



转 CRC原理详解(附crc16校验代码)

2017年12月02日 11:26:57 扑火飞蛾 阅读数：33925

参考链接：

<https://www.cnblogs.com/esestt/archive/2007/08/09/848856.html>

Cyclic Redundancy Check循环冗余检验，是基于数据计算一组效验码，用于核对数据传输过程中是否被更改或传输错误。

算法原理

假设数据传输过程中需要发送15位的二进制信息 $g=101001110100001$ ，这串二进制码可表示为代数多项式 $g(x) = x^{14} + x^{12} + x^9 + x^8 + x^7 + x^5 + 1$ ，其中 g 中第 k 位的值，对应 $g(x)$ 中 x^k 的系数。将 g 乘以 x^m ，既将 g 后加 m 个0，然后除以 m 阶多项式 $h(x)$ ，得到的 $(m-1)$ 阶余项 $r(x)$ 对应的二进制码 r 就是CRC编码。

$h(x)$ 可以自由选择或者使用国际通行标准，一般按照 $h(x)$ 的阶数 m ，将CRC算法称为CRC- m ，比如CRC-32、CRC-64等。国际通行标准可以参看http://en.wikipedia.org/wiki/Cyclic_redundancy_check

$g(x)$ 和 $h(x)$ 的除运算，可以通过 g 和 h 做xor（异或）运算。比如将11001与10101做xor运算：

1	1	0	0	1
1	0	1	0	1
0	1	1	0	0

明白了xor运算法则后，举一个例子使用CRC-8算法求101001110100001的效验码。CRC-8标准的 $h(x) = x^8 + x^7 + x^6 + x^4 + x^2 + 1$ ，既 h 是9位的二进制串111010101。



6



3



关闭



g ₀	101001110100001000000000
h	111010101
g ₁	010011011100001000000000
h	111010101
g ₂	011100010000010000000000
h	111010101
g ₃	000010001000100000000000
h	111010101
g ₄	011000100000000000000000
h	111010101
g ₅	001011101000000000000000
h	111010101
g ₆	010100001000000000000000
h	111010101
g ₇	010010111000000000000000
h	111010101
g ₈	011111011000000000000000
h	111010101
r	00010001100

经过迭代运算后，最终得到的r是10001100，这就是CRC效验码。

通过示例，可以发现一些规律，依据这些规律调整算法：

1. 每次迭代，根据g_k的首位决定b，b是与g_k进行运算的二进制码。若g_k的首位是1，则b=h；若g_k的首位是0，则b=0，或者跳过此次迭代，上面的例子中就是碰到0后直接跳到后面的非零位。



广告



关闭

g_k 首位是 1	
g_k	11111011000
$b = h$	111010101
r	00010001100

g_k 首位是 0	
g_k	01111011000
$b = 0$	0
r	01111011000

2. 每次迭代, g_k 的首位将会被移出, 所以只需考虑第2位后计算即可。这样就可以舍弃 h 的首位, 将 b 取 h 的后 m 位。比如 CRC-8 的 h 是 111010101, b 只需是 11010101。

g_k	11111011000
b	11010101
r	0010001100

3. 每次迭代, 受到影响的是 g_k 的前 m 位, 所以构建一个 m 位的寄存器 S , 此寄存器储存 g_k 的前 m 位。每次迭代计算前先将 S 的首位抛弃, 将寄存器左移一位, 同时将 g 的后一位加入寄存器。若使用此种方法, 计算步骤如下:

广告



关闭

S	10100111	g:	101001110100001000000000
S'	01001110		
b	11010101	S 首位:	1
S	10011011	g:	101001110100001000000000
S'	00110111		
b	11010101	S 首位:	1
S	11100010	g:	101001110100001000000000
S'	11000100		
b	11010101	S 首位:	1
S	00010001	g:	101001110100001000000000
S'	00100010		
b	0	S 首位:	0
S	00100010	g:	101001110100001000000000
S'	01000100		
b	0	S 首位:	0
.....		

