第14讲
数据文件处理技术

周水庚

2017年12月21日

提要

- C文件概述
- 文件类型和文件类型指针变量
- 文件打开和关闭库函数
- 文件处理程序结构和常用文件库函数
- 文件程序设计实例

提要

- C文件概述
- 文件类型和文件类型指针变量
- 文件打开和关闭库函数
- 文件处理程序结构和常用文件库函数
- 文件程序设计实例

C文件概述

- 计算机将数据存储在外部存储介质中，如磁带、磁盘等。操作系统将存储在外部存贮介质中的数据以数据流的形式来组织
- 每个独立的数据流称作文件，每个文件有一个名字。为便于管理，操作系统维持一个呈层次状的目录结构。每个文件都被置某一目录下，文件用文件名(包括文件的目录路径)来标识
- 从键盘输入的数据流和向显示屏或行印机输出的数据流也称作文件
- 在C语言中，文件中的组织形式有两种
  - 数据流由一个个字符组成，称为正文（text）文件
  - 数据流由二进制字节代码组成，称为二进制（binary）文件

C文件概述

- 正文文件(或者文本文件)
  - 文件中数据是字符，每个字符以ASCII 代码存储，占一个字节
  - 存储数值数据要占较多的存储空间。输入输出时因内存和外存的存储形式不一致，还要进行内外表示形式之间的转换
  - 正文文件数据流是字符，能让程序对文件作逐个字符处理和供人阅读

- 二进制文件
  - 文件中的数据按其在内存中的存储形式存储在文件中
  - 存储数值数据只占其内部表示所需的字节数，可以节省外存空间，并免去数据内外表示形式之间的转换
  - 用于程序与程序或程序与设备之间传递成批数据信息

C文件概述

- 程序与文件交换数据的实现过程
  - 设备的读写操作是在操作系统的管理和控制下进行
  - 操作系统为能高效地管理和控制设备，给程序正在使用的每个文件在内存中开辟一个适当大小的缓冲区
  - 程序要从文件读入一些信息时，系统先一次性地读入足够多的信息存于缓冲区中，供程序一部分一部分地使用
  - 程序要写信息到文件时，也先把信息写到缓冲区中，待缓冲区写满或写文件结束时，才把缓冲区中的内容写到文件中

C文件概述

- 程序要读取文件中的数据，首先打开要读的文件，然后才能从该文件读取数据，并在使用结束时，及时关闭文件
- 程序要向文件写入数据，也是先打开文件(可能包括建立一个新文件)，然后向文件输出数据，最后关闭文件
- C语言本身未提供有关文件操作的输入输出语句，对文件的打开关闭和读写操作都用系统提供的库函数来实现

提要

- C文件概述
- 文件类型和文件类型指针变量
- 文件打开和关闭库函数
- 文件处理程序结构和常用文件库函数
- 文件程序设计实例

文件类型和文件类型指针变量

- 文件类型 **FILE**
  - 操作系统除为文件开辟缓冲区外，还为每个被使用的文件在内存中开辟另一个存储区，用于存放有关对文件进行操作所需的控制信息，如：文件名、文件读写状态、文件缓冲区大小和位置、当前读写位置等
  - **C**系统将这些信息存储在一个结构变量中，这种结构的类型由系统预定义，取名**FILE**，习惯称文件类型
  - 不同系统的文件类型所含内容也不全相同，可查阅所使用的C系统的 **stdio.h**文件，在该文件中有**FILE**的定义

提要

- C文件概述
- 文件类型和文件类型指针变量
- **文件打开和关闭库函数**
- 文件处理程序结构和常用文件库函数
- 文件程序设计实例

文件打开库函数fopen()

- 读写文件之前先打开文件，使用库函数fopen()
- 调用函数fopen() 的一般形式为：

  fopen(文件名，使用方式)

  - 文件名(可能还包括目录路径)为字符串表达式。使用方式也是一个字符串，用来指明文件的读写方式
  - 函数fopen()将返回一个存放文件控制信息结构的指针，该文件的有关控制信息就将存放在该结构中，供以后读写文件操作之用
  - 程序应将调用函数fopen()返回的指针值赋给某个文件指针变量保存
  - 语句 fp = fopen("\\usr4\\smp.dat", "r") 以读方式打开根目录下的usr4子目录中的 smp.dat 文件

文件打开库函数fopen()

- 调用函数fopen()时，可能不能打开指定的文件
  - 读方式下打开不存在的文件
  - 写方式下，外部存贮介质已无剩余的自由空间
  - 外设故障或超过系统能同时打开的文件数
- 文件不能打开时，函数fopen()将返回一个空指针值NULL。考虑到文件不能正常打开情况，常用C代码描述打开文件如

```
if ((fp = fopen(filename, "r")) == NULL)  {
    printf("Can not open %s file.\n", filename);
    return;
}
```

  - 以上代码以读方式打开一个文件，其中 filename 是表示文件名的字符数组或者指针。在调用函数fopen()后立即检查打开是否成功，如果打开不成功，就输出该文件不能打开信息并返回

文件打开库函数fopen()

