网络安全技术 —— 基于MD5算法的文件完整性校验程序

学号：2013921
姓名：周延霖
专业：信息安全

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：

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

本章编程训练的要求如下:

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

二、实验内容

在 Linux 平台下编写应用程序，正确地实现 MD5 算法

要求程序不仅能够为任意长度的字符串生成 MD5 摘要，而且可以为任意大小的文件生成 MD5 摘要。同时，程序还可以利用 MD5 摘要验证文件的完整性

验证文件完整性分为两种方式：

1. 一种是在手动输入MD5摘要的条件下，计算出当前被测文件的MD5摘要，再将两者进行比对。若相同，则文件完好；否则，文件遭到破坏
2. 先利用Linux系统工具md5sum为被测文件生成一个后缀为.md5的同名文件，然后让程序计算出被测文件的MD5摘要，将其与.md5文件中的MD5摘要进行比较，最后得出检测结果

三、实验步骤

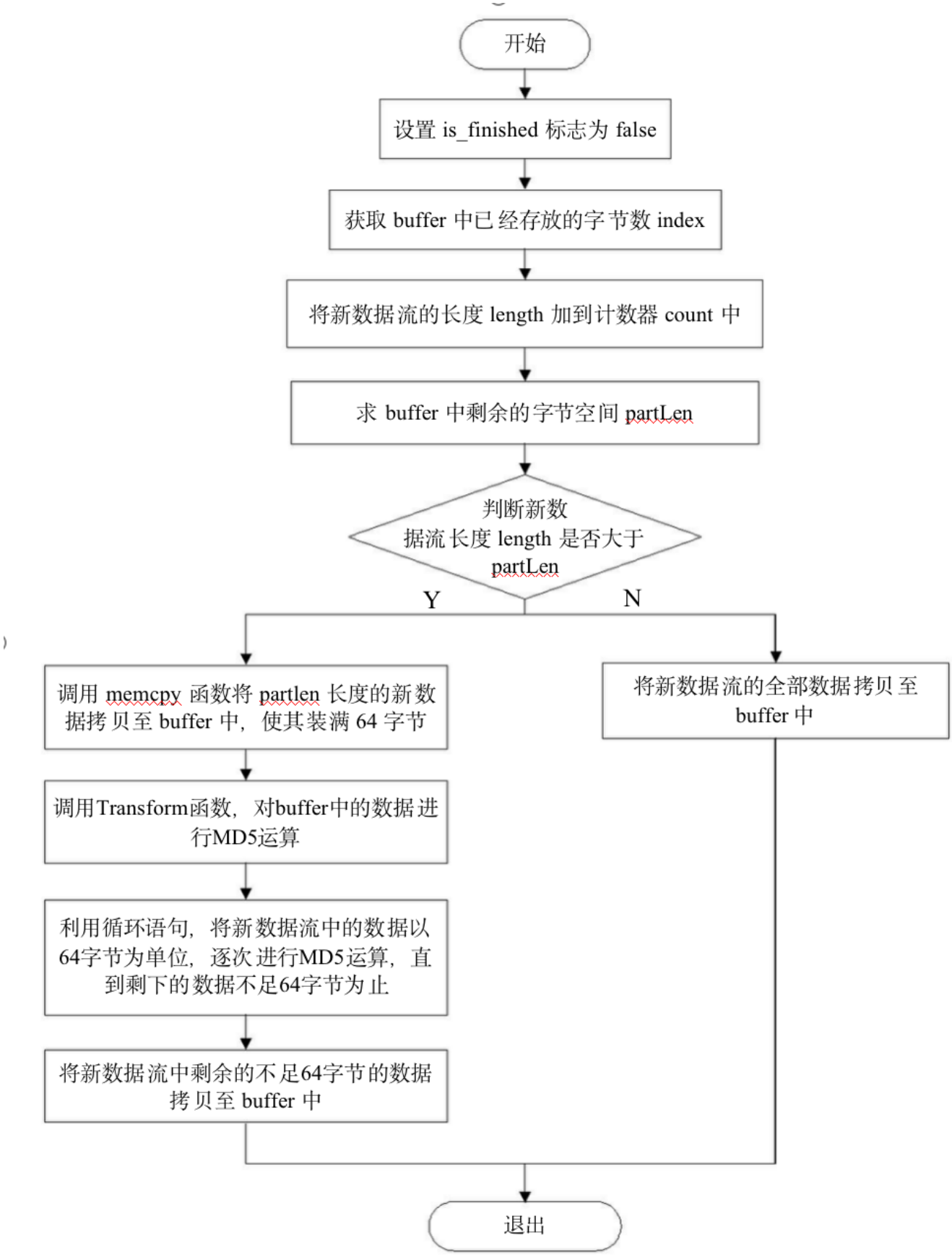
参考实验指导书进行以下实验

1、设计update函数

update函数大致流程如下：

1. 将标志is_finished 设为false，表示MD5 运算正在进行
2. 将计数器count 右移3 位后再截取后6 位，获得buffer 中已经存放的字节数
3. 更新计数器 count，将新数据流的长度加入计数器中。需要注意的是计数器 count中保存的是比特数（等于字节数 * 8）。count[0]保存的是数值的低32 位，count[1]保存的是高32 位
4. 求出buffer 中的剩余字节数partLen
5. 判断新数据流的长度length 是否大于partLen，如果length 大于partLen，将parLen长度的新数据拷贝至buffer 中，使其填满64 字节，然后调用Transform 函数对buffer 中的数据块进行MD5 运算。接着利用循环将新数据流中的数据以64 字节为单位，逐次进行MD5运算，直到剩余数据不足64 字节为止，最后将新数据流中不足64 字节的数据拷贝至buffer中。如果length 不大于partLen，将新数据流的全部数据拷贝至buffer 中即可

流程图如下所示：



最终针对三种不同的输入方式设计如下三种`update`函数

- 字节流

```
// 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的数据块进行计算
void MD5::Update(vector<uint8_t> input) {
    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
    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);
        Transform(md5input);
    }
}
```

- 字符流

```
//对给定长度的字符串进行MD5运算
