计算机专业如何系统自学？

写公众号以来有很多同学向我提问，问题基本都是大同小异，核心就是刚转到 CS 或者刚上大一不知道怎么学编程，还在背代码、背语句......

所以今天聊聊计算机专业的学习以及一个升级打怪路线。

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、抓住主线，突出重点

我一直的观点就是，在具备基础之后，学习任何新东西，都要抓住主线，突出重点。

对于关键理论的学习，要集中精力，速战速决。

而旁枝末节和非本质性的知识内容，完全可以留给实践去零敲碎打。

战线拉得越长学习效果越不好。

#

1.1 什么是主线?

就是核心内容，就是不会这些东西，你就不能算懂XX，比如 C++，我觉得核心就是 C 系的语法（if for那种）+ 类、继承、虚函数 + STL + RAII

如果 C++11，那就还要加上 auto、右值引用、lambda 这些。

其它类似怎么输入输出、怎么读取文件、map、vector 有哪些方法都属于支线，用到的时候再去学也无妨。

原因很简单，对于重点知识，只有集中学习其理论，才能确保体系性、连贯性、正确性，而对于那些旁枝末节，只有边干边学能够让你了解它们的真实价值是大是小，才能让你印象更加深刻。

比如 Unix编程中各种系统API，就属于细枝末节，对于 Unix 的各个内核子模块的整体实现、IPC 通信机制、IO 模型等才是重点。

（说个题外话，最近在制作一个《CSGuide》，会包含学习资料、路线、课程等等内容，到时候弄好了会同步到 Github 去的：

上面说的是具体某个技术的核心，那么 CS 学习的核心有哪些呢？

#

1.2 CS学习核心课程

大概是下面这些：

高级语言程序设计、数据结构与算法、电路与电子技术、集合论与图论、代数与逻辑、数字逻辑、计算机组成原理、操作系统原理、数据库原理、编译原理、软件工程、计算机网络。

学习计算机最核心的方法，也是小北一路走来总结的，按照学长推荐的这个路线走，大学毕业成为 Offer 收割机，去 BAT TMD 这些都没啥问题:

入门看视频 + 不断编码 + 深入看源码和经典大黑书 + 做国外计算机硬核Lab

大黑书有哪些？

可以看看这篇高赞文章： 计算机必读经典书籍

(opens new window)

学习计算机一定不要想着速成，要下苦功夫，去啃那些经典书和源码，然后自己动手写代码。

看书方法可以参考这篇文章：

如何阅读计算机类书籍？

(opens new window)

至于科班的课程，我们可以看下网易云课堂这个计算机科班的课程体系，虽然不是每个学校都一样，但是大同小异:

比如计组、汇编、体系结构等课程会让我们明白计算机的构成、运行机制，以及典型的冯诺依曼结构，也会对于内存和指针这样两块在任何编程语言中都是极其重要的概念理解非常深入（不要扯Java这类语言没指针，你完全可以把引用当做受限指针）。

另外，像数字逻辑、模电这样的课程主要是让你明白数字：浮点数、整数都是如何在计算机内部表示和存储的，以及一些门电路的知识。

操作系统则让你明白「一个程序是如何跑起来的」，一个它是如何编译、链接、加载、运行的，在这一整个过程几乎会涉及到操作系统的所有环节，学了这个你就会明白 synchronized 关键字在OS层面到底是如何保证的。

而计算机网络则是让你明白「一个数据包是如何从一台主机发送到另外一台主机的」，这里面涉及到路由算法、差错检测、寻址、重试策略等等，包括 TCP/IP整个协议栈

编译原理则是让你明白高级语言都是如何被编译程序识别、转换为另外一种语言的，我认为编译器的本质就是做转换。

很多人觉得平常用不到编译原理，其实，，，真的用不到哈哈，但是在一些 DSL 方面还是会有用武之地的。

#

二、计算机科班必修课程

#

2.1 CS核心课程

把操作系统、组成原理、系统编程这些学了，认认真真的把计算机基础打扎实了，那你其实就是一个「计算机科班学生了」。

接下来说下我认为对一个「计算机知识体系完整的毕业生」最重要的课程，你至少得把下面这些课 都学了：

1、 编程语言：C/C++ （学好了这两门，基本其它编程语言一周就上手了）

2、《系统级编程》(我们教材是CSAPP，这是我本科上过最值的课！System Programing)