表8-1

文件使用方式	意义
"r"	只读，为读打开正文文件
"w"	只写，为写打开正文文件
"a"	追加，从正文文件尾开始写
"rb"	只读，为读打开二进制文件
"wb"	只写，为写打开二进制文件
"ab"	追加，从二进制文件尾开始写
"r+"	读写，为读/写打开正文文件
"w+"	读写，为读/写建立并打开新的正文文件
"a+"	读写，为读/写打开正文文件
“rb+"或“r+b"	读写，为读/写打开二进制文件
“wb+"或“w+b"	读写，为读/写建立并打开新的二进制文件
"ab+"或"a+b"	读写，为读/写打开二进制文件

fopen()的使用方式参数说明

- “r”方式打开的文件只能用于从文件读入数据，且要求文件存在
- “w”方式打开的文件只能用于向文件写出数据。如打开时，文件不存在，则新建一个以指定名字命名的文件；如原文件已存在，则原文件上的数据被全部删除
- 打开文件用于写，不删除原文件中的数据，从原文件的末尾开始添加新数据，用 "a" 方式
- 用 “r+”、“w+”、“a+”方式，可以输入和输出。用“r+”方式打开已存在的文本文件；用“w+”方式可新建一个文本文件；用“a+”方式打开一个已存在的文本文件，位置指针先移到文件的末尾，准备添加数据，也可以输入数据
- 要打开二进制文件，只要在对应正文件文件打开方式中接上字符b即可，如”rb”表示以输入方式打开二进制文件

标准文件

- 正文文件与二进制文件在使用时，有一点不同
  - 对于正文文件，输入时，\r和\n合成为一个\n输入；输出时，\n转换成为\r和\n两个字符一起输出
  - 对于二进制文件，不进行上述这种转换，二进制文件中的数据形式与在内存中的数据形式是完全一致的

- 系统将常规设备上的输入输出数据流称为标准文件。程序运行时，系统自动打开这些标准文件。它们是：标准输入文件stdin 、标准输出文件stdout 、标准出错输出文件stderr和标准打印输出文件stdprn
- 程序除能直接使用前面各章都使用的不带文件指针的标准输入输出库函数外，也可使用后面介绍的带文件指针的一般形式的输入输出库函数
  - stdin指从终端输入数据；stdout是向终端输出数据

文件关闭库函数fclose()

- 在使用完一个文件后，程序应该立即关闭它。关闭文件可调用库函数 fclose() 来实现
- 调用函数 fclose() 的一般形式为

  fclose(文件指针);

  - 如 fclose(fp);
- 调用函数fclose()的作用是使文件指针变量终止原先调用函数fopen()时所建立的它与文件的联系
- 调用函数fclose()之后，不能再通过该文件指针变量对其原先相连的文件进行读写操作，除非被再次打开
- 文件被关闭后，原文件指针变量又可用来打开文件，或与别的文件相联系，或重新与原先文件建立新的联系

提要

- C文件概述
- 文件类型和文件类型指针变量
- 文件打开和关闭库函数
- **文件处理程序结构和常用文件库函数**
- 文件程序设计实例

文件处理程序结构和常用文件库函数

- 正文文件的输入处理
  - 从正文文件逐一输入字符，对输入的字符作某种处理的程序结构有：

```
int c;  /* 不能为char类型 */
  …      /* 说明有关变量和设置初值等 */
fp=fopen(文件名,"r");/* 正文文件以读方式打开 */
while((c = fgetc(fp)) != EOF) {
      … /* 这里对刚读入的字符信息 c 作某种处理  */
  }
fclose(fp);
…    /* 输出处理结果 */
```

文件处理程序结构和常用文件库函数

- 其中，函数fgetc()的说明形式为

  int fgetc(FILE *fp)

- 该函数的功能是从与fp相联系的文件中读入下一个字符
- 在文件的控制信息块中，有一个当前要读字符的位置信息，每读入一个字符后，在文件还未结束情况下，这个当前位置信息就移向其后一个字符，从而保证程序反复调用函数fgetc()能顺序读入文件中的字符
- 函数fgetc()的返回值就是读入字符的ASCII代码值
- 读入字符时，如遇到文件结束，函数返回文件结束标志EOF。对于正文文件，由于字符的ASCII代码不可能是-1，因此可用EOF(定义为-1)作为文件结束标志

getchar()和fgetc()

- getchar()是用fgetc()定义的宏
  - #define getchar fgetc(stdin)

```
while((c = getchar()) !='\n') { /*以return键结束
   … /* 这里对刚读入的字符信息 c 作某种处理 */
}
while((c = getchar()) != EOF) {/*以ctrl+z键结束
   … /* 这里对刚读入的字符信息 c 作某种处理 */
}
while((c = fgetc(stdin)) != EOF) {/*以ctrl+z键结束
   … /* 这里对刚读入的字符信息 c 作某种处理 */
}
```

文件处理示例程序-1

- 输入正文文件，统计文件中英文字母的个数，并输出。设程序要统计的正文文件名在程序启动时由输入指定

```
#include <stdio.h>
FILE *fp;
int main()
{ int count, ch;  char fname[40];
  printf("输入文件名!\n"); scanf("%s%*c", fname);  /* 读入文件名及其名后的回车符 */
  if ((fp = fopen(fname, "r")) == NULL){
     printf("Can not open %s file.\n", fname);
     return 0;  /* 程序非正常结束 */
  }
```

