

程序员练级攻略（2018）：Linux系统、内存和网络

2018-06-26 陈皓





程序员练级攻略（2018）：Linux系统、内存和网络

陈皓

- 00:00 / 19:48

这一篇章，是本文最长的一篇，其中包括了如下的内容。

- 系统底层相关。主要是以Linux系统为主，其中有大量的文章可以让你学习到Linux内核，以及内存、网络、异步I/O模型、Lock-free的无锁编程，还有其它和系统底层相关的东西。注意，系统底层要是深下去是可以完全不见底的。而且内存方面的知识也是比较多的，所以，这里还是主要给出一些非常有价值的基础性的知识和技术。学好这些东西，你会对系统有很深的理解，而且可以把这些知识反哺到架构设计上来。
- 数据库相关。数据库方面主要是MySQL和各种开源NoSQL的一些相关的有价值的文章和导读，主要是让你对这些数据库的内在有一定的了解，但又不会太深。真正的深入是需要扎入到源代码中的。需要说明的是，这块技术不是我的长项，但又是每个架构师需要知道的，所以，我在这里给的学习资源可能会比较浅，这点还希望你来补充和指正。
- 分布式架构。这一部分是最长最多的。其中有架构入门、分布式理论中各种非常有价值的经典论文，然后是一些分布式工程设计方面的文章，其中包括设计模式和工程应用，最后还有各大公司的架构供参考。
- 微服务。有了分布式架构理论和工程的基础，接下来是对微服务的学习。在这部分内容中，我会罗列几个介绍微服务架构非常系统的文章，然后比较一下微服务和SOA的差别，最后则是一些工程实践和最佳实践。
- 容器化和自动化运维。在容器化和自动化运维中，主要是学习Docker和Kubernetes这两个自动化运维的杀手型技术。而不是Salt、Puppet、Chef和Ansible这样比较传统的工具。原因很简单，因为自动化部署根本不够，还需要对环境和运行时的管理和运维才够，而只有Docker和Kubernetes才是未来。所以，这里重点让你学习这两个技术，其中有很多文章需要一些系统底层的知识。
- 机器学习和人工智能。机器学习和人工智能，也不是我的长项，我也只是一个入门者。这里，我主要给了一些基础性的知识，其中包括基本原理、图书、课程、文章和相关的算法。你顺着我画的这路走，不能说能成为一个人工智能专家，但成为一个机器学习的高级工程师甚至准专家还是可能的。
- 前端开发。这里的前端主要是H5的前端了，这一节会带你学习一下前端开发所需要知道的基础知识，尤其是对前端开发语言JavaScript的学习，我花费了相当的篇幅列出了很多很经典的学习资料，必定会让你成为一个JavaScript高手。然后你还需要了解浏览器是怎样工作的，还有相关的网络协议和一些性能优化的技巧。最后则是JavaScript框架的学习，这里我只给了React.js和Vue.js，并通过React.js带出来函数式编程的学习。我虽然不是一个前端程序员，但是，我相信我这个后端程序员给出来的这组前端开发的学习资料和路径会比前端程序员更靠谱一些。
- 信息源。最后，则是一些信息源，其中包括各大公司的技术Blog，还有相关的论文集散地。

另外，这里需要说明几点。

- 我假设你在前面已经打下了非常扎实的基础，但是要成为一个高手，基础知识只是一个地基，你还需要很多更为具体的技术。对我来说，就是看各种各样的文章、手册、论文、分享……其实，学习到一定程度，就是要从书本中走出去，到社区和大家一起学习，而且还需要自己找食吃了。所以，对于这里面的文章，有很多都是在罗列各种文章和资源，只是为你梳理信息源，而不是喂你吃饭。
- 老实说，我已经为你梳理并过滤掉了很多的信息，这里只留下了30%我觉得最经济也最有价值的信息。虽然对于不同定位和不同需求的人还可以再对这些信息进行删减，但是觉得我这么一做就会对其它人不公平了。所以，这也是我觉得最小数量集的信息和资源吧。你也可以把我这里的东西当成一个索引来对待。
- 这些内容，不能说是隔离开来的，应该说是相辅相成的。也没什么顺序，可以各取所需。虽然看上去内容很多，但你也别害怕，真的不用害怕，你会越快越快，越实践越有感觉，也越有效率。在一开始可能会很慢，但是坚持住，积累一段时间后会越来越快的。而且，我要告诉你，绝大多数人是坚持不下来的。只要你能坚持下来，我保证，你一定会成为各个大公司的抢手货，这点你一定要相信我。你不需要特别努力，只需要日进一步，3-5年后，你就会发现，绝大多数人都在你身后很远的地方了。

