开源入门课

庄表伟

2020-08-15

大纲

- 开源是什么？
- 为何要学习开源？
- 如何学习开源？
- 如何成为开源社区的一份子？

开源是什么？

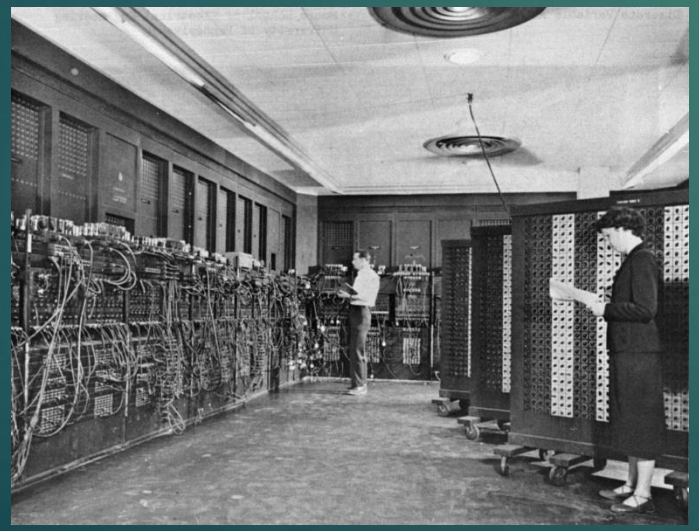
- Open Source，是一个专有名词，由开源促进会（**O**pen **S**ource **I**nitiative）在1998年定义
- OSD（Open Source Definition），是一份有OSI发布的文档，经由Debian的创始人Bruce Perens撰写的《Debian自由软件指导方针》修改而来
- OSD一共10条，规定了符合开源的10个特征，如果一个项目仅仅是把自己的源代码公开出来，就宣称自己是开源软件，那是不成立的
  - 这10条，希望同学们能够自行查阅维基百科，包括中英文版
- 但是，仅仅提到开源的定义，对于理解开源是什么，其实是不够的

开源是什么？

- 开源软件（开源硬件）容易定义，但是开源本身，却很难定义
- 开源是一种开放协作的模式
- 开源是一种开放社区的行为模式
- 开源的背后，是一种心理学现象
- 开源的成功，依赖于法律条款的保障
- 开源的崛起，在于商业企业终于意识到开源的威力
- 我们可以先从开源的历史讲起……

开源出现之前

- 出于教学与学术目的，交流各种源代码
  - 大型机、中型机、小型机的时代，大多数个人都买不起计算机
  - 有能力购买计算机的机构、大学，都聚集了大量的科研人员
  - 代码的交流，就像学术交流一样，毫不考虑挣钱的事情

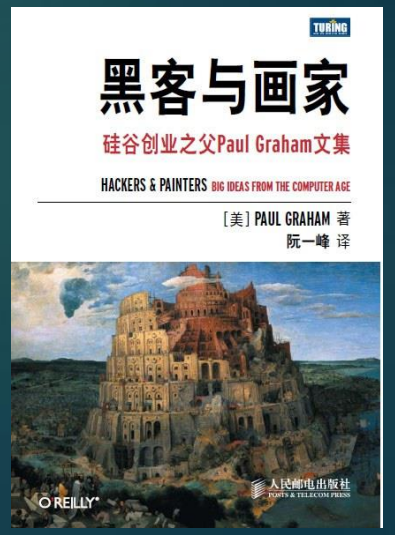


ENIAC 1946

DEC PDP-7 1965

黑客伦理

- 对计算机的访问（以及任何可能帮助你认识我们这个世界的事物）应该是不受限制的、完全的。任何人都有动手尝试的权利！
- 所有的信息都应该可以自由获取。
- 不迷信权威——促进分权。
- 评判黑客的标准应该是他们的技术，而不是那些没有实际用途的指标，比如学位、年龄、种族或职位。
- 你可以在计算机上创造艺术与美。
- 计算机可以让你的生活更美好。
- 就像阿拉丁神灯，你可以让它听从你的召唤。

