《传智播客 C语言就业班》 第四讲

文件专题（主要参考C程序设计_谭）

标准文件的读写

- 文件的打开fopen()

文件的打开操作表示将给用户指定的文件在内存分配一个FILE结构区，并将该结构的指针返回给用户程序，以后用户程序就可用此FILE指针来实现对指定文件的存取操作了。当使用打开函数时，必须给出文件名、文件操作方式(读、写或读写),如果该文件名不存在，就意味着建立(只对写文件而言，对读文件则出错)，并将文件指针指向文件开头。若已有一个同名文件存在，则删除该文件，若无同名文件，则建立该文件，并将文件指针指向文件开头。

- fopen(char *filename,char *type);

其中*filename是要打开文件的文件名指针，一般用双引号括起来的文件名表示，也可使用双反斜杠隔开的路径名。而*type参数表示了对打开文件的操作方式。其可采用的操作方式如下：

字符串	说明
r	以只读方式打开文件，该文件必须存在。
r+	以读/写方式打开文件，该文件必须存在。
rb+	以读/写方式打开一个二进制文件，只允许读/写数据。
rt+	以读/写方式打开一个文本文件，允许读和写。
w	打开只写文件，若文件存在则长度清为 0，即该文件内容消失，若不存在则创建该文件。
w+	打开可读/写文件，若文件存在则文件长度清为零，即该文件内容会消失。若文件不存在则建立该文件。
a	以附加的方式打开只写文件。若文件不存在，则会建立该文件，如果文件存在，写入的数据会被加到文件尾，即文件原先的内容会被保留（EOF 符保留）。
a+	以附加方式打开可读/写的文件。若文件不存在，则会建立该文件，如果文件存在，则写入的数据会被加到文件尾后，即文件原先的内容会被保留（原来的 EOF 符不保留）。
wb	以只写方式打开或新建一个二进制文件，只允许写数据。
wb+	以读/写方式打开或建立一个二进制文件，允许读和写。
wt+	以读/写方式打开或建立一个文本文件，允许读写。
at+	以读/写方式打开一个文本文件，允许读或在文本末追加数据。
ab+	以读/写方式打开一个二进制文件，允许读或在文件末追加数据。

当用fopen(0成功的打开一个文件时，该函数将返回一个FILE指针，如果文件打开失败，将返回一个NULL指针。

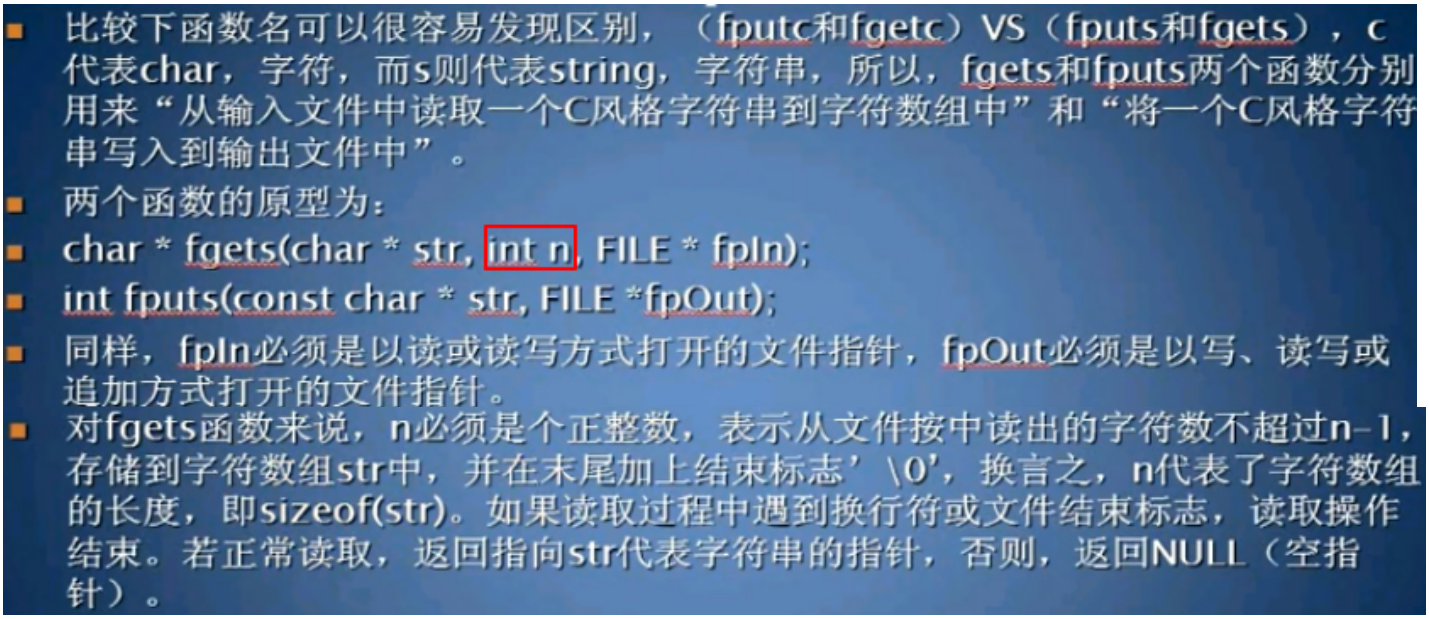
- 按字符读写文件——fgetc、fputc；这里以读为例

```
#define _CRT_SECURE_NO_WARNINGS
#include "stdio.h"
#include "string.h"
#include "stdlib.h"

void main()
{
    int i = 0;
    FILE *fp = NULL;
    char a[100];
    //不要这样写
    //char *fname2 = "c:\\1.txt";
    //统一用45°斜杠，因为Linux也通用,如下
    char *fname = "C:/Users/Administrator/Desktop/1.txt";
    fp = fopen(fname, "r"); //r只读
    if (NULL == fp) //防止只写一个等号而未报错，注意这种写法！
    {
        printf("func fopen() error: %s\n", fname); //一定注意把路径打出来！
    }
    while(!feof(fp))
    {
```

```
        char tmpC = fgetc(fp);
        printf("%c", tmpC);
    }
    if (NULL != fp) //防止只写一个等号而未报错，注意这种写法！
    {
        fclose(fp);
    }
    system("pause");
}
```

- 按行读写文件（重点！）——fgets、fputs；这里以fputs写为例

```
void main()
{
    int i = 0;
    FILE *fp = NULL;
    char a[100] = "abcdefg11111111";
    char *fname = "C:/Users/Administrator/Desktop/1.txt";
    fp = fopen(fname, "r+");  //r+ 读写文件，文件必须存在否则报错！
    if (NULL == fp)
    {
        printf("func fopen() error: %s\n", fname); //一定注意把路径打出来！
    }
    fputs(a, fp);
    if (NULL != fp)
    {
        fclose(fp);
    }
}
```



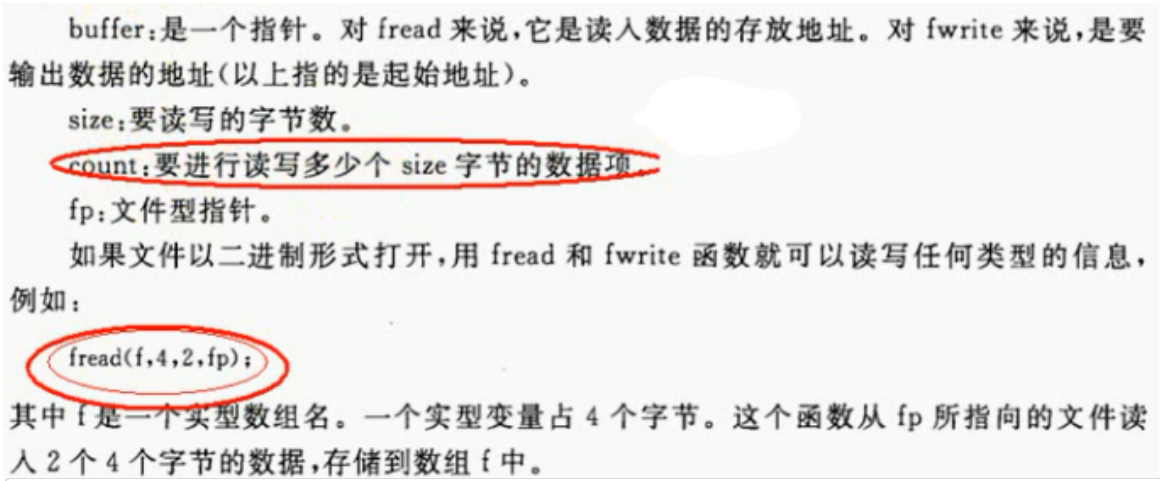
注：
1、fgets需要传入内存首地址及内存块的大小，否则容易越界；fgets 返回值为内存首地址，可以判断函数有没有操作成功，也便于函数的链式操作。
2、文件在磁盘上有两种存放方式：1. ASCII存放方式　2. 非ASCII（bin，二进制方式，非可见；） base64编码可见

