



南开大学
Nankai University

南 开 大 学

网 安 学 院

网络安全技术实验报告

第三次作业

沙璇 1911562

年级：2019 级

专业：信息安全

提交日期：2022/5/20

2022 年 5 月 19 日

摘要

基于 MD5 算法的文件完整性校验程序

关键字: MD5, linux

目录

一、 实验目的	1
二、 实验要求	1
三、 实验内容	1
(一) md5 相关知识	1
(二) 具体代码	3
1. md5.h	3
2. md5.cpp	4
3. main.cpp	12
四、 实验结果	16
五、 总结	20

一、 实验目的

MD5 算法是目前最流行的一种信息摘要算法，在数字签名，加密与解密技术，以及文件完整性检测等领域中发挥着巨大的作用。熟悉 MD5 算法对开发网络应用程序，理解网络安全的概念具有十分重要的意义。

本章编程训练的目的如下：

深入理解 MD5 算法的基本原理。

掌握利用 MD5 算法生成数据摘要的所有计算过程。

掌握 Linux 系统中检测文件完整性的基本方法。

熟悉 Linux 系统中文件的基本操作方法

二、 实验要求

本章编程训练的要求如下：

准确地实现 MD5 算法的完整计算过程。

对于任意长度的字符串能够生成 128 位 MD5 摘要。

对于任意大小的文件能够生成 128 位 MD5 摘要。

通过检查 MD5 摘要的正确性来检验原文件的完整性。

三、 实验内容

本次实验使用 C 语言编写，在 kali 中运行。

涉及到的主要源文件有：main.cpp、md5.h、md5.cpp。完整代码见附件。

1. md5.h：该文件包含 md5 算法中所需头文件。
2. md5.cpp：md5 算法的实现。
3. main.cpp：该文件是程序的主文件，用于完成 MD5 程序中功能的选择和调用。

(一) md5 相关知识

Hash 函数是将任意长的数字串转换成一个较短的定长输出数字串的函数，输出的结果称为 Hash 值。

下面从 MD5 入手来介绍 Hash 算法的实现机制。

MD 系列单向散列函数是由 Ron Rivest 设计的，MD5 算法对任意长度的输入值处理后产生 128 位的 Hash 值。流程如图1所示。

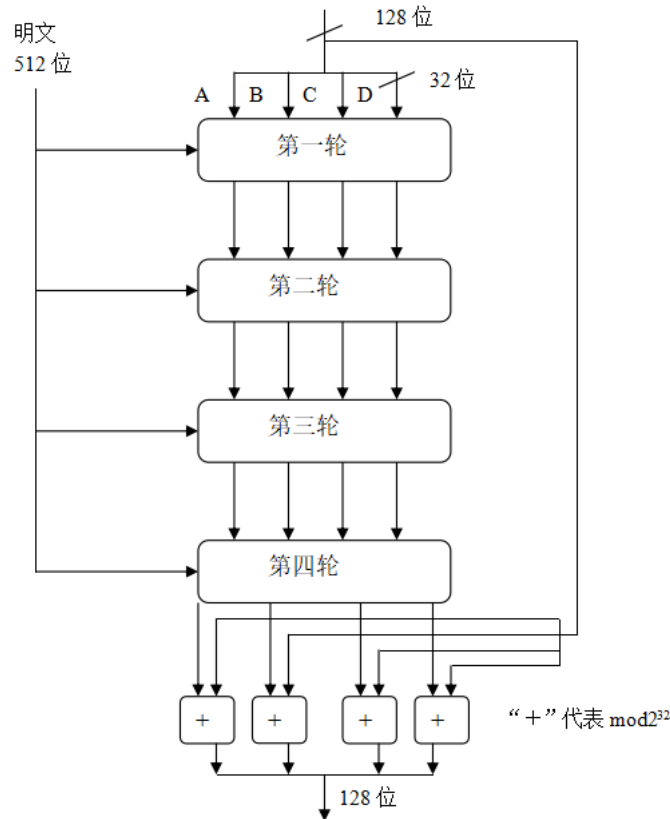


图 1: MD5

MD5 算法的实现步骤如下：

在 MD5 算法中，首先需要对信息进行填充，使其字节长度与 448 模 512 同余，即信息的字节长度扩展至 $n \times 512 + 448$ ， n 为一个正整数。填充的方法如下：在信息的后面填充第一位为 1，其余各位均为 0，直到满足上面的条件时才停止用 0 对信息的填充。然后，再在这个结果后面附加一个以 64 位二进制表示的填充前信息长度。经过这两步的处理，现在的信息字节长度为 $n \times 512 + 448 + 64 = (n+1) \times 512$ ，即长度恰好是 512 的整数倍，这样做的目的是为了后面处理中对信息长度的要求。

MD5 中有 A、B、C、D，4 个 32 位被称为链接变量的整数参数，它们的初始值分别为：

$A_0 = 0x01234567, B_0 = 0x89abcdef, C_0 = 0xfedcba98, D_0 = 0x76543210$

当设置好这 4 个链接变量后，就开始进入算法的 4 轮循环运算。循环的次数是信息中 512 位信息分组数目。

首先将上面 4 个链接变量复制到变量 A、B、C、D 中，以备后面进行处理。

然后进入主循环，主循环有 4 轮，每轮循环都很相似。第一轮进行 16 次操作，每次操作对 A、B、C、D 中的 3 个做一次非线性函数运算，然后将所得结果加上第四个变量，文本的一个子分组（32 位）和一个常数。再将所得结果向左循环移 S 位，并加上 A、B、C、D 其中之一。最后用该结果取代 A、B、C、D 其中之一。

