

网络安全技术实验报告

Lab3 基于 MD5 算法的文件完整性校验程序
网络空间安全学院 信息安全专业
2112492 刘修铭 1027

1 实验要求

实现基于 MD5 算法的文件完整性校验程序，将“实验报告、源代码、可执行程序”打包后上传，并以自己的“学号-姓名”命名。

2 实验目标

1. 深入理解 MD5 算法的基本原理。
2. 掌握利用 MD5 算法生成数据摘要的所有计算过程。
3. 掌握 Linux 系统中检测文件完整性的基本方法。
4. 熟悉 Linux 系统中文件的基本操作方法

3 实验内容

1. 准确地实现 MD5 算法的完整计算过程。
2. 对于任意长度的字符串能够生成 128 位 MD5 摘要。
3. 对于任意大小的文件能够生成 128 位 MD5 摘要。
4. 通过检查 MD5 摘要的正确性来检验原文件的完整性。

4 实验步骤

4.1 MD5 类的设计与实现

按照实验指导手册说明，并基于个人编程实现，给出如下 MD5 类的实现，各个成员的含义已在注释中给出。

```
1  class MD5
2  {
3  public:
4      MD5();
5      MD5(const string &str);
6      MD5(istream &in);
7      void Update(const BYTE *input, size_t length); // 对给定长度的输入流进行 MD5 运算
8      void Update(const string &str);                // 对给定长度的字符串进行 MD5 运算
9      void Update(istream &in);                      // 对文件中的内容进行 MD5 运算
```

```

10     const BYTE *GetDigest();                // 将 MD5 摘要以字节流形式输出
11     string ToString();                      // 将 MD5 摘要以字符串形式输出
12     void Reset();                          // 重置初始变量
13
14 private:
15     void Stop();                           // 用于终止摘要计算过程，输出
摘要
16     void Transform(const BYTE block[64]);   // 对消息分组进行 MD5 计算
17     void Encode(const DWORD *input, BYTE *output, size_t length); // 将双字流转换为字节流
18     void Decode(const BYTE *input, DWORD *output, size_t length); // 将字节流转换为双字流
19     string BytesToHexString(const BYTE *input, size_t length);    // 将字节流按照十六进制字符串
形式输出
20
21     DWORD state[4];                        // 用于表示 4 个初始向量
22     DWORD count[2];                        // 用于计数，count[0] 表示低位，count[1] 表示高位
23     BYTE buffer_block[64];                // 用于保存计算过程中按块划分后剩下的比特流
24     BYTE digest[16];                      // 用于保存 128 比特长度的摘要
25     bool is_finished;                     // 用于标志摘要计算过程是否结束
26
27     static const BYTE padding[64];        // 用于保存消息后面填充的数据块
28     static const char hex[16];           // 用于保存 16 进制的字符
29 };

```

下面将分别说明各个函数的实现思路。

参考实验指导手册，设置了如下的宏定义及常量，用于 MD5 算法实现。

```

1  #define F(x, y, z) (((x) & (y)) | ((~x) & (z)))          // F function
2  #define G(x, y, z) (((x) & (z)) | ((y) & (~z)))          // G function
3  #define H(x, y, z) ((x) ^ (y) ^ (z))                    // H function
4  #define I(x, y, z) ((y) ^ ((x) | (~z)))                  // I function
5  #define ROTATE_LEFT(x, n) (((x) << (n)) | ((x) >> (32 - (n)))) // 32 bit num x cycle left
shift
6
7  #define FF(a, b, c, d, x, s, ac)                        \
8      {                                                    \
9          (a) += F((b), (c), (d)) + (x) + ac; \
10         (a) = ROTATE_LEFT((a), (s)); \
11         (a) += (b); \
12     }
13 #define GG(a, b, c, d, x, s, ac)                        \
14     {                                                    \
15         (a) += G((b), (c), (d)) + (x) + ac; \
16         (a) = ROTATE_LEFT((a), (s)); \
17         (a) += (b); \
18     }
19 #define HH(a, b, c, d, x, s, ac)                        \
20     {                                                    \
21         (a) += H((b), (c), (d)) + (x) + ac; \
22         (a) = ROTATE_LEFT((a), (s)); \
23         (a) += (b); \
24     }
25 #define II(a, b, c, d, x, s, ac)                        \

