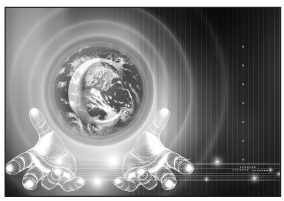


第 3 章
文　件　类

文件可分为文本文件和二进制文件两种。文本文件可用来保存字符，文件中的字节是字符的编码，源文件是文本文件。二进制文件可用来保存可执行文件，文件中的字节是机器指令或数据编码。

C 语言程序有 3 个预定义的文本流在程序执行时默认打开：标准输入（stdin）、标准输出（stdout）和标准错误（stderr）。

文件的读写属于 I/O 操作。FILE 类型保存和文件有关的信息，文件的打开操作在内存中分配一个 FILE 类型并将其指针返回。大多数 I/O 函数都接受一个指向 FILE 类型的指针作为参数，标准 I/O 库函数在头文件 stdio.h 中声明。常用的 I/O 函数有：

- FILE *fopen(const char *filename, const char *mode) 函数：用来打开文件流，接受文件名 filename 和打开模式 mode 两个字符串参数，返回一个 FILE * 类型的指针。若打开文件时遇到错误，则 fopen 函数返回 null 指针。fopen 常用打开模式如表 3-1 所示。

表 3-1　fopen 常用打开模式

*mode	含　义	注　释
r	只读	打开一个文件，仅允许从文件读取数据
w	只写	创建一个新文件或截断一个现有文件，仅允许向文件写入数据
a	追加	创建一个新文件或打开一个文件，仅允许从文件尾部追加数据
r+	读写	打开一个文件，从文件起始位置进行读 / 写
w+	读写	创建一个新文件或截断一个现有文件，对文件进行读 / 写
a+	读写	创建一个新文件或打开一个文件，对文件进行读 / 追加

- int fclose(FILE *stream) 函数：用来关闭一个打开的流 stream 并清除其所有内部数据缓冲区。如关闭时遇到错误，则返回 EOF，否则返回 0。fopen 和 fclose 函数通常成对使用。
- int fgetc(FILE *stream) 函数：从流 stream 中读取下一个字符，并把它作为一个 int 类型返回。若发生错误或已经位于文件尾，函数返回 EOF。
- int *fputc(int c, FILE *stream) 函数：把字符 c 写入流 stream 的当前位置，并将写入值按 int 类型返回。若发生错误，函数返回 EOF。

- int ungetc(int c, FILE *stream) 函数：将字符 c 压回输入流 stream，当下次读取流 stream 时会读取先前被压入的字符。若多次调用 ungetc 函数压入字符，当下次读取流 stream 时先前压入的字符按与压入相反的顺序读取。若函数调用成功，则返回被压入的字符 c；若函数调用失败，则返回 EOF。
- char *fgets(char *s, int n, FILE *stream) 函数：三个参数为指向字符数组起始位置的指针 s、计数值 n 和输入流 stream。函数从输入流 stream 读取字符到数组，直到遇到换行符、文件尾或未遇到换行符和文件尾但读取了 n-1 个字符。读取完毕后，数组中已读取的字符后会自动添加一个结束符‘\0’。若因读取到换行符而结束，换行符也存储在数组中，且在自动添加的结束符‘\0’之前。函数的返回值为指向字符数组起始位置的指针 s，当读取任何字符之前就遇到文件结束符或错误时，返回 null 指针。
- int fscanf(FILE *stream, const char *format, ...) 函数：对格式化的输入文本进行解析，从第一个输入流参数 stream 中读取字符，根据格式控制字符串参数对字符序列进行转换后存储到指针所指定的对象中。
- int fprintf(FILE *stream, const char *format, ...) 函数：根据格式控制字符串参数转换字符，将其输出给第一个参数 stream 所指定的流。
- size_t fread(void *ptr, size_t size, size_t count, FILE *stream) 函数：可用来读取二进制文件，从流 stream 读取最多 count 个大小为 size 的元素到指针 ptr 指向的数组。fread 函数最多读取 count 个元素，并返回其实际读取的元素个数，故当读到文件尾时，返回值可能小于 count。若遇到错误，则返回 0。
- size_t fwrite(const void *ptr, size_t size, size_t count, FILE *stream) 函数：可用来写入二进制文件，将指针 ptr 指向的数组中最多 count 个大小为 size 的元素写入流 stream。fwrite 函数最多写入 count 个元素，并返回实际写入的元素个数。除非出现错误，否则函数的返回值与 count 相同。
- int remove(const char *filename) 函数：删除 filename 文件。
- int rename(const char *oldname, const char *newname) 函数：把 oldname 的文件名修改为 newname。