- 按块读写文件——fwrite、fread；（按照bin方式打开文件）

调用形式：

```
fread(buffer,size,count,fp);
fwrite(buffer,size,count,fp);
```

其中：



buffer：是一个指针。对 fread 来说，它是读入数据的存放地址。对 fwrite 来说，是要输出数据的地址（以上指的是起始地址）。

size：要读写的字节数。

count：要进行读写多少个 size 字节的数据项。

fp：文件型指针。

如果文件以二进制形式打开，用 fread 和 fwrite 函数就可以读写任何类型的信息，例如：

```
fread(f,4,2,fp);
```

其中 f 是一个实型数组名。一个实型变量占 4 个字节。这个函数从 fp 所指向的文件读入 2 个 4 个字节的数据，存储到数组 f 中。

fwrite用法
size_t fwrite(const void* buffer, size_t size, size_t count, FILE* stream);
返回值：返回实际写入的数据块数目
（1）buffer：是一个指针，对fwrite来说，是要获取数据的地址；
（2）size：要写入内容的单字节数，即“**每次读多少字节**”；
（3）count:要进行写入size字节的数据项的个数，即“**读多少次**”；
（4）stream:目标文件指针；
（5）返回实际写入的数据项个数count。
说明：写入到文件的哪里？这个与文件的打开模式有关，如果是w+，则是从file pointer指向的地址开始写，替换掉之后的内容，文件的长度可以不变，stream的位置移动count个数；如果是a+，则从文件的末尾开始添加，文件长度加大。
fseek对此函数有作用，但是fwrite [1] 函数写到用户空间缓冲区，并未同步到文件中，所以修改后要将内存与文件同步可以用fflush（FILE *fp）函数同步。

注意：
buffer：主调函数分配内存，被调函数使用，把数据往fp里面写；
fwrite返回值为写入的次数count，比如往磁盘里面写100次，count=100但返回值为99，则可能在写的过程中发生I/O错误，或磁盘写满了；
代码实现：

```
for(i=0;i<5;i++)
{
   if(fwrite(&student[i], sizeof(Student), 1, fp) != 1)
      printf("file write error\n");
}

writtenlen = fwrite(cryptdata, 1, plainlen, fp2);
if(writtenlen != plainlen)
{
   ret = -3;
   printf("写新文件失败\n");
   goto End;
}
```

配置文件读写项目（文件控制）

```
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
int GetCfgItem(char *pFileName /*in*/, char *pKey /*in*/, char * pValue/*in out*/, int * pValueLen /*out*/);
int WriteCfgItem(char *pFileName /*in*/, char *pItemName /*in*/, char *pItemValue/*in*/, int itemValueLen /*in*/);
//实现流程
//打开文件
//按照行的方式 循环读文件
//解析每一行，若匹配key关键字，在进行value值的提取
//提取value值需要去除前后空格 1级指针典型应用
#define  LineMaxLen 2048
#define  KeyMaxLen 64
int GetCfgItem(char *pFileName /*in*/, char *pKey /*in*/, char * pValue/*in out*/, int * pValueLen /*out*/)
{
    int         rv = 0;
    FILE   *fp = NULL;
    char   lineBuf[LineMaxLen];
    char   *pTmp = NULL, *pBegin = NULL, *pEnd = NULL;
    if (pFileName==NULL || pKey==NULL || pValue==NULL || pValueLen==NULL)
    {
```

```
        rv = -1;
        printf("GetCfgItem() err. param err \n");
        goto End;
    }
    fp = fopen(pFileName, "r");
    if (fp == NULL)
    {
        rv = -2;
        printf("fopen() err. \n");
        goto End;
    }
    while (!feof(fp))
    {
        //读每一行
        memset(lineBuf, 0, sizeof(lineBuf));
        pTmp = fgets(lineBuf, LineMaxLen, fp);
        if (pTmp == NULL)
        {
            break;
        }
        //不含=, 非配置项
        pTmp = strchr(lineBuf, '=');
        if (pTmp == NULL)
        {
            continue;
        }
        //key是否在本行
        pTmp = strstr(lineBuf, pKey);
        if (pTmp == NULL)
        {
            continue;
        }

        //调整到=右边，取value准备
        pTmp = strchr(lineBuf, '=');
        if (pTmp == NULL)
        {
            continue;
        }
        pTmp = pTmp + 1;
        //获取value 起点
        while (1)
        {
            if (*pTmp == ' ')
            {
                pTmp ++ ;
            }
            else
            {
                pBegin = pTmp;
                if (*pBegin == '\n')
                {
                    //没有配置value
                    printf("配置项:%s 没有配置value \n", pKey);
                    goto End;
                }
                break;
            }
        }

        //获取valude结束点
        while (1)
        {
            if ((*pTmp == ' ' || *pTmp == '\n'))
            {
                break;
            }
            else
            {
                pTmp ++;
            }
        }
        pEnd = pTmp;
        //赋值
        *pValueLen = pEnd-pBegin;
        memcpy(pValue, pBegin, pEnd-pBegin);
        break;
```

```
    }
End:
    if (fp != NULL)
    {
        fclose(fp);
    }

    return rv;
}
//实现流程
//循环读每一行，检查key配置项是否存在 若存在修改对应value值
//若不存在，在文件末尾 添加 "key = value"
//难点：如何修改文件流中的值
int SetCfgItem(char *pFileName /*in*/, char *pKey /*in*/, char * pValue/*in*/, int ValueLen /*in*/)
{
    int         rv = 0, iTag = 0, length = 0;
    FILE   *fp = NULL;
    char   lineBuf[LineMaxLen];
    char   *pTmp = NULL, *pBegin = NULL, *pEnd = NULL;
    char   filebuf[1024*8] = {0};

    if (pFileName==NULL || pKey==NULL || pValue==NULL)
    {
        rv = -1;
        printf("SetCfgItem() err. param err \n");
        goto End;
    }

    fp = fopen(pFileName, "r+");
    if (fp == NULL)
    {
        rv = -2;
        printf("fopen() err. \n");
        //goto End;
    }
    if (fp == NULL)
    {
        fp = fopen(pFileName, "w+t");
        if (fp == NULL)
        {
            rv = -3;
            printf("fopen() err. \n");
            goto End;
        }
    }

    fseek(fp, 0L, SEEK_END); //把文件指针从0位置开始，移动到文件末尾
    //获取文件长度;
    length = ftell(fp);
    fseek(fp, 0L, SEEK_SET);

    if (length > 1024*8)
    {
        rv = -3;
        printf("文件超过1024*8, nunsupport");
        goto End;
    }

    while (!feof(fp))
    {
        //读每一行
        memset(lineBuf, 0, sizeof(lineBuf));
        pTmp = fgets(lineBuf, LineMaxLen, fp);
        if (pTmp == NULL)
        {
            break;
        }

        //key关键字是否在本行
        pTmp = strstr(lineBuf, pKey);
        if (pTmp == NULL)
        {
            strcat(filebuf, lineBuf);
            continue;
        }
        else
        {
```

```
            sprintf(lineBuf, "%s = %s\n", pKey, pValue);
            strcat(filebuf, lineBuf);
            //若存在key
            iTag = 1;
        }
    }
    //若不存在 追加
    if (iTag == 0)
    {
        fprintf(fp, "%s = %s\n", pKey, pValue);
    }
    else //若存在
    {
        if (fp != NULL)
        {
            fclose(fp);
            fp = NULL; //避免野指针
        }
        fp = fopen(pFileName, "w+t");
        if (fp == NULL)
        {
            rv = -4;
            printf("fopen() err. \n");
            goto End;
        }
        fputs(filebuf, fp);
        //fwrite(filebuf, sizeof(char), strlen(filebuf), fp);
    }

End:
    if (fp != NULL)
    {
        fclose(fp);
    }
    return rv;
}
```

接口封装和设计思路分析

配置文件读写案例实现分析

1、 功能划分
   a) 界面测试（功能集成）
   自己动手规划接口模型。
   b) 配置文件读写
      i. 配置文件读（根据 key，读取 value）
      ii. 配置文件写（输入 key、value）
      iii. 配置文件修改（输入 key、value）
      iv. 优化 ===》接口要求紧 模块要求松
2、 实现及代码讲解
3、 测试。

注意：在软件开发中，**接口要求紧，模块要求松。。。**（模块之间松耦合，接口尽量少，尽可能把能做的都做了，不要留给上层A做，如WriteIniCfg和ModifyIniCfg最好写成一个接口，下层B自己来处理是写入还是修改）

