网络安全技术

实
验
报
告

学　　院 计算机学院

年　　级 16 级

班　　级 3 班

学　　号 1613415

姓　　名 潘巧巧

手机号 13347629779

2020 年 6 月 10 日

目录

一、实验目标 1
二、实验内容 1
三、实验步骤 2
四、实验遇到的问题及其解决方法 12
五、实验结论 12

一、实验目的

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。
⑤ 准确地实现 MD5 算法的完整计算过程。
⑥ 对于任意长度的字符串能够生成 128 位 MD5 摘要。
⑦ 对于任意大小的文件能够生成 128 位 MD5 摘要。
⑧ 通过检查 MD5 摘要的正确性来检验原文件的完整性。

二、实验内容

本实验使用 C++语言在 Linux 平台进行编程和运行，代码运行于腾讯云服务器主机。实现了实验要求中关于 MD5 的五项功能，列举如下：

./md5 -h	查看帮助
./md5 -t	打印程序的测试信息
./md5 -c nankai.txt	计算出的被测文件的 MD5 摘要并打印
./md5 -v nankai.txt	验证文件完整性方法一
./md5 -f nankai.txt nankai.md5	验证文件完整性方法二

实验涉及的文件包括：

（1） main.cpp：主函数文件。
（2） helpFunc.h：功能函数定义。
（3） helpFunc.cpp：功能函数实现。
（4） MD5.h：MD5 类定义。
（5） MD5.cpp：MD5 类的部分函数实现。
（6） MD5Update.cpp：MD5 类中 Update 相关函数实现。
（7） MD5codePro.cpp：MD5 类中进制转换、类型变换相关函数实现。
（8） nankai.txt 和 nankai.md5：测试文件
（9） md5：可执行文件

三、实验步骤及实验结果

本次实验步骤：

（1） main.cpp：主函数文件。

通过 unordered_map 数据结构存储各操作的 string 和对应的函数指针。

```
#include "MD5.h"
int main(int argc,char *argv[]) { //argc=外部命令参数的个数，argv[]存放各参
    unordered_map<string, void(*)(int, char*[])> mapOp = {{"-t", print_t},
                                                          {"-h", print_h},
                                                          {"-c", print_c},
                                                          {"-v", print_v},
                                                          {"-f", print_f}};
    if (argc < 2) {
        cout << "参数错误，argc = " << argc << endl;
        return -1;
    }
    string op = argv[1];
    if (mapOp.find(op) != mapOp.end()) {
        mapOp[op](argc, argv);
    }
    return 0;
}
```

（2） helpFunc.h：各功能函数的定义，以及 include 程序需要的一些系统文件。

```
#include <iostream>
#include <string>
#include <string.h>
#include <cmath>
#include <vector>
#include <unordered_map>
#include <fstream>
#include <streambuf>
using namespace std;
void print_h(int argc, char *argv[]); // 打印 help 信息
void print_t(int argc, char *argv[]); // 打印程序的测试信息
void print_c(int argc, char *argv[]); // 计算出的被测文件的 MD5 摘要并打印出来
void print_v(int argc, char *argv[]); // 让用户输入被测文件 MD5，然后重新计算被测文件的
MD5，将两个摘要逐位比较
void print_f(int argc, char *argv[]); // 程序读取.md5 摘要，重新计算被测文件的 MD5，最后
将两者逐位比较
```