※蓝色表示寄存器S的首位，是需要移出的，b根据S的首位选择0或者h。黄色是需要移入寄存器的位。S'是经过位移后的S。

查表法

同样是上面的那个例子，将数据按每4位组成1个block，这样g就被分成6个block。

g	B1	B2	B3	B4	B5	B6
	0101	0011	1010	0001	0000	0000

下面的表展示了4次迭代计算步骤，灰色背景的位是保存在寄存器中的。

广告



关闭

	B1	B2	B3
S	0101	0011	1010
S'	0101	0011	1010
b	000	0000	0000
S	0101	0011	1010
S'	0101	0011	1010
b	11	0101	0100
S	0110	0110	1110
S'	0110	0110	1110
b	1	1010	1010
S	0111	1100	0100
S'	0111	1100	0100
b		1101	0101

经4次迭代，B1被移出寄存器。被移出的部分，不我们关心的，我们关心的是这4次迭代对B2和B3产生了什么影响。注意表中红色的部分，先作如下定义：

```

B23 = 00111010
b1 = 00000000
b2 = 01010100
b3 = 10101010
b4 = 11010101
b' = b1 xor b2 xor b3 xor b4

```

4次迭代对B2和B3来说,实际上就是让它们与b1,b2,b3,b4做了xor计算，既：

```
B23 xor b1 xor b2 xor b3 xor b4
```

可以证明xor运算满足交换律和结合律，于是：

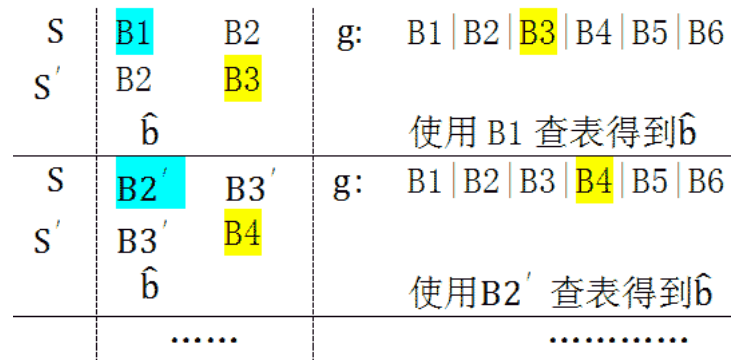
```
B23 xor b1 xor b2 xor b3 xor b4 = B23 xor (b1 xor b2 xor b3 xor b4) = B23 xor b'
```

b1是由B1的第1位决定的，b2是由B1迭代1次后的第2位决定（既是由B1的第1和第2位决定），同理,b3和b4都是由B1决定。通过B1就可以计算出b'。另外，B1由4位组成，其一共 2^4 有种可能值。于是我们就可以想到一种更快捷的算法，事先将b'所有可能的值，16个值可以看成一张表；这样就可以不必进行那4次迭代，而是用B1查表得到b'值，将B1移出，B3移入，与b'计算，然后是下一次迭代。

广告



关闭



可看到每次迭代，寄存器中的数据以4位为单位移入和移出，关键是通过寄存器前4位查表获得，这样的算法可以大大提高运算速度。

上面的方法是半字节查表法，另外还有单字节和双字节查表法，原理都是一样的——事先计算出 2^8 或 2^{16} 个b'的可能值，迭代中使用寄存器前8位或16位查表获得b'。

广告

反向算法

之前讨论的算法可以称为正向CRC算法，意思是将g左边的位看作是高位，右边的位看作低位。G的右边加m个0，然后迭代计算是从高位开始，逐步将低位加入到寄存器中。在实际的数据传送过程中，是一边接收数据，一边计算CRC码，正向算法将新接收的数据看作低位。

逆向算法顾名思义就是将左边的数据看作低位，右边的数据看作高位。这样的话需要在g的左边加m个0，h也要逆向，例如正向CRC-16算法h=0x4c11db8，逆向CRC-16算法h=0xedb88320。b的选择0还是h，由寄存器中右边第1位决定，而不是左边第1位。寄存器仍旧是向左位移，就是说迭代变成从低位到高位。

S	00000000	g:	00000000101001110100001
S'	00000001		
b	0	S 末位:	0
S	00000001	g:	00000000101001110100001
S'	00000010		
b	10101011	S 末位:	1
S	10101001	g:	00000000101001110100001
S'	01010011		
b	11010101	S 末位:	1
.....		