- **cfg_op.h**

```
#ifndef __CFG_OP_H__
#define __CFG_OP_H__

#ifdef __cplusplus
extern "C" {
#endif

//获取配置项
int GetCfgItem(char *pFileName /*in*/, char *pKey /*in*/, char * pValue/*in out*/, int * pValueLen /*out*/);

//写配置项
//int WriteCfgItem(FILE *fp /*in*/, char *pItemName /*in*/, char *pItemValue /*in*/, int itemValueLen /*in*/)
//上面的方式不合理，不应该让上层来执行file的open工作，应该传入文件名。
```

```
int WriteCfgItem(char *pFileName /*in*/, char *pItemName /*in*/, char *pItemValue/*in*/, int itemValueLen /*in*/);

#ifdef  __cplusplus
}
#endif

#endif
```

- **cfg_op.c**

```
#define  _CRT_SECURE_NO_WARNINGS
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

#define MaxLine 2048

//获取配置项
int GetCfgItem(char *pFileName /*in*/, char *pKey /*in*/, char * pValue/*in out*/, int * pValueLen /*out*/)
{
   int       ret = 0;
   FILE    *fp = NULL;
   char    *pTmp = NULL, *pEnd = NULL, *pBegin = NULL;

   char lineBuf[MaxLine];

   fp = fopen(pFileName, "r");
   if (fp == NULL)
   {
      ret = -1;
      return ret;
   }

   while (!feof(fp))
   {
      memset(lineBuf, 0, sizeof(lineBuf));
       //fgets(_Out_z_cap_(_MaxCount) char * _Buf, _In_ int _MaxCount, _Inout_ FILE * _File);
      fgets(lineBuf, MaxLine, fp);
      //printf("lineBuf:%s ",lineBuf );

      pTmp = strchr(lineBuf, '='); //
      if (pTmp == NULL) //没有=号
      {
         continue;
      }

      pTmp = strstr(lineBuf, pKey);
      if (pTmp == NULL) //判断key是不是在 //所在行 是不是有key
      {
         continue;
      }
      pTmp = pTmp + strlen(pKey); //mykey1 = myvalude11111111 ==> "= myvalude1111111"

      pTmp = strchr(pTmp, '=');
      if (pTmp == NULL) //判断key是不是在 //所在行 是不是有key
      {
         continue;
      }
      pTmp = pTmp + 1;
      //
      //printf("pTmp:%s ", pTmp);

      //获取value 起点
      while (1)
      {
         if (*pTmp == ' ')
         {
            pTmp ++ ;
         }
         else
         {
            pBegin = pTmp;
            if (*pBegin == '\n')
            {
```

```
                //没有配置value
                //printf("配置项:%s 没有配置value \n", pKey);
                goto End;
            }
            break;
        }
    }

    //获取valude结束点
    while (1)
    {
        if ((*pTmp == ' ' || *pTmp == '\n'))
        {
            break;
        }
        else
        {
            pTmp ++;
        }
    }
    pEnd = pTmp;

    //赋值
    *pValueLen = pEnd-pBegin;
    memcpy(pValue, pBegin, pEnd-pBegin);
  }

End:
  if (fp == NULL)
  {
    fclose(fp);
  }
  return 0;
}

//写配置项
//实现流程
//循环读每一行，检查key配置项是否存在 若存在修改对应value值
//若不存在，在文件末尾 添加 "key = value"
//难点：如何修改文件流中的值
int WriteCfgItem(char *pFileName /*in*/, char *pKey /*in*/, char * pValue/*in*/, int ValueLen /*in*/)
{
  int       rv = 0, iTag = 0, length = 0;
  FILE    *fp = NULL;
  char    lineBuf[MaxLine];
  char    *pTmp = NULL, *pBegin = NULL, *pEnd = NULL;
  char    filebuf[1024*8] = {0};

  if (pFileName==NULL || pKey==NULL || pValue==NULL)
  {
    rv = -1;
    printf("SetCfgItem() err. param err \n");
    goto End;
  }

  fp = fopen(pFileName, "r+");
  if (fp == NULL)
  {
    rv = -2;
    printf("fopen() err. \n");
    //goto End;
  }

  if (fp == NULL)
  {
    fp = fopen(pFileName, "w+t");
    if (fp == NULL)
    {
      rv = -3;
      printf("fopen() err. \n");
      goto End;
    }
  }

  fseek(fp, 0L, SEEK_END); //把文件指针从0位置开始，移动到文件末尾
  //获取文件长度;
  length = ftell(fp);
```

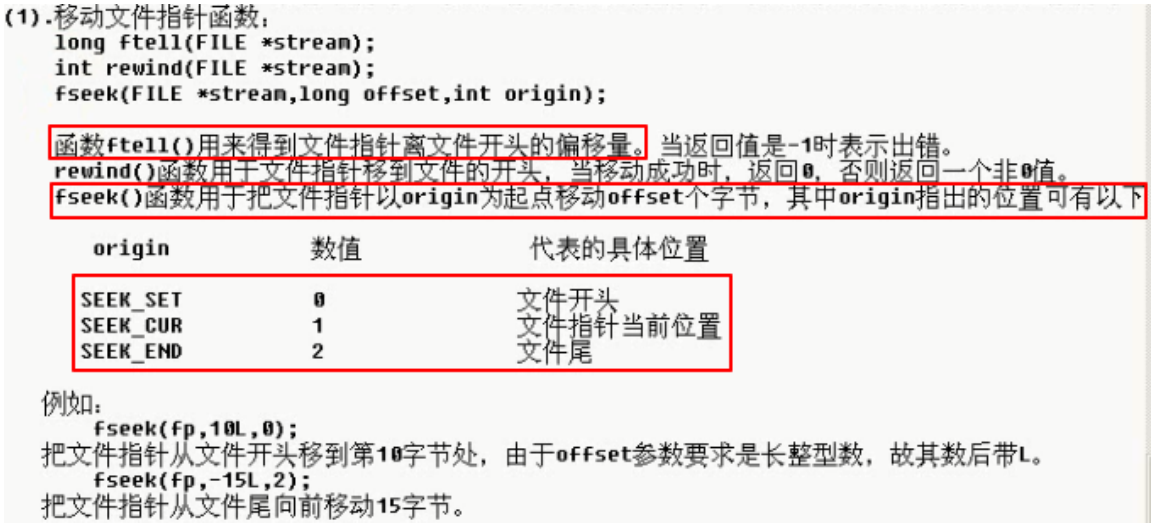
```
    fseek(fp, 0L, SEEK_SET);

    if (length > 1024*8)
    {
        rv = -3;
        printf("文件超过1024*8, nunsupport");
        goto End;
    }

    while (!feof(fp))
    {
        //读每一行
        memset(lineBuf, 0, sizeof(lineBuf));
        pTmp = fgets(lineBuf, MaxLine, fp);
        if (pTmp == NULL)
        {
            break;
        }

        //key关键字是否在本行
        pTmp = strstr(lineBuf, pKey);
        if (pTmp == NULL) //key关键字不在本行， copy到filebuf中
        {
            strcat(filebuf, lineBuf);
            continue;
        }
        else  //key关键在在本行中，替换旧的行，再copy到filebuf中
        {
            sprintf(lineBuf, "%s = %s\n", pKey, pValue);
            strcat(filebuf, lineBuf);
            //若存在key
            iTag = 1;
        }
    }

    //若key关键字，不存在 追加
    if (iTag == 0)
    {
        fprintf(fp, "%s = %s\n", pKey, pValue);
    }
    else //若key关键字，存在，则重新创建文件
    {
        if (fp != NULL)
        {
            fclose(fp);
            fp = NULL; //避免野指针
        }

        fp = fopen(pFileName, "w+t");
        if (fp == NULL)
        {
            rv = -4;
            printf("fopen() err. \n");
            goto End;
        }
        fputs(filebuf, fp);
        //fwrite(filebuf, sizeof(char), strlen(filebuf), fp);
    }

End:
    if (fp != NULL)
    {
        fclose(fp);
    }
    return rv;
}
```

- **配置文件集成测试框架.c**

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
```

```
#include "cfg_op.h"

#define CFGNAME "c:/mycfg.ini"
void mymenu()
{
    printf("============================\n");
    printf("1 测试写配置文件\n");
    printf("2 测试读配置文件\n");
    printf("0 退出\n");
    printf("============================\n");
}

//获取配置项
int TGetCfg()
{
    int       ret = 0;
    //读配置项
    char    name[1024] = {0};
    char    valude[1024] = {0};
    int       vlen = 0;

    printf("\n请键入key:");
    scanf("%s", name);

    ret = GetCfgItem(CFGNAME /*in*/, name /*in*/, valude/*in*/, &vlen);
    if (ret != 0)
    {
        printf("func WriteCfgItem err:%d \n", ret);
        return ret;
    }
    printf("valude:%s \n", valude);
}

