

Table of Contents

Introduction	1.1
README	1.2
《代码英雄》第一季	1.3
01 操作系统战争-上	1.3.1
02 操作系统战争-下-Linux 崛起	1.3.2
03 敏捷革命	1.3.3
04 DevOps, 拆掉那堵墙	1.3.4
05 容器竞赛	1.3.5
06 揭秘云计算	1.3.6
07 开启未来	1.3.7
《代码英雄》第二季	1.4
01 按下启动键	1.4.1
02 Hello, World	1.4.2
03 准备提交	1.4.3
04 更好的失败	1.4.4
05 关于 DevSecOps 的故事	1.4.5
06 数据大爆炸	1.4.6
07 无服务器	1.4.7
08 开源好奇号	1.4.8
09 特别篇-开发者推广大使圆桌会议	1.4.9
《代码英雄》第三季	1.5
01 Python 的故事	1.5.1
02 学习 BASIC	1.5.2
03 创造 JavaScript	1.5.3
04 深入 Perl 语言的世界	1.5.4
05 基础设施的影响	1.5.5
06 Bash Shell 中的英雄	1.5.6
07 与机器对话	1.5.7
08 C 语言之巨变	1.5.8
《代码英雄》第四季	1.6
01 小型机 —— 旧机器的灵魂	1.6.1
02 大型机 GE-225 和 BASIC 的诞生	1.6.2
03 个人计算机 —— Altair 8800 和革命的曙光	1.6.3
04 软盘 —— 改变世界的磁盘	1.6.4

LCRH - LCTT SIG For CommandLineHeroes



加入我们

请先加入 QQ 群 **940139452** 了解详情

如何参与

请阅读 [WIKI](#)。如需要协助，请在群内发问。

LICENSE

see [LICENSE](#)

[重制版]《代码英雄》第一季（1）：操作系统战争（上）

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频播客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《[代码英雄](#)》系列播客第一季（1）：操作系统战争（上）的[音频](#)脚本。

Saron Yitbarek: 有些故事如史诗般，惊险万分，在我脑海中似乎出现了星球大战电影开头的滑动文本。你知道的，就像 ——

配音：“第一集，操作系统大战”

Saron Yitbarek: 是的，就像那样子。

Bill Gates Steve Jobs

00:00:30 - 配音：这是一个局势加剧紧张的时期。比尔·盖茨与史蒂夫·乔布斯的帝国发起了一场无可避免的专有软件之战。盖茨与 IBM 结成了强大的联盟，而乔布斯则拒绝开放它的硬件和操作系统授权。他们争夺统治地位的战争，简直席卷了操作系统的“银河系”。与此同时，在这些“帝王们”所不知道的偏远之地，信奉开源的“反叛者们”开始聚集。

00:01:00 - Saron Yitbarek: 好吧。这也许有点戏剧性，但当我们谈论上世纪八九十年代和 2000 年左右的操作系统之争时，这也不算言过其实。确实曾经发生过一场史诗级的统治之战。史蒂夫·乔布斯和比尔·盖茨确实掌握着许多人的命运。掌控了操作系统，你就掌握了绝大多数人使用计算机的方式、互相通讯的方式、获取信息的方式。我可以一直罗列下去，不过你知道我的意思。掌握了操作系统，你就是帝王。

00:01:30 - Saron Yitbarek: 我是 Saron Yitbarek，你现在收听的是代码英雄，一
Command Line Hero

款红帽公司原创的博客节目。你问什么是代 码 英 雄？嗯，如果你愿意用创造代替使用，如果你相信开发者拥有构建美好未来的能力，如果你希望拥有一个大家都有权利用科技塑造生活的世界，那么你，我的朋友，就是一位代码英雄。在本系列节目中，我们将为你带来那些“白码起家”（LCTT 译注：原文是 “from the command line up”，应该是演绎自 “from the ground up” —— 白手起家）改变技术的程序员故事。

00:02:00 - Saron Yitbarek: 那么我是谁，凭什么指引你踏上这段艰苦的旅程？

Saron Yitbarek 是哪根葱？嗯，事实上我觉得我跟你差不多。我是一名初学者服务的开发人员，我做的任何事都依赖于开源软件，我的世界就是如此。通过在博客中讲故事，我可以跳出无聊的日常工作，鸟瞰全景，希望这对你也一样有用。

00:02:30 - Saron Yitbarek: 我迫不及待地想知道，开源技术从何而来？我的意思
Linus Torvalds

是，我对林纳斯·托瓦兹和 Linux® 的荣耀有一些了解，我相信你也一样。但是说真的，开源并不是一开始就有对吗？如果我想发表对这些最新、最棒的技术 —— 比如 DevOps 和容器的感激，我感觉我亏欠那些早期的开发者许多，我有必要了解这些东西来自何处。所以，让我们暂时先不用担心内存泄露和缓冲溢出。我们的旅程将从操作系统之战开始，这是一场波澜壮阔的桌面控制之战。

00:03:00 - Saron Yitbarek: 这场战争亘古未有，因为：首先，在计算机时代，大公司拥有指数级的规模优势；其次，从未有过这么一场控制争夺战是如此变化多端。比尔·盖茨和史蒂夫·乔布斯？目前为止他们也不知道事情会如何发展，但是到这个故事进行到一半的时候，他们所争夺的所有东西都将发生改变、进化，最终上升到云端。

00:03:30 - Saron Yitbarek: 好的，让我们回到 1983 年的秋季，还有六年我才出
Ronald Reagan

生。那时候的总统还是罗纳德·里根，美国和苏联扬言要把地球拖入核战争之中。在檀香山（火奴鲁鲁）的市政中心正在举办一年一度的苹果公司销售会议。一群苹果公司的员工正在等待史蒂夫·乔布斯上台。他 28 岁，热情洋溢，看起来非常自信。乔布斯很严肃地对着麦克风说，他邀请了三个行业专家，来就他的软件进行了一次小组讨论。

00:04:00 - Saron Yitbarek: 然而随后发生的事情你肯定想不到。超级俗气的 80 年代音乐响彻整个房间，一堆多彩灯管照亮了舞台，然后一个播音员的声音响起

配音：女士们，先生们，现在是麦金塔软件的约会游戏时间。

00:04:30 - Saron Yitbarek: 当乔布斯意识到这三个 CEO 都要向他轮流示好的时候，脸上露出一个大大的笑容。他简直就是 80 年代科技界的钻石王老五。两个软件大佬讲完话后就轮到第三个人讲话了，事情就这样结束了？才不是呢。新面孔比尔·盖茨带着一个大大的，遮住了半张脸的方框眼镜。他宣称在 1984 年，微软的一半收入将来自于麦金塔软件。他的这番话引来了观众热情的掌声。

00:05:00 - Saron Yitbarek: 但是他们不知道的是，在一个月后，比尔·盖茨将会宣布发布 Windows 1.0 的计划。你永远也猜不到乔布斯正在跟苹果未来最大的敌人打情骂俏，但微软和苹果即将举行科技史上最糟糕的婚礼。他们会彼此背叛、相互毁灭，但又深深地、痛苦地捆绑在一起。

00:05:30 - James Allworth: 我猜从哲学角度上来讲，苹果是更理想化、注重用户体验高于一切，一体化的组织，而微软则更务实，更模块化——

Saron Yitbarek: 这位是 James Allworth。他是一位高产的科技作家，曾在苹果零售的企业团队工作。注意他对苹果的定义，一个一体化的组织，那种只对自己负责，不想依赖别人的公司，这是关键。

00:06:00 - James Allworth: 苹果是一家一体化的公司，它希望专注于令人愉悦的用户体验，这意味着它希望控制整个技术栈以及交付的一切内容：从硬件到操作系统，甚至运行在操作系统上的应用程序。新的，重要的创新需要横跨软硬件才能很好地进入市场。当你能够根据自己意愿来改变软件和硬件时，你就有了极大的优势。例如——

00:06:30 - Saron Yitbarek: 很多人喜欢这种一体化的模式，并因此成为了苹果的铁杆粉丝，不过还有是很多人则选择了微软。让我们回到檀香山的销售会议上，在同一场活动中，乔布斯向观众展示了他即将发布的超级碗广告，你可能已经亲眼见 George Orwell 过这则广告了。想想乔治·奥威尔的《一九八四》。在这个冰冷、灰暗的世界里，无意识的机器人正在独裁者投射的凝视下徘徊。

00:07:00 - Saron Yitbarek: 这些机器人就像是 IBM 的用户们。然后，代表苹果 Anya Major 公司的，漂亮而健美的安娅·梅杰穿着鲜艳的衣服跑过大厅。她向着大佬们的屏幕猛地投出大锤，将它砸成了碎片。老大哥的咒语解除了，一个低沉的声音响起，苹果公司要开始介绍麦金塔了。

配音：这就是为什么我们的 1984 年，跟小说《一九八四》描写的不一样。

00:07:30 - Saron Yitbarek: 是的，现在回顾那则广告，认为苹果是一个致力于解放大众的自由斗士的想法有点过分，但这件事触动了我的神经。Ken Segal 曾在为苹果制作这则广告的广告公司工作过，他为史蒂夫·乔布斯工作了十多年。

00:08:00 - Ken Segal: 1984 这则广告的负担的风险很大。事实上，它的风险实在太大，乃至苹果公司在看到它的时候都不想播出它。你可能听说了史蒂夫喜欢它，但苹果公司董事会的人并不喜欢它。事实上他们很愤怒，为什么这么多钱被花在这样一件事情上，以至于他们想解雇广告代理商。史蒂夫则为我们公司辩护。

Saron Yitbarek: 乔布斯一如既往地，慧眼识英雄。

Ken Segal: 这则广告在公司内、在业界内都引起了共鸣，成为了苹果产品的代表。无论人们那天是否有在购买电脑，它都带来了一种持之以恒的影响，并让大家在心里定义了这家公司的立场：我们是叛军，我们是拿着大锤的人。

00:08:30 - Saron Yitbarek: 因此，在争夺数十亿潜在消费者心智的过程中，苹果公司和微软公司的帝王们正在学着把自己塑造成救世主、非凡的英雄，选择自己就是选择一种生活方式。但比尔·盖茨明白一些苹果难以理解的事情，那就是在一个相互连接的世界里，没有人——即便他是帝王，能独自完成任务。

00:09:00 - Saron Yitbarek: 1985 年 6 月 25 日。盖茨给当时的苹果 CEO John Scully 发了一份备忘录。那是一个迷失的年代。乔布斯刚刚被逐出公司，直到 1996 年才回到苹果。也许正是因为乔布斯离开了，盖茨才敢写这份东西。在备忘录中，他鼓励苹果授权制造商分发他们的操作系统。我想读一下备忘录的最后部分，让你们知道这份备忘录是多么的有洞察力。

00:09:30 - Saron Yitbarek: 盖茨写道：“如果没有其他个人电脑制造商的支持，苹果现在不可能让他们的创新技术成为标准。苹果必须开放麦金塔的架构，以获得个人建造商的支持来快速发展和建立标准。”换句话说，你们不要再自己玩自己的了。你们必须有与他人合作的意愿。你们必须与开发者合作。

00:10:00 - Saron Yitbarek: 多年后你依然可以看到这条思想的哲学性，当微软首席执行官史蒂夫·鲍尔默上台做主题演讲时，他开始大喊：“开发者、开发者、开发者、开发者、开发者、开发者、开发者、开发者、开发者。”你懂我的意思了吧。微软喜欢开发人员。虽然目前（LCTT 译注：本播客发布于 2018 年初）他们不打算与这些开发人员共享源代码，但是他们确实想建立起整个为合作伙伴服务的生态系统。

00:10:30 - Saron Yitbarek: 而当比尔·盖茨建议苹果公司也这么做时，如你可能已经猜到的，这个想法被苹果公司抛到了九霄云外。他们的关系产生了间隙，五个月后，微软发布了 Windows 1.0。战争开始了。

开发者、开发者。

00:11:00 - Saron Yitbarek: 你正在收听的是来自红帽公司的原创播客《代码英雄》。本集是第一集，我们将回到过去，重温操作系统战争的史诗，我们将会发现，科技巨头之间的战争，是如何为我们今天所生活的开源世界开辟前路的。

00:11:30 - Saron Yitbarek: 好的，让我们先来个背景故事吧，它很经典。如果你已经听过了，那么请原谅我。当时是 1979 年，史蒂夫·乔布斯开车去帕洛阿尔托的 Xerox Park research center 施乐公园研究中心。那里的工程师一直在为他们所谓的图形用户界面，开发一系列的元素。也许你听说过，它们有菜单、滚动条、按钮、文件夹和层叠的窗口。这是对计算机界面的一个前所未有的美丽新设想。作家兼记者 Steve Levy 谈到了它的潜力。

00:12:00 - Steven Levy: 这个新界面有很多令人感到激动的地方，它比以前的交互界面更友好，以前用的交互界面被称为命令行——这不是在现实生活中使用的交互方式。鼠标和电脑上的图像，让你可以像指向现实生活中的东西一样，指向电脑上的东西。这让事情变得简单多了，你不需要记住那些代码。

00:12:30 - Saron Yitbarek: 不过，施乐的高管们并没有意识到他们正坐在金矿上。一如既往地，工程师比主管们更清楚它的价值。因此那些工程师，在被要求向乔布斯展示这些东西是如何工作时，有点紧张。然而这毕竟是高管的命令。用乔布斯的话来说，他认为“这个天才产品本来能够让施乐公司垄断整个行业，可是它最终会被公司的经营者毁掉，因为他们对产品的好坏没有概念。”

00:13:00 - Saron Yitbarek: 这话有些苛刻，但是，乔布斯带着一卡车施乐高管忽视的想法离开了会议。这几乎包含了所有，他革新桌面计算体验需要的东西。1983 年，苹果发布了 Lisa 电脑，1984 年又发布了 Mac 电脑。这些设备的创意都是抄袭自施乐公司的。

00:13:50 - Saron Yitbarek: 让我感兴趣的是，乔布斯对控诉他偷了图形用户界面的反应。他对此很冷静，他引用毕加索的话：“好的艺术家抄袭，伟大的艺术家偷窃。”他告诉一位记者，“我们总是无耻地窃取伟大的创意。”伟大的艺术家偷窃，好吧，我的意思是，我们说的并不是严格意义上的“偷窃”。没人拿到了专有的源代码

并公然将其集成到他们自己的操作系统中去。这事情更温和些，更像是创意的借用。但乔布斯自己即将学到，这东西很难以控制。传奇的软件奇才、真正的代码英雄 **Andy Hertzfeld** 就是麦金塔开发团队的最初成员。

00:14:00 - Andy Hertzfeld: 是的，微软是麦金塔电脑软件的第一个合作伙伴。当时，我们并没有把他们当成是竞争对手。他们是苹果之外，我们第一家交付麦金塔电脑原型的公司。我通常每周都会和微软的技术主管聊一次，他们是第一个试用我们所编写软件的外部团队。

00:14:30 - Andy Hertzfeld: 他们给了我们非常重要的反馈，总的来说，我认为我们的关系非常好。但我也注意到，在我与技术主管的交谈中，他开始问一些系统实现方面的问题，而他本无需知道这些，我觉得他们想要复制麦金塔电脑。我很早以前就向史蒂夫·乔布斯反馈过这件事，但在 1983 年秋天，这件事发展到了高潮。

00:15:00 - Andy Hertzfeld: 我们发现，他们在 1983 年 11 月的 COMDEX 上发布了 Windows，但却没有提前告诉我们。对此史蒂夫·乔布斯勃然大怒，他认为那是一种背叛。

00:15:30 - Saron Yitbarek: 很明显，微软从苹果那里学到了，苹果从施乐那里学来的所有想法。随着新版 Windows 的发布，乔布斯变得很易怒。他发表的伟大艺术家善于偷窃的毕加索名言被别人学去了——盖茨也正是这么做的。据报道，当乔布斯怒斥盖茨偷了他们的东西时，盖茨回应道：“史蒂夫，我觉得这更像是我们都有一名叫施乐的富有邻居，我闯进他家偷电视机，却发现你已经偷过了”。苹果最终以窃取 GUI 的外观和风格为名起诉了微软。这个案子持续了好几年，但是在 1993 年，第 9 巡回上诉法院的一名法官最终站在了微软一边。

00:16:00 - Saron Yitbarek: Vaughn Walker 法官宣布外观和风格不受版权保护，这是非常重要的事情。这一决定让苹果再无法垄断桌面计算的界面。很快，苹果短暂的领先优势消失了。以下是 Steven Levy 的观点。

00:16:30 - Steven Levy: 他们之所以失去领先地位，不是因为微软方面窃取了知识产权，而是因为他们无法巩固自己在上世纪 80 年代就拥有的，更优越的操作系统的优势。坦率地说，他们的电脑索价过高。微软从 20 世纪 80 年代中期开始开发 Windows 系统，但直到 1990 年才开发出来 Windows 3。我想，这算是第一个开启黄金时代的版本，真正可供大众使用。

00:17:00 - Steven Levy: 从此以后，微软能够将数以亿计的用户迁移到图形界面，而这是苹果无法做到的。虽然苹果公司有一个非常好的操作系统，但是那已经是 1984 年的产品了。

00:17:30 - Saron Yitbarek: 现在微软主导着操作系统的战场。他们占据了 90% 的市场份额，并且针对各种各样的个人电脑进行了标准化。操作系统的未来看起来会由微软掌控。在此后发生了什么？1997 年，波士顿 Macworld 博览会上，你看到了一个几近破产的苹果，一个比之前谦逊得多的史蒂夫·乔布斯走上舞台，开始谈论伙伴关系的重要性——特别是他们与微软的。史蒂夫·乔布斯呼吁双方缓和关系，停止火拼，微软将坐享巨大的市场份额。从表面看，我们可能会认为世界和平了。

00:18:00 - Saron Yitbarek: 但当利益如此巨大时，事情就没那么简单了。就在苹果和微软在数十年的争斗中伤痕累累、最终败退到死角之际，一名 21 岁的芬兰计算机科学专业学生出现了。十分偶然的，他彻底改变了一切。

我是 Saron Yitbarek，这里是代码英雄。

00:18:30 - Saron Yitbarek: 正当某些科技巨头正忙着就专有软件相互攻击时，自由软件和开源软件的新领军者如雨后春笋般涌现。其中一位优胜者就是 Richard Stallman

理查德·斯托曼，你也许对他的工作很熟悉，他想要拥有自由软件的自由社会。

free free

这就像言论自由一样的自由，而不是像免费啤酒一样的免费。早在 80 年代，斯托尔曼就发现，除了昂贵的专有操作系统（如 UNIX）外，没有其他可行的替代品，

Free Software Foundation

因此他决定自己做一个。斯托尔曼的自由软件基金会开发了 GNU，当然，它的意思是“GNU's not UNIX”。它将是一个像 UNIX 一样的操作系统，但不包含 UNIX 代码，而且所有用户可以自由共享它。

00:19:00 - Saron Yitbarek: 为了让你体会到上世纪 80 年代自由软件概念的重要性，从不同角度来说拥有 UNIX 代码的两家公司，AT&T 贝尔实验室以及

UNIX System Laboratories

UNIX 系统实验室威胁将会起诉任何看过 UNIX 源代码后又创建自己操作系统的。这些人是次级专利所属。

00:19:30 - Saron Yitbarek: 用这两家公司的话来说，所有这些程序员都在“精神上受到了污染”，因为他们都见过 UNIX 代码。在 UNIX 系统实验室和

Berkeley Software Design

伯克利软件设计公司之间的一个著名的法庭案例中，有人认为即使它本身没有使用 UNIX 代码，拥有类似功能的系统也侵犯版权。Paul Jones 当时是一名开发人员。他现在是数字图书馆 ibiblio.org 的主管。

00:20:00 - Paul Jones: 他们的观点是，任何看过代码的人，都受到了精神污染。因此，几乎所有在安装有与 UNIX 相关操作系统的电脑上工作过的人，以及任何在计算机科学部门工作的人都受到精神上的污染。在 USENIX 的一年里，我们都得到了一个写着红色字母的白色小别针，上面写着“精神受到了污染”。我们很喜欢带着这些别针到处走，以表达我们曾经跟着贝尔实验室混，所以我们的精神受到了污染。

00:20:30 - Saron Yitbarek: 整个世界都被精神污染了。想要保持纯粹、保持事物的美好，和完整地拥有一个软件的旧思想正变得越来越不现实。正是在这被污染的现实中，历史上最伟大的代码英雄之一诞生了，他是一个芬兰男孩，名叫

Linus Torvalds

林纳斯·托瓦兹。如果这是《星球大战》，那么林纳斯·托瓦兹就是我们的 Luke Skywalker

卢克·天行者。他是赫尔辛基大学的一名温文尔雅的研究生。

00:21:00 - Saron Yitbarek: 有才华，但缺乏大志，典型的被逼上梁山的英雄。和其他年轻的英雄一样，他也感到沮丧。他想把 386 处理器整合到他的新电脑中。他对自己兼容 IBM 的电脑上运行 MS-DOS 操作系统并不感冒，也负担不起 UNIX 软件 5000 美元的价格，但只有 UNIX 才能给他一些自由编程的空间。解决方案是，托瓦兹在 1991 年春天基于 MINIX 开发了一个名为 Linux 的操作系统内核，他自己的操作系统内核。

00:21:30 - Steven Vaughan-Nichols: 林纳斯·托瓦兹真的只是想找点乐子而已。

Saron Yitbarek: Steven Vaughan-Nichols 是 ZDNet.com 的特约编辑，而且他从科技行业出现以来就一直在写科技行业相关的内容。

Steven Vaughan-Nichols: 当时有几个类似的操作系统。他最关注的是一个名叫 MINIX，旨在让学生学习如何构建操作系统的项目。林纳斯看到了这些并且觉得这个它很有趣，所以他打算建立自己的操作系统。

00:22:00 - Steven Vaughan-Nichols: 所以, Liunux 实际上始于赫尔辛基的一个 DIY 项目。一切就这样开始了, 基本上就是一个大孩子在玩耍, 学习如何做些什么。但不同之处在于, 他足够聪明、足够执着, 也足够友好, 能让其他人都参与进来, 然后他开始把这个项目进行到底。

00:22:30 - Steven Vaughan-Nichols: 27 年后, 这个项目变得比他想象的要大得多。

00:23:00 - Saron Yitbarek: 到 1991 年秋季, 托瓦兹发布了 10000 行代码, 世界各地的人们开始评头论足, 然后进行优化、添加和修改代码。对于今天的开发人员来说似乎很正常, 但请记住, 在那个时候, 微软、苹果和 IBM 已经在系统开发上做得很完善, 虽然同时这些软件也是专有的。因此像这样的开放协作操作系统, 是对这些公司一种精神上的侮辱, 但随后这种开放性被奉上神坛。托瓦兹将 Linux 置于

GPL 于 GNU 通用公共许可证之下。曾经保障斯托尔曼的 GNU 系统自由的许可证, 现在也将保障 Linux 的自由。Vaughan-Nichols 解释道, GPL 许可基本上能永远保证软件的自由和开放性, 它的重要性被怎么强调都不过分。

00:23:30 - Steven Vaughan-Nichols: 事实上, 根据 Linux 所遵循的许可协议, 即 GPL 第 2 版, 如果你想贩卖 Linux 或者向全世界展示它, 你必须与他人共享代码。所以如果你对其做了一些改进, 仅仅给别人使用打包的代码是不够的, 你必须和他们分享这些变化的所有细节代码。然后这些改进足够好时, 也许就会被 Linux 所吸收。

00:24:00 - Saron Yitbarek: 事实证明, 这种公开的方式极具吸引力。埃里克·雷蒙德</rt>Eric Raymond</rt> 是这场运动的早期传道者之一, 他在他那篇著名的文章中写道: “微软和苹果这样的公司一直在试图建造软件大教堂, 而 Linux 及类似的软件则提供了一个由不同议程和方法组成的大集市, 集市比大教堂有趣多了。”

Stormy Peters: 我认为在那个时候, 真正吸引人们的是——他们终于可以掌控这个属于他们的世界了。

Saron Yitbarek: Stormy Peters 是一位行业分析师, 也是自由和开源软件的倡导者。

00:24:30 - Stormy Peters: 当开源软件第一次出现的时候, 所有的操作系统都是专有的。如果不使用专有软件, 你甚至不能添加打印机, 不能添加耳机, 不能自己开发一个小型硬件设备, 然后让它在你的笔记本电脑上运行。你甚至不能放入 DVD 并复制它, 因为你不能改变软件, 即使你拥有这张 DVD, 你也无法改变它的内容。

00:25:00 - Stormy Peters: 你无法掌控你购买的硬件 / 软件系统。你不能从中创造出任何新的、更大的、更好的东西。这就是为什么开源操作系统在一开始就如此重要。我们需要一个开源协作环境, 在那里我们可以构建更大更好的东西。

00:25:30 - Saron Yitbarek: 请注意, Linux 并不是一个纯粹的平等乌托邦。林纳斯·托瓦兹并没有负责批准所有对内核的修改, 但他主导了内核的变更。他安排了十几个人来管理内核的不同部分。这些人也会信任自己下面的人, 以此类推, 形成信任金字塔。变化可能来自任何地方, 但它们都经过了判断和策划。

00:26:00 - Saron Yitbarek: 然而, 考虑到到林纳斯的 DIY 项目一开始是多么的简陋和随意, 这项成就令人十分惊讶。他完全不知道自己就是这一切中的卢克·天行者。当时他已经编程了半辈子, 但当时只有 21 岁。不过当魔盒第一次被打开

时，人们开始给他反馈。几十个，然后几百个，成千上万的贡献者。有了这样的众包基础，Linux 很快就开始成长，而且成长得很快，甚至最终引起了微软的注意。

Steve Ballmer

他们的首席执行官史蒂夫·鲍尔默将 Linux 称为是“一种癌症，从知识产权的角度来看，它传染了任何它接触的东西”。Steven Levy 将会描述 Ballmer 的立场。

00:26:30 - Steven Levy: 一旦微软真正巩固了它的垄断地位——而且它也确实被联邦法院判定为垄断，他们将会对任何可能对其构成威胁的事情做出强烈反应。它们很自然的将自由软件看成是一种癌症，因为微软要通过软件来赚钱。他们试图提出一个知识产权理论，来解释为什么开源对消费者不利。

00:27:00 - Saron Yitbarek: Linux 在不断传播，微软也开始担心起来。到了 2006 年，Linux 成为仅次于 Windows 的第二大常用操作系统，约有 5000 名开发者在世界各地开发它。5000 名开发者！还记得比尔·盖茨给苹果公司的备忘录吗？在那份备忘录中，他向苹果公司的员工们论述了与他人合作的重要性。事实证明，开源将把伙伴关系的概念提升到一个全新的水平，这是比尔·盖茨从未预见到的。

00:27:30 - Saron Yitbarek: 我们一直在谈论操作系统之间的大战，但是到目前为止，无名英雄和开发者们还没有完全介入战场。在下集中，情况就不同了。第二集讲的还是操作系统大战，关于 Linux 崛起的第二部分。资本们醒悟过来，认识到了开发人员的重要性。

00:28:00 - Saron Yitbarek: 这些开源反叛者变得越来越强大，战场从桌面转移到了服务器领域。这里有商业间谍活动、新的英雄人物，还有科技史上最不可思议的改变。这一切都在操作系统大战的后半集中达到了高潮。

00:28:30 - Saron Yitbarek: 要想免费自动获得新一集的代码英雄，请点击订阅苹果播客、Spotify、Google Play，或其他应用获取该播客。在这一季剩下的时间里，我们将参观最新的战场，相互争斗的版图，这里是下一代的代码英雄留下印记的地方。更多信息，请访问 <https://www.redhat.com/en/command-line-heroes>。我是 Saron Yitbarek，在下次节目之前，请坚持编程。

什么是 LCTT SIG 和 LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。欢迎加入 LCRH SIG：<https://linux.cn/article-12436-1.html>

关于重制版

本系列第一季的前三篇我们已经发布过，这次根据新的 SIG 规范重新修订发布。

via: <https://www.redhat.com/en/command-line-heroes/season-1/os-wars-part-1>

作者：redhat 选题：lujun9972 译者：lujun9972 校对：acyanbird

本文由 LCTT 原创编译，Linux中国 荣誉推出

《代码英雄》第一季（2）：操作系统战争（下）Linux 崛起

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频播客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《[代码英雄](#)》系列播客第一季（2）：操作系统战争（下）的[音频脚本](#)。

Saron Yitbarek: 这玩意开着的吗？让我们播放一段跟星球大战电影一样的开场字幕吧，第二集开始了！

00:00:30 - 配音： OS 战争第二部分：Linux 的崛起。微软帝国控制着 90% 的用桌面用户，操作系统的完全标准化似乎是板上钉钉的事了。所以公司们把它们的注意力从桌面端的个人用户，转移到了专业人士上，它们为了服务器的所有权打得不可开交。但是一个有点让人意想不到的英雄出现在开源“反叛组织”中。戴着眼镜，

Linus Torvalds

固执的林纳斯·托瓦兹免费发布了他的 Linux® 程序。微软摔了个趔趄，并且开始重新调整战略。

00:01:00 - Saron Yitbarek: 哦，我们极客们就是喜欢那样。上一次我们讲到哪了？苹果和微软互相攻伐，试图在争夺桌面用户的战争中占据主导地位。在第一集的结尾，我们看到微软获得了大部分的市场份额。很快，互联网的兴起以及随之而来的开发者大军，让整个市场都经历了一场地震。互联网将战场从在家庭和办公室中的个人电脑用户转移到拥有数百台服务器的大型商业客户中。

