转载请注明，原文地址：

http://www.lgygg.wang/lgyblog/2019/10/15/%e6%a0%a1%e9%aa%8c%e5%92%8c/

1.什么是校验和

检验和(checksum)，在数据处理和数据通信领域中，用于校验目的地一组数据项的和。它通常是以十六进制为数制表示的形式。如果校验和的数值超过十六进制的FF，也就是255. 就要求其补码作为校验和。通常用来在通信中，尤其是远距离通信中保证数据的完整性和准确性。
在源端，计算校验和并将其设置在标头中作为字段。在目标端，再次计算校验和，并与标头中的现有校验和值进行交叉校验，以查看数据包是否正常。

2.校检和计算步骤

我们知道，校检和是用来查看数据包是否完整和准确的。所有会先在发送端（即源端，发送数据的一方）计算出校检和，然后把这个校检和也放到发送的数据里，传递给接收端（即目标端，接收数据的一方）。接收方收到数据后，也会再计算一次校检和，然后和发送端发过来的校检和进行对比。如果一致，则证明接收端收到的数据是完整准确的。
下面介绍一下校验的步骤：
发送方生成检验和
1）将发送的进行检验和运算的数据分成若干个16位的位串，每个位串看成一个二进制数，这里并不管字符串代表什么，是整数、浮点数还是位图都无所谓。
2）将IP、UDP或TCP的PDU首部中的检验和字段置为0，该字段也参与检验和运算。
3）对这些16位的二进制数进行1的补码和(one’s complement sum)运算，累加的结果再取反码即生成了检验码。将检验码放入检验和字段 中。
这里需要注意的是，如果发生了进位溢出，校检和就等于数据累加结果的反码。如果没有发生溢出，那么校检和就等于数据累加的结果。
其中1的补码和运算，即带循环进位(end round carry)的加法，最高位有进位应循环进到最低位。反码即二进制各位取反，如0111的反码为 1000。
接收方校验检验和
1）接收方将接收的数据(包括检验和字段)按发送方的同样的方法进行1的补码和运算，累加的结果再取反码（必须取反码）。
2）校验，如果上步的结果为0，表示传输正确；否则，说明传输有差错。

3.例子解析

下面会通过三个例子来介绍校验和的计算过程。

1）计算没有发生溢出字串的校检和

题目：计算下面十六进制串的校检和
十六进制串: 0102030405060708
这是百度百科的题目，它给的答案是24。那么答案是怎么来的？
下面是它给出的计算过程，我们来分析一下计算步骤。

```
        #include<stdio.h>        int main()        {
    int a[8]={0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08};
       int i,sum=0;            for (i=0;i<8;i++)
sum+=a[i];//??????        if(sum>0xff)        {       sum=~s
um;                 sum+=1;                }           sum=
sum&0xff;             printf("0x%x\n",sum);        }
```

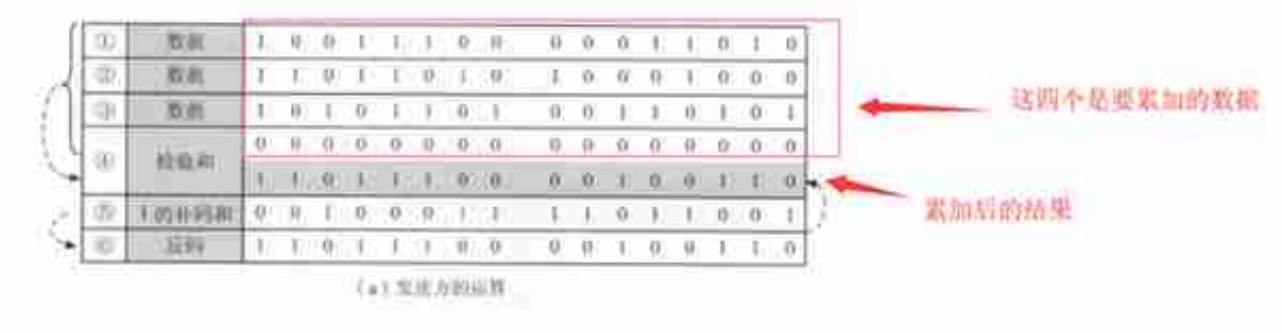
首先，他将十六进制串分为8个部分，分别是{01，02，03，04，05，06，07，08}，然后将它们全部转换为16进制数，得到a[8]={0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08};
然后将这8个数字进行累加，得到36（这是一个十进制数），其实这里我们也可以不将它转换为16进制数再进行累加，完全可以将它们进行十进制计算后，再将结果转为16进制，判断结果是否超过十六进制的FF（也就是十进制的255）。

最后，如果累加的结果超过十六进制的FF（也就是十进制的255），就取它的补码作为校检和。否则就直接取累加的记过作为校检和。我们看到上面累加的结果是36（这是一个十进制数），并没有超过255，所有，校检和结果就是36？这是不对的，因为在计算机数据传输中，校检和是一个16进制数，所有我们要将十进制的36转为16进制，结果为24,所有最终结果为0x24。

2）计算发生溢出的校检和

在计算机中，大多数情况下，校检和都是会发生溢出的。下面这个例子也是百度百科里的。下图中，
(a)所示为发送方的运算，①、②、③是3个数据，④是检验和，先置0，也参加检验和运算。⑤是它们的一的补码和，⑥是⑤的反码。发送方将⑥放到检验和字段和数

据一起发出。
(b)所示为接收方的运算，如果没有传输差错，最后结果应为0。



然后对①、②相加的结果是：
1 01110110 10100010
发生了溢出，所以要在尾部加1，得到结果为
01110110 10100011
再将结果与数据③相加，得到结果
1 00100011 11011000
因为溢出了，所以尾部加1，得到
00100011 11011001
再加上00000000 00000000
最后得到的结果为：
00100011 11011001
上面的计算过程中，发生了溢出，所以最终的校验和为00100011 11011001的反码，即
11011100 00100110。
而接收方都也差不多，也是对①、②、③进行累加，得到
00100011 11011001
再将结果和校检和11011100 00100110相加，结果为
11111111 11111111
这里虽然没有溢出，但是接收方不管有没有溢出最后累加结果都必须取反码，
00000000 00000000
结果为0代表传输数据完整。

3）一个完整的IP头校验和计算

在IP，UDP和TCP等数据包中，都有进行数据和校验的过程。下面这个例子是参考文章里的例子。下图是IP数据包的报头结构