MD5算法 >

MD5消息摘要算法(Message-Digest Algorithm)是一种被广泛使用的密码散列函数，对于任意长度的数据可以产生出一个128位(16字节)的散列值，用于确保信息传输完整一致。MD5为文件传输提供可靠性，例如服务器预先提供一个MD5校验和，用户下载完文件以后，用MD5算法计算下载文件的MD5校验和，然后通过检查这两个校验和是否一致，就能判断下载的文件是否出错。MD5的RFC文档

算法步骤 >

1.填充位

对数据进行填充，使得它的长度(按bit算)模512等于448(除以512余448)。首先在数据末尾填充一个二进制位1，接着填充二进制位0直到消息的位长模512等于448。如果数据一开始位长模512就等于448，仍要进行填充，即填充一个'1'和511个'0'。

填充完二进制位后，将数据末填充前的位长用64位整数表示(小端规则，即高位地址存高位字节)，将这64位填充到数据末尾。此时填充后的数据的位长刚好是512的倍数。

2.初始化

- 定义4个32位整数，在内存中的16进制表示如下:

整数A：01 23 45 67
整数B：89 ab cd ef
整数C：fe dc ba 98
整数D：76 54 32 10
因为内存中的数据存储是小端规则，所以4个整数的值为：
$A = 0x67452301$
$B = 0xefcdab89$
$C = 0x98badcfe$
$D = 0x10325476$

- 定义4个非线性函数：

$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$
$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$
$H(X,Y,Z) = X \oplus Y \oplus Z$
$I(X,Y,Z) = Y \oplus (X \vee \neg Z)$
$\oplus, \wedge, \vee, \neg$是$XOR, AND, OR, NOT$的符号。

- 定义一个数组

$T[i] = abs(2^{32}\sin(i)), i = 1 \ldots 64, abs$表示绝对值

3.四轮操作

将填充后的数据M按512位(64字节)分成一组，每组64个字节中每4个字节组成一个32位整数存进数组X[0...15]，共16个整数；定义<<<为循环左移操作，对每个分组进行四轮每轮16次操作，每个分组操作的结果(A,B,C,D的值)都会影响下一分组。

```
/* 64个字节分为一组. */
For i = 0 to M/64 - 1 do
  AA = A
  BB = B
  CC = C
  DD = D

  每次读取4个字节组成一个32位整数存进数组X[0...15]
  /* 第一轮，[abcd k s i]定义操作
    a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s).
    执行16次操作
  */
  [ABCD  0  7  1] [DABC  1 12  2] [CDAB  2 17  3] [BCDA  3 22  4]
  [ABCD  4  7  5] [DABC  5 12  6] [CDAB  6 17  7] [BCDA  7 22  8]
  [ABCD  8  7  9] [DABC  9 12 10] [CDAB 10 17 11] [BCDA 11 22 12]
  [ABCD 12  7 13] [DABC 13 12 14] [CDAB 14 17 15] [BCDA 15 22 16]

  /* 第二轮，[abcd k s i]定义操作
    a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s).
    执行16次操作
  */
   [ABCD  1  5 17] [DABC  6  9 18] [CDAB 11 14 19] [BCDA  0 20 20]
   [ABCD  5  5 21] [DABC 10  9 22] [CDAB 15 14 23] [BCDA  4 20 24]
   [ABCD  9  5 25] [DABC 14  9 26] [CDAB  3 14 27] [BCDA  8 20 28]
   [ABCD 13  5 29] [DABC  2  9 30] [CDAB  7 14 31] [BCDA 12 20 32]

  /* 第三轮，[abcd k s i]定义操作
    a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s).
    执行16次操作
  */
  [ABCD  5  4 33] [DABC  8 11 34] [CDAB 11 16 35] [BCDA 14 23 36]
  [ABCD  1  4 37] [DABC  4 11 38] [CDAB  7 16 39] [BCDA 10 23 40]
  [ABCD 13  4 41] [DABC  0 11 42] [CDAB  3 16 43] [BCDA  6 23 44]
  [ABCD  9  4 45] [DABC 12 11 46] [CDAB 15 16 47] [BCDA  2 23 48]

  /* 第四轮，[abcd k s i]定义操作
    a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s).
    执行16次操作
  */
  [ABCD  0  6 49] [DABC  7 10 50] [CDAB 14 15 51] [BCDA  5 21 52]
  [ABCD 12  6 53] [DABC  3 10 54] [CDAB 10 15 55] [BCDA  1 21 56]
  [ABCD  8  6 57] [DABC 15 10 58] [CDAB  6 15 59] [BCDA 13 21 60]
  [ABCD  4  6 61] [DABC 11 10 62] [CDAB  2 15 63] [BCDA  9 21 64]

  A = A + AA
  B = B + BB
  C = C + CC
  D = D + DD

end
```

