如何成为一个具备系统知识与技能体系的计算机专业学生?

一共分为八个方面来叙述这个问题，包含计算机专业能力提升路线，能力锻炼和价值提升，目标在于提升自己成为一个具备扎实技术功底，有足够丰富实践及方向明确的计算机行业从业者。

1、技术学习

大学期间对于技术学习大概分为两个类别，一个是围绕计算机、数学学科知识的学习，一部分是编程技能学习。

这两部分，前者属于计算机学科的基石，基本所有的编程，乃至计算机技术都是围绕前者建立的，想要在计算机方向深入，必不可少，接下来先就学科知识展开讲讲。

(1) 学科知识

学科知识的学习可以重点放在大一大二，练好内功，才能融会贯通更多招式。

① 计算机组成原理　难度：　几颗星

计算机是由 CPU、内存、显示器这些设备组成的硬件，我们所书写的程序，也就是我们的软件。而软件和硬件之间，就需要一个渠道去沟通和交互，那么这个渠道就是“计算机组成原理”。了解了这个渠道，那么无论是书写操作系统、编译器这种硬核代码，还是 Web 应用、手机 App 这种应用代码，都可以心里有底。

除此之外，组成原理是计算机其他核心课程的一个引导课。学习完组成原理，向下可以继续学习数字电路相关的课程，向上可以学习操作系统这些核心课程。

如果想要深入理解，甚至设计一台自己的计算机，体系结构也是必不可少的一门课。

1) 学习建议：

计算机组成原理作为专业基础课的第一课，

第一点：杂。

组成原理的内容多，且杂乱无章。概念及其广，并且每个概念的信息量也非常大。

第二：晦涩。

组成原理中的很多概念展开来基本都可以当作计算机学科中的一门核心内容。比如我们说，我们所写的高级语言，如何变成计算机可以执行的机器码？如果把这个问题展开并且输入那就会成文编译原理这一门学科。第三不能学以致用，学习这些无法和自己的日常开发联系起来。

对于以后想从事嵌入式系统、操作系统设计等和硬件结合比较紧密的工作的同学，建议所有的内容都要学好。

而如果以后想从事软件相关工作的同学也要学好存储器、输入输出系统、计算机的运算方法、指令系统、CPU 的结构和功能等章节的内容，因为做大型系统软件会涉及到调优的问题，如果只懂软件，而不知道硬件在怎么运行的话，调优会受到限制，软件无法发挥硬件的最大优势。

课程推荐：

https://www.icourse163.org/course/UESTC-1001543002

https://www.icourse163.org/course/HIT-309001#/info

https://www.icourse163.org/course/HIT-1001527001#/info

书：《计算机组成》《计算机科学导论》

② 汇编与编译原理，先讲汇编

1) 学习汇编能脱离编译器甚至编程语言的限制，对于语法总结，对于了解编程语言本质，对于解决问题的能力帮助都很大。

汇编语言的作用有很多，我们站在高级语言的角度来说，如果我们懂汇编，可以看懂每一行代码编译器生成的汇编代码，我们就知道这行代码计算机在做什么事情，就可以从本质上了解编程。

2) 学习建议：

学汇编需要一定的微机硬件基础知识，因为汇编过于底层，更加接近机器语言，所以很多东西要从机器语言的方面去理解。

学习汇编之前最好先学习一门高级语言（比如 C++、Java），因为高级语言更贴近人，当我们在学习汇编遇到问题的时候，可以跳出汇编从高级

语言的角度看问题，不会被局限起来

汇编语言是更接近于机器语言的编程语言，如果我们将机器语言看成是操作计算机的本质语言的话，那么汇编语言是最贴近于本质的一个语言。

学习汇编的目的不是非要用汇编去挣钱吃饭。而是因为汇编语言对于一个编程人员所应该具备的基本素质的培养和形成的意义是非常重大的。

课程推荐：

https://www.icourse163.org/course/UESTC-1002047009

书：《编译原理》（龙书）

③ 操作系统

1)

