C 中使用union的几点思考 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/261/2021_2022_C___E4_B8_AD_E4_BD_BF_E7_c67_261313.htm 这段时间整理旧资料，看到一些文章，虽然讲的都是些小问题，不大可能用到，但也算是一个知识点，特整理出来与大家共享。与此相关的那篇文章的作者的有些理解是错误的，我写此文，也是纠正为了作者的一些错误认识。当然，如果我的理解有任何错误，也恳请大家批评指正。 C 虽说被B.S.称作一门新语言，但它毕竟与C有着千丝万缕的联系，虽然B.S.一再坚持，但我还是愿意把C 看作是C 。 我们应该按照C中的convention去使用union，这是我这篇文章要给出的观点。虽然C 使得我们可以扩展一些新的东西进去，但是，我建议你不要那样去做，看完这篇文章之后，我想你大概也是这么想的。 C由于没有类的概念，所有类型其实都可以看作是基本类型的组合，因此在union中包含struct也就是一件很自然的事情了，到了C 之后，既然普遍认为C 中的struct与class基本等价，那么union中是否可以有类成员呢?先来看看如下的代码: struct TestUnion { TestUnion() {} }. typedef union { TestUnion obj. } UT. int main (void) { return 0. } 编译该程序，我们将被告知: error C2620: union ‘ __unnamed ‘ : member ‘ obj ‘ has user-defined constructor or non-trivial default constructor 而如果去掉那个什么也没干的构造函数，则一切OK。 为什么编译器不允许我们的union成员有构造函数呢?我无法找到关于这个问题的比较权威的解释，对这个问题，我的解释是: 如果C 标准允许我们的union有构造函数，那么，在进行空间分配的时候要不要执

行这个构造函数呢?如果答案是yes，那么如果TestUnion的构造函数中包含了一些内存分配操作，或者其它对整个application状态的修改，那么，如果我今后要用到obj的话，事情可能还比较合理，但是如果我根本就不使用obj这个成员呢?由于obj的引入造成的对系统状态的修改显然是不合理的.反之，如果答案是no，那么一旦我们今后选中了obj来进行操作，则所有信息都没有初始化(如果是普通的struct，没什么问题，但是，如果有虚函数呢?)。更进一步，假设现在我们的union不是只有一个TestUnion obj，还有一个TestUnion2 obj2，二者均有构造函数，并且都在构造函数中执行了一些内存分配的工作(甚至干了很多其它事情)，那么，如果先构造obj，后构造obj2，则执行的结果几乎可以肯定会造成内存的泄漏。 鉴于以上诸多麻烦(可能还有更多麻烦)，在构造union时，编译器只负责分配空间，而不负责去执行附加的初始化工作，为了简化工作，只要我们提供了构造函数，就会收到上面的error。 