//写配置项
int TWriteCfg()
{
    int       ret = 0;
    //写配置项
    char name[1024] = {0};
    char valude[1024] = {0};

    printf("\n请键入key:");
    scanf("%s", name);

    printf("\n请键入valude:");
    scanf("%s", valude);

    ret = WriteCfgItem(CFGNAME /*in*/, name /*in*/, valude/*in*/,strlen(valude) /*in*/);
    if (ret != 0)
    {
        printf("func WriteCfgItem err:%d \n", ret);
        return ret;
    }
    printf("你的输入是：%s = %s \n", name , valude);
    return ret;
}

void main()
{
    int choice;

    for (;;)
    {
        //显示一个菜单
        mymenu();
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:   //写配置项
            TWriteCfg();
            break;
        case 2:
            TGetCfg(); //读配置项
            break;
        case 0:
            exit(0);
        default:;
```

```
        exit(0);
      }
   }
   printf("hello...\n");
   system("pause");
   return ;
}
```



用法举例：获取文件长度

```
fseek(fp, 0L, SEEK_END);  //把文件指针从o位置开始，移动到文件末尾

length = ftell(tp);  //获取文件长度

fseek(fp, 0L, SEEK_SET);
```

配置文件读写小项目思路分析

1、项目的总体需求

- 把结构体写配置
- 把结构体读配置（显示在一个小界面）
- 结构体修改
- 配置文件读写api来完成这个功能

2、着手项目启动（项目经理来做）

- 人员分工
- 模块分工 分**三个层**：（1）界面层、（2）业务层（集成配置文件api）、（3）集成第三方api
- 初步项目模型搭建
- 资源的整合

3、项目开发流程

- 概要设计：具体模块的划分、总体的业务流、流程设计；**概要设计结束的一个明显的标志：所有的表都定义好了**；而且表的E-R图都已经设计完毕，即所有的实体实体都已经落地，都能够保存到数据库或配置文件。
- 详细设计：流程细化，重要的流程可以兑现代码；

4、编码

char []转int：用atoi()函数；
int转char []：用sprintf()函数；

加密

- 加密：y=ax+b，密钥相当于a和b；分为对称加密和非对称加密。

- 对称加密算法：加密的密钥和解密的密钥一样。des和3des加密是分组加密；des密钥为8字节，3des一般为16或24字节。注：分组加密，即把明文按密钥分组，如8个字节一组，每组给密钥做运算，形成密文；如果最后不够8个字节怎么办——缺几补几，如缺3个字节就补“333”，这样就可以区分出明文的7和打补丁的7（解密后，如果最后一个字符是几，就舍去几位即可，这样明文如果最后一位是3也不受影响）。
- 非对称加密算法：加密的密钥和解密的密钥长度不一样；非对称加密的特点是，加密速度慢，但加密强度高，如rsa1024，rsa2048.
- 加密三要素：明文密文、算法、密钥。

代码实现如下：

- **des.h**

```
/*********************************************************
*  des.h
*  用户使用des算法头文件
*
*********************************************************/
#ifndef _OPENDESS_H_
#define _OPENDESS_H_

#ifdef __cplusplus
extern "C" {
#endif

//ab\0defg

//用户使用的函数
int DesEnc(
       unsigned char *pInData,
       int           nInDataLen,
       unsigned char *pOutData,
       int          *pOutDataLen);

//用户使用函数des解密
int DesDec(
      unsigned char *pInData,
      int           nInDataLen,
      unsigned char *pOutData,
      int          *pOutDataLen);

#ifdef __cplusplus
}
#endif

#endif
```

- **des.c**

```
/*********************************************************
*
*  des.c
*  common des.....
*
*********************************************************/

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

/*********************************************************
  data type definition for Des;
*********************************************************/
#define EN0    0
#define DE1    1

#define DES_KEYBYTES    128
#define DES_KEYLONGS     32
#define DES_BLOCKLEN     8
```

```
typedef struct {
   unsigned char ek[DES_KEYBYTES];
   int    ekLen;
   unsigned char dk[DES_KEYBYTES];
   int    dkLen;
   unsigned char CbcCtx[DES_BLOCKLEN];
} DES_CTX;

typedef struct {
   unsigned char ek1[DES_KEYBYTES];
   int    ek1Len;
   unsigned char dk1[DES_KEYBYTES];
   int    dk1Len;
   unsigned char ek2[DES_KEYBYTES];
   int    ek2Len;
   unsigned char dk2[DES_KEYBYTES];
   int    dk2Len;
   unsigned char CbcCtx[DES_BLOCKLEN];
   //int    IsFirstBlock;
} DES3_CTX;

static unsigned char pc1[56] = {
   56, 48, 40, 32, 24, 16,  8,  0, 57, 49, 41, 33, 25, 17,
    9,  1, 58, 50, 42, 34, 26, 18, 10,  2, 59, 51, 43, 35,
   62, 54, 46, 38, 30, 22, 14,  6, 61, 53, 45, 37, 29, 21,
   13,  5, 60, 52, 44, 36, 28, 20, 12,  4, 27, 19, 11,  3 };

static unsigned char pc2[48] = {
   13, 16, 10, 23,  0,  4,        2, 27, 14,  5, 20,  9,
   22, 18, 11,  3, 25,  7,    15,  6, 26, 19, 12,  1,
   40, 51, 30, 36, 46, 54,       29, 39, 50, 44, 32, 47,
   43, 48, 38, 55, 33, 52,    45, 41, 49, 35, 28, 31 };

static unsigned short bytebit[8] = {0200,0100,040,020,010,04,02,01 };
static unsigned char totrot[16] = {1,2,4,6,8,10,12,14,15,17,19,21,23,25,27,28};
static unsigned long bigbyte[24] = {
   0x800000L,    0x400000L,    0x200000L,    0x100000L,
   0x80000L,    0x40000L,    0x20000L,    0x10000L,
   0x8000L,    0x4000L,    0x2000L,    0x1000L,
   0x800L,       0x400L,       0x200L,       0x100L,
   0x80L,       0x40L,       0x20L,       0x10L,
   0x8L,       0x4L,       0x2L,       0x1L    };

//insert digits
static unsigned long SP1[64] ={
      0x01010400l,0x00000000l,0x00010000l,0x01010404l,
      0x01010004l,0x00010404l,0x00000004l,0x00010000l,
      0x00000400l,0x01010400l,0x01010404l,0x00000400l,
      0x01000404l,0x01010004l,0x01000000l,0x00000004l,
      0x00000404l,0x01000400l,0x01000400l,0x00010400l,
      0x00010400l,0x01010000l,0x01010000l,0x01000404l,
      0x00010004l,0x01000004l,0x01000004l,0x00010004l,
      0x00000000l,0x00000404l,0x00010404l,0x01000000l,
      0x00010000l,0x01010404l,0x00000004l,0x01010000l,
      0x01010400l,0x01000000l,0x01000000l,0x00000400l,
      0x01010004l,0x00010000l,0x00010400l,0x01000004l,
      0x00000400l,0x00000004l,0x01000404l,0x00010404l,
      0x01010404l,0x00010004l,0x01010000l,0x01000404l,
      0x01000004l,0x00000404l,0x00010404l,0x01010400l,
      0x00000404l,0x01000400l,0x01000400l,0x00000000l,
      0x00010004l,0x00010400l,0x00000000l,0x01010004l };


static unsigned long SP2[64]={
      0x80108020l,0x80008000l,0x00008000l,0x00108020l,
      0x00100000l,0x00000020l,0x80100020l,0x80008020l,
      0x80000020l,0x80108020l,0x80108000l,0x80000000l,
      0x80008000l,0x00100000l,0x00000020l,0x80100020l,
      0x00108000l,0x00100020l,0x80008020l,0x00000000l,
      0x80000000l,0x00008000l,0x00108020l,0x80100000l,
      0x00100020l,0x80000020l,0x00000000l,0x00108000l,
      0x00008020l,0x80108000l,0x80100000l,0x00008020l,
      0x00000000l,0x00108020l,0x80100020l,0x00100000l,
      0x80008020l,0x80100000l,0x80108000l,0x00008000l,
      0x80100000l,0x80008000l,0x00000020l,0x80108020l,
      0x00108020l,0x00000020l,0x00008000l,0x80000000l,
```

