Java 后台超详细系统学习路线（2024年最新）

但是现在学 C++ 的相对 Java 来说少很多，很多学 Java 的读者也一直在催我出 Java 版。

更是在知乎立下了 flag：

所以这周唯一一天的休息日，就贡献给了 《史上最新 Java 版校招学习路线》，我们先看下思维导图吧：

Java后端开发学习路线

校招的话，其实关注「Java校招」那下面就好了，其它只是整个 Java 后端的全景图。

本来在 C++ 版本的基础上把 Java基础、JVM、Java Web 这些补充上就好了，因为 CS 基础知识都是共通的。

但是呢，为了更好的帮助到学 Java 的同学们，还是根据 Java开发岗的要求对类似 Linux 编程、网络编程等重要性进行了调整。

在这分享一下我学习过程中整理的书籍、面经、博客等，不是在网上那种打包下载的，而是自己需要学到某个方向知识的时候，去挨个找的，最后汇总而成。

有需自取: 计算机必看经典书单(含下载方式)

#

一、学习时间

先说一下，我预期中，按照这个路线学个六七成，大概需要多长时间：

1. 如果你是非科班，比如传统工科，大学学过点 C 编程，其余知识完全不具备，那么可能需要 8 个月 - 1年 以上。

学习效率高、比较拼的可能五六个月也行（每天高强度学习、看视频、刷书十几个小时）。

2. 如果你有其它语言基础、也学过操作系统、计算机网络这些 CS 必修课，那么只需要再强化突击刷题 + 系统化梳理下面试题 + 学习 Java 生态，这个过程 6 个月足够，3个月勉强。

3. 如果你本身是学 Java 的，也做过项目，也学过 CS 基础课，我觉得你强化突击 1-2 个月就足够去面试了，主要是看一些高频面试八股文和刷 LeetCode。

当然，这些都是时间都是因人而异的，我只是给出一个我觉得合理的大概时间作为参考。

#

二、Java SE

#

2.1 核心知识

第一要学的就是 Java 基础，也就是 Java SE，那么 Java SE 部分知识都有哪些呢：

基础语法：if、else、函数

面向对象：继承、封装、多态

接口

异常

I/O：文件、网络编程

泛型

反射

注解

#

2.2 有其它语言基础需要多久？

如果有 C 语言基础，1-2 个月应该很充分。

如果有 C++ 基础我估计两周就足够了？

因为实际上，把语言拆解为「语法特性的组合」，那么 Java 相对于 C 多了 面向对象、接口、异常这些东西，而 C++ 则几乎除了反射、注解不支持以外，其它都是 Java 的超集。

所以实际上对于学过 C++ 的同学，学 Java 就是换关键字，没有新的语言特性学习负担，所以我说会很快。

#

2.3 开发工具

#

2.3.1 JDK & JRE

Java 初学者需要在电脑上安装 JDK 和 JRE。

JRE（Java Runtime Environment）和JDK（Java Development Kit）都是Java平台的组件。

JRE是 Java 应用程序的运行环境，它包含Java虚拟机（JVM）和 Java 类库，可以让你的计算机可以运行Java程序。如果你只需要运行 Java 应用程序而不需要进行开发，则只需要安装JRE。

JDK是Java开发工具包，它包含JRE、Java编译器（javac）、Java文档生成工具（javadoc）、Java调试器（jdb）等开发工具，支持程序员编写、编译、调试和部署Java应用程序。

简而言之，JRE是运行Java程序所必需的，而 JDK是开发Java程序所必需的。

#

2.3.2 Intellij IDEA

集成开发环境推荐大家用 IDEA，不过如果你后续学习看的 Java 视频比较老的话，老师大多数是用的 Eclipse，这个看大家，如果刚开始可以跟着和老师采用一样的开发工具，方便学习。

熟悉后还是建议用 IDEA，Intellij IDEA在调试、自动完成和重构方面的表现都比Eclipse更优秀。

这里有一份 IDEA 的使用教程: IntelliJ-IDEA-Tutorial

(opens new window)

#

2.4 视频推荐

#

2.4.1 尚学堂高琪Java300集

Java SE 我推荐看视频学习，不要一上来就去看书，我当时大一看的是《尚学堂高琪Java300集》，不知道多少人看过这个版本，我真觉得讲得挺好的。

https://www.bilibili.com/video/BV1ct411n7oG/

(opens new window)