00:01:30 - Saron Yitbarek: 这意味着巨量资源的迁移。突然间，所有相关企业不仅被迫为服务器空间和网站建设付费，还必须集成软件来进行资源跟踪和数据库监控等工作。至少那时候大家都是这么做的，你需要很多开发人员来帮助你。

在操作系统之战的第二部分，我们将看到固有优势的巨大转变，以及像林纳斯·托
Richard Stallman
瓦兹和理查德·斯托尔曼这样的开源反叛者，是如何成功地在微软和整个软件行业
的核心引发恐惧的。

00:02:00 - Saron Yitbarek: 我是 Saron Yitbarek，你现在收听的是代码英雄，一
款红帽公司原创的播客节目。每一集，我们都会给你带来“从码开始”，用技术改变
科技的人，他们的故事。

00:02:30 - Saron Yitbarek: 好。想象一下你是 1991 年时的微软。你自我感觉良
好，对吧？你满怀信心，确立了全球主导地位的感觉真不错。你已经掌握了与其他
企业合作的艺术，但是仍然将大部分开发人员、程序员和系统管理员排除在联盟之
外，而他们才是真正的武装力量。这时出现了一个叫林纳斯·托瓦兹的芬兰极客。
他和一帮信奉开源的程序员开始发布 Linux，这个操作系统内核是由他们一起编写
出来的。

00:03:00 - Saron Yitbarek: 坦白地说，如果你是微软公司，你并不会太在意
Linux，甚至不太关心开源运动，但是最终，Linux 的规模变得如此之大，以至于微
软不可能不注意到。Linux 第一个版本出现在 1991 年，当时大概有 1 万行代码。
十年后，则是 300 万行代码。如果你想了解知道现在的 Linux 怎么样了——它有
2000 万行代码。

00:03:30 - Saron Yitbarek: 让我们在 90 年代初停留一会儿，那时 Linux 还没有
成为我们现在所知道的庞然大物。这只是一个正在蔓延的，拥有病毒一般感染力的
奇怪操作系统，不过全世界的极客和黑客都爱上了它。那时候我还太年轻，但有点
希望我曾经历过那个年代。在那个时候，发现 Linux 就如同进入了一个秘密社会，
程序员与朋友们分享刻录的 Linux CD 集，就像其他人分享地下音乐混音带一样。

00:03:40: 开发者 Tristram Oaten 讲讲你 16 岁时第一次接触 Linux 的故事吧。

00:04:00 - Tristram Oaten: 我和我的家人去了红海的 Hurghada 潜水度假，那
是一个美丽的地方，强烈推荐。也许我妈妈跟我说过不要这么做，但第一天，我还是
喝了自来水。所以我整个星期都病得很厉害，没法离开旅馆房间。当时我只带了一
台新安装了 Slackware Linux 的笔记本电脑，我只是听说过这玩意，现在却要尝试
使用它。我手上没有其他的应用程序，我能接触的所有东西只有刻录在 8 张 CD 上
的代码。这种情况下，我整个星期能做的事情，就是了解这个外星造物一般的系
统。我阅读手册，摆弄着终端。我记得当时我甚至不知道一个点（表示当前目录）
和两个点（表示前一个目录）之间的区别。

00:04:30 - Tristram Oaten: 我一点头绪都没有。犯过很多错误。但慢慢的，在这
种被迫造成的孤独中，我突破了障碍，开始明白并理解命令行到底是怎么回事。假
期结束时，我没有看过金字塔、尼罗河等任何埃及遗址，但我解锁了现代世界的一
个奇迹。我接触了 Linux，接下来的事大家都知道了。

Saron Yitbarek: 你会从很多人那里听到关于这个故事的不同版本，访问 Linux 命
令行是一种革命性的体验。

00:05:00 - David Cantrell: 它提供了源代码。我当时的感觉是，“太神奇了。”

Saron Yitbarek: 我们正在参加一个名为 Flock to Fedora 的 2017 年 Linux 开发
者大会。

David Cantrell:非常有吸引力。我觉得随着我掌控这个系统越深，它就越吸
引我。我想，从 1995 年我第一次编译 Linux 内核那时起，我就迷上了它。

Saron Yitbarek: 这是开发者 David Cantrell 与 Joe Brockmire。

00:05:30 - Joe Brockmeier: 我在 Cheap Software 转的时候发现了一套四张 CD 的 Slackware Linux。它看起来会非常令人兴奋而且很有趣，所以我把它带回家，安装在我的第二台电脑上，开始摆弄它，有两件事情让我感到很兴奋：一个是我运行的不是 Windows，另一个是 Linux 的开源特性。

00:06:00 - Saron Yitbarek: 某种程度上来说，使用命令行的人总是存在的。在开源真正开始流行起来的二十年前，人们（至少在开发人员是这样）总是希望能够做到完全控制机器。让我们回到操作系统大战之前的那个时代，在苹果和微软为他们 Paul Jones 的 GUI 而战之前，那时也有代码英雄。保罗·琼斯教授（在线图书馆 [ibiblio.org](http://www.ibiblio.org) 的负责人）就是一名那个古老年代的开发人员。

00:06:30 - Paul Jones: 从本质上讲，互联网那个时候客户端-服务器架构还是比较少的，而更多的是点对点架构的。确实，我们会说，某种 VAX 到 VAX 的连接（LCTT 译注：DEC 的一种操作系统），某种科学工作站到科学工作站的连接。这并不意味着没有客户端-服务端的架构及应用程序，但这的确意味着，最初的设计是思考如何实现点对点，它与 IBM 一直在做的东西相对立。IBM 给你的只有哑终端，这种终端只能让你管理用户界面，却无法让你像真正的终端一样为所欲为。

00:07:00 - Saron Yitbarek: 当图形用户界面在普通用户中普及的同时，在工程师和开发人员中总是存在着一股相反的力量。早在 Linux 出现之前的二十世纪七八十年代，这股力量就存在于 Emacs 和 GNU 中。斯托曼的自由软件基金会中的某些人想要使用完全命令行，但直到上世纪 90 年代的 Linux 出现，才提供了前所未有的东西。

00:07:30 - Saron Yitbarek: Linux 和其他开源软件的早期爱好者都是先驱。我正站在他们的肩膀上，我们都是。

你现在收听的是代码英雄，一款由红帽公司原创的播客。这是操作系统大战的第二部分：Linux 崛起。

Steven Vaughan-Nichols: 1998 年的时候，情况发生了变化。

00:08:00 - Saron Yitbarek: Steven Vaughan-Nichols 是 zdnet.com 的特约编辑，他已经写了几十年从商业方面论述技术的文章了。他将向我们讲述 Linux 是如何慢慢变得越来越流行，直到自愿贡献者的数量远远超过了工作于 Windows 上的微软开发人员的数量。不过，Linux 桌面版本从未真正追上 Windows，这也许就是微软最开始时忽略了 Linux 及其开发者的原因。Linux 真正大放光彩的地方是服务器机房，当企业开始线上业务时，每个企业都需要一个满足其独特需求的解决方案。

00:08:30 - Saron Yitbarek: WindowsNT 于 1993 年问世，当时它已经在与其他的服务器操作系统展开竞争了，但是许多开发人员都在想，“既然我可以通过 Apache 构建出基于 Linux 的廉价系统，那我为什么要购买 AIX 设备或大型 Windows 设备呢？”鉴于这个优点，Linux 代码已经开始渗透到几乎所有的在线机器中。

00:09:00 - Steven Vaughan-Nichols: 让微软开始意识到并感到惊讶的是，Linux 实际上已经有了一些商业应用，不是在桌面环境，而是在商业服务器上。因此，他们发起了 FUD - 恐惧、不确定和怀疑。他们说，“哦，Linux 这玩意，真的没有那么好。它不太可靠，你根本不能相信它”。

00:09:30 - Saron Yitbarek: 这种软宣传式的攻击持续了一段时间。微软也不是唯一一个对 Linux 感到紧张的公司，整个行业其实都在对抗这个奇怪新人的挑战。例如，任何与 UNIX 有利害关系的人都可能将 Linux 视为篡夺者。有一个案例很著名，那就是 SCO 组织（它发行过一种 UNIX 版本）在过去 10 多年里发起一系列的诉讼，试图阻止 Linux 的传播，而 SCO 最终失败而且破产了。与此同时，微软一直在寻找机会，他们必须要采取行动，只是不清楚具体该怎么做。

00:10:00 - Steven Vaughan-Nichols: 第二年，真正让微软担心的事情发生了。在 2000 年的时候，IBM 宣布，他们将于 2001 年投资 10 亿美元在 Linux 上。现在，IBM 已经不再涉足个人电脑业务。那时他们还没有走出那一步，但他们正朝着这个方向前进，他们将 Linux 视为服务器和大型计算机的未来，在这一点上——剧透警告，IBM 是正确的。

00:10:30 - Steven Vaughan-Nichols: Linux 将主宰服务器世界。

Saron Yitbarek: 这已经不再仅仅是一群黑客所钟爱的，对命令行绝地武士式的控制了。事实上金钱的投入对 Linux 助力极大。Linux 国际的执行董事 John “Mad Dog” Hall 有一个可以解释为什么事情会变成这样的故事分享，我们通过电话与他取得了联系。

00:11:00 - John Hall: 我有一个名叫 Dirk Holden 的朋友，他是德国德意志银行的系统管理员，他也参与了个人电脑上早期 X Windows 系统图形项目的工作。有一天我去银行拜访他，我说：“Dirk，你银行里有 3000 台服务器，用的都是 Linux。为什么不用 Microsoft NT 呢？”

00:11:30 - John Hall: 他看着我说：“是的，我有 3000 台服务器，如果使用微软的 Windows NT 系统，我需要 2999 名系统管理员。”他继续说道：“而使用 Linux，我只需要四个。”这真是完美的答案。

00:12:00 - Saron Yitbarek: 程序员们着迷的这些东西恰好对大公司也极具吸引力。但由于 FUD 的作用，一些企业对此持谨慎态度。他们听到开源，就想：“开源。这看起来不太可靠，很混乱，充满了 BUG”。但正如那位银行经理所指出的，金钱有一种有趣的魔力，可以说服人们不再踌躇。甚至那些只需要网站的小公司也加入了 Linux 阵营。与一些昂贵的专有选择相比，使用一个廉价的 Linux 系统在成本上是无法比拟的。如果你是一家雇佣专业人员来构建网站的商店，那么你肯定想让他们使用 Linux。

00:12:30 - Saron Yitbarek: 让我们快进几年。Linux 充当着几乎所有网站的服务器，Linux 已经征服了服务器世界，然后智能手机也随之诞生。当然，苹果和他们的 iPhone 占据了相当大的市场份额，而且微软也希望能进入这个市场。但令人惊讶的是，Linux 已经等在那里并做好准备，迫不及待要大展拳脚。

这是作家兼记者 James Allworth。

00:13:00 - James Allworth: 肯定还有容纳第二个竞争者的空间，那本可以是微软，但是实际上却是 Android，而 Android 是基于 Linux 的。众所周知，Android 被谷歌所收购，现在运行在世界上大部分的智能手机上，谷歌在 Linux 的基础上创建了 Android。Linux 使他们能够以零成本，基于一个非常复杂的操作系统构建一个新的东西。他们盘算着推广这个系统，并最终成功地实现了这一目标。至少从操作系统的角度来看是这样，他们将微软挡在了下一代手机竞争之外。

00:13:30 - Saron Yitbarek: 这可是个大地震，很大程度上，微软有被埋没的风险。John Gossman 是微软 Azure 团队的首席架构师。他还记得当时公司的迷茫。

00:14:00 - John Gossman: 像许多公司一样，微软也非常担心知识产权污染。他们认为，如果允许开发人员使用开源代码，即使只是将一些代码复制粘贴到某些产品中，也会让某种病毒式的许可证生效从而引发未知的风险……他们也很困惑，我认为，这跟公司文化有关，很多公司，包括微软，都对开源开发的意义和商业模式之间的分歧感到困惑。有一种观点认为，开源意味着你所有的软件都是免费的，人们永远不会付钱。

00:14:30 - Saron Yitbarek: 任何习惯于投资于旧的、专有软件模式的人都会觉得这里发生的一切对他们构成了威胁。当你威胁到像微软这样的大公司时，是的，他们一定会做出反应。他们推动所有这些 FUD —— 恐惧、不确定性和怀疑是有道理的。当时，商业运作的方式基本上就是相互竞争。不过，如果是其他公司的话，他们可能还会一直怀恨在心，抱残守缺，但到了 2013 年的微软，一切都变了。

00:15:00 - Saron Yitbarek: 微软的云计算服务 Azure 上线了，令人震惊的是，
Steve Ballmer

它从第一天开始就提供了 Linux 虚拟机。史蒂夫·鲍尔默，这位把 Linux 称为癌症的首席执行官，已经离开了，代替他的是位新的有远见的首席执行官 Satya Nadella
萨提亚·纳德拉。

John Gossman: 萨提亚有不同的看法，他属于另一个世代。比保罗、比尔和史蒂夫更年轻的世代，他对开源有不同的看法。

Saron Yitbarek: 这是 John Gossman，他还是来自微软 Azure 团队。

00:15:30 - John Gossman: 大约四年前，出于实际需要，我们在 Azure 中添加了 Linux 支持。如果访问任何一家企业客户，你都会发现他们并不是现在才决定是使用 Windows 还是使用 Linux、使用 .net 还是使用 Java。他们在很久以前就做出了决定 —— 大约 15 年前，虽然对此有一些争论。

00:16:00 - John Gossman: 现在，我见过的每一家公司都混合了 Linux 和 Java、Windows 和 .net、SQL Server、Oracle 和 MySQL —— 基于专有源代码的产品和开放源代码的产品。

如果你打算运维一个云服务，允许这些公司在云上运行他们的业务，那么你根本不能告诉他们，“你可以使用这个软件，不能使用那个软件。”

00:16:30 - Saron Yitbarek: 这正是萨提亚·纳德拉采纳的哲学思想。2014 年秋季，他站在舞台上，希望传递一个重要信息。“微软爱 Linux”。他接着说，“20% 的 Azure 业务已经是 Linux 了，微软将始终对 Linux 发行版提供一流的支持。”没有哪怕一丝对开源的宿怨。

为了说明这一点，在他们的背后有一个巨大的标志，上面写着：“Microsoft ❤️ Linux”。哇噢。对我们中的一些人来说，这种转变有点令人震惊，但实际上，无需如此震惊。下面是 Steven Levy，一名科技记者兼作家。

00:17:00 - Steven Levy: 当你在踢足球的时候，如果草坪变滑了，那么你也许会换种不同的鞋子，才能继续在上面玩耍。微软当初就是这么做的。

00:17:30 - Steven Levy: 他们不能否认现实，而且他们也是聪明人。所以他们必须意识到，这就是这个世界的运行方式。即使他们有一点尴尬，但是不管他们早些时候说了什么现在都要抛开。不然让他们之前那些“开源多么可怕”的言论影响到现在的决策，才真的是不明智之举。

00:18:00 - Saron Yitbarek: 微软低下了它高傲的头。你可能还记得苹果公司，经过多年 的孤立无援，最终转向与微软构建合作伙伴关系。现在轮到微软进行 180 度转变了。经过多年的与开源方式的战斗后，他们正在重塑自己。要么改变，要么死亡。下一个发言的是 Steven Vaughan-Nichols。

00:18:30 - Steven Vaughan-Nichols: 即使是像微软这样规模的公司，也无法与数千个开发者包括 Linux 在内的其它开源大项目的开发者竞争。很长时间以来他们都不愿意合作，前微软首席执行官史蒂夫·鲍尔默对 Linux 用接近信仰的方式深恶痛绝。由于它使用的 GPL 许可证，他视 Linux 为一种癌症。不过一旦鲍尔默被扫地出门，新的微软领导层表示，“这就好像试图命令潮流不要过来，但潮水依然会不断涌进来。我们应该与 Linux 合作，而不是与之对抗。”

00:19:30 - Steven Vaughan-Nichols: 2017 年的微软既不是史蒂夫·鲍尔默的微软，也不是比尔·盖茨的微软。这是一家完全不同的公司，有着完全不同的理念，而且，一旦使用了开源，你就无法退回到之前的阶段。开源已经吞噬了整个技术世界。从未听说过 Linux 的人可能对它并不了解，但是每次他们访问 Facebook，他们都在运行 Linux。每次执行谷歌搜索时，你都在运行 Linux。

00:20:00 - Steven Vaughan-Nichols: 每次你用 Android 手机，你都在运行 Linux。事实上它早已无处不在，微软无法阻止它，而且我认为，以为微软想用某种方式接管它的想法太天真了。

00:20:30 - Saron Yitbarek: 开源支持者可能一直担心微软会像混入羊群中的狼一样，但事实是，开源软件的本质保护了它，让它无法被完全控制。没有一家公司能够拥有 Linux 并以某种特定的方式控制它。Greg Kroah-Hartman 是 Linux 基金会的一名成员。

Greg Kroah-Hartman: 每个公司和个人都因为自己的利益对 Linux 做出贡献。他们之所以这样做是因为他们想要解决他们所面临的问题，可能是硬件无法工作，或者是他们想要添加一个新功能来做其他事情，又或者想引导 Linux 的开发轨道，这样他们的新产品就能使用它。这很棒，因为他们会把代码贡献回去，此后每个人都会从中受益，这样每个人都可以用到这份代码。正是因为这种自私，所有的公司，所有的人都能从中受益。

00:21:00 - Saron Yitbarek: 微软已经意识到，在即将到来的云战争中，与 Linux 作战就像与空气作战一样。Linux 和开源不是敌人，它们是空气。如今，微软以白金会员的身份加入了 Linux 基金会。他们成为 GitHub 开源项目的头号贡献者。

00:21:30 - Saron Yitbarek: 2017 年 9 月，他们甚至加入了 Open Source Initiative 开源促进联盟 (OSI)。现在，微软在开源许可证下发布了很多代码。微软的 John Gossman 描述了他们开源 .net 时所发生的事情。起初，他们并不认为自己能得到什么回报。

00:22:00 - John Gossman: 我们本没有指望来自社区的贡献，然而，三年后，超过 50% 的对 .net 框架库的贡献来自于微软之外，这些是大量的代码。三星为 .net 提供了 ARM 支持。Intel 和 ARM 以及其他一些芯片厂商已经为 .net 框架贡献了特定于他们处理器的代码，以及数量惊人的修复 bug、性能改进等等——既有单个贡献者也有社区。

Saron Yitbarek: 直到几年前，今天的这个微软，这个开放的微软，还是不可想象的。

00:22:30 - Saron Yitbarek: 我是 Saron Yitbarek，这里是代码英雄。好吧，我们已经看到了为了赢得数百万桌面用户的爱而战的激烈场面。我们已经看到开源软件在专有软件巨头的背后悄然崛起，并攫取了巨大的市场份额。

00:23:00 - Saron Yitbarek: 我们已经看到了一批批的代码英雄将编程领域变成了我今天看到这个样子。如今，大企业正在吸收开源软件，通过这一切，每个人都从他人那里受益。

00:23:30 - Saron Yitbarek: 在技术的西部荒野，一贯如此。苹果受到施乐的启发，微软受到苹果的启发，Linux 受到 UNIX 的启发。进化、借鉴、不断成长。如果比喻成大卫和歌利亚（LCTT 译注：西方经典的以弱胜强战争中的两个主角）的话，开源软件不再是大卫，但是，你知道吗？它也不是歌利亚。开源已经超越了这些角色，它成为了其他人战斗的战场。随着开源变得不可避免，新的战争，那些在云计算中进行的战争，那些在开源战场上进行的战争正在加剧。

这是 Steven Levy，他是一名作者。

00:24:00 - Steven Levy: 基本上，到目前为止，包括微软在内，有四到五家公司，正以各种方式努力把自己打造成为全方位的平台，比如人工智能领域。你能看到智能助手之间的战争，你猜怎么着？苹果有一个智能助手，叫 Siri。微软有一个，叫 Cortana。谷歌有谷歌助手，三星也有一个智能助手，亚马逊也有一个，叫 Alexa。我们看到这些战斗遍布各地。也许，你可以说，最热门的人工智能平台将控制我们生活中所有的东西，而这五家公司就是在为此而争斗。

00:24:30 - Saron Yitbarek: 如果你正在寻找另一个反叛者，它们就像 Linux 奇袭微软那样，偷偷躲在 Facebook、谷歌或亚马逊身后，你也许要等很久，因为正如作家 James Allworth 所指出的，成为一个真正的反叛者只会变得越来越难。

00:25:00 - James Allworth: 规模一直以来都是一种优势，但规模优势本质上……怎么说呢，我认为以前它们在本质上是线性的，现在它们在本质上是指数型的了，所以，一旦你开始以某种方法走在前面，另一个新玩家要想赶上来就变得越来越难了。我认为在互联网时代这大体来说来说是正确的，无论是因为规模，还是数据赋予组织的竞争力的重要性和优势。

00:25:30 - James Allworth: 一旦你走在前面，你就会吸引更多的客户，这就给了你更多的数据，让你能做得更好，这之后，客户还有什么理由选择排名第二的公司呢，难道是因为因为他们落后了这么远么？我认为在云的时代这个逻辑也不会有什么不同。

00:26:00 - Saron Yitbarek: 这个故事始于史蒂夫·乔布斯和比尔·盖茨这样的非凡的英雄，但科技的进步已经呈现出一种众包、自有生命的感觉。我认为据说我们的开源英雄林纳斯·托瓦兹在第一次发明 Linux 内核时甚至没有一个真正的计划。他无疑是一位才华横溢的年轻开发者，但他也像潮汐前的一滴水一样。变革是不可避免的。据估计，对于一家专有软件公司来说，用他们老式的、专有的方式创建一个 Linux 发行版将花费他们超过 100 亿美元。这说明了开源的力量。

00:26:30 - Saron Yitbarek: 最后，这并不是一个专有模型所能与之竞争的东西。成功的公司必须保持开放。这是最大、最终极的教训。还有一点要记住：当我们团结在一起的时候，我们在已有基础上成长和建设的能力是无限的。不管这些公司有多大，我们都不必坐等他们给我们更好的东西。想想那些为了纯粹的创造乐趣而学习编码的新开发者，那些自己动手丰衣足食的人。

未来的优秀程序员不论来自何方，只要能够访问代码，他们就能构建下一个大项目。

00:27:00 - Saron Yitbarek: 以上就是我们关于操作系统战争的故事。这场战争塑造了我们的数字生活的形态，争夺主导地位的斗争从桌面转移到了服务器机房，最终进入了云计算领域。过去的敌人难以置信地变成了盟友，众包的未来让一切都变得开放。

00:27:30 - Saron Yitbarek: 听着，我知道，在这段历史之旅中，还有很多英雄我们没有提到。所以给我们写信吧，分享你的故事。

Redhat.com/commandlineheroes。我恭候佳音。

在本季剩下的时间里，我们将学习今天的英雄们在创造什么，以及他们要经历什么样的战斗才能将他们的创造物带入我们的生活。让我们从壮丽的编程一线，回来看看更多的传奇故事吧。我们每两周放一集新的博客。几周后，我们将为你带来第三集：敏捷革命。

00:28:00 - Saron Yitbarek: 代码英雄是一款红帽公司原创的播客。要想免费自动获得新一集的代码英雄，请订阅我们的节目。只要在苹果播客、Spotify、Google Play，或其他应用中搜索“Command Line Heroes”。然后点击“订阅”。这样你就会第一个知道什么时候有新剧集了。

我是 Saron Yitbarek。感谢收听，在下期节目之前，请坚持编程。

什么是 LCTT SIG 和 LCTT LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。欢迎加入 LCRH SIG：<https://linux.cn/article-12436-1.html>

关于重制版

本系列第一季的前三篇我们已经发布过，这次根据新的 SIG 规范重新修订发布。

via: <https://www.redhat.com/en/command-line-heroes/season-1/os-wars-part-2-rise-of-linux>

作者: [redhat](#) 选题: [lujun9972](#) 译者: [lujun9972](#) 校对: [acyanbird,FineFan](#)

本文由 [LCTT](#) 原创编译, [Linux中国](#) 荣誉推出

《代码英雄》第一季（3）：敏捷革命

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频播客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《[代码英雄](#)》系列播客[第一季（3）：敏捷革命](#)的音频脚本。

现在是 21 世纪之交，开源软件正在改变着科技的格局，现在已经需要一种新的工作模式了。开发者们在寻找一种革命性的方法，让开源开发蓬勃发展。一群开发者在犹他州的一个滑雪场召开了会议，形成的是一份改变一切的宣言。

Manifesto for Agile Software Development Dave Thomas

《敏 捷 软 件 开 发 宣 言》的作者之一戴夫·托马斯将我们带回了那个现在著名的静修之地，敏捷革命就是在那第一次组织起来的。不过，并不是每个人都那么快就签下了这种新方法，在这一集里，我们听听原因。

Saron Yitbarek: 有些故事的走向和结局会重新定义一个行业。在这些故事中也传唱着，我们来自哪里，我们是谁，我们正在做什么。[上一集中](#)，我们追溯了 Linux® 开源技术的崛起。但这一集我要讲的，是紧接着发生的故事。操作系统之战结束后，开发者们成为了战争争夺的对象和核心。

00:00:30: 在那个新的战场，开发者们将要重塑自己的工作。本集播客，我们将深入了解以开发人员为核心，产生的一种全新的软件开发方法论。这种新颖的工作流程产生了哪些远超我们屏幕上显示的代码所能控制的、意想不到的事情。

我是 Saron Yitbarek，欢迎收听红帽的原创播客《代码英雄 第三集 敏捷革命》。今天的故事始于 2001 年 2 月，发生在美国犹他州的滑雪小屋里。

00:01:00 - Dave Thomas: 我们面前有个小屋，眼前是松树梁和壁炉，还有进入屋子的小门。我们前一天晚上到达这里时，然后基本上只是围坐一起，讨论谈我们准备探讨的内容。紧接着第二天，我们都起床，并来到了预定的会议室。先把桌子移到边上去，然后将椅子摆放成一圈，确切地说是一个椭圆，这样以来我们就可以面对面交流，一定程度上也让人感觉到可以敞开心扉，畅所欲言。

00:01:30 - Saron Yitbarek: 刚才提到的这群人都是开源开发人员，所以保持开放是他们的特点。Dave Thomas 和其他的 16 个人，在那个冬天集聚在雪鸟滑雪场。但是他们的目的并不是滑雪，而是探讨在 90 年代开发者的世界所面临的问题。在这里我用“探讨”，但实际上用“辩论”更准确。他们最初是在 Object-Oriented Programming, Languages and Systems 面向对象编程、语言及系统 (OOPSLA) 的会议上认识的，这个会议主要议题是面向对象程序设计、编程、语言和系统。

00:02:00: 实际上正是那次会议，让他们意识到当前的软件开发方式很混乱，只是对于该怎么办没有达成一致。

所以此次在雪鸟山上的开会，试图寻找解决这个问题的方法。那么究竟是这个问题具体该怎么描述？于是我询问 Dave，开发人员之前的使用方式到底出现了什么问题。

Dave Thomas: 所以，我不知道……你有没有装饰过房间？

Saron Yitbarek: 我有。

00:02:30 - Dave Thomas: ……好。如果我先告诉你，“我想让你坐下来，然后给你一张白纸。接着我希望你能描绘下来这个房间完成后大概的样子。”可以想象吗？

Saron Yitbarek: 嗯嗯（肯定的语气）。

Dave Thomas: 你能做到吗？

Saron Yitbarek: 实际上，我的办公室就是这么布置出来的。首先，我画了一个简单的草图，然后加上一些修饰，最后把所有架子摆放在我觉得合适的位置。不过这种方式没有真正起到作用，我的计划也没有实现。

Dave Thomas: 但是，即使你尝试了这种方式，你都做了什么？先把架子放起来，然后说，“哦……这样放不行，因为会挡道。”所以，你又紧接着把架子移到其它地方，或者你会说，“你知道吗，我真的不能把地毯放在那里，因为我的椅子脚会陷进去。”状况频发。

00:03:00 - Dave Thomas: 遇到未知的情况，你总需要一种“迭代”的方式去应对。人类的大脑无法准确地对现实世界的发展进行预判，从而提前预知哪里需要改变。所以，软件开发也是一样的，人们不知道他们的需求会怎么改变。对吗？

Saron Yitbarek: 嗯嗯（肯定的语气）。

00:03:30 - Dave Thomas: 我经历过太多这样的情况，当我从客户那里拿到了详细的要求，然后我已经很好地完成了每一条细则，但却总是吵得不欢而散。“这不是我们想要的。”但我想说的是，“这就是你要求的啊。”他们说，“是的，但这不是我的意思。”你懂这种情况吗？

Saron Yitbarek: 嗯嗯（肯定的语气）。

Dave Thomas: 所以说，理想情况是，你可以详细说明流程的每一步，然后通过非常机械的步骤就可以完成一切。

Saron Yitbarek: 是啊。

00:04:00 - Dave Thomas: 但是在软件行业可行不通。这种方式不适用于有任何模棱两可的情况，也不适用于需要判断的情况。就像任何艺术尝试一样，这种方式就是行不通。总是缺失了关键的一步：反馈。

Saron Yitbarek: 也许你已经听说过上世纪 90 年代的软件危机。当时的软件开发一团糟。相比于开发软件的费用，公司在修复软件上的钱花的多得多。与此同时，你我这样的开发人员进退不得。有时候，我们每隔好几年时间才能推出新的软件。

