1. **(2016判断)**因为任何一个COM组件都有QueryInterface函数,并且COM组件的客户程序在COM组件实例运行过程中可以利用该函数动态查询组件实例的接口并使用,所以,COM组件的客户程序事先并不需要知道COM组件的所有接口,只需运行时对COM组件实例以”先查询后使用”的范式使用即可,这也是COM组件支持动态链接的基本含义.

**回答:**

1. **(2016判断)**因为任何一个COM组件都支持IUnknown接口,所以,应该对该接口进行单独统一的实现,这样的实现一旦完成,就可以为任何一个COM组件的实现所复用.

**回答:**IUnknown接口的实现是不能够复用的,一方面反映在QueryInterface函数上,这个函数能返回的全部接口类型,或者称为接口集,对于用户来说就是他能接触到的COM组件的全部,不同的COM组件本质上的不同实际上就反应在这里,因为用户并不清楚接口的具体实现是什么,所以如果两个COM组件的QueryInterface返回行为完全一致,那就表示他们是同一个COM组件.所以不同的组件,其IUnknown接口的实现必然不同,另一方面,其中的引用计数必须对每个不同的组件单独生效,如果多个组件共用引用计数,必将引起混乱,所以IUnknown接口的实现是不能复用的,单独同一实现没有意义.

1. **(2016判断)**一方面,在设计并实现一个动态链接库时,有时需要提供一个回答该动态链接库能否被卸载的导出函数,假设这样的函数名为F(事实上这样的函数有固定的名称),另一方面,可以把多个COM组件实现在同一个动态链接库中,假设这样的一个动态链接库名为L,在为L实现F时,基本算法是根据L中所有组件的所有实例个数的总和是否为0给出真假返回结果.

**回答:**一个DLL可以被卸载的前提是它里面提供的组件和类工厂全部停止使用,所以仅仅统计组件的实例总和为0就得出这个DLL可以被卸载是不正确的,必须要同时确认所有的类厂也已经停止使用,在COM中,实现方法是在使用类厂和使用结束时分别调用一组函数对上述记录总个数的变量进行增减,如此便能保证在这个变量为0时,所有的实例已经注销并且所有的类厂都已经停止使用,此时可以卸载DLL

1. **(2016判断)**COM组件规范规定了每个COM组件实例都有引用计数器及相关的两个函数AddRef和Release,目的是由组件实例协助操作系统内存管理程序控制组件实例的消亡.其实,为了达到同样的效果,可以采用以他方法取代引用计数方案,例如,如果在类厂接口中增加一个控制组件实例消亡的函数并予以实现,就可以有客户程序通过类厂来控制组件实例的消亡,此方案中组件实例的产生和消亡均可以由客户程序通过类厂灵活控制.

**回答:**

1. **(2015判断)**COM组件的接口是公开的,一个COM组件的客户程序在设计时就已经知道了该组件所提供的全部接口,所以,客户程序没必要利用QueryInterface函数查询该组件的接口,知道了还查,多此一举,所以可以得出结论,COM组件不必提供QueryInterface函数.

**回答:**首先,客户能够使用COM组件提供的多少功能取决于客户知道COM组件中的多少接口,这是因为往往用户是不知道全部的接口的,例如,一个程序使用了某个COM组件中的一个接口并调用了一个函数,然后这个COM组件升级了,这个函数有了新的实现并添加到了一个新的接口中,这种情况下,很可能在某个客户环境下同时存在多个版本的组件,这时候用户程序必须先判断当前的COM组件里面有没有新版的接口,如果没有,就要调用旧版的接口完成操作,如果没有QueryInterface,就不能解决这个问题,那样客户程序将会和某个特定的COM组件绑定(因为接口的使用都被写死在代码里面了),动态链接就无法实现了.

1. **(2015判断)**为了区分不同的接口,COM组件规范规定了每个COM组件都有IUnknown接口,而不同组件的IUnknown接口毕竟是属于不同组件的,所以,为了区分不同组件的IUnknown接口,必须为每个COM组件的IUnknown接口定义不同的标识.

**回答:** 为每个COM组件的IUnknown接口定义不同的标识是不必要的.IUnknown接口的确不能区分不同的组件,但是用户程序只要使用QueryInterface查询具体的业务接口就能轻易区分出不同的组件,况且,只要找到了用户想用的接口,由于GUID的唯一性,这个接口中的方法一定是客户程序想用的那个,客户程序根本不需要去区分这个方法是从哪个COM中调用来的,所以对IUnknown接口进行唯一标识没有意义.

