",i,arr[i]);
```

```
}
```

```
}
```

例2：用二进制文件保存不同类型的数据



北京邮电大学
Beijing University of Posts and Telecommunications

```
#include<stdio.h>
int main()
{
    char ch='a',ch1;
    short a=10,a1;
    int b=100,b1;
    long c=1000,c1;
    float d=1.0,d1;
    double e=100.10,e1;
    int arr1[5]={1,2,3,4,5},arr[5];
    FILE * fptr;
    int i;
```

例2：用二进制文件保存不同类型的数据



北京邮电大学
Beijing University of Posts and Telecommunications

```
//将上述变量写入二进制文件 types.dat
if( (fptr=fopen("types.dat","wb"))==NULL)
    printf("can't open file1\n");
else{
    fwrite(&ch,sizeof(char),1,fptr);
    fwrite(&a,sizeof(short),1,fptr);
    fwrite(&b,sizeof(int),1,fptr);
    fwrite(&c,sizeof(long),1,fptr);
    fwrite(&d,sizeof(float),1,fptr);
    fwrite(&e,sizeof(double),1,fptr);
    fwrite(arr1,sizeof(int),5,fptr);
    fclose(fptr);
}
```

例2：用二进制文件保存不同类型的数据



```
//从二进制文件types.dat中读取上述数据进行输出
if( (fptr=fopen("types.dat","rb"))==NULL)
    printf("can't open file1\n");
else{
    fread(&ch1,sizeof(char),1,fptr);
    fread(&a1,sizeof(short),1,fptr);
    fread(&b1,sizeof(int),1,fptr);
    fread(&c1,sizeof(long),1,fptr);
    fread(&d1,sizeof(float),1,fptr);
    fread(&e1,sizeof(double),1,fptr);
    fread(arr,sizeof(int),5,fptr);
    fclose(fptr);
}
```

例2：用二进制文件保存不同类型的数据



北京邮电大学
Beijing University of Posts and Telecommunications

```
printf("ch1=%c\n",ch1);
printf("a1=%d\n",a1);
printf("b1=%d\n",b1);
printf("c1=%ld\n",c1);
printf("d1=%f\n",d1);
printf("e1=%lf\n",e1);
for(i=0;i<5;i++)
    printf("a[%d]=%d\n",i,arr[i]);
}
return 0;
}
```

- 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



13.6 顺序存取文件的操作



- 顺序存取文件特点：
 - 是文本文件，使用**fscanf**和**fprintf**函数。
 - 文件中的记录可以有不同的长度。
 - 不能直接快速地访问文件中的某一记录，而必须从文件中第一个记录开始访问。
 - 新的记录只能插入到文件尾。

[例5]设计函数，将键盘上输入若干记录（以Ctrl+Z结束），存储到指定顺序文件中

void writeFileFromKeyboard(char * filename)

```
{
    int account;
    char name[30];
    float balance;
    FILE * fPtr=NULL;
    if( (fPtr=fopen(filename,"w"))!=NULL){//打开文件
        scanf("%d%s%f",&account,name,&balance);
        while ( !feof(stdin)){ //组合键ctrl+Z表示文件结束符
            fprintf(fPtr,"%5d%13s%10.2f\n",account,name,bal
            ance);
            scanf("%d%s%f",&account,name,&balance);
        }
        fclose(fPtr); /*关闭文
    }
}
```

//end of if

注意：将一个记录写入文件时，必须将\n作为记录结束符写入文件，否则从文件中读取记录时最后一个记录无法读取！

1001 zhou 102

1002 yang 103

^Z

键盘 输入

1001 zhou 102.00

1002 yang 103.00

磁盘文件

[例6] 设计函数：将一顺序文件逐记录复制到另一顺序文件中。

```
void copyFile(char * sourceFileName,char *
    destFileName)
{
    int account;
    char name[30];
    float balance;
    FILE * sourcefPtr,*destfPtr;

    if((sourcefPtr=fopen(sourceFileName,"r"))==NULL)
        printf("can't open the source file\n");
    else if((destfPtr=fopen(destFileName,"w"))==NULL)
        printf("can't open the dest file\n");
```