void MD5::StrUpdate(const string& str) {
    Reset();
    // 首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);
}
```

- 文件流

```
// 对文件中的内容进行MD5运算
void MD5::FileUpdate(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);
}
```

2、设计transform函数

Transform 函数构造了MD5 摘要的计算过程。Transform 函数的的执行流程分为以下4 步:

1. 首先将初始向量state 的数值赋给变量a、b、c、d 中
2. 调用 Decode 函数，将 64 字节的数据块划分为 16 个 32 比特大小的子分组。因为每一轮计算都是对 32 比特子分组进行操作，所以重新划分后可以方便后面的计算过程
3. 依次调用函数FF、GG、HH、II 展开4 轮计算，其中每一轮计算包含16 小步，每一步对一个32 比特子分组进行运算。函数FF、GG、HH、II 的前4 个参数是变量a、b、c、d 的不同排列，参数X[k]表示对第k 个子分组进行计算，Sij 表示第i 轮第j 步计算循环左移的位数，最后一个常数T[i]表示4294967296 * abs(sin(i))的整数部分
4. 最后将变量a、b、c、d 中的运算结果加到初始向量state 上

最终设计出如下transform函数:

```
// 对一个512比特的消息分组进行MD5运算
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));

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

3、文件完整性检验的设计与实现

文件完整性检验是在main 函数中实现的。应用程序为用户提供了多个选项，不但可以在命令行下计算文件的MD5摘要，验证文件的完整性，还可以显示程序的帮助信息和MD5算法的测试信息

在main 函数中，程序通过区分参数argv[1]的不同值来启动不同的工作流程:

- 如果argv[1]等于“-h”，表示显示帮助信息
- 如果argv[1]等于“-t”，表示显示测试信息
- 如果argv[1]等于“-c”，表示计算被测文件的MD5摘要
- 如果argv[1]等于“-v”，表示根据手工输入的MD5摘要验证文件完整性
- 如果argv[1]等于“-f”，表示根据.md5文件中的摘要验证文件完整性

帮助信息可以协助用户快速地了解命令行输入格式。测试信息可以让用户验证MD5算法的正确性。用于测试的消息都是MD5 算法官方文档（RFC1321）中给出的例子，如果计算的结果相同，则充分说明程序的MD5运算过程是正确无误的

程序提供了两种验证文件完整性的方式:

1. 是让用户手工输入被测文件的MD5 摘要，然后调用 MD5 类的运算函数重新计算被测文件的 MD5 摘要，最后将两个摘要逐位进行比较，进而验证文件的完整性
2. 从与被测文件对应的.md5 文件中读取MD5 摘要，然后调用MD5 类的运算函数重新计算被测文件的MD5 摘要，最后将两个摘要逐位进行比较，进而验证文件的完整性

本次实验针对五个参数设置了不同的函数:

> (一)帮助信息

