

作者：Jason，在读研究生，熟悉硬件、STM32单片机、嵌入式Linux。已收获小米、联发科、浙江大华等大厂offer。在公众号分享嵌入式学习规划、嵌入式笔试面试题目、嵌入式求职规划、嵌入式Linux技术文章等。关注公众号回复【电子书】即可获得嵌入式全套500本电子书。

大号：《嵌入式Linux系统开发》，主做技术分享，关注博主免费修改简历。



小号：《程序员杰仔》，主要分享博主日常学习笔记。



我创建了知识星球，在知识星球中有问必答，解答读者各种问题，长期更新嵌入式精选资料。



在其他平台转载发布请联系博主，翻版必究。以下内容已在微信公众号平台标明原创。

前言

本文会分享一些嵌入式软件岗位的秋招事宜，对所有流程都会有一个描述，事无巨细，希望可以帮到各位公众号【嵌入式Linux系统开发】的读者。

下面将自己的学习和秋招经验分享给大家，如有错误，欢迎大家指出，希望能够给迷茫的人带来帮助。

第一章 个人简介

1.1 个人背景

笔者本硕均就读于双非院校（普通一本），电气工程专业，英语六级，有计算机二级C语言和计算机二级MS Office高级应用证书。

本科时期没有个人主见，学校教什么笔者就学什么，计算机方面仅限于熟悉C语言，了解51单片机。研究生时期主要跟导师做一些硬件、STM32单片机的项目，嵌入式Linux为自学。研一期间一年时间主要用Altium Designer画电路原理图和PCB板，研二上学期学习STM32F1单片机写C语言程序，研二下学期和研三上学期主要自学嵌入式Linux，并且有一段四个月Linux平台的实习经历。自学嵌入式Linux过程中，单片机的学习经验起到了很好的作用。

1.2 求职经历

笔者从2020.4.1号开始投递简历找实习，到2020.4.15号截止一共投递过10家公司，收到三家公司实习的offer，选择了其中一家Linux平台，实习了四个月。在工作中熟练使用Source Insight、Xshell、secureCRT、Beyond Compare、MobaXterm、notepad++等软件，这些软件是一个嵌入式软件工程师必会的软件。

笔者从2020.7.1号正式开始投递秋招简历，到2020.9.1号截止一共投递过90家公司，既有提前批（2020年6月-7月），也包括正式批（2020年8月-10月）；主要有手机厂（华为、小米、oppo、vivo等），安防厂（海康威视、浙江大华），芯片原厂（联发科、紫光展锐、华为海思、全志科技，北京君正等），家居家电（美的、海尔、格力），医疗（迈瑞医疗、理邦仪器），电脑厂商，还有一些互联网巨头是有嵌入式岗，奇安信、大疆、汇顶、深信服、京东、CVTE、百度、美团、思科、乐鑫科技、远景、奇虎360、网易、科大讯飞、商汤科技、小马智行、富士康、深圳安信可、汇川技术、锐捷、星际荣耀、英威腾、浙江中控、深圳康冠科技、艾为电子、深圳麦格米特、长沙景嘉微、萨基姆通讯中国深圳有限公司、杰发科技、上能电气、盛弘电气、中联重科、易事特、阿里、图森、滴滴、海浦蒙特、宇视科技、软通动力、广联达、TCL、蔚来、京东方、海信，还有一些国企和研究所。

投递可以去公司官网，推荐牛客网的【校招日程】，很方便，如下图。



笔者秋招共投递了90家企业，给了40场笔试，最后有20家公司给了面试机会，最后收到了8个offer。

1.3 求职结果

经过各种半道折戟，最终成功走到了8家公司的Offer环节：小米研发岗、浙江大华研发岗、联发科技研发岗、富士康研发岗、格力研发岗以及一些中小公司研发岗。

1.4 本文内容

对于没有经历过秋招的同学们来说，可能会对秋招感到迷茫，不知道该如何准备校招、不知道如何写简历等，本文会分享校招的所有流程，包括：

- 1、 校招介绍
- 2、 如何学习嵌入式相关知识
- 3、 如何写简历
- 4、 如何准备笔试和面试
- 5、 如何在面试中与面试官交谈
- 6、 如何选择合适自己的offer

第二章 校招规划

2.1 求职规划

2.1.1 了解校招

校招，不管对于大学生还是研究生来说都是十分重要，大部分的学生都是在校招的短短几个月里找到自己人生的第一份工作的。但校招只是一个比较统一的时间，又可以具体细分为：暑期实习招聘、秋招、秋招补录、来年春招、春招补录。

暑期实习招聘：暑期实习的招聘对象主要是大三下和研二下的学生，时间是每年的3-5月份左右，招聘规模比正式的校招要小一些。对于一些确定要工作而不是读研和读博的同学来说，一定要尽自己的最大力量去参加实习。因为实习是有一定几率转正的。通过实习上岸可比通过正式秋招上岸要容易一些的，这相当于你在正式秋招前就已经有了一个保底的offer了，在秋招过程中压力也没有那么大。并且你自己的实习经历是可以写在简历上，从而为自己的简历增色不少的。如果你有互联网大厂的实习经历那就更好了，面试官看到你有实习的时候都会深挖你的实习过程。（但是不建议因为有了实习offer就放弃正式秋招！因为秋招会有更好的机会，而且你实习的公司也不一定就都能全部转正）

秋招：秋招又分为提前批和正式批两个阶段，甚至还有秋招补录这一环节。其中提前批一般集中在6月-7月，正式批一般是7月-10月，金九银十，就是在形容秋招时期九月和十月是竞争最激烈的时候，求职者也是最多，竞争也是最大。

提前批：提前批主要是一些公司为了能够更快的抢到一些更好的人才、更优质的人才所设置的，所以提前批一般都是神仙打架（竞争极其激烈，各种本硕985人才），但是提前批是非常重要的。因为已经有越来越多的公司看中提前批，已经有不少公司都是给予二次投递机会的，也就是说如果提前批你挂了，正式批还是能够再次投递这个公司的。能够有一个“复活甲”，岂不美滋滋儿。所以同学们千万要参加提前批，千万不要因为觉得自己学历不好或者水平不够就放弃了提前批。但是也不是所有公司都会给你第二次面试机会的，如果你在提前批挂了，正式批就没有机会去进行第二次投递了，所以同学们在投递简历的时候要注意看清楚相应公司的要求，打听好消息（这一点很重要，所以提前批一定要好好准备，否则影响你的正式批）。

正式批：正式批是紧跟在提前批后的，一般提前批结束后马上就会进入正式批环节了。这是整个招聘环节HC最多的时候，建议尽量早一点投递，因为岗位空缺就是那么多，招够一定人数后就不再招了，千万不要想着等自己完全准备好了，万事俱备那种再去投递简历，最好是在提前批就进行简历的投递，因为很有可能出现即使顺利通过面试但坑位不足的情况，导致offer无法顺利审批下来。另外尽量多拿几家公司的offer，为自己带来更好的保障。

秋招补录：秋招补录一般是在每年的10月末-11月份。主要因为在某些大佬或者因为薪资或者地域等原因拒掉手中的offer后，某些岗位出现了没招满的情况下，这些岗位又重新开始招聘了。一般来说补录的名额是相对较少一些的，能够在提前批、秋招时期上岸就尽量上岸，不要把宝压在秋招补录甚至是来年的春招时期。

来年春招：来年的春招一般是在第二年的3-5月份，相较于秋招规模，春招的招聘规模要小很多，主要是因为公司在秋招过程中没有招到足够的人数进行的一次补招，这也是应届毕业生最后一次找到工作的校招机会了。

2.1.2 确立方向

一般来说，越大的公司分工也就越明确，小一点的公司要求你是个多面手。因此建议同学们第一份工作找一家大一点的公司，精进一下个人的技术，也能够镀镀金。

尽早的确定个人的方向能够节省很多时间，目前嵌入式方向主要有嵌入式应用开发、嵌入式驱动开发等。嵌入式应用开发有很多方向，可以偏向于C++的QT界面开发，也可以偏向于音视频流媒体方向，嵌入式驱动开发主要是Linux系统下的驱动开发。应用开发和驱动开发的岗位比例为8:2，各种公司都有嵌入式应用开发岗位，驱动岗位主要存在于各大芯片原厂和大公司中。嵌入式有各种平台，比如RK平台、飞思卡尔平台、海思平台、联发科平台、高通平台，做应用开发，换平台较为麻烦。做驱动开发，主要技术栈在底层，所以更偏向于技术专家，对平台的切换更加得心应手。

2.1.3 明确岗位

这里简单科普一下嵌入式技术栈方向能够投递的一些岗位，方便大家在秋招过程中参考。投递嵌入式岗位，本科专业一般为自动化、电气工程、电子信息、物联网、通信工程专业。其他专业转过来的也可以。

2.1.3.1 嵌入式应用开发工程师

嵌入式应用开发，有很多技术方向，比如音视频开发（如行车记录仪、运动相机），比如通信行业（路由器、中继器），物联网（智能家居），或者可以偏向于C++用QT开发界面。

2.1.3.2 嵌入式驱动工程师

嵌入式驱动工程师一般的职业发展是技术专家。驱动工程师一般都需要学习驱动开发和Linux内核，Linux内核和驱动不分家。熟悉Linux内核才能更清楚底层API的实现原理，才能更好地写好驱动程序，所以驱动工程师一般也兼顾内核工程师。

2.1.3.2 嵌入式Linux运维工程师

主要对公司Linux服务器进行一些运行维护的工作，技术含量较低，技术栈不深，平均薪资不如上面两个岗位高（大厂运维除外）。

2.1.4 获取校招信息

我相信有很多小伙伴跟我一样，对于各种企业的招聘信息的获取感到一头雾水，不知道如何获取这些信息，比如哪些公司开始提前批了，哪些公司开始正式批了，我可以投哪些公司的在哪些城市的哪些岗位等。

这里推荐几个信息获取源：

1、牛客网，牛客校招求职区的校招日程：<https://www.nowcoder.com/school/schedule>，会按照时间将每天开启校招日程的企业罗列出来，建议小伙伴们好好利用牛客网，海投神器。

2、各种微信群、QQ群：学校会对每一届学生建立一个就业QQ群，发布招聘信息。各个学校的就业处公众号每天也会有招聘信息发布。

3、学校官网。求职季会有很多企业来学校宣讲，到时候年级群中导师也会跟进，记得每天按时看群，注意自己中意的公司即可。

4、企业官网。如果有自己想去的企业或者公司，可以直接去他们的官网找到校招板块，直接投递个人简历，建议填上内推码。

5、各种微信公众号也会有招聘信息。

建议能用内推码就用内推码，在牛客网有很多内推码分享，各大公司都有。用内推码有时候可以免笔试，有时候能保证简历不被刷，各种好处，有时候还可以跟进投递进度。内推人推荐你入职，推荐人是有红包的，所以员工都会在各种平台上分享自己的内推码。

2.2 学习规划

这里我将自己的学习方法和相关资料推荐给大家，希望能够对大家有所帮助，毕竟经典书籍太多，每个人一天都只有24小时，有时候真是看不过来。

2.2.1 C语言

做嵌入式，一般都会操作寄存器，C语言用的较多，另外笔者主要用C语言，所以本文主要介绍C语言。C语言，随便找一个大学课本，即可入门，最经典的是谭浩强那本。入门以后，如果想要精进C语言，推荐C语言三剑客：《C和指针》、《C专家编程》、《C缺陷与陷阱》。看完三剑客，C语言基本上已经算熟悉。但是我们毕竟是嵌入式岗位，不是C语言工程师，所以要熟悉Linux下的C语言编程，所以还需要看一些Linux下C编程的书籍，比如《Linux C编程一站式学习》。

No.3 豆瓣热门编程图书TOP10

C程序设计语言



作者: (美) Brian W. Kernighan / (美) Dennis M. Ritchie
出版社: 机械工业出版社
出品方: 华章IT
副标题: 第2版·新版
原名: The C Programming Language
译者: 徐宝文 / 李志译 / 尤晋元审校
出版年: 2004-1
页数: 258
定价: 30.00元
装帧: 平装
丛书: 计算机科学丛书
ISBN: 9787111128069

豆瓣评分

9.4 ★★★★★
4688人评价

5星	75.0%
4星	20.5%
3星	3.8%
2星	0.3%
1星	0.4%

推荐指数：五颗星★★★★★

书名：《C程序设计语言》

理由：本书原著即为C语言的设计者之一Dennis M. Ritchie和著名计算机科学家Brian W. Kernighan合著的一本介绍C语言的权威经典著作。我们现在见到的大量论述C语言程序设计的教材和专著均以此书为蓝本。原著第1版中介绍的C语言成为后来广泛使用的C语言版本——标准C的基础。人们熟知的“hello, World”程序就是由本书首次引入的，现在，这一程序已经成为众多程序设计语言入门的第一课。本书只有很薄的一两百页，却是C语言的精华。

C程序设计



作者: 谭浩强
出版社: 清华大学出版社
出版年: 2010-7
页数: 268
定价: 25.00元
装帧: 平装
丛书: 中国高等教育计算机基础教育课程体系规划教材
ISBN: 9787302226727

豆瓣评分

7.7  34人评价

5星	17.6%
4星	29.4%
3星	29.4%
2星	14.7%
1星	8.8%

推荐指数：五颗星★★★★★

书名：《C程序设计》

理由：这是很多大学的教材，事无巨细。《C程序设计(第4版)学习辅导》是与谭浩强所著的《C程序设计(第四版)》（清华大学出版社出版）配合使用的参考用书。全书共分4个部分，第1部分是《C程序设计(第四版)》一书的习题和参考解答，包括了该书各章的全部习题，对全部编程习题都给出了参考解答，共计132个程序；第2部分是深入学习C程序设计，包括预处理指令、位运算和C程序案例；第3部分是上机指南，详细介绍了Visual C++ 6.0集成环境下编辑、编译、调试和运行程序的方法；第4部分是上机实验指导，包括程序的调试与测试、实验的目的与要求，并提供了本课程12个实验。

C专家编程



作者: Peter Van Der Linden
出版社: 人民邮电出版社
原名: Expert C Programming: Deep C Secrets
译者: 徐波
出版年: 2008-2
页数: 291
定价: 45.00元
装帧: 平装
丛书: C和C++经典著作
ISBN: 9787115171801

豆瓣评分

9.2  1994人评价

5星	64.4%
4星	30.2%
3星	4.6%
2星	0.5%
1星	0.3%

推荐指数：五颗星★★★★★

书名：《C专家编程》

《C专家编程》展示了最优秀的C程序员所使用的编码技巧，并专门开辟了一章对C++的基础知识进行了介绍。

书中C的历史、语言特性、声明、数组、指针、链接、运行时、内存以及如何进一步学习C++等问题进行了细致的讲解和深入的分析。全书撷取几十个实例进行讲解，对C程序员具有非常高的实用价值。

本书可以帮助有一定经验的C程序员成为C编程方面的专家，对于具备相当的C语言基础的程序员，本书可以帮助他们站在C的高度了解和学习C++。






C陷阱与缺陷



作者: 凯尼格
出版社: 人民邮电出版社
副标题: C语言调试指南
原名: C Traps and Pitfalls
译者: 高巍
出版年: 2008-2-1
页数: 172
定价: 30.00元
装帧: 平装
丛书: C和C++经典著作
ISBN: 9787115171795

豆瓣评分

8.9 
1376人评价

5星  51.7%
4星  39.5%
3星  7.7%
2星  1.0%
1星  0.1%

推荐指数：五星级★★★★★

书名：《C陷阱和缺陷》

理由：这里搬运一段百度百科上的介绍和说明：“本书的出发点不是要批判C语言，而是要帮助C程序员绕过编程过程中的陷阱和障碍。全书分为8章，分别从词法分析、语法语义、连接、库函数、预处理器、可移植性缺陷等几个方面分析了C编程中可能遇到的问题。最后，作者用一章的篇幅给出了若干具有实用价值的建议。本书适合有一定经验的C程序员阅读学习，即便你是C编程高手，本书也应该成为你的案头必备书籍。”，从这段介绍中就可以感受到这本书的分量了。

除此之外，还有一些比较优秀的书籍，笔者暂时还没有看的，如果你想要走C/C++开发的路线，这些书籍都是十分不错的资料。






C和指针



作者: Kenneth A. Reek
出版社: 人民邮电出版社
原名: Pointers on C
译者: 徐波
出版年: 2008年4月
页数: 448
定价: 65.00元
装帧: 平装
丛书: C和C++经典著作
ISBN: 9787115172013

豆瓣评分

9.0 
1491人评价

5星  63.6%
4星  30.6%
3星  5.0%
2星  0.5%
1星  0.3%

推荐指数：五星级★★★★★

书名：《C和指针》

理由：指针方面的经典好书，里面涉及了好多C语言相关的知识，比如数据、语句、操作符和表达式等，但是讲的最好的就是指针以及指针和数组的关系了，也给出了不少编程技巧和提示。

Linux C编程一站式学习



作者: 宋劲杉
出版社: 电子工业出版社
出品方: 博文视点
出版年: 2009-12
页数: 463
定价: 60.00元
装帧: 平装
ISBN: 9787121097713

豆瓣评分

8.9  634人评价

5星  62.3%
4星  30.9%
3星  5.4%
2星  0.6%
1星  0.8%

推荐指数：四颗星★★★★

书名：《Linux C编程一站式学习》

理由：本书有两条线索，一条线索是以Linux平台为载体全面深入地介绍C语言的语法和程序的工作原理，另一条线索是介绍程序设计的基本思想和开发调试方法。本书分为两部分：第一部分讲解编程语言和程序设计的基本思想方法，让读者从概念上认识C语言；第二部分结合操作系统和体系结构的知识讲解程序的工作原理，让读者从本质上认识C语言。

本书适合做零基础的初学者学习C语言的第一本教材，帮助读者打下牢固的基础。有一定的编程经验但知识体系不够完整的读者也可以对照本书查缺补漏，从而更深入地理解程序的工作原理。

2.2.2 数据结构与算法

数据结构与算法是相辅相成的关系，学好算法有助于理解数据结构，学好数据结构也更有助于理解好算法。

在秋招过程中，数据结构是极其重要的。对于经典的数据机构与算法大家都要掌握，对于一些常见的数据结构：树、链表、队列、栈等要有一定的了解。

我个人学习数据结构与算法的路线是先从简单的书籍看起，然后过渡到一些经典数据结构相关书籍，在此过程中书本后的比较好的课后习题也没有放过，学完数据结构后就开始了漫长的刷题之路了。