```
    0x00008020l,0x80108000l,0x00100000l,0x80000020l,
    0x00100020l,0x80008020l,0x80000020l,0x00100020l,
    0x00108000l,0x00000000l,0x80008000l,0x00008020l,
    0x80000000l,0x80100020l,0x80108020l,0x00108000l };


static unsigned long SP3[64]={
    0x00000208l,0x08020200l,0x00000000l,0x08020008l,
    0x08000200l,0x00000000l,0x00020208l,0x08000200l,
    0x00020008l,0x08000008l,0x08000008l,0x00020000l,
    0x08020208l,0x00020008l,0x08020000l,0x00000208l,
    0x08000000l,0x00000008l,0x08020200l,0x00000200l,
    0x00020200l,0x08020000l,0x08020008l,0x00020208l,
    0x08000208l,0x00020200l,0x00020000l,0x08000208l,
    0x00000008l,0x08020208l,0x00000200l,0x08000000l,
    0x08020200l,0x08000000l,0x00020008l,0x00000208l,
    0x00020000l,0x08020200l,0x08000200l,0x00000000l,
    0x00000200l,0x00020008l,0x08020208l,0x08000200l,
    0x08000008l,0x00000200l,0x00000000l,0x08020008l,
    0x08000208l,0x00020000l,0x08000000l,0x08020208l,
    0x00000008l,0x00020208l,0x00020200l,0x08000008l,
    0x08020000l,0x08000208l,0x00000208l,0x08020000l,
    0x00020208l,0x00000008l,0x08020008l,0x00020200l };


static unsigned long SP4[64]={
    0x00802001l,0x00002081l,0x00002081l,0x00000080l,
    0x00802080l,0x00800081l,0x00800001l,0x00002001l,
    0x00000000l,0x00802000l,0x00802000l,0x00802081l,
    0x00000081l,0x00000000l,0x00800080l,0x00800001l,
    0x00000001l,0x00002000l,0x00800000l,0x00802001l,
    0x00000080l,0x00800000l,0x00002001l,0x00002080l,
    0x00800081l,0x00000001l,0x00002080l,0x00800080l,
    0x00002000l,0x00802080l,0x00802081l,0x00000081l,
    0x00800080l,0x00800001l,0x00802000l,0x00802081l,
    0x00000081l,0x00000000l,0x00000000l,0x00802000l,
    0x00002080l,0x00800080l,0x00800081l,0x00000001l,
    0x00802001l,0x00002081l,0x00002081l,0x00000080l,
    0x00802081l,0x00000081l,0x00000001l,0x00002000l,
    0x00800001l,0x00002001l,0x00802080l,0x00800081l,
    0x00002001l,0x00002080l,0x00800000l,0x00802001l,
    0x00000080l,0x00800000l,0x00002000l,0x00802080l };


static unsigned long SP5[64]={
    0x00000100l,0x02080100l,0x02080000l,0x42000100l,
    0x00080000l,0x00000100l,0x40000000l,0x02080000l,
    0x40080100l,0x00080000l,0x02000100l,0x40080100l,
    0x42000100l,0x42080000l,0x00080100l,0x40000000l,
    0x02000000l,0x40080000l,0x40080000l,0x00000000l,
    0x40000100l,0x42080100l,0x42080100l,0x02000100l,
    0x42080000l,0x40000100l,0x00000000l,0x42000000l,
    0x02080100l,0x02000000l,0x42000000l,0x00080100l,
    0x00080000l,0x42000100l,0x00000100l,0x02000000l,
    0x40000000l,0x02080000l,0x42000100l,0x40080100l,
    0x02000100l,0x40000000l,0x42080000l,0x02080100l,
    0x40080100l,0x00000100l,0x20000000l,0x42080000l,
    0x42080100l,0x00080100l,0x42000000l,0x42080100l,
    0x02080000l,0x02000100l,0x40000100l,0x00080000l,
    0x00080100l,0x02000100l,0x40000100l,0x00080000l,
    0x00000000l,0x40080000l,0x02080100l,0x40000100l };


static unsigned long SP6[64]={
    0x20000010l,0x20400000l,0x00004000l,0x20404010l,
    0x20400000l,0x00000010l,0x20404010l,0x00400000l,
    0x20004000l,0x00404010l,0x00400000l,0x20000010l,
    0x00400010l,0x20004000l,0x20000000l,0x00004010l,
    0x00000000l,0x00400010l,0x20004010l,0x00004000l,
    0x00404000l,0x20004010l,0x00000010l,0x20400010l,
    0x20400010l,0x00000000l,0x00404010l,0x20404000l,
    0x00004010l,0x00404000l,0x20404000l,0x20000000l,
    0x20004000l,0x00000010l,0x20400010l,0x00404000l,
    0x20404010l,0x00400000l,0x00004010l,0x20000010l,
    0x00400000l,0x20004000l,0x20000000l,0x00004010l,
    0x20000010l,0x20404010l,0x00404000l,0x20400000l,
    0x00404010l,0x20404000l,0x00000000l,0x20400010l,
```

```
    0x00000010l,0x00004000l,0x20400000l,0x00404010l,
    0x00004000l,0x00400010l,0x20004010l,0x00000000l,
    0x20404000l,0x20000000l,0x00400010l,0x20004010l };

static unsigned long SP7[64] = {
   0x00200000L, 0x04200002L, 0x04000802L, 0x00000000L,
   0x00000800L, 0x04000802L, 0x00200802L, 0x04200800L,
   0x04200802L, 0x00200000L, 0x00000000L, 0x04000002L,
   0x00000002L, 0x04000000L, 0x04200002L, 0x00000802L,
   0x04000800L, 0x00200802L, 0x00200002L, 0x04000800L,
   0x04000002L, 0x04200000L, 0x04200800L, 0x00200002L,
   0x04200000L, 0x00000800L, 0x00000802L, 0x04200802L,
   0x00200800L, 0x00000002L, 0x04000000L, 0x00200800L,
   0x04000000L, 0x00200800L, 0x00200000L, 0x04000802L,
   0x04000802L, 0x04200002L, 0x04200002L, 0x00000002L,
   0x00200002L, 0x04000000L, 0x04000800L, 0x00200000L,
   0x04200800L, 0x00000802L, 0x00200802L, 0x04200800L,
   0x00000802L, 0x04000002L, 0x04200802L, 0x04200000L,
   0x00200800L, 0x00000000L, 0x00000002L, 0x04200802L,
   0x00000000L, 0x00200802L, 0x04200000L, 0x00000800L,
   0x04000002L, 0x04000800L, 0x00000800L, 0x00200002L };

static unsigned long SP8[64] = {
   0x10001040L, 0x00001000L, 0x00040000L, 0x10041040L,
   0x10000000L, 0x10001040L, 0x00000040L, 0x10000000L,
   0x00040040L, 0x10040000L, 0x10041040L, 0x00041000L,
   0x10041000L, 0x00041040L, 0x00001000L, 0x00000040L,
   0x10040000L, 0x10000040L, 0x10001000L, 0x00001040L,
   0x00041000L, 0x00040040L, 0x10040040L, 0x10041000L,
   0x00001040L, 0x00000000L, 0x00000000L, 0x10040040L,
   0x10000040L, 0x10001000L, 0x00041040L, 0x00040000L,
   0x00041040L, 0x00040000L, 0x10041000L, 0x00001000L,
   0x00000040L, 0x10040040L, 0x00001000L, 0x00041040L,
   0x10001000L, 0x00000040L, 0x10000040L, 0x10040000L,
   0x10040040L, 0x10000000L, 0x00040000L, 0x10001040L,
   0x00000000L, 0x10041040L, 0x00040040L, 0x10000040L,
   0x10040000L, 0x10001000L, 0x10001040L, 0x00000000L,
   0x10041040L, 0x00041000L, 0x00041000L, 0x00001040L,
   0x00001040L, 0x00040040L, 0x10000000L, 0x10041000L };

void deskey(unsigned char *key,short edf, unsigned long *kn);
void cookey(register unsigned long *raw1, unsigned long *dough);
//void cpkey(register unsigned long *into);
//void usekey(register unsigned long *from);
//void des(unsigned char *inblock,unsigned char *outblock);
void scrunch(register unsigned char *outof, register unsigned long *into);
void unscrun(register unsigned long *outof, register unsigned char *into);
void desfunc(register unsigned long *block,register unsigned long *keys);


/*****************  DES Function  *****************/
unsigned long OPENCOMM_DesExpandEncKey(
       unsigned char *pbDesKey,
       unsigned long  ulDesKeyLen,
       unsigned char *pbDesEncKey,
       unsigned long *ulDesEncKeyLen);

unsigned long OPENCOMM_DesExpandDecKey(
       unsigned char *pbDesKey,
       unsigned long  ulDesKeyLen,
       unsigned char *pbDesDecKey,
       unsigned long *ulDesDecKeyLen);

unsigned long OPENCOMM_DesEncRaw(
       unsigned char *pbDesEncKey,
       unsigned long  ulDesEncKeyLen,
       unsigned char *pbInData,
       unsigned long  ulInDataLen,
       unsigned char *pbOutData,
       unsigned long *ulOutDataLen);

unsigned long OPENCOMM_DesDecRaw(
       unsigned char *pbDesDecKey,
       unsigned long  ulDesDecKeyLen,
       unsigned char *pbInData,
       unsigned long  ulInDataLen,
       unsigned char *pbOutData,
```

