程序员的十个层次

本文来源 http://developer.51cto.com　2010-11-10 10:39　周伟明

觉得这篇文章写的还蛮意思，全篇很长我简要的把该文改写一下，以突出十个层次的区别：

中国的程序员水平比西方程序员水平差，还是中国有许多优秀的程序员达到或超过了西方程序员同等水平呢？要解决这个问题，必须先知道程序员有多少种技术层级，每个层级需要什么样的技术水平，然后再比较中国和西方在各个技术层级的人数，就可以知道到底有没有差距，差距有多大。

当然，对于如何划分程序员的技术层级，不同公司或不同人会有不同的划分标准，下面的划分仅代表个人的观点，如有不当之处，还请砸板砖予以纠正。

一、菜鸟

第 1 层楼属于地板层，迈进这层楼的门槛是很低的。基本上懂计算机的基本操作，了解计算机专业的一些基础知识，掌握一门基本的编程语言如 C/C++，或者 Java，或者 JavaScript，...，均可入门迈进这层。

二、大虾

从第 1 层爬到第 2 层相对容易一些，以 C/C++程序员为例，只要熟练掌握 C/C++编程语言，掌握 C 标准库和常用的各种数据结构算法，掌握 STL 的基本实现和使用方法，掌握多线程编程基础知识，掌握一种开发环境，再对各种操作系统的 API 都去使用一下，搞网络编程的当然对 socket 编程要好好掌握一下，然后再学习一些面向对象的设计知识和设计模式等，学习一些测试、软件工程和质量控制的基本知识，大部分人经过 2～3 年的努力，都可以爬到第 2 层，晋升为"大虾"。

三、牛人

由于"大虾"们经常被一些疑难问题给卡住，所以有了"大虾"们只好继续学习，他们需要将原来所学的知识进一步熟练掌握，比如以熟练掌握 C++编程语言为例，除了学一些基础性的 C++书籍如《C++ Primer》，《Effective C++》，《Think in C++》，《Exception C++》等之外，更重要的是需要了解 C++编译器的原理和实现机制，了解操作系统中的内部机制如内存管理、进程和线程的管理机制，了解处理器的基础知识和代码优化的方法，此外还需要更深入地学习更多的数据结构与算法，掌握更深入的测试和调试知识以及质量管理和控制方法，对各种设计方法有更好的理解等。

学习上面说的这些知识不是一挥而就的，不看个三五十本书并掌握它是做不到的。以数据结构算法来说，至少要看个 5～10 本这方面的著作；以软件设计来说，光懂结构化设计、面向对象设计和一些设计模式是不够的，还要了解软件架构设计、交互设计、面向方面的设计、面向使用的设计、面向数据结构算法的设计、情感化设计等，否则是很难进到这个楼层的。

四、大牛

从第 3 层爬到第 4 层可不像上面说过的那几层一样容易，要成为大牛的话，你必须要能做牛人们做不了的事情，解决牛人们解决不了问题。比如牛人们通常都不懂写操作系统，不会写编译器，不懂得 TCP/IP 协议的底层实现，如果你有能力将其中的任何一个实现得象模象样的话，那么你就从牛人升级为"大牛"了。

当然，由于各个专业领域的差别，这里举操作系统、编译器、TCP/IP 协议只是作为例子，并不代表成为"大牛"一定需要掌握这些知识，以时下热门的多核编程来说，如果你能比牛人们更深入地掌握其中的各种思想原理，能更加自如的运用，并有能力去实现一个象开源项目 TBB 库一样的东西，也可以成为"大牛"，又或者你能写出一个类似 Apache 一样的服务器，或者写出一个数据库，都可以成为"大牛"。

当"牛人"晋升为"大牛"，让"牛人们"发现有比他们更牛的人时，对"牛人"们的心灵的震撼是可想而知的。由于牛人们的数量庞大，并且牛人对大虾和菜鸟阶层有言传身教的影响，所以大牛们通常能获得非常高的社会知名度，几乎可以用"引无数菜鸟、大虾、牛人竞折腰"来形容，看看前面提过的 Linus Torvalds 等大牛，应该知道此言不虚。

