操作系统超详细系统学习路线（2024年最新）

这篇文章分享一下如何学习操作系统这个话题。

《操作系统原理》一般是针对 CS 专业大三学生开设的一门专业基础课程。

任何计算机开启的第一个操作就是加载对应的 OS，而所有的应用软件都是基于 OS 之上。

程序员写的代码也都是跑在操作系统之上的，所以了解操作系统的原理以及它提供的接口（Shell、系统API）对于我们编程有很大帮助。

所以操作系统非常重要！

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、前置知识

要学习操作系统，最好掌握一些 CS 的基础知识和编程经验，以下是一些可能需要的前置知识（非必须）：

#

1.1 编程语言

至少掌握一种编程语言，比如 C、C++ 或者 Python。操作系统的开发或者上层应用开发都依赖编程语言，强烈推荐 C 语言，这是目前开发操作系统的主流语言。

#

1.2 数据结构与算法

操作系统设计的许多关键部分都涉及到了数据结构和算法的运用，例如进程管理、内存管理、文件系统等。所以最好掌握一些基本的数据结构和算法，例如链表、栈、队列、排序算法等。

#

1.3 计算机体系结构/组成原理

要理解操作系统，需要对计算机的底层结构有一定的了解，例如 CPU、内存、存储器、输入输出设备等。没有这方面基础的话，也可以边学操作系统边补齐这些概念~

#

二、明确操作系统学习目标

操作系统发展到今天，已经非常复杂了，像 Windows、Linux 任意一个都是几千万行代码级别，想靠个人完全搞懂，几乎是不可能的了。

所以需要先明确一下我们学习的目的，不同的学习目的，适用不同的方法，这里我仅仅列一下作为一名开发人员，我们掌握 OS 的几种目标：

第一层: 知晓OS基本概念

第二层: 深入理解OS核心原理

第三层: 动手实现操作系统 & Linux内核原理

一般来讲如果仅仅是面向校招、学习、面试的话，一般第二层就足够了。

#

2.1、知晓OS基本概念

了解操作系统的基本概念和组成部分。例如，进程和线程的概念、进程调度算法、内存管理的基本原理、文件系统的组成等等。

可以通过阅读操作系统的教材或者公开课来达到这一层次。

#

2.2、深入理解OS核心原理

理解OS基本原理:

在1的基础上，结合 Linux 等实际系统设计来进一步深入学习，比如进程与线程，了解 Linux 系统中是如何实现的，task\_struct 等等。

另外完成与 OS 相关的课程练习题，可进一步理解 OS 基本原理，练习题可以参考学堂在线清华OS课程的练习题：

#

2.3、动手实现操作系统 & Linux内核原理

掌握 OS 部分核心功能实现（细节）：在 2 的基础上，能够通过编程完成类似 xv6 的几个 lab ，

比如 xv6 就会涉及以下模块：

Booting a PC

Memory Management

Multitasking

File system, Spawn and Shell

Network Driver

每个模块代码量都不会很多，但是麻雀虽小，五脏俱全，通过自己动手实现 thread、spinlock、semaphore 这些东西，会让你理解更加深刻。

当然，在往上走就是去研究最新 Linux Kernel，研究一些子模块了，比如虚拟内存、文件系统等等，有些还需要阅读最新的 System 领域论文。

最新的 Linux Kernel 里有很多新的算法、研究结果不断的加入，当然，这就超出了我所能掌握的范畴了哈哈。

#

三、学习方法

比较推荐的学习方法是，视频 + 教材 组合的形式，因为单单看书，比较枯燥乏味，而只看视频呢，又容易丢失大量细节知识，所以最好结合着来。 经典教材有：

#

3.1 书籍推荐

《操作系统概念》（Operating System Concepts）

《操作系统—精髓与设计原理》

《现代操作系统》（Modern Operating Systems）

《操作系统：原理与实现》（上交大陈波教授团队出版的）

《Operating Systems: Three Easy Pieces》

这几本书任选一本即可，基本都涵盖了操作系统的核心知识，只是侧重点略有不同。

一定要我推荐的话，我选择：

#

3.2 强烈推荐书籍

《操作系统：原理与实现》（上交大陈波教授团队出版的）

主要原因是 《操作系统：原理与实现》 这本书是出自国内 TOP 的研究系统的团队：IPADS组。

另外就是书里很多内容都很“现代”，不会像有些操作系统书籍里有挺多过时的内容。

区别于所有本人已知的教材（包括我现在用的OSTEP、很出名的MOS等），这本书最大的特点就是挑战了传统操作系统教材的权威，果断地删掉了和“现代”操作系统关系比较小的部分，真正把“现代”两个字体现得淋漓尽致。跟Tanenbaum的“伪现代”比，这个是“真现代”——以如日中天的AArch64作为底层机制，直通操作系统研究前沿，现代到有些部分标上了辣眼睛的“辣椒”，都是货真价实的干货。（节选自南大OS老师蒋炎岩：“世界上最牛气的操作系统教材是什么？

