

人较笨且记性不好,故记录在此.折叠代码打不开请F5.本博中很多是转载收录其他网友的文章(原文地址请见博文末尾),所有权为原作者所有!!!
此博客已不再更新和维护, 欢迎关注我的github和新博客。

博客园 :: 首页 :: 博问 :: 闪存 :: 新随笔 :: 联系 :: 管理 :: 124 随笔 :: 0 文章 :: 36 评论 :: 0 引用

< 2018年9月 >						
日	一	二	三	四	五	六
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

公告

此博客已不再更新和维护, 欢迎关注我的github和新博客。

昵称: 子坞
园龄: 7年9个月
粉丝: 43
关注: 41
[+加关注](#)

搜索

常用链接

[我的随笔](#)
[我的评论](#)
[我的参与](#)
[最新评论](#)
[我的标签](#)

随笔分类(131)

[Algorithm\(2\)](#)
[C#](#)
[C/C++\(33\)](#)

【转】C/C++中的联合体union及CPU大小端判定

在C/C++程序的编写中, 当多个基本数据类型或复合数据结构要占用同一片内存时, 我们要使用联合体; 当多种类型, 多个对象, 多个事物只取其一(我们姑且通俗地称其为“n 选1”), 我们也可以使用联合体来发挥其长处。首先看一段代码:

```
union myun
{
    struct { int x; int y; int z; }u;
    int k;
}a;
int main()
{
    a.u.x =4;
    a.u.y =5;
    a.u.z =6;
    a.k = 0;
    printf("%d %d %d\n",a.u.x,a.u.y,a.u.z);
    return 0;
}
```

union类型是共享内存的, 以size最大的结构作为自己的大小, 这样的话, myun这个结构就包含u这个结构体, 而大小也等于u这个结构体的大小, 在内存中的排列为声明的顺序x,y,z从低到高, 然后赋值的时候, 在内存中, 就是x的位置放置4, y的位置放置5, z的位置放置6, 现在对k赋值, 对k的赋值因为是union, 要共享内存, 所以从union的首地址开始放置, 首地址开始的位置其实是x的位置, 这样原来内存中x的位置就被k所赋的值代替了, 就变为0了, 这个时候要进行打印, 就直接看内存里就行了, x的位置也就是k的位置是0, 而y, z的位置的值没有改变, 所以应该是**0,5,6**。

再看两个试题:

Computer(10)
Cryptography(1)
Datebase(1)
DLL
Driver Studio(5)
Examination
InstallShield
Linux(3)
Math(6)
MFC(10)
Mobile(1)
Network(4)
Office(1)
Protocol
STL
Struct
Thread
TinySoft(1)
T-SQL
UML
Visual Studio(28)
Win SDK(24)
XML(1)

随笔档案(124)

2012年5月 (1)
2012年4月 (4)
2012年3月 (11)
2012年2月 (7)
2012年1月 (10)
2011年12月 (17)
2011年11月 (5)
2011年10月 (5)
2011年9月 (12)
2011年8月 (10)
2011年7月 (14)
2011年6月 (13)
2011年4月 (2)
2011年3月 (6)
2011年1月 (2)
2010年12月 (5)

积分与排名

积分 - 172645
排名 - 1744

试题一：编写一段程序判断系统中的CPU 是Little endian 还是Big endian 模式？

分析：

作为一个计算机相关专业的人，我们应该在计算机组成中都学习过什么叫Little endian 和Big endian。Little endian 和Big endian 是CPU 存放数据的两种不同顺序。对于整型、长整型等数据类型，Big endian 认为第一个字节是最高位字节（按照从低地址到高地址的顺序存放数据的高位字节到低位字节——大端）；而Little endian 则相反，它认为第一个字节是最低位字节（按照从低地址到高地址的顺序存放数据的低位字节到高位字节——小端）。

例如，假设从内存地址0x0000 开始有以下数据：

0x12 0x34 0xab 0xcd

如果我们去读取一个地址为0x0000 的四个字节变量，若字节序为big-endian，则读出结果为0x1234abcd；若字节序为little-endian，则读出结果为xcdab3412。如果我们将0x1234abcd 写入到以0x0000 开始的内存中，则Little endian 和Big endian 模式的存放结果如下：

地址	0x0000	0x0001	0x0002	0x0003
big-endian	0x12	0x34	0xab	0xcd
little-endian	0xcd	0xab	0x34	0x12

一般来说，x86 系列CPU 都是little-endian 的字节序，PowerPC 通常是Big endian，还有的CPU 能通过跳线来设置CPU 工作于Little endian 还是Big endian 模式。

解答：

显然，解答这个问题的方法只能是将一个字节（CHAR/BYTE 类型）的数据和一个整型数据存放于同样的内存开始地址，通过读取整型数据，分析CHAR/BYTE 数据在整型数据的高位还是低位来判断CPU 工作于Little endian 还是Big endian 模式。得出如下的答案：



```
typedef unsigned char BYTE;
int main(int argc, char* argv[])
{
    unsigned int num,*p;
    p = &num;
    num = 0;
    *(BYTE *)p = 0xff;
    if(num == 0xff)
    {
        printf("The endian of cpu is little\n");
    }
    else //num == 0xff000000
    {
        printf("The endian of cpu is big\n");
    }
}
```