虽然成为"大牛"的条件看起来似乎很高似的，但是这层楼并不是很难爬的一层，只要通过一定的努力，素质不是很差，还是有许多"牛人"可以爬到这一层的。由此可知，"大牛"这个楼层的人数其实并不像想像的那么少，例如比尔·盖茨之类的人好像也是属于这一层的。

五、专家

当大牛们真正动手做一个操作系统或者类似的其他软件时，他们就会发现自己的基本功仍然有很多的不足。以内存管理为例，如果直接抄袭 Linux 或者其他开源操作系统的内存管理算法，会被人看不起的，如果自动动手实现一个内存管理算法，他会发现现在有关内存管理方法的算法数量众多，自己并没有全部学过和实践过，不知道到底该用那种内存管理算法。

看到这里，可能有些人已经明白第 5 层楼的奥妙了，那就是需要做基础研究，当然在计算机里，最重要的就是"计算"二字，程序员要做基础研究，主要的内容就是研究非数值"计算"。

非数值计算可是一个非常庞大的领域，不仅时下热门的"多核计算"与"云计算"属于非数值计算范畴，就是软件需求、设计、测试、调试、评估、质量控制、软件工程等本质上也属于非数值计算的范畴，甚至芯片硬件设计也同样牵涉到非数值计算。如果你还没有真正领悟"计算"二字的含义，那么你就没有机会进到这层楼来。

六、学者

当"专家"们想继续往上一层楼爬时，他们几乎一眼就可以看到楼梯的入口，不过令他们吃惊的是，楼梯入口处竖了一道高高的门槛，上面写着"创新"二字。不幸的是，大多数人在爬到第 5 层楼时已经体能消耗过度，无力翻过这道门槛。

以查找为例，并不是去天天盯着那些复杂的查找结构和算法进行研究，你需要做的是将二分查找、哈希查找、普通二叉树查找等基础性的知识好好地复习几遍。

以哈希查找为例，首先你需要去将各种冲突解决方法如链式结构、二次哈希等编写一遍，再试试不同种类的哈希函数，然后还需要试试在硬盘中如何实现哈希查找，并考虑数据从硬盘读到内存后，如何组织硬盘中的数据才能快速地在内存中构建出哈希表来，...，这样你可能需要将一个哈希表写上十几个不同的版本，并比较各个版本的性能、功能方面的区别和适用范围。

总之，对任何一种简单的东西，你需要考虑各种各样的需求，以需求来驱动研究。最后你将各种最基础性的查找结构和算法都了然于胸后，或许某天你再看其他更复杂的查找算法，或者你在散

步时，脑袋里灵光一现，突然间就发现了更好的方法，也就从专家晋升为"学者"了。

七、大师
从第 6 层楼爬到第 7 层楼，并没有多少捷径可走，主要看你有没有足够的能量。你如果能象 Hoare 一样设计出一个快速排序的算法；或者象 Eugene W. Myers 一样设计出了一个用编辑图的最短路径模型来解决 diff 问题的算法；或者象 M.J.D. Powell 一样提出了一个能够处理非线性规划问题的 SQP 方法；或者你发现基于比较的排序算法，它的复杂度下界为 O(NLogN)；或者你发现用栈可以将递归的算法变成非递归的；或者你设计出一个红黑树或者 AVL 树之类的查找结构；或者你设计出一个象 C++或 Java 一样的语言；或者你发明了 UML；...，你就爬到了第 7 层，晋升为"大师"了。

八、科学家
科学家向来都是一个神圣的称号，因此我把他放在了“大师”之上。要成为科学家，你的贡献必须超越大师，不妨随便举一些例子。

如果你象 Dijkstra 一样设计了 ALGOL 语言，提出了程序设计的三种基本结构：顺序、选择、循环，那么你可以爬到第 8 层楼来。顺便说一下，即使抛开这个成果，Dijkstra 凭他的 PV 操作和信号量概念的提出，同样可以进到这层楼。