附录1：crc16校验表及用法

```
static unsigned short crc16_table[256] = {
0x0000, 0x1189, 0x2312, 0x329b, 0x4624, 0x57ad, 0x6536, 0x74bf,
0x8c48, 0x9dc1, 0xaf5a, 0xbed3, 0xca6c, 0xdbe5, 0xe97e, 0xf8f7,
0x1081, 0x0108, 0x3393, 0x221a, 0x56a5, 0x472c, 0x75b7, 0x643e,
0x9cc9, 0x8d40, 0xbfdb, 0xae52, 0xdaed, 0xcb64, 0xf9ff, 0xe876,
0x2102, 0x308b, 0x0210, 0x1399, 0x6726, 0x76af, 0x4434, 0x55bd,
0xad4a, 0xbcc3, 0x8e58, 0x9fd1, 0xeb6e, 0xf9e7, 0xc87c, 0xd9f5,
0x3183, 0x200a, 0x1291, 0x0318, 0x77a7, 0x662e, 0x54b5, 0x453c,
0xbdcb, 0xac42, 0x9ed9, 0x8f50, 0xfbef, 0xea66, 0xd8fd, 0xc974,
0x4204, 0x538d, 0x6116, 0x709f, 0x0420, 0x15a9, 0x2732, 0x36bb,
0xce4c, 0xdfc5, 0xed5e, 0xfcd7, 0x8868, 0x99e1, 0xab7a, 0xbaf3,
0x5285, 0x430c, 0x7197, 0x601e, 0x14a1, 0x0528, 0x37b3, 0x263a,
0xdec d, 0xcfd4, 0xfdd5, 0xec56, 0x98e9, 0x8960, 0xbbfb, 0xaa72,
0x6306, 0x728f, 0x4014, 0x519d, 0x2522, 0x34ab, 0x0630, 0x17b9,
0xef4e, 0xfec7, 0xcc5c, 0xdd5, 0xa96a, 0xb8e3, 0x8a78, 0x9bf1,
0x7387, 0x620e, 0x5095, 0x411c, 0x35a3, 0x242a, 0x16b1, 0x0738,
```