每次操作中用到的 4 个非线性函数（每轮一个）如图2所示。

$$F(B,C,D) = (B \wedge C) \vee (\bar{B} \wedge D)$$

$$G(B,C,D) = (B \wedge D) \vee (C \wedge \bar{D})$$

$$H(B,C,D) = B \oplus C \oplus D$$

$$I(B,C,D) = C \oplus (B \vee \bar{D})$$

图 2: 每轮函数

MD5 轮主要操作为:

$$a = b + ((a + f(b, c, d) + M + t) \ll s)$$

对应于四轮操作, f 分别取 F, G, H, I ; 对每一轮的 16 次运算, M 分别取 M_1, M_2, \dots, M_{16} 。对于 4 轮共 64 次运算, t 为给定的一些常数, 另外一个常数是 $2^{32} \cdot \text{abs}(\sin(i))$ 的整数部分, 其中 $i = 1, 2, \dots, 64$ 。在 $\sin(i)$ 中, i 的单位是弧度, 由此构成了 32 位的随机数源 $s(i)$, 它消除了输入数据中任何规律性的特征。

对于 4 轮 64 次操作的具体运算下面的具体代码部分会给出。

所有这些操作完成之后, 将 A, B, C, D 分别加上 A_0, B_0, C_0, D_0 。然后用下一分组数据继续进行运算, 最后得到一组 A, B, C, D 。把这组数据级联起来, 即得到 128 比特的 Hash 结果。

(二) 具体代码

MD5 程序分为三个部分。涉及到的主要源文件有 `md5.h`、`md5.cpp`、`main.cpp`。

1. `md5.h`: 该文件包含 md5 算法中所需头文件。
2. `md5.cpp`: md5 算法的实现。
3. `main.cpp`: 该文件是程序的主文件, 用于完成 MD5 程序中功能的选择和调用。

1. `md5.h`

该文件包含 md5 算法中所需头文件

`md5.h`

```

1  #include <string>
2  #include <fstream>
3  #include <cstring>
4
5  /* Type define */
6  typedef unsigned char byte;
7  typedef unsigned int uint32;
8
9  using std::string;
10 using std::ifstream;
11
12 /* MD5 declaration. */
13 class MD5 {
14 public:
15     MD5();
16     MD5(const void *input, size_t length);

```

```

17 MD5(const string &str);
18 MD5(istream &in);
19 void update(const void *input, size_t length);
20 void update(const string &str);
21 void update(istream &in);
22 const byte* digest();
23 string toString();
24 void reset();
25 private:
26 void update(const byte *input, size_t length);
27 void final();
28 void transform(const byte block[64]);
29 void encode(const uint32 *input, byte *output, size_t length);
30 void decode(const byte *input, uint32 *output, size_t length);
31 string bytesToHexString(const byte *input, size_t length);
32
33 /* class uncopyable */
34 MD5(const MD5&);
35 MD5& operator=(const MD5&);
36 private:
37 uint32 _state[4]; /* state (ABCD) */
38 uint32 _count[2]; /* number of bits, modulo 2^64 (low-order
39 word first) */
40 byte _buffer[64]; /* input buffer */
41 byte _digest[16]; /* message digest */
42 bool _finished; /* calculate finished ? */
43
44 static const byte PADDING[64]; /* padding for calculate */
45 static const char HEX[16];
46 static const size_t BUFFER_SIZE = 1024;
47 };

```

2. md5.cpp

在 MD5.h 头文件中定义了 MD5 结构，以及循环的线性函数

$$F(X, Y, Z) = (X \cap Y) \mid ((\sim X) \cap Z)$$

$$G(X, Y, Z) = (X \cap Z) \mid (Y \cap (\sim Z))$$

$$H(X, Y, Z) = X \wedge Y \wedge Z$$

$$I(X, Y, Z) = Y \wedge (X \mid (\sim Z))$$

这四个函数的说明：如果 X、Y 和 Z 的对应位是独立和均匀的，那么结果的每一位也应是独立和均匀的。F 是一个逐位运算的函数。即，如果 X，那么 Y，否则 Z。函数 H 是逐位奇偶操作符。

假设 M_j 表示消息的第 j 个子分组（从 0 到 15），常数 t_i 是 $2^{32} \cdot \text{abs}(\sin(i))$ 的整数部分， i 取值从 1 到 64，单位是弧度。

定义：

FF(a, b, c, d, M_j , s, t_i) 操作为 $a = b + ((a + F(b, c, d) + M_j + t_i) \ll s)$

GG(a, b, c, d, M_j , s, t_i) 操作为 $a = b + ((a + G(b, c, d) + M_j + t_i) \ll s)$

HH(a ,b ,c ,d ,Mj ,s ,ti) 操作为 $a = b + ((a + H(b,c,d) + Mj + ti) \ll s)$

II(a ,b ,c ,d ,Mj ,s ,ti) 操作为 $a = b + ((a + I(b,c,d) + Mj + ti) \ll s)$

MD5

```

1  #include "md5.h"
2  using namespace std;
3
4  /* Constants for MD5Transform routine. */
5  #define S11 7
6  #define S12 12
7  #define S13 17
8  #define S14 22
9  #define S21 5
10 #define S22 9
11 #define S23 14
12 #define S24 20
13 #define S31 4
14 #define S32 11
15 #define S33 16
16 #define S34 23
17 #define S41 6
18 #define S42 10
19 #define S43 15
20 #define S44 21
21
22
23 /* F, G, H and I are basic MD5 functions.
24 */
25 #define F(x, y, z) (((x) & (y)) | ((~x) & (z)))
26 #define G(x, y, z) (((x) & (z)) | ((y) & (~z)))
27 #define H(x, y, z) ((x) ^ (y) ^ (z))
28 #define I(x, y, z) ((y) ^ ((x) | (~z)))
29
30 /* ROTATE_LEFT rotates x left n bits.
31 */
32 #define ROTATE_LEFT(x, n) (((x) << (n)) | ((x) >> (32-(n))))
33
34 /* FF, GG, HH, and II transformations for rounds 1, 2, 3, and 4.
35 Rotation is separate from addition to prevent recomputation.
36 */
37 #define FF(a, b, c, d, x, s, ac) { \
38     (a) += F ((b), (c), (d)) + (x) + ac; \
39     (a) = ROTATE_LEFT ((a), (s)); \
40     (a) += (b); \
41 }
42 #define GG(a, b, c, d, x, s, ac) { \
43     (a) += G ((b), (c), (d)) + (x) + ac; \
44     (a) = ROTATE_LEFT ((a), (s)); \