今天分享的内容为系统底层知识中的Linux系统、内存和网络等方面的相关知识及推荐的学习资料。

Linux系统相关

学习Linux操作系统的原理是通向系统工程师的必经之路。我觉得，Unix/Linux操作系统里的东西并不难学。你千万不要一下子扎到源代码里去，那样没用——你还是要在上层先通过读一些不错的文档来学习。下面我罗列了一些很不错的站点，其中有很多内容供你去钻研和探索。

我在这里默认你前面已经读过并读懂了我推荐的那些和Linux/Linux相关的图书了。所以，我相信你对Unix/Linux下的编程已经是有一些基础了，因此，你继续深挖Linux下的这些知识应该也不是很难的事了。

- [Red Hat Enterprise Linux文档](#)。Red Hat Enterprise Linux（RHEL）是老牌Linux厂商Red Hat出品的面向商业的Linux发行版。Red Hat网站上的这个文档中有很多很有价值的内容，值得一看。
- [Linux Insides](#)，GitHub上的一个开源电子书，其中讲述了Linux内核是怎样启动、初始化以及进行管理的。
- [LWN's kernel page](#)，上面有很多非常不错的文章来解释Linux内核的一些东西。
- [Learn Linux Kernel from Android Perspective](#)，从Android的角度来学习Linux内核，这个站点上的Blog相对于前面的比较简单易懂一些。
- [Linux Kernel Doc](#)，Linux的内核文档也可以浏览一下。
- [Kernel Planet](#)，Linux内核开发者的Blog，有很多很不错的文章和想法。
- [Linux Performance and Tuning Guidelines](#)，这是IBM出的红皮书，虽然有点老了，但还是非常值得一读的。
- [TLK: The Linux Kernel](#)，这是一本相对比较老的书了，Linux内核版本为2.0.33，但了解一下前人的思路，也是很有帮助的。
- [Linux Performance](#)，这个网站上提供了和Linux系统性能相关的各种工具和文章收集，非常不错。
- [Optimizing web servers for high throughput and low latency](#)，这是一篇非常底层的系统调优的文章，来自DropBox，从中你可以学到很多底层的性能调优的经验和知识。

内存相关

计算机内存管理是每一个底层程序员需要了解的非常重要的事儿。当然，这里我们重点还是Linux操作系统相关的内存管理上的知识。

首先，LWN.net上有一系列的“[What every programmer should know about memory](#)”文章你需要读一下。当然，你可以直接访问一个完整的[PDF文档](#)。下面是这个系列文章的网页版列表。读完这个列表的内容，你基本上就对内存有了一个比较好的知识体系了。

- [Part 1: Introduction](#)，中译版为“[每个程序员都应该了解的内存知识【第一部分】](#)。”
- [Part 2: CPU caches](#)
- [Part 3 \(Virtual memory\)](#)
- [Part 4 \(NUMA systems\)](#)
- [Part 5 \(What programmers can do - cache optimization\)](#)
- [Part 6 \(What programmers can do - multi-threaded optimizations\)](#)
- [Part 7 \(Memory performance tools\)](#)
- [Part 8 \(Future technologies\)](#)
- [Part 9 \(Appendices and bibliography\)](#)