笔者在面试过程中就被考过其中几道经典题型：反转链表、双向链表的插入删除、字符串翻转等。

下面推荐一下经典书籍：






大话数据结构



作者: 程杰
出版社: 清华大学出版社
出版年: 2011-6
页数: 440
定价: 59.00元
装帧: 平装
丛书: 大话系列
ISBN: 9787302255659

豆瓣评分

7.9  1370人评价

5星  30.7%
4星  45.2%
3星  18.0%
2星  4.0%
1星  2.0%

推荐指数：四颗星★★★★

书名：《大话数据结构》

理由：对于一些小白来说这本书是福音了，大话系列的典范之作。将数据结构中比较晦涩难懂的链表、树等知识讲得通俗易懂，对新手比较友好。有一定数据结构基础的可以忽略了。

啊哈!算法



作者: 啊哈磊
出版社: 人民邮电出版社
出版年: 2014-6-1
页数: 246
定价: 45.00元
装帧: 平装
丛书: 图灵原创
ISBN: 9787115354594

豆瓣评分

7.7 ★★★★★
505人评价

5星 35.6%
4星 40.0%
3星 20.0%
2星 2.8%
1星 1.6%

推荐指数：四颗星★★★★

书名：《啊哈！算法》

理由：与大话数据结构一样对于新手比较友好，是一本很有趣的算法入门书，如果你有一定算法或者coding基础就不必看了。

剑指Offer



作者: 何海涛
出版社: 电子工业出版社
出品方: 博文视点
副标题: 名企面试官精讲典型编程题
出版年: 2012-1
页数: 260
定价: 45.00元
装帧: 平装
ISBN: 9787121148750

豆瓣评分

8.3 ★★★★★
786人评价

5星 44.0%
4星 42.9%
3星 12.0%
2星 0.5%
1星 0.6%

推荐指数：五颗星★★★★★

书名：《剑指Offer》

理由：这本书不需要多做介绍，校招必备！可是还是要自己看起来、刷起来，不要放在那里吃灰，如果这本书上的题目你都没有做过或者也不会的话，算法这一关基本是送人头的存在了，笔者在秋招过程中这本书看了2遍。

2.2.3 操作系统

操作系统是一门在面试过程中问的不算很深的课程，因为这门课往下走的话深度太深，也不好展开，面试官不好尝试，甚至于一些面试官对于某些具体的知识点也不熟悉（大佬除外），操作系统必须深入学习，才能学明白学透彻。

书籍推荐：

深入理解计算机系统（原书第3版）



作者: Randal E. Bryant / David O'Hallaron
出版社: 机械工业出版社
原名: Computer Systems: A Programmer's Perspective (3rd Edition)
译者: 龚奕利 / 贺莲
出版年: 2016-11
页数: 737
定价: 139.00元
装帧: 平装
丛书: 计算机科学丛书
ISBN: 9787111544937

豆瓣评分

9.8 ★★★★★
1192人评价

5星 89.7%
4星 9.0%
3星 1.0%
2星 0.1%
1星 0.3%

推荐指数：五颗星★★★★★

书名：《深入理解计算机系统》

理由：被誉为“和金子一样重要的计算机基础书籍”，就好像学霸考试只能考100分是因为试卷只有100分一样，这本书推荐指数为五星，那是因为最高就是五星了。这本书十分经典，每一次看都会有新的体会和感悟，这本书从程序执行的计算机角度开始，介绍了处理器的体系结构、程序的机器级优化、虚拟存储器、系统级IO、网络通信等等多个方面。

现代操作系统（第3版）



作者: [美] Andrew S. Tanenbaum
出版社: 机械工业出版社
原名: Modern Operating Systems
译者: 陈向群 / 马洪兵
出版年: 2009-7
页数: 582
定价: 75.00元
装帧: 平装
丛书: 计算机科学丛书
ISBN: 9787111255444

豆瓣评分

8.9 ★★★★★
777人评价

5星 58.6%
4星 32.4%
3星 7.3%
2星 1.0%
1星 0.6%

推荐指数：五颗星★★★★★

书名：《现代操作系统》

理由：同样是讲操作系统的一本好书，《深入理解计算机系统》有些操作系统的知识讲的比较泛没有这本书细致，如果赶时间的话可以把这本书中进程、死锁、缓存等重要知识点先看一下，后续有时间了再补上其他章节。

自己动手写操作系统



作者: 于渊
出版社: 电子工业出版社
出版年: 2005-8
页数: 374
定价: 48.00元
装帧: 平装
ISBN: 9787121015779

豆瓣评分

7.9 ★★★★★
319人评价

5星 32.0%
4星 41.4%
3星 23.8%
2星 2.5%
1星 0.3%

推荐指数：四颗星★★★★

书名：《自己动手写操作系统》

理由：很好的一本实践书籍，看这本书的前提是要有一定的汇编知识，如果不懂一些基本的汇编知识容易看的迷迷糊糊。本书亲自带你走一遍操作系统的实现，打造一个简易版的操作系统，笔者在学完汇编后花了二十余天跟着走了一遍，感觉很多东西豁然开朗了一样，值得一看！

2.2.4 计算机网络

计算机网络是重点之一，特别是TCP/UDP相关知识点，面试必问。考察计算机网络对于TCP/UDP，一般问一些基本的三次握手四次挥手大概过程，问TCP和UDP的区别，为什么TCP可靠。问OSI网络协议都分为几层，每一层是什么。

视频推荐：谢希仁老师的计算机网络视频，谢老师讲课很有意思，整个课堂充满了欢声笑语，也可以看视频下的留言，那里都是有很多好笔记的。

书籍推荐：这里从易到难逐步推荐一些比较好的计算机网络经典图书。

图解TCP/IP（第5版）



作者: [日]竹下隆史 / [日]村山公保 / [日]荒井透 / [日]蒔田幸雄
出版社: 人民邮电出版社
出品方: 图灵教育
原作名: マスタリングTCP/IP 入門編 第5版
译者: 乌尼日其其格
出版年: 2013-7-1
页数: 312
定价: 69.00元
装帧: 平装
丛书: 图灵程序设计丛书: 图解与入门系列
ISBN: 9787115318978

豆瓣评分

7.9 ★★★★★
944人评价

5星 24.8%
4星 50.4%
3星 22.0%
2星 2.5%
1星 0.2%

推荐指数：五颗星★★★★★

计算机网络（原书第7版）



作者: James F. Kurose / Keith W. Ross
出版社: 机械工业出版社
副标题: 自顶向下方法
原名: Computer Networking: A Top-Down Approach
译者: 陈鸣
出版年: 2018-6
页数: 480
定价: 89.00元
装帧: 平装
丛书: 计算机科学丛书
ISBN: 9787111599715

豆瓣评分

9.2 ★★★★★
265人评价

5星 69.1%
4星 25.3%
3星 3.8%
2星 0.8%
1星 1.1%

推荐指数：五星级★★★★★

书名：《计算机网络：自顶向下方法》

理由：别的常见介绍计网的书籍是从底向上即物理层到应用程序介绍网络，这本书另辟蹊径，是自顶向下介绍整个网络的。这样做的好处是从我们所接触的应用层开始逐步深入，而不是从离我们最远的物理层开始。如果不是网络安全相关专业，大多数人看重点章节也就是第三章传输层那一章，重点看TCP/UDP的各种细节基本也够用了，剩下的可以后期再补，为自己节约时间。

TCP/IP详解 卷1：协议（原书第2版）



作者: Kevin R. Fall W. Richard Stevens
出版社: 机械工业出版社
副标题: 卷1：协议（原书第2版）
原名: TCP/IP Illustrated, Volume 1: The Protocols
译者: 吴英 / 张玉 / 许昱玮
出版年: 2016-6
页数: 683
定价: CNY 129.00
装帧: 平装
丛书: 计算机科学丛书
ISBN: 9787111453833

豆瓣评分

7.2 ★★★★★
101人评价

5星 42.6%
4星 23.8%
3星 20.8%
2星 6.9%
1星 5.9%

推荐指数：五星级★★★★★

书名：《TCP/IP详解 卷1：协议》

理由：确认过眼神，是经典中的经典，没错了。不过就是，emmm太厚了...笔者买来翻了翻TCP/UDP知识点就用来垫电脑了.....对于TCP的各种机制介绍的很细致，看了之后对于TCP/UDP感觉明显上升了一个台阶。如果不是信息安全、网络安全相关岗位的，可以作为一本工具书来使用的，需要用到某些知识再来补就行。

2.2.5 数据库

嵌入式岗位一般很少涉及到数据库相关知识。

2.2.6 Linux

嵌入式跟Linux是离不开的，嵌入式开发最常用的操作系统就是Linux系统，有几个最主要的原因：1、Linux系统开源免费。2、Linux有最完好的生态，最多的参考资料。

除了Linux系统，嵌入式开发还常用一些其他的操作系统比如RTOS、FreeRTOS、RTT等小型操作系统。

Linux学习路径大概分为三个方向：Linux入门——Linux应用开发——Linux驱动——Linux内核。

推荐图书分别如下：

Linux入门：

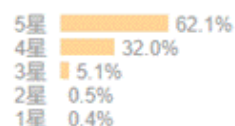
鸟哥的Linux私房菜



作者: 鸟哥
出版社: 人民邮电出版社
副标题: 基础学习篇
出版年: 2010-6-28
页数: 778
定价: 88.00元
装帧: 平装
丛书: 鸟哥的Linux私房菜
ISBN: 9787115226266

豆瓣评分

9.1 ★★★★★
3093人评价



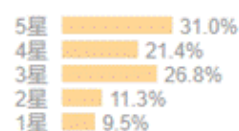
Linux就该这么学



作者: 刘遄
出版社: 人民邮电出版社
出品方: 异步图书
副标题: 必读的Linux系统与红帽认证自学书籍
出版年: 2017-11-1
页数: 450
定价: 79
装帧: 平装
ISBN: 9787115470317

豆瓣评分

5.8 ★★★☆☆
168人评价



推荐指数：四颗星★★★★

书名：《鸟哥的Linux私房菜》、《Linux就该这么学》

理由：这两本书都挺经典的，算是比较好的Linux入门书了。如果想要系统学习Linux可以照着书本上的命令老老实实在敲上一遍，Linux命令这一块基本没啥问题了。

Linux应用开发：

嵌入式Linux应用开发完全手册



作者: 韦东山 主编
出版社: 人民邮电出版社
出版年: 2008-8
页数: 579
定价: 69.00元
ISBN: 9787115182623

豆瓣评分

8.5  115人评价

5星		38.3%
4星		44.3%
3星		14.8%
2星		2.6%
1星		0.0%

推荐指数：四颗星★★★★

书名：《嵌入式Linux应用开发完全手册》

理由：韦东山老师的开山之作，非常贴合实际，讲解基础概念，五星好评。

本书全面介绍了嵌入式Linux系统开发过程中，从底层系统支持到上层GUI应用的方方面面，内容涵盖Linux操作系统的安装及相关工具的使用、配置，嵌入式编程所需要的基础知识（交叉编译工具的选项设置、Makefile语法、ARM汇编指令等），硬件部件的使用及编程（囊括了常见硬件，比如UART、I2C、LCD等），U-Boot、Linux内核的分析、配置和移植，根文件系统的构造（包括移植busybox、glibc、制作映象文件等），内核调试技术（比如添加kgdb补丁、栈回溯等），驱动程序编写及移植（LED、按键、扩展串口、网卡、硬盘、SD卡、LCD和USB等），GUI系统的移植（包含两个GUI系统：基于Qtopia和基于X），应用程序调试技术。

本书从最简单的点亮一个LED开始，由浅入深地讲解，使读者最终可以配置、移植、裁剪内核，编写驱动程序，移植GUI系统，掌握整个嵌入式Linux系统的开发方法。

本书由浅入深，循序渐进，适合刚接触嵌入式Linux的初学者学习，也可作为大、中专院校嵌入式相关专业本科生、研究生的教材。

UNIX网络编程 卷1：套接字联网API（第3版）



作者: [美]W. 理查德·史蒂文斯 (W. Richard Stevens) / 比尔·芬纳 (Bill Fenner) / 安德鲁 M. 鲁道夫 (Andrew M. Rudoff)

出版社: 人民邮电出版社

出品方: 异步图书

副标题: 套接字联网API

原作名: UNIX Network Programming, Volume 1: The Sockets Networking API, Third Edition

译者: 匿名

出版年: 2014-6-1

页数: 824

定价: 129.00

装帧: 平装

ISBN: 9787115367198

豆瓣评分

9.6 ★★★★★
82人评价

5星 75.6%
4星 19.5%
3星 3.7%
2星 1.2%
1星 0.0%

UNIX网络编程 卷2：进程间通信（第2版）



作者: [美]W. 理查德·史蒂文斯 (W. Richard Stevens)

出版社: 人民邮电出版社

出品方: 异步图书

原作名: UNIX Network Programming, Volume 2: Interprocess Communications, Second Edition

译者: 匿名

出版年: 2015-8

页数: 472

定价: 89.00

装帧: 平装

ISBN: 9787115367204

豆瓣评分

9.6 ★★★★★
27人评价

5星 81.5%
4星 14.8%
3星 3.7%
2星 0.0%
1星 0.0%

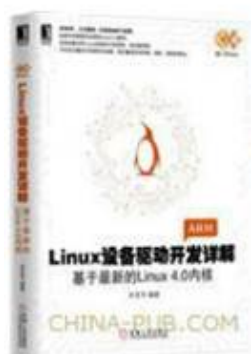
推荐指数：五颗星★★★★★

书名：《UNIX网络编程 卷1:套接字联网API》、《UNIX网络编程卷2：进程间通信》

理由：史蒂文斯大神的盖世之作。说一句“网络编程方面的圣经”不为过。对于有志进入腾讯鹅厂的小伙伴，这两本书可以说是必须要看的，可以说是网络研究和开发人员公认的权威参考书，无论网络编程的初学者还是网络专家都会大受裨益。不过因为太厚的原因，特别是《套接字联网API》可以说是我遇到的最厚的技术书了，全部啃完需要耗不少时间，可以像笔者一样，哪里不会翻哪里。

Linux驱动：

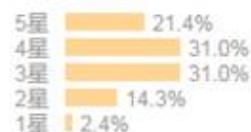
Linux设备驱动开发详解



作者: 宋宝华
出版社: 机械工业出版社
副标题: 基于最新的Linux 4.0内核
出版年: 2015-8
页数: 618
定价: 89.00元
装帧: 平装
丛书: 电子与嵌入式系统设计译丛
ISBN: 9787111507895

豆瓣评分

6.4 42人评价



推荐指数：五颗星★★★★★

书名：《Linux设备驱动开发详解》

理由：对于嵌入式工程师来说，进入更高阶段后，学习Linux设备驱动开发无疑就是职业生涯的一次“重生”。这是因为Linux设备驱动开发不仅仅涉及操作系统的转换，开发方式的转换，更重要的是思维上的转变。对于Linux这样一个复杂系统，如何从复杂的代码中抓住设备驱动开发的关键是任何一个Linux设备驱动开发者入门时需要面对的挑战。除了知识、工具之外，往往还需要思路上的指导。本书不但帮助Linux设备驱动开发的初学者厘清必要的概念，还从具体的实例、设备驱动开发的指导原则循序渐进地引导读者渐入学习佳境。为了让读者能够达到Linux设备驱动开发的至臻境界，作者更是从软件工程的角度抽象出设备驱动开发的一般思想。毫无疑问，本书将成为读者学习Linux设备驱动开发过程中的一座“灯塔”。

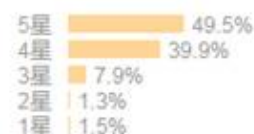
Linux设备驱动程序



作者: 科波特
出版社: 中国电力出版社
原名: Linux Device Drivers, Third Edition
译者: 魏永明 / 耿岳 / 钟书毅
出版年: 2006-1-1
页数: 569
定价: 69.00元
装帧: 平装
ISBN: 9787508338637

豆瓣评分

8.7 469人评价



推荐指数：五颗星★★★★★

书名：《Linux设备驱动程序》

理由：本书是经典著作《Linux设备驱动程序》的第三版。如果您希望在Linux操作系统上支持计算机外部设备，或者在Linux上运行新的硬件，或者只是希望一般性地了解Linux内核的编程，就一定要阅读本书。本书描述了如何针对各种设备编写驱动程序，而在过去，这些内容仅仅以口头形式交流，或者零星出现在神秘的代码注释中。

本书的作者均是Linux社区的领导者。Jonathan Corbet虽不是专职的内核代码贡献者，但他是备受关注的LWN.net新闻及信息网站的执行编辑。Alessandro Rubini是一名Linux代码贡献者，也是活跃的意大利Linux社区的灵魂人物。Greg Kroah-Hartman是目前内核中USB、PCI和驱动程序核心子系统（本书均有讲述）的维护者。

本书的这个版本已针对Linux内核的2.6.10版本彻底更新过了。内核的这个版本针对常见任务完成了合理化设计及相应的简化，如即插即用、利用sysfs文件系统和用户空间交互，以及标准总线上的多设备管理等。

要阅读并理解本书，您不必首先成为内核黑客；只要您理解C语言并具有Unix系统调用的一些背景知识即可。您将学到如何为字符设备、块设备和网络接口编写驱动程序。为此，本书提供了完整的示例程序，您不需要特殊的硬件即可编译和运行这些示例程序。本书还在单独的章节中讲述了PCI、USB和tty（终端）子系统。对期望了解操作系统内部工作原理的读者来讲，本书也深入阐述了地址空间、异步事件以及I/O等方面的内容。

Linux内核：

Linux内核设计与实现(原书第3版)



作者: Robert Love
出版社: 机械工业出版社华章公司
原名: Linux Kernel Development, 3E
译者: 陈莉君 / 康华
出版年: 2011-4-30
页数: 352
定价: 69.00元
装帧: 平装
丛书: 华章专业开发者丛书
ISBN: 9787111338291