00:04:30: 我们疲于应付这些缓慢、陈旧、瀑布式开发的工作流程。从 A 到 B 到 C，完全都是提前确定好的。因此，那时的时间消耗主要在寻找新的流程，寻找更好的软件开发方式上。事实上，每个月似乎都有新的开发者，对如何改善软件开发的过程提出宏伟的设想。

00:05:00: 其中就有极限编程、有 Kanban、还有统一软件开发过程等，不胜枚举。在这些方法论的激烈竞争中，也催生出了新的观点和改进方法。那就是 **Dave Thomas** 和他在雪鸟滑雪场的朋友们迫不及待开始探讨的领域。

值得让这群人齐声欢呼喝彩的就是《敏捷软件开发宣言》。当时的开发速度正在以前所未有的速度保持增长——而开源使开发人员变得更强大。另一方面，开发人员也需要一种新的敏捷的开发模式。

00:05:30: 顺便提一下，那些在雪鸟滑雪场会面的人，在经过一番你来我往的争论 **Agile** 后，才确定用这个词。敏捷，这个词非常切题。这种方式就好像国家地理描述大型猫科动物的方式，一个与瀑布式开发预设路径正好相反的词。随着新的信息层出不穷，这个词让那些愿意改变航向的人看到了一线曙光。请注意这可不是一个名词，而是一个形容词。

00:06:00: 敏捷更像是一种习惯，而不是一种具体的说辞。那么，那些采用敏捷的开发者提供了什么呢？他们的总体解决方案是什么？现在很多人都认为敏捷是一个 **scrum master** **scrum** 复杂的集合，包括不同的角色和系统。会有一个项目经理，一个项目团队，一个产品负责人。同时他们都要进行一到两周的冲刺工作。

00:06:30: 与此同时，工作都堆积在“冰盒”和“沙盒”中。好吧，听起来感觉流程很多。但一开始的时候是没有这些流程的。撰写该敏捷宣言的人，目标是简单和清晰。实际上，简单是它制胜的法宝。从那时起，它就具有定义几乎每个开发人员命运之路的能力。

Dave Thomas: 我们已经提到了，我们更喜欢某种方式，而不是另一种方式。事实上，在午餐这段时间，我们就写下了几乎所有的观点，现在都是敏捷宣言的一部分。

00:07:00 - Saron Yitbarek: 这是可以管理开发的四个奇思妙想。如果你尚且还不熟悉那些敏捷的诫命，他们会这样解释：

个体和互动胜过流程和工具；可工作的软件胜过文档；客户协作胜过合同谈判；响应变化胜过遵循计划。

00:07:30: 我记得第一次看到这个宣言时的情形。我刚开始学习编程，老实说，当时我并没有觉得这个想法有多棒。一直到我了解到那些支持敏捷开发的工具和平台。对我来说，这是一些模糊的概念。但是，对于长期以来一直被这些问题纠缠的开发人员来说，这是一个很好的行动方案。

该宣言是一盏灯，可以激发更多奇思妙想。这四点宣言和一些支持材料都发布在 Agilemanifesto.org 网站上，并且呼吁其他开发者签名以表示支持。

00:08:00 - Dave Thomas: 很快获得了 1000 个签名，接着 10,000 个，然后签名数一直在增长，我想我们都惊呆了。这基本上变成了一场革新运动。

Saron Yitbarek: 他们从来没有计划过把这份敏捷宣言带出滑雪小屋。这是一群热衷于软件开发的人，并且对帮助他人更好地发展充满热情。但很明显，“敏捷”本身像长了腿一样。红帽公司首席开发倡导者 Burr Sutter 谈到了“敏捷”对于还困在“瀑布”中的开发人员来说是一种解脱。

00:08:30 - Burr Sutter: 因此，敏捷的概念从根本上引起了人们的共鸣，基本上是在说：“看，我们专注于人员而不是流程。我们专注于交互和协作而不是工具和文档。我们认为工作软件高于一切，我们宁愿人们通过小批量的工作，实现高度互动、快速迭代。”

00:09:00 - Saron Yitbarek: 而对于一些人来说，这个开发者的革新走得太远。敏捷甚至被视为是给那些不负责任的黑客心态的合理说辞。早期反对敏捷最重要的声音之一是 Steve Rakitin。他是一名软件工程师，拥有超过 40 年的行业经验。

00:09:30: 当他大学毕业时，Rakitin 就开始建造第一个核电站数字控制系统。几十年来，他一直致力于研发电力软件和医疗设备软件。这些都是对安全很注重的软件。没错。你可以预料到，他可不会对这种手忙脚乱的开发方式感兴趣。

因此，在方法论战争的尾声，敏捷横空出世，Rakitin 对此翻了个白眼。

Steve Rakitin: 就像是，“好吧，我们换种方式说，如同一群人围坐着喝着啤酒，就想出了开发软件的其他办法。”顺便提一下，敏捷宣言中许多已经得到进一步发展，并应用于早期的开发方法里了。

00:10:00 - Saron Yitbarek: 他这么想其实也没有什么错。实际上你可以在“雪鸟 Kanban 峰会”前几十年就追溯到敏捷哲学的踪迹。例如，像看板这样的精益工作方法可以追溯到 20 世纪 40 年代，当时丰田受到超市货架存货技术的启发……

他们的精益制造理念最终被用于软件开发。不过 Rakitin 有另外一个担忧。

00:10:30 - Steve Rakitin: 这篇宣言发表时我非常怀疑它，因为它基本上是为了让软件工程师花更多的时间编写代码，花更少的时间搞清楚需要做什么，同时记录文档的时间少了很多。

Saron Yitbarek: 对于 Rakitin 来说，这不仅仅是提出新的工作流程创意。这也关乎到他职业生涯的清白声誉。

00:11:00 - Steve Rakitin: 长期以来，相比于电气工程和所有其他工程学科，软件工程并未被视为正规的工程学科。在我看来，部分原因是因为普遍缺乏软件工程师认可的公认实践。

00:11:30: 当我们经历了 90 年代的十年，逐渐开始明晰其中的一些流程。似乎其中一些已经在事实上被实施，而且也很合理。

然后敏捷宣言的出现了。如果软件工程将成为正规的工程学科，那么你就需要流程化的东西。其他所有工程学科都有流程，为什么软件工程就没有？

00:12:00 - Saron Yitbarek: 我是 Saron Yitbarek，你正在收听的是红帽的原创播客代码英雄。那么，如果我们把在核电站工作的人士的观点放在一边，转而关注更广阔的企业界，我们发现敏捷已经逐渐广受认可。但很自然的，也会有一些公司在抵制。

Darrell Rigby: 我倾向于认为我们在采用敏捷开发的过程中，受到的最大阻力来自中高级管理层。

00:12:30 - Saron Yitbarek: 这位是 Bain& Company 的合伙人 Darrell Rigby。他们一直尝试在软件开发公司中推行敏捷开发。不仅如此，还包括产品开发、新闻服务开发、广告计划和忠诚度计划等。不管他们要做什么，项目管理者都会面临点压力。

Darrell Rigby: 敏捷改变了他们的价值，因为他们正在逐步退出细节上的管理或干预。现在团队被赋予权力，对他们加以指导。

00:13:00 - Saron Yitbarek: 现在，敏捷并不能保证阻止中间轻微的干预。我承认，我第一次看到一个敏捷管理委员会时，我认为这是一个永无止境的待办事项清单，我有了点压迫感。但后来当我开始真正使用敏捷产品管理工具时，我完全变成了它们的粉丝。我是一个编码培训营的新人，我试图弄清楚如何确定功能的优先级并做出产品决策。

00:13:30: 那些看起来很可怕的工具让我有了所有这些想法，然后给它们命名、顺序和结构。从而可以帮助我更好地管理我的项目。所以，我确实同意 Rigby 的观点。有些人可能会看到这些工具产生的影响并认为，如果敏捷赋予开发人员权力，那么就会剥夺经理们的管理权。

但是，它的价值比任何一个职位都要大，敏捷开发的发展势如破竹。更重要的是，它正在证明自己。

00:14:00 - Darrell Rigby: 目前，成千上万的团队已经采用敏捷开发。因此，我们有大量数据能够说明在哪里可以采用敏捷的方式。答案是，无论何时你开始创新，相比你现在使用的方式，敏捷团队能更好实现目标。

00:14:30: 有许多更大的、知名的公司都在变革自身。亚马逊是敏捷方法的重要用户。奈飞、Facebook 和 Salesforce —— 他们都是敏捷的重度用户，实际上敏捷方法不仅重新定义了工作方式，更是重新定义了行业的运作方式。

agile

Saron Yitbarek: 当 Rigby 第一次听说敏捷时，他认为这是一种奇怪的语言。他当时正在与许多大型零售商的 IT 部门合作。无意间听到他们谈论 “time boxes”、“sprint” 和 “scrum master”。起初，他并不懂他们在说什么。他告诉我他实际上是试图忽略任何有关敏捷的字眼，就像这是他不需要学习的另一种语言。毕竟，他本人不是开发人员。

00:15:00: 但是如今，他却成为了敏捷信徒，把敏捷带到他的家里，带入他的教堂。

Darrell Rigby: 我不一定每天早上都和家人坐在一起，和他们一起参加敏捷开发式的会议。但是，我已经非常擅长为我要做的事情排优先级。

00:15:30 - Saron Yitbarek: 十多年来，敏捷已经从边缘走向主流。但是，企业认同还是有代价的。在某些情况下，这种同化甚至会使敏捷宣言的最初意图变得模糊。**Dave Thomas** 让我想起了这一点。他说，当他和其他 16 位雪鸟会议上的伙伴第一次写下宣言时，根本没有既定的指示。

00:16:00: 因此，即使宣言中没有告诉你如何应用这些条例，我猜想你已经对大概会发生什么，还有人们会怎么做，有一些大概的思路了吧？

Dave Thomas: 老实说啊，我还真没有。

Saron Yitbarek: 听到这里，你可能会感到惊讶。因为敏捷现在看起来很有说服力。有书籍、认证、工具、课程和产品的整个市场，向你展示如何“实现敏捷”。

Dave Thomas 表示，尽管有成千上万的手册和专业人士想要向你展示“唯一真理”，他们却错过了重点。

Dave Thomas: 实际上它是一组价值观。

00:16:30 - Saron Yitbarek: 嗯嗯（肯定的语气）。

Dave Thomas: 我想这就像黄金法则。你知道，如果你要做一些邪恶恶毒的事情，你会想，“好吧，如果有人这样做，我又怎么会喜欢。”你知道吗，这种场合也适合用黄金法则。

好吧，敏捷宣言也是如此。它并没有告诉你该做什么，不该做什么，它只是告诉你，你做的一切是否符合这个价值观。

00:17:00 - Saron Yitbarek: 是的。我想只要回到敏捷软件开发宣言的名称、真正脱颖而出并且经久不衰的一个词，也是人们真正关注的就是“敏捷”。那么现在使用“敏捷”这个词又出了什么问题呢？

00:17:30 - Dave Thomas: “敏捷”这个词的问题在于，在我们刚开始提出的时候，它是描述软件开发的形容词。但接下来人们就产生了疑问：“我该怎么着手实施敏捷呢？”

00:18:00: 突然之间，涌出了一大批咨询顾问，他们看到了 **Extreme Programming**

极 限 编 程 (XP) 的成功，看到了宣言的成功，“嘿，那里有座金山。”然后就开始告诉人们如何“做敏捷”。这是一个问题，因为你不能“做”敏捷。敏捷不是你要“做”的事情，而是你如何做事情的方式。

然而，有些公司会乐意卖给你敏捷相关的套装。我觉得这很讽刺。这里的咨询就好像是进入一家财富 1000 强企业，然后帮助他们设定“敏捷”。然后带走了 500 万美元。你懂吗？太棒了，钱真好赚。

00:18:30: 但是，现实情况是，这就像告诉老虎如何变得敏捷一样，说：“先走七步，然后左脚迈出来，然后再走两步，然后迈出右脚。”嗯，实际上只有瞪羚做同样的事情，这才会是有用的。你猜怎么着？没有人告诉瞪羚这样做。瞪羚基本都会跑到地平线尽头大笑起来，因为老虎在“邯郸学步”。

00:19:00: 当你告诉团队如何敏捷时，会发生同样的事情。如果你对他们说，“这是你必须遵循的规则，这是你必须遵循的流程”，然后他们唯一能做的就是跟随职责，因为他们已被设定好该执行的程序。管理层将根据他们服从原则或程序的程度，而不是开发软件的水平来判断表现如何。

00:19:30 - Saron Yitbarek: 所以，回顾一下，宣言发布之前和之后的开发者的角色，是如何因为你的宣言本身改变或扩展的呢？

00:20:00 - Dave Thomas: 我认为大多数程序员都能理解到关键点，这值得肯定。我觉得敏捷宣言给了许多开发人员按照这种方法去做的授权，某种程度上是他们以前就知道该如何，但从来没有权利这样做。像测试收集反馈，缩短迭代周期之类的事情。因此，在许多方面，工作变得更有趣，更充实。

同时我认为，程序员也可能会感到有点害怕，因为现在他们有了责任。过去，他们只是遵循命令。这个程序不起作用？好吧，但我遵循了规范。而如今，程序员肩负着责任。

00:20:30: 所以，我觉得这个职业因敏捷宣言而有所成长。我认为人们开始意识到，他们对自己所开发东西负有点对点的责任。

00:21:00 - Saron Yitbarek: 敏捷取得了如此广泛得成功，改变了工作流程和态度，远远超出了开发者世界的范畴——当然也超越了雪鸟会议召开的小木屋。我们不禁要问，“相比于 2001 年撰写宣言时，今天成为敏捷开发人员意味着什么？”

最初的敏捷精神是否仍然存在？如果确实发生了变化，这是一件坏事吗？对于谷歌的多元化业务合作伙伴 Ruha Devanesan 来说，敏捷的思维方式，可能已经发展到影响公平性和在工作场所中的平等性了。

00:21:30 - Ruha Devanesan: 让团队具有包容性的部分原因，是他们在进行非常基础的工作时，可以评价和反思自己。当大多数团队一起工作时，他们没有足够的时间这么做。没有足够的动力停下来思考他们团队宗旨，是否每个人都在能桌上发表意见，关于是否有人在推动其他人，或者是否有人在一直都保持沉默。如果他们保持沉默，为什么他们保持沉默？

00:22:00: 因此，在考虑包容性时，我认为敏捷团队使用的一些工具在为团队创建架构，或更具包容性的框架方面非常有用。所以多样性包括性别、种族，还有功能多样性。功能多样性为团队带来了复杂性。

00:22:30 - Saron Yitbarek: 但是，我们在这里要声明他们的不同。Ruha 并不是说敏捷就等于多样性。她的意思是，“敏捷加多样性等于更好的团队。”Ruha 的想法在她写的一篇名为《论通过敏捷方法解锁多样性》的文章中得到了体现。我们将在演示笔记中添加一个链接——这可是值得一读的文章。

在这篇文章中，她会引导你去了解，多元化不仅仅是人力资源部门一直在谈论的模糊概念。这里提供了一个强有力的商业案例。利用敏捷工具，可以创建一个包容性的工作场所，和创新效率提升。多样性可以与敏捷相辅相成。

00:23:00 - Ruha Devanesan: 这篇介绍复杂性的文章，最终目的是让大家从不同的角度看待你的结果或产品。当我们说为团队增加多样性可以带来更好的结果，带来更多的创新和创造力时，我们持有的是基本同样的观点。因为当你从多个角度去看待并协作解决工作中的问题时，你更有可能得出一个更好的结果。

00:23:30 - Saron Yitbarek: 团队中的每个人，甚至可以对日常会议这样简单的事情提出反馈，这会让内向的人或其他不爱说话的人发表自己的见解。

Ruha Devanesan: 我真正喜欢敏捷的原因是，有一些内置的机制来帮助团队停下来进行思考。这可能是因为敏捷开发是如此之快，并且有为时两周的冲刺任务。如果你没有建立这些机制，你可能会偏离轨道，没法再回到正轨。

00:24:00: 但是，我觉得，“停止并反馈”这种价值观非常重要。这对于团队的包容性增加也非常重要，因为让大家都能够提出工作反馈，并借此不断改善，这是团队能够包容的基本表现。

Saron Yitbarek: 既然我们谈论的是包容性，现在可能是指出那些敏捷宣言的 17 位创始人的好时机，是的……他们都是白人。

00:24:30 - Dave Thomas: 实际上那个房间没有多样性。这是对该组织的一种非常普遍的批评，而且我对此深表同情。

Saron Yitbarek: 如果敏捷宣言创始人采用了这些敏捷原则，并将其应用于他们自己的会议，那么他们可能在完成部分工作后，会问他们自己……“嘿，你注意到我们没有邀请任何女性参加这次会议吗？”我在想会不会有一个有色人种会持有不同意见。

00:25:00 - Ruha Devanesan: 物以类聚，人以群分。所以，如果考虑敏捷宣言的第一个人是白人，他邀请到桌上的人也是白人也就不足为奇了。但是，我们有机会在那方面做得更好，我们有机会停下来说：“让我们退后一步，让我们扩大我们的视野，寻找我们现在拥有的关系网络之外的人。谁可以带来不同的视角并帮助我们更好地改进这种开发方式。”

00:25:30 - Saron Yitbarek: 对我来说这很有道理，因为敏捷开发正是如此……好吧，敏捷，我们可以将它应用于不同的问题和行业。敏捷的应用方面，以及其在现实生活中出现时候的样子，是不断变化、不断扩展的。我想它正在将宣扬的内容付诸实践。没有最正确的答案，没有最后的终点。这是我们有时会忘记的事情：硬规则是敏捷的敌人。

00:26:00: 因此，如果一个敏捷团队告诉你敏捷的一部分意味着你必须每两周开发一个新的版本，或者你必须做什么事，那么，根据定义，这可不是敏捷。你老是说“总是”，你也不再是敏捷了。

00:26:30: 那些在犹他州雪鸟会议碰面的 17 名男子，最后宣称成立敏捷联盟。该联盟成为一个非营利组织，每年都举办一次会议。这个联盟的成长和发展，催生了更多全新的理论和方法。

这正是我感觉非常有趣的东西。在 2008 年的会议上，比利时开发人员 Patrick Debois 参加了并开始引领一条道路，他发明了一种全新的软件开发实践 DevOps。我从未想到与敏捷的一系列原则与 DevOps 和整个行业是都紧密相关。

00:27:00: 但是，现在我在想，“敏捷的兴起与 DevOps 的发明之间有多少关联？一个突破是否孕育了另一个突破？”我们会一起去探索，因为我们的下一集是正是 DevOps，对！一整集的内容。

00:27:30: 《代码英雄》是红帽的原创播客。有关我们的播客和其他更多信息，请访问 Redhat.com/commandlineheroes。在那里，你也可以关注我们的消息订阅。想要免费听取最新内容，请订阅我们的节目。也可以在 Apple Podcast、Spotify、Google Play、CastBox 中搜索 “Command Line Heroes”，然后点击“订阅”，你将在第一时间获得我们的内容更新。

我是 Saron Yitbarek，感谢收听，请坚持编程。

什么是 LCTT SIG 和 LCTT LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 [LCRH SIG](#) 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

关于重制版

本系列第一季的前三篇我们已经发布过，这次根据新的 SIG 规范重新修订发布。

via: <https://www.redhat.com/en/command-line-heroes/season-1/agile-revolution>

作者: [RedHat](#) 选题: [bestony](#) 译者: [redhat](#) 校对: [acyanbird,FineFan](#)

本文由 [LCRH](#) 原创编译, [Linux中国](#) 荣誉推出

《代码英雄》第一季（4）：DevOps，拆掉那堵墙

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频播客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《[代码英雄](#)》系列播客第一季（4）：DevOps，拆掉那堵墙的音频脚本。

当应用开发的争斗暂告一段落，横亘在开发者与运维之间的那堵墙开始崩塌。在墙彻底倒塌的时候，墙两边的人都应该学会如何合作，彼此变得亲密无间。

不过到底什么是 DevOps？开发者一方的嘉宾，包括来自微软的 Scott Hanselman 和 Cindy Sridharan（即 @copyconstruct），他们从开发者的角度认为 DevOps 是一种惯实践。而来自运维一方的成员则他们一直在努力捍卫的东西。双方依然存在差异，不过因为 DevOps 的诞生，大家的合作效率会比之前更上一层楼。这集节目讲述了这种方法的诞生对于大家有多重要。

Saron Yitbarek: 请你想象这样一堵墙：这堵墙从你目之所及的最右侧延伸到最左侧。墙比你高，你无法看见墙背后。你知道墙的另一侧有很多很多人，但你不清楚他们是否和你一样，不清楚他们是敌是友。

00:00:30 - Gordon Haff: 开发者创造代码，然后把代码扔过墙丢给运维，之后发生的问题都是运维的责任了。

Richard Henshall: 他们随心所欲，并不真正关心服务质量。

Sandra Henry-Stockier: 墙这两面的人几乎做着相反的工作——一方做出改变，另一方尽可能抵制那些改变。

Richard Henshall: 但对于他们到底想共同达成什么，却没有在同一幅蓝图中规划过。

00:01:00 - Saron Yitbarek: 我是 Saron Yitbarek，这里是《代码英雄》，由红帽公司推出的原创播客栏目。第四期，我们的标题是《DevOps，拆掉那堵墙》。

是的，数十年来，IT 界划分为各种角色。一边是开发者，他们要尽可能快地去创造更多变化。然后另一边是运维团队，他们要去阻止太多改变发生。与此同时，代码在没有充分共鸣和沟通的条件下，被盲目扔过两方之间的墙。怎样才能拆掉这堵墙呢？这需要一个重大的转变。

00:01:30 - Saron Yitbarek: 开源运动震撼了整个战场。[上一期](#)，我们看到了新的敏捷方法论，它重视不间断的迭代改进，而这种对速度的要求将迫使我们改变彼此的工作方式。一群彼此孤立的人工作的速度是有极限的，而这个极限是一个问题，因为……

00:02:00 - Richard Henshall: 因为都是为了更快的将产品推向市场、提高敏捷性、更多的迭代，而不是长期而大量的工作。

Saron Yitbarek: Richard Henshall 是 Ansible 的一位产品经理。

Richard Henshall: 是的。还记得你以前下单购买服务器，四个月后到货。所有东西都整合在一起，所以整个堆栈是一个整体，要花几年时间来设计和建造那些东西。现在这种情况已经不存在了，对于很多组织来说，这种方法已经……已经寿终正寝，偶尔拿过来试试，然后放弃它。

00:02:30 - Saron Yitbarek: 如今，像亚马逊这样的公司每分钟都会部署几次新的代码。想象一下，用按部就班的瀑布式工作流，简直不可能完成这些工作。所以为了继续快速完成工作，那些关于稳定性、安全性、可靠性的顾虑都会被运维丢到一边不管。

00:03:00 - Saron Yitbarek: 同时，开发者也没有意识到他们的责任是创造真实环境可用的代码。开发者对稳定性和安全性毫无兴趣，但这些恰恰是我们需要解决的问题。所以，我们最终会有很多无谓的修改，在双方之间来来回回。

想象一下过度分工会如何拖慢公司效率，但开发法者很少被鼓励思考除代码之外的其他事务。

Sandra Henry-Stocker: 他们的目录规模只会越来越臃肿，但他们从不清理。除非已经无法工作才不得不清理。

00:03:30 - Saron Yitbarek: Sandra Henry-Stocker 是位退休的系统管理员，为 IDG 杂志撰稿。

Sandra Henry-Stocker: 我过去经常劝说别人，说，“嘿，你看，你用了这么多的磁盘空间。是不是有什么东西你可以整理一下，这样我们就有更多的存储空间来运行了——因为服务器上的存储空间快用完了。”是的，我们经常经历这些事。

Saron Yitbarek: 归根结底，这是一个心态问题。这种开发者和运维之间的态度分裂，一方不必去理解另一方的担忧。好吧，在过去这还没太大问题，但随着开发速度成为一种重要的优势，这种分裂的文化急需改进。孤立在自己的工作圈子里，效率太低了。

Jonah Horowitz 在 Stripe 的可靠性工程团队工作。他描述了即使开发人员和运维人员想一起工作，他们也无法做到。因为从某种意义上说，他们被安排在对立的团队中。

00:04:30 - Jonah Horowitz: 运维团队经常以正常运行时间和可靠性来衡量，而提高正常运行时间的最大方法之一，就是减少系统的变化量。当然，发布新功能就是在改变系统，而做产品工作的软件工程师有动力去尽快发布尽可能多的功能。所以，当开发和运维的职责不同时，他们自然有了冲突。

00:05:00 - Saron Yitbarek: 开发者致力于新建功能，运营致力于维持运行，两者目标相互矛盾。但就像我说的，由于对速度的需求越来越大，对快速迭代发布的需求越来越大，开发和运维之间的脱节已经到了一个临界点，必须要有所改变。

00:05:30 - Saron Yitbarek: 在 2009 年左右，将开发和运维分开的那堵墙看起来像是监狱的墙。我们需要的是一种新的方法论，它能使开发和运维之间的隔阂顺畅过度，让双方以更快、更整体化的方式工作。

视频平台 **Small Town Heroes** 的首席技术官 **Patrick Debois** 为想要拆掉这堵墙的人发起了一场会议。他把这个的脑洞叫做 **DevOps Days**（开发运维日）。为了便利，他将其缩短为 **DevOps**，于是这场改革就有了名字。

00:06:00 - Saron Yitbarek: 不过名字并不代表改革的一步，我们知道为什么我们需要 **DevOps**，但究竟该如何做？我们应该如何将开发和运维结合起来而不引发战争？幸运的是，我有 **Scott Hanselman** 来指导我。**Scott** 是微软 .NET 和 ASP.NET 的首席项目经理。

所以，**Scott**，我认识你确实有几年了，但感觉还是相见恨晚啊。

00:06:30 - Scott Hanselman: 我也是，相见恨晚哈。

Saron Yitbarek: 我想和你聊聊你如何成为一个开发者，和 **DevOps** 这些年的变化。觉得如何？

Scott Hanselman: 嗯，听上去挺有意思。

00:07:00 - Saron Yitbarek: 好的。我认为究竟什么是 **DevOps** 是一个好的开场问题。你会怎么定义它呢？

Scott Hanselman: 在 2008 年，维基百科有个关于 **DevOps** 的定义确实很棒。它说，这是一套“惯例”，目的是在保证质量的前提下，缩短提交变更和变更投入生产之间的时间。所以，如果你想，假如今天是周二，我检查了一些代码，而这些代码将在 6 月的版本中上线。这就很糟糕了，因为这不是持续集成，而是一年几次的集成。

00:07:30 - Scott Hanselman: 如果你有一个健康的 **DevOps** 体系，如果你已经 set - up 有“设置 - 上线”的惯例，那么你就可以不断地将代码集成到生产中去。那么，你能做什么？你可以定义、创造怎样是最佳“惯例”，这将决定你能否成功。所以，我在周二检查的一些代码，周四就上线了。那么现在，为了保证高质量，最重要的事情就会是——谨慎上线。

00:08:00 - Saron Yitbarek: 这个定义真的很有趣呢，是个“惯例”。但我觉得当我听人们谈论 **DevOps** 时，它具体一点。他们谈论它就像它是一个角色、一个工作、一个职位或一个头衔。你觉得这与它是一套“惯例”的观点是否有冲突？

Scott Hanselman: 我认为，当一套新的方法或一个新的流行语出现时，人们喜欢把它加在名片上。我不是不尊重那些正在收听这个播客，并且感到被我冒犯、正骂骂咧咧把名片掏出来看的人们。虽然，他们现在可能正要愤怒地关上笔记本电脑、退出这个播客。

00:08:30 - Scott Hanselman: 有一个帖子写得非常好，作者是 Brian Guthrie，他是一个脑力劳动者，在 SoundCloud 工作。他是一个超级聪明的人，几天前他在 Twitter 上的帖子中说到 DevOps。他说 DevOps 就是一套惯例，不是一个工作头衔、不是一个软件工具、不是一个你安装的东西、也不是一个团队的名字。

00:09:00 - Scott Hanselman: 他的原话是：“DevOps 不是神奇的‘企业万能药’”。如果你没有好的惯例，如果你没有良好的习惯，你就没有 DevOps。所以，这更多的是一种心态，而不是摆出一个工作头衔，然后“我们要雇佣一个 DevOps 工程师，然后我们要把这些神奇的 DevOps 工程师撒到组织中。虽然整个组织没有意志力，也没有信奉 DevOps 的想法。”所以，如果你认为 DevOps 是一个工具或者是用来安装的东西，那么你就完全理解错了。

00:09:30 - Saron Yitbarek: 好吧，让我们回到过去，在 DevOps 这个名词出现之前，在我们往名片上写 DevOps 或者把它作为一套“惯例”来讨论之前。在 10 年前，你会如何描述开发者和那些运维人员之间的关系？

Scott Hanselman: 那是相当的水火不容。举个例子，运维控制着生产，但开发人员从来没有接近过生产。我们站在一堵不透明的墙的两侧。我们在开发部的时候，尽可能地去做一些看起来像生产环境能用的东西，但实际上从来没有……从来没有像样的产品。

00:10:00 - Scott Hanselman: 我们有相当多问题。我们的开发环境从各个方面来说都不像生产环境，所以你不可避免地会遇到那些“嘿，它在生产环境中的工作方式和在开发环境中的不同”的问题。然后，从开发到投入生产之间的间距是几周几周的长区间隔，所以你的大脑甚至不在正确的频道上。比如说，我在一月份的时候就在研究这个功能，现在四月份才刚刚上线，那么当 bug 不可避免地出现的时候，要等到六月份才能修复，我甚至不记得我们之前在干嘛。

