### 专访李运华：程序员如何在技术上提升自己

**CSDN：请和大家介绍下你和目前所从事的工作。**

**李运华：**大家好，我是来自UC的软件工程师，已经在软件领域摸爬滚打10年了，先后经历了电信行业和互联网行业。毕业后首先是去了华为，做核心网的系统开发，做过软件工程师、系统分析师；后来换到现在所在的UC，先后做过技术研究、基础平台开发、社区系统，现在在九游游戏中心担任项目leader。虽然说是leader，但其实管理和开发的时间是3/7比例，所以至今其实还是个“码农”。

非常爱好技术，“精通”后端开发技术（C++、Java、Linux、MySQL等），目前正在学习前端，力争成为FSE；虽然很爱好技术，但并不局限于只在技术上提升自己，管理技能、综合技能等都在持续学习和提升。业务时间喜欢看书，经济学、社会学、心理学都有广泛的涉猎。

个人的梦想是能够带领或者加入一个团队，创造一个伟大的产品，大家也可以看到我的个人技能提升基本上都是朝着这个目标的。

**CSDN：有着10余年软件行业经验的你，是如何一步步走上这软件开发之路的？**

**李运华：**我来自农村，高三的时候才见过电脑，当时就体验过一两次，我还记得是dos的系统，虽然很难用，但当时就觉得很神奇，加上我高中毕业的时候正好是计算机行业最火的时候，于是就选择了计算机专业。

但是上大学后前三年并没有投入太多时间，基本上是玩（打游戏、踢球）去了，到了大四找工作的时候，也不知道是什么原因，感觉突然觉醒了，觉得自己想做软件开发，于是开始买了一些书看，看了《深入浅出MFC》后觉得很神奇，很有意思，于是后来找工作时就坚定了找软件开发的念头。现在回想起来感觉当时真是一根筋，测试、产品、运维等都不想做，只想做开发，即使面试不过也没有动摇 :)

后来机缘巧合进入了华为，从此开始了软件工程师之路，一直到现在，都是从事软件开发相关的工作。

**CSDN：你做过软件开发工程师、系统分析师、系统设计师等，能够回顾一下自己的职场晋升之路，有什么心得和体会可分享？**

**李运华：**毕业后加入华为，先是做了大约2年多的软件开发工程师，后来加入系统组做了系统分析师（华为俗称SE，其实就是系统分析和设计），做了大约2年多后离开华为加入UC，一直从事软件开发的工作，先后做过技术研究（MySQL、Memcached等）、软件设计和开发，项目leader工作，一路从高级工程师逐步提升到资深工程师，并担任了项目leader。

回顾我的提升之路，我觉得有几点可以跟大家分享的：

**第一是 “兴趣”。**这也是我认为最重要的一点，一件事情做10年甚至做一辈子，如果没有兴趣的话，我觉得是很痛苦的。兴趣是本能的驱动力，有了兴趣，遇到问题会一直想着怎样去解决，而不是觉得“很难做”；有了兴趣，碰到一个新的东东会觉得很兴奋，而不会觉得是一种负担；有了兴趣，接触到一个东西后就像更加深入的去了解，而不是用过了就不管了。所以我认为如果想在一个行业（不限于软件行业）长期发展并有所提升的话，一定要问问自己是否有足够的兴趣。

我当时之所以换工作，也是因为我对当时的工作内容不感兴趣，因为我更喜欢亲手做出一个产品，而不是找一群人开会讨论然后写个文档就完事了。

**第二是“坚持”。**《异类》一书中提到一个10000小时理论，我觉得非常有道理，意思就是说如果你想成为顶尖人才的话，一定要积累10000小时以上的训练和经验。特别是在软件开发这个领域，技术又多，技术更新又快，如果没有坚持去积累和提升的话，是很难达到一定高度的。操作系统、数据库、网络、编程语言、设计方法等都要掌握，每个技术点又有很多更细的分类。以编程语言来说，C、C++、Java、PHP、Python等主流的都有10来种，每个语言继续深入的话又有很多内容，例如Java可以列出来的有JVM、IO、NIO、网络编程、反射。。。。。。等等。所以这么多的东东，短时间内快速入门还可以，但如果说21天就精通XXX，那是不可能的，必须经过长时间的积累。其实我现在都不敢说我精通什么，只能说相对周围其他人会精通一些。