豆瓣评分



推荐指数：五颗星★★★★★

书名：《Linux内核设计与实现(原书第3版)》

理由：详细描述了Linux内核的设计与实现。内核代码的编写者、开发者以及程序开发人员都可以通过阅读本书受益，他们可以更好理解操作系统原理，并将其应用在自己的编码中以提高效率和生产率。

详细描述了Linux内核的主要子系统和特点，包括Linux内核的设计、实现和接口。从理论到实践涵盖了Linux内核的方方面面，可以满足读者的各种兴趣和需求。

作者Robert Love是一位Linux内核核心开发人员，他分享了在开发Linux 2.6内核过程中颇具价值的知识和经验。本书的主题包括进程管理、进程调度、时间管理和定时器、系统调用接口、内存寻址、内存管理和页缓存、VFS、内核同步、移植性相关的问题以及调试技术。同时本书也涵盖了Linux 2.6内核中颇具特色的内容，包括CFS调度程序、抢占式内核、块I/O层以及I/O调度程序。

奔跑吧 Linux内核 入门篇



作者: 张天飞
出版社: 人民邮电出版社
出品方: 异步图书
出版年: 2019-2
定价: 80.50元
ISBN: 9787115502261

豆瓣评分

★★★★★
评价人数不足

推荐指数：五颗星★★★★★

书名：《奔跑吧Linux内核入门篇》

理由：本书是一本介绍Linux内核实践的入门书，基于Linux 4.0内核，重点讲解Linux内核的理论和实验。本书分为12章，包括Linux系统入门、Linux内核基础知识、内核编译和调试、内核模块、简单的字符设备驱动、系统调用、内存管理、进程管理、同步管理、中断管理、调试和性能优化，以及如何参与开源社区等内容。此外，本书还介绍了Linux内核社区常用的开发工具和理论，如Vim 8和git工具等。书中包括70多个实验，帮助读者深入理解Linux内核。

奔跑吧 Linux内核



作者: 张天飞
出版社: 人民邮电出版社
出品方: 异步图书
出版年: 2017-9-1
页数: 760
定价: CNY 158.00
装帧: 平装
ISBN: 9787115465023

豆瓣评分

7.4 ★★★★★
26人评价

5星	38.5%
4星	30.8%
3星	15.4%
2星	11.5%
1星	3.8%

推荐指数：五颗星★★★★★

书名：《奔跑吧Linux内核》

理由：本书内容基于Linux4.x内核，主要选取了Linux内核中比较基本和常用的内存管理、进程管理、并发与同步，以及中断管理这4个内核模块进行讲述。全书共分为6章，依次介绍了ARM体系结构、Linux内存管理、进程调度管理、并发与同步、中断管理、内核调试技巧等内容。本书的每节内容都是一个Linux内核的话题或者技术点，读者可以根据每小节前的问题进行思考，进而围绕问题进行内核源代码的分析。

本书内容丰富，讲解清晰透彻，不仅适合有一定Linux相关基础的人员，包括从事与Linux相关的开发人员、操作系统的研究人员、嵌入式开发人员及Android底层开发人员等学习和使用，而且适合作为对Linux感兴趣的程序员的学习用书，也可以作为大专院校相关专业师生的学习用书和培训学校的教材。

深入理解LINUX内核(第三版)



作者: (美) 博韦, 西斯特
出版社: 中国电力出版社
原作名: Understanding the Linux Kernel
译者: 陈莉君; 张琼声; 张宏伟
出版年: 2007-10-01
页数: 896
定价: 98.00元
装帧: 平装
丛书: O'Reilly系列
ISBN: 9787508353944

豆瓣评分

8.7 ★★★★★
461人评价

5星	54.2%
4星	33.8%
3星	9.5%
2星	1.1%
1星	1.3%

推荐指数：四颗星★★★★

书名：《深入理解Linux内核》

理由：堪称讲述Linux内核方面不可多得的一本好书，个人感觉比那本《Linux内核完全注释》要好上不少，不过仁者见仁智者见智。这本书可以很好的对你在内核中使用的最重要的数据结构、算法和程序设计诀窍进行一次详细解读，也能帮助你在以后使用Linux时更好地进行进程调度、文件存取和内存管理。

2.2.7 其他

还有一些经典书籍比较好，适合自己持续性充电、打基础，这里也推荐给大家。

代码整洁之道



作者: [美]Robert C. Martin
出版社: 人民邮电出版社
原名: Clean Code: A Handbook of Agile Software Craftsmanship
译者: 韩磊
出版年: 2010-1-1
页数: 388
定价: 59.00元
装帧: 平装
丛书: 异步图书-程序员必读经典系列
ISBN: 9787115216878

豆瓣评分



推荐指数：四颗星★★★★

书名：《代码整洁之道》

理由：软件质量，不但依赖于架构及项目管理，而且与代码质量紧密相关。这一点，无论是敏捷开发流派还是传统开发流派，都不得不承认。

本书提出一种观念：代码质量与其整洁度成正比。干净的代码，既在质量上较为可靠，也为后期维护、升级奠定了良好基础。作为编程领域的佼佼者，本书作者给出了一系列行之有效的整洁代码操作实践。这些实践在本书中体现为一条条规则（或称“启示”），并辅以来自现实项目的正、反两面的范例。只要遵循这些规则，就能编写出干净的代码，从而有效提升代码质量。

本书阅读对象为一切有志于改善代码质量的程序员及技术经理。书中介绍的规则均来自作者多年的实践经验，涵盖从命名到重构的多个编程方面，虽为一“家”之言，然诚有可资借鉴的价值。

Code：隐匿在计算机软硬件背后的语言（英文版）



作者: 【美】查尔斯·佩措尔德 (Charles Petzold)
出版社: 电子工业出版社
出版年: 2019-6
页数: 408
定价: 118.00元
装帧: 平装
丛书: 原味精品书系
ISBN: 9787121358036

豆瓣评分

★★★★★
评价人数不足

推荐指数：四颗星★★★★

书名：《Code：隐匿在计算机软硬件背后的语言》

理由：是一本讲述计算机工作原理的书。不过，你千万不要因为“工作原理”之类的字眼就武断地认为它是晦涩而难懂的。作者用丰富的想象和清晰的笔墨将看似烦杂的理论阐述得通俗易懂，你丝毫不会感到枯燥和生硬。更重要的是，你会因此更加深刻地理解计算机的工作原理。这种理解不是抽象层面上的，而是具有一定深度的，这种深度甚至不逊于“电气工程师”和“程序员”的理解。

不管你是计算机高手，还是对这个神奇的机器充满敬畏之心的“小白”都不妨翻阅一下本书，读一读大师的经典作品，必然会有收获。

老码识途



作者: 韩宏
出版社: 电子工业出版社
副标题: 从机器码到框架的系统观逆向修炼之路
出版年: 2012-8
页数: 344
定价: 56.00元
ISBN: 9787121173820

豆瓣评分

9.1 ★★★★★
54人评价

5星	61.1%
4星	20.4%
3星	18.5%
2星	0.0%
1星	0.0%

推荐指数：四颗星★★★★

书名：《老码识途：从机器码到框架的系统观逆向修炼之路》

理由：汇编与反汇编就好像如今的爬虫与反爬虫一样，这本书以逆向反汇编为线索，自底向上，从探索者的角度，原生态地刻画了对系统机制的学习，对于有志成为系统架构师的小伙伴来说不应该错过这本好书，这本书涉及反汇编、底层调试、链接、加载、钩子等在别的书中很少看到的知识等。

2.2.8 学习方法

我知道有些小伙伴会被上面的书籍和资料分享吓到了，可能有些人会问，这些书你都看了吗？自己能不能看完？

不过每个人的生长环境以及自身水平不同，适合自己的学习方法也不尽相同。我只是和大家分享一下自己在学习上的一些小技巧或者方法，如果有可以借鉴的地方自然是好的。

1、看视频

看视频个人所花费的精力会少于读书，因为你所获取的知识都是讲课老师消化好又传授给你的，你只需要被动接受即可，一个好的老师会让你觉得他所讲的东西容易理解与掌握。但是视频的广度和深度是不如书籍的。而且因为老师语速和自身看视频时的注意力等问题，与看书相比，看视频的效率要低得多。看视频适合于快速入门和那些自学能力不是很好的同学。

2、看书

学会善用目录。有时候，看过目录后就大概知道这章或者这小节讲的是什么了，建议在看一本书的时候先看一遍目录，挑选出自己不懂得或者感兴趣的章节来看，而将已看过的或者暂时不需要的放到后期再去看。

3、看博客与论坛

有很多大佬都是很乐于分享的，会将自己对于某个问题的看法发表在一些博客或者论坛上。博客的限制就在于一篇博客上分享的知识有限，所以这种学习方式比较适合你有了一定的基础后，再开始看别人的博客。千万不要在自己还是个小白的时候就贸然看别人写的博客。当我们对于一门科目或者知识有了整体认识后，剩下要做的就是查漏补缺了，对于书中不懂的细节问题进行逐个攻破，这往往需要我们自身对该问题就有一定的认识，自己手动的去提取某个问题，然后在各个博客上寻找答案。

4、看官方文档或者源码

这种方式适合有一定水平读者，对于学习能力很强的同学来说有的就是直接生搬，简单粗暴，过程很辛苦但是如果能够成功搬出来，收获会很大的。在学习一些知名项目的时候，也可以直接看人家的源码，一般来说正规一点的开源项目都是有代码注释的，不过不少都是英文注释，需要一定的英文水平才能驾驭。

第三章 硬实力

3.1 专业技能准备与提升

3.1.1 C/C++

一般来说，嵌入式岗位考察的主要知识点可以分为C语言基础、C++语言基础、内存管理、Linux操作系统、数据结构与算法、Linux驱动、Linux内核等这几个方面。

本系列按类别对题目进行分类整理，这样有利于大家对嵌入式的笔试面试考察框架有一个完整的理解。

3.1.1.1 C语言基础

1、new和malloc

做嵌入式，对于内存是十分在意的，因为可用内存有限，所以嵌入式笔试面试题目，内存的题目高频。

1) malloc和free是c++/c语言的库函数，需要头文件支持stdlib.h；new和delete是C++的关键字，不需要头文件，需要编译器支持；

2) 使用new操作符申请内存分配时，无需指定内存块的大小，编译器会根据类型信息自行计算。而malloc则需要显式地支持所需内存的大小。

3) new操作符内存分配成功时，返回的是对象类型的指针，类型严格与对象匹配，无需进行类型转换，故new是符合类型安全性的操作符。而malloc内存分配成功则是返回void，需要通过强制类型转换将void指针转换成我们需要的类型。

4) new内存分配失败时，会抛出bad_alloc异常。malloc分配内存失败时返回NULL。

2、在1G内存的计算机中能否malloc(1.2G)？为什么？（2021浙江大华二面问题）

答：是有可能申请1.2G的内存的。

解析：回答这个问题前需要知道malloc的作用和原理，应用程序通过malloc函数可以向程序的虚拟空间申请一块虚拟地址空间，与物理内存没有直接关系，得到的是在虚拟地址空间中的地址，之后程序运行所提供的物理内存是由操作系统完成的。

3、extern“C”的作用

我们可以在C++中使用C的已编译好的函数模块，这时候就需要用到extern“C”。也就是extern“C”都是在c++文件里添加的。

extern在链接阶段起作用（四大阶段：预处理--编译--汇编--链接）。

4、strcat、strncat、strcmp、strcpy哪些函数会导致内存溢出？如何改进？（2021浙江大华二面问题）

strcpy函数会导致内存溢出。

strcpy拷贝函数不安全，他不做任何的检查措施，也不判断拷贝大小，不判断目的地址内存是否够用。

```
1 | char *strcpy (char *strDest, const char *strSrc)
```

strncpy拷贝函数，虽然计算了复制的大小，但是也不安全，没有检查目标的边界。

```
1 | strncpy(dest, src, sizeof(dest));
```

strncpy_s是安全的

strcmp(str1,str2)，是比较函数，若str1=str2，则返回零；若str1<str2，则返回负数；若str1>str2，则返回正数。（比较字符串）

strncat()主要功能是在字符串的结尾追加n个字符。

```
1 | char * strncat(char *dest, const char *src, size_t n);
```

strcat()函数主要用来将两个char类型连接。例如：

```
1 | char d[20]="Golden";
2 | char s[20]="view";
3 | strcat(d,s);
4 | //打印d
5 | printf("%s",d);
```

输出 d 为 GoldenView（中间无空格）

延伸：

memcpy拷贝函数，它与strcpy的区别就是memcpy可以拷贝任意类型的数据，strcpy只能拷贝字符串类型。

memcpy 函数用于把资源内存（src所指向的内存区域）拷贝到目标内存（dest所指向的内存区域）；有一个size变量控制拷贝的字节数；

函数原型：

```
1 | void *memcpy(void *dest, void *src, unsigned int count);
```

5、static的用法（定义和用途）（必考）

1) 用static修饰局部变量：使其变为静态存储方式(静态数据区)，那么这个局部变量在函数执行完成之后不会被释放，而是继续保留在内存中。

2) 用static修饰全局变量：使其只在本文件内部有效，而其他文件不可连接或引用该变量。

3) 用static修饰函数：对函数的连接方式产生影响，使得函数只在本文件内部有效，对其他文件是不可见的（这一点在大工程中很重要很重要，避免很多麻烦，很常见）。这样的函数又叫作静态函数。使用静态函数的好处是，不用担心与其他文件的同名函数产生干扰，另外也是对函数本身的一种保护机制。

6、const的用法（定义和用途）（必考）

const主要用来修饰变量、函数形参和类成员函数：

1) 用const修饰常量：定义时就初始化，以后不能更改。

2) 用const修饰形参：func(const int a){};该形参在函数里不能改变

3) 用const修饰类成员函数：该函数对成员变量只能进行只读操作，就是const类成员函数是不能修改成员变量的数值的。

被const修饰的东西都受到强制保护，可以预防意外的变动，能提高程序的健壮性。

参考一个大佬的回答：

我只要一听到被面试者说："const意味着常数"，我就知道我正在和一个业余者打交道。去年Dan Saks已经在他的文章里完全概括了const的所有用法，因此ESP(译者：Embedded Systems Programming)的每一位读者应该非常熟悉const能做什么和不能做什么。如果你从没有读到那篇文章，只要能说出const意味着"只读"就可以了。尽管这个答案不是完全的答案，但我接受它作为一个正确的答案。如果应试者能正确回答这个问题，我将问他一个附加的问题：下面的声明都是什么意思？

```
1  const int a;
2  int const a;
3  const int *a;
4  int * const a;
5  int const * a const;
```

前两个的作用是一样，a是一个常整型数。

第三个意味着a是一个指向常整型数的指针（也就是，整型数是不可修改的，但指针可以）。

第四个意思a是一个指向整型数的常指针（也就是说，指针指向的整型数是可以修改的，但指针是不可修改的）。

最后一个意味着a是一个指向常整型数的常指针（也就是说，指针指向的整型数是不可修改的，同时指针也是不可修改的）。

7、volatile作用和用法

一个定义为volatile的变量是说这变量可能会被意想不到地改变，这样，编译器就不会去假设这个变量的值了。精确地说就是，优化器在用到这个变量时必须每次都小心地重新读取这个变量在内存中的值，而不是使用保存在寄存器里的备份（虽然读写寄存器比读写内存快）。

回答不出这个问题的人是不会被雇佣的。这是区分C程序员和嵌入式系统程序员的最基本的问题。搞嵌入式的小伙伴们经常同硬件、中断、RTOS等等打交道，所有这些都要求用到volatile变量。不懂得volatile的内容将会带来灾难。

以下几种情况都会用到volatile：

- 1、并行设备的硬件寄存器（如：状态寄存器）
- 2、一个中断服务子程序中会访问到的非自动变量
- 3、多线程应用中被几个任务共享的变量

8、const常量和#define的区别（编译阶段、安全性、内存占用等）

用#define max 100；定义的常量是没有类型的（不进行类型安全检查，可能会产生意想不到的错误），所给出的是一个立即数，编译器只是把所定义的常量值与所定义的常量的名字联系起来，define所定义的宏变量在预处理阶段的时候进行替换，在程序中使用到该常量的地方都要进行拷贝替换；

用const int max = 255；定义的常量有类型（编译时会进行类型检查）名字，存放在内存的静态区域中，在编译时确定其值。在程序运行过程中const变量只有一个拷贝，而#define所定义的宏变量却有多多个拷贝，所以宏定义在程序运行过程中所消耗的内存要比const变量的大得多

9、变量的作用域（全局变量和局部变量）

全局变量：在所有函数体的外部定义的，程序的所在部分（甚至其它文件中的代码）都可以使用。全局变量不受作用域的影响（也就是说，全局变量的生命期一直到程序的结束）。

局部变量：出现在一个作用域内，它们是局限于一个函数的。局部变量经常被称为自动变量，因为它们在进入作用域时自动生成，离开作用域时自动消失。关键字auto可以显式地说明这个问题，但是局部变量默认为auto，所以没有必要声明为auto。

局部变量可以和全局变量重名，在局部变量作用域范围内，全局变量失效，采用的是局部变量的值。

10、sizeof 与 strlen（字符串，数组）

1.如果是数组

```
1  #include<stdio.h>
2  int main()
3  {
4      int a[5]={1,2,3,4,5};
5      printf("sizeof 数组名=%d\n",sizeof(a));
6      printf("sizeof *数组名=%d\n",sizeof(*a));
7  }
```

运行结果

```
1  sizeof 数组名=20
2  sizeof *数组名=4
```

2.如果是指针，sizeof只会检测到是指针的类型，指针都是占用4个字节的空间（32位机）。

sizeof是什么？是一个操作符,也是关键字，就不是一个函数，这和strlen()不同，strlen()是一个函数。

那么sizeof的作用是什么？返回一个对象或者类型所占的内存字节数。我们会对sizeof()中的数据或者指针做运算吗？基本不会。例如sizeof(1+2.0),直接检测到其中类型是double,即是sizeof(double) = 8。如果是指针，sizeof只会检测到是指针的类型，指针都是占用4个字节的空间（32位机）。

```
1  char *p = "sadasdasd";
2  sizeof(p):4
3  sizeof(*p):1//指向一个char类型的
```

除非使用strlen(), 仅对字符串有效，直到'\0'为止了，计数结果不包括\0。

要是非要使用sizeof来得到指向内容的大小，就得使用数组名才行，如

```
1  char a[10];
2  sizeof(a):10 //检测到a是一个数组的类型。
```