```

```

45     (a) += (b); \
46 }
47 #define HH(a, b, c, d, x, s, ac) { \
48     (a) += H ((b), (c), (d)) + (x) + ac; \
49     (a) = ROTATE_LEFT ((a), (s)); \
50     (a) += (b); \
51 }
52 #define II(a, b, c, d, x, s, ac) { \
53     (a) += I ((b), (c), (d)) + (x) + ac; \
54     (a) = ROTATE_LEFT ((a), (s)); \
55     (a) += (b); \
56 }
57
58
59 const byte MD5::PADDING[64] = { 0x80 };
60 const char MD5::HEX[16] = {
61     '0', '1', '2', '3',
62     '4', '5', '6', '7',
63     '8', '9', 'a', 'b',
64     'c', 'd', 'e', 'f'
65 };

```

使用进行 MD5 的文件、字符串等进行 MD5 对象的构建

```

1  /* Default construct. */
2  MD5::MD5() {
3      reset();
4  }
5
6  /* Construct a MD5 object with a input buffer. */
7  MD5::MD5(const void *input, size_t length) {
8      reset();
9      update(input, length);
10 }
11
12 /* Construct a MD5 object with a string. */
13 MD5::MD5(const string &str) {
14     reset();
15     update(str);
16 }
17
18 /* Construct a MD5 object with a file. */
19 MD5::MD5(ifstream &in) {
20     reset();
21     update(in);
22 }
23
24 /* Return the message-digest */

```



```

25 const byte* MD5::digest() {
26     if (!_finished) {
27         _finished = true;
28         final();
29     }
30     return _digest;
31 }
32
33 /* Reset the calculate state */
34 void MD5::reset() {
35
36     _finished = false;
37     /* reset number of bits. */
38     _count[0] = _count[1] = 0;
39     /* Load magic initialization constants. */
40     _state[0] = 0x67452301;
41     _state[1] = 0xefcdab89;
42     _state[2] = 0x98badcfe;
43     _state[3] = 0x10325476;
44 }

```

对 MD5 进行更新操作。

```

md5
1  /* Updating the context with a input buffer. */
2  void MD5::update(const void *input, size_t length) {
3      update((const byte*)input, length);
4  }
5
6  /* Updating the context with a string. */
7  void MD5::update(const string &str) {
8      update((const byte*)str.c_str(), str.length());
9  }
10
11 /* Updating the context with a file. */
12 void MD5::update(istream &in) {
13
14     if (!in)
15         return;
16
17     std::streamsize length;
18     char buffer[BUFFER_SIZE];
19     while (!in.eof()) {
20         in.read(buffer, BUFFER_SIZE);
21         length = in.gcount();
22         if (length > 0)
23             update(buffer, length);
24     }
25     in.close();

```

```

26 }
27
28 /* MD5 block update operation. Continues an MD5 message-digest
29 operation, processing another message block, and updating the
30 context.
31 */
32 void MD5::update(const byte *input, size_t length) {
33
34     uint32 i, index, partLen;
35
36     _finished = false;
37
38     /* Compute number of bytes mod 64 */
39     index = (uint32)((_count[0] >> 3) & 0x3f);
40
41     /* update number of bits */
42     if((_count[0] += ((uint32)length << 3)) < ((uint32)length << 3))
43         _count[1]++;
44     _count[1] += ((uint32)length >> 29);
45
46     partLen = 64 - index;
47
48     /* transform as many times as possible. */
49     if(length >= partLen) {
50
51         memcpy(&_amp;buffer[index], input, partLen);
52         transform(_buffer);
53
54         for (i = partLen; i + 63 < length; i += 64)
55             transform(&input[i]);
56         index = 0;
57
58     } else {
59         i = 0;
60     }
61
62     /* Buffer remaining input */
63     memcpy(&_amp;buffer[index], &input[i], length-i);
64 }

```

结束 MD5 消息摘要操作，写入消息摘要并将上下文归零

md5

```

1 void MD5::final() {
2
3     byte bits[8];
4     uint32 oldState[4];
5     uint32 oldCount[2];
6     uint32 index, padLen;

```

```

7
8      /* Save current state and count. */
9      memcpy(oldState, _state, 16);
10     memcpy(oldCount, _count, 8);
11
12     /* Save number of bits */
13     encode(_count, bits, 8);
14
15     /* Pad out to 56 mod 64. */
16     index = (uint32)((_count[0] >> 3) & 0x3f);
17     padLen = (index < 56) ? (56 - index) : (120 - index);
18     update(PADDING, padLen);
19
20     /* Append length (before padding) */
21     update(bits, 8);
22
23     /* Store state in digest */
24     encode(_state, _digest, 16);
25
26     /* Restore current state and count. */
27     memcpy(_state, oldState, 16);
28     memcpy(_count, oldCount, 8);
29 }