所以运维团队的人，他们的工作是……他们的工作几乎就是有意识地让我们慢下来。好像他们的存在是为了让开发人员更慢，然后他们还觉得我们随时会让生产环境崩坏。

00:11:00 - Saron Yitbarek: 那么为什么会这样呢？是对开发者想要做什么和他们做了什么不了解？还是信任问题？为什么会有这么大的冲突？

Scott Hanselman: 我觉得你已经回答了，而且回答得很到位。你说的很对，确实是信任的问题。我觉得开发人员认为他们是特殊的，或者某些方面比 IT 人员更优越，而 IT 人员认为开发人员不尊重生产。

00:11:30 - Scott Hanselman: 我认为这种文化的产生，一部分来源于高层。他们认为我们是不同的组织，并且我们的目标也不同。我认为软件业正在走向成熟，因为我们都意识到，无论业务是什么，我们写软件都是为了推动业务发展。

所以现在有种“我们都在往正确的方向推进”的感觉，就像他们说的，“专注一件产品并做到极致”。但这是需要绝对的信任，可 DevOps 工程师不信任产品工程师来部署代码，对吧？

00:12:00 - Scott Hanselman: 但 DevOps 工程师传统上并不写代码，所以他们并不了解什么被修改了。所以他们对于在各个层面的人都缺乏信任。没有人理解部署过程，人们只信任自己，他们的心态……举个例子，就像“我只信任自己的工作。我不能相信 Saron 的工作，她甚至不知道她在干些什么。我会做完所有的事情。”

00:12:30 - Scott Hanselman: 所以如果没有人真正理解这个系统，那么 full stack engineer

全栈工程师的概念就是一个神话。但是现在，我们开始将一整个组织称之为
full product ownership
全栈。我们已经有了全产品所有权这样的名词，敏捷方法论也出现了，也就是说每个人都应该拥有产品。社区对于软件所有权和对于代码的想法都慢慢发生了变化，这种改变带来了一个充满信任的环境。

00:13:00 - Saron Yitbarek: 我是 Saron Yitbarek，你现在收听的是《代码英雄》，来自红帽公司的一档原创播客栏目。所以，要想让 DevOps 发挥出它的潜力，我们就需要双方都有更多的信任，这就意味着要有更多的沟通。回到 Richard Henshall 身上，他认为双方的共情是 DevOps 的基石。

00:13:30 - Richard Henshall: 一些 DevOps 的从业者，一群真正优秀的从业者，都参与过这两种角色。我认为这才是真正的力量所在——当人们真正做过了两种角色，而不是只看到其中一种。所以，你不该保持孤立，你实际上……你应该去和双方都一起工作一段时间。我想这才是让人恢复同理心的方法。

Saron Yitbarek: 现在，这不仅仅是为了温情的沟通。Richard Henshall 所描述的是行业重点的转向——Scott 刚刚提到过。

continuous integration

00:14:00 - Saron Yitbarek: 一个关于持续集成(CI)的观点。软件不仅要以小批量快速编写和发布，还要以小批量进行快速测试。这意味着，开发人员需要即时反馈他们正在编写的代码在现实世界中的表现。

随着上市时间从几个月缩短到几天，再到几个小时，我们四处寻找一套新的工具，可以将任何可以自动化的元素自动化。

00:14:30 - Gordon Haff: 你需要一个全新的生态系统和工具，来最有效地进行 DevOps。

Saron Yitbarek: Gordon Haff 是一位红帽公司高级主管。

Gordon Haff: 我们看到有很多巨大的、DevOps 可以利用的新种集合工具和平台，它们都诞生于开源。

Saron Yitbarek: Gordon 是对的。新的集合工具是很庞大，关于开源这点他说的也对。在一个严格的专有系统中，自动化工具是不可能发展的。

00:15:00 - Gordon Haff: 其中有很多监控工具，Prometheus 是其中一个常见的工具。它开始引起很多人兴趣，用于编排服务的 STO 也出自这里。

Saron Yitbarek: GitHub 让你跟踪变化，PagerDuty 管理数字业务，NFS 可以跨网络挂载文件系统，Jenkins 让你自动测试你的构建。

00:15:30 - Saron Yitbarek: 这么多工具，这么多自动化流程。最终的结果是，开发人员可以将他们的变更直接推送到生产现场，自动创建构造，实行经过严格管理的编译与针对性的自动测试。Sandra Henry-Stocker 描述了这是怎样的变化。

Sandra Henry-Stocker: 所以，我可以把我正在工作编写的东西快速部署。我可以只在一个系统上，通过命令行控制许多系统，而不是必须在在很多不同的系统上工作，也不用学习就可以利用网络，将代码部署到其他机器上。

00:16:00 - Sandra Henry-Stocker: 现在，在计算机系统中进行改动更容易了。坐在一个地方，就能实行一切操作。

Saron Yitbarek: 自动化工具已经解决了速度问题。但我不希望我们只赞美工具，而忽略了实际的方法论，文化的转变。Scott Hanselman 和我谈到了这个微妙的界限。

00:16:30 - Saron Yitbarek: 你在这次谈话开始时说，DevOps 是一套惯例，是一种心态，是一种思维方式。听起来，我们创造的工具是我们应该思考和操作方式的具体代码实现。

Scott Hanselman: 我喜欢这句话，你真是个天才。没错，我们以前让产品开发在 Word 文档中写下这些代码是如何工作的。他们写的是规范，对吧？这些文档过期了吗？

00:17:00 - Saron Yitbarek: 没错。（答非所问）

Scott Hanselman: 哈？

Saron Yitbarek: 好吧，我只是很高兴 Scott 刚才说我是天才。但我也确实认为，这些工具几乎是我们文化转变的象征。它们鼓励我们拓宽我们的角色定义。我们开发者已经被迫，至少偶尔关注代码的运行。这样一来，开发和运维的主要职责就部分统一了。事实上，DevOps 的兴起告诉我们的是，在一个速度不断提升的世界里，没有人能够保持孤岛状态。

00:17:30 - Saron Yitbarek: Jonah Horowitz 曾在湾区多家公司工作，包括 Quantcast 和 Netflix。他说即使是世界上一些最大的公司，也从这个角度重新塑造了他们的文化。

Jonah Horowitz: 我们在文化上得到了整个公司的认同，就像，“这就是我们要部署软件的方式，我们将以小批量的方式进行部署，我们将使用这些程序帮助部署。”如果 DevOps 只是被运营团队所驱动，我不认为它可以……我不认为它可以成功。

00:18:00 - Jonah Horowitz: 这必须成为公司的管理层和领导层所认同的东西才能发挥作用，而这件事很大程度上，意味着一种文化转变。

Saron Yitbarek: 当 MacKenzie 对 800 名 CIO 和 IT 高管进行调查时，80% 的人表示，他们正在规划让自己的一部分下属组织实施 DevOps，超过一半的人计划到 2020 年，在全公司范围内实施。高管们意识到，自动化工具可以提升交付速度。

00:18:30 - Saron Yitbarek: 这些人过去也是这样的人，他们习惯于让一个货板先到达数据中心，然后在新机器上线之前让它在那里放上整整一个月。不过在今天，如果你等待的时间超过 10 分钟，就说明你做错了什么。随着竞争对手的速度增加，没有人能够承受得起落后。

00:19:00 - Saron Yitbarek: 我可以想象，运维团队一定很紧张，因为他们把所有的工具都交给开发人员。运维团队习惯了做成年人，而现在叫他们把车钥匙交给一岁的孩子——开发人员？呀，我想我们开发人员正在学习，如何在不破坏东西的前提下快速移动。但随着 DevOps 革命的尘埃落定，变化最大的可能是运维团队。

00:19:30 - Saron Yitbarek: DevOps 是否真的威胁到了运维的存在？开发是否在用它闪亮的新工具来吃掉运维？Cindy Sridharan 是一位开发者，她写了一篇长篇调查文章来讨论这些问题。在你的文章和博客中，你提到运维人员对事情这样的发展并不一定满意。到底发生了什么？你会说什么？

Cindy Sridharan: 这么说吧，DevOps 的理想是责任共享。开发者和运维将有，就像你知道的，更多的是五五分成，以真正确保软件的整体交付。

00:20:00 - Cindy Sridharan: 我认为很多运维工程师的不快乐源于这样一个事实，那就是这些改变都没有实际功效。他们仍然是总被鸡蛋挑骨头的人，他们仍然是总做苦力工作的那些人，他们还是那些主要肩负着实际运行应用的责任的人，而开发者不一定要做得足够好。

Saron Yitbarek: 这个问题在未来几年将至关重要。DevOps 的作用到底有多大？随着我们的自动化进程，运维的作用是会被削弱，还是会发转变？

00:20:30 - Saron Yitbarek: 但是我们要记住，DevOps 不仅仅是工具和新技术的应用。这种方法论实际上是在塑造技术，反过来技术也在塑造方法论，这样就有了一个神奇的反馈循环。文化造就了工具，而工具又强化了文化。

00:22:00 - Saron Yitbarek: 最后，我们在节目开头描述的那堵墙，也就是把开发和运维划分开来的那堵墙，我甚至不知道五年后“把你的代码扔过墙”的比喻对一个开发者来说是否有意义，如果五年后大家都听不懂这个比喻，那真是一件大好事。不过目前为止的访谈很有价值，我听到了很多新的故事。

现在说话的是云架构师 Richard Henshall。

Richard Henshall: 我觉得 DevOps 开始让人们意识到对方关心的是什么，我看到了更多对彼此的理解。

00:23:00 - Saron Yitbarek: 现在是系统管理员 Jonah Horowitz。

00:23:00 - Jonah Horowitz: 我认为你需要很棒的技巧来写出真正好的软件，我在与我合作过最棒的开发者身上看到了一件事，那就是他们真的，他们贡献了关于的软件工程新技巧，或者说他们推动了软件开发这个行业的发展。

Saron Yitbarek: 最后一个是系统管理员 Sandra Henry-Stocker。

Sandra Henry-Stocker: 我认为，开发者会变得更加精明、更加谨慎。他们始终要提升自己的技能，我知道这需要很多辛苦的学习。

00:23:30 - Saron Yitbarek: DevOps 是个爱的结晶。原来，在那堵墙的另一边还有一些朋友，很高兴认识你们。所以，坦白一下，我以前总觉得 DevOps 很无聊，就是一堆硬核的自动化脚本和扩展问题。我的抵触情绪有一部分是出于它的实践难度。作为开发者，我每周都要面对一些新出来的工具，一些新的框架。DevOps 一直是那些可怕的、快速变化的一部分。但现在，尤其是听了这些故事之后，我明白了。

00:24:00 - Saron Yitbarek: DevOps 不仅仅是其工具。它是教导我们如何合作，更快地构建更好的产品的方法。

好消息是，随着为你我这样的开发者准备的新平台出现，我们的工作变得更好、更快、更能适应不同的环境，我的业务圈也可以不断扩大。你会看到人们将 DevOps 扩大到安全部分，所以我们能得到 Sec DevOps。或者他们开始包含商务，那我们就得到了 Business DevOps。我们现在要辩论的话题是：对于一个开发者来说，不仅要了解如何使用这些工具，还有了解所有 DevOps 的东西是如何工作的必要吗？以及我们需要所有开发者都去了解这个新世界吗？

00:24:30 - Saron Yitbarek: 这场辩论的结果将决定未来一期《代码英雄》的内容。

你可能已经注意到，在所有关于工具和自动化的谈话中，我漏掉了一些工具。好吧，我把这些留到下一期，当所有这些 DevOps 自动化达到光速时，我们将追踪容器的崛起。

00:25:00 - Saron Yitbarek: 是的，这些都会留到第五期。

《代码英雄》是红帽公司推出的原创播客栏目。想要了解更多关于本期节目和以往节目的信息，请访问 redhat.com/commandlineheroes。在那里，你还可以注册我们的新闻资讯。想免费获得新剧集的自动推送，请务必订阅该节目。

00:25:30 - Saron Yitbarek: 只要在苹果播客、Spotify、Google Play、CastBox 中搜索《代码英雄》，或者通过其他方式收听，并点击订阅，这样你就能在第一时间知道最新剧集。我是 Saron Yitbarek。感谢您的收听，编程不止。

什么是 LCTT SIG 和 LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 LCRH SIG 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-1/devops-tear-down-that-wall>

作者: Red Hat 选题: bestony 译者: LikChung 校对: acyanbird,FineFan

本文由 LCRH 原创编译, [Linux中国](#) 荣誉推出

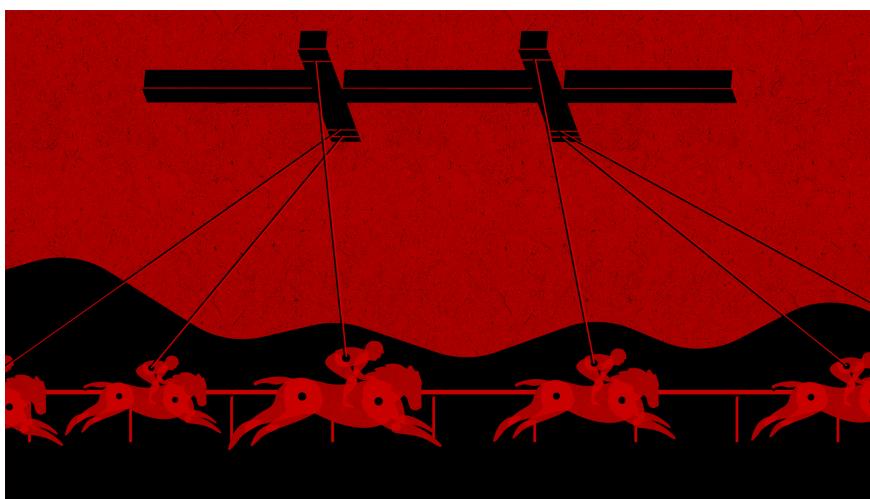
《代码英雄》第一季（5）：容器竞赛

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频博客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《[代码英雄](#)》系列播客第一季（5）：容器竞赛的音频脚本。

容器的兴起为开发者们打开了一道新的大门，它简化了在机器与机器之间传递项目的成本。随着它变得广受欢迎，一场大战也悄悄拉开帷幕。这场战斗的奖品是容器编排的控制权，参赛者包括这个行业最快最强的玩家。

容器是开源运动中最重要的一项突破之一。在这一集里，特邀嘉宾 Kelsey Hightower、Laura Frank 和 Clayton Coleman 将告诉我们容器如何为未来添砖加瓦，以及编排技术为何如此重要。

Saron Yitbarek:

你有看过赛马吗？赛马们排成一行，蹄子刨着脚下的土壤。你可以想象出这么一副画面。比赛即将开始，在这些竞争者中脱颖而出的将是优胜者。

00:00:30:

不同的是，比赛的不是马。而是科技世界的诸侯。那么是什么让比赛如此重要？是怎样的珍贵的奖励，才会让这些参赛者们排着队，迫不及待地想要得到它？这是一场赢家将掌握容器编排技术规则的竞赛，而且胜利者只有一个。重要的是，不同于其他的比赛，赢得这场比赛，你不仅仅会成为今天的冠军，更有可能在未来持续领先。

00:01:30:

我是 Saron Yitbarek，这里是代码英雄，一款红帽公司原创的博客。

第五集，容器竞赛。[上一集](#)我们见证了 DevOps 的崛起，以及一组新工具如何影响了其他人对开发者这一概念的看法。在这一集栏目中，我们会追溯容器技术崛起的历史，讲述容器技术如何通过拥有支持全新工作的可能性，来进一步扩展开发者这一角色的概念。然后我们会一起见证容器标准化是如何为容器编排奠定比赛基础的。

00:01:30:

这是一场严肃的比赛，也是一场全球性的比赛，吸引了行业里最快，最强大的选手。他们都准备好了为冲刺终点线而奋力一搏。准备好了吗？比赛开始了！

现在，随着这些“赛马”离开起点，也许我们应该弄清楚为什么这场比赛如此重要。谁会关心容器呢？好吧，算我一个。但是实际上，一开始我也并不知道容器是什么。以下我将讲述一个小故事——我是如何醒悟容器之美的。

00:02:00:

不久之前，我还在为我网站写代码，然后有天我让我的朋友 Nadia 过来实现一些新的功能。我在保持代码干爽和可读性方面做得很好，当然，代码也经过了很好的测试。所以再加入一个新的网站开发者也不是一件难事。对吗？如果你也这样以为，那就错了。这是一个非常繁琐的过程，特别是当我们跑规范化测试时，这个问题尤为明显。

00:02:30:

代码运行正常，但我们不能在两台电脑上同时通过所有测试。我们有很奇怪的电脑时区设置问题，而且她的 Ruby on Rails 版本跟我的不同。就是一个很经典的问题：“我的电脑上可以代码运行”，“可是在我的电脑上就是不行”。我只好对代码做一些修改，直到它在我这里正常运行，但当我把它发送给 Nadia 时，程序又会崩溃。

00:03:00:

我很清楚，我和 Nadia 所碰到这些问题，对于所有的开发者来说都或多或少经历过，甚至他们把这种经历当作玩笑来讲。有时候，我只能把这个当做是在我工作时必须要忍受的一部分。我没有意识到的是，这个问题有个最终解决办法。想象有一种方式可以降低人与人之间的隔阂；想象有一种方法可以让我们在开发中使用任意喜欢的工具，并且在传递工作成果时毫无阻碍；想象一下有一种办法，无论有多少人同时进行一个项目的开发，不管这些人散布在世界何地，都可以让项目从开发到测试，再到生产环境，保持连贯性。如果在我浪费好几周，用最笨的方式传递工作成果前就想到了容器该多好。

00:03:30 - Liz Rice:

一个容器实际上就是一个进程。

Saron Yitbarek:

Liz Rice 是 Aqua Security 的一名技术布道师。她描述了为何容器会如此实用。事实上容器把一切打包到了一个整洁、并且可以迁移的包中。

00:04:00 - Liz Rice:

这就像任何其他的进程一样，不同的是容器的世界非常小。比如，如果你启动一个容器，进程会被授予它自己的根目录。然后它认为自己在查看的是整台计算机的根目录，但实际上它只是在查看这个文件系统很小的一个子集。

00:04:30 - Saron Yitbarek:

通过打包一个可执行文件及其所有的依赖，容器可以在任何笔记本或者云中的虚拟机上运行。带着它自己的执行文件、库和依赖。所有的一切都包含在了容器中。所以，这就是容器神奇之处，容器在每个环境中的运行都会完全一样。这也意味着开发者可以轻松地分享并协作应用开发，而不用担心计算机之间相互不兼容这个老问题。

00:05:00:

Blue Apron

举一个类比的例子希望能够帮助你理解。你有听说过蓝围裙这个服务吗？该服务提供你做饭所需的一切，包括精心按照菜谱卡片搭配好的，所有做饭需要的原料。好的，想象一下如果蓝围裙所能带给你的不仅仅只是还没有处理过的食材，而是一个整个厨房，有煤气灶，还有你需要的全部餐具，一切你需要的都会装到小盒子里，放在门阶上。这就是一个容器。在我提到的那种情况下，容器技术就可以很好地解决 Nadia 加入进来时所碰到的问题，简单到像使用蓝围裙服务做一顿晚餐一样。虚拟机同样也可以提供一个预装好的环境。但要解释这个，我们就不得不抛弃蓝围裙这个比喻，让我们来看一看具体的细节。

00:05:30 - Liz Rice:

许多人都认为容器是某种轻量级的虚拟化技术、轻量级的虚拟机，事实上并不是。容器与虚拟机有很大不同。虚拟机有独属于自己的一整个操作系统，相比起来容器是共享操作系统的。一个计算机上的所有容器共享同一个操作系统的。

00:06:00 - Saron Yitbarek:

最后一点，容器和虚拟机可以并肩工作。容器不能替代虚拟机。虚拟化技术仍然可以提高过时系统的效率，并且对于服务器整合非常关键。但容器技术的兴起也为我们打开了新的大门。不妨这样想，如果我们全部依靠虚拟机的话，运行所有仿真服务器将产生大量的额外负担。

00:06:30:

一台虚拟机的大小至少是以 G 为单位的，然而一个容器可能也就只有 20 M 左右。一台虚拟机可能会需要若干分钟来启动，如果我尝试用它部署一个网页应用的话，这可不是一个好消息。很长时间以来，人们都期盼一个轻量级的、更快速的完整机器虚拟化替代方案出现。

00:07:07:

回顾一下历史，1979 年就出现了容器的原型。Unix V7 的开发者们设计了一种根系统调用，使环境中只包括特定的程序。该突破为我们现在看到的容器技术指明了道路。另一个巨大的进展来源于 2008 年的 Linux 容器技术。现在，我们有了操作系统级别的虚拟化技术。

00:07:30:

我们终于可以在一个单独的 Linux 内核上运行多个容器，而无需使用完整的虚拟机。这也就意味着程序对于基础架构的需求逐渐减少，但不是每一个人都能看到容器技术的潜力。

Laura Frank:

容器化真的是前所未有的、崭新的一个天才般的想法。

Saron Yitbarek:

Laura Frank 是 CloudBees 的技术总监。

00:08:00 - Laura Frank:

只有少部分人了解容器技术的来龙去脉，并可以运用它。不过相信随着时间的推移越来越多的人会接触到容器化的概念，随着越来越多的人开始使用这项技术，并且这些知识通过工程团队和工程组织，通过社区进行传播，就会变得更容易获得。

Saron Yitbarek:

因为和我们之前提到的与虚拟机的相似性，Laura 认为，因为我们之前提到的容器技术与虚拟机的相似性，容器的潜力被低估了。

00:08:30 - Laura Frank:

我在回想我的职业生涯，那是我还只是个普通的日常技术人员。如果你不是一个系统管理员或者 Linux 资深用户的话，容器还是一个你刚刚了解到的全新概念。我把它理解为使用一台虚拟机模式类似的东西，我可以去建立一个可以用完即弃的环境，而且这个环境完全独立，清理之后不留痕迹。

Saron Yitbarek:

容器除了能保持系统整洁之外，其实还大有可为。容器将会革新整个行业，并且随着开源项目和社区的兴起，在不久之后，容器标准化的充分实施将变为可能。

00:09:00 - Scott McCarty:

整个界面已经变得非常简单。

Saron Yitbarek:

Scott McCarty 是红帽的一名资深的容器策略顾问。他称得上是这个行业的资深人士，他在容器出现前，甚至是虚拟机出现前，就在做这方面的工作了。

00:09:30 - Scott McCarty:

在互联网 1.0 时代，我在一家线上零售商工作，我们有上千台实体机，我们用不同的方式，在所有这些不同的服务器上一遍又一遍地安装相同的软件。我们尝试了所有的方法。当你从原始的操作系统迁移到虚拟机，然后再到 Linux 容器、Solaris 容器，同样的问题一再出现，你仍然不得不在不同的虚拟机，或者类似操作系统实例的结构体之间管理配置。

Saron Yitbarek:

一旦容器变的规范化，一切都将改变。

00:10:00 - Scott McCarty:

比如，有了很多非常标准化的方式可以去处理现在这些打包好的应用，我认为容器技术的出现，从根本上改变了一切。它使得那些应用非常容易使用，而且容器还不会对系统本身造成损害，同时相比虚拟机更加小巧快捷。

00:10:30 - Saron Yitbarek:

借助 **Linux** 容器带来的进步，这些新的开源项目和社区使得开发者们可以更好地携手合作。很多我们对于后端的焦虑都被一扫而光。突然间，容器和由它促进的微服务变得十分有吸引力。一旦一种共同的容器语言出现了，障碍就消失了，与此同时容器技术改变了我们的工作方式，也改变了我们学习新技术的步伐。

00:11:00:

还记得之前我和同事 **Nadia** 遇到的反复出现的问题吗？“在我这代码能跑”的场景？在容器的世界，这个问题将不复存在。相比于我们之前使用的标准的操作系统，开发者社区见证了容器是如何变得更加快速，成本低廉，并且容易使用——比传统操作系统更加容易。容器技术被采纳的速度十分惊人。但是要记得：容器标准的出现仅仅是容器编排这场竞赛的热身。

赛马们已经整齐排列好，随着信号枪一声令下，它们为了这场比赛的冠军而拼尽全力。竞争的不是容器本身，而是我们部署和管理容器所使用的工具。

00:11:30:

我是 **Saron Yitbarek**，这里是代码英雄。在这场标准容器编排竞赛中，哪位会胜出成为管理所有容器的平台呢？起初有两位竞争者处于领先地位。

00:12:00:

由 **Apache** 驾驭的 **Swarm**，和 **Docker** 驾驭的 **Mesos**。但是等等，怎么？现在出现了一匹黑马改变了这个格局，那就是谷歌。**Linux** 设立了云原生计算基金会（CNCF），随后 CNCF 推动了谷歌开源的编排引擎 **Kubernetes**。

00:12:30:

现在，相比 **Kubernetes**，**Mesos** 和 **Swarm** 已经抢占了先机，对吗？它们得到了 **Apache** 和 **Docker** 的支持，已经入场了一段时间了。但是 **Kubernetes** 有其他的“赛马”所不具备的优势。**Clayton Coleman** 会告诉我们这个秘密是什么。**Clayton** 是红帽负责 **Kubernetes** 和 **OpenShift** 的一名架构师。

00:13:00 - Clayton Coleman:

在 **Kubernetes** 诞生之初，谷歌就在项目的开放上做的很好，它降低了项目的贡献和参与的难度。谷歌极其关注让开发者和运维人员能更加容易地开展工作。有这样一个强烈的关注点，就是要做一个能让大多数开发者和运维的生活更轻松的东西。我觉得 **Kubernetes** 和围绕着 **Kubernetes** 的社区找到了一个足够好的方式，让大部分人参与进来，他们让 **Kubernetes** 具有足够的可扩展性，还可以解决一些极端的用例。

Saron Yitbarek:

在早期，来自于红帽、**CoreOS** 和谷歌的工程师们都参与到了 **Kubernetes** 的开发中。随着 **Kubernetes** 开发到 1.0，不管是初创公司还是大公司都参与其中，一起构建和完善它。关键的是，所有这些增长从来都不是只归功于谷歌或者任何一方。

00:13:30 - Clayton Coleman:

在这个例子中，我喜欢以 **Linux** 打比方。**Linux** 并不是始于 **Linus** 开始编写内核，然后告诉所有人，在用户空间如何写 **GCC**，如何去建立 **NGINX** 或者 **Apache**。相反，内核团队专注于建立一个高效的操作系统内核，并与其他诸如 **GNU** 项目的开源社区合作，并且将可以在其他 **Unix** 系统上工作的工具引入 **Linux**。

00:14:00:

因此，我们如今所使用的许多工具，都不是 Linux 核心团队交付的。

但是 Linux 作为一个整体，相比于其内核涵盖的范围要宽泛得多，而且我认为这种模式的优势是 Kubernetes 取得现在成就所不可或缺的。当我们建立社区并且专注于 Kubernetes 范围时，我们可以试图从“Kubernetes 内核”的角度来考虑它，这是分布式集群操作系统的内核。

00:14:30 - Saron Yitbarek:

Kubernetes 证明了自己在开源世界中建立社区的能力，令人难以置信。正如我们在操作系统之战中谈到的 Linux 崛起一样，现如今这场关于容器的战争中，获胜者往往懂得如何借助社区力量。事实上，尽管谷歌可能开创了 Kubernetes，但目前它属于每一位开发者，并由云原生计算基金会（CNCF）管理。

00:15:00:

在 GitHub 上，Kubernetes 有大约 3 万的星标数，而 Swarm 和 Mesos 只有数千，这已经很能说明问题了。这就是由社区所生，为社区所用的技术。

我想了解谷歌的态度，一个如此庞大并且以效益为导向的大公司，是怎么做到如此擅长跟其他开发者合作的呢？我找到了很适合回答这个问题的人——Kelsey Hightower，他是谷歌负责容器技术支持的技术专家。

00:15:30:

想想谷歌的地位：它在分布式系统领域具备丰富的经验，还运行着分布在世界各地的许许多多的服务器，因此它开发的 Kubernetes 似乎有着很大的优势，并且有信心一定能在这场容器竞赛中胜出。那么，当你想到 Kubernetes 和开源时，你是如何看待这种关系的？

00:16:00 - Kelsey Hightower:

我想当谈到基础架构工具，甚至编程语言时，大家没有什么选择——你不可能用个专有工具，即使它很棒。如果它不是开源的，大多数人可能甚至都不会想去了了解。而且我认为这也是大多数人会采用像 Kubernetes 这样的基础架构工具的原因，你可能会对自己说：“好吧，我们就要坚持使用这个版本四、五年，也可能我们需要根据自己的一些独特需求来对其进行修改。”

00:16:30:

一旦走到这一步，就很难说服企业接受，“嘿，每台服务器使用程序的价格是 200 美元，而且你看不到源代码，所以有需要的话也必须等我们来修改”。

那样的日子一去不复返了，所以我不确定是否真的可以在没有开源的情况下建立基础架构。开源的另一个意味是拥有一个与项目紧密联合的社区，所以我认为 Kubernetes 一开始就锁定了胜利。

Saron Yitbarek:

让我们回到这场容器竞赛。在这里不仅仅有你提到的 Kubernetes，还有 Docker 的 Swarm Apache 的 Mesos.....