```
// 输出help帮助信息
void Help_Message(int argc, char* argv[]) {
    if (argc != 2)
        cout << "参数错误!" << endl;
```

```
    cout << "usage: " << "\t" << "[-h] --help information " << endl;
    cout << "\t" << "[-t] --test MD5 application" << endl;
    cout << "\t" << "[-c] [file path of the file computed]" << endl;
    cout << "\t" << "\t" << "--compute MD5 of the given file" << endl;
    cout << "\t" << "[-v] [file path of the file validated]" << endl;
    cout << "\t" << "\t" << "--validate the integrality of a given file by
manual input MD5 value" << endl;
    cout << "\t" << "[-f] [file path of the file validated] [file path of
the .md5 file]" << endl;
    cout << "\t" << "\t" << "--validate the integrality of a given file by
read MD5 value from .md5 file" << endl;
}
```

(二)测试信息

```
// 输出程序的测试信息
void Test_Message(int argc, char* argv[]) {
    if (argc != 2)
        cout << "参数错误!" << endl;
    vector<string> str = { "", "a", "abc", "message digest",
"abcdefghijklmnopqrstuvwxyz",
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789",
"12345678901234567890123456789012345678901234567890123456789012345678901234
4567890","周延霖和潘红forever!" };
    MD5 md5;
    for (int i = 0; i < str.size(); ++i) {
        md5.StrUpdate(str[i]);
        cout << "MD5(\"" + str[i] + "\") = " << md5.Tostring() << endl;
    }
}
```

(三)计算摘要

```
// 计算出的被测文件的MD5摘要并输出
void Copy_Message(int argc, char* argv[]) {
    if (argc != 3)
        cout << "参数错误!" << endl;
    string filePath = argv[2];
    ifstream fileStream(filePath);
    MD5 md5;
    md5.FileUpdate(fileStream);
    cout << "The MD5 value of file(\"" << filePath << "\") is " <<
md5.Tostring() << endl;
}
```

(四)手工输入

```
// 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，最后确认文件是否被修改
void Validsure_Message(int argc, char* argv[]) {
    if (argc != 3)
        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.FileUpdate(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 << "Match Error! The file has been modified!" << endl;
    else
        cout << "OK! The file is integrated" << endl;
}
```

(五).md5文件

```
// 程序读取.md5摘要，并重新计算被测文件的MD5，最后将两者逐位比较，最后确定文件是否损坏
void Filesure_Message(int argc, char* argv[]) {
    if (argc != 4)
        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<char>());
    cout << "The old MD5 value of file(\"" << filePath << "\") in " << md5Path << " is " << endl << oldMD5Str << endl;

    ifstream fileStream(filePath);
    MD5 md5;
    md5.FileUpdate(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 << "Match Error! The file has been modified!" << endl;
    else
        cout << "OK! The file is integrated" << endl;
}
```

四、实验结果

接下来展示本次实验的几种结果：

1、帮助信息

```
D:\zyl_part\netsecurity\RSA-main\zyl-client\Debug>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、计算摘要

```
D:\zyl_part\netsecurity\RSA-main\zyl-client\Debug>MD5 -c 你好.txt
The MD5 value of file("你好.txt") is 5712ce4ddf9b56d288888848ffffffc1
```

3、验证文件完整性

```
D:\zyl_part\netsecurity\RSA-main\zyl-client\Debug>MD5 -f 你好.txt 你好.md5
The old MD5 value of file("你好.txt") in 你好.md5 is
5712ce4ddf9b56d288888848ffffffc1
The new MD5 value of file("你好.txt") that has computed is
5712ce4ddf9b56d288888848ffffffc1
OK! The file is integrated
```

五、心得体会

在本次实验中，首先复习了上个学期编写MD5加密的过程并应用，也学习了如何将其用到验证文件完整性的操作以及其细节，对上学期密码学的知识也算进行了一个回顾

最后通过真正的MD5加密程序对所学到的理论知识进行相应的应用，对密码学相关编程也更加的熟练，期待自己未来更好的发展，心想事成、万事胜意、未来可期