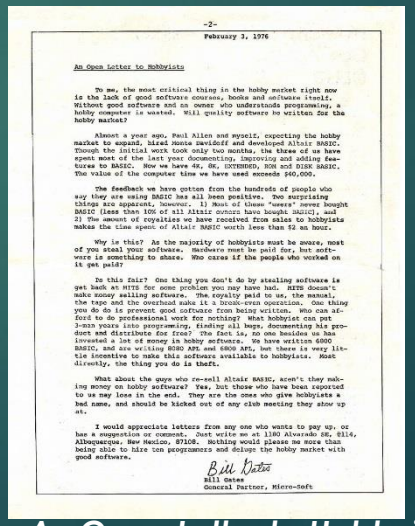


商业“魔头”现身

- 出于商业目的，封闭代码，出售拷贝
  - 微型计算机、个人电脑开始兴起，普通玩家之间，也会无偿的交换盗版拷贝
  - 1976年，比尔盖茨发表著名的《写给电脑爱好者的公开信》，倡导版权与利益。而且愤怒的将那些免费复制软件的家伙，称之为：窃贼！
    - 有谁会在没有任何报酬的情况下来做这些专业的工作？什么样的爱好者可以为他的产品投入三个人年的开发时间，并且发现所有的错误、编写文档以及免费发布这个产品？
  - 软件行业兴起！

IBM 5100 Portable Computer 1975

Apple I 1976



-2-
February 3, 1976

An Open Letter to Hobbyists

To me, the most critical thing in the hobby market right now is the lack of good software courses, books and software itself. Without good software and an owner who understands programming, a hobby computer is wasted. Will quality software be written for the hobby market?

Almost a year ago, Paul Allen and myself, expecting the hobby market to expand, hired Monte Davidoff and developed Altair BASIC. Though the initial work took only two months, the three of us have spent most of the last year documenting, improving and adding features to BASIC. Now we have 4K, 8K, EXTENDED, ROM and DISK BASIC. The value of the computer time we have used exceeds $40,000.

The feedback we have gotten from the hundreds of people who say they are using BASIC has all been positive. Two surprising things are apparent, however. 1) Most of these "users" never bought BASIC (less than 10% of all Altair owners have bought BASIC), and 2) The amount of royalties we have received from sales to hobbyists makes the time spent of Altair BASIC worth less than $2 an hour.

Why is this? As the majority of hobbyists must be aware, most of you steal your software. Hardware must be paid for, but software is something to share. Who cares if the people who worked on it get paid?

Is this fair? One thing you don't do by stealing software is get back at MITS for some problem you may have had. MITS doesn't make money selling software. The royalty paid to us, the manual, the tape and the overhead make it a break-even operation. One thing you do do is prevent good software from being written. Who can afford to do professional work for nothing? What hobbyist can put 3-man years into programming, finding all bugs, documenting his product and distribute for free? The fact is, no one besides us has invested a lot of money in hobby software. We have written 6000 BASIC, and are writing 8080 APL and 6800 APL, but there is very little incentive to make this software available to hobbyists. Most directly, the thing you do is theft.

What about the guys who re-sell Altair BASIC, aren't they making money on hobby software? Yes, but those who have been reported to us may lose in the end. They are the ones who give hobbyists a bad name, and should be kicked out of any club meeting they show up at.

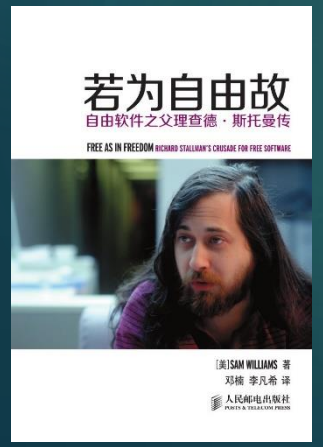
I would appreciate letters from any one who wants to pay up, or has a suggestion or comment. Just write me at 1180 Alvarado SE, #114, Albuquerque, New Mexico, 87108. Nothing would please me more than being able to hire ten programmers and deluge the hobby market with good software.