（3） helpFunc.cpp：功能函数实现。

各函数根据实验报告的要求调用 MD5 类的功能并打印信息。

```
#include "helpFunc.h"
```

```
#include "MD5.h"
// 打印 help 信息
void print_h(int argc,char *argv[]) {
    if (2 != argc) {
        cout << "参数错误" << endl;
    }
    cout << "usage: " << "\t" << "[-h] --help information " << endl
         << "\t" << "[-t] --test MD5 application" << endl
         << "\t" << "[-c] [file path of the file computed]" << endl
         << "\t" << "\t" <<"--compute MD5 of the given file" << endl
         << "\t" << "[-v] [file path of the file validated]" << endl
         << "\t" << "\t" <<"--validate the integrality of a given file by manual in
put MD5 value" << endl
         << "\t" << "[-f] [file path of the file validated] [file path of the .md5
file]" << endl
         << "\t" << "\t" <<"--validate the integrality of a given file by read MD5
value from .md5 file" << endl;
}

// 打印程序的测试信息
void print_t(int argc,char *argv[]) {
    if (2 != argc) {
        cout << "参数错误" << endl;
    }
    vector<string> str = {"", "a", "abc", "message digest", "abcdefghijklmnopqrstuv
wxyz", "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789", "123456789
012345678901234567890123456789012345678901234567890123456789012345678901234567890"};
    MD5 md5;
    for (int i = 0; i < str.size(); ++i) {
        md5.Update(str[i]);
        cout << "MD5(\"" + str[i] + "\") = " << md5.Tostring() << endl;
    }
}

// 计算出的被测文件的 MD5 摘要并打印
void print_c(int argc,char *argv[]) {
    if (3 != argc) {
        cout << "参数错误" << endl;
    }
    string filePath = argv[2];
    ifstream fileStream(filePath);
    MD5 md5;
    md5.Update(fileStream);
```

```
    cout << "The MD5 value of file(\"" << filePath << "\") is " << md5.Tostring() <
< endl;
}

// 让用户输入被测文件的 MD5 摘要，然后重新计算被测文件的 MD5 摘要，将两个摘要逐位比较
void print_v(int argc,char *argv[]) {
    if (3 != argc) {
        cout << "参数错误" << endl;
    }
    string filePath = argv[2];
    cout << "Please input the MD5 value of file(\"" << filePath << "\")..." << endl
;
    string inputMD5;
    cin >> inputMD5;
    cout << "The old MD5 value of file(\"" << filePath << "\") you have input is" <
< endl << inputMD5 << endl;
    ifstream fileStream(filePath);
    MD5 md5;
    md5.Update(fileStream);
    string genMD5 = md5.Tostring();
    cout << "The new MD5 value of file(\"" << filePath << "\") that has computed is
" << endl << genMD5 << endl;
    if (!genMD5.compare(inputMD5)) {
        cout << "OK! The file is integrated" << endl;
    }
    else {
        cout << "Match Error! The file has been modified!" << endl;
    }
}

// 程序读取.md5 摘要，重新计算被测文件的 MD5，最后将两者逐位比较
void print_f(int argc, char *argv[]) {
    if (4 != argc) {
        cout << "参数错误" << endl;
    }
    string filePath = argv[2];
    string md5Path = argv[3];

    ifstream md5Stream(md5Path);
    string oldMD5Str((istreambuf_iterator<char>(md5Stream)), istreambuf_iterator<ch
ar>());
    cout << "The old MD5 value of file(\"" << filePath << "\") in " << md5Path << "
 is " << endl << oldMD5Str << endl;
```

```
    ifstream fileStream(filePath);
    MD5 md5;
    md5.Update(fileStream);
    string genMD5 = md5.Tostring();
    cout << "The new MD5 value of file(\"" << filePath << "\") that has computed is
" << endl << genMD5 << endl;
    if (!genMD5.compare(oldMD5Str)) {
        cout << "OK! The file is integrated" << endl;
    }
    else {
        cout << "Match Error! The file has been modified!" << endl;
    }
}
```

（4） MD5.h：MD5 类定义，包括一些运算的 define。

```
#include "helpFunc.h"

#define F(x, y, z) (((x) & (y)) | ((~x) & (z)))              //F 函数
#define G(x, y, z) (((x) & (z)) | ((y) & (~z)))              //G 函数
#define H(x, y, z) ((x) ^ (y) ^ (z))                         //H 函数
#define I(x, y, z) ((y) ^ ((x) | (~z)))                      //I 函数
#define ROTATE_LEFT(x, n) (((x) << (n)) | ((x) >> (32-(n)))) //32 位数字 x 的循环左移 n 位操作

#define FF(a, b, c, d, x, s, ac) { (a) += F ((b), (c), (d)) + (x) + ac; (a) = ROTATE_LEFT ((a)
, (s)); (a) += (b); }
#define GG(a, b, c, d, x, s, ac) { (a) += G ((b), (c), (d)) + (x) + ac; (a) = ROTATE_LEFT ((a)
, (s)); (a) += (b); }
#define HH(a, b, c, d, x, s, ac) { (a) += H ((b), (c), (d)) + (x) + ac; (a) = ROTATE_LEFT ((a)
, (s)); (a) += (b); }
#define II(a, b, c, d, x, s, ac) { (a) += I ((b), (c), (d)) + (x) + ac; (a) = ROTATE_LEFT ((a)
, (s)); (a) += (b); }

#define T(i) 4294967296 * abs(sin(i))

/* 主要功能：
1、为任意长度的字符串生成 MD5 摘要
2、为任意大小的文件生成 MD5 摘要
3、利用 MD5 摘要验证文件的完整性
*/
class MD5 {
public:
    void Update(const string &str); // 对给定长度的字符串进行 MD5 运算
    void Update(ifstream &in); // 对文件中的内容进行 MD5 运算
    string Tostring(); // 将 MD5 摘要以字符串形式输出
```

```
private:
    void Reset(); // 重置初始变量
    void Update(vector<uint8_t> input); // 对给定长度的字节流进行 MD5 运算
    void Transform(const vector<uint8_t> block); // 对一个 512 比特的消息分组进行 MD5 运算
    vector<uint32_t> Decode(const vector<uint8_t>input); // 将 64byte 的数据块划分为 16 个 32bit 大小的子分组
    string from10To16(uint32_t decimal); //32 位十进制转成 16 进制，用 8 个字母的 string 表示
    vector<uint8_t> fromInt64ToInt8Vec(uint64_t num); // 将 1 个 64 位 int 转成 vector<uint8_t>

private:
    uint32_t state[4]; // 用于表示 4 个初始向量
};
```