[关闭](#)

```
0xffcf, 0xee46, 0xdcdd, 0xcd54, 0xb9eb, 0xa862, 0x9af9, 0x8b70,
0x8408, 0x9581, 0xa71a, 0xb693, 0xc22c, 0xd3a5, 0xe13e, 0xf0b7,
0x0840, 0x19c9, 0x2b52, 0x3adb, 0x4e64, 0x5fed, 0x6d76, 0x7cff,
0x9489, 0x8500, 0xb79b, 0xa612, 0xd2ad, 0xc324, 0xf1bf, 0xe036,
0x18c1, 0x0948, 0x3bd3, 0x2a5a, 0x5ee5, 0x4f6c, 0x7df7, 0x6c7e,
0xa50a, 0xb483, 0x8618, 0x9791, 0xe32e, 0xf2a7, 0xc03c, 0xd1b5,
0x2942, 0x38cb, 0x0a50, 0x1bd9, 0x6f66, 0x7eef, 0x4c74, 0x5dfd,
0xb58b, 0xa402, 0x9699, 0x8710, 0xf3af, 0xe226, 0xd0bd, 0xc134,
0x39c3, 0x284a, 0x1ad1, 0x0b58, 0x7fe7, 0x6e6e, 0x5cf5, 0x4d7c,
0xc60c, 0xd785, 0xe51e, 0xf497, 0x8028, 0x91a1, 0xa33a, 0xb2b3,
0x4a44, 0x5bcd, 0x6956, 0x78df, 0x0c60, 0x1de9, 0x2f72, 0x3efb,
0xd68d, 0xc704, 0xf59f, 0xe416, 0x90a9, 0x8120, 0xb3bb, 0xa232,
0x5ac5, 0x4b4c, 0x79d7, 0x685e, 0x1ce1, 0x0d68, 0x3ff3, 0x2e7a,
0xe70e, 0xf687, 0xc41c, 0xd595, 0xa12a, 0xb0a3, 0x8238, 0x93b1,
0x6b46, 0x7acf, 0x4854, 0x59dd, 0x2d62, 0x3ceb, 0x0e70, 0x1ff9,
0xf78f, 0xe606, 0xd49d, 0xc514, 0xb1ab, 0xa022, 0x92b9, 0x8330,
0x7bc7, 0x6a4e, 0x58d5, 0x495c, 0x3de3, 0x2c6a, 0x1ef1, 0x0f78
};

uint32 crc_check(uint8* data, uint32 length)
{
    unsigned short crc_reg = 0xFFFF;
    while (length--)
    {
        crc_reg = (crc_reg >> 8) ^ crc16_table[(crc_reg ^ *data++) & 0xff];
    }
    return (uint32)(~crc_reg) & 0x0000FFFF;
}
```

data为待校验数组，返回值为crc校验累加结果。



关闭



50万码农评论：英语对于程序员有多重要！

不背单词和语法，老司机教你一个数学公式秒懂天下英语



想对作者说点什么

CRC16常见几个标准的算法及C语言实现

阅读数 442

CRC码由发送端计算，放置于发送信息报文的尾部。接收信息的设备再重新计算接收到信息报文的CRC，比较计算得...

博文 来自： [明朝的BLOG](#)

Modbus通信CRC16校验程序

阅读数 1662

ModBus通信协议的CRC(冗余循环校验码)含2个字节,即16位二进制数。CRC码由发送设备计算,放置于所发送信息帧...

博文 来自： [睿思派克](#)

CRC16校验原理及实现

阅读数 1万

CRC码由发送端计算，放置于发送信息报文的尾部。接收信息的设备再重新计算接收到信息报文的CRC，比较计算得...

博文 来自： [lzy201s的博客](#)

陈小春哭诉：西三旗土豪怒砸2亿请他代言这款0充值正版传奇！真经典！

贪玩游戏 · 顶新

广告

CRC-CCITT CRC-16

阅读数 1583

constuint16_tTable[256]={0x0000U,0x1021U,0x2042U,0x3063U,0x4084U,0x50A5U,0x60C6U,0x70E7U,0x8108U,...

博文 来自： [超哥的专栏](#)

CRC-16/MODBUS 校验位计算

阅读数 2872

实验指令：前23位表示信息头+信息内容。24，25位是待计算的校验位。26位是结束码7E0100000110020027000C...

博文 来自： [wanyongtai的博客](#)

CRC-16 (Modbus)校验码

阅读数 2416

CRC-16(Modbus)校验码

博文 来自： [芒果菠萝π](#)

CRC的基本原理详解

阅读数 8864

CRC(CyclicRedundancyCheck)被广泛用于数据通信过程中的差错检测，具有很强的检错能力。本文详细介绍了CRC...

博文 来自： [TerryZjl的博客](#)

CRC8和CRC16的计算方法

阅读数 5059

CRC8转载地址：http://blog.csdn.net/d_leo/article/details/73572373什么是CRC校验？CRC即循环冗余校验码：...

博文 来自： [Chuck_lin的博客](#)



50万码农评论：英语对于程序员有多重要！

不背单词和语法，老司机教你一个数学公式秒懂天下英语

CRC-16校验(多项式为x16+x15+x2+1):

阅读数 5659

CRC-16校验(多项式为x16+x15+x2+1):type{CRC校验}TDataByte=arrayofbyte;const CRCHi:array[0..255]ofbyt...