```

```

26     {
27         (a) += I((b), (c), (d)) + (x) + ac; \
28         (a) = ROTATE_LEFT((a), (s));      \
29         (a) += (b);                        \
30     }
31
32 #define T(i) 4294967296 * abs(sin(i))
33
34 const BYTE MD5::padding[64] = {0x80};
35 const char MD5::hex[16] = {
36     '0', '1', '2', '3',
37     '4', '5', '6', '7',
38     '8', '9', 'a', 'b',
39     'c', 'd', 'e', 'f'};

```

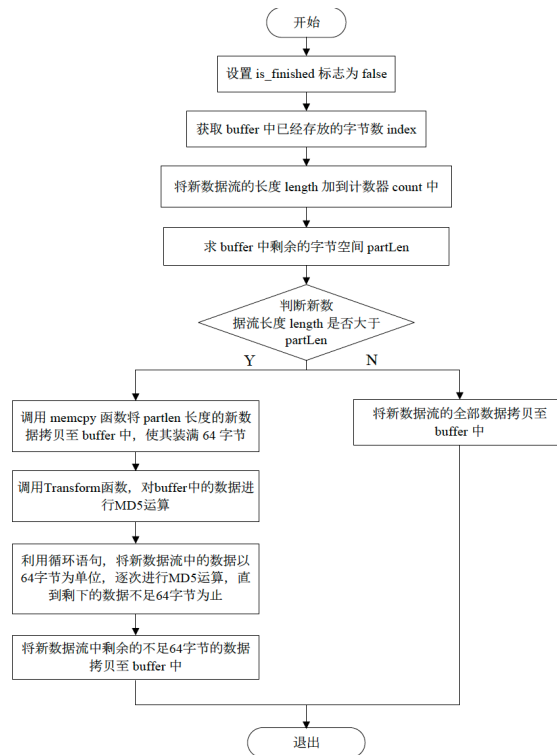
接着编写了三个构造函数，用于对不同的输入进行 MD5 计算。每个构造函数分别调用对应参数的 `Update` 函数，用于进行计算。

```

1 MD5::MD5()
2 {
3     Reset();
4 }
5
6 MD5::MD5(const string &str)
7 {
8     Reset();
9     Update(str);
10 }
11
12 MD5::MD5(istream &in)
13 {
14     Reset();
15     Update(in);
16 }

```

接着实现了三个 `Update` 函数，用于进行 MD5 计算，此处以输入定长字符串为例进行解释说明，未指定长度的字符串和输入流均可以转换成定长字符串类型使用。流程图如下：



- 首先声明 DWORD 类型的变量 i、index、partLen，并将标识 is_finished 设置为 false，表示 MD5 计算未完成。
- 接着根据输入数据流的长度更新 MD5 计算的计数器 count，其中 count[0] 记录低 32 位，count[1] 记录高 32 位。首先将 count[0] 左移 3 位，相当于乘以 8，然后加上新数据流的长度，如果发生了进位，则将 count[1] 加 1。
- 然后计算当前缓冲区中剩余空间的长度 partLen。如果输入数据流长度大于等于剩余空间长度 partLen，则将部分数据拷贝到缓冲区中，并进行 MD5 变换处理。如果输入数据流长度超过了剩余空间长度，则将剩余部分数据拷贝到缓冲区中，同时对缓冲区中的数据进行 MD5 变换处理。
- 最后将输入数据流中剩余的数据拷贝到缓冲区中，以备下一次更新计算。

```

1 void MD5::Update(const BYTE *input, size_t length)
2 {
3     DWORD i, index, partLen;
4     is_finished = false;           // 设置停止标识
5     index = (DWORD)((count[0] >> 3) & 0x3f); // 计算 buffer 已经存放的字节数
6
7     // 更新计数器 count, 将新数据流的长度加上计数器原有的值
8     if ((count[0] += ((DWORD)length << 3)) < ((DWORD)length << 3)) // 判断是否进位
9     {
10         count[1]++;
11     }
12     count[1] += ((DWORD)length >> 29);
13
14     partLen = 64 - index; // 求出 buffer 中剩余的长度
15
16     // 将数据块逐块进行 MD5 运算
17     if (length >= partLen)
18     {
19         memcpy(&buffer_block[index], input, partLen);
20         Transform(buffer_block);