Bill Gates
General Partner, Micro-Soft

An Open Letter to Hobbyists

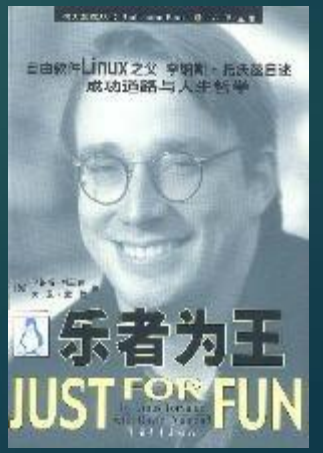
黑客的愤怒 / 只是为了好玩

- UNIX纷纷闭源，商业公司纷纷成立
  - Novell owns AT&T's Unix：IBM-AIX、SGI-IRIX、SCO UNIX、HP-UX、SUN-Solaris
  - Berkley（BSD）：SunOS、Ultrix、NetBSD、DEC-OSF、NeXTSTEP、Mac OS X
  - 这些UNIX，收费昂贵、互不兼容、而且闭源，令人抓狂，令黑客们愤怒！
- GNU/Linux的崛起
  - Richard Stallman在1985年发表了GNU宣言，并于1989年起草了GPL，提出了Copyleft的概念。Emacs、GCC等工具纷纷问世
    - GNU： GNU's Not Unix
  - 1990年， Linus Torvalds在芬兰赫尔辛基大学读书期间，开始开发Linux，1991年发布最初版本，并飞速发展至今
  - BSD家族由于受到AT&T以及后来Novell的诉讼，发展缓慢。FreeBSD始于1993年



Richard Stallman

Linus Torvalds



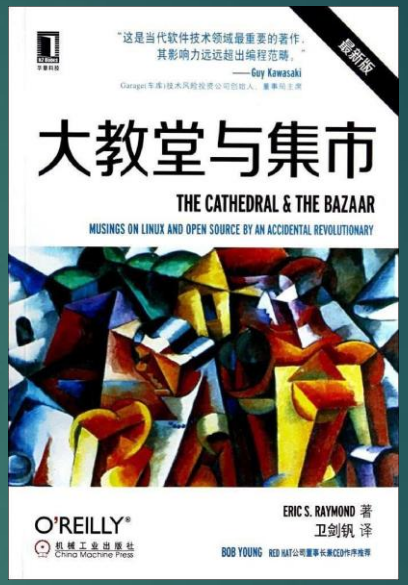
一些不可思议的事情

- Linux 1991年发布第一个开源版本
  - 通过互联网聚集了大量的志愿者，没有严格的质量标准，没有强有力的机构协调管理
  - 最简策略：**每周发布，然后接受反馈**
  - 到1993年底，Linux在稳定性与可靠性上，已经与很多商业UNIX不相上下，并能支持比商业UNIX多得多的软件
  - 大多数小型UNIX供应商倒闭

	Linux	Windows
桌面与笔记本	1.74%	>88%
服务器	38.6%	32.6%
超级计算机	96.4%	<0.4%

初步的观察、思考与总结

- Eric Steven Raymond ：《大教堂与集市》
  - 1997年5月27日发表首次公开发表
  - 1999年由O'Reilly出版
  - 2014年，中文版被翻译引进中国
  - 在开源社区，被誉为“圣经”级的文献



《大教堂与集市》是开源运动的 《圣经》，颠覆了传统的软件开发思路，影响了整个软件开发领域。作者Eric S. Raymond是开源运动的旗手、黑客文化第一理论家，他讲述了开源运动中惊心动魄的故事，提出了大量充满智慧的观念和经过检验的知识，给所有软件开发人 员带来启迪。本书囊括了作者最著名的“五部曲”，并经过作者的全面更新，增加了大量注释，提高了可读性，是经典收藏。

开源运动对软件业和互联网带来了巨大影响，本书作为开源运动的独立宣言，其影响力远远超出编程领域，如果想在互联网时代做生意，这本书是必读经典。

————豆瓣简介

企业的一系列反应

- 不屑→震惊→担忧→分析→诋毁→反思→引进→利用→拥抱
- 1998年10月底开始，微软的一系列内部文件被泄露到Raymond的手上，他在做了大量的辛辣评注后，公开发布。
  - 史称《万圣节文档》，事后微软也被迫承认，这批文件的真实性。
  - 在文件中，微软一方面表明Linux不值得担忧（因为它缺乏管理与方向），另一方面却又在谋划各种策略，试图“干掉”Linux
- 另一个极端案例，也出现在1998年1月，网景公司还是当时的世界500强，但是却在微软的打压之下，举步维艰。
  - 在读到Raymond的《大教堂与集市》之后，决心将网景浏览器开源，后更名为Mozilla Firefox