```

MD5 基本转换，用于块转换状态。

md5

```

1 void MD5::transform(const byte block[64]) {
2
3     uint32 a = _state[0], b = _state[1], c = _state[2], d = _state[3], x
4         [16];
5
6     decode(block, x, 64);
7
8     /* Round 1 */
9     FF (a, b, c, d, x[ 0], S11, 0xd76aa478); /* 1 */
10    FF (d, a, b, c, x[ 1], S12, 0xe8c7b756); /* 2 */
11    FF (c, d, a, b, x[ 2], S13, 0x242070db); /* 3 */
12    FF (b, c, d, a, x[ 3], S14, 0xc1bdcee); /* 4 */
13    FF (a, b, c, d, x[ 4], S11, 0xf57c0faf); /* 5 */
14    FF (d, a, b, c, x[ 5], S12, 0x4787c62a); /* 6 */
15    FF (c, d, a, b, x[ 6], S13, 0xa8304613); /* 7 */
16    FF (b, c, d, a, x[ 7], S14, 0xfd469501); /* 8 */
17    FF (a, b, c, d, x[ 8], S11, 0x698098d8); /* 9 */
18    FF (d, a, b, c, x[ 9], S12, 0x8b44f7af); /* 10 */
19    FF (c, d, a, b, x[10], S13, 0xffff5bb1); /* 11 */
20    FF (b, c, d, a, x[11], S14, 0x895cd7be); /* 12 */
21    FF (a, b, c, d, x[12], S11, 0x6b901122); /* 13 */
22    FF (d, a, b, c, x[13], S12, 0xfd987193); /* 14 */

```

```

22     FF (c, d, a, b, x[14], S13, 0xa679438e); /* 15 */
23     FF (b, c, d, a, x[15], S14, 0x49b40821); /* 16 */
24
25     /* Round 2 */
26     GG (a, b, c, d, x[ 1], S21, 0xf61e2562); /* 17 */
27     GG (d, a, b, c, x[ 6], S22, 0xc040b340); /* 18 */
28     GG (c, d, a, b, x[11], S23, 0x265e5a51); /* 19 */
29     GG (b, c, d, a, x[ 0], S24, 0xe9b6c7aa); /* 20 */
30     GG (a, b, c, d, x[ 5], S21, 0xd62f105d); /* 21 */
31     GG (d, a, b, c, x[10], S22, 0x2441453); /* 22 */
32     GG (c, d, a, b, x[15], S23, 0xd8a1e681); /* 23 */
33     GG (b, c, d, a, x[ 4], S24, 0xe7d3fbc8); /* 24 */
34     GG (a, b, c, d, x[ 9], S21, 0x21e1cde6); /* 25 */
35     GG (d, a, b, c, x[14], S22, 0xc33707d6); /* 26 */
36     GG (c, d, a, b, x[ 3], S23, 0xf4d50d87); /* 27 */
37     GG (b, c, d, a, x[ 8], S24, 0x455a14ed); /* 28 */
38     GG (a, b, c, d, x[13], S21, 0xa9e3e905); /* 29 */
39     GG (d, a, b, c, x[ 2], S22, 0xfcefa3f8); /* 30 */
40     GG (c, d, a, b, x[ 7], S23, 0x676f02d9); /* 31 */
41     GG (b, c, d, a, x[12], S24, 0x8d2a4c8a); /* 32 */
42
43     /* Round 3 */
44     HH (a, b, c, d, x[ 5], S31, 0xfffa3942); /* 33 */
45     HH (d, a, b, c, x[ 8], S32, 0x8771f681); /* 34 */
46     HH (c, d, a, b, x[11], S33, 0x6d9d6122); /* 35 */
47     HH (b, c, d, a, x[14], S34, 0xfde5380c); /* 36 */
48     HH (a, b, c, d, x[ 1], S31, 0xa4beea44); /* 37 */
49     HH (d, a, b, c, x[ 4], S32, 0x4bdecfa9); /* 38 */
50     HH (c, d, a, b, x[ 7], S33, 0xf6bb4b60); /* 39 */
51     HH (b, c, d, a, x[10], S34, 0xbebfbc70); /* 40 */
52     HH (a, b, c, d, x[13], S31, 0x289b7ec6); /* 41 */
53     HH (d, a, b, c, x[ 0], S32, 0xea127fa); /* 42 */
54     HH (c, d, a, b, x[ 3], S33, 0xd4ef3085); /* 43 */
55     HH (b, c, d, a, x[ 6], S34, 0x4881d05); /* 44 */
56     HH (a, b, c, d, x[ 9], S31, 0xd9d4d039); /* 45 */
57     HH (d, a, b, c, x[12], S32, 0xe6db99e5); /* 46 */
58     HH (c, d, a, b, x[15], S33, 0x1fa27cf8); /* 47 */
59     HH (b, c, d, a, x[ 2], S34, 0xc4ac5665); /* 48 */
60
61     /* Round 4 */
62     II (a, b, c, d, x[ 0], S41, 0xf4292244); /* 49 */
63     II (d, a, b, c, x[ 7], S42, 0x432aff97); /* 50 */
64     II (c, d, a, b, x[14], S43, 0xab9423a7); /* 51 */
65     II (b, c, d, a, x[ 5], S44, 0xfc93a039); /* 52 */
66     II (a, b, c, d, x[12], S41, 0x655b59c3); /* 53 */
67     II (d, a, b, c, x[ 3], S42, 0x8f0ccc92); /* 54 */
68     II (c, d, a, b, x[10], S43, 0xffefff47d); /* 55 */
69     II (b, c, d, a, x[ 1], S44, 0x85845dd1); /* 56 */