3、《数据结构与算法》

4、《组成原理和体系结构》

5、《操作系统》

6、《编译原理》

7、《计算机网络》

8、《数学课》：线代、离散、高数

首先，送大家一句话~：

万丈高楼平地起，勿在浮沙筑高台。

所以一定要在大学期间把基础打牢固，整扎实，这是科班的核心竞争力！

而且非科班同学学完这些课，也可以说是科班了！

#

2.2 那么怎么学？

看书 + 看视频 + 实践

接下来推荐一些我觉得特别好的资料，尽量保持精简：

#

2.3 编程语言：C/C++

没有比C语言更适合用来理解计算机系统了, 科班学生一定要学好 C语言。

推荐：《C程序设计语言》、《C与指针》、《C++ Primer》、《C++对象模型》、《Effective C++ 》

具体参考：

如何系统地学习 C++ 语言?

(opens new window)

如何系统学习 C 语言？

(opens new window)

#

2.4 系统级编程

不用说肯定推荐《深入理解计算机系统》

其实它的简介和第一章《计算机系统漫游》已经写得非常清楚了：

从一个简单的hello world程序在计算机上的执行过程：

预处理->编译->汇编->链接->可执行文件->装载->数据流->屏幕输出显示

汇总成一句：信息 = 位+上下文。

另外还有一本就是《程序员自我修养》+ 《Unix环境编程》。

#

2.5 数据结构与算法

《算法第四版》、《算法导论》、《剑指offer》

不过这里说一下，算法导论不建议新手入门使用，容易被劝退，可以作为进阶阅读使用。

具体参考：

如何系统学习数据结构与算法？

(opens new window)

#

2.6 组成原理与体系结构

《计算机组成与设计:软硬件接口》、《编码》、《CSAPP》

#

2.7 操作系统

操作系统我推荐 看书 + 做lab的方式，比如 MIT6.828 xv6

书可以看《现代操作系统》，如果你想学个os，可以看《操作系统真象还原》

我当时应该是看了不少网课，然后自己跟着教程”抄了“一个 mini os。

学习操作系统一定不要去死看书，最好跟着那种有配套实验完成一个os的课程或者书。

就像这种：

img

具体参考：

如何系统学习操作系统

(opens new window)

#

2.8 计算机网络

推荐《计算机网络自顶向下》、《TCP/IP详解》

具体参考：

如何系统学习计算机网络

(opens new window)

#

2.9 编译原理

直接看龙书（就叫《编译原理》）就好了，想写脚本可以看看《自制脚本语言》

说实话，把上面这几门课学好了，国内大厂offer几乎随便拿了。

但是很多人忽视了基础，往往在大学最该打基础的时候去追求所谓的新技术。

所以不如谈谈到底「如何才能成为一个计算机知识体系完整的毕业生」吧？

#

三、计算机编程领域知识

在这里，我粗浅的把计算机编程领域的知识分为三个部分：

基础知识

特定领域知识

框架和开发技能

具体可以看下这个文章：

如何成为一个计算机知识体系完整的毕业生？

(opens new window)

我推荐大家花在「基础 : 领域知识 : 技能」 接近 7 : 2 : 1。

这也是我推荐你在大学期间分配学习时间的比例，至少学习基础知识的时间不少于 50%，当然，这些东西你都学完了那可以去找找感兴趣的方向专研一下。

不要大一、大二一上来就扎进 Java Web、Python 爬虫这种东西，这些可以学，但不是重点。

那么如何检验学得如何呢？

想必你一定听说这个计网面试题：

“从 URL 输入到页面展现到底发生什么？“

这个问题换个表达就是「一个数据包是如何发送到另外一台电脑的」，如果你能完整的说出整个过程，那么计网你一定是学懂了！这就是为啥面试这么喜欢问这个问题的原因。

那么我们依葫芦画瓢提一个问题

“从代码被写下到程序运行起来到底发生了什么？”

这个问题回答得越详细越好，基本上能说清，你就理解了编译原理、操作系统、组成原理这三座大三。

推荐阅读：

四大基础课网课：CS经典课程，起飞！

作者: 编程指北

链接: https://csguide.cn/roadmap/method/cs\_self\_learning.html#%E4%B8%89%E3%80%81%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%BC%96%E7%A8%8B%E9%A2%86%E5%9F%9F%E7%9F%A5%E8%AF%86

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。