实验二　DES 的 CBC 模式

专业：密码科学与技术 姓名：柳致远　学号：2113683

一、 实验要求：

DES 加密/解密 在本项目中，我们将在 CBC（密码分组链接）模式下使用 DES 对文件进行加 密和解密。“tempdes.c”是用于加密/解密固定的 64 位块的核心文件。在本作业 中，您需要通过实现 DES-CBC 操作模式，扩展此框架代码以获取任意大小的输入 文件并对其进行加密/解密。实际上，您必须实现 CBC 模式，并且除了 tempdes.c 中的功能外，不允许使用任何内置功能。您可以在您的课本中找到有关 DES-CBC 的信息。 可以通过对照输入文件“test.txt”检查您的工作。如果您正确实现了算法， 则应该在“test.des”中获得输出。

二、 实验过程

首先分析分析该程序实现的大致过程，并由此书写主程序

```
int main(int argc, char** argv)
{
    if(argc != 5) {
        printf("USAGE ERROR \nusage: ./exec_file IV key input_file out_file\n");
         /*
         *   param:
         * ./cbcdes iv key inputfile outputfile
         *   iv initial vector 初始参数,16个16进制字符
         *   key 初始密钥，16个16进制字符
         *   inputfile 输入文件，即明文
         *   outputfile 输出文件，即密文
         */
    }
    else {
        const_DES_cblock cbc_key ;
        const_DES_cblock IV ;
```

```
int k;
 if(checkhex(argv[1])!=0)
      {printf("请输入16个16进制数作为初始向量IV！\n");return 0;}
 if(checkhex(argv[2])!=0)
      {printf("请输入16个16进制数作为初始密钥key！\n");return 0;}

strToHex((unsigned char*)argv[1], IV);//将命令行参数字符串转换为十六进制格式
strToHex((unsigned char*)argv[2], cbc_key);
printvalueOfDesBlock(IV);//打印初始向量
printvalueOfDesBlock(cbc_key);

if ((k = DES_set_key_checked(&cbc_key,&key)) != 0)
      {printf("\n生成密钥不符合要求！\n");return 0;}


 //对明文进行加密
FILE *inpFile = fopen(argv[3], "rb");
FILE *outFile = fopen(argv[4], "wb");
if(inpFile && outFile) {
     printf("加密文件创建成功！\n");
     CBCenc(inpFile,outFile,IV);
} else {
    printf("打开文件失败！\n");
}
fclose(inpFile);
fclose(outFile);

 //对密文进行解密
 FILE *incFile = fopen(argv[4], "rb");
 FILE *decFile = fopen("decfile.txt", "wb");
 if(incFile && decFile) {
     printf("解密文件创建成功！\n");
     CBCdec(incFile,decFile,IV);
} else {
    printf("打开文件失败！\n");
}
 fclose(incFile);
 fclose(decFile);
 /*
 *按字节读取文件
 FILE *mFile = fopen("decfile.txt", "rb");
 unsigned char ch[1];
 int read1 = fread(ch, 1, 1, mFile); // 读取8字节的密文
 while(read1 > 0){ // 当还有密文没有读取完的时候
```

```
            printf("%c ", ch[0]);
            ch[0] = '\0';
            read1 = fread(ch, 1, 1, mFile); // 读取下一个8字节块
        }
        fclose(mFile);
        */
        printf("\n");
    }
    return 0;
    }
```

对主函数出现的各函数进行定义及补充

```
// 无符号字符串转无符号16进制字符
void strToHex(const_DES_cblock input, unsigned char *output) {
    unsigned int byte;
    for(int i=0; i<8; i++) {
        if(sscanf((const char*)(char*)input, "%2x", &byte) != 1) {
            break;
        }
        output[i] = byte;
        input += 2;
    }
}
```

1. 首先定义一个无符号整型变量 **byte**，用于存储每次转换的十六进制数。
2. 使用 **for** 循环遍历输入的无符号字符串，循环次数为 8 次。
3. 在循环中，使用 **sscanf** 函数从输入的无符号字符串中读取两个字符（两个字符表示一个十六进制数），并将其转换为十六进制数存储到 **byte** 变量中。
4. 如果 **sscanf** 函数的返回值不等于 1，表示转换失败，可能是输入字符串格式不正确，此时退出循环。
5. 将转换后的十六进制数存储到输出数组 **output** 中，并更新输入字符串的指针，使其指向下一个十六进制数的位置。

```
// 把一个无符号字符串复制到另一个字符串
void copyValue(const_DES_cblock val1, unsigned char *val2, int size) {
    for(int i=0; i<size; i++) {
        val2[i] = val1[i];
    }
}
```