```

```

21         for (i = partLen; i + 63 < length; i += 64)
22         {
23             Transform(&input[i]);
24         }
25         index = 0;
26     }
27     else
28     {
29         i = 0;
30     }
31     memcpy(&buffer_block[index], &input[i], length - i); // 将不足 64 字节的数据复制到
buffer_block 中
32 }
33
34 void MD5::Update(const string &str)
35 {
36     Update((const BYTE *)str.c_str(), str.length());
37 }
38
39 void MD5::Update(istream &in)
40 {
41     streamsize length;
42     char buffer[1024];
43     while (!in.eof())
44     {
45         in.read(buffer, 1024);
46         length = in.gcount();
47         if (length > 0)
48         {
49             Update((const BYTE *)buffer, length);
50         }
51     }
52     in.close();
53 }

```

然后编写了一个 `Transform` 函数，对 64 字节数据块进行 MD5 计算。相关实现思路参见实验指导手册，在此不做阐述。

```

1 void MD5::Transform(const BYTE block[64])
2 {
3     DWORD a = state[0], b = state[1], c = state[2], d = state[3], x[16];
4     Decode(block, x, 64);
5
6     /* 第 1 轮 */
7     FF(a, b, c, d, x[0], 7, T(1));
8     FF(d, a, b, c, x[1], 12, T(2));
9     FF(c, d, a, b, x[2], 17, T(3));
10    FF(b, c, d, a, x[3], 22, T(4));
11    FF(a, b, c, d, x[4], 7, T(5));
12    FF(d, a, b, c, x[5], 12, T(6));
13    FF(c, d, a, b, x[6], 17, T(7));
14    FF(b, c, d, a, x[7], 22, T(8));

```

```

15 FF(a, b, c, d, x[8], 7, T(9));
16 FF(d, a, b, c, x[9], 12, T(10));
17 FF(c, d, a, b, x[10], 17, T(11));
18 FF(b, c, d, a, x[11], 22, T(12));
19 FF(a, b, c, d, x[12], 7, T(13));
20 FF(d, a, b, c, x[13], 12, T(14));
21 FF(c, d, a, b, x[14], 17, T(15));
22 FF(b, c, d, a, x[15], 22, T(16));
23
24 /* 第 2 轮 */
25 GG(a, b, c, d, x[1], 5, T(17));
26 GG(d, a, b, c, x[6], 9, T(18));
27 GG(c, d, a, b, x[11], 14, T(19));
28 GG(b, c, d, a, x[0], 20, T(20));
29 GG(a, b, c, d, x[5], 5, T(21));
30 GG(d, a, b, c, x[10], 9, T(22));
31 GG(c, d, a, b, x[15], 14, T(23));
32 GG(b, c, d, a, x[4], 20, T(24));
33 GG(a, b, c, d, x[9], 5, T(25));
34 GG(d, a, b, c, x[14], 9, T(26));
35 GG(c, d, a, b, x[3], 14, T(27));
36 GG(b, c, d, a, x[8], 20, T(28));
37 GG(a, b, c, d, x[13], 5, T(29));
38 GG(d, a, b, c, x[2], 9, T(30));
39 GG(c, d, a, b, x[7], 14, T(31));
40 GG(b, c, d, a, x[12], 20, T(32));
41
42 // /* 第 3 轮 */
43 HH(a, b, c, d, x[5], 4, T(33));
44 HH(d, a, b, c, x[8], 11, T(34));
45 HH(c, d, a, b, x[11], 16, T(35));
46 HH(b, c, d, a, x[14], 23, T(36));
47 HH(a, b, c, d, x[1], 4, T(37));
48 HH(d, a, b, c, x[4], 11, T(38));
49 HH(c, d, a, b, x[7], 16, T(39));
50 HH(b, c, d, a, x[10], 23, T(40));
51 HH(a, b, c, d, x[13], 4, T(41));
52 HH(d, a, b, c, x[0], 11, T(42));
53 HH(c, d, a, b, x[3], 16, T(43));
54 HH(b, c, d, a, x[6], 23, T(44));
55 HH(a, b, c, d, x[9], 4, T(45));
56 HH(d, a, b, c, x[12], 11, T(46));
57 HH(c, d, a, b, x[15], 16, T(47));
58 HH(b, c, d, a, x[2], 23, T(48));
59
60 // /* 第 4 轮 */
61 II(a, b, c, d, x[0], 6, T(49));
62 II(d, a, b, c, x[7], 10, T(50));
63 II(c, d, a, b, x[14], 15, T(51));
64 II(b, c, d, a, x[5], 21, T(52));
65 II(a, b, c, d, x[12], 6, T(53));
66 II(d, a, b, c, x[3], 10, T(54));