（5） MD5.cpp：MD5 类的部分功能实现，主要为 Reset 函数，Tostring 函数，Transform 函数。

```
#include "MD5.h"

void MD5::Reset() {
    state[0] = 0x67452301;
    state[1] = 0xefcdab89;
    state[2] = 0x98badcfe;
    state[3] = 0x10325476;
}

string MD5::Tostring() {
    string strResult;
    for(int i = 0; i < 4; ++i) {
        strResult += from10To16(htobe32(state[i]));
    }
    return strResult;
}

void MD5::Transform(const vector<uint8_t> block) {
    uint32_t a = state[0], b = state[1], c = state[2], d = state[3];
    vector<uint32_t> x = Decode(block);
    /* 第 1 轮 */
    FF(a, b, c, d, x[0],  7,  T(1));
    FF(d, a, b, c, x[1],  12, T(2));
    FF(c, d, a, b, x[2],  17, T(3));
    FF(b, c, d, a, x[3],  22, T(4));
    FF(a, b, c, d, x[4],  7,  T(5));
    FF(d, a, b, c, x[5],  12, T(6));
    FF(c, d, a, b, x[6],  17, T(7));
```

```
    FF(b, c, d, a, x[7],  22, T(8));
    FF(a, b, c, d, x[8],  7,  T(9));
    FF(d, a, b, c, x[9],  12, T(10));
    FF(c, d, a, b, x[10], 17, T(11));
    FF(b, c, d, a, x[11], 22, T(12));
    FF(a, b, c, d, x[12], 7,  T(13));
    FF(d, a, b, c, x[13], 12, T(14));
    FF(c, d, a, b, x[14], 17, T(15));
    FF(b, c, d, a, x[15], 22, T(16));

    /* 第 2 轮 */
    GG (a, b, c, d, x[1],  5,  T(17));
    GG (d, a, b, c, x[6],  9,  T(18));
    GG (c, d, a, b, x[11], 14, T(19));
    GG (b, c, d, a, x[0],  20, T(20));
    GG (a, b, c, d, x[5],  5,  T(21));
    GG (d, a, b, c, x[10], 9,  T(22));
    GG (c, d, a, b, x[15], 14, T(23));
    GG (b, c, d, a, x[4],  20, T(24));
    GG (a, b, c, d, x[9],  5,  T(25));
    GG (d, a, b, c, x[14], 9,  T(26));
    GG (c, d, a, b, x[3],  14, T(27));
    GG (b, c, d, a, x[8],  20, T(28));
    GG (a, b, c, d, x[13], 5,  T(29));
    GG (d, a, b, c, x[2],  9,  T(30));
    GG (c, d, a, b, x[7],  14, T(31));
    GG (b, c, d, a, x[12], 20, T(32));

    // /* 第 3 轮 */
    HH (a, b, c, d, x[5],  4,  T(33));
    HH (d, a, b, c, x[8],  11, T(34));
    HH (c, d, a, b, x[11], 16, T(35));
    HH (b, c, d, a, x[14], 23, T(36));
    HH (a, b, c, d, x[1],  4,  T(37));
    HH (d, a, b, c, x[4],  11, T(38));
    HH (c, d, a, b, x[7],  16, T(39));
    HH (b, c, d, a, x[10], 23, T(40));
    HH (a, b, c, d, x[13], 4,  T(41));
    HH (d, a, b, c, x[0],  11, T(42));
    HH (c, d, a, b, x[3],  16, T(43));
    HH (b, c, d, a, x[6],  23, T(44));
    HH (a, b, c, d, x[9],  4,  T(45));
    HH (d, a, b, c, x[12], 11, T(46));
    HH (c, d, a, b, x[15], 16, T(47));
```

```
    HH (b, c, d, a, x[2],  23, T(48));

    // /* 第 4 轮 */
    II (a, b, c, d, x[0],  6,  T(49));
    II (d, a, b, c, x[7],  10, T(50));
    II (c, d, a, b, x[14], 15, T(51));
    II (b, c, d, a, x[5],  21, T(52));
    II (a, b, c, d, x[12], 6,  T(53));
    II (d, a, b, c, x[3],  10, T(54));
    II (c, d, a, b, x[10], 15, T(55));
    II (b, c, d, a, x[1],  21, T(56));
    II (a, b, c, d, x[8],  6,  T(57));
    II (d, a, b, c, x[15], 10, T(58));
    II (c, d, a, b, x[6],  15, T(59));
    II (b, c, d, a, x[13], 21, T(60));
    II (a, b, c, d, x[4],  6,  T(61));
    II (d, a, b, c, x[11], 10, T(62));
    II (c, d, a, b, x[2],  15, T(63));
    II (b, c, d, a, x[9],  21, T(64));

    state[0] += a;
    state[1] += b;
    state[2] += c;
    state[3] += d;
}
```