1. 使用 **for** 循环遍历源字符串 **val1**。
2. 在循环中，将源字符串 **val1** 的每个字符复制到目标字符串 **val2** 的对应位置。
3. 循环结束后，完成字符串的复制。

```
// 用两个无符号长字符（4字节），对上次加密结果和这次的明文data进行异或
void LongXor(DES_LONG *xor, DES_LONG* data, const_DES_cblock iv) {
    DES_LONG temp[2];
    memcpy(temp, iv, 8*sizeof(unsigned char));   // 转换成相同的类型
```

```
    for(int i=0; i<2; i++) {
        xor[i] = temp[i] ^ data[i];
    }
}
```

1. 首先定义一个临时的 **DES_LONG** 类型的数组 **temp**，用于存储初始向量 **iv** 的副本。
2. 使用 **memcpy** 函数将初始向量 **iv** 的值复制到临时数组 **temp** 中。
3. 使用 **for** 循环遍历数组，对两个数组进行逐个元素的异或操作，并将结果存储到 **xor** 数组中。
4. 循环结束后，完成数组的异或操作。

```
// 打印一个8字节的des块
void printvalueOfDesBlock(const_DES_cblock val) {
    for(int i=0; i<7; i++) {
        printf("0x%x,", val[i]);
    }
    printf("0x%x", val[7]);
    printf("\n");
}
```

- 使用 **for** 循环遍历 DES 块中的每个字节，并使用 **printf** 函数按十六进制格式打印出每个字节的值。
- 在循环结束后，输出换行符

```
int checkhex(char* ch)
{
    if (strlen(ch)!= 16)
        return -1;
    for (int i=0;i<16;i++){
        if(!(((ch[i]>='0')&&(ch[i]<='9'))
            ||((ch[i]>='A')&&(ch[i]<='F'))
            ||((ch[i]>='a')&&(ch[i]<='f'))))
        return -1;
    }
    return 0;
}
```

- 首先检查输入字符串的长度是否为 16，如果不是，则返回 -1，表示输入的字符串不符合要求。
- 然后使用 **for** 循环遍历字符串中的每个字符，检查是否为十六进制字符（即 0-9、A-F、a-f 之一），如果不是，则返回 -1，表示输入的字符串不符合要求。
- 如果以上两个条件都满足，则返回 0，表示输入的字符串是一个合法的十六进制数。

```
void CBCenc(FILE *inpFile,FILE *outFile,const_DES_cblock IV){
    const_DES_cblock iv ;
    copyValue(IV,iv,sizeof(const_DES_cblock));
    DES_LONG data[2] = {0,0},temp[2] = {0,0}; // data用来储存每次读取的8字节64比特的数据，temp用来储存加密后的数据
```

```
    int mRead = fread(data, 1, 8, inpFile); // 从明文中读取8字节的数据
    while(mRead > 0){ // 当能够从明文中读到数据的时候
        LongXor(temp, data, iv); // 先将数据和iv异或
        DES_encrypt1(temp,&key,ENC); // 异或的结果进行加密
        fwrite(temp, 8, 1, outFile); // 将加密的结果写到密文中
        memcpy(iv, temp, 2*sizeof(DES_LONG)); // 将加密的结果作为下一次的iv进行异或
        data[0]=0;data[1]=0; // 用0填充data
        mRead = fread(data, 1, 8, inpFile);
    }
    printf("加密完成\n"); //加密完成
}
```