```

```

67     II(c, d, a, b, x[10], 15, T(55));
68     II(b, c, d, a, x[1], 21, T(56));
69     II(a, b, c, d, x[8], 6, T(57));
70     II(d, a, b, c, x[15], 10, T(58));
71     II(c, d, a, b, x[6], 15, T(59));
72     II(b, c, d, a, x[13], 21, T(60));
73     II(a, b, c, d, x[4], 6, T(61));
74     II(d, a, b, c, x[11], 10, T(62));
75     II(c, d, a, b, x[2], 15, T(63));
76     II(b, c, d, a, x[9], 21, T(64));
77
78     state[0] += a;
79     state[1] += b;
80     state[2] += c;
81     state[3] += d;
82 }

```

除此之外，还编写了几个函数用于初始化 MD5 计算器、对数据结构进行转化以及设置输出流等。

```

1  string MD5::ToString()
2  {
3      return BytesToHexString(GetDigest(), 16);
4  }
5
6  const BYTE *MD5::GetDigest()
7  {
8      if (!this->is_finished)
9      {
10         this->is_finished = true;
11         Stop();
12     }
13     return this->digest;
14 }
15
16 void MD5::Reset()
17 {
18     state[0] = 0x67452301;
19     state[1] = 0xefcdab89;
20     state[2] = 0x98badcfe;
21     state[3] = 0x10325476;
22     count[0] = 0;
23     count[1] = 0;
24     memset(buffer_block, 0, 64);
25     memset(digest, 0, 16);
26     is_finished = false;
27 }
28
29 void MD5::Stop()
30 {
31     BYTE bits[8];
32     DWORD tmp_state[4];
33     DWORD tmp_count[2];
34

```

```

35     memcpy(tmp_state, state, 16);
36     memcpy(tmp_count, count, 8);
37     Encode(count, bits, 8);
38
39     DWORD index = (count[0] / 8) % 64;
40     DWORD padLen;
41     if (index < 56)
42     {
43         padLen = 56 - index;
44     }
45     else
46     {
47         padLen = 120 - index;
48     }
49
50     Update(padding, padLen);
51     Update(bits, 8);
52     Encode(state, digest, 16);
53     memcpy(state, tmp_state, 16);
54     memcpy(count, tmp_count, 8);
55 }
56
57 void MD5::Encode(const DWORD *input, BYTE *output, size_t length)
58 {
59     for (size_t i = 0; i * 4 < length; i++)
60     {
61         for (size_t k = 0; k < 4; k++)
62         {
63             output[i * 4 + k] = (BYTE)((input[i] >> (k * 8)) & 0xff);
64         }
65     }
66 }
67
68 void MD5::Decode(const BYTE *input, DWORD *output, size_t length)
69 {
70     const BYTE *inputPtr = input;
71     const BYTE *inputEnd = input + length;
72
73     while (inputPtr < inputEnd)
74     {
75         *output = *((const DWORD *)inputPtr);
76         output++;
77         inputPtr += sizeof(DWORD);
78     }
79 }
80
81 string MD5::BytesToHexString(const BYTE *input, size_t length)
82 {
83     string tmp;
84     for (size_t i = 0; i < length; i++)
85     {
86         int t = input[i];