计算机分为硬件系统和软件系统，操作系统是软件系统及其重要的一部分。而学好这门课程也为我们的未来提供了更多的可能性和上升空间。操作系统庞大且全面，我们可以在其中领会到各种数据结构的巧妙应用，各种解决问题的奇思妙想，他为许多的工程问题提供了非常经典并且极其有效的解决思路。学好这门课程，毫无疑问裨益非凡。学习操作系统也能

够对程序运行和资源调配更清楚。

2) 学习建议：

操作系统也是一个极其抽象的东西，重在理解。

不行就先背过。先理清楚操作系统有哪些模块，每个模块都是干什么的，每个知识点，都去好好了解一下他的背景和意义。

系统是人设计的，里面每一个设计的背后都有当时设计者的思考，所以你遇到的每一个设计细节都有为什么这么设计的理由。要注意看书和做题并用。

课程推荐：

https://www.icourse163.org/course/NJTU-1003245001

https://www.icourse163.org/course/NJTU-1003245001

实验：

ttps://www.lanqiao.cn/search/?type=course&search=%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F&page_size=15

书：《深入理解计算机系统》（CSAPP）

《现代操作系统》

④ 计算机网络

1) 互联网时代，所有的流量，传输都是依靠网络，无论是应用开发还是网络安全，都需要了解网络。

作为一个编程语言学习者，当我们在学习一门编程语言的时候，可能学到一些基本语法，面向对象等等。这个时候我们总感觉计算机网络完成不重要，反正我写代码也不设计网络传输，平时的工作也只是 CRUD，没什么用武之地。当我们接触到一些具体网络技术时，例如 Nginx，或者 docker 等等我们就会发现网络基础的重要性。想要深入学习编程，计算机网络还是比较重要的一环。

2) 学习建议

计算机网络实质上是关于网络知识的导论，正因为是导论，所以会显得内容非常杂乱，那么建议在学习中注意以下几点：

1、理解计算机网络的组成

2、学习网络协议

3、了解网络分类的重要性

4、一定要学好 TCP/IP 协议（重点）

课程：

https://www.icourse163.org/course/HIT-154005

实验：

https://www.icourse163.org/course/BUAA-1002744004

书：

《图解 HTTP》

《图解 TCP/TP》

《计算机网络：自顶向下方法》

编译原理

3) 这门课多说两句，目前国内计算机学习大趋势，是迎合互联网公司招聘需求为主，对于编译器的研究和操作系统的研究基本缺失，目前这块还处于被老外卡脖子的阶段，这门课是从使用语言到创造语言的必经之路。就算不能创造，有所了解，对日后编程的学习和研究也有很大帮助

众所周知，编译技术是计算机核心之一。技术驱动的互联网巨头们都有自己的语言和生态，普遍具备了二次编程能力（最常见的就是 Office）。可见编译技术的重要性。

可能有一些初学者或者程序员会问：我不可能自己去写一门新的语言，还有必要学习编译原理么？

编译技术不仅仅是屠龙技，在实际工作和开发中我们会经常碰到需要编译技术的场景。比如 Java 现在主流框架 Spring 中有对于注解的支持和字

节码动态生成属于编译技术。如果我们想要写这些工具或者写更符合领域要求的工具，也需要编译技术。

除去应用场景，他对我们自身的竞争力和对于计算机技术的理解也有很大的帮助。我们都知道有难度的面试都涉及到了底层，因为我们只有了解底层机制才能更深入的思考问题，而编译原理可以让我们从语法、代码优化、代码与硬件结合等多个方面去看待问题，加深对于计算技术的理解。