```

```

70     II (a, b, c, d, x[ 8], S41, 0x6fa87e4f); /* 57 */
71     II (d, a, b, c, x[15], S42, 0xfe2ce6e0); /* 58 */
72     II (c, d, a, b, x[ 6], S43, 0xa3014314); /* 59 */
73     II (b, c, d, a, x[13], S44, 0x4e0811a1); /* 60 */
74     II (a, b, c, d, x[ 4], S41, 0xf7537e82); /* 61 */
75     II (d, a, b, c, x[11], S42, 0xbd3af235); /* 62 */
76     II (c, d, a, b, x[ 2], S43, 0x2ad7d2bb); /* 63 */
77     II (b, c, d, a, x[ 9], S44, 0xeb86d391); /* 64 */
78
79     _state[0] += a;
80     _state[1] += b;
81     _state[2] += c;
82     _state[3] += d;
83 }

```

最后是 md5 加解密以及二进制转换 16 进制，以及字符串输出。

md5

```

1  void MD5::encode(const uint32 *input, byte *output, size_t length) {
2
3      for(size_t i=0, j=0; j<length; i++, j+=4) {
4          output[j]= (byte)(input[i] & 0xff);
5          output[j+1] = (byte)((input[i] >> 8) & 0xff);
6          output[j+2] = (byte)((input[i] >> 16) & 0xff);
7          output[j+3] = (byte)((input[i] >> 24) & 0xff);
8      }
9  }
10
11  /* Decodes input (byte) into output (ulong). Assumes length is
12  a multiple of 4.
13  */
14  void MD5::decode(const byte *input, uint32 *output, size_t length) {
15
16      for(size_t i=0, j=0; j<length; i++, j+=4) {
17          output[i] = ((uint32)input[j]) | (((uint32)input[j+1]) << 8)
18                  |
19                  (((uint32)input[j+2]) << 16) | (((uint32)input[j+3])
20                  << 24);
21      }
22  }
23
24  /* Convert byte array to hex string. */
25  string MD5::bytesToHexString(const byte *input, size_t length) {
26      string str;
27      str.reserve(length << 1);
28      for(size_t i = 0; i < length; i++) {
29          int t = input[i];
30          int a = t / 16;
31          int b = t % 16;

```

```

30         str.append(1, HEX[a]);
31         str.append(1, HEX[b]);
32     }
33     return str;
34 }
35
36 /* Convert digest to string value */
37 string MD5::toString() {
38     return bytesToHexString(digest(), 16);
39 }
40 }

```

3. main.cpp

该文件是程序的主文件，用于完成 MD5 程序中功能的选择和调用

在终端键入./MD5 进入程序调用，输入格式为./MD5 [功能选择] ([文件路径]) ([文件路径])

具体字符串对应的功能选择见如下代码中注释

MD5

```

1  int main(int argc, char* argv[])
2  {
3      char* pFilePath;           //需要进行md5计算的文件路径
4      char* pMd5FilePath;        //存放md5摘要的.md5文件路径
5      char  Md5Digest[33];        //md5摘要，用于存放手动输入的
                                   md5摘要信息
6      char  Md5Record[50];        // .md5文件中的一行记录
7      string strTmp;
8
9      //
10
11     //字符串定义
12     char* pHelpMsg = {"-h"};    // 帮助信息
13     char* pCompute = {"-c"};    // 计算指定文件的md5摘要
14     char* pMValidate = {"-mv"};  // 手动对文件进行md5认证
15     char* pfValidate = {"-fv"};  // 通过比较对文
                                   件的md5摘要进行认证
16     char* pSpace = {" "};       // 定义空格

```

当参数个数为 2 的时候，说明无文件路径，只进行简单的功能说明；当参数个数为 3 时，说明有一个文件路径，可以进行指定文件的 md5 摘要计算以及 md5 认证；当参数个数为 4 时，说明有两个文件路径，可以根据比较其各自的 md5 进行文件的认证，具体代码如下

MD5

```

1  //参数检测
2
3      if(argc<2 || argc>4)
4      {
5          cout<<"Parameter Error !"<<endl;
6      }

