# 8 文件操作实验

## 8.1 实验目的

1. 熟悉文本文件和二进制文件在磁盘中的存储方式；
2. 熟练掌握流式文件的读写方法。

## 8.2 实验内容

**8.2.1 文件类型的程序验证题**

设有程序：

1. #include <stdio.h>
2. int main(void)
3. {
4. short a=0x253f,b=0x7b7d;
5. char ch;
6. FILE \*fp1,\*fp2;
7. fp1=fopen("d:\\abc1.bin","wb+");
8. fp2=fopen("d:\\abc2.txt","w+");
9. fwrite(&a,sizeof(short),1,fp1);
10. fwrite(&b,sizeof(short),1,fp1);
11. fprintf(fp2,"%hx %hx",a,b);
12. rewind(fp1); rewind(fp2);
13. while((ch = fgetc(fp1)) != EOF)
14. putchar(ch);
15. putchar('\n');
17. while((ch = fgetc(fp2)) != EOF)
18. putchar(ch);
19. putchar('\n');
20. fclose(fp1);
21. fclose(fp2);
22. return 0;
23. }
24. 请思考程序的输出结果，然后通过上机运行来加以验证。
25. 将两处sizeof(short)均改为sizeof(char)结果有什么不同，为什么？
26. 将fprintf(fp2,"%hx %hx",a,b) 改为 fprintf(fp2,"%d %d",a,b)结果有什么不同。

**解答：**

1. 猜测结果为：

?%}{

253f 7b7d

上机运行得到的结果如图8-1所示：

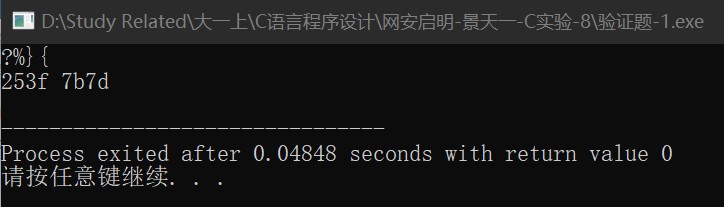


图8-1 验证题1运行结果截图

与猜测的结果一致；

1. sizeof(char)的结果是1，sizeof(short)的结果是2，因此，fwrite函数从文件中每次读取的字节数不同；sizeof(char)运行的结果如图8-2所示：

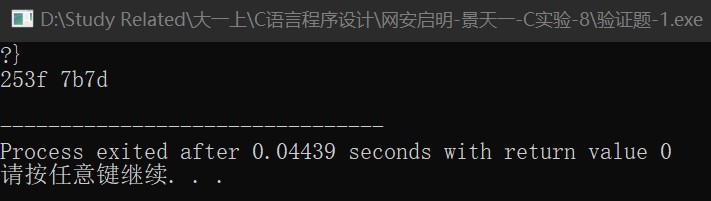


图8-2 验证题1修改后运行结果截图

1. 将fprintf(fp2,"%hx %hx",a,b) 改为 fprintf(fp2,"%d %d",a,b)后，运行结果如图8-3所示：

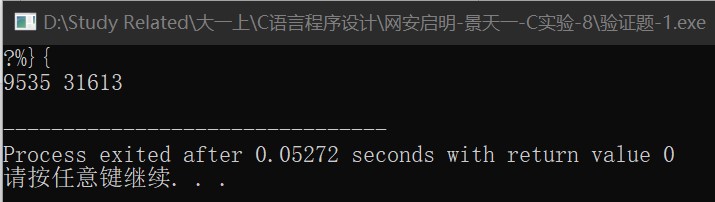


图8-3 验证题1修改后运行结果截图

可以观察到，更改后以十进制整数进行了输出。

**8.2.2 源程序修改替换**

将指定的文本文件内容在屏幕上显示出来，命令行的格式为：

type filename

1. 源程序中存在什么样的逻辑错误（先观察执行结果）？对程序进行修改、调试，使之能够正确完成指定任务。
2. #include<stdio.h>
3. #include<stdlib.h>
4. int main(int argc, char\* argv[])
5. {
6. char ch;
7. FILE \*fp;
8. if(argc!=2){
9. printf("Arguments error!\n");
10. exit(-1);
11. }
12. if((fp=fopen(argv[1],"r"))==NULL){ /\* fp 指向 filename \*/
13. printf("Can't open %s file!\n",argv[1]);
14. exit(-1);
15. }
16. while(ch=fgetc(fp)!=EOF) /\* 从filename中读字符 \*/
17. putchar(ch); /\* 向显示器中写字符 \*/
18. fclose(fp); /\* 关闭filename \*/
19. return 0;
20. }
21. 用输入输出重定向freopen改写main函数。

**解答：**

1. 程序改错：

第15行while(ch=fgetc(fp)!=EOF)中，！=的优先级高于=，因此需要在靠前的表达式外补充括号，正确行使如下：

while((ch=fgetc(fp))!=EOF)