企业与开源关系的诸多转变

- 开源这种开发模式，为什么能够成功？
  - 一个企业，是否可以借鉴这种开发模式？
    - 省钱、省力、免费的测试与开发…
  - 想要借鉴这种开发模式，是否也必须开源自己的代码？
    - Open Source vs. Inner Source
- 开源领域的创新层出不穷，企业能做些什么？
  - 在质量得到认可之后，企业开始大量选用开源产品、框架、组件、类库（当然还是要做法律方面的审查）
  - 是否有必要参与到某个开源项目中，一方面能够搭便车，一方面也能够加深了解，做一些符合自己利益的贡献
  - 开源基金会的出现，更好的规范了企业与开源之间的关系
- 互联网模式的出现，使得大企业愿意全力支持开源
  - 平台开源，建立开源生态，夺取市场主导地位（Android）
    - 转移支付

开源社区的演进

- 黑客圈，也是有代沟的
  - 每一个早期平台上的黑客群体，会普遍鄙视新兴平台的“毛头小伙子”
  - 大型机程序员，瞧不起小型机程序员；小型机程序员，瞧不起UNIX；UNIX程序员，认为Linux只是个玩具；内核开发者，瞧不起后来的各种应用开发者
- 每一代开源社区，也有不同的玩法
  - 在老一代玩法下成长起来的黑客，往往会“受不了”那种时髦的玩法
  - 邮件列表→集成管理平台→社交化编程(Github)
  - 2012年， Linus Torvalds公开抨击Github上提交的补丁，质量太低。在线编辑方面，简直就是垃圾
- 撇开针对某个具体的事例的抱怨，开源的逻辑，其实一直没有改变
  - 去中心化、降低参与者的门槛、以宽松的方式鼓励创新
  - 程序员的数量一直在增长，参与开源的程序员数量也一直在增长
    - 目前在Github上托管的开源项目，已经超过了1亿！会员接近（可能已经超过）4000万！

Github: Social Coding

- 传统的开发行行为，是很难被观察与统计的
  - 基于配置库的提交历史，我们无法判断一个人的贡献大小，也无法判断他的代码质量
  - 一个人提交了两次代码，每次二十行，中间相隔2个小时，但是我们无法推断：这个人在2个小时里，写了二十行代码
  - 一个人提交了100次，我们也无法判断，他这100次提交，有多少是新功能，有多少是修改bug
  - 当然可以在提交的注释里，约定填写更多的内容：这些需求，需要被更好的满足!
- Github从SNS网络中，借鉴了非常多的东西
  - Follow、Watch、Star、Timeline、User Page
  - 也有很多独特的创新：Fork、Pull Request、Gist、Pages、Graphs、Pulse
  - 关键点：行为分化、数据化
- 由于Social Network的激励作用，Github成为了目前最大的开源项目托管平台
- 基于大数据分析，Github也成为了可能是全球最大的开发人员人才库