博文 来自： [lchu55的博客](#)




关闭

CRC-16校验 完整代码

阅读数 2877

通信领域经常用到CRC校验。这里把CRC-16的代码转发一下。不过，我推荐一个用于生成各种校验码的开源软件Fsu...


博文 来自： [打印/传真嵌入式开](#)



小索

98篇文章


关注 排名:千里之外



tx_yu

52篇文章


关注 排名:千里之外



ChloeDimen

146篇文章

关注 排名:千里之外



半峰残月

5篇文章

关注 排名:千里之外

CRC校验

阅读数 780

CRC校验概念CRC校验原理CRC校验步骤CRC校验举例

博文 来自： [z_jingjing的博客](#)

CRC校验实现

阅读数 154

1、实验题目：CRC校验 PPP协议受到数据帧后要和数据部分连同FCS字段做CRC校验，结果若不为“0”，则可...

博文 来自： [fly夏天的博客](#)

Checksum算法

阅读数 2.5万

checksum算法，IPchecksum算法，tcpchecksum算法，udpchecksum算法。

博文 来自： [简单的过客](#)

饥荒游戏竟然出网页版了？已经玩疯了

中至科技公司 · 顶新

串口通讯之 CRC校验

阅读数 250

一、CRC16简介 循环冗余码CRC检验技术广泛应用于测控及通信领域。CRC计算可以靠专用的硬件来实现，但是对...

博文 来自： [a932432866的专栏](#)

CRC16校验码运算

阅读数 53

CRC寄存器一开始填充为16位1111111111111111然后将CRC寄存器的低8位11111111与报文数据第一个八位数据进...

博文 来自： [qq_18671205的博](#)

CRC16代码 (C语言实现)

阅读数 436

ModbusCRC16校验代码嵌入式系统crc16校验码计算函数记录include “crc.h” /***** ...

博文 来自： [lubety的博客](#)

关于modbus rtu协议的CRC (循环冗余校验) 在线计算

阅读数 2.8万

上位机须按照MODBUS协议的命令格式发送数据（包括计算的CRC值），从机才能正确辨识数据。若无CRC值，从机...

博文 来自： [大漠飞鹰lb的博客](#)

crc 校验码的计算方法（转载）

阅读数 3396

摘要：CRC(CyclicRedundancyCheck)被广泛用于数据通信过程中的差错检测，具有很强的检错能力。本文详细介绍...

博文 来自： [石头的专栏](#)

陈小春哭诉：西三旗土豪怒砸2亿请他代言这款0充值正版传奇！真经典！

贪玩游戏 · 顶新

【算法集中营】CRC16 三种算法及c实现

阅读数 2073

标准CRC生成多项式如下表： 名称 生成多项式 简记式* 标准引用 CRC-4 x4+x+1 3 IT...



博文 来自： [ZhangPY的专栏](#)