```

```
5         return -1;
6     }
7
8     //


---


9     //显示帮助信息
10    if((argc == 2)&&(!strcmp(pHelpMsg,argv[1])))
11    {
12        cout<<"MD5 usage:  [-h]  --help information"<<endl;
13        cout<<"          [-c]  [file path of the file
14                computed]"<<endl;
15        cout<<"          --compute MD5 of the given
16                file"<<endl;
17        cout<<"          [-mv] [file path of the file
18                validated]"<<endl;
19        cout<<"          --validate the integrity
20                of a given file by manual input MD5 value"<<endl;
21        cout<<"          [-fv] [file path of the file
22                validated] [file path of the .md5 file]"<<endl;
23        cout<<"          --validate the integrity
24                of a given file by read MD5 value from .md5 file"
25        <<endl;
26    }
27
28    //


---


29    //计算指定文件的md5摘要，并显示出来
30    if((argc == 3)&&(!strcmp(pCompute,argv[1])))
31    {
32        //如果没有文件路径，则参数出错
33        if(argv[2] == NULL)
34        {
35            cout<<"Parameter Error ! Please input file
36                path !"<<endl;
37            return -1;
38        }
39        else
40        {
41            pFilePath = argv[2];
42        }
43        //打开指定的文件
44        ifstream File_1(pFilePath);
45        //声明md5对象,并进行计算
46        MD5 md5_obj1(File_1);
47        //输出计算结果
48        cout<<"MD5(\""<<argv[2]<<"\"") = "<<md5_obj1.toString
```

```
41         ()<<endl;
42     }
43     //
44     //手动进行文件完整性检测
45     if((argc == 3)&&(!strcmp(pMValidate,argv[1])))
46     {
47         //如果没有文件路径，则参数出错
48         if(argv[2] == NULL)
49         {
50             cout<<"Parameter Error ! Please input file
51             path !"<<endl;
52             return -1;
53         }
54         else
55         {
56             pFilePath = argv[2];
57         }
58         //手动输入了被测文件的MD5摘要
59         cout<<"Please input the MD5 value of file(\""<<
60         pFilePath<<"\"...)..."<<endl;
61         cin>>Md5Digest;
62         //在摘要的字符串末尾加上结束符
63         Md5Digest[32] = '\0';
64         //打开指定的文件
65         ifstream File_2(pFilePath);
66         //声明md5对象,并进行计算
67         //MD5 md5_obj2(File_2);
68         MD5 md5_obj2;
69         md5_obj2.reset();
70         md5_obj2.update(File_2);
71         //读取文件内容并计算MD5摘要
72         strTmp = md5_obj2.toString();
73         const char* pMd5Digest = strTmp.c_str();
74
75         //输出两个摘要
76         cout<<"The MD5 digest of file(\""<<pFilePath<<"\"
77         which you input is: "<<endl;
78         cout<<Md5Digest<<endl;
79         cout<<"The MD5 digest of file(\""<<pFilePath<<"\"
80         which calculate by program is: "<<endl;
81         cout<<strTmp<<endl;
82
83         //比较摘要的结果是否相同
```



```
82         if (strcmp(pMd5Digest, Md5Digest))
83         {
84             cout<<"Match Error! The file is not
85                 integrated!"<<endl;
86         }
87         else
88         {
89             cout<<"Match Successfully! The file is
90                 integrated!"<<endl;
91         }
92     }
93     //
94
95     //通过.md5文件进行文件完整性检测
96     if((argc == 4)&&(!strcmp(pfValidate, argv[1])))
97     {
98         //如果没有文件路径, 则参数出错
99         if((argv[2] == NULL)|| (argv[3] == NULL))
100         {
101             cout<<"Parameter Error ! Please input file
102                 path !"<<endl;
103             return -1;
104         }
105         else
106         {
107             pFilePath = argv[2];
108             pMd5FilePath = argv[3];
109
110             //打开.md5文件
111             ifstream File_3(pMd5FilePath);
112             //读取.md5文件中的记录
113             File_3.getline(Md5Record, 50);
114
115             //以空格为标记, 获得.MD5文件中的MD5值与对应文件名
116             char* pMd5Digest_f = strtok(Md5Record, pSpace);
117             char* pFileName_f = strtok(NULL, pSpace);
118
119             //打开被测文件
120             ifstream File_4(pFilePath);
121             //声明md5对象, 并进行计算
122             //MD5 md5_obj3( File_4);
123             MD5 md5_obj3;
124             md5_obj3.reset();
125             md5_obj3.update( File_4);
126
127             //读取文件内容并计算MD5摘要
```

```
125         strTmp = md5_obj3.toString();
126         const char* pMd5Digest_c = strTmp.c_str();
127
128         // 输出两个摘要
129         cout<<"The MD5 digest of file(\"<pFileName_f<<"\"")
              which is in file(\"<pMd5FilePath<<"\"") is: "<<
              endl;
130         cout<<pMd5Digest_f<<endl;
131         cout<<"The MD5 digest of file(\"<pFilePath<<"\"")
              which calculate by programme is: "<<endl;
132         cout<<strTmp<<endl;
133
134         // 比较摘要，进行验证
135         if (strcmp(pMd5Digest_c, pMd5Digest_f))
136         {
137             cout<<"Match Error! The file is not
              integrated!"<<endl;
138         }
139         else
140         {
141             cout<<"Match Successfully! The file is
              integrated!"<<endl;
142         }
```

四、 实验结果

在 kali 系统中本文件夹下运行 make 编译文件
终端输入./MD5 执行编译文件
示例文件 oreo.txt，其具体内容如下：

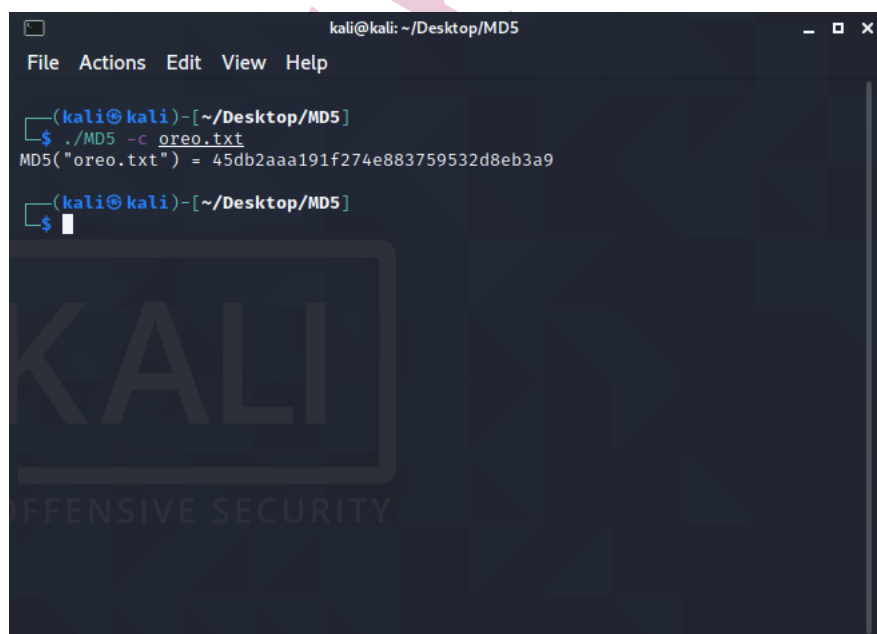
MD5

```
1 I offer you lean streets ,
2 我给你瘦落的街道，
3
4 desperate sunsets ,
5 绝望的落日，
6
7 the moon of the jagged suburbs .
8 荒郊的月亮，
9
10 I offer you the bitterness of a man who has looked long and long at the
    lonely moon.
11 我给你一个久久地望着孤月的人的悲哀。
12
13 I offer you my ancestors , my dead men ,
14 我给你我已死去的祖辈，
15
```