00:17:00 - Kelsey Hightower:

所以，我想当人们谈论容器竞赛时，我不确定竞争是否发生在我们和 **Mesos**、**Docker** 使用者之间。我认为，真正的竞争发生在争取目前没有使用容器的潜在用户身上。是的，你还在使用原生 **Bash** 脚本，你迷茫着，不知道自己该归属何方。这些尚未选择编排工具和平台之人的市场，比起已选择了 **Mesos** 或 **Swarm** 的一方，要多得多。

00:17:30:

这就是容器战争存在并将继续的原因，真正关键点在于如何帮助最终用户。**Mesos**、**Kubernetes** 或 **Docker Swarm** 是否会成为寻求更好解决方案的人们的首选？这一切都还悬而未决（SIG 译注：现在已经尘埃落定，**Kubernetes** 取得了全胜），但我会告诉你，像我一样，在这个领域工作的工程师来说，如果你不考虑市场营销和供应商，我会使用这个短语“不同的公司，相同的团队。”

00:18:00:

我们为彼此开发了许多工具，最终以某种方式出现在其他产品中。没错吧？好主意就是好主意。没有理由说，“哦，这是 **Mesos** 的人正在做的事情，那就忽略吧”，这有点愚蠢。所以从技术和社区的角度来看，我们的想法需要交流。同时也需要竞争来迫使我们来进行独立思考，然后最棒的点子就会浮出水面，接着我们再选择采用哪种方式来正确满足用户的需要。

00:18:30:

因此，就这场竞赛而言，仍处于初期阶段，而且这个事情本身不会带来利润。明白我的意思吗？我们不是直接向任何人销售这个产品，这更像是一个平台之间的游戏，对所有人开放，然后用户会选择满足他们需求的那个，这就是我认为 **Kubernetes** 在社区方面做得很好的地方，真正开放，真正能解决实际问题。

Saron Yitbarek:

听起来很棒啊。我喜欢这个想法：在同一个球队踢球，而不要管球队是在什么地方。你对于容器和编排工具，还有 **Kubernetes** 的未来有什么展望吗？

00:19:00 - Kelsey Hightower:

是的，我在 **KubeCon** 上做了一次主题演讲。所有这些工具都很棒，它们就像是乐高积木，我们有 **Kubernetes**，你可以选择一种产品用于安全，选择另一种产品用于网络，但最终，作为开发人员而言，你所想要的只是检查你的代码，并希望你的代码可以某种方式以呈现在客户面前。而我认为 **Kubernetes** 还有容器都会作为底层技术或者成为类似 **Serverless** 这种技术的基础平台。

00:19:30:

这是我的代码片段，已经打包完毕了。所有的平台都会把你的代码片段，用容器包装起来，然后帮你运行，但是不需要向你公开所有这些过程。因此，在未来，我认为随着 **Kubernetes** 变得普及，容器的应用场景将从大大小小的供应商或个人，提升到云供应商，因为这些事情往往需要专业知识和软件投资。容器将会遍布各个角落，但同时也就此隐藏起来。它会随着应用场景的扩展而渐渐隐形。

00:20:00 - Saron Yitbarek:

Kelsey Hightower 是 **Google** 的员工开发人员。在 2017 年秋天，**Docker** 宣布支持 **Kubernetes**。他们并不是说就放弃 **Swarm** 了，只是决定与容器编排竞赛的明显赢家和解。

00:20:30:

并不只有它一方，Azure 和 AWS 都宣布了对 Kubernetes 的支持。与此同时，像 OpenShift 这样的 Kubernetes 发行版仍在不断发展。我们得到的是一个可以扩展，支持新的用例的 Kubernetes 内核，这些用例包括微服务或持续集成项目。

00:21:00 - Clayton Coleman:

这个生态系统在类似 Linux 的模式下能得到最好的发展，而且我认为我们正朝着这条道路迈进。因此，就像所有优秀的开源项目一样，相对于单打独斗，让每个人都能够参与进来构建更好的东西，那就算是成功了。

00:21:30 - Saron Yitbarek:

所有这一切都在快速发生着，毕竟，这是一场竞赛，而这正是我们期望能从开源中获得的东西。在我们才刚刚理解什么是容器时，第一轮几乎就结束了，

这是来自 Red Hat 的 Scott McCarty。

Scott McCarty:

回想一下两年前，容器镜像格式还是一个巨大的战场，然后回到六个月至一年前，容器编排就成为了下一个巨大的战场。紧接着，如果你看看 2017 年的 KubeCon 及前几周，几乎每个主要供应商都宣布支持 Kubernetes。因此，很明显 Kubernetes 在这一方面上获胜了。

00:22:00 - Saron Yitbarek:

这章关于容器战争的故事即将结束。就像容器技术的开始一样迅速。

Scott McCarty:

因此，Kubernetes 已经成为标准，其美妙之处是，现在的应用定义已经变得标准化了。因此，任何人都可以在这些 YAML 文件中使用 Kubernetes 对象并定义应用，这就是我们共同所追求的事情。事实上，对于容器技术足够处理大型扩展系统这件事，我已经期待了 20 年。

00:22:30 - Saron Yitbarek:

Kubernetes 的成功看起来板上钉钉，但即使竞赛尘埃落定，我们仍然面临更大的一些问题。容器是否会成为未来几年的默认选择？是否会促使更多的云原生开发？这些转变将催生哪些工具和服务上？以下是我们目前所知道的。

00:23:00:

社区将通过 CNCF 继续改进 Kubernetes，并作为它最重要的使命之一，我们将建立一套全新的容器技术。

容器已经催生了大量新的基础设施，伴随而来的是全新的服务的需求。举个例子让你感受下容器的整合程度和发展速度，仅 Netflix 每周就运行超过一百万个容器。毫不夸张得说，容器是未来的构件。

00:23:30:

这一整季的栏目中，我们一直在追踪开源运动的演变。首先看到 Linux 如何主导战场，以及开源理念是如何改变商业、工作流程和每日使用的工具。容器真的是开源运动中最重要的里程碑之一。它们具有很好的迁移性、轻量、易于扩展。

00:24:00:

容器技术很好地体现了开源的优势，开源项目自然而然也推动了容器技术的发展。这是一个全新的世界，我们不用再担心从不同计算机或者云间的迁移产生的隔阂。

00:24:30:

容器的标准化比任何人预测的都要快。接下来的一集，我们将转向另一场悬而未决的战争。这场云间战争史无前例地催生者行业重量级人物。微软、阿里巴巴、谷歌和亚马逊四家云供应商的摩擦正在升温，随之而来的将是一场暴风骤雨。我们将会追随它们激发的闪电，和广受欢迎的几位代码英雄一起探讨云间战争。

00:25:00:

《代码英雄》是红帽公司推出的原创播客栏目。想要了解更多关于本期节目和以往节目的信息，请访问 redhat.com/commandlineheroes。在那里，你还可以注册我们的新闻资讯。想免费获得新剧集的自动推送，请务必订阅该节目。

只要在苹果播客、Spotify、Google Play、CastBox 中搜索 “Command Line Heroes”，或者通过其他方式收听，并点击订阅，这样你就能在第一时间知道最新剧集。我是 Saron Yitbarek。感谢您的收听，编程不止。

什么是 LCTT SIG 和 LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 LCRH SIG 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-1/the-containers-derby>

作者: RedHat 选题: bestony 译者: lujun9972 校对: acyanbird

本文由 LCRH 原创编译，Linux中国 荣誉推出

《代码英雄》第一季（6）：揭秘云计算

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频博客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《[代码英雄](#)》系列播客第一季（6）：揭秘云计算的音频脚本。

“没有什么云。这只是别人的电脑。”确切地说，还是服务器。大型云提供商提供了一种相对简单的方式来扩展工作负载。但真正的成本是什么？

在本期节目中，我们将讨论云中的战斗，说谁是赢家还很早。Major Hayden、微软的 Bridget Kromhout 等人会帮我们了解这场正在酝酿的风暴，以及它给开源开发者带来的影响。

Saron Yitbarek:

Ingrid Burrington 想要走进云的世界。不是真实的“一朵云”哟，而是“云计算”的世界。

Ingrid Burrington:

我不知道互联网真正的样子，我也不认为互联网是我想象中的那样，所以我想尝试找出它真实的模样。

00:00:30 - Saron Yitbarek:

Atlantic

Ingrid 是一名记者。在她为《大西洋》撰写的系列报道中，她讲述了自己参观一个数据中心的经历，一个我们网络生活越来越依赖的地方。她在那里并不是走马观花逛一圈，而是浸入式的复杂体验。首先，她要拍照登记，申请访客身份卡。然后，

要通过安检站，签署一份保密协议。最后，她才能去看机房。机房基本上就像个仓库，就像超市的那样，但比它大得多。

00:01:00:

整个机房看起来有种别样的美，所有的东西都整齐陈列着。一堆光鲜靓丽的服务器上，连接着通往世界各地的光缆，光缆沿着天花板上的轨道整齐布线。正在通讯的光电信号闪烁着点点神秘的蓝光，仿佛粒子加速器一样。但本质上，它是一排排如猛兽般动力强劲的服务器。

00:01:30:

数据中心每年消耗的能源比整个英国还要多。这就意味着它会释放惊人的巨大热量，这就是为什么当 Ingrid 环顾四周时……

Ingrid Burrington:

对的，我发现这座建筑主要的设计理念，是建造最理想最完美的暖通系统（HVAC）。

00:02:00 - Saron Yitbarek:

Ingrid 发现围绕数据中心的一切都强调经济实用，简单说就是一堆主机、一摞风扇、一大块租金便宜的地皮、用很多便宜的用来冷却的工业用水。完全没有“云”这个词本身散发的浪漫，但另一方面，我们的生活、我们的工作以及我们的话语，都在这个服务器的仓库里搏动着。

00:02:30 - Ingrid Burrington:

是的，这有点超现实主义。并不是说我就知道那台机器里存有某人的电子邮件，这台机器又存有别的东西，而是我意识到周围有很多看不见的事情正在发生，我能听到服务器的呼呼声和大量运算产生的微小噪声。说来奇怪，我开始对工业充满敬畏……

00:03:00 - Saron Yitbarek:

时刻要记住，在我们使用服务的时候，它们的基础，这些建筑都在某个隐蔽的角落嗡嗡运作着。以前，当我们谈论在云上存储东西，或创建应用程序时，我们有时会自欺欺人地认为它就像天上的云，是没有人能触碰的存在。但现实恰恰相反，一旦我们认识到云数据中心真实存在于某地，我们就会开始思考谁拥有着云了。那么是谁在控制这些服务器、线缆和风扇呢？

00:03:30 - Saron Yitbarek:

它们是如何改变开发者构建未来的方式的呢？云让我们紧密地连接在一起。

我是 Saron Yitbarek，这里是《代码英雄》，一档由红帽公司推出的原创播客栏目，第六集，揭秘云计算。

Chris Watterston:

没有所谓的“云”。那只是别人的电脑。

00:04:00 - Saron Yitbarek:

Chris Watterston 是一名设计师，他对围绕云产生的误解很是恼火。这个词模糊了数据中心的形象，就像 Ingrid 参观过的那个一样。当 Chris 把这句口号做成贴纸时，他因此成为了网红。“没有所谓的‘云’，那只是别人的电脑。”这句话现在出

现在 T 恤、帽衫、咖啡杯、海报、杯垫和许多主题演讲上。

00:04:30 - Chris Watterston:

人们完全不理解云是什么，还用的很欢乐又心安。他们可能完全误解了云，不明白他们的数据实际上是穿过铜轴电缆、或者光纤，来到某一个实际上由他人管理和拥有的存储设备。显然，对于一些有需要隐藏的私人内容的人来说，这是相当可怕的隐患。

00:05:00:

所以，下次你想把东西扔到云上的时候，想想 Chris 的贴纸吧。想想你到底要扔到哪里去。在 App 上工作也是同样道理，声称 App 跟服务器无关的说法都是骗人的，根本没有无服务器的 App。云就是别人的服务器、别人的电脑。不过云这件事情从某种意义上说，是一种成长。说到成长，在整一季节目里，我们会一直追溯开源的成长与变革。

00:05:30:

从最初的自由软件运动到 Linux 的诞生，直至今天，开源工具和方法把我们带到了远离家园的云端。可以打个比方，一个人找房东租房，他需要签合同、搬进去、把房子整理成自己的居所。当开发者寻找云供应商时，他们也在做着同样的事情。这就是我们现在所处的情况，全世界的开发者都在转向各种云上线产品，然后开始明白租赁的真实含义。

00:06:00:

严肃地发问一句，为什么我们一开始就急着跳上云端呢？

Brandon Butler:

因为开发者不想维护 App 运行所需的设备。

Saron Yitbarek:

Network World

这位是 Brandon Butler，《网络世界》的高级编辑，多年来致力于研究云计算。

00:06:30 - Brandon Butler:

开发者想要开发 App，部署 App，并只在乎 App 能不能正常运行。我们已经看到云孕育的，越来越多的服务，例如无服务器计算、功能即服务、容器和容器管理平台，如 Kubernetes。

Saron Yitbarek:

顺便打个广告，想了解容器和 Kubernetes，请看我们的[上期节目](#)。

Brandon Butler:

所有的这些成果都有助于抽象化 App 运行时所需要的底层基础设施。这将是一个可以在未来可预见的持续发展的趋势。

00:07:00 - Saron Yitbarek:

云拥有巨大吸引力的部分原因，可以用“超大规模”这个词来解释。通过云为你提供的所有基础设施，你可以根据自己的需求，快速创建和定制自己的服务器。你不再需要购买自己的设备，你只需要租赁你想要的规模的云。Brandon 解释了“超大规

模”对初创公司的具体帮助。

00:07:30 - Brandon Butler:

使用公有云进行 App 开发的整套模型，对开发者来说是一个巨大的进步。它曾经成就了一系列全新的初创公司，这些初创公司也已经成为大众都喜欢投资的公司。想想 Netflix 这样的公司，它的大部分后端都运行在亚马逊的以及其他云上。

00:08:00 - Brandon Butler:

这些公司现在如此壮大的原因，正是因为他们在使用云。因此，云对开发者的影响是不可轻视的。云已经成为过去十年，App 开发领域的主要转变。

Saron Yitbarek:

Nick Bash 是 Meadowbrook 保险公司的一位系统管理员，他还记得在云计算诞生之前，调整基础设施是多么痛苦的事。

00:08:30 - Nick Bush:

以前，有些人想出新项目的点子，我们会说，“这需要硬件支持，而我们没有。”他们会问，“那么我们该怎么办？”我们以前总是受到内存的限制，尤其是运行虚拟机软件，通常是最困难的部分。我们经常需要在任意时间启动虚拟机，但能随时启动的虚拟机数量总是不多。所以我们不得不花很多钱买新处理器、新内存、新硬件，或者花 5000 美元加新的东西。一旦我们从几个不同的供应商得到报价，就得报给管理层，他们要花时间审核。这样，仅仅是购买硬件都需要漫长的过程。

00:09:00:

更不要说构建虚拟机，再反复考虑和测试等等。所以其实我的意思是，有了云，我们可以在几个小时内完成以往需要几个月完成的前期工作。让虚拟机运行起来，第二天就交付给其他人。所以这是一个很大的转变。

00:09:30 - Saron Yitbarek:

在拓展性、速度和价格这些方面，云计算相当吸引人。还是拿租房作比喻，云就像可以让你免费得到一个管家和司机的服务，你很难对云计算说不。如今市场上有主要的四家壮志雄心的云供应商在开疆拓土。他们都想成为你在云上的“新房东”。但是且慢，每个租过房子的人都知道，租房和买房不一样。你不能自己拆掉一堵墙，或者安装一个新的按摩浴缸，你得通过房东来干这些事。

00:10:00:

那么 Brandon Butler 先生，我们使用私有云，在某种程度上会受制于一家独资公司。这会不会对我们不利？

00:10:30 - Brandon Butler:

当你使用云供应商的私有云时，你有不同的使用方法：你可以拥抱开源标准和开源平台，并且在云上运行开源软件，即便这是个私有云；你也可以使用不是开源的原生工具，这些工具可能在公有云上有更好的集成。因此，这是终端用户必须考虑的一个重大问题：我是想使用云供应商的原生工具，这些工具可能与这个云供应商提供的服务，以及其他服务更好的集成；还是拥抱开源标准，使用一个开源平台，享受更大的自由度，在自己和其他提供商的平台上也能运行？

00:11:00 - Saron Yitbarek:

随着我们所依赖的云技术不断发展，四大云供应商相互竞争，我们作为开发者有了新选择。我们是放弃一些独立性，依靠单一的云供应商来保护我们的工作，还是选择另一条路，在保持独立性的同时最大化云的拓展性？

00:11:30:

换句话说，我们能否在租房合同上写明，“房客有权任意处置该房，例如拆墙或其他装修”？

00:12:00:

那么，放弃一点点独立性又有什么问题呢？如果你是一名开发者，可能没受到什么影响。因为大多数时候都有运维团队在背后监督开发者们小心行事，他们格外留心于具体细节。这位是 Major Hayden，他是 Rackspace 的首席架构师。

00:12:30 - Major Hayden:

有些时候，开发者经常发现他们有各种不同的需要，比如某些专门化的存储，或者可能想要一定大小的虚拟机，或者想要一种云供应商未能提供的东西。还有一些东西可能开发者没有第一时间想要，但你认为他们需要的，对这些东西你还要进行成本效益分析。好吧，虽然使用公有云我们有很大的灵活性，但我们到底付出了什么代价？

Saron Yitbarek:

Major 指出了另一个问题，这个问题超越了实用性，并且触及了像我这样的开发人员所信奉的核心，那就是开源实践。即使云供应商允许你使用自己的开源工具，但云本身并不是开源的

00:13:00 - Major Hayden:

因此，开源对于云来说是一个需要处理的有趣议题，因为有大量的开源技术支持用户去高效地利用公有云，但并不是所有公有云都把它们的基础设施开源了。举个例子，如果你使用亚马逊，你无法知道他们使用的什么技术来构建虚拟机和其他服务。所以，你不可能对这些东西做点调整，或者很难了解幕后的机理和运作方法。

00:13:30 - Saron Yitbarek:

如果你听过我们之前关于 DevOps 的节目，你会知道打破开发者和运维之间的壁垒会让我们获益良多。架构师 Major 刚给了我们一些真知灼见，接下来的这位是系统管理员 Nick Bush，他所在的团队正准备向云端迁移。开发者们已经厌倦了每五年一次硬件换代，每个人都喜欢尽可能快地扩展，而 Nick 想指出一些开发者可能没有考虑到的东西。

00:14:00 - Nick Bush:

是的。我想说的是，云是存在延迟的。举个例子，就像远在蒙大拿的数据库服务器，对比我在街上用着 10-gig 的网络，本地数据库调用还是会花费更长的时间。要达到低延迟的云内数据库调用还有很长的路要走，还有其他的安全问题，因此我们暂时不需要担心物理上的前提。在本地，我们尚可以控制我们的硬件和其他类似的东西。一旦你进入了云端，你就得考虑连接问题。

00:14:30:

我认为，你也得稍微担心一下安全问题，虽然这更多也是一个成本问题。你想要按月租一个云端虚拟机，要求网速快并且带有充足的存储空间。每千兆的传输和存储都是要花钱的，以前我们都是一次性买断一个机器，我们只要买好了一个云端虚拟机，就可以存储和使用。只要余额和储存空间都还足够，我们就不用付更多钱。

00:15:00 - Saron Yitbarek:

声明一下，Nick 确实认为此事利大于弊。他只是不想让我们认为这是个完美的系统。如果你的云供应商宕机，而你想在其他云中重新部署应用程序，会发生什么情况？或者，如果在不同事务上使用不同的云能带来价格优势呢？运维人员提出的这些问题都可以被总结于一个词汇下，也就是供应商依赖。你可能很熟悉这个词。

00:15:30:

供应商锁定的意思是，在别人的服务器上构建业务会让你越来越依赖于他们的平台。你被绑定在这个平台了。可能突然之间，你被迫升级系统、付出更多成本、接受新限制，你变得身不由己。你懂的。

00:16:00:

当我们戴上 DevOps 的帽子时，我们开发者和运维就可以一起工作，面对供应商锁定，对症下药，但当我们沉浸在自己的代码中时，我们有时会忘记观览全局。为什么不找个折中方法，同时在公共和私有云上工作呢？终极解决方案可能是混合云，对于两方而言这都是最佳选择。我给 Bridget Kromhout 打了电话，询问她的看法。她是微软员工中的头号云开发提倡者，对这方面非常了解。

00:16:30:

如果我们考虑一种混合的解决方案，既包含一些公有云，也包含一些私有云，这是两者之间的完美平衡吗？对于开发者，这是理想的解决方案吗？如果云是混合的，那么我就能想做什么就做什么，想用什么工具就用什么，同时仍然可以从大型公有云提供商那里获得一些好处。

00:17:00 - Bridget Kromhout:

当然是的。举个例子，我有朋友在制造业中从事高性能计算研究工作，他们有各种各样的绝密资料，像 NDA 这样的东西，不适合放在公有云上。于是，他们可能会在自己的数据中心跟这些资料打交道，处理客户数据，或者研究数据，等等，也可能有其他的……

00:17:30:

他们也有适合放在公有云上的其他工作资料，不过我想这个问题就……有时也会有这样的问题，公有云是否适合某些工作资料，比如，如果你计划使用 InfiniBand 同步你的不同笔记，你能在公有云中做到什么程度呢？

Saron Yitbarek:

但这并不一定是完美的解决方案。Bridget 认为混合云也有自身的弊端。

00:18:00 - Bridget Kromhout:

混合云的问题在于，有时，人们欺骗自己，认为他们可以接受一些实际上不工作的东西，所以如果他们之前等待两周来获得一个虚拟机，如果有人经历过一个完整这样的情况，并且这个虚拟机还不能正常工作的话，就会有一堆的人由于失望而开始和他们的公有云提供商谈论信用卡问题了，然后他们会试着把这些东西粘合在一

起，但是还是有数据来源和延迟的问题，我不是很确定，脱同步的数据集有很多出错的方式。我认为，如果你和云服务提供商合作，你可以有一些可用的直接沟通这样你就可以更好地同步数据，这样是很有帮助的。

00:18:30 - Saron Yitbarek:

是的。当我们在开源的语境下谈到云的时候，我觉得，作为开发者，可能大多数人，都喜欢开源；如果你还在听我们的播客节目，就更是这样。对吧？你希望一切都是开放的，透明的，还向大众共享代码；但我觉得，当我们谈到云计算，因为它不会给人感觉是代码库，不会让人觉得云本身是个项目，它是环境，是可以用来帮助我们运行代码的东西，开发人员们还会坚持要让它像是传统的项目和代码库一样开源、透明吗？

Bridget Kromhout:

我觉得这是一个非常合理的问题，我觉得这可能也会归结到你到底要注目于技术栈的哪一部分。想一想，你对芯片的了解有多少？你又能在何种程度上操控它们？

Saron Yitbarek:

是的，这是真的。你说得不错。

Bridget Kromhout:

他们坐在那里，他们有硅，他们也有秘密。他们不一定会将后者给你。

00:19:30 - Saron Yitbarek

是啊，硅和秘密。顺便说一句，这是个好播客的名字。

Bridget Kromhout:

对吧？也许问题不在于是否一切都是开放的，而在于你需要开放的一切是否都是开放的，以及，当服务没有完全按照正确的方式运行时，你的服务提供者是否会对你保持信息透明，因为不该出的错误就是不该出。

00:20:00 - Saron Yitbarek:

所以，我得到了 Bridget 作为一个公有云提供商的观点，她提出了一个有趣的观点。开发者在云上的控制需要多细？至于我，我的看法不一样。我不想为了一点公有云的优势而牺牲的是什么呢？比如说，一个应用在公有云上运行，然后，等一下，现在我已经扩大了规模，或者有新的合规要求，我的应用在私有云上更合适。

00:20:30:

把应用从一个地方迁移到另一个地方之前，我需要知道它在迁移之后仍能工作。我需要知道它是以原先同样的方式打包，以同样的方式配置。换句话说，我需要知道从一个云跳到另一个云总是可能的。

除此之外，我们还有什么选择？仅仅锁定在一家云提供商？一个甚至可能完全垄断整个行业的供应商？不能选择迁移到另一个环境的话，这就像把一只手绑在背后写代码一样。

00:21:00:

所以，我们不想欠下任何一朵云的人情，并且被它困住。我们希望在合适的时候能
够在云间跳转。**Queen** 用摇滚传奇皇后乐队的名言来说，“我想要挣脱束缚”。我们希望能够获得公有云的绝佳拓展性，但又不放弃使用开源工具和方法所带来的自由。

00:21:30:

有个好消息。混合云的建设正在顺利进行中。**Mike Ferris**，红帽公司的业务架构副总裁，他给出了一个很好的解释，说明了混合云是如何帮助我们保持开源精神的。

00:22:00 - Mike Ferris:

开源是世界上几乎每一个云服务的基础，现在即便不是大多数，也有许多世界上应用程序的基础设施和工具是从这里发展出来的，管理能力，以及人们用于构建、部署应用程序（无论是任务关键型，还是非任务关键型应用程序）的工具都是基于开源的。

00:22:30:

混合云的概念和这一点非常兼容，这代表着，我们可以在混合云中处处使用开源工具，也可以最大程度地发挥出基础设施的优势。这是基于以下的一点事实：开源通过其在当今的强大影响力，能够在一定程度上定义下一代的开发模式。

Saron Yitbarek:

我认为云计算本身具有开放的意愿。在本季节目中，我们花了很多时间讨论开源的起源。你甚至可以证明，某些版本的混合云是这些相同理想的延伸。

00:23:00 - Mike Ferris:

在过去几十年里，开源开发活动的变化是越来越多的人参与进来了，包括像微软、IBM 这样的行业巨头。你知道，举个大公司的例子，他们要么使用开源软件来提供产品，要么构建开源软件并将其回馈给社区，或者两项都参与。

00:23:30:

这些来自客户的重要需求通过那些大公司涌入，确实帮助了开源世界的发展，使之从最初设想中 Solaris 和 UNIX 的替代方案，发展为不仅是社区和业余爱好者使用，而且肯定也是部分任务关键型企业使用的基础。

00:24:00 - Saron Yitbarek:

开源正在快速成长。现在，我们有机会确保我们记住我们从哪里来。当我们跃上云时，我们可以为自己声明开源的部分，以此来保持云的开放。幸运的是，由于有了 OpenStack® 平台这样的工具，在云之间构建开源桥梁变得更加容易了。

Rackspace 的首席架构师 Major Hayden 描述了它的起源。

00:24:30 - Major Hayden:

OpenStack® 来自于 Rackspace 和 NASA 的合作：“你看，这是一种构建基础设施的新方式，我们应该公开进行。我们应该得到更多的投入，应该和更多的人交流。我们应该得到更多的用例。”OpenStack® 是一组应用，它能很好地协同创建基础设施，并全面管理基础设施。无论你需要复杂的虚拟机、复杂的网络，还是有奇怪的存储要求，OpenStack® 通常可以满足大部分的要求。

Saron Yitbarek:

Major 指的是，加入一些开源知道如何提供的东西：也就是适应性。

00:25:00 - Major Hayden:

在我看来，OpenStack® 是一组相互连接的开放源码应用程序，它允许你构建你想要的基础设施。如果它不能建立你想要的，那么你可以进入社区，对它做出改变。我喜欢我去和顾客交谈时他们的反应，他们说，“我们想改变这个。我们想要改变这一切。”我们会说，“嗯，你可以。”

Saron Yitbarek:

我们如何确保这样的的适应性被包含在明天的云中呢？就像我们在之前的节目中谈到的许多问题一样，这需要强大的社区。有请 Brandon Butler，《网络世界》的高级编辑。

00:25:30 - Brandon Butler:

例如，我们已经看到了云原生计算基金会的成立，这个基金会制定标准，推广应用容器的使用，并创造了 Kubernetes。我们也看到了 OpenStack 基金会的成立，好将 OpenStack® 用户聚集在一起，讨论创建开源基础设施服务云时的最佳实践。

00:26:00:

支撑这些开源社区的社群对于开发下一波开源工具，学习如何最好地使用这些开源平台的，以及鼓励公有云厂商接受这些开源标准都非常重要。

Saron Yitbarek:

一旦我们开始构建混合云，并使其尽可能地开放，潜力似乎真的无穷无尽。
Major，请说。

00:26:30 - Major Hayden:

最让我兴奋的是看到更多的东西可以聚集在不同的云之上。例如，OpenStack® 提供了一个很好的基础设施基础层，但是你可以在它之上做很多事情。我想有时候不同的公司会采用 OpenStack®，然后说：“伙计，我现在该怎么办？我的自由程度太高了。我不知道该怎么办。”这就像你有一个装满食物的冰箱，你会想，“啊，我不知道该做什么菜。”

00:27:00 - Saron Yitbarek:

我喜欢这个问题。Chris Watterson 告诉我们的可能是对的。

Chris Watterson:

没有所谓的“云”，那只是别人的电脑。

00:27:30 - Saron Yitbarek:

但故事并未在此结束。我们要与混合云一起跨入下一章。创建混合云应用的关键可能还没有被破解。跨多云管理任务，对于今天的代码英雄们来说将是一项艰巨的任务。会有很多尝试和错误，但这是值得的，因为我们知道的唯一的一件事是，保持开源意味着开发人员总是可以构建他们想要工作的世界。这种灵活性正是紧紧抓住开源最擅长的叛逆精神的诀窍。

00:28:00:

下一集是我们本季的最后一集，我们将以一种让你惊讶的方式，从宏观角度来看开源作为一种全球现象是什么样的。我们也将展望开源的未来，我们的开发人员如何保持像 Linus Torvalds 这样的英雄的精神，即使当他们正在重塑他们的行业时。

00:28:30:

《代码英雄》是一档红帽公司推出的原创播客。想了解更多关于本期和往期节目的信息，请访问 RedHat.com/CommandLineHeroes。在那里你也可以注册我们的新闻通讯。想免费获得新一期节目推送，请务必订阅我们。只要在苹果播客、Spotify、Google Play、CastBox 和其他播客平台中搜索《代码英雄》，然后点击订阅，你就可以第一时间收听新一期。我是 Saron Yitbarek。感谢你的聆听，编程不止。

OpenStack® 和 OpenStack 标志是 OpenStack 基金会在美国和其他国家的注册商标/服务标志或商标/服务标志，并经 OpenStack 基金会许可使用。我们不是 OpenStack 基金会或 OpenStack 社区的附属机构，也没有得到 OpenStack 基金会或 OpenStack 社区的认可或赞助。

什么是 LCTT SIG 和 LCRH SIG

Special Interest Group
LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 LCRH SIG 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-1/crack-the-cloud-open>

作者: RedHat 选题: bestony 译者: LikChung 校对: acyanbird

本文由 LCRH 原创编译，Linux中国 荣誉推出

《代码英雄》第一季（7）：开启未来

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频博客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《[代码英雄](#)》系列播客第一季（7）：[开启未来的音频脚本](#)。

想象一下，在这个世界上，开源从来没有流行过，没有人认为将源代码提供给别人是个好主意。在本期节目中，我们将想象这种奇异的可能性。而我们也会庆祝那些让我们走到今天的开源工具和方法论。

加入我们，我们将对第一季进行总结，高屋建瓴地来了解开源世界是如何形成的。下一季，我们将把镜头放大，聚焦于当今的代码英雄们的史诗般的奋斗。

配音：

在一个没有开源的世界里，来自未来的执法者穿越时空去摧毁 Linus Torvalds 的计算机。

Saron Yitbarek:

天啊。我又做了那个噩梦。在梦里，我有一些很棒的想法，但我不能上手开发，因为没有相应的开源技术可以使用。

Tristram Oaten:

我认为一个没有开源的世界几乎注定是邪恶的。

00:00:30 - Saron Yitbarek:

我想，如果软件（LCTT 译注：这里的软件指 MINIX）在 20 世纪 80 年代遭到闭源，而源代码再也没有被打开过，肯定会少很多创新。

Steven Vaughan-Nichols:

那将是一个落后的世界。

Hannah Cushman:

我认为智能冰箱肯定会变得更少。

配音：

在一个没有智能冰箱的世界中。

00:01:00 - Saron Yitbarek:

好吧，好吧。你懂的。我们正在想象一个没有开放源代码技术的世界，这并不特别美好。想象一下：你的在线生活由一些大型私有公司管理，为此你得向它们缴费。网络中的每一处都被它们看守着。对于我们开发人员来说，没有开源的世界意味着更少的自由和影响力。

00:01:30:

在整整一季中，我们一直在纪录开发人员在开源世界中的角色。随着开源技术与工具的不断涌现，我们的工作也不断演进和扩展。无论是敏捷宣言，**DevOps** 的兴起，还是容器编排，我们宣称的力量和自由都与开源哲学紧密相关。

在本季的最后一集，我们将会回顾前几集中的内容。随着世界走向开源，这个词的原始含义能剩下多少呢？而我们，接下来，则将何去何从？

00:02:00:

我是 Saron Yitbarek，这里是《代码英雄》，一款红帽公司原创的播客节目。第 7 集：开启未来。

Steven Vaughan-Nichols:

没有开源的世界不是我想要的世界，也不是绝大多数人想在其中生活的世界。

00:02:30 - Saron Yitbarek:

这位是 Steven Vaughan-Nichols。你可能在[第一集](#)和[第二集](#)里谈论操作系统战争的 CBS Interactive 时候记住了他。他是哥伦比亚广播集团互动媒体公司的特约编辑，从快速调制解调器的速度还是 300 比特每秒时以来，他就一直关注着科技。

Steven Vaughan-Nichols:

除了 Linux 之外，你可能无法叫出任何一个开源程序的名字，但是你当前的生活是建立在开源之上的。

00:03:00 - Saron Yitbarek:

如果不使用开源技术，绝大多数人甚至无法上网。开源技术几乎存在于地球上的每台超级计算机中。它运行在物 联 网（IoT）中。它存在于你的手机、你的 Web 服务器，你的社交媒体——以及，大型强子对撞机中。而且，并非只有我们开发人员了解开源的诸多益处。开源态度现在已经超越了技术的范畴，影响了其它行业，例如经济学、音乐、科学和新闻业。

00:03:30:

如果建筑师以我们分享代码的同样方式分享建筑的蓝图会发什么？如果一个记者打开她的档案，让任何人不仅可以检查她发表的文章，还能检查她的研究和采访记录，会发生什么？我们不应为此而惊讶，因为开发人员培育这份哲学已有多年。每个人都可以看到代码、注释代码、复制代码、提供补丁，这实际上是一件非常基础的事情，对吧？这就是分享。

00:04:00:

自最早的人类分享膳食食谱以来，我们就知道公开分享指令集，或者说算法，对人类有净收益。在某些方面，开源技术现在能使我们重温这个基本事实。

Hannah Cushman:

我认为，使更多的事物开源会促进和鼓励人们查阅原始资源，这总是很好的。

00:04:30 - Saron Yitbarek:

这位是 Hannah Cushman，她是 DataMade 的一位开发人员，他们一直在努力使城市变得更加开放。将来自政府的大量公开数据进行整理并合理地处理，就可以让普通市民使用它来采取行动。他们使用的技术是开源的，同时他们对政治的态度也是如此。

00:05:00 - Hannah Cushman:

我们在芝加哥与一个叫做 City Bureau 的组织进行了一个项目，和他们一起为公立学校测试铅含量。我们测试了这些学校中几乎全部的供水设备。这些全部公布的测试结果有 500 份 PDF 文件之多。

Saron Yitbarek:

是的，这太好了。但这并不完全是一种使数据公开的有效方式。

00:05:30 - Hannah Cushman:

在整个系统中，很难看到例如哪里发现了铅，以及哪里的铅含量更高。我们使用了另一个叫做 Tablua 的开源工具，可以在终端上运行；它能从 500 多个 PDF 文件中提取数据并将其放在一起，帮助我们把巨量信息转储到一个对人们有用上下文中。

我认为查询源数据是一种非常有效的方式，这使人们可以了解信息的来源并验证其正确性。

00:06:00 - Saron Yitbarek:

市民可以访问健康报告的详细信息，获取游说者的数据，还可以查看城市政治的整个组织结构，DataMade 为此提供了浏览门户。这使芝加哥人有更多机会对市政的方方面面作出改变。

Cal Poly
加州州立理工大学的研究软件工程师 Carol Willing 认为，这种不断扩展的开源态度将给世界带来更广泛的变化。

00:06:30 - Carol Willing:

就个人而言，我认为开源将从开源软件发展到开放硬件、开放政府、开放教育、开放协作、开放创新等等。开源会不断进化。

Saron Yitbarek:

现在，开源逐渐变得更像是自然法则，而不仅仅是科技界的产物。

00:07:00 - Carol Willing:

人们慷慨奉献自己时间的事迹古而有之。但令人耳目一新的是，开源深刻地改变了世界，因为它使不同的小群体联合起来，一起致力于他们中任何一组都无法单独处理的大型项目。

00:07:30 - Saron Yitbarek:

“用全新的技术来践行古老的理念”——这个主意我喜欢。但先别急着高兴。随着开源这个词被越来越广泛地使用，它的定义可能会变得模糊。在某些场合下，它开始意味着“免费”，或者“众包”，甚至仅仅是“可定制”。

例如，如果我只是允许你选择在冰淇淋上哪撒些糖粉，这不代表我的冰淇淋是开源甜点。但是，如果我告诉你如何制作糖粉，让你改进我的糖粉配方——然后，如果你也想与他人分享糖粉制作的秘密，那么，恭喜你，我们得到了一些美味的开源。

00:08:00:

那么，开源的原始定义又是什么？它很简单，但我们应当恪守。要实现真正的开源，你需要公开代码，或者蓝图，或者你的菜谱。换句话说，就是使任何人都可以随意研究、修改和重新分发原始数据。这种哲学将带来变革，不过对于命令行外的世界，一切才刚刚开始。

00:08:30 - Thomas Cameron:

这是一种非常惊人的技术开发方式。无论是它的成功，还是我已经参与其中的这一事实，都让我感到震惊。

Saron Yitbarek:

Thomas Cameron 在 1998 年“开源”这个词被发明之前就一直从事开源工作。今天，他是 Red Hat 的高级首席云推广人员。他完全有资格谈论开源在如今的进展，以及其发展的过程中发生了多少斗争。

00:09:00 - Thomas Cameron:

你知道，经理们不想承担风险，这是巨大的阻力。因为它是免费的，所以他们会想，“我没办法打电话寻求技术支持”，“我不得不依靠特定的开源软件”之类。这一类斗争还算简单，在部门服务器、群组服务器、小型 Web 服务器、小型文件服务器和打印服务器上，我们也赢过不少。随着时间的推移，在赢得这些简单的战斗之后，更艰难的战斗出现了。在每次“作战”中，你会发现，系统管理员和系统工程师对开源越来越着迷。

00:09:30 - Saron Yitbarek:

尽管有这些斗争，你也无法否认一直以来的进展。

00:10:00 - Thomas Cameron:

我目睹了开源给 IT 行业带来的改变，它最开始时用在某些系统管理员办公桌下私搭乱建的服务器里，并最终传播到家喻户晓的大公司之中——英特尔、IBM、AMD，每个你能想到的组织都开始为开源项目做出贡献。这绝对是一场斗争，我在

不同的企业职位上都参加过如此多的相关争论；我曾说过，“我们需要把 Linux，或其他开源技术，引入数据中心。”

00:10:30 - Saron Yitbarek:

Thomas 观察到，开源软件开发正逐步占据市场。但对于某些人来说，这很令人不安。

Thomas Cameron:

我们能够分享信息与分析结果，这让那些一直占有信息并从中牟利的人感到害怕。这种模式使他们无法轻易获取利润，也难以哪怕得到对一个组织的完全控制，这是巨大的变化，随之而来的是恐惧。

00:11:00 - Saron Yitbarek:

我们在本季开始时描述的支持开源的反叛者们现在领导着这个行业。但从更长远的角度来看，故事绝不会在这里结束。Christopher Tozzi 是 Fixate IO 的高级编辑。他将开源带来的颠覆视为某种根本性转变的开始，这种转变将使世界各地的人们都能够协同合作——而不仅是在软件开发行业中。

00:11:30 - Christopher Tozzi:

我认为，在过去的二十年里，开源变得如此强大的原因之一就是人们对去中心化一直保有兴趣。我认为，这也解释了开源如何影响其它技术创新。比如，区块链也建立在这样的思想上：如果我们摆脱集中的生产方式，去中心化的数据库或交易方式可能会更高效、更安全。重申一次，我认为今天的开源，自从 Torvalds 出现以来，就与开发工作的去中心化息息相关。

00:12:00 - Saron Yitbarek:

全面的分权意味着整个世界都在走向开源。体现了这一理念的开发人员，他们是最能想象未来的人。

这是 Tristram Oaten，他是伦敦的一位开发者，他肯定在考虑这场漫长的比赛。

00:12:30 - Tristram Oaten:

就像是 3D 打印机将通过在家生产零件来使我们的生活更轻松——而且多半能更环保一样。无论什么时候有东西坏了，你都可以在家里新制造一个。这是星际迷航式的未来的复制器，就和理想中一模一样。希望这样的生产方式能真正投入使用，这样，说不定，整座房子都能变得开源了。

Saron Yitbarek:

Tristram 设想了一个世界，在这个世界上，开源是各个领域都遵守的规则。这意味着，开发者即使不是大师，至少也会成为人们的向导——这种向导是至关重要的。

00:13:00 - Tristram Oaten:

在未来，我们作为开发人员的角色将愈加重要，我们会变得越来越像“魔法师”——如果我们现在还不够像的话。

Saron Yitbarek:

好吧，魔法师。我们会成为魔法师。

00:13:30 - Tristram Oaten:

我们能用奇怪的语言驱使机器做奇妙的事，于是，我们会被高薪雇来做宫廷魔法师，或者公司魔法师。当每个人的身体中都有设备，并且无处不在的设备都可以通过互联网访问，还能被远程控制时，我们则需要作为一个团体，一个行会，以最好的信念行事，就像医疗行业要有不伤害他人的宪章等等。这非常重要。

我认为，开发人员需要共同决定不要制造杀手机器人，也不要在每个人的路由器和每个人的助听器中安置间谍软件。我们需要彼此确认，并向所有人保证，我们将为更大的利益而努力，而非为了伤害人类。

00:14:00 - Saron Yitbarek:

让我们现在保证不会造机器人杀手，对吧？好。在此之上，我确实认为 Tristram 说得对。在某些方面，我们开发者已经看到了未来，这代表我们有机会在塑造未来上出一份力。

10 年后，开源开发的道德标准将会是什么样子的？

00:14:30 - Tristram Oaten:

我们现在有着极大的特权，因此，我们有责任做正确的事。

Saron Yitbarek:

那么，魔法师们，我们将去向何方？我们能为开源创造一个健康的未来吗？我想和一个对这一切进行了深入思考的人谈谈，于是，我找到了这一位。Safia Abdalla 是 Interact Project 一名软件工程师，她一直在为交互设计做开源贡献。我们将会讨论，真正的“可持续的、广泛的开源”会是什么样子。听听看。

在你心里，未来的开源会是怎么样的，和现在有什么不同之处？

00:15:00 - Safia Abdalla:

嗯，我想，我所看到的最大的新兴趋势之一就是对开源可持续性的高度关注，也就是关于如何让开源项目一直得到良好维护和更新的讨论。有一些项目对整个技术界生态都至关重要，讨论也集中于它们。我认为，在该领域，已经有了许多有趣的进展。

00:15:30 - Saron Yitbarek:

Safia 让我思考，如果我们能够建立她所描述的可持续发展的方法，如果公司能够贡献时间、代码和资源，我们的工作能够得到多少改善？又会发生多少变化？所以我问她，这样可持续的方式会给我们所创造的产品，和我们建构的工具，带来怎么样的改变？

00:16:00 - Safia Abdalla:

可悲的现实是，当你没有精力，时间和金钱来为每个人打造好东西时，你就会倾向于只为自己做好它。

Saron Yitbarek:

嗯，确实。

Safia Abdalla:

在这种情况下，你构建的产品会将把很多人排除在用户之外。因此，我相信，如果我们发现开放源代码的更具可持续性的模型，那么我们实际上将开始构建可供盲人、听障者或以其它方式残障的个人使用的软件。

00:16:30 - Saron Yitbarek:

有趣，我喜欢。考虑到开放源代码的原则、流程、文化和社区，以及你提到的所有这些内容该如何适用于技术之外、软件开发之外的行业时，你认为真正可以从开源中受益的领域是哪些？你认为接下来开源可能会出现在哪里？

00:17:00 - Safia Abdalla:

喔，这是个有趣的角度。我第一个想到的是，开源思维可以被用于科学界，使科学变得更加开放。之所以我会想到这一点，是因为，当你以开放方式分享软件时，你所分享的并不是一行行的代码——好吧，你确实在分享代码——但，更重要的是，你在分享知识和细节，在与其他人交流该如何做事。因此，你真正分享的是知识。

00:17:30:

开源的方式可以相当直接地应用于科学界，因为研究人员也会花费大量的时间来探究特定课题，随后就此课题发表论文。而且，我认为，我们需要向科学界倡议更加开放的科研方式，以确保科研成果能够对所有人开放，使所有人都能理解、分享和参与，这将会提高社会对科学的理解，并在一定程度上促进科学研究本身。

00:18:00 - Saron Yitbarek:

当我上大学时，我从事生物化学研究；我非常习惯于带着热情进行实验、研究、尝试新事物。然而，同时，你也不能让人随意插手研究，因为你需要做通讯作者。你需要信誉。这是学术界进步的一个很重要、很重要的组成部分。

00:18:30:

因此，开源的原则——即分享原始数据、贡献劳动，以及将未完成的产品公之于众，依靠集体智慧将其完工的这些原则——和某些更具保护意识的行业必然发生冲突。你如何看待这种冲突？

00:19:00 - Safia Abdalla:

这是个好问题。我认为这涉及一个更大、更复杂的问题。为了成功地将开源引入其他行业，外在的动机和鼓励在很大程度上是必要的。你不能依赖于鼓励人们只专注于自己的目标的现有体系，因为那样会牺牲他人的利益和社会的更大利好。

00:19:30:

我认为，我们必须根本地改变我们看待很多事情的方式，并更改许多体系的运作方式，以使它们关注集体利益而非单一利益。很难撤销像终身教职这样对大学有很多负面影响的制度。很难消除其它可能损害生态、损害他人，或者阻碍社会进步的激励机制。不管是采用开源的思维方式，还是主动取消这些体系，我们都有极长的路要走。

00:20:00 - Saron Yitbarek:

确实。如果你可以从头重新创造开源，那么，属于你的开源版本又会是什么样的？

00:20:30 - Safia Abdalla:

哇噻。关于开源，我要改变的第一件事是它的公共关系和形象。我可能会尝试建立一种开源文化或社区，让人们认识到，你即使不是精英，或者精湛的开发人员，也能活跃并取得成功。现有开源文化中的精英倾向使我感到震慑。

00:21:00:

另外，我也要重点关注开源的可持续性，增加公司的责任感和开源体系的健康程度。我认为很多人没有意识到的事情之一是，许多受欢迎的技术公司和平台都包含了开源要素。例如，有很多 Rails Web 应用程序是极其成功的，盈利能力可观。而且，我认为，对我们来说，重要的是确保这些公司对开源社区有管理权，认识到其价值所在，并给予回馈。

00:21:30 - Saron Yitbarek:

好的，在 **Safia** 的开源中，企业将会负起责任，并对开源的可持续性做出贡献。项目的贡献者和维护者可能会因为自己所做的工作而获得报酬。这样的开源多半会是更加开放、更具有关怀之心的模式。

Safia Abdalla:

是的。

Saron Yitbarek:

听起来像是一个很棒的开源的版本，我喜欢它。

Safia Abdalla:

谢谢。

00:22:00 - Saron Yitbarek:

Interact Project

Safia Abdalla 是交 互 计 划的软件工程师和贡献者。她是一位新一代开发人员。而她在平时也对开源抱有自己的期望。所以，我想向这支新的代码英雄军队大声呼喊：“你们都将会向我们展示未来，因为你们现在正生活在未来之中，而你们也将负责领导我们走向明天。”

00:22:30:

现在，尽管我对开源革命心怀热情，但我也不想成为一个盲目乐观的人。开源也将受到挑战。开源的存在越广泛，我们就越需要确保它的可持续性。我们找到了一种可扩展的方式来维护开源项目吗？我的意思是，虽然 Linux 内核的贡献者中有一些是全职员工，但是大多数的开源项目仍然是由志愿者维护的。

00:23:00:

开源的工作并不会因为我们不再“反叛”而终结。市值数十亿美元的公司都在运行 Linux，而开源先锋们现在是技术领袖。我们需要跟随这条道路，并试着想象接下来会发生什么。

尤其是，可能会出现什么问题？**Christopher Tozzi** 告诉我们，曾经作为规则破坏者的开源并不对破坏本身免疫。

00:23:30 - Christopher Tozzi:

开源革命尚未结束，因为挑战并没有停止。即使今天基本上地球上每个使用计算机的人都在以这样或那样的方式使用开源软件，但这并不代表开源必然是绝对安全的。尤其是从致力于开源社区最初目标的人们的角度来看，诸如云计算之类的事物确实使情况变得复杂了。

00:24:00 - Saron Yitbarek:

开源能有多开源呢？Christopher 提到了云计算，而在[第六集中](#)，我们描述了依赖于别人的数据中心肯定会使开源最初的目标复杂化。

这是一个棘手的领域，我们仍在了解这个领域的情况。在前进的过程中，我们必须不忘初心。

00:24:30:

Obi Wan

每个年轻的反叛者都需要欧比旺·克诺比的原力全息仪。他们会从过去吸取经验教训吗？[Linus Torvalds](#) 训吗？林纳斯·托瓦兹曾经说过，“在真正的开源中，你有权掌握自己的命运。”如果开发人员能够在更大的世界中宣扬这种精神，那么，干得漂亮。

00:25:00:

到此为止，这是第一季的最后一集，真不敢相信。这一季过得很快。在着手编写这一系列播客之前，我从没有想过，是谁创造了 DevOps、敏捷开发和云计算；我从没有想过它们从何而来。我从没想过它们会有自己的家，有团队贡献才能、照顾它们，帮助它们成长。它们只是我工具箱里的一堆工具。但我现在不是这样看待它们的。

00:25:30:

它们不仅仅是随机的工具，而是我生活环境的一部分。在我之前，开发者们已经花了几十年的时间来塑造这一局面。现在，我要致力于塑造未来。真是棒极了。

00:26:00:

第一季即将结束，但好消息是，我们已经在准备第二季了。在这七集中，我们专注于开源工具和方法论，这些工具和方法使我们有了今天。这有点像是从 3 万英尺的高度看开源世界的形成史。在第二季中，我们将着眼于细节，并关注当今代码英雄们史诗般的奋斗。我们将跟随每一集，看看开发者如何挑战常规。这些都是塑造我们行业未来的真实故事。

00:26:30:

当我们寻找这些故事的时候，我们很希望收到您的来信。告诉我们，你的代码故事是什么？你参与过哪些史诗般的开源战役？访问 redhat.com/command-line-heroes 留下你的故事。我们正在倾听。

00:27:00:

如果现在你还在听的话，你可能想要看看将于 5 月 8 日至 10 日在旧金山举行的 2018 红帽峰会的阵容。峰会包括为期三天的分组会议、动手实验和主题演讲，其中包括了有关开源的主题演讲，而你也可以参与其中。希望能在那见到你。

00:27:30:

代码英雄是红帽的原创播客。请确保你订阅了该节目，以便在您的设备上免费获取第一季中的所有剧集，并在第二季开始时收到通知。只要在苹果播客、Spotify、Google Play、CastBox 和其他播客平台中搜索《代码英雄》，然后点击订阅，你就可以第一时间收听新一期。我是 Saron Yitbarek。感谢你的聆听，编程不止。

什么是 LCTT SIG 和 LCTR LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 LCRH SIG 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-1/days-of-future-open>

作者: RedHat 选题: bestony 译者: LaingKe 校对: Northurland

本文由 LCRH 原创编译, Linux中国 荣誉推出

《代码英雄》第二季（1）：按下启动键

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频播客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《[代码英雄](#)》系列播客第二季（1）：按下启动键的音频脚本。

导语：在“开源”和“互联网”这两个词被发明出来之前，就有了游戏玩家。他们创建了早期的开源社区，在彼此的工作基础上分享和创造。对于许多程序员来说，游戏引领他们走向了职业生涯。

在本期节目中，我们将探索在 ARPANET 平台上的，早期游戏开发天马行空的创意。游戏开发汇集了大量的创意并聚集了编程人才。虽然创建视频游戏在最开始是一个开放的过程，但如今很多事情已经发生了变化。听听你该如何参与打造我们自己的《命令行英雄》游戏，以及本着游戏的精神，找找本集的复活节彩蛋。

00:00:01 - Saron Yitbarek:

Dungeons and Dragons

一群朋友正在玩《D&D》（龙与地下城）游戏。他们靠在一起，听他们的地城城主（DM）说话。

00:00:09 - D20 菜鸟地下城城主：

好的，你在念咒语，你拿起你的权杖，把自然的力量注入其中。你可以看到藤蔓从里面伸出来，和你结合在一起。它在你手中的重量明显不同了，你感觉到了更强大的力量。

00:00:26:

所以我要施一个魔法……

00:00:27:

好的，你做到了，你还有一-次行动机会。你要做什么呢？

00:00:34 - Saron Yitbarek:

好吧，我得承认：我小的时候，从来没有坐在地下室里玩过《D&D》游戏，我也从来没有渴望过成为一名 DM，不管它是什么。我不会在真人角色扮演游戏中的树林里找我的第一个男朋友，也不会在游览动漫展的过道上和我的死党黏在一起。那不是我。

00:00:52:

但我知道是，游戏把人们聚集到了一起并形成了社区。而且，对于众多的开发者而言，游戏是编程的入门良方。正是游戏让他们了解计算机，并第一次将他们带入一个，以极客为骄傲的空间。

00:01:12:

正是对游戏的热爱让他们想自己开发一款游戏，然后去打造游戏，不停超越自身。这是我非常喜欢的东西。

00:01:23:

在我们的第一季中，我们探讨了开源从何而来，以及它如何影响了开发者世界的每个部分。这一季我们就在代码里面讲述：在今天，成为一名开发人员意味着什么。

00:01:39:

这一切都是从找到你的伙伴开始的。所以，开始游戏吧。

00:01:51:

open source Internet

甚至在“开源”和“互联网”这两个术语被创造之前，就已经有了游戏玩家。那些游戏玩家想要相互连接起来。因此，当世界开始联网时，他们走在了前列。他们要建立连接和共享所需的一切，并且——

00:02:09 - D20 菜鸟甲:

哦，它上楼了。哦，天哪，要疯。

00:02:11 - D20 菜鸟乙:

放心，这把武器可以打死它，伤害是 8 点呢。

00:02:16 - D20 菜鸟地下城城主:

随便你，来吧！

00:02:17 - D20 菜鸟乙:

所以，捅它！耶！

00:02:19 - D20 菜鸟丙:

捅谁，德鲁伊吗？

00:02:19 - D20 菜鸟甲:

我干掉它了！

00:02:27 - Saron Yitbarek:

好吧，接着玩吧。我是 Saron Yitbarek，这里是红帽原创播客
Command Line Heroes

代 码 英 雄 第二季。今天的节目，《按下启动键，游戏和开源社区》。

00:02:45 - 配音:

你站在一条路的尽头，前面是一座砖砌的小型建筑。你周围是一片森林。一条小溪从建筑物里流出，顺着沟壑而下。

00:02:56 - Saron Yitbarek:

你对这些话有印象吗？如果你还记得，可以给你的历史知识打 10 分。但是，如果

Colossal Cave Adventure

你和我一样，对它们没有印象，那么这些都是《巨洞探险》的开场白。

00:03:09:

什么是《巨洞探险》？我的朋友，一切的改变始于 1976 年。那条路，森林边的砖房，那条流入沟壑的小溪，当时没有人知道，但这款基于文本的游戏（是的，根本没有图形）将是一扇闪亮的红色大门，通向了新的社区和协作形式。

00:03:38 - Jon-Paul Dyson:

《巨洞探险》是一种被称为文本冒险的游戏。你可以通过输入命令与计算机交互：向西走、拿上剑、爬山等等。

00:03:51 - Saron Yitbarek:

Strong National Museum of Play

这是 Jon-Paul Dyson。他是斯特朗国家游乐博物馆的副馆长，也是其电子游戏历史中心的主任。是的，相当有趣的工作。

00:04:04 - Jon-Paul Dyson:

《巨洞探险》是一种非常不同类型的游戏。它更像是一款流程自由的探险游戏，就像在此时问世的《龙与地下城》一样。

00:04:17:

因此，它开启了想象力。这是一场真正革命性的游戏.....

