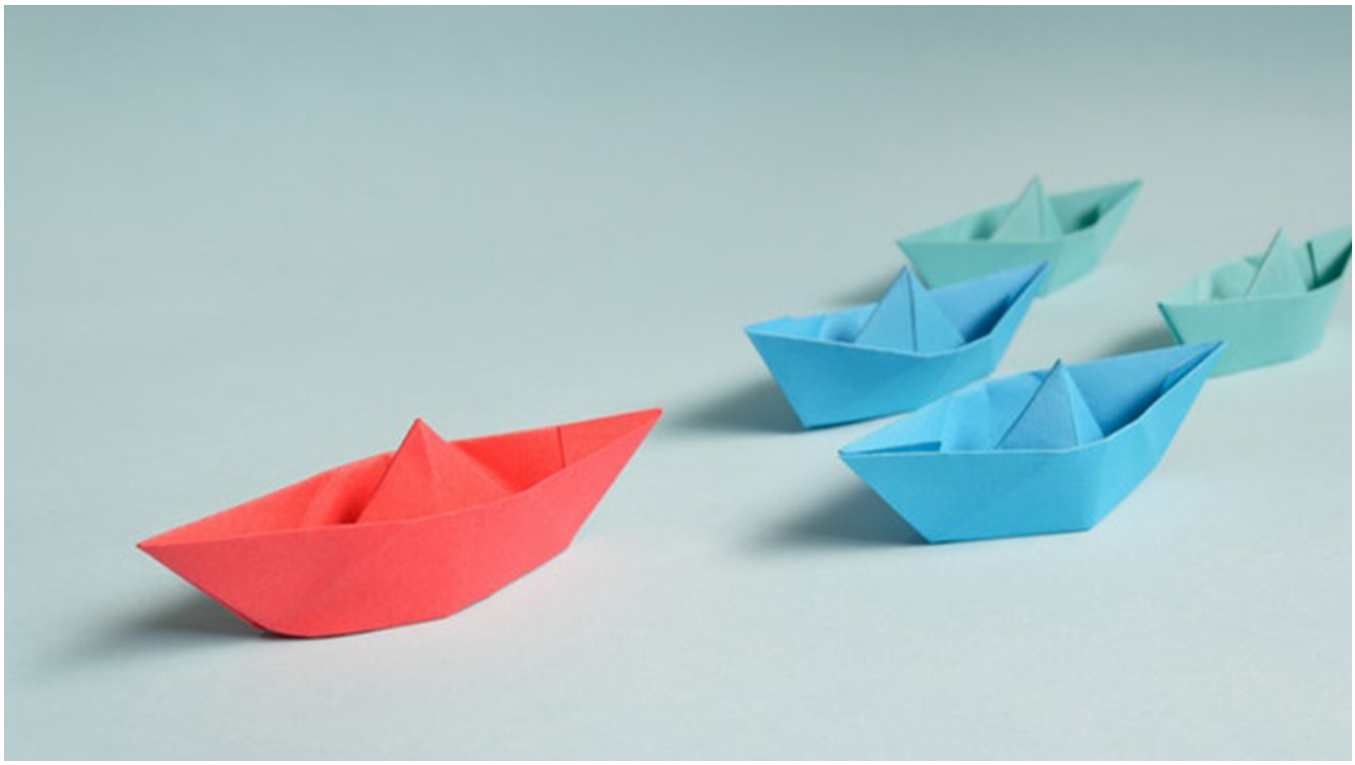
如何拥有技术领导力

2017-10-19 陈皓，杨爽



如何拥有技术领导力
朗读人：柴巍　16′46″ | 7.68M

通过上篇文章，相信你现在已经理解“什么才是技术领导力”。今天，我就要跟你聊聊，怎样才能拥有技术领导力。

—

第一，要吃透基础技术。基础技术是各种上层技术共同的基础。吃透基础技术是为了更好地理解程序的运行原理，并基于这些基础技术进化出更优化的产品。吃透基础技术，好处很多。