文件处理示例程序-1(续)

```
count = 0;
  while((ch = fgetc(fp)) != EOF) {
    /* 这里对刚读入的字符信息 ch 作某种处理 */
    if(ch>='a'&&ch<='z'||ch>='A'&&ch<='Z')
            count++;
  }
  fclose(fp);
  /* 输出处理结果 */
  printf("文件%s有英文字母%d个.\n",fname, count);
  return 1; /* 程序执行正常结束 */
}
```

文件处理程序结构和常用文件库函数

- 二进制文件的输入处理
  - 从二进制文件逐一输入字节，并作某种处理

```
char c;  /* 也可以是int类型 */
 …   /* 说明有关变量和设置初值等 */
fp = fopen(文件名，"rb");
while(!feof(fp)) {
    c = fgetc(fp);
    …   /* 这里对字节信息 c 作某种处理 */
 }
fclose(fp);
… /* 输出处理结果 */
```

文件处理程序结构和常用文件库函数

- 函数 feof()用来判断文件是否结束
- 函数调用feof(fp)用来测试与fp相联系的文件当前状态是否为“文件结束”状态
  - 如果是文件结束状态，函数调用feof(fp)返回非零值，否则返回零值
  - 函数feof()也可用于测试正文文件
- 对于二进制文件，读入的某个字节信息变换成整数后可能是-1，所以一般不能以读入字节的值是否为-1来判定二进制文件是否结束，而应该用函数feof()

文件处理程序结构和常用文件库函数

- 逐一输入字符(或字节)，生成形成新文件
  - 字符(或字节)逐一生成输出，形成新文件的程序结构

```
int c;  /* 也可以是char类型 */
  … /* 说明有关变量和设置初值等 */
fp=fopen(文件名,"w"); /* 或fp=fopen(文件,"wb") */
while((c=getchar())!='\n') {
      … /* 生成字符(或字节)存于变量c */
   fputc(c, fp); /* 将生成的字符(或字节)输出 */
 }
fclose(fp);
 … /* 输出程序结束报告 */
```

文件处理程序结构和常用文件库函数

- 函数 fputc()的说明形式为

  int fputc(char ch, FILE *fp)

  - 该函数的功能是将 ch 中的字符输出到与fp文件指针相联系的文件中。函数fputc() 返回一个整值。如果输出成功，则返回值就是输出字符的ASCII代码值；如果输出失败，则返回 EOF，即 -1
- 函数 putchar() 是在 stdio.h 中用函数 fputc() 定义的宏：

  #define putchar(c) fputc(c, stdout)
- 用宏 putchar(c) 比写 fputc(c, stdout) 在概念上更简单一些。从使用者来说，可以把 putchar(c) 看作函数调用，不必严格地称它为宏调用

文件处理示例程序-2

- 逐行复制从键盘读入字符到指定文件，直至输入空行结束。要求复制时，在数字符序列与其它字符序列之间插入１个空格符。在复制过程中，程序还统计输入字符和输出字符个数，并将统计结果输出
  - 程序为了判定最近处理字符和当前字符的属性，即是数字符或是其它字符，引入标志变量pre_d和c_d。是数字符为1，不是数字符为0

```
#include <stdio.h>
#define INT_CH ''
FILE *fp;
void main()
{ int incount, outcount, ch, pre_d, c_d;
  char fname[40];
```

文件处理示例程序-2(续)

```
printf("输入文件名!\n");scanf("%s%*c",fname);
fp=fopen(fname,"w"); /* 以写方式打开正文文件 */
incount = outcount = 0;
while((ch=fgetc(stdin))!='\n') { /* 逐行处理 */
  incount++; pre_d = ch >= '0' && ch <= '9';
  fputc(ch, fp); outcount++;
  while ((ch = fgetc(stdin)) != '\n') {
    incount++;  c_d = ch >= '0' && ch <= '9';
    if (c_d != pre_d) {
      fputc(INT_CH, fp);  pre_d = c_d;
      outcount++;
    }
    fputc(ch, fp);    outcount++;
  }
```

文件处理示例程序-2(续)

```
    incount++;
    fputc(ch, fp);  /* 输出换行符 */
    outcount++;
  }
  incount++;/* 统计最后一个换行符，但不输入 */
  fclose(fp);
  /* 输出处理结果 */
  printf("共输入%d个字符，输出%d个字符。\n", incount, outcount);
}
```

文件处理程序结构和常用文件库函数

- 参照已知文件，生成新文件
  - 参照已知文件，生成新文件，需同时打开两个文件，一个用于读，另一个用于写
  - 以读文件的状态为循环控制条件
  - 在循环过程中，程序将读入内容作某种处理后生成新的内容输出到新文件

文件处理示例程序-3

- 逐行复制已知原文件，生成新文件。要求新文件是行定长文件，即新文件的每行字符个数保持一样多。为此，程序逐行复制文件，当源文件的一行字符不足时，用空白符补充；当源文件一行字符太长时，就被截制成定长的多行复制

```
#include <stdio.h>
#define LEN 30
FILE *rfp, *wfp;
int main()
{ int inlen, ch;  char rfname[40], wfname[40];
  printf("输入源文件名!\n");scanf("%s%*c",rfname);
  printf("输入新文件名!\n");scanf("%s%*c",wfname);
  wfp=fopen(wfname,"w");/*以写方式打开正文文件 */
```