如果你象 Don Knuth 一样，是数据结构与算法这门学科的重要奠基者，你也可以进到这层楼来。当然，数据结构和算法这门学科不是某个人开创的，是许多大师和科学家集体开创的。
如果你象巴科斯一样发明了 Fortran 语言，并提出了巴科斯范式，对高级程序语言的发展起了重要作用，你也可以进到这层楼来。

九、大科学家
进入这层楼的门槛通常需要一些运气，比如某天有个苹果砸到你头上时，你碰巧发现了万有引力，那么你可以进到这层楼来。当然，万有引力几百年前就被人发现了，如果你现在到处嚷嚷着说你发现了万有引力，恐怕马上会有人打 110，然后警察会把你送到不正常人类的聚集地去。因此，这里举万有引力的例子，只是说你要有类似的成就才能进到这层楼来。

当然，程序员们最关心的是自己有没有机会变成大科学家。既然计算机这门大学科的开创性成果早就被冯·诺伊曼、图灵等人摘走了，那么程序员们是不是没有机会变成大科学家了呢？我们的古人说得好：“江山代有才人出，各领风骚数百年”，现在在计算机这门学科下面诞生了许多非常重要的大的分支，所以你还是有足够的机会进到这层楼的。

如果你能够彻底解决自然语言理解（机器翻译）这门学科中的核心问题， 或者你在人工智能或者机器视觉（图像识别）方面有突破性的发现，那么你同样可以轻易地晋升为“大科学家”。这样当某天你老了去世时，或许那天国人已经觉醒，你也能享受到如 Dijkstra 一样的待遇，有满城甚至全国的人去为你送葬。

十、大哲
看了这层楼的名字“大哲”，可能不少人已经猜到了这层楼的秘密，那就是你的成果必须要上升到哲学的高度，你才有机会能进到这层来。

当然，上升到哲学高度只是一个必要条件，牛顿的万有引力似乎也上升到了哲学的高度，因为不知道引力到底是怎么来的，但是牛顿没有被划到这一层，因为进到这层还有另外的条件，那就是你的成果必须引起了哲学上的深度思考，并能让人们的世界观向前跨进一大步。窃以为牛顿、爱因斯坦等人的成就还达不到让人们世界观向前跨进一大步的程度。

所以，这层楼中的人的成就对我们普通人认识世界非常重要，你可以不学相对论，但是你不可以不对这层楼的人所作出的成就不了解，否则你的世界观就是极其不完整的，会犯许多认识上的错误。不幸的是，中国的科普知识普及还不够到位，知道这层楼成就的人好像并不多，程序员中恐怕更少。下面就来看看这些用一只手的手指数得清的大哲们，到底有什么成就，能比万有引力定律和相对论还重要。

1、希尔伯特 (1862～1943)

第 1 位进到此楼层是一位名叫“希尔伯特”的大数学家，如果你学过《泛函分析》，那么你在学习希尔伯特空间时可能已经对这位大数学家有所了解；如果你不是学数学出身的，又对数学史不感兴趣的话，恐怕你从来没有听说过这个名字。不过如果我问一下，知不知道二次世界大战前世界数学中心在那里，你肯定会有兴趣想知道。

不妨说一下，二战前整个世界的数学中心就在德国的哥廷根，而我们这位大数学家希尔伯特便是它的统帅和灵魂人物。即使在二战期间，希特勒和丘吉尔也有协定，德国不轰炸牛津和剑桥，作为回报，英国不轰炸海德堡和哥廷根。

2、哥德尔 (1906~1978)

这位大哲的名字叫“哥德尔 (G?del) ”，你可能从来也没有听说过这个名字，即使你读了一个数学系的博士学位，如果你的研究方向不和这位大哲对口的话，你也不一定了解这位大哲的成就，更不知道他的成果对我们这个世界有何意义。

