MD5 实验报告

16340063 巩泽群

一、算法原理概述

MD5 算法是一种哈希算法，MD5 将任意不定长的消息作为输入，进行哈希变换之后产生 128bits 的输出。MD5 算法是一种分块的算法，它以 512bits 为一块，末尾不足 512bits 的部分要进行填充，然后按序以每个块作为输入进行迭代。MD5 存在一个 128-bit 的缓冲区，该缓冲区会被初始化。每个 512-bit 块都会和当前的缓冲区内容进行运算，然后得到新的 128-bit 的缓冲区内容。

1、 填充（padding）

填充的主要目的是要把源消息填充为完整的 512-bit 块，因此最后不足 512-bit 的部分要按照一定的规则进行填充。其具体的规则为：

- **首先**在长度为 K bits 的消息后附加长度为 P = 448 - (K mod 512)个 bit 的 100000…0000（即 K+P≡448(mod 512)，**注意**，P 应当满足 1⩽P⩽512，因此如果计算出来 P=0 时，应当使得 P=512）；
- **然后**将 K 值的后 64 位按照小端模式附加在源消息的最后 64 位；
- **最后**将源消息补全为一个长度为 K+P+64≡0(mod 512)，即长度可以 512 的整数倍的消息。

例如：

Before padding.

01001000 01100101 01101100 01101100 01101111 **0**0000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

After padding.

01001000 01100101 01101100 01101100 01101111 **1**0000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00101000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

源消息为 Hello，使用 ASCII 表示为 48 65 6C 6C 6F，即上文的前 5 个字节；

第一个加粗部分是置为 10000…000 的开始，直到最后一行；

最后一行即 64 bits，按照小端模式解释这样的一个 64 位无符号整数，其值为 0x0000000000000028=40，

即源消息 Hello=>5bytes=40bits。

2、 分块

分块即将源消息分割成 512-bit 的分组，在实际实现中，我们选择每次读入 512bits 的内容，将其加密，然后再读入下一 512-bit 的分组。

每个 512-bit 的分组可以再次被划分为 16 个 32-bit 的分组（也即 16 个 32 位字），考虑到循环压缩过程中我们需要使用 32-bit 的字进行运算，因此我们选择将每个 32 位字记录在一个无符号整型(unsigned int)中。

3、 初始化

MD5 算法具有一个 128bit 的缓冲区，将其记为 4 个 32 位字（A,B,C,D）。每个字分别初始化为

- $A = 0x67452301$
- $B = 0xEFCDAB89$
- $C = 0x98BADCFE$
- $D = 0x10325476$

应当注意的是，MD5 算法要求使用小端模式存储，因此该数实际上在内存中存储为：

01	23	45	67
低地址			高地址

4、 循环压缩

对每个 512-bit 的分组都进行如下压缩过程：

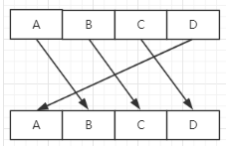
首先，对于每个分组记录下当前的 A、B、C、D 值。

然后开始进行 **4 轮循环**，每轮循环都对于当前分组的 16 个 32 位字进行如下的 16 轮迭代操作：

1) **对 A 迭代：A = B + ((A + g(B, C, D) + X[k] + T[i]) <<<s)**

- g 是轮函数中的一个，分别按照当前的循环轮数分别选择如下的一个
  第 1 轮：`F(b,c,d)=(b&c)|(~b&d)`
  第 2 轮：`G(b,c,d)=(b&d)|(c&~d)`
  第 3 轮：`H(b,c,d)=b^c^d`
  第 4 轮：`I(b,c,d)=c^(b|~d)`
- X[k]为当前操作分组里的第 k 个 32bit 分组；
- T[i]根据 T 表选择，i 为当前进行的是第 i 次迭代运算（四轮循环共有 4*16=64 次迭代运算）；
- <<< 表示循环左移操作；
- s 值根据给出的 s 表进行选择，为循环左移的位数；

2) **缓冲区（A, B, C, D）作循环轮换**：



如图所示，进行轮换。

在四轮循环结束后，要把当前的 A、B、C、D 与之前我们记录下来的初始的 A、B、C、D 值分别相加，得到最终的 A、B、C、D 值，并将其作为最终的结果进行输出。

二、程序结构

我为当前的 C++程序写了一个 MD5 类，将 MD5 的加密方法封装在类里，提高了可复用性。

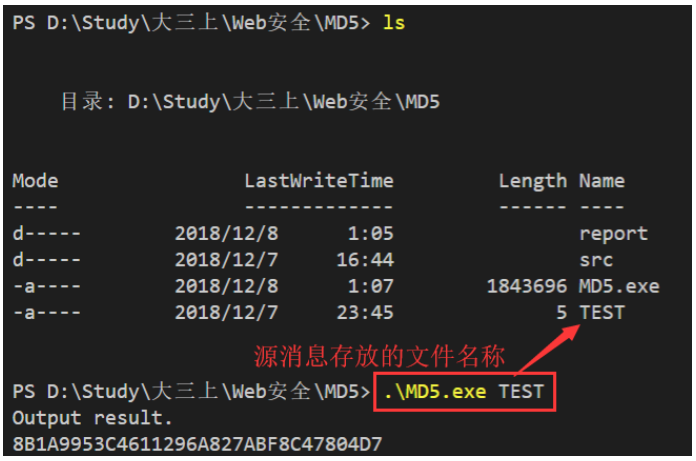
下面给出主要伪码:

```
Function encrypt (file path)
Begin
    While (left file size > 64 bytes)
        read 64 bytes from file
        compress the 512-bit block
    End While

    read the left file content

    If left nothing
        padding an empty block
    Else
        padding to 512-bit block
    End If

    compress the last 1 or 2 512-bit block(s)

    output the result in hex
End
```

```
Function compress (char src[64])
Begin
    Transform the 64 chars into 16 unsigned int (by memcpy)
    For i from 0 to 3
        Choose a function to generate k.
        For j from 0 to 15
            A = B + ((A + g(B, C, D) + X[k] + T[i]) <<<s)
            Cycle the value of A, B, C, D
        End For
    End For
    A = A + origin A
    ...
    D = D + origin D
End
```

三、程序使用

已编写 MD5.cpp 并编译生成可执行文件 MD5.exe，使用了命令行参数，使用方式如下：



如图所示，运行程序后参数部分要输入源文件名称。