开源：已经到来的大繁荣！

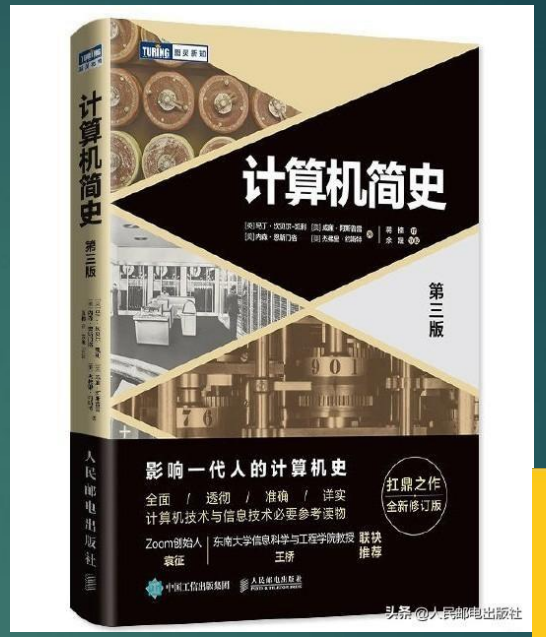

早期文化基础
成功案例 Linux/Apache
理论总结
商业投入
工具与平台支持
社交网络与新兴开发模式
已经到来的大繁荣！

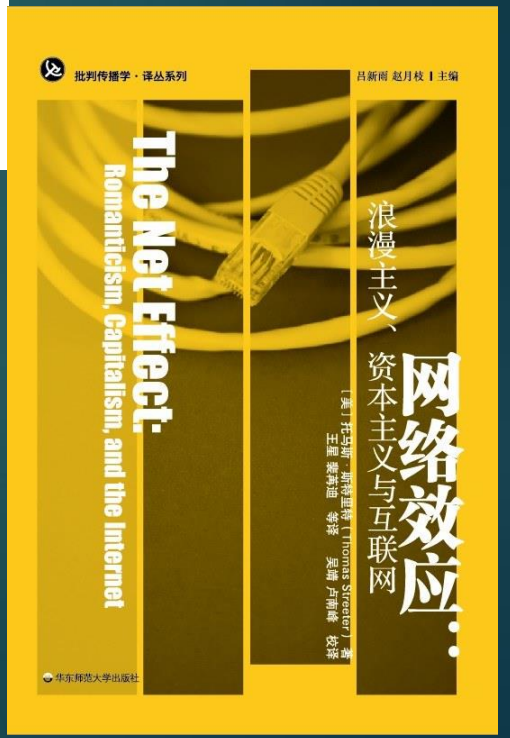
为何要学习开源?

- 职业生涯视角
- 个人成长视角

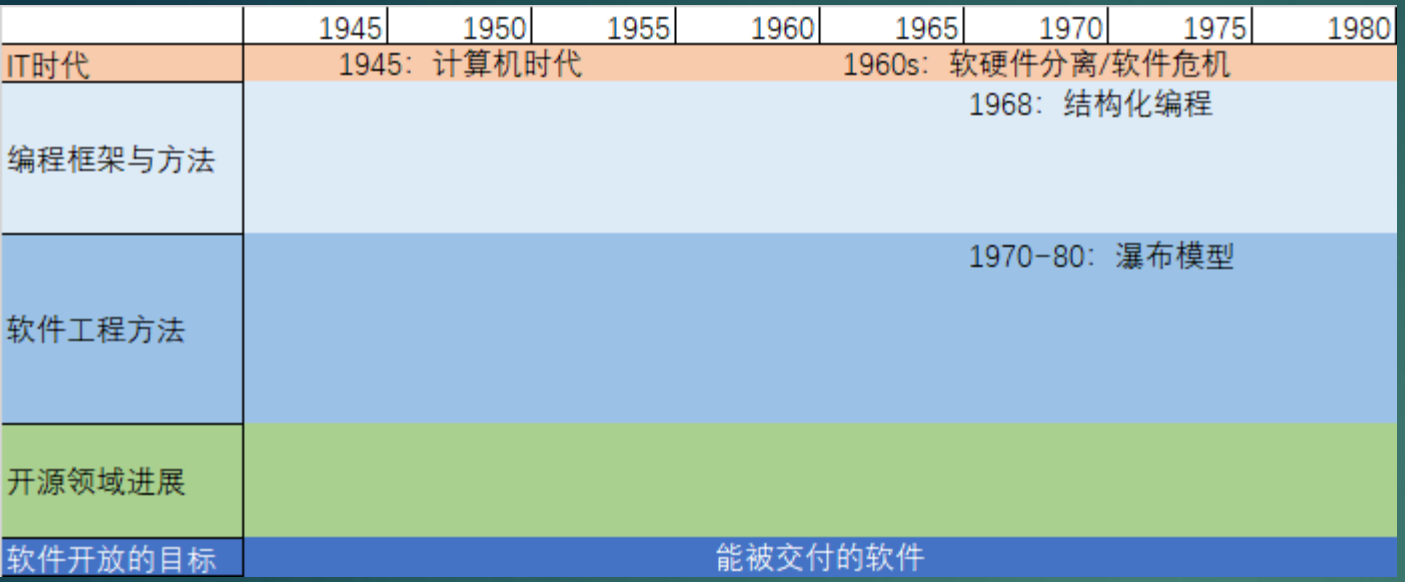
软件工程的三个发展阶段

- 军工驱动的软件工程
  - 半自动地面防空系统（SAGE）
  - 高级研究计划局网络（ARPANET）
- 商业软件驱动的软件工程
  - 微软的诞生
  - Oracle的诞生
- 互联网驱动的软件工程
  - Google的诞生
  - 云计算的诞生
- 最近进展：可信软件工程