```



```
87         int a = t / 16;
88         int b = t % 16;
89         tmp.append(1, hex[a]);
90         tmp.append(1, hex[b]);
91     }
92     return tmp;
93 }
```

4.2 文件完整性检验的设计与实现

此部分主要在 `main.cpp` 中进行编写。主要思路是根据用户输入的参数进行功能的调用，并对产生的错误进行处理。

- 首先，如果参数数量过少或者过多，则输出错误信息并退出。

```
1  if (argc < 2 || argc > 4)
2  {
3      cout << "Parameter Error !" << endl;
4      return -1;
5  }
```

- 如果参数数量为 2 且参数值为 -h，则输出 MD5 的使用方法。

```

1 else if ((argc == 2) && (strcmp(argv[1], "-h") == 0))
2 {
3     cout << "MD5 usage:\t[-h]\t--help information\n";
4     cout << "\t\t[-t]\t--test MD5 application\n";
5     cout << "\t\t[-c]\t[file path of the file computed]\n";
6     cout << "\t\t\t--compute MD5 of the given file\n";
7     cout << "\t\t[-v]\t[file path of the file validated]\n";
8     cout << "\t\t\t--validate the integrity of a given file by manual input MD5
value\n";
9     cout << "\t\t[-f]\t[file path of the file validated] [file path of the .md5 file]\n";
10    cout << "\t\t\t--validate the integrity of a given file by read MD5 value from .md5
file\n";
11 }

```

- 如果参数数量为 2 且参数值为 -t，则输出 MD5 的测试样例。

```

1 else if ((argc == 2) && (strcmp(argv[1], "-t") == 0))
2 {
3     string strlist[] = {"",
4                         "a",
5                         "abc",
6                         "message digest",
7                         "abcdefghijklmnopqrstuvwxyz",
8                         "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789",
9                         "123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890"};
10    for (int i = 0; i < 7; i++)
11    {
12        cout << "MD5(\"" << strlist[i] << "\") = " << MD5(strlist[i]).ToString() << endl;
13    }
14 }

```

- 如若参数数量为 3 且参数值为 -c，表示计算给定路径的文件的 MD5 值。

```

1  else if ((argc == 3) && (strcmp(argv[1], "-c") == 0))
2  {
3      if (argv[2] == NULL)
4      {
5          cout << "Error arameter!" << endl;
6          return -1;
7      }
8      string pFilePath = argv[2];
9      ifstream File_1(pFilePath);
10     cout << "The MD5 value of file(\"" << pFilePath << "\") is " << MD5(File_1).ToString()
    << endl;
11 }

```

- 如果参数数量为 3 且参数值为 -v，表示对文件的 MD5 值进行手动匹配验证。

```

1  else if ((argc == 3) && (strcmp(argv[1], "-v") == 0))
2  {
3      if (argv[2] == NULL)
4      {
5          cout << "Error arameter!" << endl;
6          return -1;
7      }
8      cout << "Please input the MD5 value of file(\"" << argv[2] << "\")..." << endl;
9      char InputMD5[33];
10     cin >> InputMD5;
11     InputMD5[32] = '\0';
12     string pFilePath = argv[2];
13     ifstream File_2(pFilePath);
14     string str = MD5(File_2).ToString();
15     cout << "The MD5 of the file(\"" << argv[2] << "\") is " << str << endl;
16     const char *pResult = str.c_str();
17     if (strcmp(InputMD5, pResult) != 0)
18     {
19         cout << "The file is incomplete!" << endl;
20         return 0;
21     }
22     else
23     {
24         cout << "The file is complete!" << endl;
25         return 0;
26     }
27 }

