C++后台开发学习路线

编程指北 陈同学在搬砖 2020-10-14 17:52

前言

这一篇的主题是「Linux C/C++ 服务器/后台开发学习路线」。

这样的文章相信大家都见得不少了，写之前也非常忐忑，能不能和其它人写得不一样， 也定下了一个目标，这篇文章，不能是简单的堆砌学习资源和书单推荐，更要细化如何有效的去执行落地。

争取做到让看到的同学有一种相见恨晚的感觉哈哈哈。

所以大家可以试着看一下，帮我检查下有没有达到预想的效果哈哈，希望不要被打脸🅰️🅰️

那就正式开始吧。

这篇文章会有点长有点干，可以先去冲杯咖啡，慢慢看~

正文 | 干货 | 收藏

一、后端/后台/服务器开发?

经常在各大公司招聘上看到后端、后台、服务器开发等等，有些同学经常被这些名词搞混。

其实这些名词都是相近的，但是也有点区别，这里说说我的理解：

- 首先一般公司分为前端和后端，前端就是和用户打交道的，负责用良好的视觉效果将数据呈现给用户，广义的前端包括客户端（安卓、IOS）、Web前端、小程序等。
- 而与之对应的后端则是负责业务逻辑处理，比如下单、支付等，重在业务流程的处理。

后台一般和后端是一个意思，而服务器开发则稍微广义一点，不仅包含了后台开发，而且也包括支撑整个后台应用的基础开发，比如搜索引擎、微服务、RPC 框架、KV、存储、MQ 等。

后台/后端重在业务处理，是偏向应用层开发，而服务器开发不仅包括应用层开发，更是囊括了整个支撑后台业务的相关组件的开发。

那 Linux C/C++ 服务器/后台开发指的什么呢，其实就是基于 Linux 上 的 C++ 编程。

但是相比 Java 系更强调 Linux 系统编程、网络编程能力，有的还会涉及到服务端底层协议和网络框架开发。

传统的 Java 、Go后台开发偏向 Web 开发，也就是接收前端请求，通过微服务互相调用，完成业务逻辑处理，然后返回给前端。

实际上在腾讯这边的 C++ 后台开发，也是类似的，本身有非常成熟的基于 C++ 的微服务体系，大多数开发也只需要关注业务逻辑就好，不过还是会要求 Linux 系统编程、网络编程等能力。

二、后台开发都考察哪些?

一般来说 Linux C/C++ 后台开发方向涉及以下这些基础知识：

- C/C++ 语言特性和实现原理
- 计算机网络
- 网络编程 和 Linux 系统编程
- 操作系统原理
- 部分 Linux 内核原理，如内存管理、文件系统、虚拟内存等
- Linux 常见命令使用
- 算法与数据结构
- 数据库使用及原理
- 常见 NoSQL组件，如 Redis、Memcached
- 版本控制 Git

非必选加分项：

- 分布式相关，如一致性协议比如 Raft 算法、分布式存储等
- docker、k8s 等虚拟化和云计算相关的
- 系统设计能力，如短链服务、评论服务、Feed流系统、抢红包、秒杀等

由于篇幅限制，这篇文章主要介绍基础知识的学习路线和方法，其它加分项以后再单独写。

三、C/C++

首先是语言的基础知识，一些关键字和实现原理等：

- 指针、引用、数组、内存
- 引用与指针区别
- C 和 C++ 的一些区别，比如 new、delete 和 malloc、free 的区别
- 虚机制：虚函数、虚函数表、纯虚函数
- 继承、虚继承、菱形继承等
- 多态：动态绑定，静态多态
- 重写、重载
- 智能指针原理：引用计数、RAII（资源获取即初始化）思想
- 智能指针使用：shared_ptr、weak_ptr、unique_ptr等
- 一些关键字的作用：static、const、volatile、extern
- 四种类型转换：static_cast, dynamic_cast, const_cast, reinterpret_cast
- STL部分容器的实现原理，如 vector、deque、map、hashmap

- 模板特化、偏特化，萃取 traits 技巧
- 编译链接机制、内存布局（memory layout）、对象模型
- C++11 部分新特性，比如右值引用、完美转发等

这里列出来的只是一些比较重要的部分，实际上可能只算 C++ 的冰山一角， 大家且学且珍惜吧，这不 C++11 还没整透彻，C++ 20 又出来了，生命不息，学习不止。

怎么学?