文件处理示例程序-3(续)

```
if ((rfp = fopen(rfname, "r")) == NULL) {
   printf("不能打开源文件 %s.\n", rfname);      return 0;
}
while((ch = fgetc(rfp)) != EOF) {
 inlen = 0;
 while (ch != '\n'||ch!= EOF) {
      fputc(ch, wfp);     inlen++;
      if (inlen == LEN) {  /* 对太长的行作截制 */
        fputc('\n', wfp); /* 换行 */         inlen = 0;
      }
     ch = fgetc(rfp);
 }
```

文件处理示例程序-3(续)

```
    if(inlen){ /* 可能一行的最后一段不足定长要求 */
        while (inlen++ < LEN)  fputc(' ', wfp);
        if(ch!=EOF){fputc('\n', wfp); /* 换行 */  inlen = 0;}
        else break;
     }
   }

fclose(wfp);    fclose(rfp);
  printf("程序结束。\n");    return 1;
}
```

文件处理程序结构和常用文件库函数

- 结构化文件的输入输出
  - 应用程序直接以流式文件为基础编写，是一件较繁琐的工作
  - 应用程序可以把文件看作是结构化的文件。在结构化文件中，文件中的数据信息呈现某种结构形式
    - 人事档案管理应用中，代表每个人的信息是一个结构，读写的信息块以人的基本信息结构为单位，或每次读写一个人的信息或同时读写多个人的信息
  - 对于大容量文件，也希望能成批输入或输出，即程序每次调用输入输出库函数能交换更多字符或字节，这能大大地减少程序调用库函数的次数
- C的输入输出函数库也包含成批输入输出的库函数
  - 最常用的两个成批读写数据函数fread()和fwrite()

文件处理程序结构和常用文件库函数

- 函数fread()的说明形式为

  int fread(char *buf,int size,int count,FILE *rfp);
- 函数fwrite()的说明形式为

  int fwrite(char *buf,int size,int count,FILE*wfp);
  - buf是字符数组首元指针
    - 对fread()来说，是存放读入数据的开始地址
    - 对 fwrite()来说，是要输出数据的开始地址
  - size是读写的数据块的字节数
  - count为要进行读写的数据块的个数
  - fp为文件指针
  - 调用上述函数共读写size*count个字节或字符
- 如果是读写二进制文件，用函数 fread() 和 fwrite() 可以读写任何类型的信息

文件处理程序结构和常用文件库函数

- 如有一个如下形式的通信录结构类型

```
typedef struct  {
    char name[21];     /* 名字 */
    char phone[15];    /* 电话 */
    char  zip[10];     /* 邮编 */
    char addr[31];     /* 地址 */
} STU_TYPE;
```

  - 定义数组如STU_TYPE stud[30]; 表示结构数组stud[]能存放30个通信录数据
  - 则下面两个函数调用能分别实现20个通信录信息从某文件读出和写入某文件

```
fread(stud, sizeof(STU_TYPE), 20, rfp);
fwrite(stud, sizeof(STU_TYPE), 20, wfp);
```

文件处理程序结构和常用文件库函数

- 如要逐个读入或写出结构数组的每个元素，也可用循环如

```
for(i = 0; i < 20; i++)
  fread(&stud[i], sizeof(STU_TYPE), 1, rfp);
```

- 和

```
for(i = 0; i < 20; i++)
  fwrite(&stud[i], sizeof(STU_TYPE), 1, wfp);
```

- 调用函数fread()和fwrite()也有返回值，它的返回值是实际完成输入或输出的数据块的个数

文件处理示例程序-4

- 程序从键盘输入通信录信息，并把它们写到文件中。然后，从文件读出，并把它们显示在终端显示屏上
  - 从终端读入通信录数据，存于变量stud。在输入过程中，从输入字符流中截取通信录的一条有效信息存于stud中。然后程序又将这条信息存于文件中。接着程序又将它们从文件读出，存于stud，并将它显示在显示屏上

```
#include <stdio.h>
#include <string.h>
char   fname[40];
typedef struct  {
        char name[15];      /* 名字 */
        char phone[15];     /* 电话 */
        char  zip[10];      /* 邮编 */
        char addr[31];      /* 地址 */
} STU_TYPE;
```

文件处理示例程序-4(续)

```
STU_TYPE stud;   FILE *fp;   char s[80];
int main()
{ int count;
  printf("输入文件名!\n");  scanf("%s%*c", fname);
  if ((fp=fopen(fname, "wb")) == NULL)  {
      printf("Can not open %s file.\n", fname);
      return 0;
  }
  printf("Enter data[名 电话号码 邮编 地址].\n");
  for(count = 1;;count++) {
    printf("输入第 %d个通信录（空行结束）：  \n", count);
```

文件处理示例程序-4(续)

```
printf("名：");   gets(s);
if ( strlen(s) == 0) break;
strncpy(stud.name, s, 14);
printf("电话号码：");  scanf("%s", s);
strncpy(stud.phone, s, 14);
printf("邮编："); scanf("%s", s);
strncpy(stud.zip, s, 6);
printf("地址：");  scanf("%s%*c", s);
strncpy(stud.addr, s, 30);
if(fwrite(&stud,sizeof(STU_TYPE),1,fp)!=1){
   printf("文件输出出错！.\n");
   fclose(fp);   return 1;
}
```