**第三就是“方法”。**掌握正确的方法，能够让我们事半功倍，更快的提升，一些常见的方法我就不啰嗦了，这里特别分享独家秘方：

**一个秘方是“写博客”，注意这里不是“看博客”，也不是“转载博客”而是“亲自写博客”。**哪样东西你觉得你比较懂了，那么你就写成博客。当你真正去写的时候，你会发现，其实还有很多不懂或者不清楚的地方，这样就会促使你又去学习研究；当你的博客发表后，其他人除了能够从你的博客中学到东西外，也能够帮你发现一些问题或者错误，这样你就更进一步的掌握了；

**另外一个秘方就是“链式学习”。**形象点说，就是你抓住了一个链条的一个链，然后慢慢慢慢把所有的链都拉出来。举一个很简单的例子：socket sever编程。很多人在编程的时候，都是去搜索引擎搜索一下“socket server样例”，然后对照样例很快就写完功能了。然后呢。。。。。。很多人没有然后了，完成任务就不管了。其实这样做就错过了一次提升自己的好机会。

“链式学习”则不一样，它是这样做的：我通过搜索引擎搜索到样例完成工作后，我会问自己很多问题：样例中的api每个参数都是什么含义，有哪些注意事项，还有其它API么？为了解决这些问题，我就可能去找本书看，某个编程语言的socket编程；看完以后我知道socket编程的全貌和一些注意事项，而这些是通过搜索引擎搜索的样例中没有的；知道socket编程的全貌后，我又会问自己：操作系统是怎么做的呢？ 那我又会去看《UNIX网络编程》，看完后我就对操作系统层面的又掌握更多了；看完《UNIX网络编程》后，我又知道socket是和tcp/ip相关的，那我又会去看《TCP/IP协议详解》。。。。。。

这样去做就是一条学习链： socket server编程 -> socket 编程 -> UNIX网络编程 -> TCP/IP协议，后面还可以继续不断拓展下去。如此不断的拓展和深入，一个很小的契机就能初始你学到很多东西，而这些东西在以后的工作中某些时刻就派上用场了。

我之前在华为是在Windosw平台上用MFC开发，后来到了UC转为Linux平台开发，用这种方法，大约用了2年就熟练掌握了Linux平台相关的开发技术，包括Linux、MySQL、C++、Java、PHP等

**第三个秘方就是“闭环学习”。**“链式学习”适合于一组相关联的知识或者技能的学习，而“闭环学习”更适合业务、相互配合的知识和技能的学习。由于软件开发是需要多个团队分工合作的，所以绝大部分人都只负责整个系统或者全流程中的一环，这样导致很多人以为只需要将自己负责部分精通就可以了。其实这样不利于个人的发展，一个原因是自己负责的一般都比较窄，可学习和提升的空间可能不多，另外一个原因是很难设计整体上优秀的方案。

而“闭环学习”则不一样，它是这样做的：了解整个功能或者业务的全流程实现，涉及了哪些模块和系统，每个模块和系统主要负责什么功能，涉及到什么技术，性能怎样，有什么注意点。举一个我做个的HTTP的业务样例：从用户点击一个url开始，经过了 浏览器 -> 网络 -> CDN -> Nginx -> PHP -> MySQL -> PHP -> Nginx -> 网络 -> 浏览器，最后呈现在用户面前。我开发的时候只是用PHP开发，但并不只局限于PHP本身，闭环学习就要求全流程中的每个环节都要去了解和熟悉，这样你就可以学到了“浏览器、Nginx、CDN、MySQL”等很多知识。

有的朋友可能会问：这样做有什么用呢？其实用处非常大，一个是当出现问题的时候，有了闭环学习掌握的知识和技能，你就知道哪些地方可能有问题，应该如何处理；另外一个用处是，当你考虑设计方案的时候，就不局限于PHP本身了，也许某个功能Nginx或者CDN或者前端能做的更好，用PHP实现反而很蹩脚。

**CSDN：你对代码有着怎样的情感？对丰富的编程语言是否有什么特别的喜好吗？**

**李运华：**我喜欢写代码，在写代码的过程就是一个创造的过程，我特别享受这种创造的过程，夸张点说：我就是我的软件的造物主，在代码里面我就是“神”，我可以设计我的作品，实现我的作品，改进我的作品。

对于目前丰富的编程语言，我没有什么特别的爱好，对我来说，编程语言就是一个工具，用来实现业务目标或者自己的想法。每种编程语言都有特点和适用场景，哪种语言更适合我的需求我就用哪种，不会执着于用1门语言来包打天下。我先后用过C、C++、Java、PHP、Python、Ruby，现在在学前端编程的JavaScript、Coffeescript，也关注目前还未流行的Go语言等。