1.《C++ Primer》

> 这本书基本包括了 C++ 11 的全部特性，最好把前面三部分：C++基础、C++标准库、类设计者的工具都看一遍，我当时花了一个多月断断续续看到了第16章模板那里。

2. Effective 系列：《Effective C++》、《More Effective C++》、《Effective STL》

> 第一本是重点，光看《C++ Primer》缺少实践的话，大概率还写不出合格的 C++ 代码，而《Effective C++》就是通过 55 条非常具体的做法告诉你什么样才是符合 C++ 编码规范的，可以缩短你写出合格 C++ 代码的时间，减少踩坑，强烈推荐必读，后面两本优先级稍低，可以有时间再读。

3.《STL 源码剖析》和《深度探索 C++ 对象模型》

> 看完 Primer 和 Effective，你应该已经能够比较熟练的使用C++了，但是还缺少对 C++ 底层实现机制的认识。比如虚函数表、成员变量布局等，同时对于 STL 库可能也仅仅停留在使用上。
> 推荐的这两本可以分别完善你在 C++ 底层实现和 STL 源码、原理上的认识。

以上书籍同时建议和侯捷老师的视频配合服用，效果更佳。

直接在 B 站搜索「候捷 C++」即可，主要有以下几个系列：

- 《C++内存管理》
- 《STL源码分析》
- 《C++ STL与泛型编程高级》
- 《C++11 新特性》

我基本都看了，收获挺大的，建议看下，可以开倍速。

看完以上资料，算是 C++ 入门了，应付面试也是足够的，基本到达了正确高效地使用 C++ 这一层面。

是不是听到这有点崩溃，特么的看了这么多，才入门？？？

如果你想在 C++ 语言上更进一步，那么有以下的书籍推荐：

- 《C++ 语言的设计与演化》

> 这本书是 C++ 之父 Bjarne Stroustrup 写的，关于 C++ 的前世今生，以及未来的演进方向，可以了解 C++ 的设计哲学。C++ 复杂的语言特性一直让人诟病，通过这本书，可以看到各种特性引入的目的，也更深入了解到了 C 和 C++ 之间关系。比如 C++ 里有个原则就是所有的实现机制都不能带来额外的运行时开销。

我也正在看这本书。

- 《C++ 沉思录》

- 《C++ Templates》和《C++模版元编程》

> C++模板元编程属于另外一个世界了，一般公司里开发用得比较少，这个也是一个大坑，如果实在感兴趣可以去看看，感受下 C++ 的博大精深，不过这玩意我也不太会，也不推荐你去花时间在上面。

- CppCon视频

> 这是 C++ 社区组织的类似开源峰会那种，每次都会讨论一些关于 C++ 的话题，没事去刷一个，还是挺有意思的。Youtube 直接搜 CppCon 即可找到。

四、操作系统

操作系统这门课，我的感觉是易学难精，但是掌握到日常编程和面试够用还是比较容易的。

那么毕业生或者说你去准备校招面试应该达到怎样的水平:

- OS 四大模块的理论知识: 进程与线程管理、内存管理、IO与文件系统、设备管理
- 了解 Linux 内核部分实现原理，如内存管理、进程管理、虚拟文件系统等

其中内存、进程、IO 是重点，这几块也是和编程关系最密切的，这里推荐先挑本偏理论的书看看，了解操作系统的全貌：

- 《现代操作系统》
- 《操作系统—精髓与设计原理》

不必全看，两者任选一本都不错，我自己是仔细看了第二本，因为是我们教材，同时挑着看了现代操作系统部分章节。

这部分看完你应该对下面这些话题有一个清晰认知了：

- 操作系统由哪些构成
- 进程的状态、切换、调度
- 进程间通信方式（共享内存、管道、消息）
- 进程和线程的区别
- 线程的实现方式（一对一、多对一等）
- 互斥与同步（信号量、管程、锁）
- 死锁检测与避免
- 并发经典的问题：读者写者、哲学家就餐问题

- 为什么需要虚拟内存，MMU 具体如何做地址转换的
- 内存为什么分段、分页
- 页面置换算法
- 文件系统是如何组织的
- 虚拟文件系统（VFS）是如何抽象的
- ...