文件处理示例程序-4(续)

```
    }
    fclose(fp);
    fp = fopen(fname, "rb");
    while(fread(&stud,sizeof(STU_TYPE),1,fp)==1){
        printf("%-16s%15s%12s%32s\n",
                stud.name, stud.phone, stud.zip, stud.addr);
    }
    fclose(fp);  return 2;
}
```

其他常用库函数简介-1

- 函数fprintf()和fscanf()
  - fprintf()和fscanf()的作用分别与函数printf()和scanf()相仿，都是格式输入和输出函数。只是函数fprintf()和fscanf()的输出输入的对象是一般的文件。因此，也多了一个文件指针形参。一般调用形式为

    fprintf(文件指针, 格式字符串, 输出项表)

    fscanf(文件指针, 格式字符串, 输入项地址表)
  - fprintf(wp, "i = %d r = %6.4f\n", i, r); 表示将整型变量i和实型变量r的值按格式输出到与wp相联系的文件上
  - fscanf(rp, "%d %f", &i, &r); 表示从与rp相联系的文件为变量i和r读入数据

其他常用库函数简介-2

- 函数fgets()和fputs()
  - 函数fgets()和fputs()分别用于正文文件输入和输出字符串
  - 说明形式分别为

    char *fgets(char *str, int n, FILE *fp)

    fputs(char *str, FILE *fp)
  - fgets()用于从文件读取字符序列，并存于字符指针所指出的存储区域中。当遇到换行符时，读字符过程结束
  - 函数fgets()与函数gets()都在读入的字符序列之后自动存储字符串结束标志'\0',并返回字符串的首字符指针。函数fgets()除增加整型参数和文件指针参数之外，还在读到换行符时，**存储换行符**，而函数gets()不存储换行符
  - fputs()将由str所指的字符串复制到fp所指文件，不在复制的字符序列之后另外再添加换行符

其他常用库函数简介-3

- 函数rewind()、fseek()和ftell()
  - 程序即刻能读写的是文件某一位置上的数据，这个位置称为文件的当前读写位置
  - 对于顺序读写，每读写一个字符后，当前位置就自动向后移一个字符位置
  - 对于随机读写的文件，要读写其它位置上的数据，就得改变文件当前读写位置
  - 实现读写位置移动可调用库函数 rewind()和fseek()
  - 另设有函数ftell()用于查询文件当前读写位置

其他常用库函数简介-3

- 函数rewind()与fseek()
  - 函数rewind()的说明形式为

    void rewind(FILE *fp)
    - 函数rewind()的作用是使文件当前位置重新回到文件之首。该函数没有返回值
  - 函数 fseek() 的说明形式为

    fseek(FILE *fp, long offset, int ptrname)
    - ptrname表示定位基准
    - 函数fseek()是文件随机存取的最主要函数，用它可以将文件的当前位置移到文件的任意位置上

其他常用库函数简介-3

- 函数fseek()
  - 函数fseek() 中ptrname只允许0，1，或2。其中0代表以文件首为基准，1以当前位置为基准，2以文件尾为基准；0、1和2分别被定义有名称SEEK_SET、SEEK_CUR和SEEK_END。long型形参offset是位移量，表示以ptrname为基准，移动的字节数
  - 调用函数 fseek() 的例子

```
fseek(fp, 40L, SEEK_SET);当前位置定于离文件头40个字节处
fseek(fp, 20L, SEEK_CUR);文件当前位置定于离当前位置20个字节处
fseek(fp, -30L, SEEK_END);将文件的当前位置定于文件尾后退30个字节处
```

  - 函数fseek()一般用于二进制文件的随机读写，因为二进制文件的数据表示形式与数据在内存中的表示形式相同，各种类型的数据表示都有确定的字节数
    - 正文文件的数据表示形式与数据在内存中的表示形式不同，在输入输出时，数据内外表示形式要进行转换。数值类型的数据在文件中的表示没有固定的字节数，计算位置时会发生混乱

其他常用库函数简介-3

- 函数ftell()
  - 函数ftell()用于得到文件当前位置相对于文件首的偏移字节数。在随机方式存取文件时，由于文件位置频繁的前后移动，程序不容易确定文件的当前位置。调用函数ftell()就能非常容易地确定文件的当前位置
  - 函数 ftell() 的说明形式为

    long ftell(FILE *fp)
  - 利用函数 ftell() 也能方便地知道一个文件的长：

    fseek(fp, 0L, SEEK_END);当前位置移到文件的末尾

    len = ftell(fp); 获得当前位置相对于文件首的位移

输入输出数据流重定向

- 程序常从标准输入文件读数据，向标准输出文件输出结果
- UNIX和DOS都支持数据流重定向
  - 在启动程序的命令行中，用改向输入标志 “<”，能把标准输入改成从某个文件读入
  - 改向输出标志“>”，能把标准输出改成向某个文件输出