然后是几篇和内存相关的论文。下面这三篇论文是我个人觉得能对你非常有帮助的文章，尤其是你要做一些程序的性能优化方面。

- [Memory Barriers: a Hardware View for Software Hackers](#)。内存的读写屏障是线程并发访问共享的内存数据时，从程序本身、编译器到CPU都必须遵循的一个规范。有了这个规范，才能保证访问共享的内存数据时，一个线程对该数据的更新能被另一个线程以正确的顺序感知到。在SMP（对称多处理）这种类型的多处理器系统（包括多核系统）上，这种读写屏障还包含了复杂的缓存一致性策略。这篇文章做了详细解释。
- [A Tutorial Introduction to the ARM and POWER Relaxed Memory Models](#)，对ARM和POWER的宽松内存模型的一个教程式的简介。本篇文章的焦点是ARM和POWER体系结构下多处理器系统内存并发访问一致性的设计思路和使用方法。与支持较强的TSO模型的x86体系结构不同，ARM和POWER这两种体系结构出于对功耗和性能的考虑，使用了一种更为宽松的内存模型。本文详细讨论了ARM和POWER的模型。
- [x86-TSO: A Rigorous and Usable Programmer's Model for x86 Multiprocessors](#)，介绍x86的多处理器内存并发访问的一致性模型TSO。

接下来是开发者最关心的内存管理方面的lib库。通常来说，我们有三种内存分配管理模块。就目前而言，BSD的jemalloc有很大的影响力。后面我们可以看到不同公司的实践性文章。

- [ptmalloc](#) 是glibc的内存分配管理。
- [tcmalloc](#) 是Google的内存分配管理模块，全称是Thread-Caching malloc，基本上来说比glibc的ptmalloc快两倍以上。
- [jemalloc](#) 是BSD提供的内存分配管理。其论文为 [A Scalable Concurrent malloc\(3\) Implementation for FreeBSD](#)，这是一个可以并行处理的内存分配管理器。

关于 C 的这些内存分配器，你可以参看Wikipedia的“[C Dynamic Memory Allocation](#)”这个词条。

下面是几篇不错的文章，让你感觉一下上面那三种内存分配器的一些比较和工程实践。

- [ptmalloc、tcmalloc和jemalloc内存分配策略研究](#)
- [内存优化总结：ptmalloc、tcmalloc和jemalloc](#)
- [Scalable memory allocation using jemalloc](#)
- [Decreasing RAM Usage by 40% Using jemalloc with Python & Celery](#)

计算机网络

网络学习

首先，推荐一本书——《[计算机网络（第五版）](#)》，这本“计算机网络”和前面推荐的那本计算机网络不一样，前面那本偏扫盲，这本中有很多细节。这本书是国内使用最广泛、最权威的计算机网络经典教材。全书按照网络协议模型自下而上（物理层、数据链路层、介质访问控制层、网络层、传输层和应用层）有系统地介绍了计算机网络的基本原理，并结合Internet给出了大量的协议实例。

这本书还与时俱进地引入了最新的网络技术，包括无线网络、3G蜂窝网络、RFID与传感器网络、内容分发与P2P网络、流媒体传输与IP语音，以及延迟容忍网络等。另外，本书针对当前网络应用中日益突出的安全问题，用了一整章的篇幅对计算机网络的安全性进行了深入讨论，而且把相关内容与最新网络技术结合起来阐述。这本书读起来并不枯燥，因为其占有很多小故事和小段子。

然后，有两个网上的教程和讲义也可以让人入门。

- 渥太华大学的一个课程讲义你也可以一看 [Computer Network Design](#)。
- GeeksforGeeks 上也有一个简单的 [Computer Network Tutorials](#)。

网络调优

接下来，你可能需要一些非常实用的可以操作的技术，下面的几篇文章相信可以帮助你。

- 《Linux的高级路由和流量控制 HowTo》（[Linux Advanced Routing & Traffic Control HOWTO](#)），这是一个非常容易上手的关于 iproute2、流量整形和一点 netfilter 的指南。
- 关于网络调优，你可以看一下这个文档 [Red Hat Enterprise Linux Network Performance Tuning Guide](#)。
- 还有一些网络工具能够帮上你的大忙，这里有一个网络工具的Awesome列表 [Awesome Pcap Tools](#)，其中罗列了各种网络工具，能够让你更从容地调试网络相关的程序。
- [Making Linux TCP Fast](#)，一篇非常不错的TCP调优的论文。
- 下面是在PackageCloud上的两篇关于Linux网络栈相关的底层文章，非常值得一读。
 - [Monitoring and Tuning the Linux Networking Stack: Receiving Data](#)
 - [Monitoring and Tuning the Linux Networking Stack: Sending Data](#)