4) 学习建议

编译原理并非是可以随随便便入门的一个学科，他的学习和实践通常基于对计算机编译过程，计算机基本工作原理，甚至有一定的数学知识的积累。

我们在看编译原理之前，至少把 C 语言指针的思想搞清楚，学习离散数学，学习了数据结构。然后要理论和实践的结合，争取自己能写出一版玩具编译器。

课程：

https://www.bilibili.com/video/BV1zW411t7YE?from=search&seid=11631273910877958758

https://www.icourse163.org/course/UESTC-1002047009

书：《编译原理》（龙书）

⑤ 数据结构与算法

1) 所有业务的实现都是算法，所有数据的存储都有结构。程序=算法+数据结构，某种意义上来说，这是最粗略区分码农和程序员的一门课。

同一个项目、或者框架，架构差不多，设计思路差不多，实现的功能差不多。为什么有的人做出来的框架 Bug 也多，性能也一般，可扩展和可玩性也不好， 只能在自己的公司仅有的几个项目里面用一下。而有的人做出来的就可以开源出来给很多人用，甚至被 Apache 收录。这么大的差距形成的原因，就是因为细节。这个细节就包括，你的内存是否节省，你的数据存取是不是足够的高效，你的算法是不是足够优化，等等等等，这些结合起来决定了一个框架是否足够优秀。如果你不懂数据结构与算法，不懂复杂度分析，那就只能写“凑合能用”的代码。

2) 学习建议

1、至少要掌握一门编程语言，以此为基础进行学习

2、要理解透彻数据结构的思想，然后一定要能够独立完成代码的实现。

3、多刷一些题目，通过一些难题去领会到数据结构的魅力。

4、做好规划，把每种结构的基本原理和应用列清楚。

5、简单点说，动手多练。

6、再提一句，一定要勤奋刷题

刷题：

牛客

力扣

书（电子档）：

《大话数据结构》

《数据结构与算法分析》

课：https://www.icourse163.org/course/XMU-1206002801

ACM 题目解析：

（https://www.nowcoder.com/ta/acm-solutions）

⑥ 总结一下，以上学科内容为“内功”，内功不一定能让你多拿多少工资，但是能够帮助建议对于计算机学科比较全面的认知。

我们学的不是编程语言也不不是编程，是整个计算机学科。

同时能够打造个人的适配性和稀缺性。 如果对技术深入兴趣不是太大， 或只是想以技术入行再找适合自己的定位的话，以上内容除数据结构和基础网络知识外，其他的都可以忽略。

(2) 编程技能

编程技能是指用于实现产品开发的具体技能、编程语言等，例如 Java、C++

等。

编程语言的划分有脚本语言、标记语言、解释型语言等，简单了解下即可，不用纠结概念。

脚本语言，他缩短了传统的编译的过程，他不需要编译器，只需要解释器，就好像一个专用的软件，你想要跟我进行交互就需要特定的一些操作。而这些操作的集合就是解释器，进行的操作就是脚本语言。

编程语言，准确的说是编译语言，需要由编译器将计算机翻译成计算机所认识的二进制代码的工具。

脚本语言相对而言会更加容易学习，但是不够全面，缺乏系统性，而且语言较为散漫。而高级语言相对比较难学，但是规则性较强，并且可读性相对较强，适合。

随着技术的发展，脚本语言变得越来越强大，和编程语言的界限越来越模糊。比如 Python，因为他足够强大。

1、编程语言

① C 语言　可以说作为当今所有流行程序语言的老祖宗，对老祖宗有所了解，对于学习他的子孙后代等其他语言（C#，Java，PHP....）等等来说都有很大的帮助。而且又了这个基础，我们就有了选择的机会，是沿着老祖宗的道路继续发扬光大，还是根据自己的兴趣或者适应社会的各个需求灵活的进入技术领域。

② Java Java 一开始设计之初是在 C 的基础上做一个扩展，当 Java 的使用者越来越多的时候，才越来越偏向新手。发展到现在，相对 C++，他是更贴近新手的一门语言。