```
    unsigned long *ulOutDataLen);

int myic_DESDecrypt(
    unsigned char *pDesKey,
    int           nDesKeyLen,
    unsigned char *pInData,
    int           nInDataLen,
    unsigned char *pOutData,
    int          *pOutDataLen);

int myic_DESEncrypt(
    unsigned char *pDesKey,
    int           nDesKeyLen,
    unsigned char *pInData,
    int           nInDataLen,
    unsigned char *pOutData,
    int          *pOutDataLen);

void deskey(unsigned char *key,short edf, unsigned long *kn)
{
  register int i, j, l, m, n;
  unsigned long pc1m[56],pcr[56];


  for ( j = 0; j < 56; j++ )
  {
    l = pc1[j];
    m = l & 07;
    pc1m[j] = (((unsigned long) key[l >> 3] & (unsigned long)bytebit[m] ) ? 1:0);
  }
  for ( i = 0;i < 16; i++)
  {
    if ( edf == DE1 )   m = (15 - i) << 1;
    else   m = i << 1;
    n = m + 1;
    kn[m] = kn[n] = 0L;
    for ( j = 0; j < 28; j++ )
    {
      l = j + totrot[i];
      if ( l < 28 )   pcr[j] = pc1m[l];
      else   pcr[j] = pc1m[l-28];
    }
    for (j = 28; j < 56; j++ )
    {
      l = j + totrot[i];
      if ( l < 56 )   pcr[j] = pc1m[l];
      else   pcr[j] = pc1m[l-28];
    }
    for ( j = 0; j < 24; j++ )
    {
      if ( pcr[pc2[j]] )   kn[m] |= bigbyte[j];
      if ( pcr[pc2[j+24]] )   kn[n] |= bigbyte[j];
    }
  }
  return;
}

void cookey(register unsigned long *raw1, unsigned long *dough)
{
  register unsigned long *cook,*raw0;
  register int i;

  cook = dough;
  for ( i = 0; i < 16; i++, raw1++ ) {
    raw0 = raw1++;
    *cook    = (*raw0 & 0x00fc0000L) << 6;
    *cook   |= (*raw0 & 0x00000fc0L) << 10;
    *cook   |= (*raw1 & 0x00fc0000L) >> 10;
    *cook++   |= (*raw1 & 0x00000fc0L) >> 6;
    *cook    = (*raw0 & 0x0003f000L) << 12;
    *cook   |= (*raw0 & 0x0000003fL) << 16;
    *cook   |= (*raw1 & 0x0003f000L) >> 4;
    *cook++   |= (*raw1 & 0x0000003fL);
  }
  return;
}
```

```
void scrunch(register unsigned char *outof, register unsigned long *into)
{
   *into    = (*outof++ & 0xffL) << 24;
   *into   |= (*outof++ & 0xffL) << 16;
   *into   |= (*outof++ & 0xffL) << 8;
   *into++   |= (*outof++ & 0xffL);
   *into    = (*outof++ & 0xffL) << 24;
   *into   |= (*outof++ & 0xffL) << 16;
   *into   |= (*outof++ & 0xffL) << 8;
   *into++   |= (*outof   & 0xffL);
   return;
}

void unscrun(register unsigned long *outof, register unsigned char *into)
{
   *into++    = (unsigned char)((*outof >> 24) & 0xffL);
   *into++    = (unsigned char)((*outof >> 16) & 0xffL);
   *into++    = (unsigned char)((*outof >>  8) & 0xffL);
   *into++    = (unsigned char)( *outof++      & 0xffL);
   *into++    = (unsigned char)((*outof >> 24) & 0xffL);
   *into++    = (unsigned char)((*outof >> 16) & 0xffL);
   *into++    = (unsigned char)((*outof >>  8) & 0xffL);
   *into    = (unsigned char)( *outof         & 0xffL);
   return;
}

void desfunc(register unsigned long *block,register unsigned long *keys)
{
   register unsigned long fval, work, right, leftt;
   register int round;

   leftt = block[0];
   right = block[1];
   work = ((leftt >> 4) ^ right) & 0x0f0f0f0fL;

   right ^= work;
   leftt ^= (work << 4);
   work = ((leftt >> 16) ^ right) & 0x0000ffffL;

   right ^= work;
   leftt ^= (work << 16);
   work = ((right >> 2) ^ leftt) & 0x33333333L;

   leftt ^= work;
   right ^= (work << 2);
   work = ((right >> 8) ^ leftt) & 0x00ff00ffL;

   leftt ^= work;
   right ^= (work << 8);
   right = ((right << 1) | ((right >>31) & 1L)) & 0xffffffffL;
   work = (leftt ^ right) & 0xaaaaaaaaL;

   leftt ^= work;
   right ^= work;
   leftt = ((leftt << 1) | ((leftt >> 31)&1L)) & 0xffffffffL;

   for (round = 0; round < 8; round++) {
      work  = (right << 28) | (right >> 4);
      work ^= *keys++;
      fval  = SP7[ work    & 0x3fL];
      fval |= SP5[(work >>  8) & 0x3fL];
      fval |= SP3[(work >> 16) & 0x3fL];
      fval |= SP1[(work >> 24) & 0x3fL];
      work  = right ^ *keys++;
      fval |= SP8[ work     & 0x3fL];
      fval |= SP6[(work >>  8) & 0x3fL];
      fval |= SP4[(work >> 16) & 0x3fL];
      fval |= SP2[(work >> 24) & 0x3fL];
      leftt ^= fval;
      work  = (leftt << 28) | (leftt >> 4);
      work ^= *keys++;
      fval  = SP7[ work     & 0x3fL];
      fval |= SP5[(work >>  8) & 0x3fL];
      fval |= SP3[(work >> 16) & 0x3fL];
      fval |= SP1[(work >> 24) & 0x3fL];
      work  = leftt ^ *keys++;
      fval |= SP8[ work     & 0x3fL];
```

```
        fval |= SP6[(work >>  8) & 0x3fL];
        fval |= SP4[(work >> 16) & 0x3fL];
        fval |= SP2[(work >> 24) & 0x3fL];
        right ^= fval;
    }

    right = (right << 31) | (right >> 1);
    work = (leftt ^ right) & 0xaaaaaaaaL;
    leftt ^= work;
    right ^= work;
    leftt = (leftt << 31) | (leftt >> 1);
    work = ((leftt >>  8) ^ right) & 0x00ff00ffL;
    right ^= work;
    leftt ^= (work << 8);
    work = ((leftt >>  2) ^ right) & 0x33333333L;
    right ^= work;
    leftt ^= (work << 2);
    work = ((right >> 16) ^ leftt) & 0x0000ffffL;
    leftt ^= work;
    right ^= (work << 16);
    work = ((right >>  4) ^ leftt) & 0x0f0f0f0fL;
    leftt ^= work;
    right ^= (work << 4);
    *block++ = right;
    *block = leftt;
    return;
}

/*****************************************************************
    OPENCOMM_DesExpandEncKey    : Expand Des Enc Key 扩展des加密密钥
    Return value:
        0         : Success
        other     : failed
    Parameters:
        pbDesKey       : 扩展前的DES密钥(8字节)       input
        ulDesKeyLen    : 扩展前的DES密钥长度          input
        pbDesEncKey    : 扩展后的DES加密密钥(128字节)  output
        *ulDesEncKeyLen : 扩展后的DES加密密钥长度       output
*****************************************************************/
unsigned long OPENCOMM_DesExpandEncKey(
        unsigned char *pbDesKey,
        unsigned long  ulDesKeyLen,
        unsigned char *pbDesEncKey,
        unsigned long *ulDesEncKeyLen)
{
    unsigned long kn[32], dough[32];

    if (ulDesKeyLen != 8)
        return 0xEE20;

    deskey(pbDesKey, EN0, kn);
    cookey(kn, dough);
    *ulDesEncKeyLen = DES_KEYBYTES;  //32 long = 128 bytes
    memcpy(pbDesEncKey, dough, *ulDesEncKeyLen);

    return 0;
}

/*****************************************************************
    OPENCOMM_DesExpandDecKey    : Expand Des Dec Key 扩展des解密密钥
    Return value:
        0       : Success
        other   : failed
    Parameters:
        pbDesKey       : 扩展前的DES密钥(8字节)      input
        ulDesKeyLen    : 扩展前的DES密钥长度         input
        pbDesDecKey    : 扩展后的DES解密密钥(128字节) output
        *ulDesDecKeyLen : 扩展后的DES解密密钥长度      output
*****************************************************************/
unsigned long OPENCOMM_DesExpandDecKey(
        unsigned char *pbDesKey,
        unsigned long  ulDesKeyLen,
        unsigned char *pbDesDecKey,
        unsigned long *ulDesDecKeyLen)
{
    unsigned long kn[32], dough[32];
```