3.1 文件复制

【例 3.1】编写文本文件复制程序，命令行格式为：mycopy filename1 filename2。

题目分析

每个 C 语言程序必须定义一个 main 函数作为程序的入口。程序启动从 main 函数开始执行，main 函数返回时程序终止，若 main 函数末尾没有返回语句，则默认执行 return 0 语句。

标准 C 语言的 main 函数有两种形式：接受 0 个参数的 int main(void)，即 int main()；接受 2 个参数的 int main(int argc, char *argv[])。对于后者，形参 argc 表示传递给程序的

参数个数；形参 argv 是指针数组，其每个指针分别指向传递给程序的字符串参数，第一个字符串 argv[0] 是程序的名称。

文件的打开和关闭可调用 fopen 和 fclose 函数。文件的读取和写入可通过三种方式实现：1）fgetc 和 fputc 函数配合，每次读写一个字符，见参考程序 mycopyV1.c；2）fscanf 和 fprintf 函数配合，每次读写一个字符，见参考程序 mycopyV2.c；3）fread 和 fwrite 函数配合，每次读写 BUFSIZE 个字符，见参考程序 mycopyV3.c。方式 3）通过增设缓冲区 buffer[BUFSIZE]，减少 I/O 操作的次数，提高程序运行速度，可用于读写量较大的程序。

检测到异常时，用 exit 标准库函数退出。void exit(int status) 函数可正常终止一个程序并执行清理操作，形参 status 值为 0 表示程序成功退出，非零值可表示各类异常终止。从 main 函数通过 return 语句返回一个整数值相当于用这个整数值调用 exit 函数。exit 函数在头文件 stdlib.h 中声明。

参考程序

```
/*
 * 文件名：mycopyV1.c
 * 描述：编写文件复制命令行程序，命令行格式为：mycopy filename1 filename2
 * 作者：刘博
 */

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

int main(int argc, char *argv[])
{
    FILE *fpSrc, *fpDes;
    char ch;

    /* 检查命令行参数个数 */
    if (argc != 3) {
        printf("Usage: mycopy SourceFile DestinationFile\n");
        exit(1);
    }

    /* 打开文件 */
    if ((fpSrc = fopen(argv[1], "r")) == NULL) {
        printf("open file error!\n");
        exit(1);
    }

    if ((fpDes = fopen(argv[2], "w")) == NULL) {
        printf("creat file error!\n");
        exit(1);
    }

    /* 复制文件 */
    while ((ch = fgetc(fpSrc)) != EOF) {
```

```
            fputc(ch, fpDes);
        }

        /* 关闭文件 */
        fclose(fpSrc);
        fclose(fpDes);
    }
```

```
/*
 * 文件名：mycopyV2.c
 * 描述：编写文件复制命令行程序，命令行格式为：mycopy filename1 filename2
 * 作者：刘博
 */

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

int main(int argc, char *argv[])
{
    FILE *fpSrc, *fpDes;
    char ch;

    /* 检查命令行参数个数 */
    if (argc != 3) {
        printf("Usage: mycopy SourceFile DestinationFile\n");
        exit(1);
    }

    /* 打开文件 */
    if ((fpSrc = fopen(argv[1], "r")) == NULL) {
        printf("open file error!\n");
        exit(1);
    }

    if ((fpDes = fopen(argv[2], "w")) == NULL) {
        printf("creat file error!\n");
        exit(1);
    }

    /* 复制文件 */
    while (fscanf(fpSrc, "%c", &ch) != EOF) {
        fprintf(fpDes, "%c", ch);
    }

    /* 关闭文件 */
    fclose(fpSrc);
    fclose(fpDes);
}
```

```
/*
 * 文件名：mycopyV3.c
```

```
 * 描述：编写文件复制命令行程序，命令行格式为：mycopy filename1 filename2
 * 作者：刘博
 */

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

#define BUFSIZE 1024

int main(int argc, char *argv[])
{
    FILE *fpSrc, *fpDes;
    char buffer[BUFSIZE];
    int in, out;

    /* 检查命令行参数个数 */
    if (argc != 3) {
        printf("Usage: mycopy SourceFile DestinationFile\n");
        exit(1);
    }

    /* 打开文件 */
    if ((fpSrc = fopen(argv[1], "r")) == NULL) {
        printf("open file error!\n");
        exit(1);
    }

    if ((fpDes = fopen(argv[2], "w")) == NULL) {
        printf("creat file error!\n");
        exit(1);
    }

    /* 复制文件 */
    while ((in = fread(buffer, 1, BUFSIZE, fpSrc)) > 0) {
        out = fwrite(buffer, 1, in, fpDes);
        /* 判断读出和写入元素个数是否一致 */
         if (in != out) {
            printf("copy error!\n");
            exit(1);
        }
    }

    /* 关闭文件 */
    fclose(fpSrc);
    fclose(fpDes);
}
```