1. 首先，将初始向量（Initialization Vector，IV）复制到一个临时变量 **iv** 中，以便在加密过程中使用。这是为了不改变输入 IV 的值，以便后续可能需要使用它。
2. 初始化两个数组 **data** 和 **temp**，它们分别用于存储每次从输入文件中读取的明文数据和加密后的数据。**data** 用来储存每次读取的 8 字节（64 比特）的数据，**temp** 用来储存加密后的数据。
3. 使用 **fread** 函数从输入文件中读取 8 字节的数据到数组 **data** 中，这个操作相当于读取一个数据块的大小。
4. 进入一个 **while** 循环，当从输入文件中读取到的数据大小大于 0 时，执行循环体内的操作。这样可以确保每次读取到数据时都会进行加密处理。
5. 在循环体内，首先对读取到的明文数据 **data** 和初始向量 **iv** 进行异或操作，将结果存储到临时数组 **temp** 中。这一步是 CBC 模式中的关键步骤，通过与前一个密文块异或来增加密码的随机性和安全性。
6. 使用 DES 加密算法对异或结果 **temp** 进行加密，加密的密钥为 **key**，加密模式为 **ENC**（加密模式）。
7. 使用 **fwrite** 函数将加密后的数据 **temp** 写入输出文件中，这样就生成了密文。
8. 将加密后的数据 **temp** 复制到初始向量 **iv** 中，以便作为下一次加密操作的初始向量。
9. 清空 **data** 数组，将其元素置为 0，以便下一次循环使用。
10. 重新使用 **fread** 函数从输入文件中读取数据，判断是否读取成功，如果成功，则继续循环；否则退出循环。
11. 最后，输出一条信息表示加密完成。

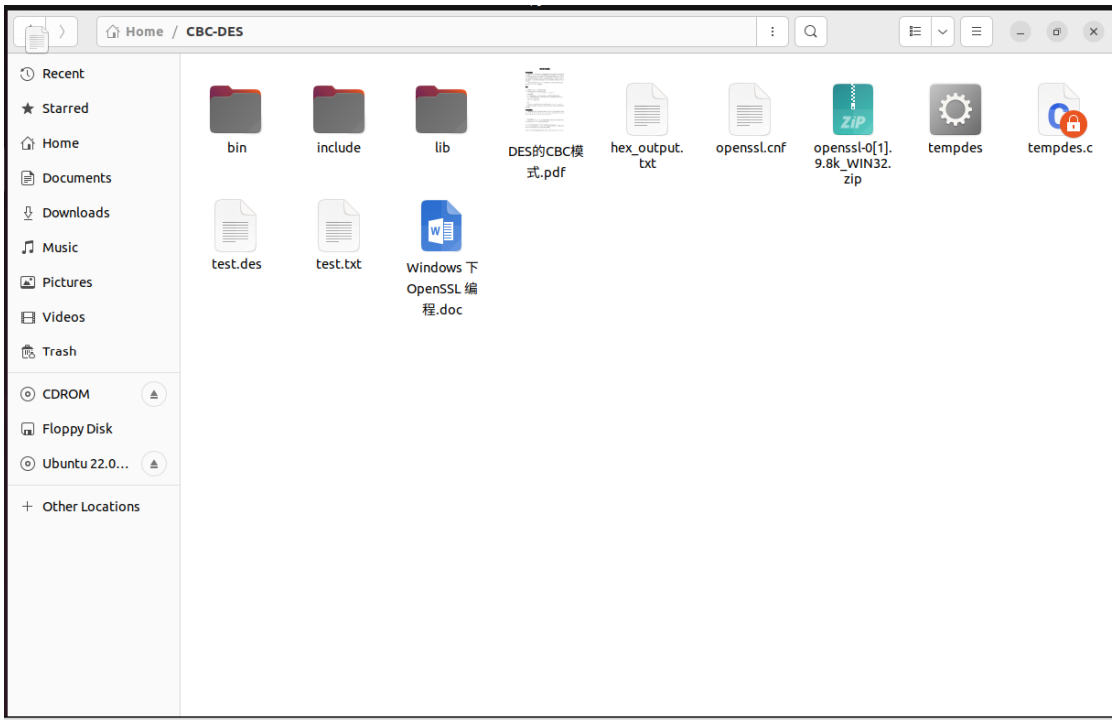
```
void CBCdec(FILE *inpFile,FILE *outFile,const_DES_cblock IV){
    const_DES_cblock iv ;
    copyValue(IV,iv,sizeof(const_DES_cblock));
    DES_LONG data[2] = {0,0}; //data为读取的密文，temp1用来储存下一步的iv，temp2用来储存解密后的结果
    int cRead = fread(data, 1, 8, inpFile); // 读取8字节的密文
    while(cRead > 0){ // 当还有密文没有读取完的时候
        DES_LONG temp1[2],temp2[2];
        memcpy(temp1, data, 2*sizeof(DES_LONG)); // 将本轮的密文作为下一次的iv进行异或
        DES_encrypt1(data,&key,DEC); // 将密文解密
```

```
        LongXor(temp2, data, iv); // 解密后的再与iv异或一次得到明文
        fwrite(temp2, 8, 1, outFile); // 将得到的明文写到文件中
        memcpy(iv, temp1, 2*sizeof(DES_LONG)); // 把这一轮的密文作为下一轮的iv
        data[0]=0;data[1]=0; // 用来进行填充0
        cRead = fread(data, 1, 8, inpFile); // 读取下一个8字节块
    }
    printf("解密完成\n");
}
```