```
    if (ulDesKeyLen != 8)
        return 0xEE20;

    deskey(pbDesKey, DE1, kn);
    cookey(kn, dough);
    *ulDesDecKeyLen = DES_KEYBYTES;  //32 long = 128 bytes
    memcpy(pbDesDecKey, dough, *ulDesDecKeyLen);

    return 0;
}

/*****************************************************************
    OPENCOMM_DesEncRaw       : Des算法加密小整块明文8字节
    Return value:
        0       : Success
        other   : failed
    Parameters:
        pbDesEncKey    : DES加密密钥    input
        ulDesEncKeyLen : DES加密密钥长度 input
        pbInData       : 待加密的明文    input
        ulInDataLen    : 待加密的明文长度 input
        pbOutData      : 加密后的密文    output
        *ulOutDataLen  : 加密后的密文长度 output
*****************************************************************/
unsigned long OPENCOMM_DesEncRaw(
        unsigned char *pbDesEncKey,
        unsigned long  ulDesEncKeyLen,
        unsigned char *pbInData,
        unsigned long  ulInDataLen,
        unsigned char *pbOutData,
        unsigned long *ulOutDataLen)
{
    unsigned long work[2], ek[DES_KEYLONGS];
    unsigned char cp[DES_BLOCKLEN];


    if (ulInDataLen != DES_BLOCKLEN)
        return 0xEE20;

    if (ulDesEncKeyLen != DES_KEYBYTES)
        return 0xEE20;

    memcpy(cp, pbInData, DES_BLOCKLEN);
    scrunch(cp,work);  // 8 bytes -> 2 long
    memcpy(ek, pbDesEncKey, ulDesEncKeyLen);
    desfunc(work,ek);
    unscrun(work,cp); // 2 long -> 8 bytes
    memcpy(pbOutData, cp, DES_BLOCKLEN);
    *ulOutDataLen = DES_BLOCKLEN;

    return 0;
}


/*****************************************************************
    OPENCOMM_DesDecRaw : Des算法解密小整块密文8字节
    Return value:
        0    : Success
        other : failed
    Parameters:
        pbDesDecKey    : DES解密密钥     input
        ulDesDecKeyLen : DES解密密钥长度  input
        pbInData       : 待解密的密文     input
        ulInDataLen    : 待解密的密文长度  input
        pbOutData      : 解密后的明文     output
        *ulOutDataLen  : 解密后的明文长度  output
*****************************************************************/
unsigned long OPENCOMM_DesDecRaw(
        unsigned char *pbDesDecKey,
        unsigned long  ulDesDecKeyLen,
        unsigned char *pbInData,
        unsigned long  ulInDataLen,
        unsigned char *pbOutData,
        unsigned long *ulOutDataLen)
{
    unsigned long work[2], dk[DES_KEYLONGS];
    unsigned char cp[DES_BLOCKLEN];
```

```
    if (ulInDataLen != DES_BLOCKLEN)
        return 0xEE20;

    if (ulDesDecKeyLen != DES_KEYBYTES)
        return 0xEE20;

    memcpy(cp, pbInData, DES_BLOCKLEN);
    scrunch(cp,work);  // 8 bytes -> 2 long
    memcpy(dk, pbDesDecKey, ulDesDecKeyLen);
    desfunc(work,dk);
    unscrun(work,cp); // 2 long -> 8 bytes
    memcpy(pbOutData, cp, DES_BLOCKLEN);
//    des_enc(pbDesEncKey, pbInData, pbOutData);
    *ulOutDataLen = DES_BLOCKLEN;

    return 0;
}

/*********************   DES    *********************/
int myic_DESEncrypt(
        unsigned char *pDesKey,
        int           nDesKeyLen,
        unsigned char *pInData,
        int           nInDataLen,
        unsigned char *pOutData,
        int          *pOutDataLen)
{
    unsigned char DesKeyBuf[32];
    unsigned char DesEncKeyBuf[128];
    int EncKeyLen, KeyLen = 0;
    int retval = 0, loops, i;

    if(nInDataLen%8 != 0)
        return 0xEE20;

    if(nDesKeyLen != 8)
        return 0xEE20;
    KeyLen = nDesKeyLen;
    memcpy(DesKeyBuf, pDesKey, nDesKeyLen);

    retval = OPENCOMM_DesExpandEncKey(DesKeyBuf, KeyLen,
        DesEncKeyBuf, (unsigned long *)&EncKeyLen);
    if(retval != 0)
        return retval;


    loops = nInDataLen/8;
    for(i = 0; i < loops; i++)
    {
        retval = OPENCOMM_DesEncRaw(DesEncKeyBuf, EncKeyLen, pInData + i*8,
            8, pOutData + i*8, (unsigned long *)pOutDataLen);
        if(retval != 0)
            return retval;
    }
    *pOutDataLen = nInDataLen;
    return retval;
}

int myic_DESDecrypt(
        unsigned char *pDesKey,
        int           nDesKeyLen,
        unsigned char *pInData,
        int           nInDataLen,
        unsigned char *pOutData,
        int          *pOutDataLen)
{
    unsigned char DesKeyBuf[32];
    unsigned char DesDecKeyBuf[128];
    int DecKeyLen, KeyLen = 0;
    int retval = 0, loops, i;

    if(nInDataLen%8 != 0)
        return 0xEE20;

    if(nDesKeyLen != 8)
        return 0xEE20;
```

```
    KeyLen = nDesKeyLen;
    memcpy(DesKeyBuf, pDesKey, nDesKeyLen);

    retval = OPENCOMM_DesExpandDecKey(DesKeyBuf, KeyLen,
        DesDecKeyBuf, (unsigned long *)&DecKeyLen);
    if(retval != 0)
        return retval;

    loops = nInDataLen/8;
    for(i = 0; i < loops; i++)
    {
        retval = OPENCOMM_DesDecRaw(DesDecKeyBuf, DecKeyLen, pInData + i*8,
            8, pOutData + i*8, (unsigned long *)pOutDataLen);
        if(retval != 0)
            return retval;
    }
    *pOutDataLen = nInDataLen;
    return retval;
}

//对称明文数据打pading
void  CW_dataPadAdd(int tag, unsigned char *date, unsigned int dateLen,
                    unsigned char **padDate, unsigned int *padDateLen)
{
    int          i, padLen;
    unsigned char *pTmp   = NULL;

    pTmp = (unsigned char *)malloc(dateLen+24);
    if (pTmp == NULL)
    {
        *padDate = NULL;
        return ;
    }
    memset(pTmp, 0, dateLen+24);
    memcpy(pTmp, date, dateLen);

    if (tag == 0)
    {
        padLen = 8 - dateLen % 8;
        for (i=0; i<padLen; i++)
        {
            pTmp[dateLen+i] = (char)padLen;
        }
        *padDateLen = dateLen + padLen;
    }
    else
    {
        padLen = 16 - dateLen % 16;
        for (i=0; i<padLen; i++)
        {
            pTmp[dateLen+i] = (char)padLen;
        }
    }

    *padDateLen = dateLen + padLen;
    *padDate = pTmp;
}

#define  USER_PASSWORD_KEY "abcd1234"

//数据加密
int DesEnc(
        unsigned char *pInData,
        int           nInDataLen,
        unsigned char *pOutData,
        int          *pOutDataLen)
{
    int              rv;
    unsigned char    *padDate = NULL;
    unsigned int    padDateLen = 0;

    CW_dataPadAdd(0, pInData, (unsigned int )nInDataLen, &padDate, &padDateLen);

    rv = myic_DESEncrypt((unsigned char *)USER_PASSWORD_KEY, strlen(USER_PASSWORD_KEY),
        padDate, (int)padDateLen, pOutData, pOutDataLen);
    if (rv != 0)
    {
```

```
    if (padDate != NULL)
    {
        free(padDate);
    }
    return rv;
  }

  if (padDate != NULL)
  {
    free(padDate);
  }
  return 0;
}

//数据加密
int DesEnc_raw(
  unsigned char *pInData,
  int           nInDataLen,
  unsigned char *pOutData,
  int          *pOutDataLen)
{
  int              rv;
  unsigned char    *padDate = NULL;
  unsigned int    padDateLen = 0;

  rv = myic_DESEncrypt((unsigned char *)USER_PASSWORD_KEY, strlen(USER_PASSWORD_KEY),
    pInData, (int)nInDataLen, pOutData, pOutDataLen);
  if (rv != 0)
  {
    return rv;
  }
  return 0;
}

//解密分配内存错误
#define  ERR_MALLOC 20
//密码长度不是8的整数倍, 不合法
#define  ERR_FILECONT 20

//用户使用函数des解密
int DesDec(
        unsigned char *pInData,
        int           nInDataLen,
        unsigned char *pOutData,
        int          *pOutDataLen)
{
  int              rv;
  char           padChar;
  unsigned char     *tmpPlain = NULL;

  tmpPlain =       (unsigned char *)malloc(nInDataLen+24);
  if (tmpPlain == NULL)
  {
    return ERR_MALLOC;
  }
  memset(tmpPlain, 0, nInDataLen+24);