1. 万丈高楼平地起。一栋楼能盖多高，一座大桥能造多长，重要的是它们的地基。同样对于技术人员来说，基础知识越扎实，走得就会越远。
2. 计算机技术太多了，但是仔细分析你会发现，只是表现形式很多，而基础技术并不多。学好基础技术，能让你一通百通，更快地使用各种技术形式，从而可以更容易地跟上时代。
3. 很多分布式系统架构，以及高可用、高性能、高并发的解决方案基本都可以在基础技术上找到它们的身影。所以，对基础技术的学习能让你更好地掌握更高维度的技术。

那么，哪些才是基础技术呢？我在下面罗列了一些。老实说，这些技术你学起来可能会感到非常枯燥无味，但是，我还是鼓励你能够克服人性的弱点，努力啃完。

具体来说，就是分成两个部分：系统和编程。

编程部分

- C 语言：相对于很多其他高级语言来说，C 语言更接近底层。在具备跨平台能力的前提下，它可以比较容易地被人工翻译成相应的汇编代码。它的内存管理很直接，让程序员直接和内存地址打交道。

学习好 C 语言的好处是能掌握程序的运行情况，并能进行应用程序和操作系统编程（操作系统一般是汇编 +C 语言）。要学好 C 语言，可以阅读 C 语言的经典书籍《C 程序设计语言》第二版（K&R），并多写程序，再读一些优秀开源项目的源代码。

除了让你更为了解操作系统之外，学习 C 语言还能让你更清楚地知道程序是怎么精细控制底层资源的，比如内存管理、文件操作、网络通信……

这里需要说明的是，我们还是需要学习汇编语言的。因为如果你想更深入地了解计算机是怎么运作的，那么是需要了解汇编语言的。虽然我们几乎不再用汇编语言编程了，但是如果你需要写一些如 lock free 之类高并发的东西，那么了解汇编语言，能有助于你更好地理解和思考。

- 编程范式：各种编程语言都有它们各自的编程范式，用于解决各种问题。比如面向对象编程（C++、Java）、泛型编程（C++、Go、C#）、函数式编程（JavaScript、 Python、Lisp、Haskell、Erlang）等。

学习好编程范式，有助于培养编程的抽象思维，提高编程效率，提高程序的结构合理性、可读性和可维护性，降低代码的冗余度，提高代码的运行效率。要学习编程范式，可以多了解各种程序设计语言的功能特性。

- 算法和数据结构：算法（及其相应的数据结构）是程序设计的有力支撑。适当地应用算法，可以有效地抽象问题，提高程序的合理性和执行效率。算法是编程中最最重要的东西，也是计算机科学中最重要的东西。

任何有技术含量的软件中一定有高级的算法和数据结构。比如 epoll 中使用了红黑树，数据库索引使用了 B+ 树……而就算是你的业务系统中，也一定使用各种排序、过滤和查找算法。学习算法不仅是为了写出运转更为高效的代码，而且更是为了能够写出可以覆盖更多场景的正确代码。

系统部分

- 计算机系统原理：CPU 的体系结构（指令集 [CISC/RISC]、分支预测、缓存结构、总线、DMA、中断、陷阱、多任务、虚拟内存、虚拟化等），内存的原理与性能特点（SRAM、

DRAM、DDR-SDRAM 等），磁盘的原理（机械硬盘 [盘面、磁头臂、磁头、启停区、寻道等]、固态硬盘 [页映射、块的合并与回收算法、TRIM 指令等]），GPU 的原理等。

学习计算机系统原理的价值在于，除了能够了解计算机的原理之外，你还能举一反三地反推出高维度的分布式架构和高并发高可用的架构设计。

比如：虚拟化内存就和今天云计算中的虚拟化的原理是相通的，计算机总线和分布式架构中的 ESB 也有相通之处，计算机指令调度、并发控制可以让你更好地理解并发编程和做程序性能调优……这里，推荐书籍《深入理解计算机系统》（Randal E. Bryant）。