1. 源代码清单：
2. #include<stdio.h>
3. #include<stdlib.h>
4. int main(int argc, char\* argv[]) {
5. char ch;
6. FILE \*fp;
7. if(argc!=2) {
8. printf("Arguments error!\n");
9. exit(-1);
10. }
11. if((fp=fopen(argv[1],"r"))==NULL) { /\* fp 指向 filename \*/
12. printf("Can't open %s file!\n",argv[1]);
13. exit(-1);
14. }
15. freopen(argv[1], "r", fp);
16. while((ch=fgetc(fp))!=EOF) /\* 从filename中读字符 \*/ // 优先级错误
17. putchar(ch); /\* 向显示器中写字符 \*/
18. fclose(fp); /\* 关闭filename \*/
19. return 0;
20. }
21. 测试：

改错题1的运行结果如图8-4所示：

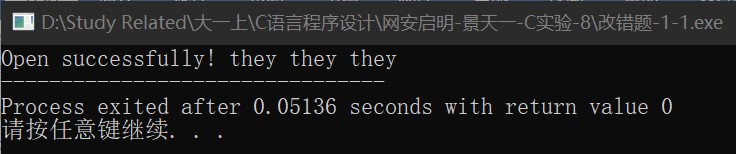


图8-4 改错题1修改后运行结果截图

1. 用freopen函数重写main函数：freopen函数的原型是：

FILE \*freopen(const char \* restrict filename, const char \* restrict mode, FILE \* restrict stream);

其作用是，使文件指针stream以mode重定向至打开的文件filename处；

源程序清单如下：

1. #include<stdio.h>
2. #include<stdlib.h>
3. int main(int argc, char\* argv[]) {
4. char ch;
5. FILE \*fp;
6. if(argc!=2) {
7. printf("Arguments error!\n");
8. exit(-1);
9. }
10. if((fp=fopen(argv[1],"r"))==NULL) { /\* fp 指向 filename \*/
11. printf("Can't open %s file!\n",argv[1]);
12. exit(-1);
13. }
14. freopen(argv[1], "r", fp);
15. while((ch=fgetc(fp))!=EOF) /\* 从filename中读字符 \*/
16. putchar(ch); /\* 向显示器中写字符 \*/
17. fclose(fp); /\* 关闭filename \*/
18. return 0;
19. }
20. 测试：

用freopen函数改写main函数后运行结果如图8-5所示：

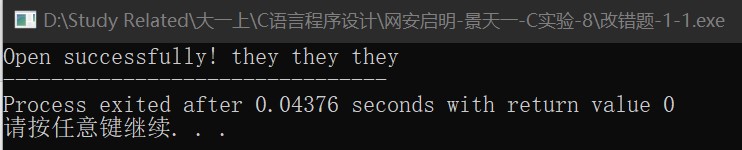


图8-5 改错题1修改后运行结果截图

**8.2.3 程序设计**

1. 编写一个程序replace，采用命令行方式，用给定的字符串替换指定文件中的目标字符串，并显示输出替换的个数。例如，命令行：

replace filename.txt you they

**解答：**

1） 算法流程如图8-6所示。

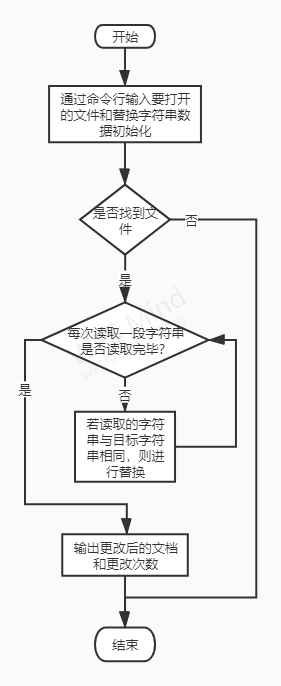


图8-6 程序设计题1的程序流程图

1. 源程序清单：
2. #include <stdio.h>
3. #include <string.h>
4. #include <stdlib.h>
5. int main(int argc, char \*argv[]) {
6. if (argc != 4) {
7. printf("Undefined input");
8. exit(0);
9. }
10. FILE \*open\_file, \*temp = fopen("temp.txt", "w+");
11. int count = 0;
12. char temp\_char[50];
13. char sep\_char;
14. if ((open\_file = fopen(argv[1], "r+")) == NULL) {
15. printf("Can't open file:%s !\n", argv[1]);
16. exit(0);
17. }
18. while (fscanf(open\_file, "%s", temp\_char) != EOF) {
19. sep\_char = fgetc(open\_file);
20. if (!strcmp(temp\_char, argv[2])) {
21. fprintf(temp, "%s%c", argv[3], sep\_char);
22. count++;
23. } else {
24. fprintf(temp, "%s%c", temp\_char, sep\_char);
25. }
26. }
27. rewind(open\_file);
28. rewind(temp);
29. while((sep\_char = fgetc(temp)) != EOF) {
30. fputc(sep\_char, open\_file);
31. }
32. printf("Replaced %d words in total", count);
33. remove("temp.txt");
34. return 0;
35. 测试