  //解密
  rv = myic_DESDecrypt((unsigned char *)USER_PASSWORD_KEY, strlen(USER_PASSWORD_KEY),
    pInData, nInDataLen, tmpPlain, pOutDataLen);
  if (rv != 0)
  {
    if (tmpPlain != NULL) free(tmpPlain);
    return rv;
  }


  //去pading
  padChar = tmpPlain[*pOutDataLen - 1];
  if ( (int)padChar<=0 || (int)padChar>8) //异常处理
  {
    if (tmpPlain) free(tmpPlain);
    return ERR_FILECONT;
  }

  *pOutDataLen = *pOutDataLen - (int)padChar;
```

```
    //memset(tmpPlain + *pOutDataLen, 0, (int)padChar);
    memcpy(pOutData, tmpPlain, *pOutDataLen);
    if (tmpPlain) free(tmpPlain);
    return 0;
}

//用户使用函数des解密
int DesDec_raw(
    unsigned char *pInData,
    int            nInDataLen,
    unsigned char *pOutData,
    int           *pOutDataLen)
{
    int               rv;
    //char            padChar;
    //unsigned char     *tmpPlain = NULL;

    /*
    tmpPlain =        (unsigned char *)malloc(nInDataLen+24);
    if (tmpPlain == NULL)
    {
        return ERR_MALLOC;
    }
    memset(tmpPlain, 0, nInDataLen+24);
    */

    //解密
    rv = myic_DESDecrypt((unsigned char *)USER_PASSWORD_KEY, strlen(USER_PASSWORD_KEY),
        pInData, nInDataLen, pOutData, pOutDataLen);
    if (rv != 0)
    {
        //if (tmpPlain != NULL) free(tmpPlain);
        return rv;
    }
    /*
    //去pading
    padChar = tmpPlain[*pOutDataLen - 1];
    if ( (int)padChar<=0 || (int)padChar>8) //异常处理
    {
        if (tmpPlain) free(tmpPlain);
        return ERR_FILECONT;
    }

    *pOutDataLen = *pOutDataLen - (int)padChar;
    //memset(tmpPlain + *pOutDataLen, 0, (int)padChar);
    memcpy(pOutData, tmpPlain, *pOutDataLen);
    if (tmpPlain) free(tmpPlain);
    */
    return 0;
}
```

- 文件加解密框架.c

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "des.h"

int FileSymEnc(const char *pfile1, const char *pfile2)
{
    int       ret = 0;
    FILE *fp1= NULL, *fp2 = NULL;
    unsigned char plain[4096];
    int plainlen = 0;

    unsigned char cryptbuf[4096] = {0};
    int cryptlen = 0;

    int tmplen;

    fp1 = fopen(pfile1, "rb");
    if (fp1 == NULL)
    {
```

```
        goto END;
    }

    fp2 = fopen(pfile2, "wb");
    if (fp2 == NULL)
    {
        goto END;
    }

    while (!feof(fp1))
    {
        plainlen = fread(plain, 1, 4096, fp1);
        if (feof(fp1))  //读完数据以后，判断是否文件结束
        {
            break;
        }

        //加密==4k的数据
        ret =  DesDec_raw(plain, plainlen, cryptbuf, &cryptlen);
        if (ret != 0)
        {
            printf("func DesEnc() err:%d \n", ret);
            goto END;
        }

        tmplen = fwrite(cryptbuf, 1, cryptlen, fp2);
        if (tmplen != cryptlen)
        {
            ret  = -3;
            printf("写密文文件失败，请检查是否磁盘已满\n");
            goto END;
        }

        //if (plainlen == 4096)
    }

    //加密小于4k的数据
    ret =  DesEnc(plain, plainlen, cryptbuf, &cryptlen);
    if (ret != 0)
    {
        printf("func DesEnc() err:%d \n", ret);
        goto END;
    }

    tmplen = fwrite(cryptbuf, 1, cryptlen, fp2);
    if (cryptlen != tmplen)
    {
        ret  = -3;
        printf("写小于4k文件密文失败，请检查是否磁盘已满\n");
        goto END;
    }

END:
    if (fp1 != NULL)
    {
        fclose(fp1);
    }
    if (fp2 != NULL)
    {
        fclose(fp2);
    }
    return 0;
}

void main()
{
    int       ret = 0;
    const char *file1 = "c:/socketclient.dll";
    const char *file2 = "c:/socketclientend.dll";

    //const char *file1 = "c:/22.txt";
    //const char *file2 = "c:/22enc.txt";

    ret =  FileSymEnc(file1, file2);
    if (ret != 0)
    {
        printf("func FileSymEnc() err\n " );
```

```
        return ;
    }

    system("pause");
    return ;
}
```

- **二级指针做输入三种内存模型综合考试**

```
#define _CRT_SECURE_NO_WARNINGS

#include "stdio.h"
#include "stdlib.h"
#include "string.h"

int getArray3_Free(char ***p3, int p3num)
{
    int i;
    char **tmp = NULL;
    if (p3 == NULL)
    {
        return -1;
    }
    tmp = *p3;
    for (i = 0; i < p3num; i++)
    {
        if (tmp[i] != NULL)
        {
            free(tmp[i]);
        }
    }
    free(tmp);
    *p3 = NULL; //通过间接赋值，去间接地修改实参的值
    return 0;
}

int getArray3(char **myp1, int num1, char(*myp2)[30], int num2, char ***myp3, int *num3)
{
    int ret = 0;
    int i, j;
    int tmpNum3 = 0;
    char *tmp;
    char *tmpbuf[100];
    char **tmpp3 = NULL;
    if (myp1 == NULL || myp2 == NULL || num3 == NULL || myp3==NULL)
    {
        ret = -1;
        return ret;
    }
    //准备内存
    tmpNum3 = num1 + num2;
    //分配第一维
    tmpp3 = (char **)malloc(tmpNum3 * sizeof(char *));
    if (tmpNum3 == NULL)
    {
        ret = -2;
        return ret;
    }
    //分配第二维
    for (i = 0; i < num1; i++)
    {
        //printf("%d\n", strlen(myp1[i]));
        tmpp3[i] = (char*)malloc(strlen(myp1[i]) + 1);
        if (tmpp3[i] == NULL)
        {
            ret = -3;
            return ret;
        }
    }
    for (i = 0; i < num2; i++)
    {
        //printf("%d\n", strlen(myp2[i]));
        tmpp3[i+num1] = (char*)malloc(strlen(myp2[i]) + 1);
        if (tmpp3[i + num1] == NULL)
```

```
        {
            ret = -4;
            return ret;
        }
    }
    //把第一种内存模型数据和第二种内存模型数据，copy到第3中内存模型中
    for (i = 0; i < num1; i++)
    {
        //printf("%d\n", strlen(myp1[i]));
        strcpy(tmpp3[i], myp1[i]);
    }
    for (i = 0; i < num2; i++)
    {
        //printf("%d\n", strlen(myp2[i]));
        strcpy(tmpp3[i + num1], myp2[i]);
    }

    //选择排序实现，比较顺序：01,02,03,04;12,13,14;
    for (i = 0; i < tmpNum3; i++) //最后一个数不用排，比如54321，第一轮结束
    //变为43215，最大的已经到最后了
    {
        for (j = i+1; j < tmpNum3; j++)
        {
            if (strcmp(tmpp3[i], tmpp3[j]) > 0)
            {
                //错误：使用strcpy拷贝字符串因分配空间不足导致漏洞！free时会宕掉
                /*strcpy(tmpbuf, tmpp3[i]);
                strcpy(tmpp3[i], tmpp3[j]);
                strcpy(tmpp3[j], tmpbuf);*/
                //正确：利用[]相当于对二级指针tmpp3解引用，从而修改一级指针，
                //      而不是修改一级指针所指的内存空间！
                tmp = tmpp3[i];
                tmpp3[i] = tmpp3[j];
                tmpp3[j] = tmp;
            }
        }
    }
    *num3 = tmpNum3;
    *myp3 = tmpp3;
    return 0;
}

void main()
{
    int ret = 0;
    int num3 = 0, i = 0;
    char *p1[] = { "222222", "1111111", "33333333" };
    char p2[4][30] = { "bbbbb", "aaaaa", "zzzzzz", "ccccccc" };
    char **p3 = NULL;
    ret = getArray3(p1, 3, p2, 4, &p3, &num3);
    if (ret != 0)
    {
        return ret;
    }
    for (i = 0; i < num3; i++)
    {
        printf("%s \n", p3[i]);
    }
    getArray3_Free(&p3, num3);
    printf("p3:%d \n", p3);
    system("pause");
}
```