当然，现在也可以去找一些最新出的类似视频，我就不推荐了。

#

2.4.2 练手小Demo

看完视频，写一点单机小项目，比如我当时写的五指棋、聊天室（网络编程）、HTTP Server，一般这个视频里面也是有一些小项目的。

#

2.5 书籍推荐

#

2.5.1 《Head First Java》

Head First 系列书籍相对来说语言轻松，具有趣味性，如果对于没基础的同学，建议从这本书入手，比较深入浅出。

看完就能掌握 IO、集合、多线程、网络等关键知识点。

#

2.5.2 《Java核心技术》

这本书在豆瓣评分非常高，9.2 分。

书如其名，这本书就是讲解 Java 中核心的技术点，不过里面包含了一些过时的，不用学习的点，比如 Java 图形化界面 Swing，还有 applet 等彻底淘汰的技术。

我的建议如果是看完了 Head First Java，那么这本书你还是以翻目录的方式阅读，看其中关于多线程、集合、IO 等重点章节。

#

2.5.3 《Java编程思想》

豆瓣评分 9.0，这本书被誉为 Java 领域圣经，但是非常不建议新手去阅读。

中文版的翻译其实也不怎么好，读起来很多地方会比较难以理解。

这本书的作者还著有《C++编程思想》，所以这本书里有印象非常深刻的一个点就是，作者会将一些 Java 的特性和 C++ 做对比，比如 Java 的自动内存管理和 C++ 手动内存管理，C++ 的指针和 Java引用，C++ 支持多重继承、操作符重载等。

所以对于有 C++ 基础的读者来说，会特别有亲切感。

这本书对于不同阶段的 Javaer 都有不同的收获，但是真的不适合新手去读。

#

2.5.5 《阿里巴巴Java开发手册》

众所周知，阿里巴巴是以 Java 生态著称，这本书也不是讲解具体的 Java 知识，而是一本涵盖 Java 语言、异常处理、MySQL、工程架构等方面的规范。

建议大家学完 Java SE 部分后就去阅读以下这本书中编程规约的部分，能让大家对编程规范有一定的认识，其它部分可以后续学完 Java Web 再看。

#

三、Java Web

#

3.1 Java Web学习路线

大概路线就是：

Servlet、Tomcat -> JDBC - > HTTP 协议-> Spring MVC -> Spring 全家桶。

完成这部分学习后，可以完整的做一个 CRUD MIS 系统，然后去整个云服务器，部署到云上。

然后让你同学都来用用哈哈哈，相信你一定会体验到 CRUD Boy 的乐趣。

基本上就算体验了完整的 Java Web 开发流程，当然，这是野路子开发流程。

我当时是从 Java Web的基石: JSP、Servlet 开始学的，但是说实话，这两个技术确实非常老旧的。

但是我还是觉得值得学下，毕竟后来的 Spring MVC 这些东西都是为了解决 JSP、Servlet 这些在开发效率上的问题。

比如 Spring MVC 中的 DispatcherServlet 会拦截所有的请求，负责去查找有没有合适的处理器，一个前端控制器就可以，就不用去每个 Servlet 都去web.xml配置。

可以看下《Head First Servlets and JSP》这本书，虽然里面很多技术过时了，但是有时间还是可以挑一些感兴趣的章节，了解下 Java Web 的技术演变。

对于 Java Web 使用类的学习都是到how2j.cn这个网站：

基本上包含了 Java 学习的方方面面。

学 Java Web 是为了啥？

当然是 CRUD 啊，所以还需要学习数据库知识，这个时候会安装 MySQL，会用 JDBC 进行增删查改就可以了。

参考：https://www.zhihu.com/question/56110328/answer/370752381

#

三、Java深入

上面的 Java 基础和 Java Web 是实际编程中常用的东西，而这块就是面试常问的。

包括 JVM、多线程、锁，这块主要包含以下内容：

类加载机制、字节码执行机制、JVM内存模型

GC算法

线程池、锁、原子类、JUC包

这里就不展开讲了，基本上就是看书为主，看博客补充。

#

3.1 JVM

Java 虚拟机 的话推荐：《深入理解Java虚拟机》

#

3.2 并发

并发推荐: 《Java并发编程艺术》、《Java并发编程实战》

#

四、操作系统

这里其实和 C++ 的学习路劲是一致的，我就直接复制过来了。

#

4.1 操作系统核心内容

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 、网络是重点，这几块也是和编程关系最密切的。

#

4.2 推荐书籍

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》

这本书会涉及操作系统的各种核心概念、技术和实现，涵盖了多种操作系统类型，包括分布式操作系统、嵌入式操作系统和实时操作系统等。

#

《操作系统—精髓与设计原理》

#

《Operating Systems: Three Easy Pieces》

这本书主要分为三个方希：

Virtualization(虚拟化)

Concurrency(并发)

Persistence（持久化）

后来补充了 Security（安全）。

可以在线看英文版本：https://pages.cs.wisc.edu/~remzi/OSTEP/

(opens new window)

国内引入了中文版，并且有人民邮电出版社进行出版了，中文名：《操作系统导论》

中文在线翻译版可以参考一下这个：https://github.com/remzi-arpacidusseau/ostep-translations/tree/master/chinese

(opens new window)

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。

#

4.3 操作系统必备知识点

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

...

