C语言中的面向对象(4)－面向对象思想 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/135/2021_2022_C_E8_AF_AD_E8_A8_80_E4_B8_AD_c97_135020.htm 经常听见别人说面向对象的程序设计，以前在学校上课的时候，也有开面向对象程序设计这门课。可是不幸的是，这些都是以C＋＋，甚至VC＋＋为基础的。而更加不幸的是，多年以来我一直是一个C的使用者。在学校的时候，我主要做的是硬件上的驱动层，和底层功能层。在工作以后，又做的是手机上的软件开发，所有这些都是和C离不开的。虽然我不得不说，C＋＋是一门很好的语言，但是它的编译速度，代码效率，编译后的代码大小都限制了它在嵌入式上的应用。（但现在的嵌入式CPU越来越快，内存容量变大。我觉得用C＋＋也应该没有什么问题。这使我觉得似乎是嵌入式编译器的限制。虽然菲利普和TI好像都有C＋＋的编译器，但是似乎没人用这个。难道是太贵了？ 但不管怎么说，嵌入式应用中，C语言的普遍使用是肯定的） 那么在面向过程的时代产生的C语言能否使用面向对象的思想呢？我认为是肯定可以的，C＋＋不过是在语言级别上加入了对对象的支持，同时提供了丰富的对象库。而在C语言下，我们只好自力更生了。 一、 面向对象思想的目的是框架化，手段是抽象 相信很多人都明白面向对象讲了什么：类，抽象类，继承，多态。但是是什么原因促使这些概念的产生呢？ 打个比方说：你去买显示器，然而显示器的品牌样式是多种多样的，你在买的过程中发生的事情也是不可预测的。对于这样的事情，我们在程序语言中如何去描述呢。面向对象的思想就是为了解决这样的问题。编写

一个程序（甚至说是一个工程），从无到用是困难的，从有到丰富是更加困难的。面向对象将程序的各个行为化为对象，而又用抽象的办法将这些对象归类（抽象），从而将错综复杂的事情简化为几个主要的有机组合（框架化）。 其实我们的身边很多东西都是这样组成的：比如说电脑：电脑是由主板，CPU加上各种卡组成的。这就是一个框架化。而忽略不同的CPU，不同的主板，不同的声卡，网卡，显卡的区别，这就是抽象。再比如说现在的教育网：是由主核心节点：清华，北大，北邮等几个，然后是各个子节点，依次组成了整个教育网网络。 所以我觉得面向对象的编程思想就是：一个大型工程是分层次结构的，每层又由抽象的结构连接为整体（框架化），各个抽象结构之间是彼此独立的，可以独立进化（继承，多态）。层次之间，结构之间各有统一的通讯方式（通常是消息，事件机制）。 二、 以前 C 语言编程中常用的“ 面向对象 ”方法 其实C语言诞生以来，人们就想了很多办法来体现“ 面向对象 ”的思想。下面就来说说我所知道的方法。先说一些大家熟悉的东东，慢慢再讲诡异的。呵呵 1． 宏定义： 有的人不禁要问，宏定义怎么扯到这里来了，我们可以先看一个简单的例子： ＃define MacroFunction Afunction 然后在程序里面你调用了大量的AFunction，但是有一天，你突然发现你要用BFunction了，（不过AFunction又不能不要，很有可能你以后还要调用），这个时候，你就可以＃define MacroFunction Bfunction来达到这样的目的。 当然，不得不说这样的办法是too simple，sometime na&iuml.ve的，因为一个很滑稽的问题是如果我一般要改为BFunction，一半不变怎么办？ 那就只好查找替换了。 2． 静态的入口函数，保

证函数名相同，利用标志位调用子函数： 这样的典型应用很多，比如说网卡驱动里面有一个入口函数Nilan（int FunctionCode，Para*）。具体的参数是什么记不清楚了。不过NiLan的主体是这样的： Long Nilan（int FunctionCode，Para*）{ Switch(FunctionCode){ Case SendPacket: send(….) Case ReceivePacket: receive(…) …..} 写到这里大家明白什么意思了吧。保证相同的函数名就是说：网卡驱动是和pNA 协议栈互连的，那么如何保证pNA 协议栈和不同的驱动都兼容呢，一个简单的办法就是仅仅使用一个入口函数。通过改变如果函数的参数值，来调用内部的各个函数。这样的做法是可以进化的：如果以后想调用新的函数，增加相应的函数参数值就好了。如果我们将网卡驱动和pNA 协议栈看作两个层的话，我们可以发现： 层与层之间的互连接口是很小的（这里是一个入口函数），一般是采用名字解析的办法而不是具体的函数调用（利用FunctionCode调用函数，Nilan仅仅实现名字解析的功能） ——！接口限制和名字解析 接口限制：层与层之间仅仅知道有限的函数 名字解析：层与层之间建立共同的名字与函数的对应关系，之间利用名字调用功能。 3．CALLBACK函数。 我觉得这是C语言的一个创举，虽然它很简单，就象如何把鸡蛋竖起来一样，但是你如果没想到的话，嘿嘿。如果说静态入口函数实现了一个可管理的宏观的话，CallBack就是实现了一个可进化的微观：它使得一个函数可以在不重新编译的情况下实现功能的添加！但是在最最早期的时候，也有蛮多人持反对态度，因为它用了函数指针。函数指针虽然灵活，但是由于它要访问内存两次才可以调用到函数，第一次访问函数指针，第二次才是真正的函数调用

。它的效率是不如普通函数的。但是在一个不太苛刻的环境下，函数调用本身就不怎么耗时，函数指针的性能又不是特别糟糕，使用函数指针其实是一个最好的选择。但是函数指针除了性能，最麻烦的地方就是会导致程序的“ 支离破碎 ”。试想：在程序中，你读到一个函数指针的时候，如果你愣是不知道这个函数指针指向的是哪个函数，那个感觉真的很糟糕。（可以看后面的文章，要使用先进的程序框架，避免这样的情况） 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com