（6） MD5Update.cpp：MD5 类中 Update 相关函数，包括 2 个 public 函数和 1 个 private 函数。

```
#include "MD5.h"

//对给定长度的字符串进行 MD5 运算
void MD5::Update(const string &str) {
    Reset();
    // 先将输入转化为标准字节流，再调用私有函数 Update
    vector<uint8_t> input;
    for (int i = 0; i < str.size(); ++i) {
        input.push_back(str[i]);
    }
    Update(input);
}

// 对文件中的内容进行 MD5 运算
void MD5::Update(ifstream &in) {
    Reset();
```

```
    string str((istreambuf_iterator<char>(in)), istreambuf_iterator<char>());
    vector<uint8_t> input;
    for (int i = 0; i < str.size(); ++i) {
        input.push_back(str[i]);
    }
    Update(input);
}

// 私有函数 Update
// 对长为 length 的字节流进行预处理，然后再调用 transform 函数对每一个 64byte（512bit）数据块进行计算
void MD5::Update(vector<uint8_t> input) {
    // 消息填充：N * 512 + 448 （N 为正整数）
    vector<uint8_t> trueLen = fromInt64ToInt8Vec(input.size() * 8);// 真实长度，trueLen.size()=8
    vector<uint8_t> fillHelp(64, (uint8_t)0); // 最多填充 512bit=64*8
    fillHelp[0] = (uint8_t)128;

    if (input.size() * 8 % 512 == 448) {
        input.insert(input.end(), fillHelp.begin(), fillHelp.end());
    }
    else {
        int index = 0;
        while(input.size() * 8 % 512 != 448) {
            input.push_back(fillHelp[index++]);
        }
    }
    input.insert(input.end(), trueLen.begin(), trueLen.end());

    // 开始 MD5 运算，每次 512bit=8*64
    int transformTime = input.size() / 64;
    for (int i = 0; i < transformTime; ++i) {
        vector<uint8_t> md5input;
        md5input.insert(md5input.end(), input.begin() + i * 64, input.begin() + (i + 1) * 64);
        // 包含参数 2，不包含参数 3
        Transform(md5input);
    }
}
```