简单地说，这位大哲 20 多岁时就证明了两个定理，一个叫做“哥德尔完全性定理”，另一个更重要的叫做“哥德尔不完全性定理”。你也许会觉得奇怪，第 9 层楼的成就就已经上升到了公理的高度，这种证明定理的事情不是学者和大师们做的事情吗？怎么能比第 9 层楼的成就还高呢？下面就来简单说一下这两个定理的含义，你就会明白这属于系统级的定理，绝不是普通的定理和公理所能比拟的。

“哥德尔完全性定理”证明了逻辑学的几条公理是完备的，即任何一个由这些公理所产生出的问题，在这个公理系统内可以判定它是真的还是假的，这个结论表明了我们人类所拥有的逻辑思维能力是完备的。这条定理并不能将其带入这层楼来，带其进入这层楼的是另一条定理。

可能你看过《未来战士》、《黑客帝国》、《I，Robot》之类的科幻电影，于是你产生制造一个和人一样或者比人更高一级的智能机器人的想法，这就引入了一个达到哲学高度的问题，“人到底能不能制造出具有和人一样的思维能力的机器来？”。

我只能告诉你，“你的愿望是良好的，但现实是残酷的”。如果你仔细思考一下不完全性定理的含义，并结合现代计算机所具有的能力分析一下，你会发现这个问题的答案暂时是否定的。如果你

想造出和人一样思维能力的机器，那么你需要去好好学习这位大哲及其后续研究者的成果，并在他们的基础上有新的突破才行。

3、海森堡 (1901~1976)

海森堡这个名字相信没有几个人不知道的，大部分人在学习物理时都学过他的“测不准关系”，也就是因为这个“测不准关系”，海森堡爬到了第十层楼。

如果你看过《时间简史》和《霍金讲演录－黑洞、婴儿宇宙及其他》，你也许已经了解测不准关系的威力，所以这里不想做过多的讨论，只谈一些和本土产生的哲学思想相关的东西。

十一、超越第十层的上帝
看了上面的小标题，你可能会觉得奇怪，这篇文章不是讲“程序员的十层楼”吗？怎么冒出了第11层来了？

其实这并不矛盾，程序员确实只有十层楼，因为爬到第11层时，已经变成上帝，不再是程序员了；所以超出10层楼本身并不重要，关键的问题是看你有没有能力变成上帝。

1、谁是上帝？

菜鸟们认为Linus Torvalds是程序员中的上帝，看完了前面各层楼的介绍，此时再看到这句话，相信你要忍不住在心里笑起来。当然，你会不会笑起来是事先注定的。Don Knuth也不是上帝，他离上帝还有三层楼的距离。即使是大哲们，他们离天堂也还差一层楼，因此这个世界上有史以来还没有任何一个人变成过上帝。

我们感兴趣的是，将来会不会有人爬到比大哲们更高的楼层上，变成了上帝。

要变成上帝，你得有上帝一样的能力，上帝会造人，你会吗？

你也许会怯生生地问：“我可以和爱人生小孩，算不算造人？”，你可能还会理直气壮地说：“现在生物学上都可以克隆人了，早就有人掌握了造人的方法”。

事实上克隆人需要有人的体细胞，必须要先有人才会有体细胞。上帝造人时，这个世界上并没有人，是从无生命的物质“尘土”中创造出的人。因此，用最原始的方法生人和克隆人都是从有生命信息的物质中生人，不能算作造人。

读后感：
终于轮到我来发表一下看法了，这也是我为什么要把这篇文章摘抄下来的原因。可以看出本文作者是为C/C++程序员并且受过良好的教育，以及高于编程以外的思考。要说作者参透了一切，看破了红尘。那到未必，不过作者的十个层次分级对一名程序员来说一个很好的指导性意见。最后用《天道》中的《自嘲》做为结束：
卜算子・自嘲
本是后山人，偶做前堂客，醉舞经阁半卷书，坐井说天阔。
大志戏功名，海斗量福祸，论到囊中羞涩时，怒指乾坤错。