```
else{
    //从源文件中读取一条记录
    fscanf(sourcefPtr,"%d%s%f",&account,name,&balance);

    while(!feof(sourcefPtr)){
        //向目标文件写入一条记录

        fprintf(destfPtr,"%5d%13s%10.2f\n",account,name,balance);
        fscanf(sourcefPtr,"%d%s%f",&account,name,&balance);
    }

    fclose(sourcefPtr);
    fclose(destfPtr);
}
}
```

- 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



13.7 随机存取文件的操作



一、随机存取文件特点：

- 是二进制文件，使用**fread**和**fwrite**函数
- 文件中的记录具有相同的长度。
- 能够直接快速地定位、访问文件中的某一记录。
- 新记录可以插入到希望的位置。
- 为了能在文件中不同的位置随机写入记录，必须对文件先初始化。

13.7 随机存取文件的操作



二、按顺序建立一个随机存取文件（文件初始化）

```
#include<stdio.h>
struct clientData
{
    int acctNum;
    char lastName[15];
    char firstName[10];
    float balance;
};
```



```
main()
{
    int i;
    struct clientData blankClient={0,"","",0.0};
    FILE * cfPtr;
    if ( (cfPtr=fopen("client.dat","wb"))==NULL ) /*打开文件*/
        printf("File could not be opened\n");
    else{
        for(i=1;i<=100;i++)
            fwrite(&blankClient,sizeof(struct clientData),1,cfPtr);
        fclose(cfPtr);
    }
    return 0;
}
```

因为用fwrite函数，所以创建的是二进制文件

13.7 随机存取文件的操作



三、把数据随机地写入随机存取文件

```
main()
```

```
{
```

```
    struct clientData client;
```

```
    FILE * cfPtr;
```

```
    if ( (cfPtr=fopen("client.dat","rb+"))==NULL )  
        printf("File could not be opened\n");
```

思考：文件打
开模式设为
“wb”可否？



```
else{
    printf("Enter account number(1 to 100,0 to end input)\n");
    scanf("%d",&client.acctNum);
    while (client.acctNum!=0){
        printf("Enter lastname,firstname,balance\n?");

        scanf("%s%s%f",client.lastName,client.firstName,&client.balance);
        //根据帐户号码将文件位置指针定位到待写入位置，然后写入记录
        fseek(cfPtr,(client.acctNum-1)*sizeof(struct
clientData),SEEK_SET);
        fwrite(&client,sizeof(struct clientData),1,cfPtr);
        printf("Enter account number(1 to 100,0 to end input)\n");
        scanf("%d",&client.acctNum);
    }
    fclose(cfPtr);
}
return 0;
```


13.7 随机存取文件的操作



四、按顺序读取一个随机存取文件

```
main()
{
    struct clientData client;
    FILE * cfPtr;

    if ( (cfPtr=fopen("client.dat","rb"))==NULL )
        printf("File could not be opened\n");
```

13.7 随机存取文件的操作



```
else{
    printf("%-6s%-16s%-
11s%10s\n","Acct","LastName","FirstName","Balance");

    fread(&client,sizeof(struct clientData),1,cfPtr);
    while ( !feof(cfPtr)) {
        if (client.acctNum!=0)
            printf("%-6d%-16s%-11s%10.2f\n",
                client.acctNum,client.lastName,client.firstName,client.balance);
        fread(&client,sizeof(struct clientData),1,cfPtr);
    }
    fclose(cfPtr);
}
return 0;
}
```



五. 从随机文件中一次性读取多条记录到数组中

```
main()
```

```
{
```

```
    struct clientData client[3];
```

```
    FILE * cfPtr;
```

```
    if ( (cfPtr=fopen("client.dat","r"))==NULL )
```

```
        printf("File could not be opened\n");
```

```
    else{
```

```
    /*从文件中读取3块内容，每块大小为记录长度，放到数组中*/
```

```
        fread(clients, sizeof(struct clientData ),3,cfPtr);
```

```
        fclose(cfPtr);
```

```
    }
```

```
    .....
```

```
}
```



六. 将数组内容一次性写入随机文件中

main()

```
{  
    struct clientData client[3];  
    FILE * cfPtr;  
    if ( (cfPtr=fopen("client.dat","w"))==NULL )  
        printf("File could not be opened\n");  
    else{  
        ...../*设置数组元素的值*/  
        /*将数组中前三个元素写入文件中*/  
        fwrite(clients, sizeof(struct clientData ),3,cfPtr);  
        fclose(cfPtr);  
    }
```



- 可能会发生的问题：
- 文件路径：**fopen("c:\\newdir\\file.dat", "r")**
- 二进制文件打开方式不要忘记**b**，如“**rb**”、“**wb**”
- 在读写“**+**”模式下，读操作、写操作之前必须要用**fseek**来定位文件位置指针。例如：
fseek fread; fseek fwrite
- 不能用**fprintf**向二进制文件写记录！