最新评论

1. Re:[转]HTTP 头部详细解释
推荐+1

--InkFx

2. Re:WinPcap移植出现"const long * __w64 "转换为"const time_t *"问题
受益匪浅，谢谢

--jiwy

3. Re:<C++沉思录>学习笔记
坚其志，苦其心，劳其力，事无大小，必有所成。

说得好！

--itfanr

阅读排行榜

1. [转]Linux-Ubuntu 启用root账户(40849)
2. SetForegroundWindow的正确用法(33067)
3. MultiByteToWideChar和WideCharToMultiByte的正确使用方法及参数详解(30724)

评论排行榜

1. [转]NPF驱动核心指南(含与NDIS区别)(7)
2. WaitForInputIdle 的注意事项(4)
3. 【转】C/C++中的联合体union及CPU大小端判定(3)

```
return 0;
}
```

除了上述方法(通过指针类型强制转换并对整型数据首字节赋值，判断该赋值赋给了高位还是低位)外，还有没有更好的办法呢？我们知道，union 的成员本身就被存放在相同的内存空间（共享内存，正是union 发挥作用、做贡献的去处），因此，我们可以将一个CHAR/BYTE 数据和一个整型数据同时作为一个union 的成员，得出

如下答案：

```
int checkCPU()
{
    union w
    {
        int a;
        char b;
    }c;
    c.a = 1;
    return (c.b == 1); // 小端返回TRUE,大端返回FALSE
}
```

实现同样的功能，我们来看看Linux 操作系统中相关的源代码是怎么做的：

```
static union { char c[4]; unsigned long mylong; } endian_test = {{ 'l', '?', '?', 'b' } };
#define ENDIANNESS ((char)endian_test.mylong)
```

Linux 的内核作者们仅仅用一个union 变量和一个简单的宏定义就实现了一大段代码同样的功能！由以上一段代码我们可以深刻领会到Linux 源代码的精妙之处！（如果ENDIANNESS='l'表示系统为little endian,为'b'表示big endian）

试题二：假设网络节点A 和网络节点B 中的通信协议涉及四类报文，报文格式为“报文类型字段+报文内容的结构体”，四个报文内容的结构体类型分别为STRUCTTYPE1~ STRUCTTYPE4，请编写程序以最简单的方式组织一个统一的报文数据结构。



织一个统一的报文数据结构。

分析：


报文的格式为“报文类型+报文内容的结构体”，在真实的通信中，每次只能发四类报文中的一种，我们可以将四类报文的结构体组织为一个union（共享一段内存，但每次有效的只是一种），然后和报文类型字段统一组织成一个报文数据结构。

解答:

根据上述分析, 我们很自然地得出如下答案:

```
typedef unsigned char BYTE;  
//报文内容联合体  
typedef union tagPacketContent  
{  
    STRUCTTYPE1 pkt1;  
    STRUCTTYPE2 pkt2;  
    STRUCTTYPE3 pkt1;  
    STRUCTTYPE4 pkt2;  
}PacketContent;  
//统一的报文数据结构  
typedef struct tagPacket  
{  
    BYTE pktType;  
    PacketContent pktContent;  
}Packet;
```



【参考资料 感谢作者】

以上摘自: http://blog.chinaunix.net/u2/84450/showart_1829958.html

另外, 我看到篇文章: http://hi.baidu.com/ilotus_y/blog/item/e8d68c29130875f998250aaa.html 讨论C++联合体的高级应用。

快捷操作:

坚其志, 苦其心, 劳其力, 事无大小, 必有所成。

@如有侵权, 请作者本人尽快与我(chrayo#163.com)联系, 我将及时删除侵权内容。

分类: [C/C++](#)

标签: [联合体](#), [大端](#), [小端](#)

好文要顶

关注我

收藏该文



子坞

关注 - 41

粉丝 - 43

[+加关注](#)[« 上一篇: 【转】枚举类型是什么意思,怎么用?](#)[» 下一篇: WS_POPUP WS_OVERLAPPED WS_CHILD的区别](#)posted on 2010-12-27 11:12 [子坞](#) 阅读(2720) 评论(3) [编辑](#) [收藏](#)

1

0

评论

#1楼 2012-03-14 17:09 菜丝inside

```
static union { char c[4]; unsigned long mylong; } endian_test = { { 'l', '?', '?', 'b' } };
```

```
#define ENDIANNESS ((char)endian_test.mylong)
```

为什么非得是UNIX实现 windows和vxworks下一样可以使用

[支持\(0\)](#) [反对\(0\)](#)[回复](#) [引用](#)

#2楼 2012-03-14 17:10 菜丝inside

不错此篇blog甚好 我已经转载^_^

[支持\(0\)](#) [反对\(0\)](#)[回复](#) [引用](#)

#3楼[楼主] 2012-03-14 20:45 子坞

@ 菜丝inside

你好! 文中已经说了是“Linux 的内核作者们仅仅用一个union 变量和一个简单的宏定义就实现了一大段代码同样的功能!”,并未说linux以外的平台不可以用。

[支持\(0\)](#) [反对\(0\)](#)