- 操作系统原理和基础：进程、进程管理、线程、线程调度、多核的缓存一致性、信号量、物理内存管理、虚拟内存管理、内存分配、文件系统、磁盘管理等。

学习操作系统的价值在于理解程序是怎样被管理的，操作系统对应用程序提供了怎样的支持，抽象出怎样的编程接口（比如 POSIX/Win32 API），性能特性如何（比如控制合理的上下文切换次数），怎样进行进程间通信（如管道、套接字、内存映射等），以便让不同的软件配合一起运行等。

要学习操作系统知识，一是仔细观察和探索当前使用的操作系统，二是阅读讲操作系统原理的图书，三是阅读 API 文档（如 man pages 和 MSDN Library），并编写调用操作系统功能的程序。这里推荐三本书《UNIX 环境高级编程》、《UNIX 网络编程》和《Windows 核心编程》。

学习操作系统基础原理的好处是，这是所有程序运行的物理世界，无论上层是像 C/C++ 这样编译成机器码的语言，还是像 Java 这样有 JVM 做中间层的语言，还是像 Python/PHP/Perl/Node.js 这样直接在运行时解释的语言，其在底层都逃离不了操作系统这个物理世界的“物理定律”。

所以，了解操作系统的原理，可以让你更能本质地理解各种语言或是技术的底层原理。一眼看透本质将让你更容易地掌握和使用高阶技术。

- 网络基础：计算机网络是现代计算机不可或缺的一部分。需要了解基本的网络层次结构（ISO/OSI 模型、TCP/IP 协议栈），包括物理层、数据链路层（包含错误重发机制）、网络层（包含路由机制）、传输层（包含连接保持机制）、会话层、表示层、应用层（在 TCP/IP 协议栈里，这三层可以并为一层）。

比如，底层的 ARP 协议、中间的 TCP/UDP 协议，以及高层的 HTTP 协议。这里推荐书籍《TCP/IP 详解》，学习这些基础的网络协议，可以为我们的高维分布式架构中的一些技术问题提供很多的技术方案。比如：TCP 的滑动窗口限流，完全可以用于分布式服务中的限流方案。

- 数据库原理：数据库管理系统是管理数据库的利器。通常操作系统提供文件系统来管理文件数据，而文件比较适合保存连续的信息，如一篇文章、一个图片等。但有时需要保存一个名字等较短的信息。如果单个文件只保存名字这样的几个字节的信息的话，会浪费大量的磁盘空间，而且无法方便地查询（除非使用索引服务）。

但数据库则更适合保存这种短的数据，而且可以方便地按字段进行查询。现代流行的数据库管理系统有两大类：SQL（基于 B+ 树，强一致性）和 NoSQL（较弱的一致性，较高的存取效率，基于哈希表或其他技术）。

学习了数据库原理之后更能了解数据库访问性能调优的要点，以及保证并发情况下数据操作原子性的方法。要学习数据库，可以阅读各类数据库图书，并多做数据库操作以及数据库编程，多观察分析数据库在运行时的性能。

- 分布式技术架构：数据库和应用程序服务器在应对互联网上数以亿计的访问量的时候，需要能进行横向扩展，才能提供足够的性能。为了做到这一点，要学习分布式技术架构，包括负载均衡、DNS 解析、多子域名、无状态应用层、缓存层、数据库分片、容错和恢复机制、Paxos、Map/Reduce 操作、分布式 SQL 数据库一致性（以 Google Cloud Spanner 为代表）等知识点。

学习分布式技术架构的有效途径是参与到分布式项目的开发中去，并阅读相关论文。

注意，上面这些基础知识通常不是可以速成的。虽然说，你可以在一两年内看完相关的书籍或论文，但是，我想说的是，这些基础技术是需要你用一生的时间来学习的，因为基础上的技术和知识，会随着阅历和经验的增加而有不同的感悟。

