

知识点1 【文件的格式化操作】 0-1

格式化的读写

函数调用：

fprintf (文件指针, 格式字符串, 输出表列) ;

fscanf (文件指针, 格式字符串, 输入表列) ;

1、fprintf 文件的写操作

```
1 void test03()
2 {
3     HERO hero[]={
4         {"德玛西亚",80, 60},
5         {"盲僧",90, 80},
6         {"小法",40, 85},
7         {"小炮",50, 90}
8     };
9     int n = sizeof(hero)/sizeof(hero[0]);
10    FILE *fp = NULL;
11    int i=0;
12
13    fp = fopen("hero.txt", "w");
14    if(fp == NULL)
15    {
16        perror("fopen");
17        return;
18    }
19
20    for(i=0;i<n;i++)
21    {
22        fprintf(fp,"英雄:%s 防御:%d 攻击:%d\n",hero[i].name,
23            hero[i].deff,hero[i].atk);
24    }
25    fclose(fp);
26 }
```

运行结果：



hero.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

英雄:德玛西亚 防御:80 攻击:60

英雄:盲僧 防御:90 攻击:80

英雄:小法 防御:40 攻击:85

英雄:小炮 防御:50 攻击:90

2、fscanf格式化 读操作

```
1 void test04()
2 {
3     HERO hero[4];
4     int i=0;
5     FILE *fp = NULL;
6     fp = fopen("hero.txt", "r");
7     if(fp == NULL)
8     {
9         perror("fopen");
10        return;
11    }
12
13    for(i=0; i<4; i++)
14    {
15        fscanf(fp, "英雄:%s 防御:%d 攻击:%d\n", hero[i].name, &hero[i].deff, &hero[i].atk);
16    }
17
18    for(i=0; i<4; i++)
19    {
20        printf("%s %d %d\n", hero[i].name, hero[i].deff, hero[i].atk);
21    }
22
23    fclose(fp);
24 }
```

运行结果：

"C:\work\c\code\day18\00_test\Debug\00_test.exe"

德玛西亚 80 60

盲僧 90 80

小法 40 85

小炮 50 90

Press any key to continue

总结：

注意：

1. 用fprintf和fscanf函数对磁盘文件读写使用方便，容易理解，但在输入时要将ASCII码转换为二进制形式，在输出时将二进制形式转换成字符，花费时间较多
2. 在内存与磁盘频繁交换数据的情况下，最好不用fprintf和fscanf函数，而用fread和fwrite函数

知识点2 【文件的随机读写】

1、知识点引入

```
1 void test05()  
2 {  
3     char buf[128]="";  
4     FILE *fp = NULL;  
5     fp = fopen("a.txt","w+");  
6     if(fp == NULL)  
7     {  
8         perror("fopen");  
9         return;  
10    }  
11  
12    //先往文件中写入 一个字符串"hello file"
```

```
13  fputs("hello file", fp);
14  //fclose(fp);
15
16  //重新打开文件 让文件的流指针 回到 文件首部
17  //fp = fopen("a.txt", "r");
18  //在从文件找那个读取 该字符串
19  fgets(buf, sizeof(buf), fp);
20  printf("buf=%s\n", buf);
21
22  fclose(fp);
23 }
```

上述代码：为啥出现这个现象（）

```
buf=屯屯屯屯屯屯屯屯屯屯屯屯  
屯屯屯屯屯屯屯屯屯屯屯屯屯屯  
屯屯屯屯屯屯屯屯屯屯屯屯屯屯？  
Press any key to continue
```

