CRC 原理

CRC 是什么东西呢？你用过 RAR 和 ZIP 等压缩软件吗？它们是不是常常会给你一个恼人的“CRC 校验错误”信息呢？我想你应该明白了吧，CRC 就是块数据的计算值，它的全称是“Cyclic Redundancy Check”，中文名是“循环冗余码”，“CRC 校验”就是“循环冗余校验”。

CRC 有什么用呢？它的应用范围很广泛，最常见的就是在网络传输中进行信息的校对。其实我们大可以把它应用到软件保护中去，因为它的计算是非常非常非常严格的。严格到什么程度呢？你的程序只要被改动了一个字节（甚至只是大小写的改动），它的值就会跟原来的不同。所以只要给你的“原”程序计算好 CRC 值，储存在某个地方，然后在程序中随机地再对文件进行 CRC 校验，接着跟第一次生成并保存好的 CRC 值进行比较，如果相等的话就说明你的程序没有被修改/破解过，如果不等的话，那么很可能你的程序遭到了病毒的感染，或者被 Cracker 用 16 进制工具暴力破解过了。

我们先来看看 CRC 的原理。首先看两个式子：
式一：9 / 3 = 3 （余数 = 0）
式二：(9 + 2) / 3 = 3 （余数 = 2）
在小学里我们就知道，除法运算就是将被减数重复地减去除数 X 次，然后留下余数。所以上面的两个式子可以用二进制计算为：（不会二进制计算的可以撤退了）
式一：1001 - 0011 =0110 对应的十进制等式为：9-3=6；
0110 - 0011 =0011 对应的十进制等式为：6-3=3；
0011 - 0011 =0000 对应的十进制等式为：3-3=0；
一共减了 3 次，所以商是 3，而最后一次减出来的结果是 0，所以余数为 0。
式二：1011 - 0011 =1000 对应的十进制等式为：11-3=8；
1000 - 0011 =0101 对应的十进制等式为：8-3=5；
0101 - 0011 =0010 对应的十进制等式为：5-3=2；
一共减了 3 次，所以商是 3，而最后一次减出来的结果是 2，所以余数为 2。

二进制减法运算的规则是，如果遇到 0-1 的情况，那么要从高位借 1，就变成了(10+0)-1=1，这里的 10 是二进制数，相当于十进制的 2。CRC 运算有什么不同呢？让我们看下面的例子：

这次用式子 30 / 9，不过请注意最后的余数：
11110 --> 30
1001 - --> 9

1100 --> 12 （很奇怪吧？为什么不是 21 呢？）
1001 - --> 9
----
101 --> 5，余数 --> the CRC!

这个式子的计算过程是不是很奇怪呢？它不是直接减的，而是用 XOR 的方式来运算（程序员应该都很熟悉 XOR 吧），最后得到一个余数。对啦，这个就是 CRC 的运算方法，明白了吗？CRC 的本质是进行 XOR 运算，运算的过程我们不用管它，因为运算过程对最后的结果没

有意义；我们真正感兴趣的只是最终得到的余数，这个余数就是 CRC 值。

进行一个 CRC 运算我们需要选择一个除数，这个除数我们叫它为“poly”，宽度 W 就是最高位的位置，所以我刚才举的例子中的除数 9，这个 poly 1001 的 W 是 3，而不是 4，注意最高位总是 1。（别问为什么，这个是规定）

如果我们想计算一个位串的 CRC 码，我们想确定每一个位都被处理过，因此，我们要在目标“位串”后面加上 W 个 0 位。现在让我们根据 CRC 的规范来改写一下上面的例子：

```
Poly = 1001，宽度 W = 3
位串 Bitstring = 11110
Bitstring + W zeroes = 11110 + 000 = 11110000

11110000
1001|||| -
-------------
1100|||
1001||| -
------------
1010||
1001|| -
-----------
0110|
0000| -
----------
1100
1001 -
---------
101 --> 5，余数 --> the CRC!
```

还有两点重要声明如下：

1、 只有当 Bitstring 的最高位为 1，我们才将它与 poly 进行 XOR 运算，否则我们只是将 Bitstring 左移一位。
2、 XOR 运算的结果就是被操作位串 Bitstring 与 poly 的低 W 位进行 XOR 运算，因为最高位总为 0。

原理介绍到这里，下面讲讲具体怎么编程。

由于速度的关系，CRC 的实现主要是通过查表法，对于 CRC-16 和 CRC-32，各自有一个现成的表，大家可以直接引入到程序中使用。（由于这两个表太长，在这里不列出来了，请读者自行在网络上查找，很容易找到的）。如果我们没有这个表怎么办呢？或者你跟我一样，懒得自己输入？不用急，我们可以“自己动手，丰衣足食”。你可能会说，自己编程来生成这个表，会不会太慢了？其实大可不必担心，因为我们是在汇编代码的级别进行运算的，而这个表只有区区 256 个双字，根本影响不了速度。

这个表的 C 语言描述如下：

```
for (i = 0; i < 256; i++)
{
crc = i;
for (j = 0; j < 8; j++)
{
if (crc & 1)
crc = (crc >> 1) ^ 0xEDB88320;
else
crc >>= 1;
}
crc32tbl[i] = crc;
}
```

生成表之后，就可以进行运算了。我们的算法如下：

1、 将寄存器向右边移动一个字节。
2、 将刚移出的那个字节与我们的字符串中的新字节进行 XOR 运算，得出一个指向值表 table[0..255]的索引。
3、 将索引所指的表值与寄存器做 XOR 运算。
4、 如果数据没有全部处理完，则跳到步骤 1。

这个算法的 C 语言描述如下：

```
temp = (oldcrc ^ abyte) & 0x000000FF;
crc = (( oldcrc >> 8) & 0x00FFFFFF) ^ crc32tbl[temp];
return crc;
```