广告



关闭

https://blog.csdn.net/xing414736597/article/details/78693781

10/14

循环冗余校验（CRC）计算的C++ 代码		阅读数 3732
最近经常有人发私信给我找我帮忙计算CRC校验。其实我有一篇博客 http://blog.csdn.net/liyuanbhu/article/detail...		博文 来自： Ivan 的专栏
CRC算法及原理		阅读数 3300
CRC算法及原理 本文转自： http://www.cnblogs.com/FPGA_DSP/archive/2010/05/08/1730529.html CRC校验码...		博文 来自： qinghecool的专栏
Java中的ModBus CRC16校验		阅读数 1644
工程项目中需用MBCRC16校验，网上查到很多不能用，下面几个符合ModBusCRC16的算法，以备后查。...		博文 来自： Iven.Yang
CRC原理简述		阅读数 1462
以下内容摘自笔者即将出版的最新著作《深入理解计算机网络》一书。本书将于12月底出版上市，敬请留意！！ 本...		博文 来自： huohongpeng的
<div><div>50万码农评论：英语对于程序员有多重要！ 不背单词和语法，老司机教你一个数学公式秒懂天下英语</div></div>		
[redis] crc16		阅读数 138
一、背景redis集群使用哈希槽实现，其对key进行哈希计算采用crc16(key) &0x3fff,得到在16384个槽的其...		博文 来自： gonaYet的博客
三种CRC16 C语言算法理解（CCITT）		阅读数 496
三种CRC16的算法实现，结合自己的理解记录一下；一、先说算法的C语言实现和各算法的优缺点：1、按位计算CRC...		博文 来自： j824117879的博客
CRC16码与Python 中各种数据类型		阅读数 253
CRC-16/MODBUS校验的在线计算网站： http://www.ip33.com/crc.html 然后附上用Python写的CRC-16/MODBU...		博文 来自： 足迹
【算法集中营】循环冗余校验		阅读数 1167
CRC的全称为CyclicRedundancyCheck，中文名称为循环冗余校验。它是一类重要的线性分组码，编码和解码方法...		博文 来自： ZhangPY的专栏
CRC计算方法与实例		阅读数 1.2万
来自： http://www.baiheee.com/Documents/090107/090107122214.htm CRC冗余循环校验，传输的帧或者序列...		博文 来自： 自由理想的足迹
<div><div>50万码农评论：英语对于程序员有多重要！ 不背单词和语法，老司机教你一个数学公式秒懂天下英语</div></div>		
CCITT CRC-16计算原理与实现【整理综合】		阅读数 6969
CRC的全称为CyclicRedundancyCheck，中文名称为循环冗余校验。它是一类重要的线性分组码，编码和解码方法...		博文 来自： Carl_Ding的专栏
CCITT CRC-16计算原理与实现		阅读数 810
http://blog.chinaunix.net/uid-20416869-id-173134.html CRC的全称为CyclicRedundancyCheck，中文名称为循...		博文 来自： threshold1980的

CRC校验的快速算法的C语言实现		阅读数 3467
CRC校验的快速算法的C语言实现 摘要：CRC循环冗余校验算法，是一种在数据存储和数据通讯领域中使用十分广泛...		
计算法和查表法实现的CRC16校验码生成		03-19
实现CRC16校验码的生成。使用计算法和查表法。 运行环境是VS2008，双击文件夹CRC_table中的CRC_table.sln打开项目。直接运行即可。 ...		下载
python写的具有CRC16生成、校验功能的简单类		阅读数 7841
利用的是查表法，多项式是X16+		博文 来自： fenglifeng1987的
<div></div> <div>程序猿不会英语怎么行？英语文档都看不懂！ 不背单词和语法，一个公式教你读懂天下英文→</div>		
小学霸修炼秘籍之FPGA篇--CRC16校验码的计算（ Verilog ）		阅读数 942
小学霸这次做的项目需要Modbus-TCP协议转化为Modbus-RTU协议并通过串口输出，RTU协议中在数据发送的最...		博文 来自： weixin_41576149
串口通信中CRC16校验类的实现		阅读数 5041
一、CRC16简介 循环冗余码CRC检验技术广泛应用于测控及通信领域。CRC计算可以靠专用的硬件来实现，但是对...		博文 来自： tx_yu的专栏
CRC16（查表法）在c语言和Java的验证		阅读数 181
c语言：const unsigned short crc16_table[256]={0x0000,0x1021,0x2042,0x3063,0x4084,0x50a5,0x60c6,0x70e7,0x...		博文 来自： Dimen-zhou的博客
CRC查找表法推导及代码实现比较		阅读数 2.2万
2018/02/08再次更新——...		博文 来自： huang_shiyang的
最详细易懂的CRC-16校验原理（附源程序）		阅读数 8231
from:http://www.openhw.org/chudonganjin/blog/12-08/230184_515e6.html 最详细易懂的CRC-16校验原理...		博文 来自： Jorry Zhao的专栏
<div></div> <div>英语文档看不懂？教你一个公式秒懂英语！ 软件工程出身的英语老师，教你用数学公式读懂天下英文→</div>		
CRC16校验的c语言实现		阅读数 4857
使用c语言和查表法来实现CRC16（IBM标准）		博文 来自： arenascats的博客
CRC16浅析		阅读数 714
CRC即循环冗余校验码（CyclicRedundancyCheck），是数据通信领域中最常用的一种差错校验码。额，原理/理论...		博文 来自： 上发条的博客
CRC16校验使用体验		阅读数 864
CRC16校验最近开发有用到CRC16校验，但是网上普遍是CRC-16/MODBUS的，项目上使用的是CRC-16/X25，只有...		博文 来自： c331043的专栏