但是这还不够，看完偏理论的书，当面试官问「进程和线程的区别」时。

大概只能回答出「进程是资源分配的最小单位，线程是CPU调度的最小单位，balabala...」这样正确却普通的答案。

但是如果你了解 Linux 内核的实现，就可以实际出发，讲讲 Linux 中进程和线程是如何创建的，区别在哪里。

比如在 Linux 中进程和线程实际上都是用一个结构体 `task_struct` 来表示一个执行任务的实体。进程创建调用 `fork` 系统调用，而线程创建则是 `pthread_create` 方法，但是这两个方法最终都会调用到 `do_fork` 来做具体的创建操作，区别就在于传入的参数不同。

深究下去，你会发现 Linux 实现线程的方式简直太巧妙了，实际上根本没有**线程**，它创建的就是进程，只不过通过参数指定多个进程之间共享某些资源（如虚拟内存、页表、文件描述符等），函数调用栈、寄存器等线程私有数据则独立。

这样是不是非常符合理论书上的定义：同一进程内的多个线程共享该进程的资源，但线程并不拥有资源，只是使用他们。

这也算符合 Unix 的哲学了— KISS（Keep It Simple, Stupid）。

但是在其它提供了专门线程支持的系统中，则会在进程控制块（PCB）中增加一个包含指向该进程所有线程的指针，然后再每个线程中再去包含自己独占的资源。

这算是非常正统的实现方式了，比如 Windows 就是这样干的。

但是相比之下 Linux 就显得取巧很多，也很简洁。

对于进程、线程这块你还可以把 fork、vfork、clone 、pthread_create 这些模块关系彻底搞清楚，对你理解 Linux 下的进程实现有非常大的帮助。

说了这么多，就是想强调一下理论联系实际的重要性。

特别是操作系统，最好的实践就是看下 Linux 内核是怎么实现的，当然不是叫你直接去啃 Linux 源码，那不是一般人能掌握的。

最好的方式是看书，书的脉络给你理得很清晰。

书籍推荐：

- 《Linux内核设计与实现》

> 这本书恰到好处，即讲清楚了内核实现的要点，又不会通篇源码。

这本书重点关注「第 3 章进程管理」、「第 5 章系统调用」、「第12章内存管理」、「第13章虚拟文件系统」、「第15 章进程地址空间」

这些章节属于操作系统核心部分，其它如中断处理、块 IO、设备管理根据你自己兴趣选择看下就可以了。

基本上做到这里，操作系统就没什么大问题了。

五、计算机网络

需要掌握的网络协议和知识：

- HTTP、TCP、IP、ICMP、UDP、DNS、ARP
- IP地址、MAC地址、OSI七层模型（或者 TCP/IP 五层模型）
- HTTPS安全相关的：数字签名、数字证书、TLS
- 常见网络攻击：局域网ARP泛洪、DDoS、TCP SYN Flood、XSS等

计网知识比较繁杂，很多同学都反映网络很难学，一大堆的网络协议，依次学完后，还是不知道网络是怎么构成的。

这就是没有用对学习方法，导致只见树木，不见森林。

学习时，推荐你抓住一条主线 **「一个数据包是如何发送出去的？」**

带着这个问题依次去学应用层、传输层、网络层、链路层，思考这些层之间是如何串联起来的。

这就是自顶向下的思路，那自然要推荐:

- 《计算机网络：自顶向下方法》

> 这本书从我们最常接触的 HTTP、FTP、SMTP 等应用层协议讲起，可以清晰看到引入各个层的作用。

比如为了区分同一个主机的不用应用，引入了传输层，并使用不用的端口号作为区别；

为了在不同子网间传输数据引入了网络层，并使用 IP 地址寻址路由;

网络层解决了不同子网间路由的问题，但是同一个局域网内确定主机却是通过 MAC 地址，所以引入了链路层来承载 IP 数据包;

同时为了将 IP 地址和 MAC 地址做转换映射又产生了 ARP 协议。

层层递进，逐层揭开网络，非常推荐！

还有一本书：

- 《网络是怎样连接的》

> 非常浅显易懂的描述了「一个数据包是如何发送出去的」，也不费时间，看惯了机工社的大黑书，看这种反而有种看小人书的感觉，有基础的话，一天左右就过完了。

只有把握住了整个网络脉络主线才不至于被纷繁复杂的网络协议所搞晕，剩下的就是不断的细化，填充这些主干上的细枝末节。

那么有哪些细节可以去填充呢?