提要

- C文件概述
- 文件类型和文件类型指针变量
- 文件打开和关闭库函数
- 文件处理程序结构和常用文件库函数
- 文件程序设计实例

文件程序设计实例-1

- 把几个文件串接在一起显示的程序
  - 设运行本程序的命令行呈以下形式

    cat 文件$_1$ 文件$_2$ … 文件$_n$
  - 并当命令行不带参数时，缺省文件为标准输入文件。遇参数文件不能打开情况，程序在标准出错输出文件上输出不能打开文件的通知信息

```
#include <stdio.h>
#include <stdlib.h>
int main( int argc, char *argv[])
{  FILE *fp;
   printf("Program start!\n");
   if (argc == 1)  filecopy(stdin);
```

文件程序设计实例-1(续)

```
  else
    while (--argc > 0)
      if ((fp = fopen(*++argv, "r")) == NULL)  {
       fprintf(stderr, "Can't open %s.\n", *argv);
       return 1;
      }else{ filecopy(fp);  fclose(fp); }
 return 0;
}
void filecopy(FILE *fp)
{  int c;
   while ((c = fgetc(fp)) !=  EOF) putchar(c);
}
```

文件程序设计实例-2

- 一个分栏输出程序：将正文分成页、每页又分若干栏的形式输出。为此程序用一个三维数组buf[][][]，控制从正文中读出的当前字符c填入buf[col][row][p]，表示将c填入col栏row行的第p列
  - 每填入一个字符，p应增1。填满了栏中一行,或遇到正文中的换行符时，row应增1。当填满一栏时，col应增1，准备填下一栏。当一页都填满时，程序就组织整页版面的逐行输出
  - 程序为了区别正文的实际分行和因栏的总列数太少，程序给予的强制分行，程序在强制分行的续行之前不输出正文的行号，对于正文原来的正文行，程序在输出它们时先输出它的行号。程序另假定正文不超过10000行

文件程序设计实例-2(续)

```
#include <stdio.h>
#define LL     80    /* 每行80列 */
#define COL    2 /* 每页2栏 */
#define INTV   9 /* */
#define CSIZE  (LL/COL - INTV) /* 每栏宽度 */
#define PL     20     /* 每页20行 */
#define MARGIN 3 /* 每页前后各空3行*/
char  buf[COL][PL][CSIZE];
int ln[COL][PL];   /* 记录对应正文行在正文中的行号 */
int col, row, p;
void newline();void printpage();
```

文件程序设计实例-2(续)

```
void fileprint(FILE *fp) /* 正文按页按栏输出 */
{  int lin, c;
   lin = col = row = p = 0;
   while ((c = fgetc(fp)) != EOF) {
     ln[col][row] = ++lin; /* 记录目前栏中行对应行号 */
     while (c != '\n' && c != EOF) {
       if (p >= CSIZE)  {
         newline();
         ln[col][row] = 0; /* 强制分行没有行号 */
       }
       buf[col][row][p++] = c;   c = fgetc(fp);
     }  /* end of while (c != '\n' && c != EOF) */
```

文件程序设计实例-2(续)

```
    newline();
    if (c == EOF) break;
    } /* while ((c = fgetc(fp)) != EOF)  */
    while (col != 0 || row != 0 || p != 0) { /* 用空行填满当前页 */
       ln[col][row] = 0;    newline();
    }
   fclose(fp);
}
```

文件程序设计实例-2(续)

```
void newline()
{  while (p < CSIZE)  /* 当前行剩余部分填空白符 */
       buf[col][row][p++] = ' ';
   p = 0;
   if (++row >= PL) {  /* 换行,一栏满 */
       row = 0;
       if (++col >= COL)  {  /* 换栏,一页满 */
           col = 0;
           printpage(); /* 一页满输出 */
       }
   }
}
```

文件程序设计实例-2(续)

```
void printpage()
{  int k, p, lpos, col, d;    char aline[LL];
   aline[LL-1] = '\0';
   for(k = 0; k < MARGIN; k++)  putchar('\n'); /*页前空行*/
   for(k = 0; k < PL; k++) { /* 输出一页内的PL行 */
      for(p = 0; p < LL-1; p++) aline[p] = ' '; /* 用' ' 初始化行 */
      for(lpos = 0, col = 0; col < COL; lpos += CSIZE+INTV, col++){
         d = ln[col][k];  p = lpos + 4; /* 行号最多4位数 */
         while(d > 0){ aline[p--] = d%10 + '0'; /* 行号 */
               d /= 10;
          } /* 行号求解并赋值 */
         for(p = 0; p < CSIZE; p++)
         aline[lpos + 7 + p] = buf[col][k][p]; /* 行号和正文之间有空*/
       } /* 在一行内，按栏输出*/
```

文件程序设计实例-2(续)

```
    printf("%s\n", aline);
   }
  for(k = 0; k < MARGIN; k++)  putchar('\n'); /*页后空行*/
}
void main()
{  char fname[40];   FILE *fp;
   printf("Enter file name.\n");
   scanf("%s*c",fname);
   if ((fp = fopen(fname, "r")) == NULL) {
      printf("Can't open file %s!\n", fname);
      return;
   }
   fileprint(fp);
}
```

文件程序设计实例-3

- 一个通信录管理程序有以下功能：
  - 插入新的通信记录；
  - 查找某人的通信记录；
  - 删除某人的通信记录；
  - 浏览通信录；
  - 结束程序运行。
- 设每条通信录包含以下内容：
  - 姓名
  - 地址
  - 邮政编码
  - 电话号码
- 并设通信录全部以字符行形式存于文件中，每四个字符行构成一个通信录记录