数据类型	32位机 (字节)	64位机 (字节)	备注
char	1	1	
short	2	2	
int	4	4	
long	4	8	32位与64位不同
float	4	4	
char *	4	8	其他指针类型如long *, int * 也是如此
long long	8	8	
double	8	8	
long double	10/12	10/16	有效位10字节。32位为了对齐实际分配12字节；64位分配16字节。

关于strlen(), 它是一个函数, 考察的比较简单:

```
1 | strlen "\n\t\ttag\AAtang"
```

答案：11

11、经典的sizeof(struct)和sizeof(union)内存对齐

内存对齐作用：

- 1.平台原因(移植原因):不是所有的硬件平台都能访问任意地址上的任意数据的;某些硬件平台只能在某些地址处取某些特定类型的数据,否则抛出硬件异常。
- 2.性能原因:数据结构(尤其是栈)应该尽可能地在自然边界上对齐。原因在于,为了访问未对齐的内存,处理器需要作两次内存访问;而对齐的内存访问仅需要一次访问。

结构体struct内存对齐的3大规则:

- 1.对于结构体的各个成员，第一个成员的偏移量是0，排列在后面的成员其当前偏移量必须是当前成员类型的整数倍；
- 2.结构体内所有数据成员各自内存对齐后，结构体本身还要进行一次内存对齐，保证整个结构体占用内存大小是结构体内最大数据成员的最小整数倍；
- 3.如程序中有`#pragma pack(n)`预编译指令，则所有成员对齐以n字节为准(即偏移量是n的整数倍)，不再考虑当前类型以及最大结构体内类型。

```
1 #pragma pack(1)
2
3 struct fun{
4     int i;
5     double d;
6     char c;
7 };
```

sizeof(fun) = 13

```
1 struct CAT_s
2 {
3     int ld;
4     char color;
5     unsigned short Age;
6     char *Name;
7     void(*Jump)(void);
8 }Garfield;
```

1.使用32位编译，int占4，char占1，unsigned short占2，char*占4，函数指针占4个，由于是32位编译是4字节对齐，所以该结构体占16个字节。（说明：按几字节对齐，是根据结构体的最长类型决定的，这里是int是最长的字节，所以按4字节对齐）；

2.使用64位编译，int占4，char占1，unsigned short占2，char*占8，函数指针占8个，由于是64位编译是8字节对齐（说明：按几字节对齐，是根据结构体的最长类型决定的，这里是函数指针是最长的字节，所以按8字节对齐）所以该结构体占24个字节。

```
1 //64位
2 struct C
3 {
4     double t; //8 1111 1111
5     char b; //1 1
6     int a; //4 0001111
7     short c; //2 11000000
8 };
9 sizeof(C) = 24; //注意：1 4 2 不能拼在一起
```

char是1，然后在int之前，地址偏移量得是4的倍数，所以char后面补三个字节，也就是char占了4个字节，然后int四个字节，最后是short，只占两个字节，但是总的偏移量得是double的倍数，也就是8的倍数，所以short后面补六个字节

联合体union内存对齐的2大规则:

1.找到占用字节最多的成员；

2.union的字节数必须是占用字节最多的成员的字节的倍数，而且需要能够容纳其他的成员

```
1 //x64
2 typedef union {
3     long i;
4     int k[5];
5     char c;
6 }D
```

要计算union的大小,首先要找到占用字节最多的成员,本例中是long,占用8个字节,int k[5]中都是int类型,仍然是占用4个字节的，然后union的字节数必须是占用字节最多的成员的字节的倍数,而且需要能够容纳其他的成员,为了要容纳k(20个字节),就必须要保证是8的倍数的同时还要大于20个字节,所以是24个字节。

引申：位域（大疆笔试题）：

C语言允许在一个结构体中以位为单位来指定其成员所占内存长度，这种以位为单位的成员称为“位段”或称“位域”(bit field)。利用位段能够用较少的位数存储数据。一个位段必须存储在同一存储单元中，不能跨两个单元。如果第一个单元空间不能容纳下一个位段，则该空间不用，而从下一个单元起存放该位段。

1.位段声明和结构体类似

2.位段的成员必须是int、unsigned int、signed int

3.位段的成员名后边有一个冒号和一个数字

```
1  typedef struct_data{
2      char m:3;
3      char n:5;
4      short s;
5
6      union{
7          int a;
8          char b;
9      };
10
11     int h;
12 }_attribute__((packed)) data_t;
```

答案12

m和n一起，刚好占用一个字节内存，因为后面是short类型变量，所以在short s之前，应该补一个字节。所以m和n其实是占了两个字节的，然后是short两个字节，加起来就4个字节，然后联合体占了四个字节，总共8个字节了，最后int h占了四个字节，就是12个字节了

`attribute((packed))` 取消对齐

GNU C的一大特色就是**attribute**机制。**attribute**可以设置函数属性（Function Attribute）、变量属性（Variable Attribute）和类型属性（Type Attribute）。

attribute书写特征是：**attribute**前后都有两个下划线，并且后面会紧跟一对括弧，括弧里面是相应的**attribute**参数。

跨平台通信时用到。不同平台内存对齐方式不同。如果使用结构体进行平台间的通信，会有问题。例如，发送消息的平台上，结构体为24字节，接受消息的平台上，此结构体为32字节（只是随便举个例子），那么每个变量对应的值就不对了。

不同框架的处理器对齐方式会有不同，这个时候不指定对齐的话，会产生错误结果

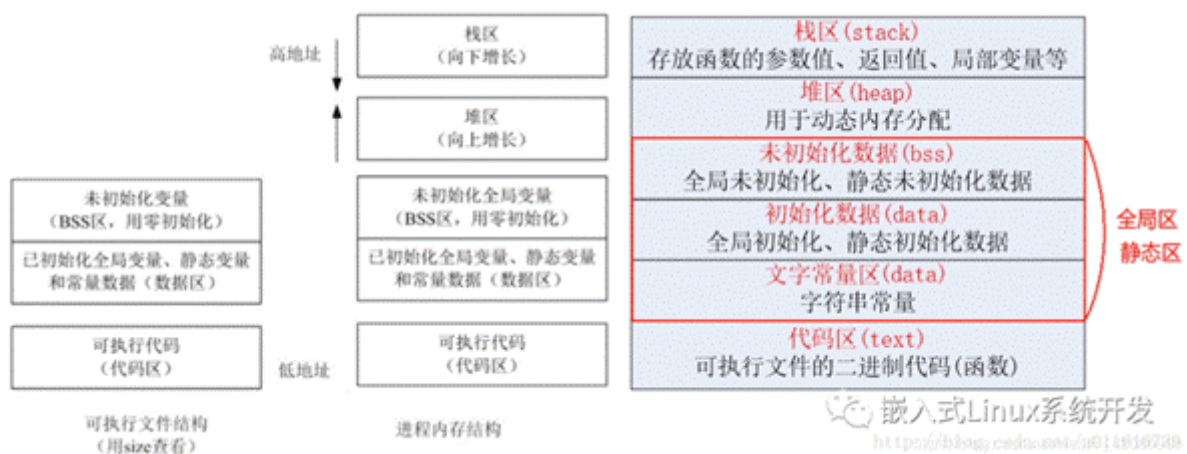
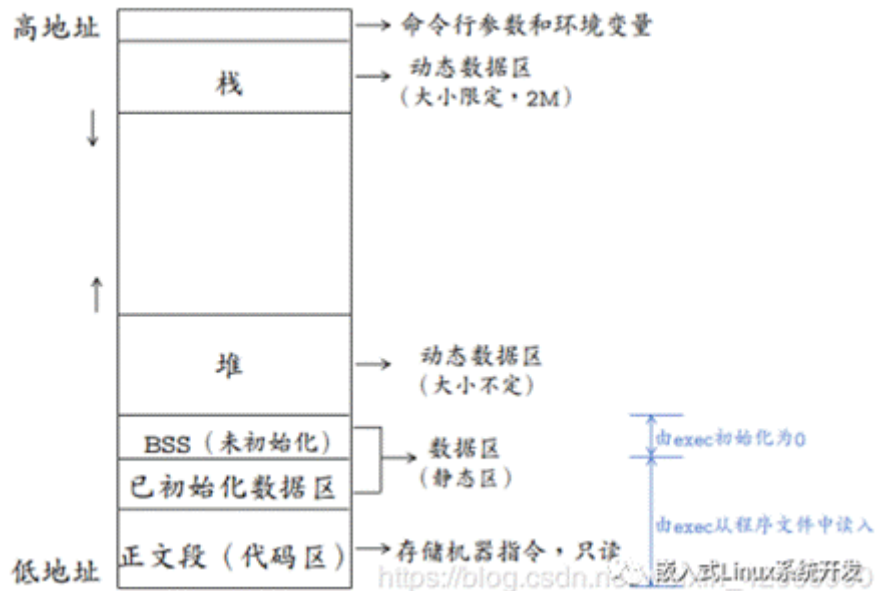
12、inline函数

在C语言中，如果一些函数被频繁调用，不断地有函数入栈，即函数栈，会造成栈空间或栈内存的大量消耗。为了解决这个问题，特别的引入了inline修饰符，表示为内联函数。

大多数的机器上，调用函数都要做很多工作：调用前要先保存寄存器，并在返回时恢复，复制实参，程序还必须转向一个新位置执行C++中支持内联函数，其目的是为了提高函数的执行效率，用关键字 inline 放在函数定义(注意是定义而非声明)的前面即可将函数指定为内联函数，内联函数通常就是将它程序中的每个调用点上“内联地”展开。

内联是以代码膨胀（复制）为代价，仅仅省去了函数调用的开销，从而提高函数的执行效率。

13、内存四区，什么变量分别存储在什么区域，堆上还是栈上。



文字常量区, 叫.rodata, 不可以改变, 改变会导致段错误

```

1  int a0=1;
2  static int a1;
3  const static a2=0;
4  extern int a3;
5
6  void fun(void)
7  {
8      int a4;
9      volatile int a5;
10     return;
11 }

```

a0 : 全局初始化变量 ; 生命周期为整个程序运行期间 ; 作用域为所有文件 ; 存储位置为data段。

a1 : 全局静态未初始化变量 ; 生命周期为整个程序运行期间 ; 作用域为当前文件 ; 储存位置为BSS段。

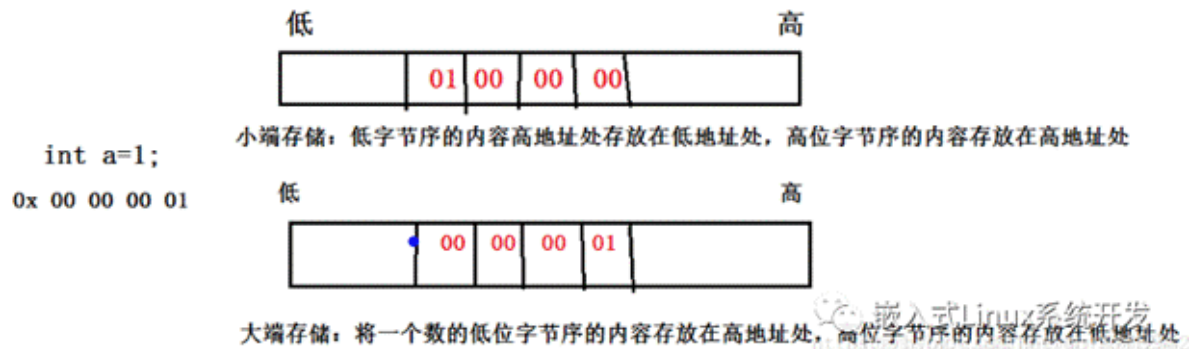
a2 : 全局静态变量

a3 : 全局初始化变量 ; 其他同a0。

a4 : 局部变量 ; 生命周期为fun函数运行期间 ; 作用域为fun函数内部 ; 储存位置为栈。

a5 : 局部易变变量 ;

14、使用32位编译情况下, 给出判断所使用机器大小端的方法。



联合体方法判断方法：利用union结构体的从低地址开始存，且同一时间内只有一个成员占有内存的特性。大端储存符合阅读习惯。联合体占用内存是最大的那个，和结构体不一样。

a和c公用同一片内存区域，所以更改c，必然会影响a的数据

```
1 #include<stdio.h>
2
3 int main(){
4     union w
5     {
6         int a;
7         char b;
8     }c;
9     c.a = 1;
10    if(c.b == 1)
11        printf("小端存储\n");
12    else
13        printf("大端存储\n");
14    return 0;
15 }
```

指针方法

通过将int强制类型转换成char单字节，p指向a的起始字节（低字节）

```
1 #include <stdio.h>
2 int main ()
3 {
4     int a = 1;
5     char *p = (char *)&a;
6     if(*p == 1)
7     {
8         printf("小端存储\n");
9     }
10    else
11    {
12        printf("大端存储\n");
13    }
14    return 0;
15 }
```

15、用变量a给出下面的定义

- ```
1 a) 一个整型数;
2 b) 一个指向整型数的指针;
```

```

3 c) 一个指向指针的指针，它指向的指针是指向一个整型数；
4 d) 一个有10个整型的数组；
5 e) 一个有10个指针的数组，该指针是指向一个整型数；
6 f) 一个指向有10个整型数数组的指针；
7 g) 一个指向函数的指针，该函数有一个整型参数并返回一个整型数；
8 h) 一个有10个指针的数组，该指针指向一个函数，该函数有一个整型参数并返回一个整型数
9 答案：
10 a) int a
11 b) int *a;
12 c) int **a;
13 d) int a[10];
14 e) int *a [10];
15 f) int a[10], *p=a;
16 g) int (*a)(int)
17 h) int(*a[10])(int)

```

## 16、与或非，异或。运算符优先级

sum=a&b<<c+a^c;

其中a=3,b=5,c=4 (先加再移位再&再异或) 答案4

| 优先级 | 运算符                                          | 结合性  |
|-----|----------------------------------------------|------|
| 1   | () [] .                                      | 从左到右 |
| 2   | ! + (正) - (负) ~ ++ --                        | 从右向左 |
| 3   | * / %                                        | 从左向右 |
| 4   | + (加) - (减)                                  | 从左向右 |
| 5   | << >> >>>                                    | 从左向右 |
| 6   | < <= > >= instanceof                         | 从左向右 |
| 7   | == !=                                        | 从左向右 |
| 8   | & (按位与)                                      | 从左向右 |
| 9   | ^                                            | 从左向右 |
| 10  |                                              | 从左向右 |
| 11  | &&                                           | 从左向右 |
| 12  |                                              | 从左向右 |
| 13  | ?:                                           | 从右向左 |
| 14  | = += -= *= /= %<br>= &=  = ^= ~< <= >>= >>>= | 从右向左 |

### 3.1.1.2 C++语言基础

最喜欢问的莫过于strlen与sizeof的区别、explicit关键字、mutable关键字、指针和引用、public、protected、private三者继承情况下的一些访问权限、菱形继承、友元函数等。这些随便一本基础的C++书籍都会讲到。

### 3.1.2 数据结构与算法

### 3.1.2.1十大排序

我想对于每一个经历过秋招的小伙伴们来说，十大排序基本都被问过（快速排序、归并排序、堆排序、冒泡排序、插入排序、选择排序、希尔排序、桶排序、基数排序）。

对这十大排序的考察主要有两点：

1、考察时间复杂度、空间复杂度、稳定与否。

2、手写某种排序。

第一点：对于时间复杂度的考察，可能会考察插入排序的平均复杂度是多少？最坏和最好复杂度又是多少？有时候也会从别的角度来对你进行考察，直接会问你了解到的排序中哪些排序是稳定的？哪些不是稳定的？

第二点：快速排序手写次数绝对占据第一名，因为现在企业招聘基本都是有代码要求的，有时候面试官可能也拿不准让你写什么算法题，“算了，写个快排吧”，快排的频率就是就是这样被拉高的；第二高频率的应该就是归并排序了。在笔者秋招过程中，手写过5次归并，3次快排。因为归并是刚好比快排难度大一点，但也不是那种特别难的排序方法。对于女生来说，让手写的排序一般是冒泡排序、快速排序、归并排序，对于男生同学而言，要求手写的排序一般有快速排序、归并排序以及堆排序。