比如 ARP 工作过程、IP 地址、IP 分片、NAT（UDP 打洞）、链路层访问控制协议等等。

还有最重要的 TCP 协议，TCP 也是面试和计网中最重要的概念：

- 三次握手、四次挥手
- 状态转换
- TCP 状态中 TIME_WAIT
- 拥塞控制
- 快速重传、慢启动等

这么多东西肯定需要背，但不要死记，最好带着问题去思考为什么要这样做。

这里列几个问题：

- TCP 如何实现可靠传输的（画外音：如何基于 UDP 实现可靠传输
- TCP 连接建立为什么不是两次握手（画外音：三次握手的充分必要性说明
- TIME_WAIT 的存在解决了什么问题，等待时间为什么是 2 MSL

整个 TCP 的核心就是围绕着 **可靠传输** + **高效传输（流量控制和窗口管理）**

由于 TCP 的细节实在太多，自顶向下那本书有点不太够，所以你需要去看看：

- 《TCP/IP详解卷1：协议》

> 这本书不要从头看，而是挑出其中涉及到 TCP 的章节

到这里，对于整个网络以及 TCP 都应该有了一个全面而细致的认识。

但是计网中还是有一些有意思的问题，如果你没思考过，也许回答不出来。

比如：

- 为什么有了 MAC 地址还要 IP 地址，IP 地址和 MAC 地址的区别是什么?
- 如何理解广播域和冲突域?
- 路由器和交换机有什么区别?
- TCP 连接的本质是什么，真的是“链接”吗？（曾经被问过：Java socket 创建的 TCP 连接，对于主机挂了和 JVM 挂了有什么区别?

这些问题只有当你真正理解了才能回答出，仅仅记住协议的话，估计很难应对灵活的面试题。

此外，网络部分还需要准备 HTTP、HTTPS，推荐：

- 《图解HTTP》

最后别忘了自己回答一遍那被问烂了、写烂了的问题：

- 从 URL 输入到页面展现到底发生什么

越细越好，五百字以上吧，哈哈哈

六、网络编程

C++ 后台开发基本是离不开网络编程的，其实甚至整个后台开发也可以看做是在做网络编程。

只不过别人的框架帮我们做了协议解析、网络数据传输、解封包这些底层操作。

比如 SpringBoot 这种保姆级框架，基本上属于将一个框架能干的事都干完了，以至于我们开发业务只需要定义接收和返回包的数据格式，然后做逻辑处理就完了。

像序列化、解封包、IO 处理这种网络编程必备的脏活业务开发根本不会接触到。

但是网络编程技能还是很重要的，特别是对于 Linux C++ 开发来说。

Linux 下网络编程核心的包括系统编程和网络 IO 两个部分：

- 进程间通信方式：信号量、管道、共享内存、socket 等
- 多线程编程：互斥锁、条件变量、读写锁、线程池等
- 五大 IO 模型：同步、异步、阻塞、非阻塞、信号驱动
- 高性能 IO 两种模式：Reactor 和 Proactor（ 但是 Linux 下由于缺少异步 IO 支持，基本没有 Proactor
- IO 复用机制：epoll、select、poll（破解 C10K 问题的利器）

推荐的书：

- 《Unix网络编程》
- 《Unix环境高级编程》

> 这两本是砖头书，虽然是网络编程和 Unix 系统编程方面的无出其右的圣经，但主要用途还是垫显示器（逃， 个人觉得这种书不是面向读者的，具体原因和如何阅读这种书在后文介绍。

- 《Linux高性能服务器编程》

> 我强烈推荐，这本书前半部分基本是在重复计网基础知识，但是后面几章关于高性能服务器程序框架、高性能IO、IO复用、定时器、多线程编程、线程池和进程池还是讲得非常全面到位的，值得一看，看完基本上对于整个网络编程就有了框架。

- 《Linux多线程服务器端编程》

> 这本书同样强烈推荐，这是陈硕大佬写的书，说实话第一部分：C++ 多线程系统编程都直接把我看蒙了，没有想到 C++ 里要做到线程安全这么难，第一章我看了两三遍才看懂吧。。。这是难得的讲解 C++ 多线程编程的书。

并且在书中，陈硕大佬用了一章讲解了 Muduo 网络库设计与实现，Muduo 比较适合学完基础的网络编程后继续进阶学习如何设计和写一个网络库，是一个高质量的 Reactor 网络库，采用 **one loop per thread + thread pool** 实现，代码比较简洁，书和源码搭配着看作为学习网络编程方面来说是非常不错。

学完网络编程就可以写点小项目练手了，这里列举几个项目：

- HTTP 服务器，这个似乎成了 Linux C/C++ 人手一个的项目了？

  这里推荐两个做为参考：

  https://github.com/imarvinle/WebServer：这是我看完高性能服务器编程后写的

  https://github.com/linyacool/WebServer ： 这是牛客 linya 大佬写的

  不过 HTTP 服务器看着挺简单的，但是可以扩展写的地方还是挺多的，比如可以加入代理功能，这部分我在留学生 lab 中写过，但是没有集成到这个里面来，可以加入日志库，可以添加 CGI 支持等等。