解决上述文件：文件写完后需要关闭文件然后重新打开文件让文件流指针指向文件开始位置让变下次的文件读操作

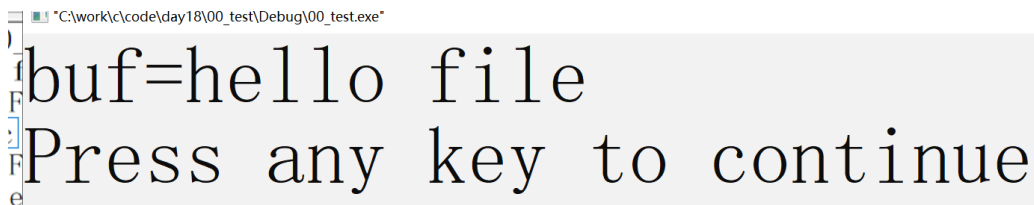
```
1 void test05()  
2 {  
3     char buf[128]="";  
4     FILE *fp = NULL;  
5     fp = fopen("a.txt", "w+");  
6     if(fp == NULL)  
7     {  
8         perror("fopen");  
9         return;  
10    }  
11  
12    //先往文件中写入 一个字符串"hello file"  
13    fputs("hello file", fp);  
14    fclose(fp);
```

```

15
16 //重新打开文件 让文件的流指针 回到 文件首部
17 fp = fopen("a.txt", "r");
18 //在从文件找那个读取 该字符串
19 fgets(buf, sizeof(buf), fp);
20 printf("buf=%s\n", buf);
21
22 fclose(fp);
23 }

```

运行结果:



```

)
f
F
]
F
e
buf=hello file
Press any key to continue

```

2、复位文件流指针 rewind 0-2

rewind函数

void rewind(文件指针);

函数功能:

把文件内部的位置指针移到文件首

调用形式:

rewind(文件指针);

例如:

```

fwrite(pa, sizeof(struct stu), 2, fp );
rewind(fp);
fread( pb, sizeof(struct stu), 2, fp);

```

```

1 void test05()
2 {
3     char buf[128]="";
4     FILE *fp = NULL;
5     fp = fopen("a.txt", "w+");
6     if(fp == NULL)
7     {
8         perror("fopen");
9     }
10    return;

```

```

10  }
11
12  //先往文件中写入 一个字符串"hello file"
13  fputs("hello file", fp);
14
15  //复位文件流指针（位置 指针）
16  rewind(fp); //注意
17
18  //在从文件找那个读取 该字符串
19  fgets(buf, sizeof(buf), fp);
20  printf("buf=%s\n", buf);
21
22  fclose(fp);
23  }

```

运行结果：



```

buf=hello file
Press any key to cont

```

3、获得 文件流指针 距离 文件首部 的字节数 ftell

ftell函数

定义函数：

```
long ftell(文件指针);
```

函数功能：

取得文件流目前的读写位置。

返回值：

返回当前位置(距离文件起始的字节数)，出错 时返回-1。

例如：

```

int length;
length = ftell(fp);

```

```

1 void test05()
2 {
3     char buf[128] = "";
4     long file_len = 0;
5     FILE *fp = NULL;

```

```
6  fp = fopen("a.txt","w+");
7  if(fp == NULL)
8  {
9      perror("fopen");
10     return;
11 }
12
13 //先往文件中写入 一个字符串"hello file"
14 fputs("hello file", fp);
15
16 //获取 文件流指针 距离 文件首部的 字节数
17 file_len = ftell(fp);
18 printf("file_len = %ld\n", file_len);
19
20 fclose(fp);
21 }
```

运行结果：

"C:\work\c\code\day18\00_test\Debug\00_test.exe"

```
file_len = 10
Press any key to c
```

4、定位 文件的流指针 fseek

fseek函数（一般用于二进制文件）

定义函数：

int fseek(文件类型指针, 位移量, 起始点);

函数功能：

移动文件流的读写位置.

说明： 起始位置

文件开头 SEEK_SET 0

文件当前位置 SEEK_CUR 1

文件末尾 SEEK_END 2

位移量: 以起始点为基点, 向前、后移动的字节数.

