西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

C程序设计

Programming in C

1011014

主讲：姜学锋，计算机学院

用C程序读写文件

1、文件读写操作的基本形式

2、从文件中读写字符与字符串数据

10.3 文件读写操作

- 只要文件创建或打开后，数据就能顺利地写入到文件中，而文件读入前需要判断是否还有数据可以读入（即文件是否到末尾）。

10.3.1 文件读写操作的基本形式

- 文件读写操作过程基本上是通用的，写操作可以直接调用文件写函数，而读操作的基本形式为：

```
while (!feof(fp)) { //fp文件是否到末尾
   ......//调用文件读函数
}
```

- feof函数为真表示已到文件末尾，逻辑取反使while语句的条件为如果文件没有到末尾继续执行循环。当有多个文件操作时，循环条件应是多个feof函数的组合逻辑。

10.3.2 读写字符数据

- fgetc函数从文件中读入一个字符数据，其函数原型为：

```
int fgetc(FILE *stream); //读文件字符数据函数
```

- 参数stream是已打开的文件指针，该文件必须是以读或读写方式打开的。fgetc函数返回读取从文件中得到的字符数据（低8位），如果在文件末尾则返回EOF。EOF是在头文件stdio.h中定义的符号常量，值为－1。

10.3.2 读写字符数据

- fputc函数将一个字符数据写入文件中，其函数原型为：

```
int fputc(int c, FILE *stream); //写文件字符数据函数
```

- 参数stream是已打开的文件指针，该文件必须是以写、读写或追加方式打开的。参数c是输出字符，使用低8位。如果写入成功函数返回非零值，否则返回EOF。

- 使用fgetc和fputc函数可以处理文本文件和二进制文件。

10.3.2 读写字符数据

【例10.2】

复制源文件到目的文件，支持命令行文件名输入。

10.3.2 读写字符数据

例10.2

```
1  #include <stdio.h>
2  #include <string.h>
3  int main(int argc, char *argv[])
4  { //使用带参数的main函数版本获取命令行信息
5    char src[260],dest[260];
6    FILE *in,*out;
7    if (argc<2) gets(src); //若无命令行参数输入源文件名
8    else strcpy(src,argv[1]); //否则第1个命令行参数为源文件名
9    if (argc<3) gets(dest); //若只有1个命令行参数输入目的文件名
10   else strcpy(dest,argv[2]); //否则第2个命令行参数为目的文件名
11   in=fopen(src,"rb"); //打开源文件读
12   if (in!=NULL) {
13     out=fopen(dest,"wb"); //创建目的文件写
14     while (!feof(in)) //是否到源文件末尾
15       fputc(fgetc(in),out); //从源文件读一个字符写入到目的文件
```

10.3.2 读写字符数据

例10.2

```
16        fclose(out); //关闭目的文件
17        fclose(in); //关闭源文件
18    }
19    return 0;
20 }
```

程序支持从命令行输入源文件名（第1个参数）和目的文件名（第2个参数），若未提供相应的命令行参数，程序运行时会要求输入。假定程序取名dup，则命令行形式为：

C:\>dup src.dat dest.dat

程序运行后将源文件src.dat复制得到目的文件dest.dat。

10.3.3 读写字符串数据

- fgets函数从文件中读入一行字符串，其函数原型为：

```
char *fgets(char *string, int n, FILE *stream);
//读文件字符串函数
```

10.3.3 读写字符串数据

- 参数stream是已打开的文件指针，该文件必须是以读或读写方式打开的。string是字符数组或能容纳字符串的内存区起始地址（如动态分配得到的内存区），用于存储读取到的字符串。参数n表示最大读取多少个字符，一般由string内存长度决定。fgets函数将文件中的一行字符串读入到string所指的内存区存放，文件中字符串按（\n和\r）分行，实际读入的字符串长度小于等于n，且读进来的字符串自动在末尾增加一个空字符（\0）作为结束标记。如果读取成功fgets函数返回string指针，否则返回NULL指针。

10.3.3 读写字符串数据

- 例如：

```
char str[100], A[3][80], *p;
fgets(str,sizeof(str)-1,fp);
//读1个字符串，需要为空字符（\0）留下位置
fgets(A[0],80-1,fp); //读1个字符串，参数string为字符串数组元素（地址）
p=(char*)malloc(1000*sizeof(char));
//动态分配能容纳1000个字符的字符串
fgets(p,1000-1,fp); //读1个字符串，参数string为指针值（地址）
```

- 如果string参数实际内存长度不足以存储读入数据时，fgets函数会导致崩溃性错误。

10.3.3 读写字符串数据

- fputs函数将一行字符串写入文件中，其函数原型为：

```
int fputs(const char *string, FILE *stream);
//写文件字符串函数
```

- 参数stream是已打开的文件指针，该文件必须是以写、读写或追加方式打开的。参数string是输出字符串，输出到文件后自动增加一个换行（\r）。如果写入成功fputs函数返回非零值，否则返回EOF。
- 使用fgets和fputs函数处理由一行一行字符串组成的文本文件最方便。