#

4.4 Linux 内核原理

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。

#

4.5 Linux 原理的书

#

《Linux内核设计与实现》

推荐看下这本书，讲得恰到好处，既讲清了内核实现的要点，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。

#

4.6 系统学习操作系统

关于系统学习操作系统，操作系统实验等内容可以看这篇文章: 如何系统学习操作系统?

#

五、计算机网络

#

5.1 计网核心内容

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

#

5.2 计网学习主线

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。

#

5.4 网络细节

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（大家可以在这直接评论区回答~）：

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》

这是一本介绍HTTP协议的图书，其中详细介绍了HTTP协议的基本概念、请求和响应消息格式、HTTP方法、状态码、Cookie和Session、缓存、安全性等方面的内容。 关于 HTTP、HTTPS 基本上看完这本书就算了解得比较透彻了~

#

5.5 从 URL 输入到页面展现到底发生什么

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，写完可以分享评论在这篇帖子下方~

#

六、网络编程

#

6.1 网络编程核心

网络编程的核心思想和学习路线和 C++ 学习路线也大体相同，只不过 C++ 和 Java 的侧重点不同。

比如 C++ 没有一个统一的网络库，基本都是裸写 socket，但是 Java 本身的网络 API 封装得已经很好了，再加上有 Netty 这种一统江湖的库。 学习路径也很清晰。

Socket API + 多线程 + 网络模型/IO模型 + IO复用 简单学下 Socket API，然后就是学习 Reactor、Proactor 模式，理解下为什么需要这个，适用什么场景就好了。

能自己写一个最好，不想写，可以直接去学习 Netty。

其实网络编程的重点不是在 socket 本身，而是线程模型 + IO模型，所以本质上还是要回到多线程。

网络编程核心的一些点：

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）

#

6.2 很少用到为什么还要学？

实际工作中，除了做 Infra 开发，很少会涉及网络编程的，现在几乎都是做 Web 开发，各种框架已经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，学完了，可以去写一些网络应用，比如 P2P 下载，仿QQ聊天室、文件传输等。

#

6.3 推荐的书

#

6.3.1 《Unix网络编程》& 《Unix环境高级编程》

这两本书简称UNP和APUE，曾被称为鹅厂后台开发敲门砖~

有趣的是，我来鹅厂之后，确实发现很多人的工位高频出现这两本书，不过一般是在垫显示器~哈哈哈

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，但是不适合挨着挨着一页页看。

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。

但是里面有一些章节是值得细看的，比如 UNP 里关于 IO 模型的介绍。

#

6.3.2《Linux高性能服务器编程》

这本书比较推荐，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。

PS：不过这本书中内容大多是用 C/C++ 演示的，但是其中的网络编程思想可以复用到 Java。

#

6.4 网络编程项目

学完网络编程就可以写点小项目练手了，主要是熟悉 Java Socket 编程API，以及 IO 模型，这里列举几个项目：

#

6.4.1 HTTP 服务器（WebServer）

这个似乎成了 Linux C/C++ 人手一个的项目了。 但是 Java 同学也可以做这个来练习网络编程，这个项目本身烂大街不重要，关键的是通过这个项目我们可以学到网络编程、服务器开发中的一些重要思想。

这里推荐两个做为参考（C++写的，不过思想都是共通的）：

https://github.com/imarvinle/WebServer

(opens new window)： 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

https://github.com/qinguoyi/TinyWebServer: 这是 Github 上Star 比较多的项目 https://www.nowcoder.com/study/live/504

(opens new window)： 这是牛客网上免费的视频课程，讲解 WebServer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。

#

6.4.2 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？

另外这部分可以重点学习一点 Netty 这个框架的设计和使用。

#

6.4.3 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

比较有名的 RPC 库可以学习下设计思想：

gRPC Java https://github.com/grpc/grpc-java 由 Google 开源的高性能 RPC 框架，支持多种编程语言，包括 Java。它使用 Protocol Buffers 作为数据传输格式，提供了同步和异步调用的支持。

Apache Dubbo https://github.com/apache/dubbo 阿里巴巴开源的高性能 RPC 框架，支持多种协议和序列化方式，包括 Java。它提供了丰富的功能，如负载均衡、服务治理、容错等。

Apache Thrift https://github.com/apache/thrift 由 Facebook 开源的跨语言 RPC 框架，支持多种编程语言和传输协议，包括 Java。它提供了代码生成工具，可以根据 IDL 文件自动生成客户端和服务端代码。

#

6.4.4 网络聊天室（类似QQ）

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。