- 网络库

  这个也算是造轮子了， 可以就采用 one loop per thread + thread pool 这种模式，先去看懂 Muduo 源码，然后自己再写一个类似的，这个过程就算是抄，你也可以学到不少东西的，学编程不就是这样先看，再模仿、修改，然后创新吗？

- RPC

  写一个 PRC 你需要考虑到序列化、网络传输、服务发现等，比较有名的有 grpc、brpc，这两个网上文档都比较完善，可以学习一下实现原理。

  这里还有一个简单版本的：https://github.com/guangqianpeng/jrpc

- 类似QQ的网络聊天室

  简单版的就可以直接在局域网内实现群聊、单聊等。

  更进一步可以考虑一下如何不通过服务器中转消息实现 P2P 聊天，类似 QQ，这里会涉及到 UDP 打洞、NAT 转换等知识，还是很有意思的，我大二用 Java 搞过。

七、系统级编程

作为 C/C++ 程序员，编写的程序不像 Java、Python 这些是在虚拟机上，直接就是在操作系统上运行，那么就必须了解操作系统底层机制和运行原理。

就和 Java 程序员要求了解 JVM 是一个道理，你得熟悉代码运行的平台，才能在出问题的时候准确定位到。

这个也是在我们学校在大三开设的一门课程《System Programing》，从 CMU 引进的，教材也是沿用 CSAPP，这也是我觉得大学上过最值的课了。

我重新认真读 CSAPP 就是在大三上这个课期间，包括做了每个章节附带的 lab， 这是我当时做 Bomblab 的题解：

https://www.jianshu.com/p/479333cbccc4

这里推荐两本书：

- 《深入理解计算机系统》

> 不需要我多介绍了

- 《程序员自我修养》

> 别被名字欺骗了，这不是教你养生的，而是学了会掉头发的硬核知识

两本书侧重点各不相同，CSAPP 非常巧妙的把数字电路基础、二进制与指令集体系、汇编语言、程序设计及优化、存储器体系结构、链接与装载、进程、虚拟内存这一堆来自各不同的计算机学科的核心知识点串在一起，并以程序员的视角呈现，所以这本书的英文名字叫《Computer Systems A Programmer's perspective》。

而程序员自我修养则重在链接、目标文件、装载、库与运行时，看完这本书你会了解到一个 C/C++ 程序是如何被编译成目标文件的，以及 Linux 下目标文件的格式，不同目标文件又是如何被链接成一个可执行程序，在链接时如何处理符号、重定位、地址解析等，以及静态链接、动态链接区别等等，最后可执行文件又是如何被加载进内存，如何和虚拟内存空间映射的。

你可能会觉得这个又是只能用于面试，实际派不上用场的知识?

那简直大错特错，说真的，这两本书，我是反复看了三遍以上，当然后续看都是挑着重点看的。

举个例子吧，写 C/C++ 的同学没少遇到这些编译错误吧：

`undefined reference to xxx`

`Symbol key multiply defined (by xxx.o and yyy.o)`

在我大一的时候遇到这些问题简直一脸懵逼，根本连报错都看不懂。

特别是涉及到多文件编程的时候，经常傻乎乎的在头文件中定义变量，导致变量多重定义，这些问题没有学过链接知识的其实很难理解。

在实际编程也是经常会遇到的。

又比如 `extern`、`static` 这些关键字是如何在编译链接时起作用，变量的申明与定义又有什么区别?

这部分可以算是真正的内功了，提升你对计算机系统的理解，也有助于解决实际编程过程中会出现的问题，当然也会在面试中出现。

八、数据库

数据库首先要学会 SQL 的使用，这里推荐《MySQL必知必会》。

数据库原理方面可以看看《数据库系统概念》，这本书挺厚的，包含了从 SQL 到数据库设计再到数据库原理、分布式数据库都有，可以挑着看，比如关系模型、数据库设计（三大范式）、数据磁盘存储和组织方式、索引、并发控制等。

当然了整个数据库最重要的还是索引和并发控制（锁、MVCC等），这部分也是面试常考的：

- 索引存储结构：B树、B+树索引、Hash索引
- 索引的使用：主键索引、覆盖索引、最左前缀原则、索引下推等
- 锁：乐观锁、悲观锁、表锁，行锁，意向锁，读锁，写锁等等
- MySQL InnoDB MVCC 实现机制
- 存储引擎：InnoDB、MyISAM等，各自的优缺点
- 事务：ACID理论