10.3.3 读写字符串数据

【例10.3】

将源文件每行文本前添加一个行号输出到目的文件中。

10.3.3 读写字符串数据

例10.3

```
1  #include <stdio.h>
2  int main()
3  {
4    char s1[100],s2[110];
5    int cnt=0;
6    FILE *in,*out;
7    in=fopen("a.c","r"); //打开源文件读
8    if (in!=NULL) {
9      out=fopen("b.c","w"); //创建目的文件写
10     while (!feof(in)) { //是否到源文件末尾
11       if (fgets(s1,sizeof(s1)-1,in)==NULL) continue;
12       sprintf(s2,"%04d %s",++cnt,s1); //将字符串添加行号输出
13       fputs(s2,out); //输出到目的文件
14     }
15     fclose(out); //关闭目的文件
```

10.3.3 读写字符串数据

例10.3

```
16        fclose(in); //关闭源文件
17    }
18    return 0;
19 }
```

10.3.3 读写字符串数据

- 程序运行前提供a.c文件，运行后产生b.c新文件，两个文件的内容如下：

假定a.c内容：

```
#include <stdio.h>
int main()
{
  printf("hello,world\n");
  return 0;
}
```

则输出b.c内容：

```
0001 #include <stdio.h>
0002 int main()
0003 {
0004   printf("hello,world\n");
0005   return 0;
0006 }
```

10.3.4 读写格式数据

- fscanf函数从文件中读入格式化数据，其函数原型为：

```
int fscanf(FILE *stream, const char *format [,argument ]... );
//读格式化数据函数
```

- 参数stream是已打开的文件指针，该文件必须是以读或读写方式打开的。format 是格式字符串，用于指明输入数据的格式。后面参数是输入项，可以是任意数目，均按地址形式提供。
- fscanf函数返回成功读取到的输入项个数，非读取到字符数，如果有错误发生（如文件读入错误、按格式得不到输入项）返回EOF。

10.3.4 读写格式数据

- fscanf函数与scanf函数很相似，只不过fscanf函数从文件读取输入数据，scanf函数从键盘设备读取输入数据，因此scanf函数实质上就是：

```
int fscanf(stdin, const char *format [,argument ]... );
```

10.3.4 读写格式数据

- fprintf函数将格式化数据写入文件中，其函数原型为：

```
int fprintf(FILE *stream, const char *format [,argument ]...);
//写格式化数据函数
```

- 参数stream是已打开的文件指针，该文件必须是以写、读写或追加方式打开的。format 是格式字符串，用于指明输出数据的格式。
- 如果写入成功，fprintf函数返回写入的总字节数，否则返回负值。
- 使用fscanf和fprintf函数处理格式化数据的文本文件最方便。

10.3.4 读写格式数据

- fprintf函数与printf函数很相似，只不过fprintf函数向文件写入输出数据，printf函数向显示器设备写入输出数据，因此printf函数就是：

```
int fprintf(stdout, const char *format [,argument ]...);
```

10.3.4 读写格式数据

【例10.4】

已知文件book.dat中有100个书籍销售记录，每个销售记录由代码（字符串4位）、书名（字符串10位）、单价（整型）、数量（整型）、金额（整型）五个部分组成。文件每行包含代码、书名、单价、数量数据，用TAB间隔，如下格式：

1001 软件世界　5　　100
1002 计算机工程 6　　120
……

其中金额＝单价×数量。读取这100个销售记录，按金额从大到小进行排列，若金额相等，则按代码从小到大排列，将最终结果输出到out.dat中。

10.3.4 读写格式数据

例题分析

根据题意先设计结构体类型描述书籍销售记录，100个书籍销售记录用结构体数组表示。

10.3.4 读写格式数据

例10.4

```
#include <stdio.h>
#include <string.h>
typedef struct tagBOOK { //书籍销售记录类型
  char c[5]; //产品代码
  char n[11]; //产品名称
  int p; //单价
  int q; //数量
  int t; //金额
} BOOK;
int main()
{
  FILE *fp;
  BOOK A[100],t; //书籍销售记录数组
  int i,j,cnt=0;
  fp=fopen("BOOK.dat","r"); //打开文件读
```

10.3.4 读写格式数据

例10.4

```
 16    if (fp==NULL) return -1; //文件打开失败退出运行
 17    while(!feof(fp)) { //是否读到文件末尾
 18      fscanf(fp,"%s%s%d%d",A[cnt].c,A[cnt].n,&A[cnt].p,&A[cn
t].q);
 19      A[cnt].t=A[cnt].p*A[cnt].q; //计算金额
 20      cnt++;
 21    }
 22    fclose(fp);
 23    for (i=0; i<cnt-1; i++) //冒泡法排序
 24    for (j=i+1; j<cnt; j++)
 25      if ( (A[i].t<A[j].t) || //按金额由大到小
 26        (A[i].t==A[j].t && strcmp(A[i].c,A[j].c)>0) )
 27          t=A[i], A[i]=A[j], A[j]=t;
 28    fp=fopen("out.dat", "w");
 29    for(i=0; i<100; i++)
```

10.3.4 读写格式数据

例10.4

```
 30      fprintf(fp,"%s %s %d %d %d\n",A[i].c,A[i].n,A[i].p,A
[i].q,A[i].t);
 31    fclose(fp) ;
 32    return 0;
 33 }
```


程序设计