十大排序中考察最多的就是冒泡排序、快速排序、归并排序、堆排序以及可能会出现的桶排序和基数排序了，其余排序出现的概率稍小一点。

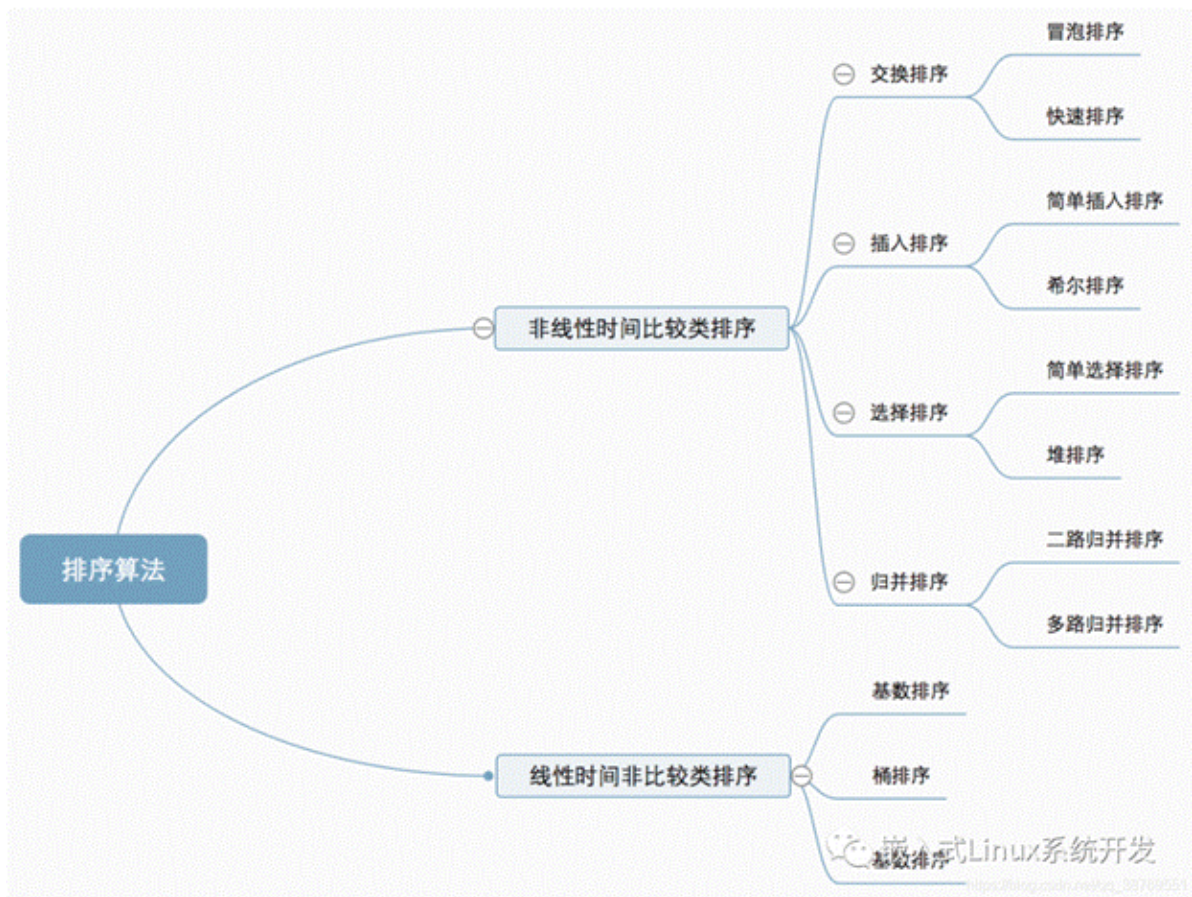
### 3.1.2.2算法

嵌入式考察算法大多都是双向链表、二叉树、字符串翻转和复制这些普通题目。刷题可以在LeetCode、牛客网、杭电OJ等。

十种常见排序算法可以分为两大类：

非线性时间比较类排序：通过比较来决定元素间的相对次序，由于其时间复杂度不能突破 $O(n\log n)$ ，因此称为非线性时间比较类排序。

线性时间非比较类排序：不通过比较来决定元素间的相对次序，它可以突破基于比较排序的时间下界，以线性时间运行，因此称为线性时间非比较类排序。



算法优劣评价术语：

稳定性:

稳定：如果  $a$  原本在  $b$  前面，而  $a = b$ ，排序之后  $a$  仍然在  $b$  的前面；

不稳定：如果  $a$  原本在  $b$  的前面，而  $a = b$ ，排序之后  $a$  可能会出现在  $b$  的后面；

排序方式:

内排序：所有排序操作都在内存中完成，占用常数内存，不占用额外内存。

外排序：由于数据太大，因此把数据放在磁盘中，而排序通过磁盘和内存的数据传输才能进行，占用额外内存。

复杂度:

时间复杂度: 一个算法执行所耗费的时间。

空间复杂度: 运行完一个程序所需内存的大小。



| 排序方法 | 最好时间         | 平均时间          | 最坏时间         | 辅助空间       | 稳定性 |
|------|--------------|---------------|--------------|------------|-----|
| 直接插入 | $O(n)$       | $O(n^2)$      | $O(n^2)$     | $O(1)$     | 稳定  |
| 二分插入 | $O(n)$       | $O(n^2)$      | $O(n^2)$     | $O(1)$     | 稳定  |
| 希 尔  |              | $O(n^{1.25})$ |              | $O(1)$     | 不稳定 |
| 冒 泡  | $O(n)$       | $O(n^2)$      | $O(n^2)$     | $O(1)$     | 稳定  |
| 快 速  | $O(n \lg n)$ | $O(n \lg n)$  | $O(n^2)$     | $O(\lg n)$ | 不稳定 |
| 直接选择 | $O(n^2)$     | $O(n^2)$      | $O(n^2)$     | $O(1)$     | 不稳定 |
| 堆    | $O(n \lg n)$ | $O(n \lg n)$  | $O(n \lg n)$ |            | 不稳定 |
| 归 并  | $O(n \lg n)$ | $O(n \lg n)$  | $O(n \lg n)$ | $O(n)$     | 稳定  |
| 基 数  | $O(d(rd+n))$ | $O(d(rd+n))$  | $O(d(rd+n))$ | $O(n)$     | 稳定  |

| 排序算法 | 平均时间复杂度         | 最好情况               | 最坏情况               | 空间复杂度       | 排序方式 | 稳定性 |
|------|-----------------|--------------------|--------------------|-------------|------|-----|
| 冒泡排序 | $O(n^2)$        | $O(n)$             | $O(n^2)$           | $O(1)$      | 内排序  | 稳定  |
| 选择排序 | $O(n^2)$        | $O(n^2)$           | $O(n^2)$           | $O(1)$      | 内排序  | 不稳定 |
| 插入排序 | $O(n^2)$        | $O(n)$             | $O(n^2)$           | $O(1)$      | 内排序  | 稳定  |
| 希尔排序 | $O(n \log n)$   | $O(n(\log_2^2 n))$ | $O(n(\log_2^2 n))$ | $O(1)$      | 内排序  | 不稳定 |
| 归并排序 | $O(n \log n)$   | $O(n \log n)$      | $O(n \log n)$      | $O(n)$      | 外排序  | 稳定  |
| 快速排序 | $O(n \log n)$   | $O(n \log n)$      | $O(n^2)$           | $O(\log n)$ | 内排序  | 不稳定 |
| 堆排序  | $O(n \log n)$   | $O(n \log n)$      | $O(n \log n)$      | $O(1)$      | 内排序  | 不稳定 |
| 计数排序 | $O(n + k)$      | $O(n + k)$         | $O(n + k)$         | $O(k)$      | 外排序  | 稳定  |
| 桶排序  | $O(n + k)$      | $O(n + k)$         | $O(n^2)$           | $O(n + k)$  | 外排序  | 稳定  |
| 基数排序 | $O(n \times k)$ | $O(n \times k)$    | $O(n \times k)$    | $O(n + k)$  | 外排序  | 稳定  |

至于各种算法的原理以及代码实现，由于太多并且比较复杂，不在本文列出。但推荐两本入门的书：《啊哈！算法》、《大话数据结构》。

排序算法很多，嵌入式要求的不会太多，你会冒泡排序、快速排序、插入排序就可以解决很多问题。难的比如动态规划问题，图的路径问题，嵌入式考的比较少，纯软才会考这些。（大公司和独角兽公司考的会相对难一些）



### 3.1.3 操作系统

#### 1、什么是进程、线程，有什么区别？

进程是资源（CPU、内存等）分配的基本单位，线程是CPU调度和分配的基本单位（程序执行的最小单位）。同一时间，如果CPU是单核，只有一个进程在执行，所谓的并发执行，也是顺序执行，只不过由于切换速度太快，你以为这些进程在同步执行而已。多核CPU可以同一时间点有多个进程在执行。

#### 2、多进程、多线程的优缺点

说明：一个进程由进程控制块、数据段、代码段组成，进程本身不可以运行程序，而是像一个容器一样，先创建出一个主线程，分配给主线程一定的系统资源，这时候就可以在主线程开始实现各种功能。当我们需要实现更复杂的功能时，可以在主线程里创建多个子线程，多个线程在同一个进程里，利用这个进程所拥有的系统资源合作完成某些功能。

优缺点：1）一个进程死了不影响其他进程，一个线程崩溃很可能影响到它本身所处的整个进程。2）创建多进程的系统花销大于创建多线程。3）多进程通讯因为需要跨越进程边界，不适合大量数据的传送，适合小数据或者密集数据的传送。多线程无需跨越进程边界，适合各线程间大量数据的传送。并且多线程可以共享同一进程里的共享内存和变量。

#### 3、什么时候用进程，什么时候用线程

1）创建和销毁较频繁使用线程，因为创建进程花销大。2）需要大量数据传送使用线程，因为多线程切换速度快，不需要跨越进程边界。3）安全稳定选进程；快速频繁选线程；

#### 4、多进程、多线程同步（通讯）的方法

进程间通讯：

（1）有名管道/无名管道（2）信号（3）共享内存（4）消息队列（5）信号量（6）socket

线程通讯（锁）：

（1）信号量（2）读写锁（3）条件变量（4）互斥锁（5）自旋锁

#### 5、进程线程的状态转换图

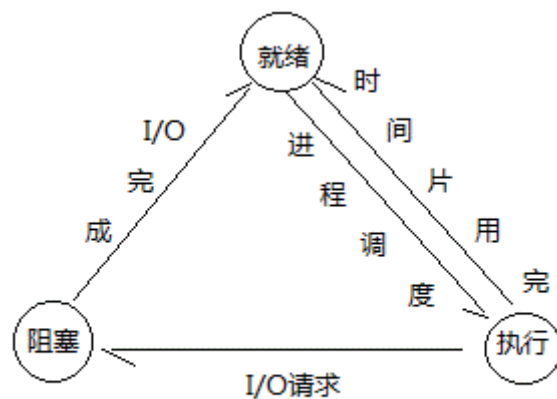
（1）就绪状态：进程已获得除CPU外的所有必要资源，只等待CPU时的状态。一个系统会将多个处于就绪状态的进程排成一个就绪队列。

（2）执行状态：进程已获CPU，正在执行。单处理机系统中，处于执行状态的进程只一个；多处理机系统中，有多个处于执行状态的进程。

（3）阻塞状态：正在执行的进程由于某种原因而暂时无法继续执行，便放弃处理机而处于暂停状态，即进程执行受阻。（这种状态又称等待状态或封锁状态）

通常导致进程阻塞的典型事件有：请求I/O，申请缓冲空间等。

一般，将处于阻塞状态的进程排成一个队列，有的系统还根据阻塞原因不同把这些阻塞集成排成多个队列。



进程的三种基本状态及转换

#### (1) 就绪→执行

处于就绪状态的进程，当进程调度程序为之分配了处理机后，该进程便由就绪状态转变成执行状态。

#### (2) 执行→就绪

处于执行状态的进程在其执行过程中，因分配给它的一个时间片已用完而不得出让出处理机，于是进程从执行状态转变成就绪状态。

#### (3) 执行→阻塞

正在执行的进程因等待某种事件发生而无法继续执行时，便从执行状态变成阻塞状态。

#### (4) 阻塞→就绪

处于阻塞状态的进程，若其等待的事件已经发生，于是进程由阻塞状态转变为就绪状态。

### 6、父进程、子进程

父进程调用fork()以后，克隆出一个子进程，子进程和父进程拥有相同内容的代码段、数据段和用户堆栈。父进程和子进程谁先执行不一定，看CPU。所以我们一般会设置父进程等待子进程执行完毕。

### 7、说明什么是上下文切换？

你可以有很多角度，有进程上下文，有中断上下文。

**进程上下文：**一个进程在执行的时候，CPU的所有寄存器中的值、进程的状态以及堆栈中的内容，当内核需要切换到另一个进程时，它需要保存当前进程的所有状态，即保存当前进程的进程上下文，以便再次执行该进程时，能够恢复切换时的状态，继续执行。

**中断上下文：**由于触发信号，导致CPU中断当前进程，转而去执行另外的程序。那么当前进程的所有资源要保存，比如堆栈和指针。保存过后转而去执行中断处理程序，快速执行完毕返回，返回后恢复上一个进程的资源，继续执行。这就是中断的上下文。

## 3.1.4 计算机网络

对于计算机网络的考察，并不会仔细问OSI七层模型中的每一层，只会挑选比较重要的应用层、传输层、网络层来进行考察，其余层偶尔也会有所涉及，但频率较小。但这并不意味着你可以不去看其它层的知识，因为计网知识是具有相关性的，OSI七层模型，上一层要用到下一层的数据，它们是层层相关的。切记切记！

### 1、TCP、UDP的区别

TCP---传输控制协议,提供的是面向连接、可靠的字节流服务。当客户和服务器彼此交换数据前，必须先在双方之间建立一个TCP连接，之后才能传输数据。

UDP---用户数据报协议，是一个简单的面向数据报的运输层协议。UDP不提供可靠性，它只是把应用程序传给IP层的数据报发送出去，但是并不能保证它们能到达目的地。

- 1) TCP是面向连接的，UDP是面向无连接的
- 2) UDP程序结构较简单
- 3) TCP是面向字节流的，UDP是基于数据报的
- 4) TCP保证数据正确性，UDP可能丢包
- 5) TCP保证数据顺序到达，UDP不保证

## 2、TCP、UDP的优缺点

TCP优点：可靠稳定

TCP的可靠体现在TCP在传输数据之前，会有三次握手来建立连接，而且在数据传递时，有确认、窗口、重传、拥塞控制机制，在数据传完之后，还会断开来连接用来节约系统资源。

TCP缺点：慢，效率低，占用系统资源高，易被攻击

在传递数据之前要先建立连接，这会消耗时间，而且在数据传递时，确认机制、重传机制、拥塞机制等都会消耗大量时间，而且要在每台设备上维护所有的传输连接。然而，每个连接都会占用系统的CPU，内存等硬件资源。因为TCP有确认机制、三次握手机制，这些也导致TCP容易被利用，实现DOS、DDOS、CC等攻击。

UDP优点：快，比TCP稍安全

UDP没有TCP拥有的各种机制，是一种无状态的传输协议，所以传输数据非常快，没有TCP的这些机制，被攻击利用的机会就少一些，但是也无法避免被攻击。

UDP缺点：不可靠，不稳定

因为没有TCP的这些机制，UDP在传输数据时，如果网络质量不好，就会很容易丢包，造成数据的缺失。

## 3、TCP UDP适用场景

TCP：传输一些对信号完整性，信号质量有要求的信息。

UDP：对网络通讯质量要求不高时，要求网络通讯速度要快的场景。

## 4、TCP为什么是可靠连接？

因为tcp传输的数据满足3大条件，不丢失，不重复，按顺序到达。

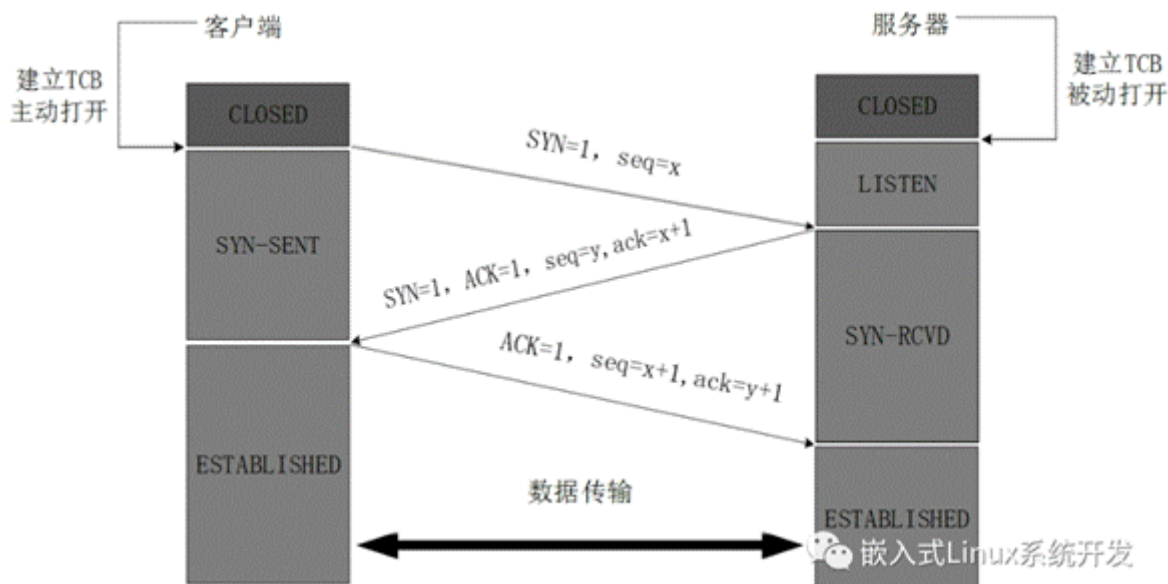
## 5、OSI典型网络模型，简单说说有哪些

| ISO/OSI 模型 |                              |                        | TCP/IP 协议                 |                       |                      |       | TCP/IP 模型 |
|------------|------------------------------|------------------------|---------------------------|-----------------------|----------------------|-------|-----------|
| 应用层        | 文件传输<br>协议<br>(FTP)          | 远程登录<br>协议<br>(Telnet) | 电子邮件<br>协议<br>(SMTP)      | 网络文件服<br>务协议<br>(NFS) | 网络管理<br>协议<br>(SNMP) | 应用层   |           |
| 表示层        |                              |                        |                           |                       |                      |       |           |
| 会话层        |                              |                        |                           |                       |                      |       |           |
| 传输层        | TCP                      UDP |                        |                           |                       |                      | 传输层   |           |
| 网络层        | IP                           | ICMP                   | ARP      RARP             |                       |                      | 网际层   |           |
| 数据链路层      | Ethernet<br>IEEE 802.3       | FDDI                   | Token-Ring/<br>IEEE 802.5 | ARCnet                | PPP/SLIP             | 网络接口层 |           |
| 物理层        |                              |                        |                           |                       |                      | 硬件层   |           |