这部分推荐两本书：

- 《高性能MySQL》
- 《MySQL技术内幕》

> 这两本主要对索引、innodb存储引擎、锁、并发控制讲得比较清楚，建议挑对应章节看。

九、算法和数据结构

首先需要掌握常见的数据结构：

- 线性表、数组、链表
- 栈与队列
- 树、二叉树、多叉树实现和遍历方式，AVL树实现以及插入删除过程、红黑树（了解定义即可）
- 图，以及图的实现方式、遍历
- B树、B+树
- 堆
- 散列函数和散列表

常见的算法：

- 排序算法：冒泡、插入、快速、希尔、堆排、基数、归并等
- 字符串匹配算法：KMP
- 常见算法思想：递归、枚举、递推、分治、贪心、动态规划等

视频可以看看：

- mooc 上浙大的《数据结构》
- 学堂在线上清华邓俊辉老师的《数据结构与算法》

这两个是我看过觉得不错的才在这推荐，第一个是初学数据结构时跟着看，第二个是大三复习时刷的。

入门版书籍可以看看：

- 《啊哈算法》
- 《算法图解》

稍微进阶点的：

- 《算法第四版》

> 这本书强烈推荐，难度适中，但是全面。

终极版：

- 《算法导论》

> 这个量力而行就好了。。。
> 推荐理由是：我不推荐显得没有逼格（：

有了基础的算法思想和数据结构储备，剩下的就是刷题了：

- 《剑指offer》

> 建议必刷

- leetcode

> 建议分类刷，先易后难，比如数组、二分、二叉树、动态规划，一个一个系列搞定，总结经验，保证 150 道简单和中等以上吧

最重要的是，保持手感，有空就刷一道。

十、网站和视频

有些同学喜欢看视频，那我也在这里统一推荐一下吧

- B站

> 你的一站式学习网站，用你想学的关键字在这搜就完了

- 中国大学 mooc

> 基础课程学习

- 网易云课程、学堂在线

> 一些技术课程、公开课学习

Linux C/C++ 可以去看看黑马的，我试看过几集 IO，讲得还是不错的，就是有点慢，个人觉得不如看书。

刷题可以看看牛客左神的视频

计算机网络可以看看 B 站韩立刚或者 mooc 哈工大的

操作系统可以看看学堂在线上清华的《操作系统》

推荐了这么多书，从哪看起?

一、浅谈学习方法

如果你认真看过我前面介绍每一部分的学习路线时，可以发现我特别强调学习抓**主线**，并且将每一个基础知识的主线给大家列出来了。

这里再次强推，学习新东西的时候，重点是先对整体脉络、知识结构有一个大概的映像和了解，然后抓住这个领域的主线，顺着主干，突出重点去学习。

集中时间，速战速决，不要将时间线拉得太长，越长可能越坚持不下去，效果越差。

对于细枝末节的内容，可以留到实践的时候，用到了再去查！

如果一头扎进零碎的知识，去看手册、字典型的书，那必然是事倍功半的。

细节留给实践去补充，我们的时间要花在刀刃上，注重知识的体系性和框架的建立。

二、常见问题

说实话，其实这些书籍或多或少都被各路大神推荐过，确实经典。但是大神们却很少告诉你他们是如何去看的，该怎么去看这些书，难道一本本一页页的挨着啃吗?

这部分才是我今天最想说的部分，「该如何去看大厚书」。

比如我后台就经常有同学问：

- 有些书看的找不到重点，看不下去了，怎么办，比如深入理解计算机系统，UNIX 网络编程，APUE，求指教

- 我不是科班 CSAPP 可能不是全部看得懂，该怎么办?
- 大佬，这么多书看得完吗?
- 刚开始看这些书很痛苦怎么办?

这些书该怎么看，可能过来人，准备过秋招的都比较清楚，但是作为还在大二、大三的会比较懵逼。

就拿我当时亲身经历来说，在网上搜网络编程如何学习，很多人都推荐 UNP、APUE。

好嘛，买来看，从第一页挨着挨着看，而且书中的示例代码我大部分也照着敲了，最后看了七八章，发现始终是在学一些 socket api 和 系统 api 的用法，没摸到网络编程的框架思维。