文件程序设计实例-3(续)

- 程序启动后，自动从指定的文件中读取通信录信息。程序运行结束后，又自动将内存中修改过的通信录信息保护到文件中。为了查找，插入，删除等操作的方便，以双向链表组织通信录信息
- 设启动程序的命令行可带通信录文件参数。如启动时未给出文件名参数，则程序首先要求用户键入通信录文件名
- 程序运行时，反复请求输入操作命令的提示信息

  请输入命令：[i，f，d，s，q]

  - 即要求用户打入一条命令，它可以是 i(插入)、f(寻找)、d(删除)、s(显示)以及q(结束程序运行)。如果打入命令不是其中之一， 将详细显示命令符及其意义的说明

文件程序设计实例-3(续)

- 命令表:
  - i：插入一条新的通信记录.
  - f：按输入名查找通信录.
  - d：按输入名删除一条通信录.
  - s：浏览通信录.
  - q：退出.
- 程序将全部通信录组织成一个双向勾链的链表。接受显示通信录表的命令后，首先显示的是第一条通信录，由用户键入u键或d键分别实现向上或向下选择，显示下一条通信录。键入Esc键结束显示命令。上述三键在显示一条通信录之后，在下面给出提示。重新接受用户的选择

文件程序设计实例-3(续)

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAXLEN 120
typedef struct saddr {
   char  *name;
   char  *address;
   char  *zip;
   char  *phone;
   struct saddr *next,  *pre;
} ADDR;
void insert( ADDR *ptr); /* 输入新的通信录 */
void make(ADDR *ptr, ADDR *p); /* 将记录*p插入链表 */
```

文件程序设计实例-3(续)

```
void find( ADDR *ptr ); /* 按输入名查找通信录，并显示 */
void del( ADDR *ptr );  /* 按输入名删除通信录 */
void display( ADDR *ptr ); /* 显示一个通信录 */
void usage(void); /* 告知命令用法 */
void show( ADDR *ptr ); /* 浏览通信录 */
void save(ADDR *head, char *fname); /* 保存通信录到文件 */
ADDR *load(char *fname); /* 读出通信录，建立双向链表 */
int getstr(FILE *rfp, char **s);
int getbuffer(FILE *rfp);
void freeall(ADDR *head);
char buffer[MAXLEN], fname[40];    ADDR *load();
FILE *fp;      int modified = 0;
```

文件程序设计实例-3(续)

```
void main(int argc, char **argv)
{  ADDR *head;  char c;
   if (argc == 1) {  printf("请输入通信录文件名. ");
         scanf("%s%*c", fname);
    }
   else  strcpy(fname, *++argv);
   head = load(fname);
   while (1)  {
       printf("\n请输入命令: [i, f, d, s, q]\n");
       while (getbuffer(stdin) == 0);
       if ((c = buffer[0]) == 'q') {
          if (modified) {
             printf("修改后的通信录未保存，要保存吗?(y/n) ");
```

文件程序设计实例-3(续)

```
            while (!(((c=getchar())>='a'&&c<='z')||(c>='A'&&c<='Z')));
            if (c == 'y' || c == 'Y') save(head, fname);
          }
          freeall(head);    break;
        }
        switch (c) {
          case 'i' : insert(head);  break;
          case 'f' : find(head);    break;
          case 'd' : del(head);  break;
          case 's' : show(head);    break;
          default  : usage();       break;
        }
      }
}
```

文件程序设计实例-3(续)

```
void insert( ADDR *ptr)  /* 输入新的通信录 */
{ char *spt;  ADDR *p;
  while (1) {  printf("名字? (立即回车表示结束) ");
    if (getstr(stdin, &spt) ==0)   break;
    p   = (ADDR *)malloc(sizeof(ADDR));  p->name = spt;
    printf("地址? ");      getstr(stdin, &p->address);
    printf("邮编? ");      getstr(stdin, &p->zip);
    printf("电话号码? ");  getstr(stdin, &p->phone);
    p->next = NULL;        p->pre  = NULL;
    make(ptr, p);    modified = 1;
  }
}
```

文件程序设计实例-3(续)

```
void make(ADDR *ptr, ADDR *p)  /* 将*p插入链表 */
{ ADDR *pre, *suc;
  pre = ptr; suc = pre->next;
          while (suc) {
    if (strcmp(suc->name,p->name) >= 0)   break;
    pre = suc;       suc = pre->next;
  }
  pre->next = p;   p->pre    = pre;
  p->next   = suc; if(suc) suc->pre  = p;
}
```

文件程序设计实例-3(续)

```
void find( ADDR *ptr ) /* 按输入名查找通信录，并显示 */
{ ADDR *p;
  printf("输入寻找的名: ");
  while (getbuffer(stdin) == 0);
  p = ptr->next;
  while (p) {
     if (strcmp(buffer, p->name) == 0)  break;
     p = p->next;
  }
  if (p)  display(p);   else    printf("未找到!\n");
}
```

文件程序设计实例-3(续)