时间线



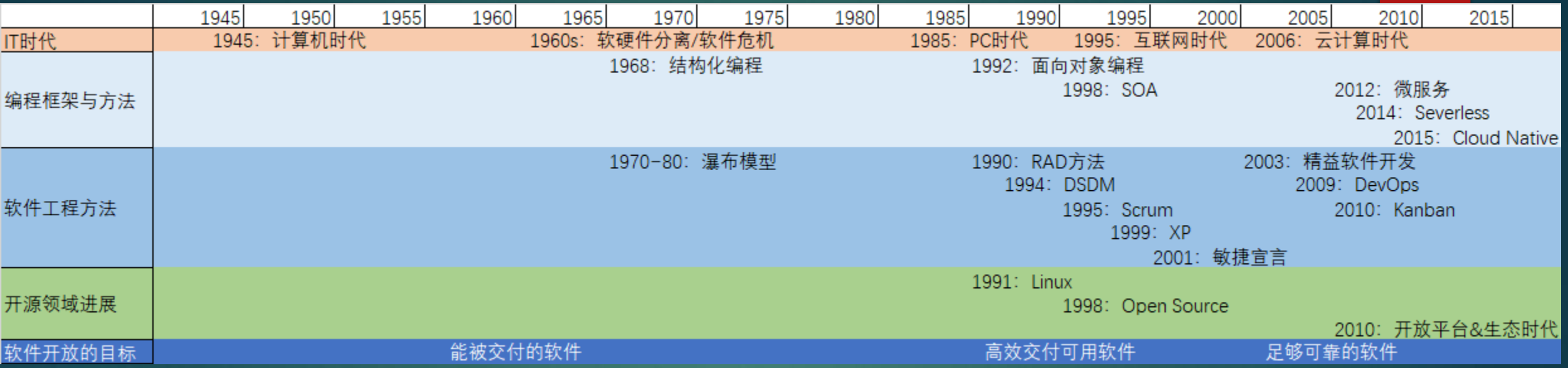
- 从软硬件一体，到软硬件分离，软件被解放出来了
- 不再受制于硬件之后的软件，需要新的开发方法，在没有方法的时代——软件危机诞生
- 从编程思想来看：大家都很朴素，把代码写得有条理些，有结构化一些，也就可以了
- 从软件工程方法来看，因为软件开发是一个全新的行业，只能向传统行业取经，瀑布模型应运而生
- 以上的一切努力，目标只是：争取能把软件做出来
- 开源：还在蒙昧阶段，源代码的所有权意识不强，代码在世界上随意的流动着

时间线

	1945	1950	1955	1960	1965	1970	1975	1980	1985	1990	1995	2000
IT时代	1945：计算机时代				1960s：软硬件分离/软件危机				1985：PC时代		1995：互联网时代	
编程框架与方法						1968：结构化编程				1992：面向对象编程	1998：SOA	
软件工程方法						1970-80：瀑布模型				1990：RAD方法	1994：DSDM	1995：Scrum
											1999：XP	2001：敏捷宣言
开源领域进展										1991：Linux	1998：Open Source	
软件开放的目标	能被交付的软件									高效交付可用软件		

- 进入PC时代，拥有个人电脑的人大量增加，采购PC的企业也大量增加
  - 软件成为一门赚钱的生意
  - 比尔盖茨：《写给电脑爱好者的公开信》
- 围绕软件，商业公司开始出现，商业竞争日趋激烈
  - 仅仅把软件写出来，是不够的——还要足够快！
- 软件越来越复杂，简单的结构化编程已经无法驾驭，必须探索更加高级的架构模式
  - 面向对象和面向服务的架构，成为朴素思考的延伸
- 工程方法与研发工具，都开始涌现
  - 在方法领域，最终汇聚为敏捷宣言
  - 在工具领域，Visual Studio成为王者
- 作为对商业软件的反抗，自由软件/开源软件开始出现
  - 但是还远远不成气候

时间线



- 互联网时代改变了一切，但是在没有云计算之前，没有引发质变
- 架构的变化：
  - 服务化、微服务化、无服务化、云原生架构
- 工程方法的变化：
  - DevOps兴起（研发关注的领域扩展到Ops域）
  - 看板兴起：软件开发找到了自己的节奏感
- 开源软件崛起，引发开发模式的变化
  - 大教堂与集市
  - 基于Packages的代码重用，开发变成一项永远在线的工作
  - 开源生态渗透到企业