网络协议

接下来，想要学习网络协议最好的方式就是学习通讯相关的RFC。所以，在这里我会推荐一系列值得读的RFC给你。读RFC有几个好处，一方面可以学习技术，另一方面，你可以通过RFC学习到一个好的技术文档是怎么写的，还能看到各种解决问题的方案和思路。

对于第2层链路层，你可能需要了解一下ARP：

- [RFC 826 - An Ethernet Address Resolution Protocol](#)

以及Tunnel相关的协议：

- [RFC 1853 - IP in IP Tunneling](#)
- [RFC 2784 - Generic Routing Encapsulation \(GRE\)](#)
- [RFC 2661 - Layer Two Tunneling Protocol "L2TP"](#)
- [RFC 2637 - Point-to-Point Tunneling Protocol \(PPTP\)](#)

对于第4层，你最需要了解的是TCP/IP了。和TCP相关的RFC相当多，这里给一系列经典的RFC。这些RFC我都引用在了我在CoolShell上的《[TCP的那些事儿（上）](#)》和《[TCP的那些事儿（下）](#)》两篇文章中。如果你看不懂RFC，你也可以去看我上述的文章。

- [RFC 793 - Transmission Control Protocol](#) - 最初的TCP标准定义，但不包括TCP相关细节。
- [RFC 813 - Window and Acknowledgement Strategy in TCP](#) - TCP窗口与确认策略，并讨论了在使用该机制时可能遇到的问题及解决方法。
- [RFC 879 - The TCP Maximum Segment Size and Related Topics](#) - 讨论MSS参数对控制TCP分组大小的重要性，以及该参数与IP分段大小的关系等。
- [RFC 896 - Congestion Control in IP/TCP Internetworks](#) - 讨论拥塞问题和TCP如何控制拥塞。
- [RFC 2581 - TCP Congestion Control](#) - 描述用于拥塞控制的四种机制：慢启动、拥塞防御、快重传和快恢复。后面这个RFC被 [RFC 5681](#) 所更新。还有 [RFC 6582 - The NewReno Modification to TCP's Fast Recovery Algorithm](#) 中一个改进的快速恢复算法。
- [RFC 2018 - TCP Selective Acknowledgment Options](#) - TCP的选择确认。
- [RFC 2883 - An Extension to the Selective Acknowledgement \(SACK\) Option for TCP](#) - 对于RFC 2018的改进。
- [RFC 2988 - Computing TCP's Retransmission Timer](#) - 讨论与TCP重传计时器设置相关的话题，重传计时器控制报文在重传前应等待多长时间。也就是经典的TCP Karn/Partridge重传算法。
- [RFC 6298 - Computing TCP's Retransmission Timer](#) - TCP Jacobson/Karels Algorithm重传算法。

我个人觉得TCP最牛的不是不丢包，而是拥塞控制。对此，如果你感兴趣，可以读一下经典论文《[Congestion Avoidance and Control](#)》。

关于Linux下的TCP参数，你需要仔仔细细地读一下[TCP的man page](#)。

对于第7层协议，HTTP协议是重点要学习的。

首先推荐的是《[HTTP权威指南](#)》，这本书有点厚，可以当参考书来看。这本书中没有提到HTTP/2的事，但是可以让你了解到HTTP协议的绝大多数特性。

HTTP 1.1的原始RFC是1999年6月的 [RFC 2616](#)，但其在2014后很快被下面这些RFC给取代了。

- [RFC 7230 - Hypertext Transfer Protocol \(HTTP/1.1\): Message Syntax and Routing](#)
- [RFC 7231 - Hypertext Transfer Protocol \(HTTP/1.1\): Semantics and Content](#)
- [RFC 7232 - Hypertext Transfer Protocol \(HTTP/1.1\): Conditional Requests](#)
- [RFC 7233 - Hypertext Transfer Protocol \(HTTP/1.1\): Range Requests](#)
- [RFC 7234 - Hypertext Transfer Protocol \(HTTP/1.1\): Caching](#)
- [RFC 7235 - Hypertext Transfer Protocol \(HTTP/1.1\): Authentication](#)

关于[HTTP/2](#)，这是HTTP的一个比较新的协议，它于2015年被批准通过，现在基本上所有的主流浏览器都默认启用这个协议。所以，你有必要学习一下这个协议。下面是相关的学习资源。