二

第二，提高学习能力。所谓学习能力，就是能够很快地学习新技术，又能在关键技术上深入的能力。只有在掌握了上述的基础原理之上，你才能拥有好的学习能力。

下面是让你提升学习能力的一些做法。

- 学习的信息源。信息源很重要，有好的信息源就可以更快速地获取有价值的信息，极大提升学习效率。常见的信息源有：Google 等搜索引擎，Stack Overflow、Quora 等社区，图书，API 文档，论文和博客等。

这么说吧，如果今天使用中文搜索就可以满足你的知识需求，那么你就远远落后于这个时代了。如果用英文搜索才能找到你想要的知识，那么你才能算得上跟上这个时代。而如果你连用英文搜索都找不到，只能到社区里去找作者或是和大众交流，那么可以说你已真正和时代靠近了。

- 与高手交流。程序员可以通过技术社区以及参加技术会议与高手交流，也可以通过参加开源项目来和高手切磋。常闻“听君一席话，胜读十年书”便是如此。与高手交流对程序员的学

习和成长很有益处，不仅有助于了解热门的技术方向及关键的技术点，更可以通过观察和学习高手的技术思维及解决问题的方式，提高自己的技术前瞻性和技术决策力。

我在 Amazon 的时候，就有人和我说，多和美国的 Principle SDE 以上的工程师交流，无论交流什么，你都会有收获的。其实，这里说的就是，学习这些牛人的思维方式和看问题的角度，这会让你有质的提高。

- 举一反三的思考。比如，了解了操作系统的缓存和网页缓存以后，思考其相同点和不同点。了解了 C++ 语言的面向对象特性以后，思考 Java 面向对象的相同点和不同点。遇到故障的时候，举一反三，把同类问题一次性地处理掉。

- 不怕困难的态度。遇到难点，有时不花一番力气，是不可能突破的。此时如果没有不怕困难的态度，就容易打退堂鼓。但如果能坚持住，多思考，多下功夫，往往就能找到出路。绝大多数人是害怕困难的，所以，如果你能够不怕困难，并可以找到解决困难的方法和路径，时间一长，你就能拥有别人所不能拥有的能力。

- 开放的心态。实现一个目的通常有多种办法。带有开放的心态，不拘泥于一个平台、一种语言，往往能带来更多思考，也能得到更好的结果。而且，能在不同的方法和方案间做比较，比较它们的优缺点，那么你会知道在什么样的场景下用什么样的方案，你就会比一般人能够有更全面和更完整的思路。

三

第三，坚持做正确的事。做正确的事，比用正确的方式做事更重要，因为这样才始终会向目的地靠拢。

- 提高效率的事。学习和掌握良好的时间管理方式，管理好自己的时间，能显著提高自己的效率。

- 自动化的事。程序员要充分利用自己的职业特质，当看见有可以自动化的步骤时，编写程序来自动化操作，可以显著提高效率。

- 掌握前沿技术的事。掌握前沿的技术，有利于拓展自己的眼界，也有利于找到更好的工作。需要注意的是，有些技术虽然当下很火，但未必前沿，而是因为它比较易学易用，或者性价比高。由于学习一门技术需要花费不少时间，应该选择自己最感兴趣的，有的放矢地去学习。

- 知识密集型的事。知识密集型是相对于劳动密集型来说的。基本上，劳动密集型的事都能通过程序和机器来完成，而知识密集型的事却仍需要人来完成，所以人的价值就在此时体现了。虽然现在人工智能似乎能做一些知识密集型的事（包括下围棋的 AlphaGo），但是在开放领域中相对于人的智能来说还是相去甚远。掌握了领域知识的人的价值依然很高。

- 技术驱动的事。不仅是指用程序驱动的事，而且包括一切技术改变生活的事。比如自动驾驶、火星登陆等。就算自己一时用不着，也要了解这些，以便将来这些技术来临时能适应这些新技术。