时间线

	1945	1950	1955	1960	1965	1970	1975	1980	1985	1990	1995	2000	2005	2010	2015	2020
IT时代	1945：计算机时代				1960s：软硬件分离/软件危机				1985：PC时代		1995：互联网时代		2006：云计算时代		2016：AI时代	
编程框架与方法						1968：结构化编程				1992：面向对象编程	1998：SOA			2012：微服务 2014：Severless 2015：Cloud Native		
软件工程方法						1970-80：瀑布模型				1990：RAD方法 1994：DSDM	1995：Scrum 1999：XP 2001：敏捷宣言		2003：精益软件开发 2009：DevOps	2010：Kanban		
开源领域进展										1991：Linux	1998：Open Source			2010：开放平台&生态时代		
软件开放的目标	能被交付的软件									高效交付可用软件			足够可靠的软件		安全、可信的软件	

最新的挑战：开源、平台、生态、云计算、物联网、AI

- 大规模跨组织协作
- 开源生态无处不在
- 面向云的开发组织与架构
- 围绕API组织开发
- 面向AI的开发与被AI增强的开发

如何开发可信的软件？

推动软件工程不断发展的几股力量

- 对于软件的需求，推动软件的发展，进而推动软件工程的发展
  - 越来越复杂的软件，对于软件工程的发展，提出了更高的要求
  - 软件类型的分化，也使得软件工程的各种方法论，出现分化
- 开源软件的出现，极大的改变了软件开发行业
  - 可复用的开源组件，成为软件开发的基石
  - 开源软件开发模式，对于商业软件开发的各种启发
- 软件工程师，不仅仅是(使用工具) 开发软件的人，也是开发工具的人
  - 层出不穷的开发工具，成为工程师手中的利器
  - 软件行业——为自己开发工具的行业，绝无仅有！

推荐的职业生涯发展策略

理解开源的逻辑

理解开源对于软件开发、软件工程的影响

关注业界最新发展方向

积累相关技能

敢吹牛的人，才能成为技术大牛

- Linus Torvalds在1991年发了一个邮件：我在做一个开源的操作系统...
  - 感兴趣的人们，蜂拥而至
  - Bug报告、需求、补丁也蜂拥而至
  - 然后，Linux就慢慢长大了，Linus也成了大神！

乐趣第一

- 关于生活的意义，Linus有自己的理论，在《Just for fun》中，这个理论甚至被阐述了两遍
  - 首先是生存，人类做事，最基础的动机，就是为了生存
  - 其次是社会关系，有时候也会被称之为“秩序”。无论是为了维系社会关系，还是为了维持社会秩序，这都是重要的动机
  - 最后是娱乐，或者说乐趣，在Linus看来，这才是动机的最高境
- Linux为啥会继续开发下去？
- Linus：“我本可以在1991年底就收手不干了，因为那时我已经完成了大量有意思的工作……幸好接下来发生了两件事，我才有了继续下去的动力：第一，我不小心损坏了Minix系统的分区；第二，人们不断给我发来反馈意见。”
- 乐趣→吹牛 →社区压力→被迫成为大牛

成长的正循环

在获得乐趣的过程中，产生创造的冲动

在创造的过程中，产生真实的练习目标

在练习中，收获越来越多的乐趣

开源社区作为个人成长的催化剂

推荐的个人成长策略

理解开源的逻辑

投身开源社区

收获乐趣、不断成长

如何学习开源?

- 报名参加《开源特训营》
- 找到自己感兴趣的开源项目，投入贡献
- 找到自己感兴趣的开源社区，投入贡献
- 提高搜索技能：Google、GitHub、Stack Overflow ……
- 掌握基础知识：一门编程语言、Git、Linux ……
- 推荐书单：
  - 《大教堂与集市》
  - 《黑客与画家》、《黑客》
  - 《若为自由故》、《乐者为王》
  - 《计算机简史》、《网络效应》

如何成为开源社区的一份子？

- 社区是一群人构成的，成为社区的一份子，就是和这一群人“混熟”
- 找一个自己感兴趣的项目
- 从阅读社区的交流记录开始
  - Issue list
  - Mail list
  - Pull Request list
- 从用户做起，开始自己下载、编译、安装、使用——然后掉在坑里
- 做一个合格的用户：如何报告Bug？
- 尝试自己修复这个Bug
- 尝试在社区里回答别人的问题
- 在社区里找任务，由易到难

谢谢