**CSDN：平时你最大的爱好是看书，请问这给了你什么样的帮助？有什么好书推荐吗？**

**李运华：**看书有很多好处，对于专业上的书，扩大了知识面和提升了技能；对于非专业的书来说，帮助就更多了，有的书让我学到了很多沟通、交流、为人处世的一些方法和技巧；有的书改变了我的思维方式；有的书提升了自己的知识水平；甚至有的书改变了我的命运。

我看过的书比较多，但我特别向大家推荐如下几本：

**《羊皮卷》**：目前市面上的《羊皮卷》大部分都是心灵鸡汤式的文章的组合，但有一本其中有一篇《选择的力量》，我看了后醍醐灌顶，真的是就像佛家禅宗说的突然“悟道”一样，看了后深受启发，从此后很多为人处世方式都因此而改变了；

**《异类》**：一本从不同视角讲述成功人士到底是如何成功的，会告诉你很多不为人知的故事，能够让你免受心灵鸡汤之害，也能够明白成功既要靠自己努力，也要靠机遇；

**《随机漫步的傻瓜》**：这本书看起来是讲投资的，但其实讲述了一个关于“运气和命运”的问题，让你能够以不同的视角来评价和判断所谓的“成功”

我不太会写书评，光看我的介绍可能有的人没什么感觉，但强烈建议都去看看，真的会有很大收获。

**程序员如何在技术上提升自己**

**CSDN：你因个人时间不多问题，一直专注在CSDN博客，你一天的时间是安排的呢？**

**李运华：**我工作日的一天大约是这样的：早上7:00起床， 7:20送小孩上学，8:00去地铁，9:10分到公司，晚上20:00下班，21:10分回到家，回家后陪小孩玩到大约10:30，然后自己洗澡准备睡觉，大约23:30分上床睡觉；双休日也差不多，就是把上班的时间改为陪小孩了。

平时一般是早上或者晚上浏览一下CSDN，了解一些技术趋势，学习一些技术。

**CSDN：在你眼中，一个优秀的程序员应该具备哪些技能和修养？**

**李运华：**这个问题就像“1千个人眼中有1千个哈姆雷特”一样，每个人都有不同的看法。我认为一个优秀的程序员应该具备如下技能和修养：

**首先是“快速学习能力”。**这里不是说一定要去快速去学习各种各样的新技术，而是说当有需要时，能够快速的学习。很多人开始学新的技术和技能时，一开始就一头扎进去写样例、写Demo、看源码，我认为这不是好的方法，而且比较耗费时间，收效也不明显。

我给大家分享我的4W2H快速学习方法。我在学习新的技术的时候，都是按照这样的步骤去了解的：1）这个技术能解决什么问题（why） 2）比较适合在哪些场景应用（where + when) 3）这个技术跟我已经掌握的哪个知识或技能类似，有什么差别、有什么特点、 有什么优点和缺点（what），4）了解前面的问题后，我才会开始去尝试写写Demo，或者更进一步去应用（How to use） 5）觉得有兴趣或者其实现很牛逼的情况下，我就去研究一下原理机制，看看源码等 （How it implements）

**其次是“良好的理解能力”。**程序员需要将产品人员或者用户用自然语言表述的需求翻译成程序语言。自然语言有一个特点就是通俗但不严谨，而程序语言必须是非常严谨的。如果对产品人员或者用户提出的需求没有很好的理解，即使程序语言写的再漂亮，技巧再高，最后做出来也是一个不符合要求的产品。

记得有一个关于“美女”的笑话：人听到“美女”后的反应是想到“天使面孔魔鬼身材童颜巨乳”，而猪听到“美女”后的反应是“乌克兰大白猪”，猫听到“美女”后的反应是“有着金色光滑皮毛的波斯猫”。如果程序员给了猫一个“天使面孔魔鬼身材童颜巨乳”的美女，猫一定会觉得很难看。

**第三是“持续不断的学习”。**软件开发领域设计的知识和技能太多了。从广度上来说，有操作系统、数据库、编程语言、网络、设计等，编程语言又有几十种；从深度上来说，操作系统、数据库、编程语言等都是可以不断深入去学习的。无论你是从事对技能广度要求更高的业务开发，还是从事对技能深度要求更高开发专项系统，都需要不断的学习，这样才能不断的提升自己的能力。