3.2 文件比较

【例 3.2】打印出两个文本文件第一个不相同的行（每行字符数不多于 80），命令行格式为：filecmp filename1 filename2。

题目分析

文件的按行读取可调用 fgets 标准库函数。比较字符串可调用 strcmp 或 strncmp 标准库函数。

自定义函数 filecomp 用来比较不同行。当找到第一个不同行时 filecomp 函数返回到 main() 函数。lp1 和 lp2 的作用为：当 filename1 和 filename2 出现第一个不相同行时，控制循环结束；当 filename1 或 filename2 未出现不相同行但已读到文件结束符或发生读取错误时，控制循环结束。

参考程序

```
/*
 * 文件名：filecmp.c
 * 描述：打印出两个文本文件第一个不相同的行。
 *       命令行格式为：filecmp filename1 filename2
 * 作者：刘博
 */

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

#define MAXLINE 80 /* 每行字符数不多于80 */

void filecomp(FILE *fp1, FILE *fp2);

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

    /* 检查命令行参数个数 */
    if (argc != 3) {
        printf("Usage: filecmp File1 File2\n");
        exit(1);   /* 异常退出 */
    }

    /* 打开文件 */
    if ((fp1 = fopen(argv[1], "r")) == NULL) {
        printf("can't open %s\n", argv[1]);
        exit(1);   /* 异常退出 */
    }
    if ((fp2 = fopen(argv[2], "r")) == NULL) {
        printf("can't open %s\n", argv[2]);
        exit(1);   /* 异常退出 */
    }

    filecomp(fp1, fp2);

    /* 关闭文件 */
```

```
    fclose(fp1);
    fclose(fp2);
}

/* 比较两个文件第一个不同行 */
void filecomp(FILE *fp1, FILE *fp2)
{
    char line1[MAXLINE], line2[MAXLINE];
    char *lp1, *lp2;

    do {
        lp1 = fgets(line1, sizeof(line1), fp1);
        lp2 = fgets(line2, sizeof(line2), fp2);

        if (lp1 == line1 && lp2 == line2) {
            if (strncmp(line1, line2, MAXLINE) != 0) {
                printf("first difference line:\nfile1: %sfile2: %s",
                        line1, line2);
                lp1 = lp2 = NULL;  /* 找到不同行，循环结束 */
            }
        } else if (lp1 != line1 && lp2 == line2) {
            printf("end of file1.");
        } else if (lp1 == line1 && lp2 != line2) {
            printf("end of second file2.");
        }
    } while (lp1 == line1 && lp2 == line2);
}
```

3.3 删除 C 程序注释

【例 3.3】删除合法 C 程序的注释部分，命令行格式为：rcomment inputFile outputFile。

题目分析

程序遇到注释的开始“/*”时，调用 RemoveComment 函数跳过注释部分，直到遇到匹配的注释结束“*/”。

需要注意，当遇到多个连续的斜杠‘/’但无星号‘*’匹配时，应保留斜杠‘/’。当遇到前单引号或前双引号时，引号内的“/*”和“*/”不应当做注释处理。主程序调用 EchoQuote 函数，保留引号内的所有字符，直到遇到匹配的后前单引号或后双引号。还应注意，不要将引号内转义符‘\’后的引号匹配成后引号。

参考程序

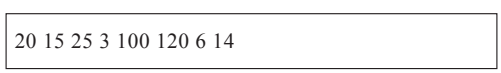
```
/*
 * 文件名：rcomment.c
 * 描述：删除合法C程序的注释部分。
 *       命令行格式为：rcomment inputFile outputFile
 * 作者：刘博
 */
```