00:04:22 - Saron Yitbarek:

在 70 年代中期出现了一种新型游戏，这绝非偶然。正在其时，互联网的鼻祖 —— ARPANET
阿 帕 网 出现了。

00:04:32 - Jon-Paul Dyson:

事情是这样的，一位在 ARPANET 上工作的人，他叫 Will Crowther，有了开发这个洞穴探险游戏的想法。他大致依据他曾探索过的，肯塔基州猛犸象洞穴的一部分为基础，创作了这款游戏。

00:04:50:

这真是革命性的突破，因为它给了玩家一个探索环境的机会。不过更有趣的是，他去度假了。而另一个人，叫 Don Woods 的伙计，因为当时 ARPANET 的存在而发现了这个游戏，然后做了些调整。

00:05:09:

因此，几乎从一开始，这个游戏的开发就是一个协作过程，因为它已在网络上被共享。这恰是一个很好的例子，说明了该游戏是如何被开发、修改、改进，然后广泛传播的，这一切都是因为这些计算机是连接在一起的。

00:05:28 - Saron Yitbarek:

因此，在计算机联网后，我们可以立即开始使用这些网络来分享游戏。而这些游戏本身也在不断演变。

00:05:38:

事情是这样的：不仅仅是网络改善了游戏，游戏也改善了网络。因为越多的人想要分享这些游戏，他们就越需要一个社区的论坛。

00:05:53:

因此，有了游戏技术，以及热爱它们的社区，它们彼此在相互促进。这是一个正反馈回路。同样，游戏开发人员彼此启发，相互借鉴。

00:06:09:

ARPANET 真是一片肥沃的土地。Jon-Paul Dyson 再次发言：

00:06:16 - Jon-Paul Dyson:

Adventure

所以像《冒 险》这样的文本冒险游戏就在这个空间里运转，这个空间被技术先锋们占据，他们知道如何无厘头、搞笑，知道如何才好玩。

00:06:28:

早期的游戏确实为开发者社区该如何协同工作提供了一种模式。

00:06:36 - Saron Yitbarek:

Minecraft League of Legends
记住，我们这里谈论的不是《我的世界》，也不是《英雄联盟》，我们谈论的是黑色屏幕上一行行绿色的文字，读完游戏内置的故事，并让你做出决定。这是一种相当简单的游戏文化，但它给了我们巨大的回报。

00:06:56 - Jon-Paul Dyson:

有一种共同体的信念，认为分享有好处，即与生产专有产品相比，更多的协作可以产生更好的结果。因此，结果就是你开发的游戏都是从社区中涌现出来的，这些游戏本身对改变持开放态度，并鼓励人们改变游戏，可能是小的方面，也可能是大的方面。但有一种感觉是，如果游戏变得更好，那么一切都是好的。

00:07:31

所以我认为历史上的这种早期的理念，尤其是计算机游戏方面，在推动计算机社区的发展方面确实很重要。

00:07:45 - Saron Yitbarek:

Dennis Jerz 教授特别研究了游戏的历史，尤其是《巨洞探险》。对他来说，这些原始开源社区对于所有创造力来说，都是一次自由的释放。

00:07:58 - Dennis Jerz:

当时的文化是，人们在自己工作的基础上建立和分享自己的想法。而且很常见的是找到源代码后，第一件事就是在现有的源代码上再增加一个属于自己的空间。

00:08:22:

这和同人小说的想法很像，就是人们创造自己的故事，这些故事穿插在《哈利·波特》的不同章节之间，或者是《饥饿游戏》中 Katniss 的世界里的小角色的故事。这种在主叙事的基础上进行补充、阐述和构建的文化。

00:08:44 - D20 菜鸟玩家甲:

好吧，所以我现在急着要找 Van Tyler，看看她的伤口。哦，上面说武器的伤害变成了 d8。

00:08:56 - Saron Yitbarek:

在基于图像或视频的游戏出现之前，这些基于想象力的游戏风格为大规模的在线社区铺平了道路。游戏社区和在线社区之间的是共生的，拥有强韧的联系。

00:09:11:

但如果有一件事是游戏玩家都知道，那就是强大的玩家可以将任务推动到一个新的方向。随着网络游戏的发展，它以社区为基础的根基开始被侵蚀。

00:09:24 - D20 菜鸟玩家甲:

这次试着把它放在他的脖子后面。搞定！

00:09:33 - Saron Yitbarek:

好的，让我们快进到如今。互联网已经成熟。有了它，在线游戏社区已经有了很大进步。如今，游戏每年的收入超过 1000 亿美元。预计在未来十年，这一数字将翻一番以上。

00:09:53 - Saron Yitbarek:

但这些取得的成功也改变了游戏规则。这不是一群独立开发者在一个摇摇欲坠的论坛上分享他们的作品。今天尚存的游戏社区与《巨洞探险》早期的情况完全不同。

00:10:11 - Saku Panditharatne:

大家好，我叫 Saku，我是一家名为 Asteroid 的初创公司的创始人，我公司为游戏开发者制作增强现实（AR）工具。

00:10:19 - Saron Yitbarek:

Saku Panditharatne 觉得开放的互联网与游戏开发之间的关系最近已经扭曲。她说的很清楚，为什么游戏公司之间不共享部分代码，而有时候大型科技公司之间却能共享代码呢？

00:10:37 - Saku Panditharatne:

如果你是唯一一个与所有人共享你的代码的游戏公司，那么你的竞争对手只会复制你的代码，他们会拿走你的所有秘密，并且他们会制作出比你更好的 3A 游戏。

00:10:47:

这对大型科技公司来说也是个问题。但实际上，我认为是软件工程师的努力打破了这种平衡。因为如果没有人为一家闭源的公司工作，那么每个人都会被迫开源，然后所有的大科技公司都能尽可能地共享，这对软件整体来说是非常好的。

00:11:11:

但这在游戏行业中从未发生过。

00:11:14 - Saron Yitbarek:

Saku 的理论是，与其他领域相比，在游戏领域的传统上，开发者没有同样的决策控制权。那太糟糕了，因为这意味着我们都错过了开放。

00:11:26 - Saku Panditharatne:

我们基本上知道具体如何将渲染、着色以及物理操作做到很高的标准。那不应该是每个游戏工作室都在复制的东西。但是，奇怪的是，游戏工作室通常仍会在内部构建引擎。不幸的是，他们陷入了这种.....

00:11:46:

我觉得游戏有点像电影，而不是像软件。你会使游戏开发陷入到地狱。你会遇到所有的问题，比如制作人的融资问题等等。

00:11:59:

我认为所有这些因素都不利于软件变得更好。

00:12:05 - Saron Yitbarek:

因此，游戏开发中的专有制度，导致了大量重复的工作，每个工作室都必须解决同样的问题。而且它们中的大多数都不是最佳解决方案。

00:12:18:

同时，对于可能会提供新东西的独立开发人员来说，事情很快变得难以承受。游戏开发人员通常必须购买某个主流游戏引擎的许可证。或者，他们必须得购买一个脚本。

00:12:33:

另一方面，Web 开发人员拥有更多可接受的选择。

00:12:37 - Saku Panditharatne:

我觉得真正有趣的一个事实是，拍一部电影比做一架飞机更复杂。这只是因为你拥有的所有这些各种人都具有不同的技能，他们的工作时间表不同，他们受到的激励也不同。

00:12:51:

所以，让他们一起工作就像一场组织挑战。而游戏引擎和其他游戏软件所做的事情之一：它是用来弥合这种鸿沟的，几乎成为最有效的工作软件。

00:13:06 - Saron Yitbarek:

游戏开发的问题很像电影制作问题。你有艺术家、编剧、角色设计师，都在与程序员和系统管理员角力。如果不采取开源的态度，或者没有建立共享和同步的方式，那一群不同角色的人就会变得混乱。

00:13:27:

这一切都归结为建设社区的挑战。那些最初的游戏，是由开发者利用自己的空闲，例如午休时间开发的。

00:13:38:

World of Warcraft

另一方面，像《魔兽世界》这样的游戏，他们有大量的创意人员和程序员。社区已经达到了极限。例如，我认为没有任何论坛可以让开发人员向角色设计师提出“PR”以供审查。

00:13:54:

但也许应该有 PR，Saku 知道我在说什么。

00:13:57 - Saku Panditharatne:

我认为游戏的一个大问题是，它是一个跨学科的东西。跨学科做任何事情都是困难
The Two Cultures
的，因为这几乎就像 C.P.Snow Essay 的文章《两种文化》，你必须把用右脑的、富有创造性思维的人和用左脑的、富有逻辑思维的人聚集在一起，让他们共同努力做一些事情。

00:14:18 - Saron Yitbarek:

我们能不能花点时间让全世界做点什么？如果 GitHub 上有人想找出一种开源游戏开发的方法，在开发者和艺术家之间进行转译，你就为世界做出了贡献。

00:14:32 - Saku Panditharatne:

我有点觉得，游戏的限制一直都是把两群不怎么说话的人聚集在一起，然后努力创造一些东西。

00:14:42 - Saron Yitbarek:

对我来说，简而言之就是开源的事。不过，特别值得一提的是，Saku 在想办法弥合两个社区之间的差距，从而形成一个更大的社区。

00:14:53 - Saku Panditharatne:

是的，我认为这实际上可以真正改善整个行业。我认为那些大型的新兴创业公司很多都出现在 2007 年左右，都是因为他们可以使用大量的开源软件。

00:15:07:

而且我认为，如果没有开发者把他们的东西免费放到网上，那股创业热潮就不会发生。

00:15:13:

如果类似的事情发生在游戏行业中，我想我们会有比现在更多的独立游戏开发者。

00:15:21 - Saron Yitbarek:

但有个好消息。游戏领域的开源革命，可能已经开始了。Jon-Paul Dyson 再次出现：

00:15:30 - Jon-Paul Dyson:

在视频游戏的历史上，确实存在两个流派：一个是人们创造专有产品的地方，即封闭系统。它们通常经过精心打磨，出于商业目的而发布。想想任天堂这样的公司。任天堂创造了令人赞叹的游戏，但你无法真正进行修改。

00:15:52:

但在电子游戏和电脑游戏的历史上也出现了一种相反的趋势，这是一种更具协作性的方法。像《我的世界》这样的游戏就是一个现代的例子，人们正在参与对游戏的修改，那里有一个围绕游戏而生的社区。

00:16:14:

你可以从互联网上的某个地方下载 MOD，然后在游戏中引入。因此，你拥有的是一种非常不同的，几乎是有机的开发游戏的方式，而不是一种更结构化的，可能是商业化的、工程化的方式。

00:16:35:

在很多方面，《我的世界》都是诸如《太空大战》或《巨洞探险》等早期游戏的继承者，该游戏在社区中发展，对进行修改和改变的人们更开放。而且，其治理理念是，从某种意义上说，以长远角度来看，群体的工作将会比一个小团队或个人的工作更好。

00:17:04 - Saron Yitbarek:

我认为我们现在要说的是，游戏行业并没有完成社区的建设，你也永远不能消灭这种开源精神。

00:17:14:

这就是我们所知道的：游戏仍然激发了许多开发人员，这就是许多代码英雄最初进入该行业的原因。比如像 Josh Berkus 这样的代码英雄，

00:17:28 - Josh Berkus:

嗯，我是从 Atari 800 开始的，因为它可以设计自己的游戏。

00:17:34 - Saron Yitbarek:

以及 Ann Barcomb,

00:17:36 - Ann Barcomb:

我写的是大多数冒险游戏，那是可怕的意大利面代码。

00:17:42 - Saron Yitbarek:

还有 Justin Flory。

00:17:43 - Justin Flory:

在我意识到我在做开源之前，我就已经开源了。我从一台运行着《我的世界》的游戏服务器开始的，那时我 15 岁。

00:17:54 - Saron Yitbarek:

当人们热爱某事时，那么对社区、对开源的态度就会蓬勃发展。Saku 对专有游戏元素的警告，实际上正是游戏在其旅程中必须克服的障碍。

00:18:09:

一些游戏行业的领先者正开始实现这一飞跃。

00:18:17 - 在玩堡垒之夜的孩子们:

我们实际上做得很好。

00:18:18:

我当时正在疯狂射击，我干掉了 3 个，他们干掉了 1 个。

00:18:21:

哦，Campbell 在背着我。

00:18:24 - Saron Yitbarek:

Fortnite

如果你去年就快到 12 岁了，你很可能听说过一个叫《堡垒之夜》的小游戏。嗯，很小的游戏。在它发布的第一年，它就获得了超过 1.25 亿的玩家，所有玩家都在一场大逃杀中。

00:18:43 - 在玩堡垒之夜的孩子们:

我得下线了，好吧，就一会，我得给我父母打个电话。

00:18:47 - Saron Yitbarek:

是的，游戏在建立社区方面还是很不错的。实际上，《堡垒之夜》是有史以来最大的免费主机游戏。它背后的工作室 Epic 还创建了《堡垒之夜》所使用的 Unreal engine 虚幻游戏引擎。

00:19:02:

很多其他的游戏也都使用虚幻引擎，因为 Epic 向世界免费发布了虚幻引擎。它不是直接开源的，但它的源代码是可访问的。这远超了大多数工作室。

00:19:16:

有了虚幻引擎，任何人都可以获取到源代码。他们不仅提供了代码，还致力于建立一个答案中心、论坛、开发者支持。**Epic** 正在加倍努力建设一个尽可能大而广泛的社区。

00:19:32:

而且他们并不孤单。你已经有了像 **Godot** 和 **MonoGame** 这样的完整的开源游戏引擎，它们正在催生自己的开发社区。红帽公司的软件工程师 **Jared Sprague** 是新一代的一员，他们可以使用所有这些工具和引擎来构建令人惊叹的游戏，如果没有这些社区，他们是永远不可能构建出来这种东西的。

00:19:55 - Jared Sprague:

独立游戏行业最近发展很快。因此，你应该发现几年前，大多数游戏都是由大型工作室制作的。这些都还在市场上，对吧，任天堂之类的。

00:20:09:

独立产业正在爆炸式增长。我认为这很大程度上是因为 **Steam**。

00:20:17 - Saron Yitbarek:

他说的就是 **Steam** 平台，它让开发者可以将游戏内聊天功能、社交网络服务等所有好的东西整合在一起。他们在这方面已经做了 15 年了。

00:20:29 - Jared Sprague:

Steam 使任何游戏开发者，都有可能发布一款游戏给数百万人。有很多大型游戏是从独立游戏开始的，也许只有一两个人在做，最后大放异彩。

00:20:48:

《我的世界》就是一个很好的例子。刚开始，只有一个人，是独立开发者。现在这是有史以来最大的游戏之一。但这只是一个例子，说明独立游戏是如何真正成为游戏行业的主要力量。

00:21:06 - Saron Yitbarek:

Jared 与他的红帽伙伴 **Michael Clayton** 合作制作了游戏 **Jams**，一群独立游戏开发人员聚集在一起，在很短的时间内开发出游戏。就像黑客马拉松一样，但用于游戏。

00:21:17:

GitHub 上有一个叫做 **Game Off** 的版本，**Jared** 的版本叫做 **Open Jam**。如果你问他，他会说这是基于社区的游戏开发未来的一部分，而且会变得越来越大。

00:21:31 - Jared Sprague:

例如，你可以使用 **Linux**，因为它是开源的，所有人都可以为它做贡献，添加工具、补丁等等，这就是它如此伟大的原因。比一个人独自完成的要好得多。

00:21:47:

而且我认为开源的游戏或某种游戏系统或虚拟世界仍有潜力，成千上万的人可以做出比一个人可以做的更大的东西。

00:22:03 - Saron Yitbarek:

游戏工作室已经开始转变了对开源态度。

00:22:10 - Jordan Weisman:

我认为游戏能够让你的社区开启创造，它的这种力量是毋庸置疑的。

00:22:18 - Saron Yitbarek:

Jordan Weisman 是 Harebrained Schemes 的联合创始人。《血色苍穹》、
Battle Tech Mech Warrior
《战斗机甲》、《机械战士》这些都是他的作品，他长期从事游戏开发。而
他现在看到的是，随着工作室开始意识到好处，对开源的态度也在转变。

00:22:35 - Jordan Weisman:

甚至像《战斗机甲》之类的游戏也有很多……我们已经尝试向玩家开放尽可能多的
游戏数据。而且我们已经看到玩家为游戏提供了惊人的 MOD。看到他们的创意和
成果，令我们兴奋，这与我们的原创内容相比，它以不同的方式吸引着玩家。

00:22:58:

这只是关于你能做什么的一个小例子。随着游戏的发展，我们更加知晓了这一点，
越来越开放，让玩家表达他们的创造力。

00:23:10 - Saron Yitbarek:

Jordan 认为，不断变化的文化有助于他的游戏成为最好的。但是，你知道，就像
我们在本期节目的《D&D》开场白中所暗示的那样，对社区的推动始终存在。

00:23:24 - Jordan Weisman:

TRPG 来自于 GM 与玩家之间，互相用语言编织一场冒险的 RPG 游戏，人们围坐
在一张桌子前合作讲故事并分享经验。在某段时间内，当我们开始分享某些电子游
戏时，这两派非常分裂。

00:23:41:

我认为，随着 MOD 工具和大家都能接触的引擎的诞生，以及工具更高的可访问
性，和开发商转向开源。这在某种程度上，让我们重新回到了游戏的创建之初，这
种非常社交化的。具有娱乐形式的协作。

00:23:58:

所以我觉得这很令人兴奋。

00:24:01 - Saron Yitbarek:

我也这样认为。至于 Jared Sprague，他将再次与 Michael Clayton 合作，开发一
款全新的社区驱动的开源游戏。它被称为……代 码 英 雄。

00:24:21:

是的，我们正式成为一个多平台 IP。当然，代 码 英 雄 游戏将成为一个
开源项目。所以我们邀请社区（也就是你）与我们一起打造这款游戏。我们所要做
的就是一个目标。

00:24:40:

灵感来自 NASA，这将是一个火星之旅。它的细节？这取决于我们所有人的共同努力。这将是一个基于网络的游戏，我们将在整整这一季的播客节里致力于这项工作。GitHub 仓库将对任何想要贡献的人开放。

00:24:59:

而且，再说一次，如果你正在收听此内容，则邀请你参与，与我们一起开发该游戏。设计一个角色，提一个拉取请求。或者只是玩测试版，然后将你的反馈发送给我们。

00:25:21:

我们将利用我们在本期节目中谈到的所有开源魔法，共同构建一些东西。

00:25:22:

趣事：还记得《巨洞探险》吗？好吧，虽然这个游戏的代码一直可以访问的，《巨洞探险》在 40 年后终于完全开源了。2017 年，官方版本在 BSD 许可证下发布了。

00:25:39:

从最初的互动式文字游戏开始，游戏社区已经走过了很长一段路。

00:25:45 - 配音：

你站在一条路的尽头.....

00:25:50 - Saron Yitbarek:

问题是，我们不是站在路的尽头。这条游戏之路绵延不绝。在此过程中，游戏正在帮助我们联合起来并建立整个宇宙。坦率地说，我们不知道我们的社区将如何安排自己沿着这条路走下去。

00:26:07:

但我们知道游戏将会推动这一进程，因为我们从来没有停止过——按启动键。

00:26:15:

想了解更多关于早期开源社区和开源游戏起源的知识吗？在 [代 码 英 雄](#) 网站上有更多的内容等着你。你可以通过访问 redhat.com/comandlineheroes 更深入地挖掘。在我们告别之前，我想让你知道我们在导语中有一项特殊礼物在等着你。这是 [Warren Robinett 的故事](#)，他创造了 Atari 游戏《冒 冒 险》，这款游戏的灵感来自于《巨洞探险》。

00:26:48:

他告诉我们，他与老板的争论是如何导致出现了第一个视频游戏的复活节彩蛋。这是一个很棒的故事，是我们发展史上很可爱的一部分，但我们想给它应有的空间。

00:27:03:

所以请到 redhat.com/commandlineheroes 网站上听一听。

00:27:10:

Command Line Heroes

代 码 英 雄 是 Red Hat 的原创播客。在 Apple Podcast、Google Podcast 或任何你收听播客的地方免费收听它。

00:27:20 - Saron Yitbarek:

我是 Saron Yitbarek，下次见，编程不止。

什么是 LCTT SIG 和 LCTR LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 LCRH SIG 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-2/press-start>

作者: Red Hat 选题: bestony 译者: gxlct008 校对: Bestony, wxy, acyanbird

本文由 LCRH 原创编译, Linux中国 荣誉推出

《代码英雄》第二季（2）：Hello, World

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频播客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《代码英雄》系列播客第二季（2）：Hello, World的[音频脚本](#)。

导语：每一种新的编程语言的诞生，都是为了做一些以前无法完成的事情。
如今，有非常多编程语言可以选择，但哪些才是你真正需要了解的？

本集将深入探讨编程语言的历史。我们将会了解到被称为“神奇葛丽丝”的天才——海军少将葛丽丝·哈伯。多亏了她，开发者不再需要数学博士的学历就能使用机器代码编写程序。参与录制的还有 Jupyter 项目的 Carol New York Times Magazine Willing，她是 Python 基金会的前理事；以及《纽约时报杂志》Wired 和《连线》的撰稿人 Clive Thompson，他最近正在撰写一本关于程序员如何思考的书。

00:00:07 - 各种语言：

“你好，世界。”

00:00:12 - Saron Yitbarek:**

你好，世界。现在我们发出的是信号还是噪音？我们日日夜夜无休止地写代码，做测试工作，我们追求的事情只有一个：我们是否传递了我们的信号？

00:00:29 - 各种语言：

你好。

00:00:29 - Saron Yitbarek:

或者我们的信号是不是丢失了？难道我们一直在制造无意义的杂音吗？

00:00:40:

我是 Saron Yitbarek，这是代码英雄第二季，来自红帽原创播客。今天的话题将在“翻译”中挖掘。这关于编程语言的故事：它们是怎么来的，如何选择其中一个语言来学习。我们将深入研究如何与机器对话。探究这些语言是如何演化的，以及我们如何用它们来让我们的工作变得更好。

00:01:08 - Sari:

Daisy, daisy，来告诉我你的正确回答。

00:01:11 - Saron Yitbarek:

如果你像我一样是一个开发者，你可能也经受着去精通多种语言的压力。最起码你得会 Java，更好一些得懂 Python、Go、JavaScript、Node，甚至能够用 C++ 思考。也许为了增加可信度，可能你还要熟悉古典的 COBOL。甚至你还得学点新东西，比如 Dart。实在太累了。

00:01:36:

奇怪的是，我们正在讨论的计算机只讲一种语言：机器语言，以二进制形式飘过的 1 和 0。说到底，我们学习的每个语言都是殊途同归。不管有多抽象，它都是用来简化我们工作的道具。

00:02:03:

这就是我希望你们记住的，故事开始了，从那一刻，那一个经典辉煌的时刻，一个女人说：“你知道么？我是人类，我不想用二进制来讨论思考，我就想用英语去编程。”

00:02:22:

在今天可能看来是一个简单的概念。但曾几何时，使用人类的方式与计算机对话这个主意——说轻点是一个笑话，严重点就是亵渎。

00:02:37:

代码英雄第二季，将讲述成就我们基础的螺丝钉展开。这一讲的英雄是一个女主人公，如果你想要进一步了解故事，你必须知道她的经历。在此给你们隆重介绍：软件界的第一夫人。

00:02:58 - 发言人:

Grace Mary Hopper

尊敬的先生们和女士们，我非常荣幸的给你们介绍葛丽丝·玛丽·哈伯准将，谢谢。

00:03:02 - 葛丽丝·哈伯:

我曾到加拿大的圭尔夫大学演讲，那时候我必须在多伦多的机场办理入境手续。我把护照递给移民官，他打量了护照和我说：“你是在哪里工作的？”

00:03:17:

然后我说，“美国海军。”

00:03:20:

她再次深深的打量我一下。然后他说：“你一定是海军里最老的。”

00:03:28 - Saron Yitbarek:

她就是 1985 年担任海军少将的葛丽丝·哈伯。在麻省理工学院发表了著名的演讲时，她 79 岁。即使在那时，她也是一个传奇人物。她是独立编程语言之母，实现通过编译器，用人类语言替代数学符号编程这一伟大成就的女性。

00:03:51:

她获得了国家技术奖章，她去世后（1992 年），被追授国家自由勋章。以上记录纯属真实，他们称她为“神奇葛丽丝”。

00:04:03 - Claire Evans:

如果有人能够被称为天生的程序员，那么这个人一定是葛丽丝。

00:04:06 - Saron Yitbarek:

Broad Band

这是 Claire Evans，一个科技记者和《宽带》一书的作者，而这本书讲述了科技领域的女性先锋。Evans 描绘了早期的计算机世界，在 1940 年，当葛丽丝·哈伯还是一个年轻的女孩时，她加入了美军的海军预备役部队。当时的电脑有一个小房子那么大。

00:04:25 - Claire Evans:

早期的那些程序员，他们就像一个圣职者。他们是一群擅长做枯燥乏味事情的人，因为那会儿没有编译器，没有编程语言，人们是确实是一个比特一个比特地对机器编程。

00:04:42:

想要改变这种现实，你真的必须是一个特别的人，而葛丽丝真的就是那种人。

00:04:49 - Saron Yitbarek:

现在，她意识到让人类使用机器语言的是一种多么疯狂的限制。就像走路的时候，去告诉你身体的每个原子该怎么做。这是可能的吗？是，效率呢？不高。

00:05:06:

在程序员的意图和计算机的行为之间一定有一条捷径。自从有了算盘，人类一直在将数学思维机械化。一定有一种方法可以在计算机上再次实现这一点。

00:05:19 - Claire Evans:

在过去，数学家必须做出所有的步骤。所有这些枯燥的，一步一步的工作都是为了得到一个真实而有趣的答案。然后有了计算机后，它尽可能地承担了这些机械的运算。所以解放了数学家去进行高层面的、更智慧的、面向系统的思考。

00:05:39:

只不过，事实并没有如此发展。有了计算机，数学家就变成了程序员。他们又一次被这些枯燥无味的编程所束缚，做着这些一点一滴的繁琐的规则导向的小计算。

00:05:53:

因此，我认为，葛丽丝的期望是，她希望计算机能够辅助数学家，然后数学家就可以去更进行宏大的思考，做更伟大的事情，而不至于让他们陷入细节之中。

00:06:12 - Saron Yitbarek:

Pascal

葛丽丝也加入了思想家的行列。我们都知道，帕斯卡在 17 世纪发明了第一台计算 Babbage 器。巴比奇在 19 世纪发明了分析机。这些伟大的杰作是发明家们解放人类的大脑的创作。这些发明让我们处理数据的能力空前绝后。这就是葛丽丝踏入的领域，发明了所有人都认为不可能的东西。

00:06:42 - Claire Evans:

我觉得，葛丽丝的想法是，人们应该用人类语言与计算机沟通。这个语言必须独立于机器的，因此不同计算机可以交互操作，这在当时是革命性的。

00:06:59:

早期她称这个为“自动化编程”，在程序员和计算机人的群体中，被认为是像太空学员一样。

00:07:12:

而那些迟迟不上岸的人们被看成了尼安德特人，这在当时是计算机界的一个巨大的分歧。

00:07:21 - Saron Yitbarek:

葛丽丝要说服她的上级跨越这个分歧不容易。但她认为这是她的责任，她必须尝试。

00:07:30 - 葛丽丝·哈伯:

这里总是有一条线，那条线代表了你的老板在那个时候会相信什么。当然，如果你过线了是得不到预算的。所以你对那条线有双重责任：不要踩过它，但也要不断教育你的老板，让你把这条线移得更远，这样下次你就能走得更远。

00:07:52 - Saron Yitbarek:

她推动的那个未来，就是我们的现在。几乎你我依赖的每一种语言都归功于她和她的第一个编译器。因此太空学员们活了下来，尼安德特人灭亡了。

00:08:07:

写代码不用数字，而是用人类风格的语言是她的观点。你敲下“执行操作 8”，或“关闭文件 C”的精简操作，才是符合人类习惯的编程。

00:08:21 - Claire Evans:

葛丽丝独特地意识到，计算将成为改变世界的事情。但是如果没有人知道如何接触并使用计算机，那么它就不会成为改变世界的事情。因此她想确保它对尽可能多的人开放，尽可能多的人可以使用。

00:08:37 - Claire Evans:

这就需要编程在可理解性和可读性上做一些妥协。因此最终创造一个编程语言的目标就是给机器提供更多切入点，让它脱离这个神职，让它的开发面向大众和未来新一代。

00:08:59 - Saron Yitbarek:

我想在这里打断并强调下 Claire 的说法：现在我们所已知的编程语言，都来源于科技开放的愿望。这让计算机不再是数学博士们的专属玩具，让编程开发更容易。

00:09:14:

编译器所有工作的本质，是让程序变得更好理解，更有人性。

00:09:21:

Claire 有一个猜测，为什么葛丽丝是做出改变的那个人，这与她在二战期间的工作有关。

00:09:29 - Claire Evans:

她的工作是解决扫雷问题、弹道问题和海洋学问题。她用很多不同的、交叉的学科来模拟战争里的所有的暴力、混乱和现实的灾难，并将他们转化成在 Mark I 计算机上运行的程序。

00:09:45:

她知道如何在语言之间去做翻译转换。我的意思不是说计算机语言，是人类语言。她知道如何倾听一个提出复杂问题的人的意见，并尝试理解问题的背景，其信息和所涉及的专业规律，然后将这些转为计算机可以理解的程序。

00:10:06:

从这个角度看她如同早期的编译器。就像一个人类版本的编译器，因为她知道你必须理解他们才能满足他们的需求。

00:10:17 - Saron Yitbarek:

编译器干的事情就是一个解释和理解。我觉得我们应该在学习新语言，或想知道为什么有一些东西根本不能编译的时候牢记这个理念。编译器的工作就是满足你使用生活中说的语言来编程。

00:10:32:

葛丽丝知道人类一旦可以学会讲编程语言，而编译器可以将我们的意图转换为机器语言，就像为洪水打开了大门一样。

00:10:43 - Claire Evans:

她知道如果计算机太孤立和太具体，就不会发展为一个产业，从而成为改变世界的力量。也就是说计算机的从业者，可以让提出问题的人跟机器直接沟通。

00:11:00:

因此她真的是一个善于人类语言翻译的专家，至少我是这么想的。这使她有独一无二的机会，去思考和创建编程语言。

00:11:15 - Saron Yitbarek:

葛丽丝对英文式数据处理语言的研究最终演变成了 COBOL，它在某种意味上很不错。因为它不浮华，很适合商务用途，葛丽丝·哈伯也是这样的人。

00:11:28:

从某种角度看，她给了我们一个很像她的语言。像葛丽丝一样，COBOL 语言也很长寿，它现在已经 60 多了。

00:11:50:

babelfish

葛丽丝的编译器就像一个巴别鱼，在程序员和机器之间充当翻译，不过它们翻译的都是高度概括性的语言。

00:12:03:

然后，几十年之后，另一个重要的思潮在语言界兴起。想象下画面：自由软件社区在 1980 年出现，但是 Unix 的替代品 GNU 被开发出来后，却没有任何自由开放的编译器随之出现。

00:12:22:

为了让 GNU 给我们提供一个真正的开源 UNIX 替代品，为了让编程语言在开源世界蓬勃发展，社区需要找来一个格蕾丝·哈伯——我们需要一个开源编译器。

00:12:38:

这是 Langdon White，红帽的平台架构师，来讲讲他对这个事情的理解。

00:12:45 - Langdon White:

在 80 年代，你要买一个编译器很轻松就要花上上万块钱。费用是最大的问题，我没有多余的钱去给自己买编译器。再一个事实是，我必须为所有我想要的目标平台买一个对应的编译器。那个时代大部分是 Unix 平台，但是细节和风格各不相同。