③ C++ 可以将 C++看作是 C 语言的加强。他完全兼容 C 语言的特性。但是就

难度来说，他的学期周期会更长。

2、脚本语言

Python：随着人工智能，python 越来越火热，同时也越来越强大。其在数据分析， 测试运维，快速开发等都有着很强大的作用。

这里我想说的是,有些人会觉得我已经会了 XXX 语言的,还有必要去学习 XXX 么？暂且不说语言是一通百通的，当我们学会一门语言（尤其是编程语言的时候），再去学其他语言就很快乐。要知道，语言就是我们的工具，就好像砸钉子就用锤子，拧螺丝就是用螺丝刀。我们学习一门语言是用来解决实际的问题，解决某个问题用什么最好就用什么，而不是单纯为了学某个语言而去学习。

对于新人而言，建议以 C 或者 Java 作为自己的第一门编程语言 ，如果想要深入，建议在 C++或者 Java 中选择一门，相较其他语言，这两门语言不仅生态成熟，更具备软件工程设计的概念，更便于以此掌握编程的学习方法和对编程思维，同时这两门语言也是企业应用的技术常青树。

这里的学习路线过长，资源也较多，就不贴出来了，有兴趣的可以私信。

(3) 数学

高等数学：

高等数学是基础，虽然说，一般的程序是很难用上高等数学的。但是如果你进入算法、数据库或者其他很专业的东西。它的基础作用就很明显了。个人认为高等数学的目的：1、培养思维（另外还有一门学科叫数学分析，对于数学逻辑的建立也有一定的帮助） 2、算法分析 3、程序本身（如果做图像处理，高数就尤为重

要）。

离散数学：

本科阶段离散数学四大块：逻辑学、集合论、图论、代数结构。计算机科学中普遍采用了离散数学中的一些基本概念、基本思想、基本方法，使得计算机科学趋于完善和成熟。他在计算机体系结构、数据结构、数据库、编译原理、人工智能中都有着很广泛的应用。

线性代数：

矩阵计算在计算机科学中也有着独一无二的地位，在游戏开发、计算机图形学、计算机视觉、机器学习等方面都有着巨大的作用。最著名的用途之一就是谷歌的 Pagerank 算法。

(4) 物理

一些领域会涉及到一起物理概念，这要具体和某一个东西相关联，比如机器人就需要人体力学等物理相关概念，在学习相应领域知识时简单了解即可。

2、竞赛

(1) 蓝桥杯

① 蓝桥杯在大赛事中不算很难得，所以才有了类似暴力杯这种名号（当然这个名号是针对省赛的），然后刷历年的真题，甚至题目未必都用得到算法的知识，所以省赛没有想象中那么难，勤奋一点就好。

② 国赛 如果进入国赛，其实你会发现题目类型很清晰，而且最好有一定的数学功底，有几年数学类型的题目还是较多的。把历年的题目刷好。

蓝桥杯总体来讲不是很难，独立做出各种类型的题目，做够 500 道基本都比

较稳妥。做题量够冲国一还是希望极大的。

(2) PAT

① 乙级，中文，题目简单，就是简单模拟，字符串处理，散列 Hash，排序二分链表一类的。

② 甲级：英文 在乙级的基础上加入了数据结构

③ 顶级：英文，ACM 那一套

PAT 的含金量（甲级和顶级），说高不高，说低不低，在江浙沪可能作用会更大（知名度更高），其他地方可能没有太多的认知度（高分区还是会被各大企业瓜分的），如果是考研的话还是很有作用的（尤其浙大）。