16 the ghosts that living men have honoured in marble:
17 后人们用大理石祭奠的先魂,
18
19 my father's father killed in the frontier of Buenos Aires,
20 我父亲的父亲阵亡于布宜诺斯艾利斯的边境,
21
22 two bullets through his lungs,
23 两颗子弹射穿了他的胸膛,
24
25 bearded and dead,
26 死的时候蓄着胡子,
27
28 wrapped by his soldiers in the hide of a cow;
29 尸体被士兵们用牛皮裹起。
30
31 my mother's grandfather
32 我母亲的祖父,
33
34 -just twentyfour-
35 那年才二十四岁,
36
37 heading a charge of three hundred men in Peru,
38 在秘鲁率领三百人冲锋,
39
40 now ghosts on vanished horses.
41 如今都成了消失的马背上的亡魂。
42
43 I offer you whatever insight my books may hold.
44 我给你我的书中所能蕴含的一切悟力,
45
46 whatever manliness or humour my life.
47 以及我生活中所能有的男子气概和幽默,
48
49 I offer you the loyalty of a man who has never been loyal.
50 我给你一个从未有过信仰的人的忠诚。
51
52 I offer you that kernel of myself that I have saved somehow -the central
53 heart
54 我给你我设法保全的我自己的核心,
55
56 that deals not in words, traffics not with dreams
57 不营字造句, 不和梦交易,
58
59 and is untouched by time, by joy, by adversities.
60 不被时间、欢乐和逆境触动的核心。
61
62 I offer you the memory of a yellow rose seen at sunset,
years before you were born.

```
63 我给你早在你出生前多年的一个傍晚
64 看到的一朵黄玫瑰的记忆。
65
66 I offer you explanationsof yourself,
67 我给你关于你生命的诠释,
68
69 theories about yourself,
70 关于你自己的理论,
71
72 authentic and surprising news of yourself.
73 你的真实而惊人的存在。
74
75 I can give you my loneliness,
76 我给我寂寞,
77
78 my darkness,
79 我的黑暗,
80
81 the hunger of my heart;
82 我心的饥渴,
83
84 I am trying to bribe you with uncertainty, with danger, with defeat.
85 我试图用困惑、危险、失败来打动你。
```

对其进行 MD5 计算，输入./MD5 -c oreo.txt，结果如图3所示

A screenshot of a Kali Linux terminal window. The title bar shows 'kali@kali: ~/Desktop/MD5'. The terminal has a menu bar with 'File', 'Actions', 'Edit', 'View', and 'Help'. The prompt is '(kali@kali)-[~/Desktop/MD5]'. The user enters the command '\$./MD5 -c oreo.txt'. The output is 'MD5("oreo.txt") = 45db2aaa191f274e883759532d8eb3a9'. The prompt returns to '\$'. A large 'KALI' watermark is visible in the background.

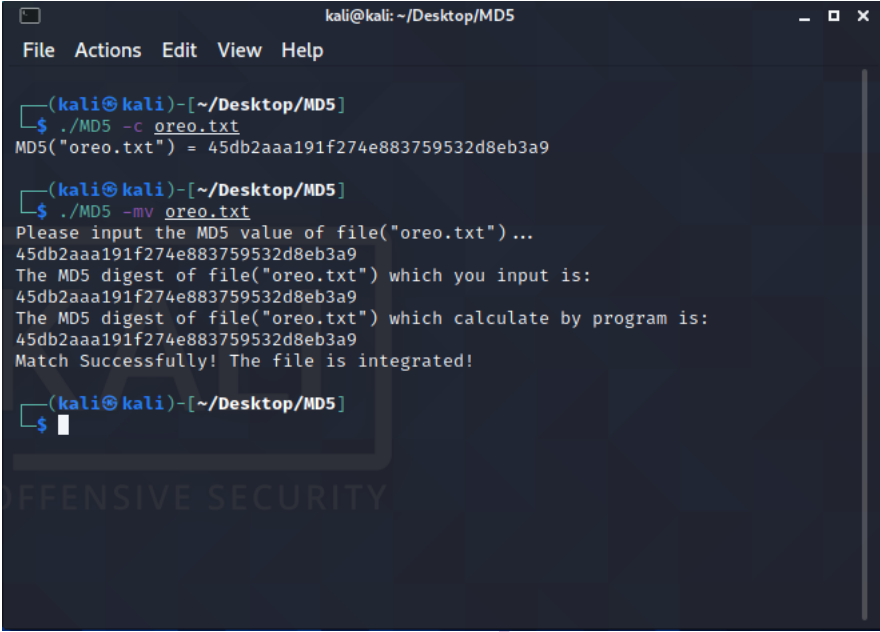
```
kali@kali: ~/Desktop/MD5
File Actions Edit View Help
(kali@kali)-[~/Desktop/MD5]
$ ./MD5 -c oreo.txt
MD5("oreo.txt") = 45db2aaa191f274e883759532d8eb3a9
(kali@kali)-[~/Desktop/MD5]
$
```