<div>CRC校验源码分析</div> <div>CRC校验源码分析(freebirds)循环冗余校验CRC的算法分析和程序实现(PDF)CRC校验源码分析(PDF)CRC校验源码分...</div>	<div>博文</div> <div>来自： Everest的博客 (杂</div>	<div>阅读数 2726</div>
<div>FPGA-CRC校验</div> <div>一、CRC原理。 CRC校验的原理非常简单，如下图所示。其中，生成多项式是利用抽象代数的一些规则推导出来...</div>	<div>博文</div> <div>来自： 匠心=专一+持之以</div>	<div>阅读数 1204</div>
<div>程序猿不会英语怎么行？英语文档都看不懂！</div> <div>不背单词和语法，一个公式教你读懂天下英文→</div>		
<div>CRC7校验</div> <div>CRC校验原理，参见：https://www.baidu.com/s?wd=crc%E6%A0%A1%E9%AA%8C%E7%AE%97%E6%B3%95...</div>	<div>博文</div> <div>来自： xupu1594908983</div>	<div>阅读数 109</div>
<div>CRC16校验码C语言实现</div> <div>一、目的 阐述CRC16的原理，并以C语言代码实现。二、校验码的作用 校验码用于校验数据的有效性/正确性。校验...</div>	<div>博文</div> <div>来自： btchengzi0的专栏</div>	<div>阅读数 251</div>
<div>CRC校验原理及查表码表由来</div> <div>CRC校验是编程中使用比较多的一种检验方式，包括CRC8，CRC16，CRC32校验等。校验长度越长，校验所需要的...</div>	<div>博文</div> <div>来自： watterwu的博客</div>	<div>阅读数 6977</div>
<div>CRC16校验 ---复制，查表法，用于高速通信校验</div> <div>CRC16校验 ---复制，查表法，用于高速通信校验 (2011-08-2520:45:07)转载▼ 前段时间用单片机做高速通信，在主...</div>	<div>博文</div> <div>来自： suding666的专栏</div>	<div>阅读数 5259</div>
<div>CRC算法原理及其Verilog实现</div> <div>一．CRC简介CRC校验是一种在数据通信系统和其它串行传输系统中广泛使用的错误检测手段。通用的CRC标准有CR...</div>	<div>博文</div> <div>来自： 海鲜小王子的专栏</div>	<div>阅读数 8595</div>
<div>50万码农评论：英语对于程序员有多重要！</div> <div>不背单词和语法，老司机教你一个数学公式秒懂天下英语</div>		
<div>VS2015 MFC 编写 常见的CRC校验，CRC4,CRC5,CRC8,CRC16,CRC32等</div> <div>CRC4-ITU X4+X+1 CRC5-EPC X4+X3+1 CRC5-ITU X5+X4+X2+1 CRC5-USB X5+X2+1 CRC6-ITU X5+X2+1 CRC7-MMC X7+X</div>	<div>博文</div> <div>来自： 北回归线的博客</div>	<div>04-12</div> <div>下载</div>
<div>CRC校验的理解和C语言实现</div> <div>1、CRC是什么CRC检验的基本思想是利用线性编码理论，在发送端根据要传送的k位二进制码序列，以一定的规则产...</div>	<div>博文</div> <div>来自： Ape55的博客</div>	<div>阅读数 4055</div>
<div>plsql的命令 (command) 窗口与sql窗口有什么区别20170620</div> <div>command窗口是命令窗口，即为sqlplus窗口，有命令提示符，识别sqlplus命令，基本的命令都可以执行 sql仅可执...</div>	<div>博文</div> <div>来自： 明明</div>	<div>阅读数 7867</div>
<div>SSH的端口转发:本地转发Local Forward和远程转发Remote Forward</div> <div>http://zhumeng8337797.blog.163.com/blog/static/100768914201172125444948/ 实战 SSH 端口转发 htt...</div>	<div>博文</div> <div>来自： 明明</div>	<div>阅读数 65431</div>