**第四是“乐于分享”。**如果单纯从个人完成工作的能力来看，可能确实也有很多程序员不爱分享但确实很厉害。但我认为真正优秀的程序员一定是除了自己优秀外，还能让其他人也变得优秀，或者能够贡献优秀的开源项目以降低别人的重复工作。分享的途径有很多种，可以给公司人员做培训，可以写博客，可以贡献开源项目等。

**CSDN：你认为程序员如何“挤”出时间来提升自己或在技术上提升自己？**

**李运华：**前面我提到了“要坚持”，“要持续学习”，大家自然想到的问题就是“时间从哪里来”？毕竟大家都比较辛苦，加班也比较多，有的还需要带孩子，到哪里去找时间学习呢？

我的方法就是“挤”，那么问题又来了，如何挤？ 比如说大家看我的工作日作息时间，早上7:00起床到晚上10:30，基本上就没有什么时间了，10:30以后还要自己休闲一下，打打魔兽玩玩CS，恩，真的很难有时间了，那怎么办？

**关键就在于一个“挤”字，其实就是“利用时间碎片”，10分钟就可以了**，以我最近为例谈谈我现在怎么挤时间的，大家可能就明白了。

1. 7:40 ~ 8:00：送完小孩后到出门去地铁中间有20分钟，看《失控》一章
2. 08:20 ~ 09:00：地铁上带一本书，大约可以看完一章，容易看的书可以看两章
3. 09:30 ~09:50：看技术类的PDF，比如说最近看JQuery，大约可以看一章，不用担心，每天20分钟完全不会影响工作
4. 20:20 ~ 21:00：和早上地铁一样
5. 睡觉前15分钟：看15分钟书，前段时间这样花了大约2个月看完《三国演义》

这样累积下来，**我一天大约可以看2个小时书，今年上半年看了大约23本书。**

以上只是我的一个案例，其实每个人都有很多机会挤出时间的，比如说你习惯8点起床，那么把闹钟改为07:40起床，对你休息没有什么影响，多出来20分钟就可以看几页书了；如果你习惯晚上玩游戏玩到12点才睡，那么提前到11:40就不玩了，对你玩游戏也不会有什么影响；如果你坐地铁太挤了没法看书，那么可以听英语……总之一个人的一天中是有很多时间碎片可以挤出来的，关键就在于自己是否有这个“意识”。

通过这种方法，我既没有“头悬梁锥刺股”，也没有放弃自己的爱好“打魔兽玩CS”，工作也没有受到影响，同时知识和技能又提升了很多。也许开始1个月两个月你还感受不到这样做带来的明显好处，但坚持半年一年你就会发现有很大不同，坚持5年10年，也许你就会发现人生境界和成就都会不一样了。

**CSDN：你目前在UC担任软件主任工程师职位，同时兼顾团队leader，在技术团队管理方面，你有什么心得和体会可分享？**

**李运华：**虽然说是leader，但管理投入和技术投入大约是3/7，所以大部分还是在做技术工作。技术团队管理方面谈不上很有经验，简单谈一些我的体会吧。我是一个基层的小头头，因此以下分享都是针对基层的leader说的，总监和CEO没当过，不敢妄加评论 ：）

**首先，基层leader不能完全放弃技术，变成一个项目管理者或者任务管道。**对于技术人员来讲，多少都有点技术情结了，如果一个leader不懂技术或者不管技术，就开开会，转发邮件，安排一下任务，慢慢就会与其他成员脱离，技术水平就会逐渐下降，这样其实就是威信不断下降。当一个leader都无法和组员探讨技术问题、解决技术问题的时候，这个leader其实就是一个摆设了，我相信很多时候组员都会从心里鄙视这样的leader的。

**其次，基层leader的最重要任务是帮助组员提升。**基层leader受限于环境和资源的约束，很难做到帮组员实现宏伟的梦想，或者提供一个很大的平台给组员施展。这种情况下以什么来激励组员呢？我觉得最好的方式就是帮助组员不断提升，包括技术上的提升、做事能力的提升、沟通技巧的提升等，这些是能够做到也比较容易做到的。如果真的遇到各方面能力都比自己强的人，赶紧向上推荐，让他有更大的空间发展。

**第三，基层leader要身先士卒。**有一个管理的说法说的很好：10个人时你要冲在最前面，1000个人时你要走在中间，10000个人的时候你要走在最后面。基层leader就是要冲在最前面的那个，遇到难的设计问题，你要带领团队攻坚；遇到线上质量事故，你要能够带领团队快速解决；出了重大问题，你要能够用于承担责任；这样才能通过实际的行动言传身教，帮助团队成员提升，也能在这个过程中树立威信。