TCP/IP 模型与 OSI 模型的对比

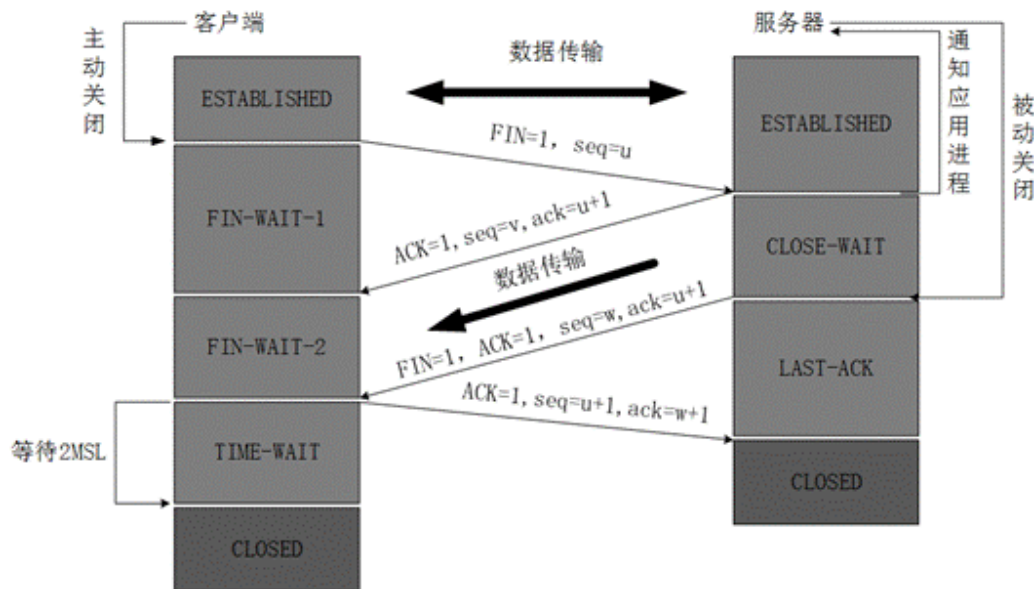
## 6、三次握手、四次挥手

三次握手：



- 1、TCP服务器进程先创建传输控制块TCB，时刻准备接受客户进程的连接请求，此时服务器就进入了LISTEN（监听）状态；
- 2、TCP客户进程也是先创建传输控制块TCB，然后向服务器发出连接请求报文，这是报文首部中的同部位SYN=1，同时选择一个初始序列号 seq=x，此时，TCP客户端进程进入了 SYN-SENT（同步已发送状态）状态。TCP规定，SYN报文段（SYN=1的报文段）不能携带数据，但需要消耗掉一个序号。
- 3、TCP服务器收到请求报文后，如果同意连接，则发出确认报文。确认报文中应该 ACK=1，SYN=1，确认号是ack=x+1，同时也要为自己初始化一个序列号 seq=y，此时，TCP服务器进程进入了SYN-RCVD（同步收到）状态。这个报文也不能携带数据，但是同样要消耗一个序号。
- 4、TCP客户进程收到确认后，还要向服务器给出确认。确认报文的ACK=1，ack=y+1，自己的序列号 seq=x+1，此时，TCP连接建立，客户端进入ESTABLISHED（已建立连接）状态。TCP规定，ACK报文段可以携带数据，但是如果不携带数据则不消耗序号。
- 5、当服务器收到客户端的确认后也进入ESTABLISHED状态，此后双方就可以开始通信了。

四次挥手：



- 1、客户端进程发出连接释放报文，并且停止发送数据。释放数据报文首部，FIN=1，其序列号为seq=u（等于前面已经传送过来的数据的最后一个字节的序号加1），此时，客户端进入FIN-WAIT-1（终止等待1）状态。TCP规定，FIN报文段即使不携带数据，也要消耗一个序号。
- 2、服务器收到连接释放报文，发出确认报文，ACK=1，ack=u+1，并且带上自己的序列号seq=v，此时，服务端就进入了CLOSE-WAIT（关闭等待）状态。TCP服务器通知高层的应用进程，客户端向服务器的方向就释放了，这时候处于半关闭状态，即客户端已经没有数据要发送了，但是服务器若发送数据，客户端依然要接受。这个状态还要持续一段时间，也就是整个CLOSE-WAIT状态持续的时间。
- 3、客户端收到服务器的确认请求后，此时，客户端就进入FIN-WAIT-2（终止等待2）状态，等待服务器发送连接释放报文（在这之前还需要接受服务器发送的最后的的数据）。
- 4、服务器将最后的数据发送完毕后，就向客户端发送连接释放报文，FIN=1，ack=u+1，由于在半关闭状态，服务器很可能又发送了一些数据，假定此时的序列号为seq=w，此时，服务器就进入了LAST-ACK（最后确认）状态，等待客户端的确认。
- 5、客户端收到服务器的连接释放报文后，必须发出确认，ACK=1，ack=w+1，而自己的序列号是seq=u+1，此时，客户端就进入了TIME-WAIT（时间等待）状态。注意此时TCP连接还没有释放，必须经过 $2 \times \text{MSL}$ （最长报文段寿命）的时间后，当客户端撤销相应的TCB后，才进入CLOSED状态。
- 6、服务器只要收到了客户端发出的确认，立即进入CLOSED状态。同样，撤销TCB后，就结束了这次的TCP连接。可以看到，服务器结束TCP连接的时间要比客户端早一些。

### 3.1.6 Linux

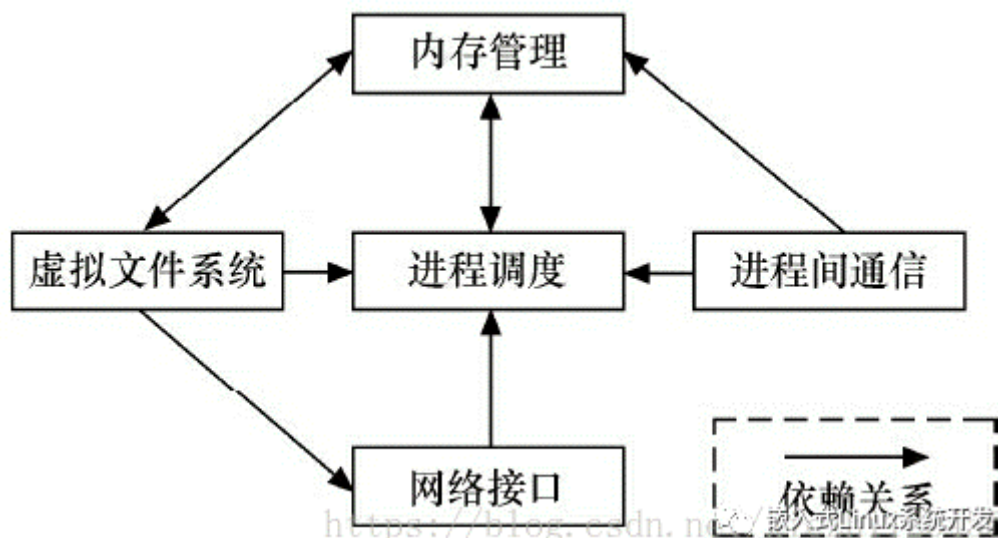
校招过程中对于Linux的要求远远没有社招高，一般只是会问一些简单的命令，基本不会涉及到Linux内核源码这些东西，除非你的岗位是Linux开发工程师这样的。而在Linux命令中考察比较多的是文件处理命令grep、awk、sed这三个命令，如何查看CPU利用如何查看一个进程、如何查看网络情况、远程登录之类的命令，软连接与硬链接也是常问的一个知识点。

常见问题如下：

#### 1、Linux内核的组成部分

Linux内核主要由五个子系统组成：进程调度，内存管理，虚拟文件系统，网络接口，进程间通信。

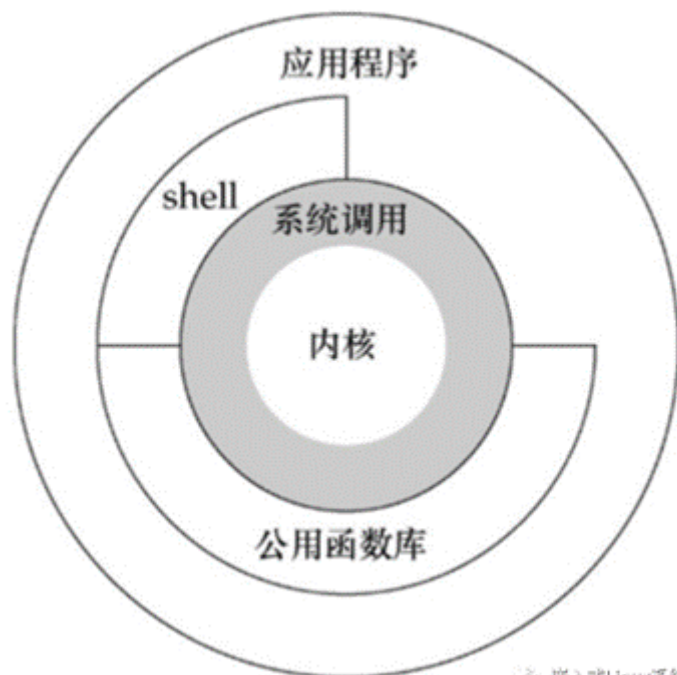


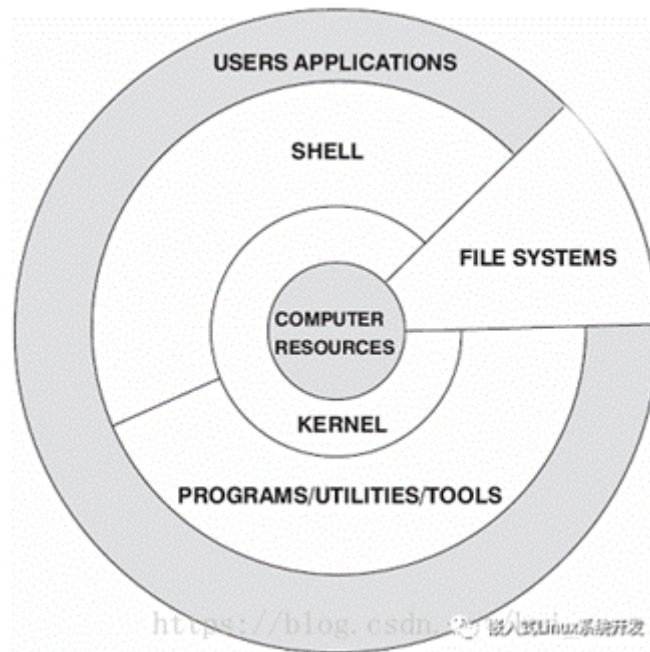


## 2、Linux系统的组成部分

Linux系统一般有4个主要部分：

内核、shell、文件系统和应用程序。





### 3、用户空间与内核通信方式有哪些？

- 1)系统调用。用户空间进程通过系统调用进入内核空间，访问指定的内核空间数据；
- 2)驱动程序。用户空间进程可以使用封装后的系统调用接口访问驱动设备节点，以和运行在内核空间的驱动程序通信；
- 3)共享内存mmap。在代码中调用接口，实现内核空间与用户空间的地址映射，在实时性要求很高的项目中为首选，省去拷贝数据的时间等资源，但缺点是不好控制；
- 4)copy\_to\_user()、copy\_from\_user()，是在驱动程序中调用接口，实现用户空间与内核空间的数据拷贝操作，应用于实时性要求不高的项目中。

以及：

```
1 | procfs(/proc)
2 | sysctl (/proc/sys)
3 | sysfs(/sys)
4 | netlink 套接口
```

### 4、系统调用与普通函数调用的区别

系统调用：

- 1.使用INT和IRET指令，内核和应用程序使用的是不同的堆栈，因此存在堆栈的切换，从用户态切换到内核态，从而可以使用特权指令操控设备
- 2.依赖于内核，不保证移植性
- 3.在用户空间和内核上下文环境间切换，开销较大
- 4.是操作系统的一个入口点

普通函数调用：

- 1.使用CALL和RET指令，调用时没有堆栈切换
- 2.平台移植性好
- 3.属于过程调用，调用开销较小
- 4.一个普通功能函数的调用

## 5、内核态，用户态的区别

内核态，操作系统在内核态运行——运行操作系统程序

用户态，应用程序只能在用户态运行——运行用户程序

当一个进程在执行用户自己的代码时处于用户运行态（用户态），此时特权级最低，为3级，是普通的用户进程运行的特权级，大部分用户直接面对的程序都是运行在用户态。Ring3状态不能访问Ring0的地址空间，包括代码和数据；当一个进程因为系统调用陷入内核代码中执行时处于内核运行态（内核态），此时特权级最高，为0级。执行的内核代码会使用当前进程的内核栈，每个进程都有自己的内核栈。

## 6、bootloader、内核、根文件的关系

启动顺序：bootloader->linux kernel->rootfile->app

Bootloader全名为启动引导程序，是第一段代码，它主要用来初始化处理器及外设，然后调用Linux内核。Linux内核在完成系统的初始化之后需要挂载某个文件系统作为根文件系统（RootFilesystem），然后加载必要的内核模块，启动应用程序。（一个嵌入式Linux系统从软件角度看可以分为四个部分：引导加载程序（Bootloader），Linux内核，文件系统，应用程序。）

## 7、Bootloader启动的两个阶段：

Stage1:汇编语言

- 1) 基本的硬件初始化（关闭看门狗和中断，MMU（带操作系统），CACHE。配置系统工作时钟）
- 2) 为加载stage2准备RAM空间
- 3) 拷贝内核映像和文件系统映像到RAM中
- 4) 设置堆栈指针sp
- 5) 跳到stage2的入口点

Stage2:c语言

- 1) 初始化本阶段要使用到的硬件设备（led uart等）
- 2) 检测系统的内存映射
- 3) 加载内核映像和文件系统映像
- 4) 设置内核的启动参数

嵌入式系统中广泛采用的非易失性存储器通常是Flash，而Bootloader就位于该存储器的最前端，所以系统上电或复位后执行的第一段程序便是Bootloader。

## 8、linux下检查内存状态的命令

- |   |                                   |
|---|-----------------------------------|
| 1 | 1) 查看进程: <code>top</code>         |
| 2 | 2) 查看内存: <code>free</code>        |
| 3 | 3) <code>cat /proc/meminfo</code> |
| 4 | 4) <code>vmstat</code>            |

假如一个公司服务器有很多用户，你使用top命令，可以看到哪个同事在使用什么命令，做什么事情，占用了多少CPU。

## 9、一个程序从开始运行到结束的完整过程（四个过程）

预处理（Pre-Processing）、编译（Compiling）、汇编（Assembling）、链接（Linking）

## 10、什么是堆，栈，内存泄漏和内存溢出？

栈由系统操作，程序员不可以操作。

所以内存泄漏是指堆内存的泄漏。堆内存是指程序从堆中分配的，大小任意的（内存块的大小可以在程序运行期决定），使用完后必须显式释放的内存。应用程序一般使用malloc，new等函数从堆中分配到一块内存，使用完后，程序必须负责相应的调用free或delete释放该内存块，否则，这块内存就不能被再次使用。

内存溢出：你要求分配的内存超出了系统能给你的，系统不能满足需求，于是产生溢出。

内存越界：向系统申请了一块内存，而在使用内存时，超出了申请的范围（常见的有使用特定大小数组时发生内存越界）

内存溢出问题是C语言或者C++语言所固有的缺陷，它们既不检查数组边界，又不检查类型可靠性(type-safety)。众所周知，用C/C++语言开发的程序由于目标代码非常接近机器内核，因而能够直接访问内存和寄存器，这种特性大大提升了C/C++语言代码的性能。只要合理编码，C/C++应用程序在执行效率上必然优于其它高级语言。然而，C/C++语言导致内存溢出问题的可能性也要大许多。

## 11、死锁的原因、条件

产生死锁的原因主要是：

- （1）因为系统资源不足。
- （2）进程运行推进的顺序不合适。
- （3）资源分配不当等。

如果系统资源充足，进程的资源请求都能够得到满足，死锁出现的可能性就很低，否则就会因争夺有限的资源而陷入死锁。其次，进程运行推进顺序与速度不同，也可能产生死锁

这四个条件是死锁的必要条件，只要系统发生死锁，这些条件必然成立，而只要上述条件之一不满足，就不会发生死锁。

- （1）互斥条件：一个资源每次只能被一个进程使用。
- （2）请求与保持条件：一个进程因请求资源而阻塞时，对已获得的资源保持不放。
- （3）不剥夺条件:进程已获得的资源，在未使用完之前，不能强行剥夺。
- （4）循环等待条件:若干进程之间形成一种头尾相接的循环等待资源关系。

## 12、硬链接与软链接

链接操作实际上是给系统中已有的某个文件指定另外一个可用于访问它的名称。对于这个新的文件名，我们可以为之指定不同的访问权限，以控制对信息的共享和安全性的问题。如果链接指向目录，用户就可以利用该链接直接进入被链接的目录而不用打一大堆的路径名。而且，即使我们删除这个链接，也不会破坏原来的目录。

### 1>硬链接

硬链接只能引用同一文件系统中的文件。它引用的是文件在文件系统中的物理索引(也称为inode)。当您移动或删除原始文件时，硬链接不会被破坏，因为它所引用的是文件的物理数据而不是文件在文件结构中的位置。硬链接的文件不需要用户有访问原始文件的权限，也不会显示原始文件的位置，这样有助于文件的安全。如果您删除的文件有相应的硬链接，那么这个文件依然会保留，直到所有对它的引用都被删除。

### 2>软链接（符号链接）

软连接，其实就是新建立一个文件，这个文件就是专门用来指向别的文件的（那就和windows下的快捷方式的那个文件有很接近的意味）。软连接产生的是一个新的文件，但这个文件的作用就是专门指向某个文件的，删了这个软连接文件，那就等于不需要这个连接，和原来的存在的实体原文件没有任何关系，但删除原来的文件，则相应的软连接不可用。

### 13、计算机中，32bit与64bit有什么区别

64bit计算主要有两大优点：可以进行更大范围的整数运算；可以支持更大的内存。

64位操作系统下的虚拟内存空间大小：地址空间大小不是 $2^{32}$ ，也不是 $2^{64}$ ，而一般是 $2^{48}$ 。因为并不需要 $2^{64}$ 那么大的寻址空间，过大的空间只会造成资源的浪费。所以64位Linux一般使用48位表示虚拟空间地址，40位标识物理地址。