EasyIPCamera实现Windows PC桌面、安卓Android桌面同屏直播，助力无纸化会议系统

阅读数 6782

最近在EasyDarwin开源群里，有不少用户私信需求，要做一种能够多端同屏的系统，细分下来有屏幕采集端和同屏...

博文 来自： Babosa的专栏

linux上安装Docker(非常简单的安装方法)

阅读数 110302

最近比较有空，大四出来实习几个月了，作为实习狗的我，被叫去研究Docker了，汗汗！ Docker的三大核心概念：...

博文 来自： 我走小路的博客

thymeleaf模板实现html5标签的非严格检查

阅读数 7053

一、概述最近在springboot项目引入thymeleaf模板时，使用非严格标签时，运行会报错。默认thymeleaf模板对ht...

博文 来自： Luck_ZZ的博客

强连通分量及缩点tarjan算法解析

阅读数 134110

强连通分量： 简言之 就是找环（每条边只走一次，两两可达） 孤立的一个点也是一个连通分量 使用tarjan算法 在...

博文 来自： 九野的博客

expat介绍文档翻译

阅读数 14412

原文地址：http://www.xml.com/pub/a/1999/09/expat/index.html 因为需要用，所以才翻译了这个文档。但总归...

博文 来自： ymj7150697的专栏

python图片处理类之~PIL.Image模块(ios android icon图标自动生成处理)

阅读数 13872

1.从pyCharm提示下载PIL包 http://www.pythonware.com/products/pil/ 2.解压后，进入到目录下 cd /Users/jia...

博文 来自： 专注于全栈游戏开发

Android 合并生成分享图片（View截图）

阅读数 9819

用以前以前写过的自定义课表软件,Android 自定义View课程表表格 原生View截图合成分享的图片 看到的是图片只...

博文 来自： ShallCheek

开源Json解析器的浅述

阅读数 187

之前在做项目中，Json 这种数据转换格式经常用，为什么呢？我认为是 1、它的易用性，跨平台性，它是JS(JavaScri...

博文 来自： qq_34417408的博

关于SpringBoot bean无法注入的问题（与文件包位置有关）

阅读数 60289

问题场景描述整个项目通过Maven构建，大致结构如下： 核心Spring框架一个module spring-boot-base service...

博文 来自： 开发随笔

android 实时高斯模糊 毛玻璃效果

阅读数 9055

在找遍了网上所有关于实时高斯模糊的效果，并没有直接现成的例子，所以自己东拼西凑，在加上自己的改动，终于...

博文 来自： grp0916的专栏

R语言逻辑回归、ROC曲线和十折交叉验证

阅读数 16479

自己整理编写的逻辑回归模板，作为学习笔记记录分享。数据集用的是14个自变量Xi，一个因变量Y的australian数据...

博文 来自： Tiaaaaaa的博客

xgboost原理详解 论文详解 卷积神经网络详解 numpy详解 算术编码原理

c# crc16校验 c#写的crc16校验 crc16校验c#实现 c#串口crc校验 crc校验算法c++程序 python3教程详解 python初级教程:入门详解

广告



关闭



扑火飞蛾

关注