- [Gitbook - HTTP/2详解](#)
- [http2 explained](#)（中译版）
- [HTTP/2 for a Faster Web](#)
- [Nginx HTTP/2 白皮书](#)

- HTTP/2 的两个RFC：
 - [RFC 7540 - Hypertext Transfer Protocol Version 2 \(HTTP/2\)](#)，HTTP/2的协议本身

- [RFC 7541 - HPACK: Header Compression for HTTP/2](#) , [HTTP/2的压缩算法](#)

最后，你可以上Wikipedia的 [Internet Protocol Suite](#) 上看看，这是一个很不错的网络协议的词条汇集地。顺着这些协议，你可以找到很多有用的东西。

小结

好了，总结一下今天的内容。这是程序员练级攻略2018版第五篇章——高手成长篇的第一篇文章。前面的内容先介绍了一些这一系列内容的总体构成，及每一部分的学习重点。后面是这一篇章第一个主题系统底层知识中的部分内容，即Linux系统、内存和计算机网络，并给出了相应的学习资料。

我认为，学习到一定程度，就是要从书本中走出去，到社区里和大家一起学习，而且还需要自己找食吃了。所以，这篇文章中，我罗列了各种文章和资源，并给出了简短的推荐语言，就是在你梳理信息源，而不是喂你吃饭。我更希望看到你自趋势地成长。

下篇文章中，我们分享的内容为系统底层知识中的异步I/O模型、Lock-Free编程以及其他一些相关的知识点和学习资源。敬请期待。

下面是《程序员练级攻略（2018）》系列文章的目录（持续更新中）。

- [开篇词](#)
- 入门篇
 - [零基础启蒙](#)
 - [正式入门](#)
- 修养篇
 - [程序员修养](#)
- 专业基础篇
 - [编程语言](#)
 - [理论学科](#)
 - [系统知识](#)
- 软件设计篇
 - [软件设计](#)
- 高手成长篇
 - [Linux系统、内存和网络（系统底层知识）](#)
 - [异步I/O模型和Lock-Free编程（系统底层知识）](#)
 - [Java底层知识](#)
 - [数据库](#)
 - [分布式架构入门（分布式架构）](#)
 - [分布式架构经典图书和论文（分布式架构）](#)
 -



寻路之人	2018-06-26
耗哥，很期待你能介绍一下一些github上源码阅读的方法论和经验，对于想阅读源码的人有个指导方向。期待你的回复	作者回复
后面，我专门开一篇文章来讲这个事，敬请期待。	2018-06-26
颇试爱	2018-06-26
学无止境，33岁developer再出发！	2018-06-26
yangwnaghoayu	2018-06-26
特地来感谢下耗子叔解答疑惑，顺便说一下有几个感觉“一篇文章得让人学四五年没听懂”的评论：没必要花四五年学完一篇文章中提到的知识后才找工作，你学习的过程就在提高，能力高了现公司待遇不够了就跳槽，没有非得四五年学成再跳。最重要的是，文章的意义在于指引方向，没人指引而自己视野不够宽和远的话，自己不知道接下来要干嘛，会有些迷茫，甚至有些人停滞不前也不夸张。文章里面的成长框架可以让成长中的程序员思考自己，当你技术停滞不前时候结合这个看一下，自己可以提高下哪块的技能点	
落叶观禅	2018-06-26
最爽的一篇总结，用价值连城来形容一点不过！其中一部分已经学习使用过，一部分可以参考这个列表补充学习。耗哥真乃灯塔偶像！	
Terence	