```

- 如果参数数量为 4 且参数值为 -f，表示基于 .md5 文件对给定的文件进行验证。

```

1  else if ((argc == 4) && (strcmp(argv[1], "-f") == 0))
2  {
3      if (argv[2] == NULL || argv[3] == NULL)
4      {
5          cout << "Error arameter!" << endl;
6          return -1;

```

```

7     }
8     string pFilePath = argv[3];
9     ifstream File_3(pFilePath);
10    char Record[50];
11    File_3.getline(Record, 50);
12    char *pMD5 = strtok(Record, "");
13    char *pFileName = strtok(NULL, "");
14    pFilePath = argv[2];
15    ifstream File_4(pFilePath);
16    string str = MD5(File_4).ToString();
17    cout << "The MD5 of the file(\"" << argv[2] << "\") is " << str << endl;
18    const char *pResult = str.c_str();
19    if (strcmp(pMD5, pResult) != 0)
20    {
21        cout << "The file is incomplete!" << endl;
22        return 0;
23    }
24    else
25    {
26        cout << "The file is complete!" << endl;
27        return 0;
28    }
29 }

```

- 如果参数非法，则提示错误并退出。

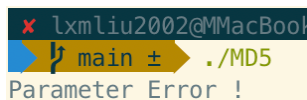
```

1 else
2 {
3     cout << "Parameter Error !" << endl;
4     return -1;
5 }

```

5 实验结论

输入 `./MD5`，可以看到，由于输入参数较少，输出错误信息。

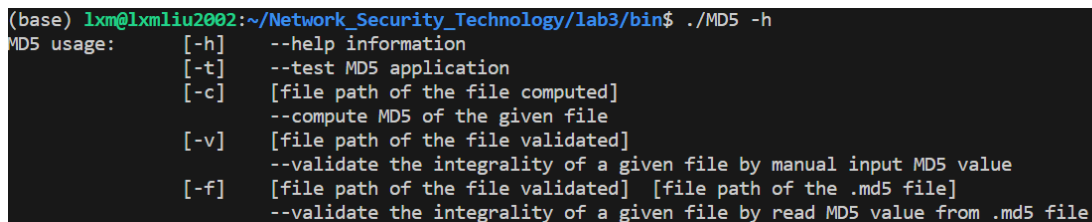


```

x lxmliu2002@MMacBook
main ± ./MD5
Parameter Error !

```

输入 `./MD5 -h`，可以看到，输出得到其用法说明。



```

(base) lxmliu2002:~/Network_Security_Technology/lab3/bin$ ./MD5 -h
MD5 usage:      [-h]      --help information
                [-t]      --test MD5 application
                [-c]      [file path of the file computed]
                        --compute MD5 of the given file
                [-v]      [file path of the file validated]
                        --validate the integrity of a given file by manual input MD5 value
                [-f]      [file path of the file validated] [file path of the .md5 file]
                        --validate the integrity of a given file by read MD5 value from .md5 file

```

输入 `./MD5 -t`，可以看到，输出测试用例的 MD5 值。将其与实验手册上的值进行对比，可以确定

```
(base) lxm@lxmliu2002:~/Network_Security_Technology/lab3/bin$ ./MD5 -t
MD5("") = d41d8cd98f00b204e9800998ecf8427e
MD5("a") = 0cc175b9c0f1b6a831c399e269772661
MD5("abc") = 900150983cd24fb0d6963f7d28e17f72
MD5("message digest") = f96b697d7cb7938d525a2f31aaf161d0
MD5("abcdefghijklmnopqrstuvwxyz") = c3fcd3d76192e4007dfb496cca67e13b
MD5("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz123456789") = d174ab98d277d9f5a5611c2c9f419d9f
MD5("123456789012345678901234567890123456789012345678901234567890") = 57edf4a22be3c955ac49da2e2107b67a
```