4.输出结果

将第3步得到的A,B,C,D的16进制值用小端规则表示，进行拼接即可得到最终的结果，128位(32个16进制的值)的消息摘要。

算法实现 >

本程序使用Python2.7，用lambda表达式表示FGHI四个非线性函数，因为Python中的整数不会溢出，所以循环左移时需要将整数转为32位整数，在填充位和最后输出注意要把数据表示成小端规则，另外四轮操作中根据规律可以整合成一个函数，填充位时我用了另一种方式，删添加01二进制位不原是一样的，这里以8位(一个字节)为一个单位，在原信息后面添加字符，每个字符都是一个字节。一开始添加一个二进制位1，后面添加二进制位0直到位长模512等于448，所以第一个字符就是chr(0x80)，使用公式可计算出应添加多少个chr(0)。使用该种方法代码可以生成字符串的摘要也能生成二进制文件的摘要。

myMD5.py

```
# -*- coding: utf-8 -*-
import math
import struct
F = lambda x, y, z: (x&y)|((~x)&z)   # return z if x == 0 else y
G = lambda x, y, z: (x&z)|(y&(~z))   # return y if z == 0 else x
H = lambda x, y, z: x^y^z
I = lambda x, y, z: y^(x|(~z))
int32 = lambda x: x & 0xffffffff     # 把整数x转为32位整数
rotating = lambda x, s: int32(x << s) | (int32(x) >> (32 - s))  # x循环左移s位
littleEndian = lambda hexs: hexs.decode('hex')[::-1].encode('hex') # 转化为小端
roundOp = lambda a,b,c,d,k,s,i,fun: b + rotating(a+fun(b,c,d)+X[k]+T[i], s)

# 对信息进行位扩展，每次按一个字节8位扩展添加对应的字符
def extend(msg):
  bitslen = len(msg) << 3
  hexlen = littleEndian(hex(bitslen)[2:].zfill(16))
  extlen = [chr(int(hexlen[i:i+2], 16)) for i in range(0, 16, 2)]
  mod = bitslen % 512
  padding = chr(0x80) + chr(0) * ((mod >= 448) * 64 + 55 - (mod >> 3))
  return msg + padding + ''.join(extlen)

# 将4轮操作统一成一个函数，通过改变fun调用对应的操作
def Round(kstep, k, ss, i, fun):
  global A, B, C, D
  for j in range(4):
    A = roundOp(A,B,C,D, k, ss[0], i, fun)
    D = roundOp(D,A,B,C, (k + kstep) % 16, ss[1], i + 1, fun)
    C = roundOp(C,D,A,B, (k + kstep * 2) % 16, ss[2], i + 2, fun)
    B = roundOp(B,C,D,A, (k + kstep * 3) % 16, ss[3], i + 3, fun)
    k, i = (k + kstep * 4) % 16, i + 4

# md5主函数，适用于字符串适用于二进制文件
def md5(s, isFile = False):
  global X, T, A, B, C, D
  msg = extend(open(s, 'rb').read() if isFile else s)
  # 小端规则 在内存中为01 23 45 67这种形式
  A, B, C, D = 0x67452301, 0xefcdab89, 0x98badcfe, 0x10325476
  T = tuple(int(4294967296*abs(math.sin(x))) for x in range(65))
  for b in range(0, len(msg), 64):  # 每64个字节分为一组
    # 每4个字节转化为一个32位整数存进X
    X = tuple(struct.unpack('i', msg[b+j:b+j+4])[0] for j in range(0, 64, 4))
    AA, BB, CC, DD, i = A, B, C, D, 1
    Round(1, 0, (7, 12, 17, 22), i, F)       # Round 1
    Round(5, 1, (5, 9, 14, 20), i + 16, G)   # Round 2
    Round(3, 5, (4, 11, 16, 23), i + 32, H)  # Round 3
    Round(7, 0, (6, 10, 15, 21), i + 48, I)  # Round 4
    A, B, C, D = int32(A+AA), int32(B+BB), int32(C+CC), int32(D+DD)

  return ''.join(littleEndian(hex(i)[2:-1].zfill(8)) for i in (A,B,C,D))
```

Python自带的hashlib库中有MD5函数，可将程序输出与库的输出对比，测试自己写的程序是否正确。运行后myMD5.py会被编译成myMD5.pyc，刚好拿来测试二进制文件的摘要。

testMD5.py

```
# -*- coding: utf-8 -*-
import myMD5
import hashlib
import platform

if __name__ == '__main__':
  if platform.python_version()[0] == '3':
    print(u'请使用Python2.7 运行程序')
    input()
    quit()
  assert myMD5.md5('') == hashlib.md5('').hexdigest()
  assert myMD5.md5('a') == hashlib.md5('a').hexdigest()
  assert myMD5.md5('test') == hashlib.md5('test').hexdigest()
  assert myMD5.md5('Web security') == hashlib.md5('Web security').hexdigest()
  assert myMD5.md5('longstr'*1000) == hashlib.md5('longstr'*1000).hexdigest()
  with open('myMD5.pyc', 'rb') as f:
    MD5 = hashlib.md5()
    MD5.update(f.read())
    assert myMD5.md5('myMD5.pyc', isFile = True) == MD5.hexdigest()
  print('Pass MD5 test!!!')
```