第四，高标准要求自己。只有不断地提高标准 ，你才可能越走越高，所以，要以高标准要求自己，不断地反思、总结和审视自己，才能够提升自己。

- Google 的自我评分卡。Google 的评分卡是在面试 Google 时，要求应聘人对自己的技能做出评估的工具，可以看出应聘人在各个领域的技术水平。我们可以参考 Google 的这个评分卡来给自己做评估，并通过它来不断地提高对自己的要求。（该评分卡见后附录）。

- 敏锐的技术嗅觉。这是一个相对综合的能力，你需要充分利用信息源，GET 到新的技术动态，并通过参与技术社区的讨论，丰富自己了解技术的角度。思考一下是否是自己感兴趣的，能解决哪些实际问题，以及其背后的原因，新技术也好，旧有技术的重大版本变化也罢。

- 强调实践，学以致用。学习知识，一定要实际用一用，可以是工作中的项目，也可以是自己的项目，不仅有利于吸收理解，更有利于深入到技术的本质。并可以与现有技术对比一下，同样的问题，用新技术解决有什么不同，带来了哪些优势，还有哪些有待改进的地方。

- Lead by Example。永远在编程。不写代码，你就对技术细节不敏感，你无法做出可以实践的技术决定和方案。

不要小看这些方法和习惯，坚持下来很有益处。谁说下一个改进方向或者重大修改建议，不可以是你给出的呢，尤其是在一些开源项目中。何为领导力，能力体现之一不就是指明技术未来的发展方向吗？

吃透基础技术、提高学习能力、坚持做正确的事、高标准要求自己，不仅会让你全面提升技术技能，还能很好地锻炼自己的技术思维，培养技术前瞻性和决策力，进而形成技术领导力。

然而，仅有技术还不够。作为一名合格的技术领导者，还需要有解决问题的各种软技能。比如，良好的沟通能力、组织能力、驱动力、团队协作能力，等等。《技术领导之路》、《卓有成效的管理者》等多本经典图书中均有细致的讲解，这里不展开讲述。

附：

Google 评分卡

0 - you are unfamiliar with the subject area.

1 - you can read / understand the most fundamental aspects of the subject area.

2 - ability to implement small changes, understand basic principles and able to figure out additional details with minimal help.

3 - basic proficiency in a subject area without relying on help.

4 - you are comfortable with the subject area and all routine work on it: For software areas - ability to develop medium programs using all basic language features w/o book, awareness of more esoteric features (with book).

For systems areas - understanding of many fundamentals of networking and systems administration, ability to run a small network of systems including recovery, debugging and nontrivial troubleshooting that relies on the knowledge of internals.

5 - an even lower degree of reliance on reference materials. Deeper skills in a field or specific technology in the subject area.

6 - ability to develop large programs and systems from scratch. Understanding of low level details and internals. Ability to design / deploy most large, distributed systems from scratch.

7 - you understand and make use of most lesser known language features, technologies, and associated internals. Ability to automate significant amounts of systems administration.

8 - deep understanding of corner cases, esoteric features, protocols and systems including "theory of operation". Demonstrated ability to design, deploy and own very critical or large infrastructure, build accompanying automation.

9 - could have written the book about the subject area but didn't; works with standards committees on defining new standards and methodologies.

10 - wrote the book on the subject area (there actually has to be a book). Recognized industry expert in the field, might have invented it.