位移量: -10 往左边移动10字节 +10往右边移动10字节

一次性读取文件总大小

```
1  oid test06()
2  {
3      char *file_data=NULL;
4      long file_len = 0;
5      FILE *fp = NULL;
6      fp = fopen("a.txt","r");
7      if(fp == NULL)
8      {
9          perror("fopen");
10         return;
11     }
12
13     //需求：一次性的将文件数据 读取到 内存中
14     //1、得到文件的总大小
15     //a、使用fseek 将文件指针 定位到文件尾部
16     fseek(fp, 0, 2);
17     //b、使用ftell计算文件的偏移量 == 文件的总大小
18     file_len = ftell(fp);
19     //c、使用rewind复位文件流指针
20     rewind(fp);
21
22     //2、根据文件的总大小 合理申请 内存空间
23     file_data = (char *)calloc(1,file_len+1); //+1的目的 内存末尾存放'\0'
```



```

24  if(file_data == NULL)
25  {
26  fclose(fp);
27  return;
28  }
29
30  //3、一次性 将文件数据 读入到 内存空间
31  fread(file_data, file_len, 1, fp);
32
33  //4、遍历读取到的文件内容
34  printf("file_len=%ld\n", file_len);
35  printf("%s\n", file_data);
36
37  //5、释放堆区空间
38  if(file_data != NULL)
39  {
40  free(file_data);
41  file_data = NULL;
42  }
43
44  //关闭文件
45  fclose(fp);
46  }

```

运行结果：

| "C:\work\c\code\day18\00_test\Debug\00_test.exe"

[illegible]

5、feof () 函数 判断文件是否到达文件末尾

EOF宏 只能用于 文本文件

feof函数 可以用于 文本文件 二进制文件

```
1 void test07()
2 {
3     FILE *fp = NULL;
4     fp = fopen("a.txt", "r");
5     if(fp == NULL)
6     {
7         perror("fopen");
8         return;
9     }
10
11     //feof(fp) 判断文件是否 结束 0: 未结束 非0: 表示结束
12     //while( feof(fp) == 0)//文件未结束 才循环
13     while( !feof(fp) )
14     {
15         char ch = fgetc(fp);
16         printf("%c", ch);
17     }
18
19     fclose(fp);
20 }
```

运行结果;

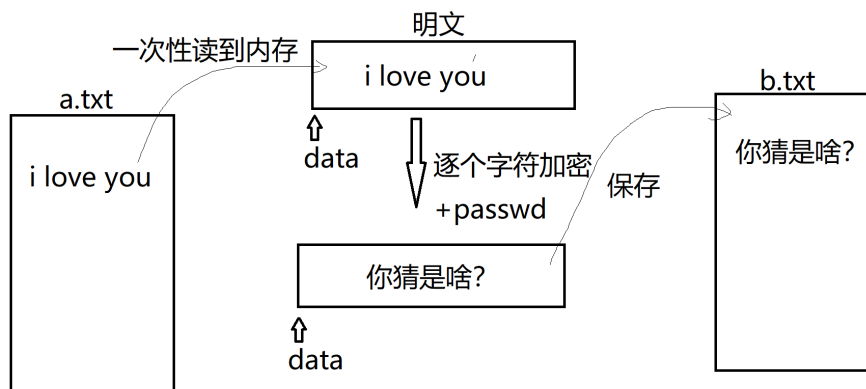
C:\work\code\day18\00_test\Debug\00_test.exe

[illegible]

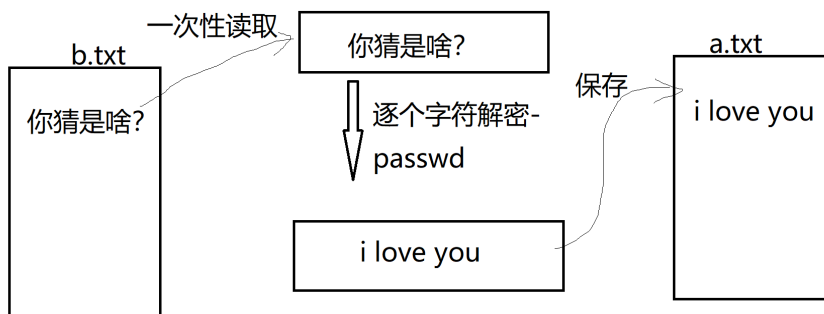
知识点3 【文件加密器】

1、原理的分析

加密过程:



解密过程:



main.c

```

1  #include<stdio.h>
2  #include"fun.h"
3  int main(int argc,char *argv[])
4  {
5      while(1)
6      {
7          int cmd = 0;
8          print_help();
9
10         scanf("%d", &cmd);
11
12         if(cmd == 1)
13         {
14             char src_file[31]="";
15             char dst_file[31]="";
16             char *file_data = NULL;
17             unsigned long file_length = 0;
18             unsigned int passwd = 0;
19
20             //1、获取源文件 目的文件名(参考API实现)
  
```

```

21  get_file_name(dst_file, src_file);
22
23  //2、获取源文件名 对应的文件 内容(参考API实现)
24  //char * read_src_file(unsigned long *file_length,char *src_file_name)
25  file_data = read_src_file(&file_length, src_file);
26
27  //3、获取用户输入的密码（自定义实现）
28  printf("请输入你的密码:");
29  scanf("%u", &passwd);
30
31  //4、对文件内容加密(参考API实现)
32  file_data = file_text_encrypt(file_data,file_length,passwd);
33  //char * file_text_encrypt(char * src_file_text,unsigned long int length,unsigned int password)
34
35  //5、对加密好的 文件内容 保存到 目的文件名中
36  save_file(file_data, file_length,dst_file);
37  //void save_file(char* text,unsigned long int length,char * file_name)
38  }
39  else if(cmd == 2)
40  {
41  char src_file[31]="";
42  char dst_file[31]="";
43  char *file_data = NULL;
44  unsigned long file_length = 0;
45  unsigned int passwd = 0;
46
47  //1、获取源文件 目的文件名(参考API实现)
48  get_file_name(dst_file, src_file);
49  //void get_file_name(char * dest_file_name,char * src_file_name)
50
51  //2、获取源文件名 对应的文件 内容(参考API实现)
52  file_data = read_src_file(&file_length, src_file);
53  //char * read_src_file(unsigned long int *file_length,char *src_file_name)
54
55  //3、获取用户输入的密码（自定义实现）
56  printf("请输入你的密码:");
57  scanf("%u", &passwd);
58
59  //4、对文件内容解密(参考API实现)

```

```

60  file_data = file_text_decrypt(file_data, file_length,passwd);
61  //char * file_text_decrypt(char * src_file_text,unsigned long int length
h,unsigned int password)
62
63  //5、对加密好的 文件内容 保存到 目的文件名中
64  save_file(file_data, file_length,dst_file);
65  //void save_file(char* text,unsigned long int length,char * file_name)
66  }
67  else if(cmd == 3)
68  {
69  break;
70  }
71  else
72  {
73  printf("请输入一个正确的选项\n");
74  }
75
76  }
77
78  return 0;
79  }

```

fun.c

```

1  #include<stdio.h>
2  #include<stdlib.h>
3  void print_help(void)
4  {
5  printf("*****1:加密文件*****\n");
6  printf("*****2:解密文件*****\n");
7  printf("*****3:退出程序*****\n");
8  }
9  void get_file_name(char * dest_file_name,char * src_file_name)
10 {
11 printf("请输入你的源文件名称(30个字符):");
12 scanf("%s", src_file_name);
13 printf("请输入你的目的文件名称(30个字符):");
14 scanf("%s", dest_file_name);
15 return;
16 }
17
18 char * read_src_file(unsigned long *file_length,char *src_file_name)