输入 `./MD5 -c ./nankai.txt`，可以看到，输出得到该文件的 MD5 值。

```
(base) lxm@lxmliu2002:~/Network_Security_Technology/lab3/bin$ ./MD5 -c ./nankai.txt
The MD5 value of file("./nankai.txt") is 78944e46be54be728f82f2ab458e717c
```

使用在线计算器，得到如下的 MD5 值，可以看到二者是相同的，说明程序编写成功。

文件MD5值:

执行过程:

```

计算耗时: 2ms
计算成功, MD5值: 78944e46be54be728f82f2ab458e717c
加载数据: 第1部分, 总1部分
开始计算, 文件名: (nankai.txt)
  
```

输入 `./MD5 -v ./nankai.txt`，然后输入文件的 MD5 值，可以看到，经过计算对比，该文件是完整的。

```
(base) lxm@lxmliu2002:~/Network_Security_Technology/lab3/bin$ ./MD5 -v ./nankai.txt
Please input the MD5 value of file("./nankai.txt")...
78944e46be54be728f82f2ab458e717c
The MD5 of the file("./nankai.txt") is 78944e46be54be728f82f2ab458e717c
The file is complete!
```

对文件进行删改。

```

Network_Security_Technology > lab3 > bin > nankai.txt
1 1 About Nankai University (written by sky)
2 3 A key multidisciplinary and research-oriented universi
3 the Ministry of Education, Nankai University, located in
  alma mater of our beloved late Premier Zhou Enlai.
  
```

使用原来的 MD5 值进行对比，可以看到，文件不再完整。

```
(base) lxm@lxmliu2002:~/Network_Security_Technology/lab3/bin$ ./MD5 -v ./nankai.txt
Please input the MD5 value of file("./nankai.txt")...
78944e46be54be728f82f2ab458e717c
The MD5 of the file("./nankai.txt") is a30048b3b631f42731486a953621b58f
The file is incomplete!
```

输入 `./MD5 -f ./nankai.txt ./nankai.md5`，可以看到，经过计算对比，该文件是完整的。

```
(base) lxm@lxmliu2002:~/Network_Security_Technology/lab3/bin$ ./MD5 -f ./nankai.txt ./nankai.md5
The MD5 of the file("./nankai.txt") is 78944e46be54be728f82f2ab458e717c
The file is complete!
```

输入 `./MD5 1`，可以看到，由于输入参数非法，输出错误信息。

```

lxmliu2002@MMacBook-
main ➔ ./MD5 1
Parameter Error !
  
```

6 实验遇到的问题及其解决方法

本次实验中遇到的最大问题就是对 MD5 算法的不了解。但是经过仔细阅读实验指导书，以及查阅相关博客，最终还是完成了项目编写。

7 实验收获

对 MD5 算法有了深入了解，对 cmake 编译组件进一步熟练。

8 文件组织说明

本次实验使用 cmake 进行编译组织。在根目录下有一个 `report.pdf` 为本次实验的实验报告，另有一个文件夹 `code`，存放本次实验用到的所有代码。

- `./code/Readme.md` 为编译及运行说明
- `./code/bin/MD5` 为可执行文件，直接运行即可
- `./code/build` 文件夹为编译文件夹，存放编译用的代码，与 `CMakeLists.txt` 及 `Makefile` 配合使用
- `./code/include` 文件夹存放编写的 DES 算法代码
- `./code/src` 文件夹则为主要的 cpp 代码

```
1  |
2  | └─ code
3  |   │   └─ CMakeLists.txt
4  |   │   └─ Readme.md
5  |   │   └─ bin
6  |   │       │   └─ MD5
7  |   │       │   └─ nankai.txt
8  |   │       └─ nankai.md5
9  |   └─ build
10 |       │   └─ Makefile
11 |   └─ include
12 |       │   └─ MD5.hpp
13 |   └─ src
14 |       │   └─ CMakeLists.txt
15 |       └─ main.cpp
16 └─ report.pdf
```

9 实验参考

吴功宜主编.网络安全高级软件编程技术.清华大学出版社.2010

<https://zhuanlan.zhihu.com/p/351883327>