```
void del( ADDR *ptr )  /* 删除指定的通信录 */
{ ADDR *pre, *suc;
  printf("输入要删除的通信录的名: ");
  while (getbuffer(stdin) == 0) ;  /* 跳过空行 */
  pre = ptr;
  suc = pre->next;
  while (suc) {
    if (strcmp(suc->name, buffer) == 0)
       break;
    pre = suc;
    suc = pre->next;
  }
```

文件程序设计实例-3(续)

```
if (suc) {
    if (suc->next)
        suc->next->pre = pre;
    pre->next = suc->next;
    if(suc->name) free(suc->name);
    if(suc->address) free(suc->address);
    if(suc->phone) free(suc->phone);
    if(suc->zip) free(suc->zip);
    free(suc);
    modified = 1;
 } else  printf("Not found !\n");
}
```

文件程序设计实例-3(续)

```
void display( ADDR *ptr )  /* 显示一个通信录 */
{  printf("\n\t名：%s\n", ptr->name);
   printf("\t地址：%s\n", ptr->address);
   printf("\t邮编：%s\n", ptr->zip);
   printf("\t电话：%s\n", ptr->phone);
}
void usage() /* 告知命令用法 */
{ printf("\n命令表:\n");
  printf("i：插入一条新的通信录.\n");
  printf("f：按输入名寻找通信录.\n");
  printf("d：按输入名删除一条通信录.\n");
  printf("s：浏览通信录.\n");
  printf("q：退出.\n");
}
```

文件程序设计实例-3(续)

```
#define  UP    'u'
#define  DOWN  'd'
#define  Esc   'e'
void show( ADDR *ptr )  /* 浏览通信录 */
{  ADDR *p; int flg, c;
   p = ptr->next;
   if (p == NULL) return;
   while (1) { display(p);  printf("\n\t\t");
       if (p != ptr->next)   printf("up    ");
       if (p->next != NULL)  printf("down  ");
       printf("esc\n\n");
       flg = 1;
```

文件程序设计实例-3(续)

```
        do { c = getchar();
           switch (c)  {
             case UP   : if (p != ptr->next) { p = p->pre;  flg = 0;
                  }
                 break;
             case DOWN : if (p->next) { p = p->next;  flg = 0;
                  }
                  break;
         case Esc  : return;
         }
      } while (flg);
    }
}
```

文件程序设计实例-3(续)

```
void save(ADDR *head, char *fname) /* 保存通信录到文件 */
{ FILE *fp;  ADDR *p;
  p = head->next;
  if ((fp = fopen(fname, "w")) == NULL) {
        fprintf(stderr, "Can't open %s.\n", fname);     return;
  }
 while (p != NULL)  {
  fprintf(fp,"%s\n%s\n%s\n%s\n",p->name,p->address,p->zip,p->phone);
   p = p->next;
 }
 fclose(fp);
}
```

文件程序设计实例-3(续)

```
ADDR *load(char *fname)  /* 读出通信录，建立双向链表 */
{ ADDR *p, *h;
  h = (ADDR *)malloc(sizeof(ADDR));   h->pre  = NULL;
  h->next = NULL;
  if ((fp = fopen(fname, "r")) == NULL)  return h;
  while (!feof(fp)){ p = (ADDR *)malloc(sizeof(ADDR));
     if(getstr(fp, &p->name) == 0) { free(p);  break;
      }
     getstr(fp, &p->address); getstr(fp, &p->zip);
     getstr(fp, &p->phone);    make(h, p);
   }
  fclose(fp);     return h;
}
```

文件程序设计实例-3(续)

```
/* 从文件读入一行，并删除前导和尾随空白符后保存于堆空间 */
int getstr(FILE *rfp, char **s)
{ int len;
  if ((len = getbuffer(rfp)) == 0) {
                    *s = NULL;  return 0;
          }
  *s = (char *)malloc(len+1);   strcpy(*s, buffer);
  return len;
}
```

文件程序设计实例-3(续)

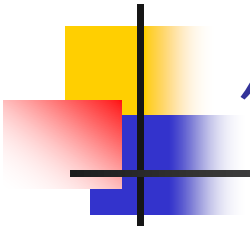
```
/* 读入一行，并删除前导和尾随空白符后保存于buffer */
int getbuffer(FILE *rfp)
{ int len;  char *hp, *tp;
  if (fgets(buffer, MAXLEN, rfp) == NULL)     return 0;
  hp = buffer;
  while(*hp == ' ' || *hp == '\t') hp++;
  tp = hp + strlen(hp) - 1;
  while(tp>=hp&&(*tp==' '||*tp=='\t'||*tp=='\n'))    tp--;
  *(tp+1) = '\0';
  for(tp = buffer; *tp++ = *hp++;);
  len = strlen(buffer);
  return len;
}
```

文件程序设计实例-3(续)

```
void freeall(ADDR *head)  /* 释放内存空间 */
 { ADDR *p;
   p = head->next;   free(head);
   while (p != NULL) {
         if (p->name != NULL)  free(p->name);
         if (p->address != NULL)  free(p->address);
         if (p->phone != NULL)  free(p->phone);
         if (p->zip != NULL)  free(p->zip);
         head = p->next;
         free(p);
         p = head;
   }
}
```

本讲(第8章) 小结

- C文件概述
- 文件类型和文件类型指针变量
- 文件打开和关闭库函数
- 文件处理程序结构和常用文件库函数
- 文件程序设计实例



作业

- 习题8：第5、8、10题