各篇文章的超链接，在iPhone上需要长按弹出“拷贝”消息后，再继续长按，才有真正的地址链接拷贝或用Safari打开。	2018-06-26
我想，大部分人按住链接，就是为了copy相对应的地址，以便在别的地方查阅。	
就这一点来讲，目前这个方面的产品体验非常糟糕。	
作者回复	2018-06-26
在web上看应该会更好一点	
每日开箱测评	
你感觉写这么多有什么意义，一篇文章就让别人学四五年。四五年之后没公司需要，你管分配吗？	2018-06-26
作者回复	2018-06-26
照你这么讲，上大学有意义么？四五年后没公司要，还不管分配。	
另外，我这篇文章的开篇语不是说的很清楚了么？这条路上没有速成的，现在你怕了吧.....	
你要是能把之前的基础吃透，BAT随你去了。不过看到你连四五年都不愿意付出，你也只能平庸下去了	
每日开箱测评	2018-06-27
之前的留言太偏激了，对老师感到十分抱歉。但是还是希望老师能给一些一个月或者三个月左右的学习方法，毕竟四五年对任何人来说都很长。塞班系统被安卓系统干掉也只是很短的时间就完成了，学技术对时间的要求还是挺敏感的。我想大部分都是应用型人才，可能对大多数人来说不太适用。	
李康	2018-06-26
机器学习相关的推荐呢	
作者回复	2018-06-26
在后面，敬请期待	
每日开箱测评	
都看书名标题，我知乎上去搜就完了。不比你这全？	2018-06-26
作者回复	2018-06-26
那里没有路径。当然，那是你的选择	
iLeGeND	2018-06-26
走要这这补 也得小10年吧	
作者回复	2018-06-26
前面的基础打好了，这些东西1年左右就搞定，基础没打好，10年也不行	
Case	2018-06-26
学海无涯	
zzz	2018-06-26
耗子叔，你的极客时间的专栏文章，会一直都在吗？还是过了今年这期就删除了呢？	
作者回复	2018-06-26
应该会一直在的	
qwerboo	2018-06-26
满满的干货	
Abner	2018-07-17
突然觉得做了10年开发刚入门	
小鱼儿	2018-07-09
感谢耗哥！一直反复看	
lion_fly	2018-06-28
文章中给出的链接，有的需要翻墙才能访问，手机有时候翻墙不是很方便，能不能给出具体的链接。	
作者回复	2018-06-29
建议Web网页版阅读	
唐朝首都	2018-06-28
早上读了几篇，醍醐灌顶呀！！将这么多知识勾连起来，还有丰富资料，感谢耗子叔，起码当我觉得哪些方面需要提高的时候有非常清晰的路径。	
作者回复	2018-06-28
有收获就好，加油	
和💎💎	2018-06-27
再出发，再出发！	
卢瑾文	2018-06-27
不错，看了这么多，终于有了清晰的编程学习方法	

南山南_领云时代	2018-06-27
写的非常好，受教了！！	
ruby	2018-06-27
正在瓶颈期，浩哥一系列文章正是指路明灯啊，感激感激	
还有这种操作	2018-06-27
最近正在看内存相关的东西，苦于网上没有资料，感恩	
Bin	2018-06-26
我是一名java的程序员，最近从最基础的排序开始学习算法，学了一个月，从排序到堆栈链表二叉树过了一遍，代码都有敲多次。今天抽了两道之前做过的题目重新做，明显感觉到有些解题思路忘了费了很久才能写出来。我想知道学习算法的正确姿势，无限的刷题下去吗？	
jwcjlu	2018-06-26
皓哥，最近遇到一个网络问题，tcpdump发现有大量的tcp dup ack和tcp retransmission包，ping对端没有丢包情况，这个问题从那里入手，还望皓哥指点	
jony	2018-06-26
要加倍努力啦，耗子哥给的总结真的太给力啦，这就是我梦寐以求的导师啊，真想见一下耗子哥	
ghm,jm	2018-06-26
说到找书，就不得不说二手书的市场了，我觉得以后可能会越来越多的。	
夏洛克的救赎	2018-06-26
性能调优可不可以这样理解，就是充分利用操作系统对硬件资源进行合理高效的管理及利用，这些资源包括内存 CPU io。	
不做码农	2018-06-26
武功秘籍已经给出来了，能不能成功更多的是自己的坚持，当有了一定成绩以后，可能感觉学什么都不那么难了，这可能就是水到渠成吧，加油！	
霍骨	2018-06-26
这恐怕是最全的从0到真大佬学习手册了，让人热血沸腾💎💎	
wahaha	2018-06-26
tcmalloc 是 Google 的内存分配管理模块，全称是 Thread-Caching malloc，基本上来说比 glibc 的 ptmalloc 快两倍以上。	
Thread 读错了。 用极客时间几个月了，发现英文发音问题时而出现，还有同一个缩写，不同播音员发音不一致。	