(3) ACM

就省赛来说，要分强省或者弱省。就该赛事含金量来说，要从区预赛开始说起。虽然一直说区预赛的奖牌含金量逐年变低，但是就考研或者面试来说还是一个很好的加分项的。

(4) 大厂组织竞赛（全国微信小程序开发大赛）

华为软件精英挑战赛

阿里巴巴人工智能对抗算法竞赛

腾讯广告算法大赛等

3、实验室，项目，实习

(1) 实战项目经验

项目实战经验对于计算机专业的学生来说非常重要，更多的项目经验意味着在毕业的时候拥有更好的竞争力。一般来说，对于本科生而言，获取项目经验无非以下几

种途径：

1、参加学校老师的课题或者项目（重点大学的机会较多，普通大学能过加入的机会不多）

2、参加一些专业比赛，通过比赛往往能提高自己的实践经验，而且主要能够获得专业老师的指导

3、参加一些开源的项目，这个对于学生的技术要求相对会更高，需要学生对于主流技术有了一定的掌握以后，再去一些开源社区阅读开源系统的源代码。

4、对于本科生，可以在暑期投一些实习，对于研究生，是有专门的实习岗位的（一般研二以上，并且有时间的情况下），一周大概三天左右工作时间。

4、扩宽视野

整个计算机行业是个庞大的领域，在这个领域里技术是核心，但从商业价值和个人价值来讲，技术并不是全部，无论是为了建立对行业更全面的认知，还是能够让自己更全面的成长，抬头看天，扩宽视野对我们都非常重要。

(1) 了解互联网行业

除了技术本身层面，及时的了解互联网与行业发展方向也是必要的，有助于我们提前做好技术积累以及发现更适合自己的机会，也是为了不让自己盲目跟风，有自己的技术大局观，不断的精进提升。所以基于种种益处，我们都应该时刻关注行业发展，

关注技术： GitHub、掘金、ITEye、OSChina、SegmentFault

关于互联网的发展：可以关注下创业邦、36 氪、笔记侠等等

新技术的出现会推动互联网的发展速度，同样的，互联网的发展所创造出的场景会反过来倒逼技术的发展。

(2) 建立自己的产品概念

① 技术的存在主要是为了解决问，创造价值。就像一家公司的一个最小基本单元产品、运营、研发。我们每天不停码代码是为了干嘛？最终呈现的形态或是 app 或是小程序或是网站，最终是变成解决业务场景的一个产品。而且和研发打交道最多的应该就是产品经理了，我们不仅仅是个合格的开发者，我们也应该具备产品的思维。带着问题去思考，小到一个功能点出现的价值和意义，大到公司整个生态下这个产品存在的意义。锻炼自己的产品思维目的在于

其一更好的理解产品思维，提高工作效率，做好桥梁，得到认可。

二是让自己具备更强的就业竞争力，具备更多的可能性，做一个能开发里最懂产品的，产品里最懂开发的人，这种复合型人才是未来的趋势。

② 平时可以关注：人人都是产品经理　PMCAFF 梅花网等

(3) 职业规划

① 关于技术发展路线上文已经把主流的发展路径都梳理清楚了，这里我们更多的是放眼整个职业生涯，我们选择开发作为主线技能时，我们未来的职业规划的方向和方法应该是怎样的。

② 基本的策略是，当我们初期以开发进入互联网后，持续的运用自己的技能在公司中去创造价值，且不断提升自己的技术，在 2~5 年的过程中持续的聚焦这块长板，它会把我们带到另一个领域，那就是公司的管理层，当我们能达到这

个领域，看这个世界的视角和出发点会发生很大的变化，个人价值体现就不再是写几个功能模块儿了，而是在如何调动团队和资源去创造更大的价值，进而能让自己实现持续的影响力。