（7） MD5codePro.cpp：MD5 类定义中进制转换、类型变换相关函数实现。

```
#include "MD5.h"
// 将 64byte（64*8 bit）的数据块划分为 16 个 32bit 大小的子分组，input.size()=64
vector<uint32_t> MD5::Decode(const vector<uint8_t>input) {
```

```
    // input 的 4 个 8bit 数字合并成一个 output 的 32bit 数字，但是要反过来，ABCD->DCBA
    vector<uint32_t> output;
    for (int i = 0; i < input.size() / 4; ++i) {
        uint32_t temp = 0;
        for (int j = 3; j >= 0; --j) {
            temp += input[i * 4 + j];
            if (j != 0) {
                temp = temp << 8;
            }
        }
        output.push_back(temp);
    }
    return output;
}

// decimal 是 32 位十进制的表示，转成 16 进制，用 8 个字母的 string 表示
string MD5::from10To16(uint32_t decimal) {
    string hex;
    uint32_t help = 4026531840;
    unordered_map<int, string> map10To16 = { {0, "0"},  {1, "1"},  {2, "2"},  {3, "3"},
                                             {4, "4"},  {5, "5"},  {6, "6"},  {7, "7"},
                                             {8, "8"},  {9, "9"},  {10, "a"}, {11, "b"},
                                             {12, "c"}, {13, "d"}, {14, "e"}, {15, "f"} };
    for (int i = 0; i < 8; ++i) {
        int tempResult = (decimal & help) >> 28;
        hex += map10To16[tempResult];
        decimal = decimal << 4;
    }
    return hex;
}

// 将 1 个 64 位 int 转成 vector<uint8_t>
vector<uint8_t> MD5::fromInt64ToInt8Vec(uint64_t num) {
    vector<uint8_t> result(8);
    uint8_t help = 255;
    for (int i = 0; i < 8; ++i) {
        result[i] = num & help;
        num = num >> 8;
    }
    return result;
}
```

本次实验结果：

（1）./md5 -h

```
[root@VM_0_13_centos hw3Catnip]# ./md5 -h
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 integrality of a given file by manual input MD5 value
        [-f] [file path of the file validated] [file path of the .md5 file]
                --validate the integrality of a given file by read MD5 value from .md5 file
```

（2）./md5 -t

```
[root@VM_0_13_centos hw3Catnip]# ./md5 -t
MD5("") = d41d8cd98f00b204e9800998ecf8427e
MD5("a") = 0cc175b9c0f1b6a831c399e269772661
MD5("abc") = 900150983cd24fb0d6963f7d28e17f72
MD5("message digest") = f96b697d7cb7938d525a2f31aaf161d0
MD5("abcdefghijklmnopqrstuvwxyz") = c3fcd3d76192e4007dfb496cca67e13b
MD5("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789") = d174ab98d277d9f5a5611c2c9f419d9f
MD5("12345678901234567890123456789012345678901234567890123456789012345678901234567890") = 57edf4a22be3c955ac49da2e2107b67a
```

（3）./md5 -c nankai.txt

```
[root@VM_0_13_centos hw3Catnip]# ./md5 -c nankai.txt
The MD5 value of file("nankai.txt") is 900150983cd24fb0d6963f7d28e17f72
```

（4）./md5 -v nankai.txt

```
[root@VM_0_13_centos hw3Catnip]# ./md5 -v nankai.txt
Please input the MD5 value of file("nankai.txt")...
1234
The old MD5 value of file("nankai.txt") you have input is
1234
The new MD5 value of file("nankai.txt") that has computed is
900150983cd24fb0d6963f7d28e17f72
Match Error! The file has been modified!
[root@VM_0_13_centos hw3Catnip]# ./md5 -v nankai.txt
Please input the MD5 value of file("nankai.txt")...
900150983cd24fb0d6963f7d28e17f72
The old MD5 value of file("nankai.txt") you have input is
900150983cd24fb0d6963f7d28e17f72
The new MD5 value of file("nankai.txt") that has computed is
900150983cd24fb0d6963f7d28e17f72
OK! The file is integrated
```

（5）./md5 -f nankai.txt nankai.md5

```
[root@VM_0_13_centos hw3Catnip]# ./md5 -f nankai.txt nankai.md5
The old MD5 value of file("nankai.txt") in nankai.md5 is
900150983cd24fb0d6963f7d28e17f72
The new MD5 value of file("nankai.txt") that has computed is
900150983cd24fb0d6963f7d28e17f72
OK! The file is integrated
```

四、实验遇到的问题及其解决方法

本次实验中遇到的问题和解决方法列举如下：

（1）问题：大小端编址问题。

解决方法：遇到大小端编址问题，需要仔细检查，如 MD5 算法中有四个 32 比特的初始向量。它们分别是：A=0x01234567，B=0x89abcdef，C=0xfedcba98，D=0x76543210。实际赋值的时候应该赋值为 A=0x67452301，B=0xefcdab89，C=0x98badcfe，D=0x10325476。

五、实验结论

学习了 MD5 的原理和实现过程

学习了大端编址和小端编址

学习了在 C++的 STL 中存放函数指针

学习了 main()函数中的 argv 和 argc 参数的使用