**CSDN：你在10年写了篇《如何与你的老大沟通？ 》的文章受到热捧，如今成为leader的你，能否再简要的和大家分享下你的经验？**

**李运华：**虽然我现在是一个leader，但因为我是技术出身而且一直在从事技术相关的工作，所以我和团队成员交流是没有问题的。但我在向上沟通和汇报的时候，用到的还是这些方法。比如说我做了一个系统解耦的工作，我不会只说“系统结构更优美”，而是说“解耦后版本开发速度大约可以提升1倍”这样的说法。

**面向对象（Object Oriented,OO）和编程语言**

**CSDN：程序员在应聘时，面试官问应聘者如何理解面向对象，你认为应聘者该怎么说？**

**李运华：**首先要了解面向对象出现的历史背景，然后要了解面向对象和面向过程的对比，面向对象的优点缺点等，最后才是面向对象的这些概念、原则、方法。如果谈到面向对象就只知道“抽象、类……”等这些概念，是知其然不知其所以然，在实际中也很难更好的应用面向对象。

这部分内容可以参考我的博客专栏《面向对象葵花宝典》。

**CSDN：技术大师们在对软件架构的研究中经历了很长时间的摸索，从面向过程到面向对象，从设计原则到设计模式，请问常用的设计模式和原则有哪些？**

**李运华：**毫无疑问，设计模式就是GoF提出的23个设计模式，其它模式基本都是这些模式的变体或者组合；设计原则就是SOLID原则。 设计模式和设计原则在我的博客专栏[《面向对象葵花宝典》](http://blog.csdn.net/column/details/oobaodian.html)都有深入和别出心裁的阐述，大家可以参考。（编者注：李老师在专栏中针对设计模式和原则进行了全面深入的阐述，由于内容较多，这里不再引用，还请移步[专栏](http://blog.csdn.net/column/details/oobaodian.html)阅读。）

**CSDN：软件工程相关的软件开发流程有很多，结合你这么多的软件开发经历，面向对象的开发技术流程应该是如何的？与面向过程的开发过程有什么不同？**

**李运华：**面向对象的开发技术流程我总结起来就是如下阶段：

1. 需求模型

通过和客户沟通，结合行业经验和知识，明确要求客户的需求。

在这个阶段中， 面向对象和面向过程的开发方式没有差异，如果有人认为分析方法不一样，那是因为没有明白需求分析具体的目标是什么，

需求分析的目标就是要准确把握客户需求，和具体的实现技术、开发流程都没有关系。

2. 领域模型

基于需求模型，提炼出领域相关的概念，为后面的面向对象设计打下基础。 面向过程的开发流程中对应的阶段是“功能提取”。

在这个阶段，面向对象和面向过程的开发过程差异就很明显了，面向对象的流程主要任务是从需求模型中提炼“领域类”，而面向过程的开发要求的是从需求模型中提炼“功能的处理过程”。

3. 设计模型

以领域模型为基础，综合面向对象的各种设计技巧，完成类的设计。 面向过程的开发流程对应的阶段是“功能分解”

在这个阶段，面向对象的开发过程要求完成从“领域类”演进到“软件类”，并逐步细化求精，完成最终的软件类设计，而面向过程的开发此时需要做的是将全流程的功能分解为子功能，并细化子功能的具体步骤。

4. 实现模型

以设计模型为基础，将设计模型翻译为具体的语言实现，完成编码。

在这个阶段，面向对象开发和面向过程开发都是使用具体的编程语言来实现各自的设计，流程上没有什么差别，但使用不同的语言来开发的话，效率上是有一定的差别的，

总体来说，面向对象编程语言在语言层面上效率要比面向过程的开发语言效率要高一些。但编程的效率是受很多因素影响的，单纯的语言因素对总体的效率影响并不明显。

总结一下，面向对象的开发流程其实是以“类”来串联整个开发流程，而面向过程的开发流程是以“功能”来串联整个开发流程。

**CSDN：当前很多资料都在讲述面向对象的优点，可是在软件开发或者大型软件项目中，面向对象却带来了巨大的类关系设计开销，你认为面向对象编程的重要性和适用范围在哪？以及面向对象编程的弊端是什么？**

**李运华：**面向对象思想作为应对软件复杂性日益攀升的解决之道提出来的，其重要性在于通过封装的方式将复杂性隔离开来。但面向对象不是“银弹”，不是所有场景都适合应用的，我觉得面向对象特别适合于“容易发生变化”的业务系统开发，例如互联网相关业务、游戏开发、ERP等行业，不太适合底层系统如操作系统、驱动、数据库等开发。