```

```
19 {
20     char *data = NULL;
21     FILE *fp = NULL;
22     fp = fopen(src_file_name, "r");
23     if(fp == NULL)
24     {
25         perror("fopen");
26         return NULL;
27     }
28
29     //获取文件总长度
30     //将文件流指针 定位 到文件 尾部
31     fseek(fp, 0, 2);
32     //获取文件长度
33     *file_length = ftell(fp);
34     //复位文件流指针
35     rewind(fp);
36
37
38     //根据 文件的长度 申请 堆区空间
39     data = (char *)calloc(1, *file_length);
40     if(data == NULL)
41     {
42         perror("calloc");
43         return NULL;
44     }
45
46     //一次性 读取文件内容
47     fread(data, *file_length, 1, fp);
48
49     fclose(fp);
50     //将空间首地址 返回
51     return data;
52 }
53 char * file_text_encrypt(char * src_file_text, unsigned long int length, unsigned int password)
54 {
55     int i=0;
56     for(i=0; i<length; i++)
57     {
58         src_file_text[i] += password; //加密的过程
```

```

59  }
60
61  return src_file_text;
62  }
63
64  void save_file(char* text,unsigned long int length,char * file_name)
65  {
66      FILE *fp = NULL;
67      fp = fopen(file_name, "w");
68      if(fp == NULL)
69      {
70          perror("fopen");
71          return;
72      }
73
74      //将data数据保存到 文件中
75      fwrite(text, length, 1, fp);
76
77      fclose(fp);
78
79      //释放text指向堆区空间
80      if(text != NULL)
81      {
82          free(text);
83          text =NULL;
84      }
85
86      printf("保存成功!\n");
87      return;
88  }
89
90  char * file_text_decrypt(char * src_file_text,unsigned long int length,unsigned int password)
91  {
92      int i=0;
93      for(i=0;i<length;i++)
94      {
95          src_file_text[i] -= password;//解密的过程
96      }
97
98      return src_file_text;

```

fun.h

```

1  #ifndef __FUN_H__
2  #define __FUN_H__
3
4  extern void print_help(void);
5  extern void get_file_name(char * dest_file_name,char * src_file_name);
6  extern char * read_src_file(unsigned long *file_length,char *src_file_name);
7  extern char * file_text_encrypt(char * src_file_text,unsigned long int length,unsigned int password);
8
9  extern void save_file(char* text,unsigned long int length,char * file_name);
10 #endif

```

项目提示：

- 1：打开文件的时候用二进制方式打开进行读写。
- 2：测文件大小方法：
 - 1) 用fseek() 定位流指针到文件的末尾。
 - 2) 用ftell()函数测流指针的位置即文件的大小。
- 3：读文件内容：
 - 1) 根据文件的大小用malloc申请内存空间保存读出的内容
 - 2) 读文件数据的时候要从文件的开始读（rewind（））。

详细设计API设计参考：

1) 从键盘获取源文件和目的文件名字

```

/*****/
//函数功能:获取 目的文件和源文件的名字
//参数:  src_file_name:源文件名字字符数组首地址。
// dest_file_name:目的文件的名字字符数组首地址
/*****/
void get_file_name(char * dest_file_name,char * src_file_name)

```

2) 从文件中读出内容

```

/*****/

```



```

//函数功能:读出文件内容
//参数: file_length:整型指针, 此地址中保存文件字节数。
// src_file_name:文件名字, 从此文件中读取内容。
// 返回值:读出字符串的首地址
// 在此函数中测文件的大小, 并malloc空间, 再把文件内容读出返回, 读出字符数组的首地址
/*****/
char * read_src_file(unsigned long int *file_length,char *src_file_name)

```

3) 字符数组加密

```

/*****/
//函数功能:加密字符串
//参数:
// src_file_text:要加密的字符串。 length:字符串的长度
// password: 加密密码
// 返回值: 加密后的字符串的首地址
// 加密原理字符数组中每个元素加上password
/*****/
char * file_text_encrypt(char * src_file_text,unsigned long int length,unsigned
int password)

```

4) 解密字符串

```

/*****/
//函数功能:解密字符串
//参数:
// src_file_text:要解密的字符串。 length:字符串的长度
// password: 解密密码
// 返回值: 解密后的字符串的首地址
//思想;把数组中的每个元素减去password 给自己赋值。
/*****/
char * file_text_decrypt(char * src_file_text,unsigned long int length,unsigned
int password)

```

5) 保存文件

```

/*****/

```

```

//函数功能:将字符串保存到目的文件中
//参数:
// text:要保存的字符串首地址
// file_name :目的文件的名字
// length:字符串的长度
//思想: 传入字符数组的首地址和数组的大小及保存后的文件的名字, 即可保存数组到文件中

/*****/
void save_file(char* text,unsigned long int length,char * file_name)

```

6) 打印文件信息

```

/*****/
//
//  函数功能:打印帮助信息
//
//
/*****/
void print_help()
{
printf("*****1:加密文件*****\n");
printf("*****2:解密文件*****\n");
printf("*****3:退出程序*****\n");
}

```