后面我又去搜，看到有人推荐《Linux高性能服务器编程》，去豆瓣看了下了目录，似乎正是我想要的东西。

直接找来 PDF 开始看，果然这本书才真正让我理解了网络编程的整个套路和框架，学到了各种事件处理模式、计时器、信号处理、线程池这些网络编程中很重要的东西。

当然 UNP 和 APUE 也是不能丢的，这两本书我当做了字典查询，比如学到了 IO 部分，回去看 UNP 中关于五种 IO 模型的介绍。

用到 connect、listen、bind 这些函数，再回去看 UNP 第四章，不得不说，UNP 关于这些 socket API 的使用和各种异常情况的处理方式都介绍得非常详细和深入，不愧是网络编程领域的圣经。

但是初学者看却容易在细节中迷失，抓不到纲领，这也是这类书的缺点。

类似 UNP、APUE 这种书本身是面向知识体系的，而不是面向读者，它们其实更像字典，把这个领域内的所有知识，非常细致的堆叠在一起，看上去就是平铺直叙，充斥着细节，对读者极其不友好。并且书里内容实在大而全，很多根本不用学。

比如 UNP 讲了 sctp 这种协议用法、多播、unix 域协议这些实际用得很少的东西，挨着看不仅会看不下去，而且比较浪费时间。

但是它们又是经典的，确实是这些领域内在体系性和深入性上都做得非常好的书。

什么书才是面向读者的呢?

那就是抓住该领域的核心主干，提纲挈领，带领读者由浅入深，同时又有一定的细节，看完让人茅塞顿开。

比如《自顶向下》、《Linux高性能服务器编程》、《Linux多线程服务端编程》、《STL源码剖析》、《Effective C++》、《CSAPP》、《程序员自我修养》等都有各自想要论述的主线在里面，看起来也是一环扣一环，非常循序渐进。

我的看书方法就是对于面向知识体系那种堆砌细节的书，我们先浏览目录，做到对整本书有映像，再大致看一些我们关心的部分，比如 UNP 和 APUE 中 IO、文件、进程控制、信号、线程、线程控制、基本套接字编程 这些是比较重要的模块，其它边角知识，可以用到再去查。

还有一点，很多同学反映看不懂类似 CSAPP 这样的书，那我们都知道，任何一本书基本上都是有前置依赖的。

没有掌握要求的背景知识去看肯定很吃力的。

就比如我大一下只有基础的 C 知识和一丁点计算机导论知识，然后屁颠屁颠的跑去看 CSAPP（学长毕业摆地摊卖书我瞎买的），那时候只知道这本书被誉为神书，但是看到前两三章就蒙了，真的有点难，对于当时的我来说太底层了，根本不知道在说啥，看过也只是看过，就像天空飞过鸟儿，但没有痕迹。

直到后来大三再次拿起，我才意识到这本书的伟大之处就在于将计算机不同学科知识有机的串在了一起。那时候看，更多是一种补充、深入学习以及完善了，因为很多知识分别在数字逻辑、汇编语言、操作系统这些课程中学过了。

所以要明白，你看不懂不是因为你笨没天赋，而是你有前置依赖的知识没有完成，还没学会走，就想跑了。

一般来说，每本书的首页会介绍看这本书需要哪些前置知识，可以关注一下。

还有一种看书的方法，我在复习的时候采用过，那就是横向学习。

比如我复习操作系统，在《操作系统：精髓和设计原理》中看到了关于内存、虚拟内存的各种介绍，看完理论再去看《Linux内核设计与实现》12 章「虚拟内存」、15 章『进程地址空间』，最后再去看《CSAPP》第 9 章『虚拟内存』，这样看下来，基本上内存这块理解得比较透了，这些书关于这块的介绍是各有优缺点的，正好互补。

又比如在《精髓与设计原理》中介绍了进程加载和链接，其实讲得比较偏理论，看完还是觉得似懂非懂。

那我又会去《CSAPP》看第 7 章「链接」，这一章基本讲清了静态链接、目标文件、可重定位目标文件、引用解析、加载这些关于链接的核心概念，但是一个章节讲这么多，难免不够深入。

我又会去看《程序员自我修养》这本书第 4 章「静态链接」、第 6 章 「可执行文件的装载与进程」、第 7 章 「动态链接」，这本书核心主题就是链接、加载，所以这一路看下来，对于链接、加载这块基本上搞得比较透彻了，也许没几个面试官有你清楚。

