<https://www.cnblogs.com/shirishiqi/p/5416935.html>

转自：http://blog.csdn.net/feimor/article/details/6858103

使用C语言时，常常使用struct，对于union类型却几乎没有用过，只知道它是联合类型，各字段共享一块内存，实际应用中却不知道它的具体用途。

今天读《编程卓越之道》中看到有对其使用的简单介绍，记录在此：

首先摘录一下struct和union的重要区别：

    struct为每个字段在不同的偏移处分配存储空间，而联合则将所有的字段重叠在内存中的相同偏移处。

联合类型的用途：

             1、创建别名。别名是内存对象原名之外的其他名字。比如在程序中经常会用到将一个数据类型强制转换为另一个类型，这个操作可以使用联合来代替。

比如typedef union{

char c;

uint32 u;

}CharOfUnion v;

可以通过v.u来操作一个uint32类型的对象，当需要将uint32变量的低端字节看做一个字符的时候，只需要访问v.c就可以了。

实际应用中，大多数的协议结构定义中，都会使用union类型来定义。

2、使用联合来将较大的对象分解成组成这个对象的各个字节。（尤其在单片机编程中将float拆解成char）

typedef union{

uint u;

char bytes[4];

}asBytes composite;

可以使用composite.bytes数组来访问composite.u字段的各个字节。

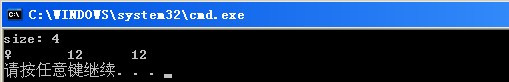
转自：http://blog.163.com/yx\_xie2007/blog/static/1024642532011101411940162/

一直以来，union都是个很少用到的东西，对于这些不常用的结构往往记不住。这次看书又看到了，还是学习一下吧。  
一般在Windows API的一些数据结构中才能看到这个union，其实并不复杂。本质上来说和结构体是一样的，但是从包装的角度来看有差异。

1、union中可以定义多个成员，union的大小由最大的成员的大小决定。   
2、union成员共享同一块大小的内存，一次只能使用其中的一个成员。   
3、对某一个成员赋值，会覆盖其他成员的值（也不奇怪，因为他们共享一块内存。但前提是成员所占字节数相同，当成员所占字节数不同时只会覆盖相应字节上的值，比如对char成员赋值就不会把整个int成员覆盖掉，因为char只占一个字节，而int占四个字节）

4、联合体union的存放顺序是所有成员都从低地址开始存放的。  
下面看一个简单的代码：

 #include <stdio.h>  
   
typedef union{  
 char c;  
 int a;  
 int b;  
}Demo;  
   
int main(int argc, char \*\*argv)  
{  
 Demo d;  
 d.c = 'H';  
 d.a = 10;  
 d.b = 12;  
   
 printf("size: %d\n", sizeof(d));  
 printf("%c\t%d\t%d\n", d.c, d.a, d.b);  
   
 return 0;  
}



原创文章，转载请注明:  
本文出自[程序人生](http://www.programlife.net/) >> [C语言union(联合体 共用体)](http://www.programlife.net/union-struct-in-c.html)