面向对象编程本身我觉得并不存在固有的弊端。比如说有的人认为面向对象增加了类关系设计开销，但这个开销和隔离复杂度带来的收益相比基本可以忽略不计，加上设计模式和设计原则的成熟，设计良好的面向对象方案并不会带来多大的开销；有的人说面向对象带来了更多的复杂性，但这个复杂性主要是语言本身的复杂性，一旦熟练掌握后就不会成为障碍，而且我觉得面向对象更加符合人类的思维特点，是一种“人的思维”，一旦转换后，人其实更加适应面向对象的思维。

**CSDN：正所谓树大招风，越是流行的东西越容易变成人们的众矢之的。网络上流传着“面向对象是一个骗局”、“面向对象已死”的言论，对此你怎么看？**

**李运华：**从面向对象的诞生历史背景来看，面向对象是70~80年代为了解决日益增加的软件复杂度而提出的，那么我们现在评估一下这个背景消失了么？ 软件复杂度下降了么？显然没有，原有的软件复杂度并没有消失或者减少，反而越来越增加了，现在的软件规模和80年代相比已经不可同日而语了。我们现在也没有找到一种新的思想或者技术来解决或者减少软件复杂度，所以说“面向对象已死”、“面向对象是一个骗局”都是很偏激的评价，没必要去争论。面向对象现在活得很好，也很流行，看看排名前10的编程语言就知道了。

当然，就像《人月神话》中说的“没有银弹”一样，面向对象也不是解决软件复杂度的银弹，因此不能期望用面向对象搞定一切问题。

如果真要说“面向对象已死”，那一定是另外一个更优秀的思想出来替代面向对象，但目前来看并没有这样的思想出现。

**CSDN：对于初学者，你觉得他们怎样才能知道有没有掌握面向对象编程的思想？**

**李运华：**掌握面向对象编程的技能是很容易的，会用Java写个类，写个继承，写个接口，基本上就可以说自己掌握了面向对象编程的技能，如果再加上设计模式和设计原则，基本上就可以说熟练掌握了面向对象编程。

但如果说真正掌握了面向对象编程的思想，我觉得一定是要看是否用面向对象的方式去分析和理解问题。举个很简单的例子：很多人都是用面向过程的方法去分析和理解问题，然后在实现的时候觉得这里可以封装为一个类，那里可以用一个接口，有的地方可以用继承，有的地方可以用抽象类等，这样其实就是用面向对象实现了面向过程。

**心中的大牛：“能够设计和实现一个完整的系统”**

**CSDN：你梦想有一天成为技术大牛，你对技术大牛的定义是什么？**

**李运华：**我心目中的大牛是这样的，对某一方面的技术很精通，对相关的技术都有一定的掌握和了解，能够清楚的理解和实现业务，对行业有一定的敏感度和理解，一句话总结一下就是：能够设计和实现一个完整的系统。

**CSDN：“梦想一定要有的，万一有一天实现了呢？”你期望能设计或实现一个伟大的产品，目前有什么好的想法吗？对于未来你还有什么样的规划和期许？**

**李运华：**实现一个伟大的产品更多时候是可遇不可求，所以现在暂时也没有什么打算和计划，只是努力在做各种技术和技能储备，当机遇来临的时候能够抓住机遇。目前在UC做的很开心，也在不断的提升自己，希望有一天能在UC实现这样的梦想 。

**CSDN：你是什么时候接触CSDN的？它对你学习和工作都带来哪些影响？同时，对CSDN有什么建议？**

**李运华：**基本上上大学的时候就接触CSDN了，后来开始在CSDN写博客。CSDN对我来说主要提供了一个交流和展现的平台，这里可以学到新的技术，掌握技术潮流，结识更多爱好技术的朋友，通过在CSDN写博客，既提升了自己的表达能力和写作能力，也扩大的自己的技术影响力。

谈到建议的话，CSDN能否提供一个类似微博的技术分享平台，因为有的技术点和经验可能比较简短，但很关键很重要，写成博客有点太重，论坛发的话不利于传播（要到指定版块才能看到），另外很多技术牛人有很多积累，但写博客可能没有那么多时间，写微博式的内容就很轻松了，所以如果CSDN能够提供这样的功能，一个是能够吸引更多有能力没时间的牛人来分享，二是能够让更多人能学到一些别人压箱底的功夫，这些功夫通过搜索引擎搜索基本上是搜不到的。