(opens new window)”。）

《Operating Systems: Three Easy Pieces》

另外一本就是 《Operating Systems: Three Easy Pieces》，简称 OSTEP，阅读体验比较好，讨论问题由浅入深。

而书名也是很有意思，Three Easy Piece是为了致敬费曼的关于物理学的书籍：《Six Easy Pieces: Essentials Of Physics Explained By Its Most Brilliant Teacher》。

用作者的话说，操作系统只有物理学一半难，那就叫 《Three Easy Pieces》好了。

Three 也是指代本书三个部分：

虚拟化（Virtualization）

并发（Concurrency）

持久化（Persistence）

#

3.3 视频推荐

《清华-操作系统原理》

由向勇、陈渝老师讲授，在学堂在线上可以参与学习：

学堂在线操作系统

(opens new window) 也可以选择在 B 站看，B站弹幕会有趣很多，评论区也有很多不错的建议和资料:

B站-清华-操作系统

(opens new window)

内容非常完善，涵盖了现代操作系统的所有核心知识，并且会结合具体的教学操作系统(ucore) 来讲。

该视频配套的教材可以选择《现代操作系统》或者是《操作系统：精髓与设计原理》。

#

四、动手写操作系统

#

4.1 MIT 6.828

给大家推荐一下6.828： MIT6.828： Operating System Engineering

(opens new window)

官网主页：https://pdos.csail.mit.edu/6.828/2020/index.html

(opens new window)

在官网，包含了Schedule、Labs、xv6-book 等等。

尤其是 xv6-lab 包含了操作系统中最为核心的模块：

每个模块都搭好了框架，留下最为核心的部分给我们去实现，你可以把这个看做一种「完形填空」：

System Call、Page tables

Traps

Lazy allocation、Copy On-Write

Multithreading、Lock

File System、mmpa

network driver

如果想看视频的话可以去 Youtube 或者 B站：

MIT 6.828: Operating System Engineering

(opens new window)

#

4.2 MIT 6.S081: Operating System Engineering

这是 MIT 基于 6.828 重新出的一门课，会更加的简单一些，不过依然会有类似 xv6 的实验部分，只是设计上更适合基础薄弱一点的同学。 多一些文献阅读部分。

#

4.3 uCore

如果你想找一个国内的类似 xv6 的 lab，那么这里推荐清华大学 ucore OS。

它其实也是起源于MIT CSAIL PDOS课题组开发的 xv6 & jos、哈佛大学开发的 OS161教学操作系统、以及Linux-2.4 内核。

然后清华的老师在这个基础上，增加、调整了一些代码，为清华大学 OS 课程设计的一个lab。

另外，这个课程的很多资料、文档都是中文的，还有清华老师的配套视频课程。

清华大学-操作系统

(opens new window)：https://www.xuetangx.com/course/THU08091000267/5883104

#

5.4 自制OS

另外，如果你还想自己完全从头写一个（模仿） mini os，那么你需要一些前置依赖知识，否则估计会被很多硬件、汇编知识劝退。

筑基内功-- 需要学习计算机原理、C语言（需要熟练运用）、汇编语言、简单数据结构知识

工欲善其事，必先利其器，掌握以下Linux 下常用命令行：

pwd、cd、find等，1-2 款趁手的编辑器，推荐掌握 vim 的基本操作 gcc、gdb、ld、make等编译构建链、objdump、nm、readif、dd 等 ELF 文件分析、烧录工具。

#

5.4.1 《操作系统真象还原》

这里只推荐一本我看过的书《操作系统真象还原》，作者实在太好了，所有需要前置依赖的部分他都会简单的写个教程，基本上照着这本书就能撸一个属于自己的 os，不过缺点也是书太厚了，太细致了。

我当时在图书馆苦哈哈的啃了一个多月，才看到五百多页，而整本书有八百多页~

#

五、Linux内核原理

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。

这本书重点关注以下几章：

「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。

#

六、校招重点

操作系统这门课，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

了解和熟悉常用的系统调用已经 System API

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。

#

6.1 必备知识

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

...

#

6.2 理论结合实践

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。

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。

所以还是推荐我上面提到的 《LInux内核设计与实现》

#

END

好了，操作系统的学习路线就到这里了，后面我会持续更新操作系统每一个具体部分的知识，争取出一份快速入门的文档。

作者: 编程指北

链接: https://csguide.cn/roadmap/basic/os.html#%E5%85%AD%E3%80%81%E6%A0%A1%E6%8B%9B%E9%87%8D%E7%82%B9

来源: https://csguide.cn

著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。