图 3: MD5 计算

记录 MD5 摘要为 45db2aaa191f274e883759532d8eb3a9。

手动输入摘要进行文件完整性的检测。

输入./MD5 -mv oreo.txt 结果如图4所示



```
kali@kali: ~/Desktop/MD5
File Actions Edit View Help

(kali@kali)-[~/Desktop/MD5]
$ ./MD5 -c oreo.txt
MD5("oreo.txt") = 45db2aaa191f274e883759532d8eb3a9

(kali@kali)-[~/Desktop/MD5]
$ ./MD5 -mv oreo.txt
Please input the MD5 value of file("oreo.txt") ...
45db2aaa191f274e883759532d8eb3a9
The MD5 digest of file("oreo.txt") which you input is:
45db2aaa191f274e883759532d8eb3a9
The MD5 digest of file("oreo.txt") which calculate by program is:
45db2aaa191f274e883759532d8eb3a9
Match Successfully! The file is integrated!

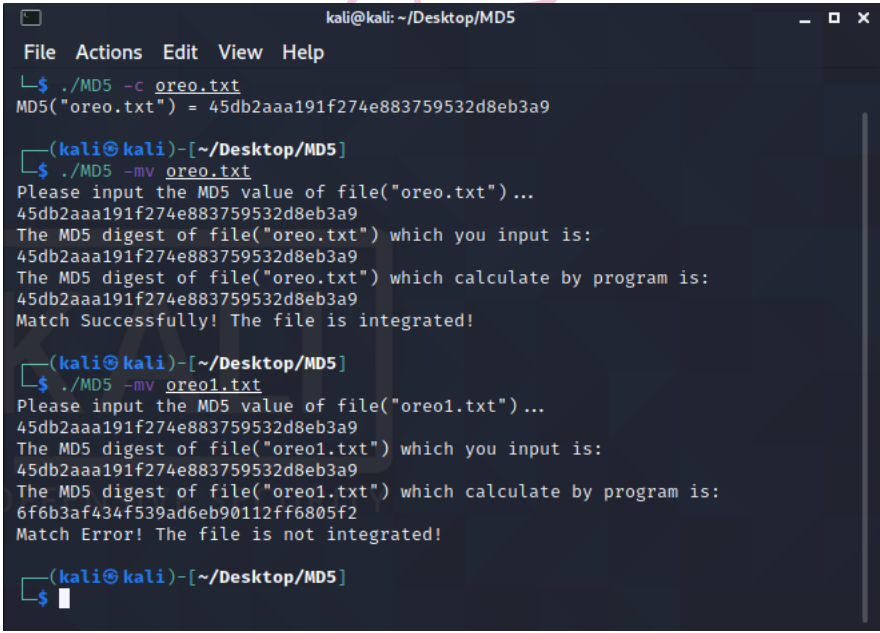
(kali@kali)-[~/Desktop/MD5]
$
```

图 4: MD5 校验

此时文件完整性未被破坏。

创建 oreo1.txt, 在 oreo.txt 的基础上删去最后一小节。

输入./MD5 -mv oreo1.txt 结果如图5所示



```
kali@kali: ~/Desktop/MD5
File Actions Edit View Help

(kali@kali)-[~/Desktop/MD5]
$ ./MD5 -c oreo.txt
MD5("oreo.txt") = 45db2aaa191f274e883759532d8eb3a9

(kali@kali)-[~/Desktop/MD5]
$ ./MD5 -mv oreo.txt
Please input the MD5 value of file("oreo.txt") ...
45db2aaa191f274e883759532d8eb3a9
The MD5 digest of file("oreo.txt") which you input is:
45db2aaa191f274e883759532d8eb3a9
The MD5 digest of file("oreo.txt") which calculate by program is:
45db2aaa191f274e883759532d8eb3a9
Match Successfully! The file is integrated!

(kali@kali)-[~/Desktop/MD5]
$ ./MD5 -mv oreo1.txt
Please input the MD5 value of file("oreo1.txt") ...
45db2aaa191f274e883759532d8eb3a9
The MD5 digest of file("oreo1.txt") which you input is:
45db2aaa191f274e883759532d8eb3a9
The MD5 digest of file("oreo1.txt") which calculate by program is:
6f6b3af434f539ad6eb90112ff6805f2
Match Error! The file is not integrated!

(kali@kali)-[~/Desktop/MD5]
$
```

图 5: MD5 校验

此时文件计算所得 MD5 摘要与记录的 MD5 摘要不匹配, 说明文件完整性遭到破坏。

也可以通过比对文件与文件的 MD5 摘要进行文件完整性的认定。

输入./MD5 -fv oreo.txt oreo1.txt 结果如图6所示

```
kali@kali: ~/Desktop/MD5
File Actions Edit View Help
45db2aaa191f274e883759532d8eb3a9
The MD5 digest of file("oreo.txt") which calculate by program is:
45db2aaa191f274e883759532d8eb3a9
Match Successfully! The file is integrated!

(kali@kali)-[~/Desktop/MD5]
$ ./MD5 -mv oreo1.txt
Please input the MD5 value of file("oreo1.txt")...
45db2aaa191f274e883759532d8eb3a9
The MD5 digest of file("oreo1.txt") which you input is:
45db2aaa191f274e883759532d8eb3a9
The MD5 digest of file("oreo1.txt") which calculate by program is:
6f6b3af434f539ad6eb90112ff6805f2
Match Error! The file is not integrated!

(kali@kali)-[~/Desktop/MD5]
$ ./MD5 -fv oreo.txt oreo1.txt
The MD5 digest of file("offer") which is in file("oreo1.txt") is:
I
The MD5 digest of file("oreo.txt") which calculate by programme is:
45db2aaa191f274e883759532d8eb3a9
Match Error! The file is not integrated!

(kali@kali)-[~/Desktop/MD5]
$
```

图 6: MD5 校验

五、 总结

通过本次实验，我加深了对 MD5 算法基本工作原理的理解，掌握利用了 MD5 算法生成数据摘要的所有计算过程，掌握了 Linux 系统中检测文件完整性的基本方法了，熟悉了 Linux 系统中文件的基本操作方法。