③ 当然除了技术本身外，可迁移能力的锻炼和学习是必要的

解决问题的能力[不仅仅是技术问题]，沟通能力、执行力、人格魅力、学习能力、信息收集筛查能力等等

④ 工作中宝贵的经验是需要沉淀积累的，成长过程中积累的人脉关系是需要维持的。这些都是我们在职业发展到中期，想要建立一些事业的关键助力。

这里也抛出一些我之前看到或者自己当年思考的一些问题，有助于对自己当前职业价值进行衡量：

我今天学习和成长了吗？哪怕一点点

我为了避免被机器取代，正在做怎样的努力？

我能在哪些公司/组织以什么方式找到工作？

我是否对某些人、对公司拥有一定的影响力？

我是否得到了应有的经济奖励，且自己也创造了经济价值？

当前的工作是否能让我感受到些成就感、幸福感？

(4) 职业规划

5、明确方向（就业、考研）

(1) 考研与否

① 对于考研也是初学者较为迷茫的问题，常见于计算机是否需要读研，xx 方向是否需要读研，当然也有临近毕业选择考研逃避就业的。

② 计算机读研的意义在于对专业知识的深入研究，以及提供就业跳板。前者如果

仅谈就业价值，常规的应用开发，在专业知识足够的情况下，CS 本科就已经足够，国内一线大厂也是本科居多，如果是一些类似人工智能等交叉学科，则读研更好一些。对于考研作为跳板，二本能考上 985 硕士，无论是学习、圈子、就业跳板都是很大的提升。

③ 大一大二对于是否读研可以不用着急，该学什么学什么，该参加竞赛参加竞赛，对于该抉择的时期，脑子里有货才能在 offer 和读研里做选择，而不是担心应届生校招资格和读研失败与否的得失。

(2) 就业方向

① 计算机发展到现在，仅技术的细分方向就已经很多了，笼统的前端、后端，细分的各语言，方向性的开发方向、网络方向、算法方向等，本科也好，读研也好，总要有个大体的方向才行。

② 在学习初期，自己没有建立对行业整体认知，或是没有找到真正喜欢的方向时，可以先按照上面的学习建议来，计算机技术说到底还是在这些内容基础上进行发展。内功修好了，才能有更多的选择，但是在学习过程中一定要记得抬头多看看世界，找到自己喜欢，擅长的方向深入下去。

③ 对于就业选择，刚毕业可以选择 996，但如果不是特别缺钱，尽量不要选择长期 996，996 牺牲的除了健康，还有自己思考和成长的时间，再好的鸡汤也没办法让连续工作 12 小时，还要花两个小时在通勤路上的人保持学习的动力，**这种牺牲会让你输掉未来。**

6、信息检索及筛选能力

(1) 善用互联网

① 互联网的存在解决了人与人，信息的传递，为主动学习的人提供了极大的便利性。很多产品为了更好地体验，有很多无微不至的服务，做产品有条准则，就是把用户都当成傻福。所以很多人在这种环境中养成了伸手党的习惯。但成长从来都是内驱的事，从初学开始就善用搜索引擎，用各种平台，培养自己收集信息，整理资源的能力非常重要。

7、自我提升

(1) 突破环境的桎梏

① 没有贫穷的环境，只有贫穷的思想。大学刚毕业那会儿，我总想，要是我能遇见一个技术牛逼的大佬就好了，我就能跟着他技术上一路突飞猛进，快速跃迁。现在看来其实很幼稚，真的是技术大佬凭什么会带你？就算你遇到了，大佬的主要关注点根本不在这里，这是可遇不可求的机会。假设你 2 年后能遇见这个大佬，难道你就一直这样傻乎乎的工作 2 年等待着被大佬青睐吗？

要知道，想找到 100%自己满意的工作环境是几乎不可能的！不管是在小公司，做着小业务，还是在大公司，做着大项目。能帮自己的，永远是自己！把自己变强才是硬道理。

现在信息越来越对称，自己只要想学想了解的，几乎都能找到，永远不要被当前的环境所束缚，别给自己找理由。

(2) 尽量和优秀的人一起

① 近朱者赤，近墨者黑。能看得多远，决定了能走多远。前面说到利用自己的能

力去为公司创造价值，从而让自己能实现职业上的晋升，同时也要利用自己职业晋升去认识更多优秀的人：

1) 比如公司 leader，公司的老大 CEO

2) 还有这个行业中优秀的人，积极的去参加一些沙龙、峰会、活动。

3) 还有一些平台，比如**脉脉、知乎**上的一些大佬，还有**在行 app** 这样的平台，付小额的费用，就能找到各个领域的大佬

一个人最快的成长就是获得别人的**间接经验，**学习知识就是这个过程。让自己保持开放的心态，与优秀的人为伍。

8、保重身体，找个女朋友