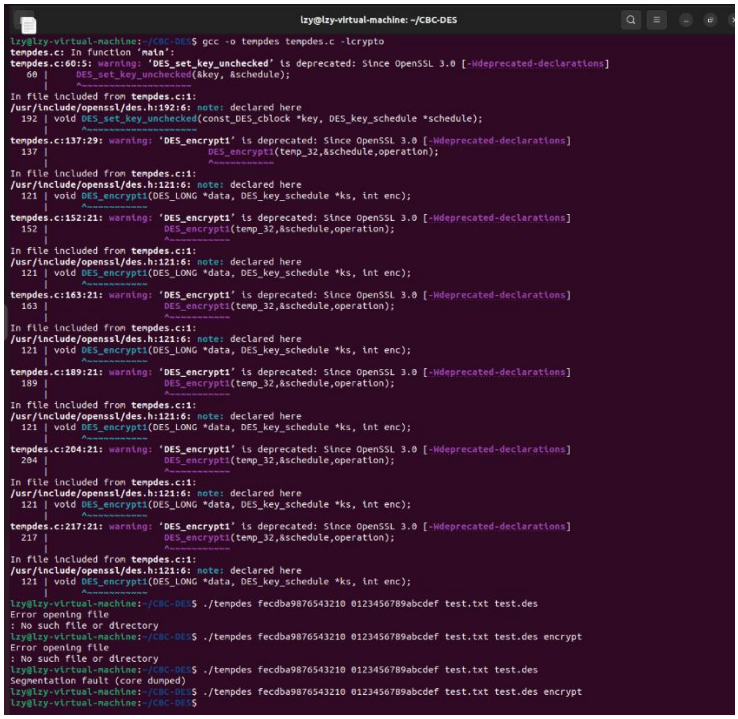
1. 将初始向量（Initialization Vector，IV）复制到一个临时变量 **iv** 中，以便在解密过程中使用。这是为了不改变输入 IV 的值，以便后续可能需要使用它。
2. 初始化数组 **data**，用于存储每次从输入文件中读取的密文数据。
3. 使用 **fread** 函数从输入文件中读取 8 字节的密文数据到数组 **data** 中，这相当于读取一个密文数据块的大小。
4. 进入一个 **while** 循环，当从输入文件中读取到的数据大小大于 0 时，执行循环体内的操作。这样可以确保每次读取到数据时都会进行解密处理。
5. 在循环体内，首先将本轮的密文 **data** 复制到临时数组 **temp1** 中，以备后续作为下一次加密操作的初始向量。
6. 使用 DES 解密算法对读取到的密文数据 **data** 进行解密，加密密钥为 **key**，解密模式为 **DEC**（解密模式）。
7. 对解密后的密文数据 **data** 和初始向量 **iv** 进行异或操作，将结果存储到临时数组 **temp2** 中。这一步是 CBC 模式中的关键步骤，用于恢复出原始的明文数据。
8. 使用 **fwrite** 函数将解密后的明文数据 **temp2** 写入输出文件中，这样就完成了解密过程。
9. 将本轮的密文 **temp1** 复制到初始向量 **iv** 中，以备下一次解密操作的初始向量。
10. 清空 **data** 数组，将其元素置为 0，以便下一次循环使用。
11. 重新使用 **fread** 函数从输入文件中读取数据，判断是否读取成功，如果成功，则继续循环；否则退出循环。
12. 最后，输出一条信息表示解密完成

三、 实验结果

在 linux 环境下输入命令 gcc -o tempdes tempdes.c -lcrypto 得到可执行文件



后运行命令./tempdes fecdba9876543210 0123456789abcdef test.txt test.des encrypt 对 test.txt 文件执行加密操作，并得到 text.des 文件



运行./tempdes fecdba9876543210 0123456789abcdef test.des test.txt decrypt进行解密操作获得初始明文文件