向命令行输入的参数如图8-7所示：

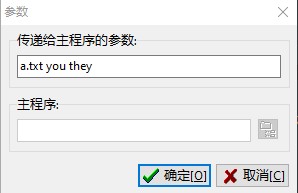


图8-6 程序设计题1向命令行传递的参数

a.txt文件中的内容为：Open successfully! they you you

程序设计题运行的结果如图8-8所示：

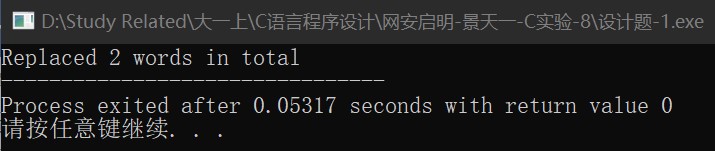


图8-8 程序设计题1的运行结果

a.txt文件中的内容为：Open successfully! they they they，与预期结果相符，证明了程序的正确性。

1. 从键盘输入10个单精度浮点数，以二进制形式存入文件float.dat中。再从文件中读出这10个单精度浮点数显示在屏幕上。之后要求将float.dat中的单精度浮点数按字节读出来，观察写入文件的浮点数字节数据是不是和计算机内存中表示的浮点数字节数据一致。

**解答：**

1. 解题思路如下所示：
2. 以wb+格式打开float.dat文件；
3. 输入10个float型数据，以二进制形式存入文件float.dat中
4. 利用fread函数将这10个数据打印出来；
5. 利用fread函数按sizeof(char)（一个字节）为单位读取float.dat中的数据；
6. 关闭文件，结束；
7. 源程序清单：
8. #include <stdio.h>
9. int main(void)
10. {
11. FILE \*open\_file = fopen("float.dat", "wb+");
12. float temp = 0;
13. for (int i = 0; i < 10; i++)
14. { printf("Input: Num%d:", i + 1);
15. scanf("%1f",&temp);
16. fwrite(&temp, sizeof(float), 1, open\_file);
17. }
18. rewind(open\_file);
19. char ch;
20. freopen("float.dat", "r", open\_file);
21. while(!feof(open\_file))
22. {
23. fread(&ch, sizeof(char), 1, open\_file);
24. printf("%d\n", ch);
25. }
26. fclose(open\_file);
27. return 0;
28. }
29. 测试：

程序设计题-2的运行结果如图8-9所示：

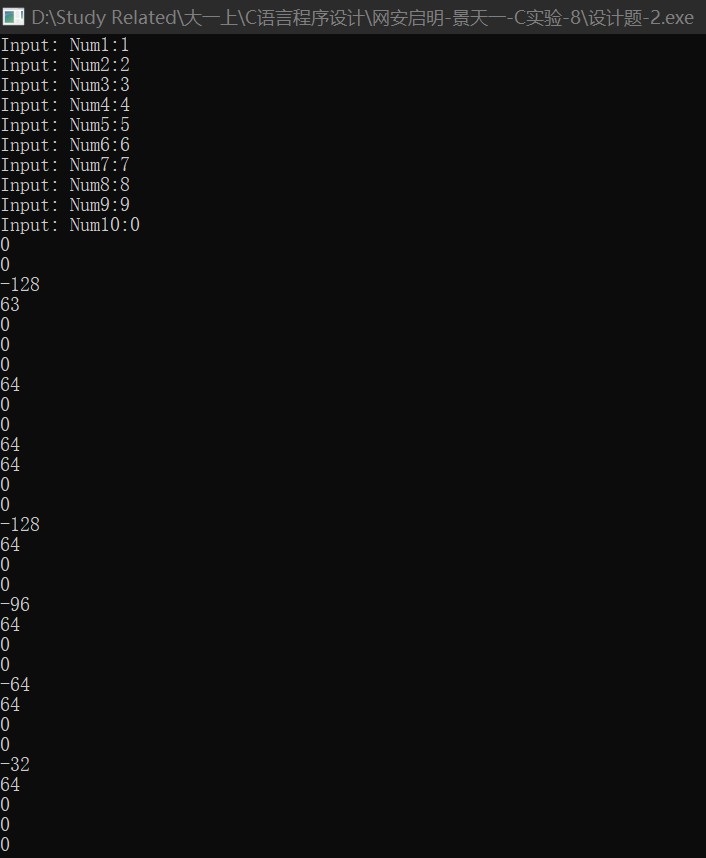


图8-9 程序设计题2的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

## 8.3 实验小结

本次实验主要学习了C语言中流式文件的使用和文件的存储方式。文件在磁盘中以二进制或字符形式存储，在程序中可以通过函数打开或读写，以特定的格式。格式决定了程序能对文件进行的操作。

尽管现阶段我学习的主要是对基本输入输出进行操作，但文件系统事实上至关重要。借由文件，程序可以进行更广泛的数据读取，不再局限于键盘的输入。流式文件的使用尽管复杂，但更贴近底层，固定的打开格式也在一定程度上保护率文件的读写安全。