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

void RemoveComment(FILE *fp);
void EchoQuote(int c, FILE *fpIn, FILE *fpOut);

int main(int argc, char *argv[])
{
    FILE *fpIn, *fpOut;
    int c, n;

    /* 检查命令行参数个数 */
    if (argc != 3) {
        printf("Usage: rcomment InputFile OutputFile\n");
        exit(1);
    }

    /* 打开文件 */
    if ((fpIn = fopen(argv[1], "r")) == NULL) {
        printf("rcomment: can't open %s\n", argv[1]);
        exit(1);
    }
    if ((fpOut = fopen(argv[2], "w")) == NULL) {
        printf("rcomment: can't open %s\n", argv[2]);
        exit(1);
    }

    while ((c = fgetc(fpIn)) != EOF) {
        if (c == '/') {
            if ((n = fgetc(fpIn)) == '*') {      /* 注释开始 */
                RemoveComment(fpIn);
            } else if (n == '/'){                /* 处理连续多个斜杠 */
                fputc(c, fpOut);
                ungetc(n, fpIn);
            } else {                             /* 非注释部分 */
                fputc(c, fpOut);
                fputc(n, fpOut);
            }
        } else if (c == '\'' || c == '\"') {     /* 单引号和双引号内处理 */
            EchoQuote(c, fpIn, fpOut);
        } else {
            fputc(c, fpOut);                     /* 非注释部分 */
        }
    }

    /* 关闭文件 */
    fclose(fpIn);
    fclose(fpOut);
}

void RemoveComment(FILE *fp)
```

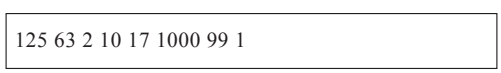
```
{
    int c, n;

    c = fgetc(fp);
    n = fgetc(fp);

    while (c != '*' || n != '/') {              /* 寻找注释结尾 */
        c = n;
        n = fgetc(fp);
    }
}

void EchoQuote(int c, FILE *fpIn, FILE *fpOut)
{
    int n;

    fputc(c, fpOut);
    while ((n = fgetc(fpIn)) != c) {            /* 寻找后引号 */
        fputc(n, fpOut);
        if (n == '\\') {  /* 引号内转义符后的引号非后引号 */
            fputc(fgetc(fpIn), fpOut);
        }
    }
    fputc(n, fpOut);
}
```

习题

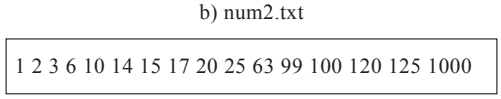
1. 比较两个文本文件并打印出它们第一个不相同的行（文件每行字符数不多于 80）。
2. 文本文件 num1.txt 和 num2.txt 中各有一组用空格分隔的整数，将 num1.txt 和 num2.txt 联合排序，并将结果保存在 num3.txt 中，如图 3-1 所示。



a) num1.txt



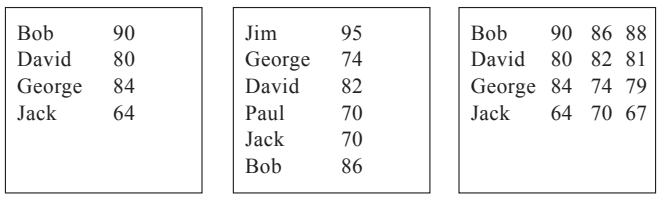
b) num2.txt



c) num3.txt

图 3-1 文件 num1.txt、num2.txt 和 num3.txt 举例

3. 现有两个文本文件 db1.txt 和 db2.txt。db1.txt 中第一列为姓名，第二列为英语成绩；db2.txt 中第一列为姓名，第二列为数学成绩。通过姓名字段将 db1.txt 文件关联到 db2.txt 文件生成 db3.txt 文件。db3.txt 文件第一列为姓名，第二列为英语成绩，第三列为数学成绩，第四列为平均成绩，如图 3-2 所示。



a) db1.txt　　b) db2.txt　　c) db3.txt

图 3-2　db1.txt、db2.txt 和 db3.txt 文件内容

4. 检查 C 源程序的圆括号和大括号是否匹配。正确的例子如（{((…)(…))}()），不正确的例子如 {(}。