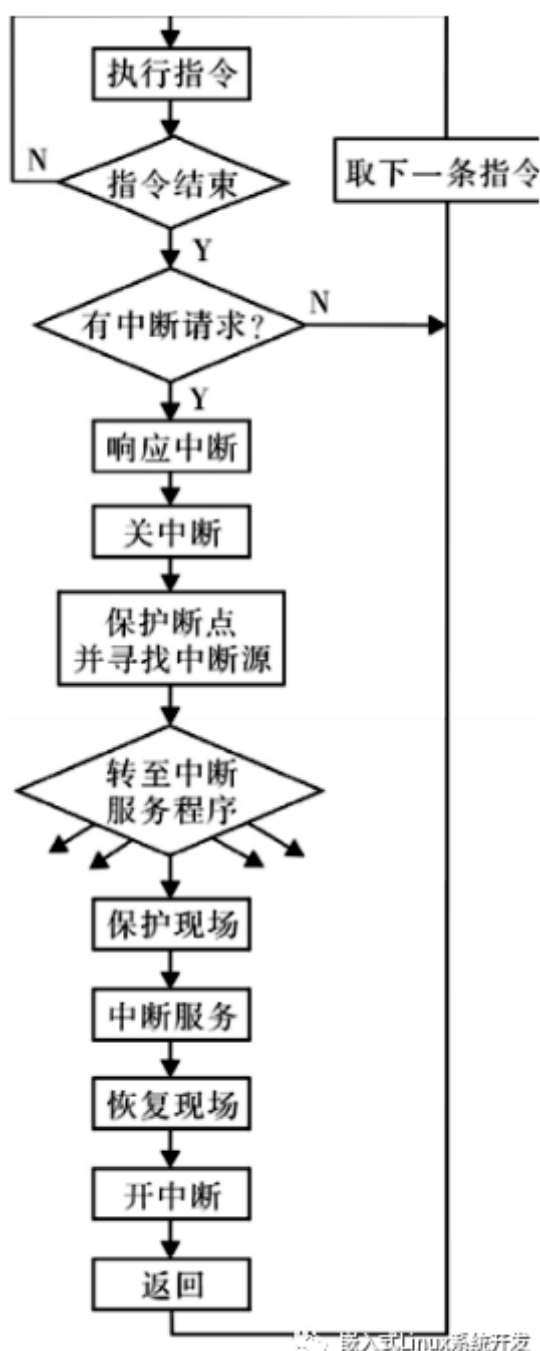
### 14、中断和异常的区别

内中断：同步中断（异常）是由cpu内部的电信号产生的中断，其特点为当前执行的指令结束后才转而产生中断，由于有cpu主动产生，其执行点必然是可控的。

外中断：异步中断是由cpu的外设产生的电信号引起的中断，其发生的时间点不可预期。

### 15、中断怎么发生，中断处理流程

请求中断→响应中断→关闭中断→保留断点→中断源识别→保护现场→中断服务子程序→恢复现场→中断返回。



嵌入式Linux系统开发

### 16\*、Linux 操作系统挂起、休眠、关机相关命令



关机命令有halt , init 0 , poweroff , shutdown -h 时间, 其中shutdown是最安全的

重启命令有reboot , init 6 , shutdown -r时间

在linux命令中reboot是重新启动, shutdown -r now是立即停止然后重新启动

具体可用参数可以百度。

## 17、说一个linux下编译优化选项：

加：-o

## 18、在有数据cache情况下，DMA数据链路为：

外设-DMA-DDR-cache-CPU

## 19、linux命令

1、改变文件属性的命令：chmod (chmod 777 /etc/squid 运行命令后，squid文件夹（目录）的权限就被修改为777（可读可写可执行））

2、查找文件中匹配字符串的命令：grep

3、查找当前目录：pwd

4、删除目录：rm -rf 目录名

5、删除文件：rm 文件名

6、创建目录（文件夹）：mkdir

7、创建文件：touch

8、vi和vim 文件名也可以创建

9、解压：tar -xvzf 压缩包

打包：tar -cvzf 目录（文件夹）

10、查看进程对应的端口号

```
1 1、先查看进程pid
2 ps -ef | grep 进程名
3
4 2、通过pid查看占用端口
5 netstat -nap | grep 进程pid
```

## 20、硬实时系统和软实时系统

软实时系统：

Windows、Linux系统通常为软实时，当然有补丁可以将内核做成硬实时的系统，不过商用没有这么做的。

硬实时系统：

对时间要求很高，限定时间内不管做没做完必须返回。

VxWorks , uCOS , FreeRTOS , WinCE , RT-thread等实时系统；

## 21、MMU基础

现代操作系统普遍采用虚拟内存管理（Virtual Memory Management）机制，这需要MMU（Memory Management Unit，内存管理单元）的支持。有些嵌入式处理器没有MMU，则不能运行依赖于虚拟内存管理的操作系统。

也就是说：操作系统可以分成两类，用MMU的、不用MMU的。

用MMU的是：Windows、MacOS、Linux、Android；不用MMU的是：FreeRTOS、VxWorks、UCOS.....

与此相对应的：CPU也可以分成两类，带MMU的、不带MMU的。

带MMU的是：Cortex-A系列、ARM9、ARM11系列；

不带MMU的是：Cortex-M系列.....（STM32是M系列，没有MMU，不能运行Linux，只能运行一些UCOS、FreeRTOS等等）。

MMU就是负责虚拟地址（virtual address）转化成物理地址（physical address），转换过程比较复杂，可以自行百度。

### 3.1.7 单片机常见面试题

#### 1、IO口工作方式（学过STM32的人应该很熟悉）

上拉输入、下拉输入、推挽输出、开漏输出。

#### 2、请说明总线接口USRT、I2C、USB的异同点

（串/并、速度、全/半双工、总线拓扑等）

| 总线接口 | 串/并 | 同步/异步 | 速率         | 工作方式 | 用线                                 | 总线拓扑结构                                                                                                                                                                                  | 信距离          |
|------|-----|-------|------------|------|------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| UART | 串   | 异步    | 慢<br>波特率设置 | 全双工  | 2线<br>Rx、Tx                        | RS485支持总线式、<br>星形、树形                                                                                                                                                                    | 远<br>最远1200m |
| I2C  | 串   | 同步    | 慢          | 半双工  | 2线<br>SDA、SCL                      | 总线型（特殊的树形）                                                                                                                                                                              | 近            |
| SPI  | 串   | 同步    | 快          | 全双工  | 3线或4线<br>SCLK、SIMO、<br>SOMI、SS(片选) | 环形                                                                                                                                                                                      | 远            |
| USB  | 串   | 同步    | 快          | 半双工  | 4线<br>Vbus（5V）、GND、<br>D+、D-（3.3V） |  嵌入式Linux系统开发<br><a href="https://blog.csdn.net/jq_38769551">https://blog.csdn.net/jq_38769551</a> |              |

#### 3、IIC协议时序图

必须会画出来，我面试被问到过，让我画，我画了个大概。

IIC协议有两根线，一根SCL时钟线，一根SDA数据线，如图可以看到开始信号和结束信号的电平状态。开始后，因为IIC总线可以挂在很多设备（不超过8个），所以先发送一个设备地址，选中这个设备，设备地址最后一位代表了是写还是读。选中设备后，再发送寄存器地址，代表选中某个寄存器，再开始传输数据。

八位设备地址=7位从机地址+读/写地址，

再给地址添加一个方向位用来表示接下来数据传输的方向，

0表示主设备向从设备(write)写数据，

1表示主设备向从设备(read)读数据

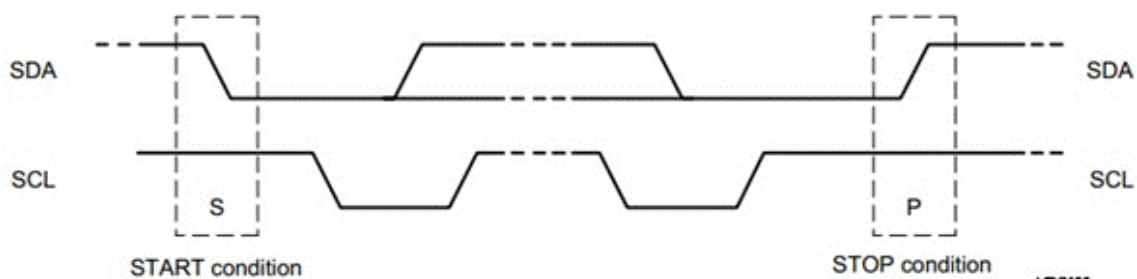


图 5 起始和停止条件

嵌入式Linux系统开发

开始信号：SCL 为高电平时，SDA 由高电平向低电平跳变，开始传送数据。

结束信号：SCL 为高电平时，SDA 由低电平向高电平跳变，结束传送数据。

应答信号：接收数据的 IC 在接收到 8bit 数据后，向发送数据的 IC 发出特定的低电平脉冲，表示已收到数据。CPU 向受控单元发出一个信号后，等待受控单元发出一个应答信号，CPU 接收到应答信号后，根据实际情况作出是否继续传递信号的判断。若未收到应答信号，由判断为受控单元出现故障。

IIC信号在数据传输过程中，当SCL=1高电平时，数据线SDA必须保持稳定状态，不允许有电平跳变，只有在时钟线上的信号为低电平期间，数据线上的高电平或低电平状态才允许变化。SCL=1时 数据线SDA的任何电平变换会看做是总线的起始信号或者停止信号。

#### 4、单片机的SP指针始终指向

栈顶

#### 5、IIC总线在传送数据过程中共有三种类型信号：

它们分别是：开始信号、结束信号和应答信号。

#### 6、FIQ中断向量入口地址：

FIQ和IRQ是两种不同类型的中断，ARM为了支持这两种不同的中断，提供了对应的叫做FIQ和IRQ处理器模式（ARM有7种处理模式）。

FIQ的中断向量地址在0x0000001C，而IRQ的在0x00000018。

#### 7、SPI四种模式，简述其中一种模式，画出时序图

在芯片资料上极性和相位一般表示为CPOL（Clock POLarity）和CPHA(Clock PHase), 极性和相位组合成4种工作模式。

|       | CPOL | CPHA |
|-------|------|------|
| MODE0 | 0    | 0    |
| MODE1 | 0    | 1    |
| MODE2 | 1    | 0    |
| MODE3 | 1    | 1    |

spi四种模式SPI的相位(CPHA)和极性(CPOL)分别可以为0或1，对应的4种组合构成了SPI的4种模式(mode)

Mode 0 CPOL=0, CPHA=0

Mode 1 CPOL=0, CPHA=1

Mode 2 CPOL=1, CPHA=0

Mode 3 CPOL=1, CPHA=1

时钟极性CPOL: 即SPI空闲时, 时钟信号SCLK的电平(1:空闲时高电平; 0:空闲时低电平) 时钟相位

CPHA: 即SPI在SCLK第几个边沿开始采样(0:第一个边沿开始; 1:第二个边沿开始)

sd卡的spi常用的是mode 0 和mode 3, 这两种模式的相同的地方是都在时钟上升沿采样传输数据, 区别这两种方式的简单方法就是看空闲时, 时钟的电平状态, 低电平为mode 0, 高电平为mode 3。

具体的通信过程请自行百度, 2021年秋招大疆笔试题考了这道题。

### 3.1.8 杂项面试题

#### 1、讲一讲冯诺依曼和哈佛体系的区别

哈佛结构是一种将程序指令存储和数据存储分开的存储器结构。目前使用哈佛结构的中央处理器和微控制器有很多, ARM9、ARM10和ARM11, 51单片机属于哈佛结构。

冯·诺伊曼结构也称普林斯顿结构, 是一种将程序指令存储器和数据存储器合并在一起的存储器结构。

#### 2、面向对象编程的三大特性

以及重载的意思。重载, 是指允许存在多个同名函数, 而这些函数的参数表不同(或许参数个数不同, 或许参数类型不同, 或许两者都不同)。

#### 3、http默认端口号

80

#### 4、linux中mysql数据库默认的端口是

3306

#### 5、编程习惯小知识点

C语言编程中, 单片机平台, 一般有.c和.h文件, 如果一个人在.h文件中定义了一个变量, 会有什么后果。(讨论编程习惯的问题, 我一般是只在.h文件中声明函数, 不会做变量定义; 另外, 编程中每一个模块都会有对应的.c和.h文件, 最终的总程序自己定义一个comm.c和comm.h去调用各个模块, 这样的习惯我觉得还行)

if语句中如果是或运算( $|$ ), 第一个条件满足时, 第二个条件还会判断吗。或运算的话, 当然不会, 因为 $0|1=1$ , 中断了。

## 3.2 项目准备与提升

### 3.2.1 有实习经历

如果有机会去实习的话, 一定要去实习, 而且要尽可能的去一些大厂实习。嵌入式岗位, 比如华为、联发科技、小米、京东等这些知名公司去实习。嵌入式校招非常看重实习经历, 因为很多技术内容是你在学校或者实验室项目中接触不到的, 一段好的实习可以为你的秋招增色不少, 实习项目也会成为面试中的主要了解话题, 让你在面试过程中也跟面试官有的聊, 不至于冷场和尴尬, 并且大厂也更愿意招收一些有过中大厂实习经历的人。

在自己有实习的情况下，是完全可以将自己的两段或者三段实习经历写在简历上，进而跟面试官聊实习经历的。大多数情况下，有实习经历是肯定加分的，能够去实习还是很推荐大家去学习的。但是需要注意的是实习经历也不要太多，一般最多三段实习经历已经很多了，大多数情况下写一到二段实习经历就足够了，记得要写上自己最拿手的、最有成就的实习经历写上去。

### 3.2.2 没有实习经历

如果你没有实习，那需要做的就是好好打磨自己的项目了，因为一般面试是至少半小时以上的，除了一些基本知识的考察外，能聊的就是项目了，聊项目实现的技术、项目的代码框架、项目的业务、你关于这个项目的收获等。

校招面试更注重的是项目的深度而不是广度，所以至少得准备一个比较精、比较有难度的项目，而且要带入自己的思考，面试官希望看到的是你在项目中的个人思考，而不是死板的去copy一个项目，如果你能够提出自己的见解和思考，那么毫无疑问是非常加分的。

**项目准备：**很多人苦于没有项目，不知道该如何去找自己的项目去做。一般来说有以下几个途径：

1. 如果你所在的实验室有自己的项目，那么你完全可以把实验室的项目拿出来作为你自己的项目。
2. 网络上的项目。其实网络上有很多好的资源可以去利用，gitHub上的经典项目都可以去学习，将它慢慢消化掉，融入自己的思想，这样就将它们变成了自己的项目了。
3. 某些书本或者网课上的具体项目。大学课本上会有一些小的项目或者会有一些课程设计，也可以写到简历里面。
4. 直接看知名Demo的源码。比如智能小车的源码。

**项目思考：**项目不一定要高大上、功能很齐全，最主要的是，你要在这个项目中收获到了哪些？这个项目你学到了哪些？你有没有自己关于这个项目的思考？如果项目是别人的，你可以在面试的时候跟面试官明确说明，自己的项目是在某个项目基础上加以改进形成的，进而你可以谈一谈自己为什么要提出这个改进？改进效果怎么样？在这个过程中你遇到了哪些困难？自己又是如何解决的。

**亲身经历：**笔者是嵌入式技术栈，做了一个Linux平台的运动相机项目、一个单片机项目、一个硬件项目。因为嵌入式很多时候是和硬件打交道，所以有过硬件经历对面试是加分的。

## 3.3 如何与面试官聊项目

### 3.3.1 有实习经历

如果你有实习的话，那么恭喜你，你在面试过程中可以跟面试官聊的东西就比那些没有实习的要多一些了。如果你在简历中明确写到自己有过一段或者若干段的实习的话，面试官一般都会主动问到你的实习主要是做了什么的，这个时候记得在不违背原则的情况下，对自己的实习经历进行一定的美化，毕竟对于大多数的人来说都是“面试造火箭，进去拧螺丝”。

### 3.3.2 没有实习经历

如果你的简历中有实习经历的话，面试官都会优先询问实习经历，如果没有那面试官在问完一些基本知识比如操作系统、计算机网络之后就会开始直接问你的项目。

一般的流程都是面试官会让你先主动介绍一下你的项目，然后从你的介绍中找出一个他所擅长或者感兴趣的点来问你，所以在叙述自己简历上的项目时，千万不要有结巴或者磕磕绊绊，要不卑不亢的把你的项目完整的叙述出来。

一定要做到对简历上的项目描述中的每一句话负责，因为你所写在项目描述上的每一句话都有可能成为面试官询问你的点。

如果可以，你可以引导面试官往你擅长的方向走。



# 第四章 软实力

## 4.1 打造属于自己的简历

### 4.1.1 简历内容

一般来说公司内部都是有简历筛选系统的，你的简历会经过内部系统筛选一遍，然后是HR人为筛选一遍，HR阅读一份简历的时间只有短短的十几秒甚至几秒，那么这么短的时间，HR在看什么呢？主要针对简历中的信息作客观评估，主要包括个人信息、教育经历、实习/项目经历，这是最关键的几个点。很多企业的HR并不是技术出生或者并不是计算机相关专业毕业的，所以HR涉及技术方面的评估会很少，他们可能会依靠你简历中出现的关键词眼来进行评估。

一份成功的简历往往具备以下几个特征：简历内容和岗位匹配、简介明了、言简意赅、重点突出。下面根据简历的通用排版，给出笔者的一些小建议：

#### 1、个人信息

首先个人信息要全，一般这些要放在最前面的，该有的要有，不该有的就算了，别往上面写。必写的有姓名、联系方式、邮箱、应聘岗位，其次可以选写的：年龄（出生年月）、性别，最后放上自己的照片。

#### 2、专业技能

这一栏是比较重要的内容了，希望同学们好好重视这一栏，将自己所掌握的专业技能说清楚。对于描述性词语的运用要把握好，常见的有：了解、熟悉、掌握、精通，不是万分确定还是不要写精通了，要不然可能会给自己挖坑。

还有一点就是注意技能叙述的完善性。比如有一句话大家可能都会写：“熟悉常见数据结构与算法”，这一句话相信在很多技术岗的同学们简历上都有，我的建议是可以再清楚一些，比如“熟悉常见数据结构如栈、队列、链表、二叉树等，了解常见算法如十大排序、二分查找、双指针、单调栈等”，这样是不是会好一点呢。

注意要对自己在简历上写的东西负责，不要把自己不明白不了解的东西写上去，如果被问到，结果你不会或者答错了，是很扣分的。

#### 3、教育经历

首先，如果个人信息和教育经历不能符合要求，就通常不会再浏览其他内容，先排斥在外，一般大厂校招最低学历要求本科，如果是社招走内推通道可能会放宽到大专。

还有就是同学们注意将自己的受教育经历按照受教育程度从高到低叙述，如果你是研究生，那就先写研究生学校，再写本科学校。如果你是专升本，那就先写本科学校再写专科学校，其次要备注好大学教育的起始时间和类别，比如全日制本科，如果你不写清楚会让筛选你简历的人认为你在混淆统分、成人教育或者专科本科的区别，企图蒙混过关，掩盖自己过往的教育经历。所以最好诚实得将自己的教育经历写出来，写清楚。