同样索引你也能从《数据库系统概念》、《高性能MySQL》、《MySQL技术内幕》中挑选对应的章节，串起来看，取每本书优点，这样学习真的很高效也很深入。

这就是我在复习的时候采用的用知识点串联，跨多本书高效精准的复习方式，效果也很不错，春招十几次面试没有一次因为这些基础知识挂过。

啰嗦了一大堆，就是回答这些问题的：

> “有些书看的找不到重点，看不下去了，怎么办，比如深入理解计算机系统，UNIX 网络编程，APUE，求指教”、“我不是科班 CSAPP 可能不是全部看得懂，该怎么办？”、“大佬，这么多书看得完吗”、“刚开始看这些书很痛苦怎么办”

三、要花多久时间才能学完

这个不好说，根据你的基础和学习效率不同，比如我大一、大二对于一些基础的知识学得比较认真，基础还算可以，按照这样一套走下来也就大半年。大概每天花四五个小时以上吧。

如果真的是连计网、操作系统理论这样的东西一点基础都没的话，那估计得一年以上，毕竟这些内容基本覆盖了科班的核心课程，人家上三年课，你一年解决，已经算很快了好吧。

> 一年真的足够从小白学起吗?

感觉是差不太多的，但是估计得每天付出五六小时以上了，并且学习方法得当。

你去牛客就会发现，存在各路大三、研一自学转码的同学，最后还能成为 offer 收割机，所以，不要怀疑一年不够，最关键的是你要找到正确的路线，然后执行下去。

> 文中推荐的书真的全部要看吗?

当然不是，我自己都没看完，但是我的策略已经说过了，基本上大部分书都看了重要的章节，这样看起来是很快的。

并且随着你看书越来越多，基础越来越好，你会发现每本书前面几章都是铺垫基础知识，大部分可以直接跳过，举个例子

《Linux高性能服务器编程》这本书前几章是这样的：

- 第1章 TCPIP协议族
- 第2章 IP协议详解
- 第3章 TCP协议详解
- 第四章 TCPIP通信案例：访问Internet上的Web服务器
- 第五章 Linux网络编程基础API

你觉得这些章节在看过《自顶向下》、《TCP/IP详解》之后还有必要看吗？我反正是半天扫过去就完了。

最后，不管说再多方法，再多的路线，最终都需要自己花时间去啃、去执行。

四、语言疑惑

还有一个很多选择 C++ 方向同学都存在的疑惑，在这里我也想解释一下：

> C++ 语言特性多，又难学，很多都是底层开发才会用到，C++ 就是个坑，是否应该转 Java、Go 呢?

当然不是的，的确在头条、美团、阿里这种业务部门使用 Go、Java 系更多，首先还是那个观点，校招生对于企业来说都是一张白纸，面试官考察的是你的基础知识和聪明度，来决定是否有培养潜力，语言确实不重要。

那你可能会说，明明各种面经上常常出现 ConcurrentHashMap、虚表、虚函数实现机制这样和语言强相关的问题。

在我看来啊，面试深入问一些语言实现细节，其实不是在考你语言，而是看你是否有主动钻研的意识，是不是只停留在应用的层面，同时也借语言考察一些数据结构、操作系统方面的基础知识。

所以呢，我觉得 C/C++、Java、Go 你深入学习哪一个都可以，关键还是找对相应的学习路线，一直坚持学下去，不要每天都停留在我到底是学 Java 好还是 C++ 好这样无解的问题。

另外，想对学 C++ 的同学说，可能你会发现身边同学都在搞 Java、Go 之类的，找工作缺少一些一起复习准备的朋友，有些甚至劝你别学 C++。那这个时候你一定要坚定自己的选择，多在牛客或者网上找找同方向的朋友一起交流、学习。

说实话，就找工作这块来说，个人觉得区别真的是不大的，不管从薪资、面试难度来说都是差不太多，更多的还是算法和基础知识。

而且也有不少同学 Java 进腾讯需要转 C++，C++ 进阿里、美团需要转 Java 的，这都不是事儿。

> 那 C++ 目前应用场景有哪些呢?

一句话，对性能或者执行效率要求比较高的应用，比如游戏引擎、infra、推荐引擎、存储等，当然也能拿来写业务（没错说的就是鹅厂），也有做 C++ 客户端开发的，主要是 MFC、QT 等。

说实话，像游戏引擎、infra这类都是门槛比较高的，并且招聘的数量也有限，一般人很难进，而且目前互联网公司的业务部门大多使用的是 Java、Go这类语言。

所以 C++ 的需求量是相比 Java、Go 这类少很多的，但是同时学习 C++ 也没 Java 那么多，所以相对来说竞争还没那么大，并且 C++ 学的不错，你同样可以去面阿里、美团这种 Java 技术栈的公司，大厂基本不会限制语言的。