1. **(2015判断)**COM组件规范规定了接口不变性,即对已经发出去的接口不做任何变化,新版本的组件只能增加新的接口,该规定的好处是利用兼容,但是也为COM的演化带来了不利影响,使得新版本的COM组件只能增减新的功能,而对于已有功能不能做任何修改.

**回答:**并不是只能新增功能,不能修改功能.对于已有功能的修改可以通过继承原有接口实现新接口的方式,或者通过内部委托调用并开放新接口等方法进行,无论怎么修改,都保证原有的应用程序依然可以按照原先发布的方法调用原来的接口,不需要做任何修改

1. **(2015判断)**COM组件之所以支持动态链接,本质原因是他利用了动态链接库进行实现.

**回答:**COM组件支持动态链接的根本原因是IUnknown接口和QueryInterface方法,这允许客户应用程序能够在运行时根据情况决定使用COM中提供的哪个接口,进一步决定调用何种方法,如果不使用COM标准,即使使用动态链接库,也不一定能最大限度的利用动态链接功能.所以,利用动态链接的最佳方法是COM,动态链接能力来自于动态链接库,但是使用DLL却不是COM支持动态链接的根本原因.

1. **(2014简答)**COM组件区别于普通可执行程序的特征有哪些?

**回答:**使用COM组件编写的程序由多个部分组成,他们分别编译,在运行时进行动态链接,每个组件可以使用不同的编程语言遵循COM标准实现,组件升级时不改变原有接口,对原有客户程序保持兼容,因此升级时不需要将整个程序重新编译,组件可以分布在同一个机器上,也可以分布在网络环境中.普通可执行程序必须进行统一的编译链接,某个部分发生升级,必须将整个程序进行重新发布,一般以单文件形式存在于一台机器上.整个程序的各个功能模块等部分是一个整体,不可分割和独立升级,也不能与其他程序共享.

1. **(2014简答)**用C++或Java给出IUnknown接口中的QueryInterface函数原型,简述函数的参数以及返回结果的语义.

**回答:**HRESULT \_stdcall QueryInterface(const IID&iid,void\*\*ppv)

第一个参数是一个能够唯一标识用户要查询的接口的ID;

第二个参数是输出参数,如果查询成功,被赋值为用户请求的接口类型的指针,否则赋值为NULL

函数的返回值表示QueryInterface函数的执行结果状态,是一个32位值,其中包含成功/失败等执行信息.

1. **(2014简单)**简述工厂接口的作用.

**回答:**IClassFactory接口是创建组件实例的标准接口,客户程序可以使用此接口创建组件的实例,使用时要先使用CoGetClassObject函数创建某个CLSID对应的类厂,得到的类厂对象只能生成创建它时使用的那个CLSID所对应的那个组件的实例,IClassFactory接口中的方法CreateInstance在创建组件实例的同时还将提供用户请求的接口指针.同时IClassFactory接口中还有LockServer函数用来增加类厂的引用计数,该引用计数用于判断动态链接库是否可以卸载.

1. **(2014简答)**简述引用计数器以及IUnknown中AddRef和Release函数的作用.

**回答:**引用计数用于判断一个组件什么时候能够从内存中注销,每当有客户程序获取接口时,调用AddRef()方法,引用计数加一,每当某个接口使用完毕,调用Release()方法,引用计数减一,当引用计数为0时,表明该组件已经没有必要存在于内存中了,此时可以注销.

1. **(2014简答)**叙述利用DLL实现一个COM组件的一般过程.

**回答:**

* 1. 定义业务接口,每个接口都要继承IUnknown接口
  2. 为组件和每个接口分配唯一标识符
  3. 实现接口中的功能
  4. 定义类厂接口,继承自IClassFactory接口,分配唯一标识符CLSID
  5. 实现类厂接口中的功能
  6. 编写DLL所需的注册组件到注册表的函数,解除注册的函数,与类厂相关的函数等函数和所需的DEF文件
  7. 进行编译链接,生成DLL

1. **(2012简答)**什么是动态链接?COM组件规范是如何支持动态链接的?

**回答:**动态链接指的是,在程序的编译时期,不对各个组成部分进行链接,而在运行时进行链接.COM标准规定所有的组件都必须支持IUnknown接口并实现QueryInterface方法,当用户程序进入运行时,可以根据情况决定使用组件中的哪个接口,并通过QueryInterface方法查询得到这个接口进而使用,这使得客户程序可以在运行时决定使用组件提供的哪个函数,因此COM组件支持进行动态链接.

1. **(2012简答)**接口不变性约束的内容是什么?其目的是什么?