Subject Areas:

- TCP/IP Networking (OSI stack, DNS etc)
- Unix/Linux internals
- Unix/Linux Systems administration
- Algorithms and Data Structures
- C
- C++

- Python
- Java
- Perl
- Go
- Shell Scripting (sh, Bash, ksh, csh)
- SQL and/or Database Admin
- Scripting language of your choice (not already mentioned) _
- People Management
- Project Management

精选留言

卖桃者 35

学习好编程范式，有助于培养编程的抽象思维，提高编程效率，提高程序的结构合理性、可读性和可维护性，降低代码的冗余度，提高代码的运行效率。要学习编程范式，可以多了解各种程序设计语言的功能特性。

这段写的真好，好的程序员没有只掌握一门编程语言的

2017-10-19

阴明 13

很多人在阶段性的技术领先下获得了一定的管理权限后，很快就会开始转行管理。而国内的激励机制里，仍然对业务进展、财务进展的激励优于技术进展。
这使得很多技术高手后期以管理为重，渐渐失去了技术的领导力。

2017-10-23

陈俊 9

耗哥这个专栏是真好，这篇文章尤其好！既有系统地阐述和分析，也有具体的操作建议。真是让我有醍醐灌顶的感觉。结合这几年的工作经验，基本上我认同耗哥的指导，感谢分享！

2017-10-24

Lincoln 4

做了一个思维导图梳理了一下。在动手敲这些文字的过程中，感受到了耗子叔的用心。都是实实在在的话。也发现自己以后要走的路还有很远很远。

2017-11-05

笨笨熊 4

一贯的好文，谢谢耗子叔叔分享！

2017-10-19

bubble 2

谢谢耗叔的建议，从现在开始努力打基础，万丈高楼平地起，这才是成功的捷径。

2017-10-23

macworks 2

即使是CSAPP，很多地方也只是点到为止，计算机这门手艺真的没法速成。

2017-10-22

ydp 1

指明了努力方向，虽然实现较难，感谢分享！

2018-05-10

作者回复

参看AWS的SWF

2018-05-10

ﾉ梦深处丶 1

你好，请问对提升英文水平的方式有何建议

2018-03-19

睿睿睿睿睿睿、 1

看了这篇文章顿时觉得该朝什么方向去学习发展 而不是一直再追各种新框架

2017-10-30

宋桓公 0

我这些年机缘巧合学习路线是c，单片机，c# Verilog，FPGA，c，arm，zynq，c++，ActiveX，nodejs Python。

感觉搞底层的时候圈子不大，很快认识很多牛人。
渐渐偏软了之后，发现牛人太多。

2018-06-10

鱼的记忆 0

不错 收藏了

2018-06-10

夏洛克的救赎 0

这么说吧，如果今天使用中文搜索就可以满足你的知识需求，那么你就远远落后于这个时代了。如果用英文搜索才能找到你想要的知识，那么你才能算得上跟上这个时代。而如果你连用英文搜索都找不到，只能到社区里去找作者或是和大众交流，那么可以说你已真正和时代靠近了。

涨姿势了 认知升级

2018-06-08

夏洛克的救赎 0

学习操作系统基础原理的好处是，这是所有程序运行的物理世界，无论上层是像 C/C++ 这样编译成机器码的语言，还是像 Java 这样有 JVM 做中间层的语言，还是像 Python/PHP/Perl/Node.js 这样直接在运行时解释的语言，其在底层都逃离不了操作系统这个物理世界的“物理定律”。

可是操作系统不也是基于硬件的吗

2018-06-08

夏洛克的救赎 0

了解编程范式是不是得学习编译原理

2018-06-08

作者回复

不一定。大多数情况下不用

2018-06-11

MarksGui 0

好文！指明方向，感谢皓哥！

2018-06-03

细雨平湖 0

与我计划积累的路线不谋而合。除了以上技术外，还建议熟悉数据仓库、人工智能、机器学习(含深度学习、强化学习)、概率编程、自然语言处理、机器视觉。

2018-05-30

刘二 0

快速学习能力和时间管理真的是相辅相成的，如何有效的利用好时间也蛮重要的

2018-05-15

魔术师Carvendy 0

看到了森林，也看到了树木，有收获。赞👍。

2018-04-22

郑一一 0

我有个疑惑是，很多人说学习最高效的方式带着问题去学习，可是很多东西我们在实际生活中真的碰不到，那应该如何权衡呢？

2018-04-20