同学们也要记得写上自己的相应专业，比如计算机专业/通信专业等。如果某些同学知道自己在学校/班级的排名，可以一并写上。比如 4/34，代表班级34人，自己排名第四名，或者Top 10%这样的院系综合排名。

#### 4、实习/项目经历

如果有实习经历要记得把实习经历写上去，包括实习公司、担任的职位、起始时间，一般都会在实习经历后写上自己在实习期间的主要工作，要写清楚自己在实习期间干了什么事，采用哪些方法取得了什么样的成果。最好是书面用户 + 数据支撑，这样给人的信服度也更高一些。

对于项目经历也是的，项目描述要清晰列出在项目中使用的技术点。还有就是如果是个人的项目，可以选择性的把相关github链接或者博客贴上去。项目要按照时间前后的顺序写，一般会把自己最拿手的放在项目经历的第一个。

## 5、校园经历

校园经历如果拿得出手，比如参加XXX社团，参加某某比赛之类的，也建议写上去。

如果比赛或者社团比较有意义可以适当展开，比如获得国家级比赛/省级比赛，也可以考虑把比赛的过程叙述一下，毕竟国奖还是挺有含金量的。

## 6、其余

在该模块中可以说一下自己的英语水平或者对自身的评价，记得要简要概括，评价需要真正思考一下，不要写那种性格开朗、积极学习之类的话，一看就很假。

如果有实力证明那就更加好啦。有写博客的习惯就把博客贴上去，有github就把github贴上去。

这里放一份腾讯官方推荐的简历模板（建议一张纸写完，也就是最多正反两面），投递需要pdf格式，word格式会乱码：

## 姓名 (Name)

手机: 1xx-xxxx-xxx | 邮箱: 1xxxxx@qq.com

求职意向: 产品运营 (根据自身求职意向替换)



## 教育背景

|      |         |    |                   |
|------|---------|----|-------------------|
| 腾讯大学 | 院系 (专业) | 硕士 | 2022.09 - 2024.07 |
| 腾讯大学 | 院系 (专业) | 本科 | 2018.09 - 2022.07 |

## 实习经历 (\*提示: 此部分建议清晰罗列自己的实习经历, 并尽量用数据展现实习时的成果)

|          |         |                   |
|----------|---------|-------------------|
| XX科技有限公司 | XX岗位实习生 | 2020.09 - 2020.12 |
|----------|---------|-------------------|

- 协助产品经理负责前期产品的规划, 流程和原型设计并进行相关调研, 并对需求进行整理和分析, 撰写需求文档;
- 对接运营, 技术, 设计各方保证产品按期上线, 上线后一个月获得2万新用户, 次月留存率30%;
- 上线后监测用户拉新, 留存等数据并汇报, 定期根据数据进行复盘并提出优化意见, 首次产品优化后初次登陆用户留存率提升4.6%。

|          |         |                   |
|----------|---------|-------------------|
| XX科技有限公司 | XX岗位实习生 | 2020.03 - 2020.08 |
|----------|---------|-------------------|

- 独立负责产品市场推广计划, 主要分为线上推广, 对外合作, 微信公众号及微博运营等渠道, 分别管理线上线下媒体渠道的拓展和投放;
- 分析本产品市场自媒体情况, 触达优质自媒体资源, 与平台各大V合作进行推广, 大幅提高产品销售额, 单日销售额最多上涨300%;
- 多次策划节日电商活动, 包括撰写策划案, 产品文案及活动盈利分析, 成功提高月销售额20%。

## 项目经历 (\*提示: 此部分可根据个人情况填写实习以外经历, 如校园经历、竞赛经历等)

|         |    |                   |
|---------|----|-------------------|
| 校园自媒体团队 | 组长 | 2018.09 - 2020.07 |
|---------|----|-------------------|

- 负责团队内容运营工作, 包括参与团队选题策划, 文案撰写等, 每周平均贡献2篇图文, 平均阅读量2000+, “在看”率突破20%;
- 负责日常公众号运营规划, 收集, 分析和整理用户反馈及数据, 提高用户粘度和用户活性, 协同团队工作期间, 粉丝数目突破2w+, 增长超过50%。

|          |      |                   |
|----------|------|-------------------|
| XXXX品牌商赛 | 团队成员 | 2020.07 - 2020.09 |
|----------|------|-------------------|

- 作为主要负责人, 组织实地店访和访谈调研工作, 高效收集3个城市的消费者数据, 共计访谈品牌及市场专业人士15人, 实地走访线下门店超过50家, 并回收消费者有效问卷1000+份;
- 结合收集的一手数据和10+份行业研报, 通过定性 (品牌调性) 和定量 (消费水平和渠道数据) 分析, 对消费者人群进行细分, 提炼市场洞察, 完成共计15000字的项目报告。

## 其他 (\*提示: 此部分可根据个人情况填写补充信息, 如获奖情况、技能等)

- 语言: 英语 (CET-6) ;日语 (N1)
- 技能: Excel (熟悉使用数据透视表, VBA) ; Photoshop (基本图片处理)

## 4.1.2 简历

大家可以用一些简历模板，在此基础上修改，或者用一些简历网站，牛客网是有在线简历功能的。

## 4.1.3 投递简历

对于应届生来说，投递简历的途径有很多。主要有三种，1、校园宣讲会 2、网申 3、内推。

### 1、 校园宣讲会

校招宣讲会一般都会有一些笔试，笔试过了即可参加正式的面试。

### 2、网申

网申基本算是投递人数最多的一种方式，通过招聘网站或者官网进行信息的填写，直接投递即可。需要注意的是，不同公司用的简历系统都是独立的，所以很有可能你每投递一家公司就要填写一次，真的真的很累人。如果浏览器有插件会快一些。

### 3、内推

这也是近几年越来越流行的一种方式了，直接找意向公司内部员工帮忙推荐即可，比如已经入职成功的学长学姐，建议能找内推一定要找内推，否则简历很容易石沉大海，连求职进度都不知道。牛客网上也经常有一些人发布招聘信息，基本都会附赠内推码或者内推邮箱或者搜寻自己想要投递的岗位获取内推码，然后发邮件过去就好。还有部分公众号也是可以找到内推信息的，关注即可。内推对于内推人来说也是有好处的，基本上每个大厂内推成功都是有奖金的，他内推你，如果你能够顺利入职的话，内推员也是会拿到奖金的，内推基本是双赢的。

这里有两个需要注意的点：

1、注意添加内推人的联系方式比如微信QQ等，及时了解内推信息，跟进投递进度。

2、注意发送简历的格式，一定是pdf格式的，这样简历格式才不会改变，word在不同电脑上格式可能会有所变化。发送给招聘邮箱的简历也要注意命名和格式：姓名+职位，邮件标题也命名为：姓名+意向工作地+岗位。

## 4.2 如何通过笔试

笔试主要是为了初步过滤面试者，减少人工面试的工作量。大厂笔试淘汰率基本都是超过50%以上。

一般来说，笔试共分为三种：性格测试、行测、专业笔试。

### 1、 性格测试

性格测试主要测试你的性格是否正常，是否具有极端情绪，测试你与本公司的文化是否匹配。

建议诚实作答，因为有的企业会放置重复的题，如果你前后选择不一样会被认为诚信有问题而导致分数偏低，真实做自己就好。

### 2、 行测

行测主要测试求职者的逻辑思维能力和反应能力，常见题型包括资料分析题、图形题、数量关系题等。这种行测题都是有时间限制的，通常为40秒/50秒/60秒，要注意把握时间，时间到了没有选中选项的话，会自动跳到下一题的。

行测的常见题型并不多，甚至可能出现一道题在多家行测中重复遇到多次的情况出现，同学们是通过短期训练来进行弥补的。

### 3、专业笔试

专业笔试是笔试中最难的一关了，主要考察求职者的专业技术能力，考察方向就是你所申请岗位的方向了，一般考察的就是计算机基础、数据结构与算法、操作系统、计算机网络、Linux、数据库了。

这部分我写在了另外一篇pdf文档中，大家多刷题是没问题的。我总结了122页的嵌入式软件笔试面试题目，大家在公众号后台加博主获取。

## 4.3 如何通过面试

面试环节是求职应聘中最重要的环节，因为是面试官直接与求职者面对面的交流，如果是中小型企业，面试两次基本就可以了；如果是大公司，一般至少需要面试三到四次甚至五到六次才能确定是否录用你。

面试组成基本上是10%手写代码+20%基础问题+40%深挖项目+20%开放问题+10%聊人生。

其中手写代码是必要的，一般会共享屏幕或者在指定oj上手写代码，同学们可以不用担心，面试过程中的代码题比笔试过程中的代码题要简单多了，难度基本都是easy或者medium的，hard的很少。除此之外面试中的手写代码还有一个目的就是看你的代码风格和debug能力，毕竟代码风格不是一朝一夕能养成的，面试官看你的代码风格也是能够看出来你是不是经常写代码，看你的变量命名是否合理等。经常写代码和不经常写代码的人代码风格完全不一样的。Debug能力更不用说，考察的就是你能否快速定位到bug，进而解决它。

面试过程中有一些需要注意的地方：

- 1、尽量展示自己的长处，一般面试官也不会太为难你，为难你没有用处，面试关键是在最短的时间里确定你的水平，所以一般都会循序渐进，看看你到底什么水平。
- 2、一定要诚实，会就是会，不会就是不会，不要浪费面试官时间。说了谎话被发现，绝对gg，毫无疑问。
- 3、对于编程题，就算不会，也要说思路，实际工作中，只要有思路，都可以解决。
- 4、自己要多总结，多回顾，博主也是总结的比较多，才会有这一篇文章出来。

### 4.3.1 一面

一般来说，第一面都是基础技术面，就是考察面试者的计算机基础，也就是操作系统、计算机网络、数据库、数据结构与算法、编程语言等，有时候也会问一下你的项目，不过一面深挖项目的不多，主要是考察基础，要求面试者具备扎实且广泛的计算机基本知识。

可以说一面是考察范围最广的一轮面试了，面试时间也比较长，互联网大厂一面基本都在30-60分钟之间，如果你的面试时间小于30min，很有可能凉凉。

面试开始的时候都会让你简单介绍一下自己，为什么明明简历上都写了自己的信息，还需要自我介绍呢，主要有以下2个原因：

面试官很忙，没看你的简历。很多面试官本身就是公司的一些部门主管或者技术leader，他们本来就很忙，每天要处理很多的事情，可能他刚拿到你的简历没几分钟，HR就安排了这次面试。在你进行自我介绍的时候，他也可以看看你的简历，熟悉一下你的技术栈和项目。

了解面试者的沟通能力，语言表达能力。面试官通过听你的自我介绍也能看到的你总结概括能力、逻辑思维能力等。在职场中，除了基本的技能外就是跟同事合作，一起去完成某项任务。如果你在介绍自己的时候都介绍的一塌糊涂，以后能指望你跟身边的同事沟通效率高吗？

一面最后的时候，面试官一般会问：你有什么要问我的吗？这个时候不要乱问，你可以问以下几个问题：

- 1、你们部门在做些什么？主要业务是什么？如果自己很荣幸的能够进入贵部门会负责些什么？因为面试者就是这个部门的，通过他的回答，你也能够了解到这个部门正在做的产品和使用的技术。



2、您认为我在哪些方面还存在着不足？这是一个很巧妙的问题，因为它可以从侧面反映出你这次面试的结果。如果面试官带有指导性的回答出了你的不足，你需要补充的知识点，这样就代表你这次面试差不多了，应该是能好好准备二面了；如果面试官直言不讳的说你很差或者基础太弱这样的话，你也知道凉凉了。

3、请问面试官对自己职业规划的建议？面试官大概率是技术大佬或者工作过几年的前辈了，在社会上摸爬滚打了几年，知道的肯定比在校生多。这个问题既表达了对面试官身份的认可，也表现出求职者对当前这份工作得在意程度，并且还能得到技术大佬的分享，怎么看都不是亏本的买卖。

## 4.3.2 二面

互联网一般二面面试官都是技术leader级别的了。二面就开始考察你的实习/项目了，而一般中小厂可能将二面和HR面放在一起了。

二面没有一面那么注重基础，会开始问你一些这个项目的细节部分。这个时候你就要跟面试官讲你精心准备的实习或者项目，一般都会是让你说一下你这个项目是用来做什么的？为什么会有这个项目？如何实现某某细节的，用的是什么技术和框架？一般面试官问你问题的都是他们擅长或者喜欢的技术点，所以你如果仔细讲述清楚并且能加入一些自己的思考会加很多面试分，比如当前这个项目还存在着那些不足，可以用什么样的技术去改进它之类的。

## 4.3.3 三面

三面一般都是综合面考察，并不是很在乎你的基础了，而是会考察你这个人的思维能力、分析能力等，将事务看清楚、看明白，提炼总结的能力，换句话说就是看你这个人是不是脑子够灵活，是不是够聪明。

在三面过程中，还有一些问题是看你的抗压能力以及处理意外情况的能力，比如：

- 1、分享一件你觉得压力比较大的事？你的压力从何来？你是如何克服他的？
- 2、你长这么大以来遭受过的最大挫折是什么？你是如何克服它的？
- 3、二十多年来，你取得的最大成就是什么？
- 4、你通过多年努力获得的一项技能是什么？你是如何学习从而获得这项技能的，做了哪些工作去改善、精进这项技能？

从面试官的角度来看，他问你经历过的最大困难是什么是真的对你所经历的困难感兴趣吗？不是的，这个问题的重点是在考察你面对困难时所做的思考和应对，是想看到你的努力以及解决问题的能力。困难人人都会遇到，克服困难固然值得鼓励，可更重要的是从这个困难中学到了什么，即使没能够克服困难也不意味着一无所获，面试官希望看到的是你如何从过往的苦难和失败中总结出经验，并在以后的工作中能够用上这些经验，更好的指导日后的工作。

## 4.3.4 交叉面

如果求职者被HR告知要进行一轮交叉面或者加面一轮，基本是出自以下两个原因：

- 1、前面三轮还不足以确定你的程度，属于那种对你基本满意但是还差点意思，需要加面一轮才能确定你的评级，才能最终给你定薪资。这种情况就属于比较危险的，如果交叉面没答好，很有可能前功尽弃。
- 2、第二种情况就是求职者过于优秀，惊动高层的那种优秀，哈哈。加面一场，如果你答得不错的话，给你更高的面试评级，这也意味着SP、甚至是SSP。答得不好也不会取消offer和降低原有的评级，这一点不需要担心。

### 4.3.5 HR面

很多人觉得前面几轮的技术面过了就基本稳了，其实HR面也很重要，很多公司的HR权力是很大的，拥有绝对的一票否决权，即使部门主管想要你，HR不同意那也没有办法。

HR面主要是看你对公司文化的理解和价值观的认同，笔者建议在HR面前，先去了解一下公司的文化和公司的优势之处，这样在被问到为什么选择本公司的时候能够把自己对公司的了解和优势说出来，体现自己的诚意。

HR面的时候也会问一些其余的问题，比如你的最大优点和缺点，这也是HR面试高频问题，大家最好提前准备好这个问题的答案，真的很高频。

另外HR面会问你家庭情况，男女朋友情况，主要是确定你的稳定性，你到底会不会来。这里你就看情况回答啦。

## 第五章 Offer选择

---

影响个人offer的一些因素主要有：学校、学历、面试评价，其中第三个面试评价是占比较大的比重的。

这里简单为同学们科普一下校招offer中的一些名词：意向书与烂白菜、白菜、大白菜、SP(Special Offer)、SSP(Special Special Offer)、SSP+(Special Special Offer +)

**意向书**：意向书的意思是指，在秋招中顺利通过面试获得口头offer的一种录用意向，没有具体的薪资待遇说明，换句话说就是：你很不错，我们打算要你了。

**烂白菜**：offer的最低一等，也是最近两年新出来的一个词，以前是只有白菜价这一个说法的。

**白菜**：指校招生中占据60%-70%的人拿到的offer薪资待遇，毕竟大佬还是比较少的，大部分人的offer都是白菜价格。

**大白菜**：也是最近两年新兴的一个词语，意思就是比白菜价稍高一档的薪资待遇。

**SP**：大佬拿到的offer偏低的一档，但也比大白菜要高一点。

**SSP**：这是真大佬才能拿到的，能够拿到SSP的同学基本都是校招生中的人中龙凤，极其稀少。比如快手的快star、中兴的蓝剑计划、阿里的阿里星、小米的未来星、京东的猎聘计划等，能够通过上述考核人拿到的基本都是SSP。

**SSP+**：一般很少出现SSP+，这种offer一般只发给那些神仙选手，就是很厉害很厉害很厉害的选手。

### 5.1 Offer选择与比较

---

大家在选择offer的时候一般会在意的关键因素，主要有：薪资待遇、地域、岗位前景。

**薪资待遇**：大部分人都是喜欢待遇高一点的offer。

**地域**：主要是个人意向工作城市，因为会关系到很多人的日后规划，涉及到买房定居之类的。比如这个offer的base是在北京，但是我就想去广州或者上海，不能接受北京。

**岗位前景**：对于一些想要一直从事于IT行业的人来说，可能会认真考虑当前岗位的前景问题。自己做的技术栈发展前景到底如何，尽量多找前辈问问。

### 5.2 如何要更多的钱

---

大部分的Offer在正式发放前都会有电话沟通，主要是了解你手上是否具有其他的offer，然后给你介绍公司的各种情况和福利待遇等，最终你给予口头答应后才会正式发放offer，然后给你几天的时间考虑是否接受。

Offer中的薪资待遇不是定死的，如果你觉得当前公司给的薪资待遇较差，但是又不想放弃手里的Offer，认为自己可以拿的更多。这个时候是可以跟HR提出申请，要求增加薪资的。而这个时候你去要求增加薪资待遇或者签字费的底气就是你手上的其他家的offer。

## 结束语

秋招 = 实力 + 面试技巧 + 运气 + 心态，每一个环节都是充满了不确定性，随时都有可能因为不确定的因素而挂掉，作为求职者的我们无法预知某个公司某个岗位的难度，所以也不建议在一棵树上吊死。

最后祝愿同学们都能成功上岸，拿到自己满意的offer！！

原创不易，如果对你有帮助，谢谢支持。

推荐使用微信支付



Jason(\*杰)



微信支付

支付宝

推荐使用支付宝



小杰(\*杰)

打开支付宝[扫一扫]

申请官方收钱码：拨打95188-6