**回答:**接口不变性指的是,一旦接口发布出去,就不能发生包括函数的顺序在内的任何变化,其目的是保障组件可以向前兼容,更新组件不会使得依赖旧版组件的应用程序无法正常运行.

1. **(2012简答)**在利用动态链接库实现一个COM组件时,为了使得客户程序合理的得到并使用组件实例,如何利用工厂方法设计导出函数?请给出函数原型和说明,并回答这样设计的好处是什么.
2. **(2012简答)** 每个接口有唯一的标识,每个组件有唯一的标识,标识应该由程序生成,根据你对标识的理解,描述能够生成标识的程序的设计思路.

**回答:**考虑到可能需要大量的标识,所以标识的位数要足够多(例如128位似乎足够大了),为了保证不同区域的用户不会生成同样的标识,我们可以采集运行程序的机器的网卡物理地址或者CPU编号等唯一硬件编号,另外为了保证同一用户生成的标识不能重复,再采集系统时间或者网络时间(例如相对于1970年1月1日的毫秒数等),对这两项组合形成的字符串进行散列运算(例如MD5或者RSA等),最终生成等长的字符串,该字符串能够满足对于唯一标识的要求.

1. **(2011简答)**简述IUnknown接口中QueryInterface函数的作用.

**回答:**QueryInterface函数以所需使用的接口IID为参数,能够查询到该IID所对应的接口指针,进而可以使用该接口,也可以用于确定组件中是否含有某个特定的接口,该函数用于在客户应用程序的运行时获取所需的接口,进而调用所需方法,是COM组件支持动态链接的基础.

1. **(2011简答)**客户程序通过接口调用COM组件实例的AddRef函数和Release函数的语义(含义、意图)是什么?

**回答:**调用AddRef可以使引用计数加一,release可以使引用计数减一,使用这对方法调整引用计数,组件将在引用计数为0的时候从内存注销,有利于节省内存,控制组件的生命周期.

1. **(2011简答)**在利用动态链接库实现COM组件时,为了能够让客户程序控制组件实例产生的时机以及个数,如何设计导出函数以及相关的接口?
2. **(2010判断正误)**一个COM组件的所有接口发布出去后,就不能对这个组件进行任何修改变动.

**回答:错误.**COM组件的发布后,不能对接口进行修改,但是组件却可以修改和升级,可以通过新增接口的方式修改原有方法的实现或提供新方法.

1. **(2010判断正误)**任何COM组件必须要提供IUnknown接口和IClassFactory接口.

**回答:错误.**任何组件都要提供IUnknown接口,但却不是所有的组件都要提供IClassFactory接口.

1. **(2009判断正误)**在不考虑了类厂接口的情况下,对于COM组件的客户程序来说,获得了COM组件的一个接口就相当于获得了COM组件的一个实例.

**回答:**

1. **(2009判断正误)**对于一个COM组件已经发布出去的接口不能做任何修改.

**回答:正确**

1. **(2008简答)**实现COM组件时,一般提供IClassFactory接口,简述其作用.

**回答:**IClassFactory接口是创建组件实例的标准接口,客户程序可以使用此接口创建组件的实例,使用时要先使用CoGetClassObject函数创建某个CLSID对应的类厂,得到的类厂对象只能生成创建它时使用的那个CLSID所对应的那个组件的实例,IClassFactory接口中的方法CreateInstance在创建组件实例的同时还将提供用户请求的接口指针.同时IClassFactory接口中还有LockServer函数用来增加类厂的引用计数,该引用计数用于判断动态链接库是否可以卸载.

1. **(2007简答)**用新版本组件替换旧版本组件,不会影响旧版本程序的使用,其原因是什么?

**回答:**因为COM组件有接口不变形约定,一旦接口发布,就不能对接口进行修改,新版本只能添加新的接口或者通过新接口重写原有的功能,由于原来的接口还在,所以对于旧版程序来说,他仍然可以使用和未升级时完全一样的方法与组件交互,因此用新版本替换旧版本的组件,不会影响旧程序的使用.

1. **(2007简答)**简述COM组件和程序模块之间的不同.

**回答:**一个COM组件同一个应用程序类似,都是已经编译连接好并可以使用的二进制代码,二进制形式的组件能有效的嵌入到其他组件中,从而组成一个应用程序,而一个程序模块是固化了的整体系统的一部分,缺少灵活性,不易维护更新,也难以被其他人开发的程序复用.

1. **(2007简答)**采用何种技术手段区分不同的组件?

**回答:**用户可见的组件实际上是接口集,且所有接口都是IUnknown接口的子接口,如果我们两个组件查找相同的接口,若都找到且找到的接口指针指向同一处,则这两个组件就是同一个组件,否则(比如一个组件中含有的接口在另一个组件中没有,或者同一接口的指针不同)就是不同的组件.