00:13:06:

因此你就不能买一个，你需要为不同的架构，不同的供应商购买多个编译器。

00:13:14 - Saron Yitbarek:

Langdon 指出这不仅仅是成本问题，在一些情况下，对编码工作也带来了问题。

00:13:21 - Langdon White:

大家都沒有意识到，你如何用很特殊的方式来组织你的代码是很重要的。做嵌套 `for` 循环或者做嵌套 `while` 循环之类的事情可能是可以的，这取决于编译器。

00:13:38:

因此，大家都应该明白，如果你不知道编译是如何优化你的代码的，就很难知道如何写出最优的代码。

00:13:49 - Saron Yitbarek:

必须要提的是，我们需要一个免费的、可获得的、可值得信赖的编译器。这就是 **GNU C 语言编译器：GCC**。它横空出世在 1987 年，它是格蕾丝·哈伯的编译器革命和自由软件运动的结合。

00:14:12:

它是使编译更标准化，从而让所有人可以编译别人写的代码。我们编程语言的丰富性要归功于它。

00:14:22 - Carol Willing:

GCC 是开放的，可以说将编程语言推向一个更高的层次。

00:14:29 - Saron Yitbarek:

这是 Jupiter 项目成员 Carol Willing，她是 Python 软件基金会的前理事。

00:14:35 - Carol Willing:

人们开始意识到，专有的语言会在特定时间内被服务于一个目的，但并不能得到开发者社区的认可和热爱。因为如果你是一个开发者，你希望学习最常用的，以及未来最流行的东西。

00:14:59:

我没必要去发展一种将我锁定在一种技术上的技能。我想 Python 成功的一个原因是它是开源的，它有非常简洁的语法。

00:15:11:

它的特点就是允许人们用常见的方法，快速高效地解决问题，也允许大家合作解决问题。我想这就是好的程序和代码库的标志：如果你可以用最小的工作量完成你的工作，并且与他人分享，这是确实很棒的事情。

00:15:35 - Saron Yitbarek:

这么多年过去了，GCC 逐渐的支持了 Java、C++、Ada 和 Fortran 语言，我还可以继续说说它的进步。

00:15:43 - Carol Willing:

通过像 `GCC` 这样的通用底层接口，人们可以根据自己的特殊需求来定制语言。例如，在 `Python` 的世界里，有大量的库，甚至具体到科学 `Python` 世界里，我们有 `numpy`，还有 `scikit-image`、`scikit-learn` 这样的东西。

00:16:08:

每个库都是为一个特定目的而工作。因此，我们也看到了生物信息学和自然语言处理之类的东西。而人们可以在一个共同的基础上，做出很多不同的事情。而且可以把它们放到编程语言或库里，使他们能够在他们特定的行业或领域中优化问题的解决。

00:16:42 - Saron Yitbarek:

因此，这就是编译器技术一头撞进开源运动的结果吧？随着时间的推移，这种不同技术的碰撞，爆炸般地创造了一个新的、被社区支持的语言，大家都可以学习和完善它。

00:16:58:

现在有成百上千的编程语言存活者。

00:17:03 - Carol Willing:

随着开源软件越来越流行和被广泛接受，我们看到了编程语言的大量激增。现在有大量围绕着手机技术的编程语言，不同的编程语言也让游戏开发更加简单快速便捷。各种用途的语言，如 `Python` 和 `Ruby`，它们算是现代网页开发和交付网页应用和网站的基础。

00:17:34 - Saron Yitbarek:

这个队伍还会继续壮大。是的，我们建造的这个巴别塔在未来会更加拥挤。但是你可以把它当作一个聚宝盆，一个语言的盛宴。下面我们将帮你们梳理这个盛宴。

00:17:55:

现在我们已经知道编程语言泛滥的原因了。但是这些对我们有什么具体的意义？我们如何选择对我们重要的语言呢？这是个大问题，因此我们找了一些人帮忙：`Clive Thompson`，是最好的作家之一，他能让科技世界变得更有意义。他是《连线》的专栏记者，《纽约时报》杂志的特约撰稿人，他现在正在写一本关于计算机程序员心理学的书。

00:18:24:

当我们挑选我们想要学习的编程语言时，我们需要知道我们到底想要什么。

00:18:31:

这是我和 `Clive` 反复讨论得出的结论。

00:18:35:

当我作为一个开发新人第一次入手的时候，我就说：“让我选一个最好的编程语言，然后掌握并熟练运用它，不就完事了么。”

00:18:44:

不过事情不会这样简单，否则为什么有那么多的编程语言呢？

00:18:49 - Clive Thompson:

每个语言都有那么点它的特长。因此通常来说，有人创造一个新语言是因为现有的语言满足不了他们的需求。**JavaScript** 就是一个典型的例子。

00:19:03:

Netscape

网景公司曾经在 90 年代中开发了一款浏览器，所有的网站管理员想做一些更具交互性的事情。他们希望有一种方法，使其可以在网站上运行一些脚本。

00:19:16:

当然这些需求都提给了网景。然后他们说：“好吧，现在我们没有可以做到这一点的办法，我们需要一个可以集成到我们浏览器的脚本语言。”

00:19:25:

于是他们雇佣了 **Brendan Eich**，一个被认为很资深的家伙。他当时 32 岁，其他人 21 岁的样子。

00:19:32 - Saron Yitbarek:

这在开发者圈里就很资深了

00:19:35 - Clive Thompson:

他们给他 10 天时间开发 **JavaScript**。因此他真的就 10 天没有睡觉，然后疯狂地捣鼓出一个脚本语言。它看起来有点挫，有点糟，但是可以运行。它允许人们做一些简单的事情，它还有按钮，对话框，弹框和其他东西。

00:19:54:

这就是造一个编程语言的例子，用来做在当时没有办法完成的事情。

00:20:01 - Saron Yitbarek:

是不是很有意思。

00:20:03 - Clive Thompson:

这就是为什么存在这么多编程语言，人们不断进取，去做别人做不到的事。

00:20:11 - Saron Yitbarek:

这也是为什么我对开发者和编程语言之间的关系很感兴趣，我们对这些工具有很强的依赖性。为什么呢？

00:20:22 - Clive Thompson:

有几个原因。一个是有时你学习你的第一个编程语言完全是一场意外，有点像你的初恋。

00:20:30:

我觉得不同性格的人，也会匹配不同类型的编程语言。例如 **Facebook** 是用 **PHP** 设计的。**PHP** 是有点像毛球，它很不规则，它的成长断断续续，**Facebook** 也有点这种感觉。

00:20:49:

与此相反，来自谷歌的伙计决定：“我们想要一个超高性能的语言，因为在谷歌我们的东西都高速运行着，维护着万亿级的用户吞吐。”

00:21:02:

因此他们决定开发一个 Go 语言，Go 非常适合高性能计算。

00:21:08 - Saron Yitbarek:

让我们深入一点。作为一个开发者，我有自己的个性，我是不是对一部分编程语言有天然的喜欢呢？或者我工作用的编程语言可能会影响我的个性？

00:21:25 - Clive Thompson:

两者肯定都存在。但我确实认为当你遇到你喜欢的编程语言的时候，有一种强烈的共鸣感。你上计算机课程时学习了必修语言：他们都教 Java，有时会多教一门 JavaScript 或 Python。

00:21:46:

问题是，你被强制要求，所以你学了它。但当你有选择余地时会怎么样呢？这就是你真正看到一个人的那种情感或心理风格是如何倾向于一门语言的地方，因为我和一个人谈过，他试着学了一堆 JavaScript。

00:22:03:

这是个看起来有点丑的语言，它简直是花括号的噩梦。所以他试了又试，试了又试，失败了又失败。直到有一天他看到朋友在用 Python。在他看来，这是多么的纯净和美妙的语言。因为他是个作家，而 Python 被认为是一个书写型的编程语言，缩进使一切都易于阅读。

00:22:28 - Clive Thompson:

它和他一拍即合，主要在于 Python 的工作方式和外观，漂亮的排版打动了他。

00:22:39 - Saron Yitbarek:

和 Clive 的交流让我意识到，有一些编程语言是硬塞给我们的。我说的是那些古老的编程语言，已经类似于拉丁语了。

00:22:53:

其实我们也有很多选择，去选择适合我们个性的编程语言。如果你想知道最新的东西有哪些，你去问问开发者们周末在用什么就知道了。

00:23:05:

下面是我们对话的更多内容：

00:23:08 - Clive Thompson:

当我问人们：“你闲暇的时候做什么？”回答肯定是一些稀奇古怪的东西。我想实际上其中一个好的，最值得称颂的开发者的特点，就是他们是富有好奇心。

00:23:22:

我知道有人会说，“我要学习 Erlang 了，就是为了好玩。”

00:23:26 - Saron Yitbarek:

我尝试想象人们在周末开发这些项目，项目不是最重要的，他们在学习工具，编程语言才是重要的事情。这才是他们真的想要的状态。

00:23:41 - Clive Thompson:

确切的说，你将看到为什么大家不停的重复迭代开发这些日历和待办事项，因为这是一个快速的学习方法。但需要指出的是，编程语言做了什么以及如何工作都与我们没关系，我只需要尽可能的构建我的东西。

00:23:56:

他们想知道用那样的编程语言思考是什么感觉。是不是会感觉很轻松、刺激、游刃有余？我目前的使用的语言没这个感觉，是不是入门之后一切事情都变得简单了？

00:24:13 - Clive Thompson:

很有可能你遇到一个新语言后会非常兴奋，想起来都很兴奋。

00:24:20 - Saron Yitbarek:

我是一个 Ruby 从业者，作为 Ruby on Rails 开发者我非常自豪，我想大概是两年前，另一个非常著名的 Ruby 开发者，Justin Serrals 做了一个非常好的演讲，倡导编程语言从实用角度出发，没有必要那么性感。

00:24:41:

他的观点是，Ruby 之所以是一个令人兴奋的编程语言的原因在于它很新，而且差不多几年前都已经得到印证了，它已经不需要更新什么了。它已经是一个稳定、成熟的编程语言了。不过因为它的成熟，开发者对它也就不那么兴奋，我们逐步把目光转移到新的花样来了。

00:25:05:

从某个意义上说，很有可能我们的好奇心会扼杀了一个编程语言，或者让它变得庸俗，而这与编程语言本身的好坏没有关系。

00:25:18 - Clive Thompson:

我认为这是完全正确的。事实上，你看到的开发者这个极度好奇心和自学欲望的特点，会让他们不断地寻找新鲜事物。并用这种语言去复制其他语言已经基本做得很好的东西。

00:25:37 - Saron Yitbarek:

是的。

00:25:37 - Clive Thompson:

就是这样，好奇心有好处，同时也是一个陷阱。

00:25:43 - Saron Yitbarek:

是的，说的很好。

00:25:49:

有时我们的好奇心可能是一个陷阱，但是它也是编程语言革命的动力。开发每个新编程语言的时候他们会说，“有没有别的可能？”它们出现的原因在于，开发人员想要做不一样的事情。

00:26:06:

我们想要全新的表达方式。

00:26:08 - 葛丽丝·哈伯:

我向你们承诺一些事情。

00:26:11 - Saron Yitbarek:

我想葛丽丝·哈伯想在这儿最后讲两句。

00:26:15 - 葛丽丝·哈伯:

在未来 12 个月里，你们谁要是说我们一直都是这么做的，我将会瞬间实质化在你的旁边，然后我 24 小时围绕着你。我倒是要看看我能不能引起你的注意。我们不能再相信那句话，这是很危险的。

00:26:34 - Saron Yitbarek:

葛丽丝的故事和第一款编译器提醒我们，只要能可以找到表达方法，我们总有更好的办法做事情。

00:26:43:

不管编程语言有多抽象，不管我们在机器的 1 和 0 高位还是低位浮动，我们都需要确保它是一个明智选择。我们选择一个语言，是让它帮助我们的想法更容易表达出来。

00:27:03:

如果你想更深一步学习编程语言和编译器，你不会很孤独，我们收集了一些很有用的文档帮你深入学习。敬请在我们节目下面留言。请查看

redhat.com/commandlineheroes

00:27:20:

下一期节目，我们将关注开源贡献者的艰辛之路。那些维护者的真实奋斗生活是什么样的？我们如何提出第一个拉取请求？

00:27:32:

我们将带你们简单认识开源贡献。

00:27:39:

代码英雄是红帽的原创播客。你可以在苹果播客、谷歌播客以及其他可能的地方免费收听这个节目。

00:27:48:

我是 Saron Yitbarek，坚持编程，下期再见。

什么是 LCTT SIG 和 LCTT LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 LCRH SIG 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-2/hello-world>

作者: Red Hat 选题: bestony 译者: guevaraya 校对: acyanbird, wxy

本文由 LCRH 原创编译, Linux中国 荣誉推出

《代码英雄》第二季（3）：准备提交

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频博客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《代码英雄》系列播客第二季（3）：准备提交的音频脚本。

导语：想进入开源领域但不知道从哪里开始？你是一个贡献者，想知道为什么只有一些拉取请求被接受？或者，你是一个感觉有点不知所措的维护者？

这一集将探讨投身于一个开源项目的意义。我们将跟随我们的英雄们，跟着他们在开源贡献者的角色中不断进步：从寻找项目并为其做出贡献，到建立和维护繁荣的社区。Shannon Crabill 在 2017 年的 Hacktoberfest 上分享了她是如何开始从事开源工作的，而 Corinne Warnshuis 则介绍了将来自各种背景的人纳入到创造优秀软件的过程中是多么重要。

为开源做出贡献的方式有很多。让我们一起来了解一下。

00:00:03 - Nolan Lawson:

在我刚开始做软件开发的时候，我基本上只做些让自己开心的小项目，像小应用程序、命令行小工具之类的。

00:00:12 - Lindsey Tulloch:

我只是真的不知道作出贡献那么容易。而且你不需要解决 $P = MP$ 那样的难题，你的投入依然可以是很有价值的。

00:00:21 - Kanika Muraka:

本地社区使我有了足够的信心去做出贡献。

00:00:28 - Saron Yitbarek:

当我还完全是个编程新手的时候，我和我的朋友 Dan 一起发起了我的第一个开源 Pull Request
拉取请求（PR），这也是我的第一次开源贡献。

00:00:42:

我听过很多为开源做贡献的故事，说它有多么神奇，有多么可怕。我很清楚，并非所有社区都和善，也不是所有维护者都很友好。

00:00:57:

那个项目本身对新手来说相当不错。我们只是添加了一个 JavaScript 库，让人们可以在线预览网站。这是一个很好的适用范围很广、自成体系的项目。而且如果这玩意儿不起作用，我基本上确信它不会毁掉整个网站。

00:01:18:

然而，我对这个 PR 非常紧张。我和 Dan 阅读了这个库的文档，埋头于写我们的代码。我仍记得当我们最终完成的时候，只是互相看着彼此，好像在说：“就这样吗？”我们发起了 PR。它被审查，然后合并，我想我至今还是对这一切感到惊讶，我还是不知道这些机制是怎么运行的。

00:01:43:

这并不是什么只有他们才能做到的，也不是什么神秘或神奇的事情。我意识到我确实也可以对开源作出贡献。这是我们在这一集中想要传递的一点知识——为开源做出贡献并不神奇，它也不一定可怕。我们将带你一起走过这个过程。

00:02:05 - Lindsey Tulloch:

这是一个突破性的认识，这些项目实际上是完全开放的，我也可以做出贡献。

00:02:15 - Saron Yitbarek:

在这一场的开场白中，你会听到像你一样的代码英雄在加入开源行列时，经历着同样的恐惧。他们分别是 **Nolan Lawson**、**Lindsey Tulloch**、**Kanika Muraka**，这些是今天来做客的代码英雄。

00:02:34:

你正在收听的是红帽公司的原创播客节目《代码英雄》。我是主持人 **Saron Yitbarek**。

00:02:47:

这是一个关于两个代码英雄的故事，他们只是想在广阔的开源世界中，做出更好的东西。他们其中一个人是贡献者，另一个人是维护者。他们都不是真实存在的人物，而是两个虚构人物，用来代表所有的贡献者和维护者，他们和我们分享了他们的故事。希望你也可以在他们的历程中看到一些你自己的影子。

00:03:16:

首先来见一见我们的朋友——贡献者。她是个新手，就像曾经的我们那样。她不确定基本的工作流是什么，但是她看到了一个需求，并且认为她可以添加一个功能来提供帮助。我们这个虚构的贡献者很想提交代码，但是该怎么做呢？

00:03:44 - Corinne Warnshuis:

你一直在成长，学习新技能。而且你不一定必须在孩提时代拆过电脑，才有资格在成年后学写代码。

00:03:52 - Saron Yitbarek:

这位是 **Corinne Warnshuis**，一个名叫“**Girl Develop It**”的很棒的组织的执行董事。该组织的目的是帮助那些可能不太敢提问的，或在聚会中觉得自己不太受欢迎的女性。

00:04:07:

Girl Develop It 意识到做出贡献的难度并不是对所有人而言都是一样的——这是社会造成的问题。作为社区，我们的一部分职责，就是要让世界多一点同情心，以及包容健康多元文化。

00:04:22 - Corinne Warnshuis:

在我们看来，加入开源的壁垒有很多，但我们喜欢称它们为“没有充分的理由”，有三个这样的壁垒将女性专门排斥在技术之外，它们是：刻板印象、可及性和环境。

00:04:40 - Saron Yitbarek:

值得记住的是，促进多元化不仅具有良好的道德意义，同时也有商业意义。

00:04:48 - Corinne Warnshuis:

作为一个行业，技术可能拥有着最大潜力，能给当今世界带来最大的变化。你确确实实希望，来自各行各业的人们都为塑造世界的工具、服务和其他事物做出贡献。我认为来自各种背景的人们一起开发软件，并为开源项目做出贡献真的非常重要。

00:05:13 - Saron Yitbarek:

事实是，我们并非一开始就拥有同样的优势和经验。下一代的伟大开发者可能看上去并不像硅谷的程序员。

00:05:23 - Corinne Warnshuis:

面对面指导对人们而言，历来是非常昂贵而又难以获取到的。再者，我认为从2014年至今，情况有所改善。我认为除了 Girl Develop It 之外的组织（比如 Outreachy 和 CodeNewbie），正通过提供安全的网络或空间，来提出问题并获得更多的舒适感来做到这一点。

00:05:49:

为这些想法和问题提供一个安全而友好的测试平台，是一个不错的起点。

00:06:02 - Saron Yitbarek:

说到新手，回到我们正在追踪的那个贡献者。倘若你不是来自主流背景时，那么第一次提交可能会压力非常大。感觉就好像你必须得证明你自己。不过一旦我们将加入的壁垒降得足够低，我们在准备做出贡献时，实际需要考虑的是什么呢？

00:06:23 - Vincent Batts:

对某些项目感到兴奋这事很酷，但你想要解决的用例是什么呢？

00:06:31 - Saron Yitbarek:

Vincent Batts 在红帽主要从事容器架构方面的工作。他鼓励新的贡献者尝试并有针对性地开展工作。找到你和项目一起有意义的那个利基。

00:06:45 - Vincent Batts:

我认为一个可让人持续贡献的项目通常来自于互惠关系。它满足了你的一个需求，而且在这个过程中你又发现了满足它的需求的方式。即使它是一个由人构成的社区，它也已经成为了一种共生体系。

00:07:01 - Saron Yitbarek:

比方说：

00:07:02 - Vincent Batts:

因为朋友的推荐，我最终弄了一台 Slackware Linux 机器。它非常适合我想做的事情，我帮着将其打包到了主流 Slackware Linux 社区，并最终成为了这个项目的贡献者和维护者。这并不是因为我想成为 Slackware Linux 社区的贡献者，而是因为这是最适合我的一个项目，它的用例正好是我一直致力于解决的事情。

00:07:33:

我想很多人都会遇到这种情况，他们因自己量身定制的用例而编写一个数据库软件。他们发现用 Golang 编写很合适，因此他们写了这样一个高性能的数据库，所以他们能够对 Golang 做出修复或优化方面的贡献，以帮助他们的数据库运行得更快。

00:07:54 - Saron Yitbarek:

你可以找到你自己的小众领域，并开始自然发展。关键在于，从某处开始去做。你不必等待有了学位或绝对的自信才开始。

00:08:08 - Vincent Batts:

如果你需要直接编写代码或文档的经历，或是成为一个后端数据库、Web 服务器的系统管理员，那么好消息是，大部分社区都是基于志愿者构成的。你可以参与诸如 Debian、Fedora 或其他一些类似的项目，这些社区都建立起了文档页面。那些项目必须在某处的 Web 服务器上运行，一些人，甚至是一个没有薪酬的、正在积累经验的社区成员，负责去检查 Web 服务器是否停机或出了什么问题。

00:08:43 - Saron Yitbarek:

Vincent 强调了开源的平等主义本质。无论你来自哪里，只要你愿意，都可以真正开始做出贡献。如果你想的话，你可以为自己扬名立万。

00:08:57 - Vincent Batts:

一旦它被合并，你的名字就会附在某个项目上。你可以公开表示你在某个地方做出了改进，这是相当有意义的事情。

00:09:11:

我曾与那些不是从事日常技术工作的电视修理工和教师共事过，他们很有代表性。他们也在社区上做出了很多贡献。另一方面，我也曾和一些开发者合作过，他们有的能有 30 年开发经验，但他们从来没有像那样公开贡献过代码。

00:09:40 - Saron Yitbarek:

对了，我们的贡献者怎么样了？嗯，她找到了她的定位。她克服了她的恐惧，并最终发起了她的第一个 PR。现在她可以休息一下，并战战兢兢地等待维护者作出回复。

00:09:56 - Vincent Batts:

向上游做贡献有点像第一次上台做才艺表演。你会感到紧张，走上舞台后手心冒汗。你做了一件事，然后它好像就变成了你的成就。你将被彻底改变，跟过去不再一样。

00:10:17 - Saron Yitbarek:

彻底改变？或许吧。事实上，维护者有四种可能的回应：沉默——这很有趣，也可能是完全拒绝，或是直接接受，或是柔和的中间立场——要求修改。

00:10:37:

提交 PR 的几天后，我们虚构的贡献者终于收到了来自维护者的回复。跪迎结果吧，是要求修改。作为一个新手，她将这视为一场小型灾难。她还不知道要求修改是个实际上成功的一步。她甚至还对维护者的简略语气感到一点气愤，听上去就像维护者没空搭理她一样。

00:11:03:

这里存在着一堵墙，新的贡献者不知道墙的另一边正在发生什么。

00:11:12:

我们来认识一位维护者。他正在维护的项目并不是他的全职工作。这是一个周末项目，他时常熬到深夜，给工单排优先级，并且提醒人们发起 PR 时更新文档，然后你就明白了。他相当忙碌，有时甚至出现了一些维护倦怠。

00:11:38:

一位现实生活中的维护者，Nolan Lawson 写了一篇很棒的有关倦怠的文章，最近引起了很多关注。

00:11:51 - Nolan Lawson:

老实说，我认为那篇博文有一部分实在寻求帮助。这是我表达我自己偶然发现了这个开源的项目，起初我觉得它确实很有趣，但现在却感觉没那么有趣了。我不确定该如何恢复兴致。

00:12:05 - Saron Yitbarek:

Nolan 有一份日常工作，但和大多数维护者一样，他在开源项目中投入了大量的休息时间，因为这家伙真的很在意这个项目。讽刺的是，他的部分痛苦来自于，他清楚贡献者也同样很在意这个项目。

00:12:23 - Nolan Lawson:

真正使我精疲力竭的实际上只是如潮水般涌来的好心人。你真的想帮他们，真的很想。他们所做的只是问一个问题，他们所做的就是——他们在你项目中发现了阻碍他们的 bug，或者他们所做的只是——他们真的费心去下载代码并弄清楚它是如何构建的，并提供 bug 修复。他们只是希望你审查他们贡献的代码。

00:12:43 - Saron Yitbarek:

像 Nolan 这样的维护者正不断地审查 PR 库，弄清每个提交将如何发挥作用。他们促使贡献者尽可能做到最好，遵守内部限制，为了能让项目达到更高的水准，用最有意义的方式做出贡献。

00:13:06:

我的观点是，那些维护者有可能不是一个新贡献者所担心的混蛋。他们正努力地想去做一切。他们甚至会花时间标注一些东西保留给新手（很多维护者都这样），以便新手有机会施展自己。

00:13:27:

但到最后，PR 和提交总是非常的多。我们要如何确保这种情况不会发生呢？我们该如何为维护者创造合理的环境呢？

00:13:41:

一种解决方案是——像 Fedora 这样拥有强大社区的开源项目。Fedora 项目负责人 Matthew Miller 解释了项目吸引维护者和贡献者的地方。

00:13:55 - Matthew Miller:

Fedora 项目中许多不光是开发，还有开发过程中所相关的一切。总体来说，IT、CS（计算机科学）事实上都是如此。开源可能并没有足够的这类围绕开源的支持性的角色。

00:14:11 - Saron Yitbarek:

那种支持实际上看起来是怎样的呢？

00:14:14 - Matthew Miller:

我们拿来举例的带薪角色之一是 **Fedora** 项目经理，他帮着让计划按部就班进行，并且监管着人们来保证文书工作完成。让人有酬劳地来做这份工作事实上可以帮着减少官僚主义，因为他们可以把时间花在使项目成为人看懂的事情，而不只是一堆杂乱的文件。

00:14:34 - Matthew Miller:

我认为，像这样的企业参与某些角色，可以赋予你无法用志愿者保证的稳定性。

00:14:43 - Saron Yitbarek:

这有点让我想起了自由职业者使用的共享工作空间。那里有一个共享的接待区、共享的 WiFi 和共享的咖啡。有一个人在管理它，然后你就可以自由地做你自己的事情。

00:14:55:

Matthew 告诉了我们另一个 **Fedora** 采取的策略。他们让你想在项目中途停下来休息一下时，可以很自然，而不是感觉一切都崩溃了。

00:15:04 - Matthew Miller:

我们研究的一个问题是确保领袖角色的漂亮退场，所谓的职位并不一定是终身的委任。你可以重新申请担任角色，并且不会在一年任期后结束后，看起来像被踢出去似的。如果六个月后你想离开，你可以优雅地这样做，而不必觉得这会让人失望。我们已在努力确保人们可以没有障碍地退出。

00:15:27 - Saron Yitbarek:

Matthew 认为，找到支持该开源社区的方法，而又不至于重蹈覆辙才是关键。

00:15:35 - Matthew Miller:

社区几乎就像一个家庭，而不是工作场所之类的东西。在这里做出贡献很有意义，因为你不仅只在为自己或是某些薪酬或终端产品工作。而且由于共事的是你的朋友，你们一起工作能做出比单人作品更伟大的东西。

00:15:56 - Saron Yitbarek:

无论是 **Fedora** 还是其他社区，它们都造就了一个开源贡献得以持续的世界，这个世界还值得努力让它继续变好。

00:16:10:

同时，让我们回到办公桌旁，我们关注的那个贡献者刚完成了维护者要求的修改。她还没意识到，但她即将拥有第一个被接受的 PR。

00:16:24:

当我们谈论诸如倦怠之类的长期工作的问题时，很容易忽视那些早期问题。每天都有很多新的贡献者加入进来。这实际上就是为什么我们需要构建可持续的社区，为所有这些开源魔术提供场地。

00:16:49:

最终，我们的贡献者和维护者共同努力，推动事情向前发展。故事的最后一句话——记住所有的交流都依赖于 GitHub 和 GitLab 之类的开发平台，这些平台使得我们可以聚集到一起。

00:17:09:

我想深入探讨一下这些社区是如何使我们的工作成为可能的。我和 **Shannon Crabill** 谈过这个问题。**Shannon** 白天是个电子邮件开发者，但到了晚上，她正在学习前端开发。她也是一个对社区价值有第一手了解的人。

00:17:28:

去年她参加了一个长达一个月的名为 Hacktoberfest 的开源活动，该活动旨在让更多的人为开源做出贡献。当时，**Shannon** 完全是一个开源新手。

00:17:44 - Shannon Crabill:

回想起十月那时候，我感觉我没有太多工作要做，而且可能还有更多的新手，没找到需要做的东西。如果我提出一些相对容易的待办事项，他们就可以找到用于尝试和学习的突破口，并且开始习惯使用 Git 和 GitHub。

00:18:04:

我认为最困难的部分，在于习惯工作流程并付诸实践。如何推送存储库？如何分享项目？如何处理 PR 和相关的东西？我帮助人们对开源做出了贡献，这事令人惊讶，感觉也确实很棒。

00:18:21 - Saron Yitbarek:

真的很恐怖吗？我觉得如果你是个新手——尽管你拥有存储库，也要走出自己的小世界，把自己放在社区里。现在，维护者所在的社区里出现了正在作出贡献的人，你必须和他们交谈，审查他们的代码并发表意见。这听上去是有点吓人。

00:18:42 - Shannon Crabill:

我认为最初的反应是，“哦，我的天哪，这太酷了”，还有，“天哪，我让自己陷入了什么境地？”我意识到我只合并过自己的代码，合并过自己的 PR 并将其推送到自己的站点上，以及其他所有只关于自己的事情，我从没处理过别人的东西。我认为我之前尚未完成过真正意味上的合并 PR，所以我不得不把这弄清楚。总的来说，将复杂的东西简单地合并起来，仍然是我要努力熟练解决的问题。

00:19:09:

这是旋风一样的感觉，“这很酷，但我不知道该怎么做。”每个人都很好，并且我也努力保持非常友好和真诚，即使只是，“嘿，我有点不知所措。我看到了每个人提交的 PR。虽然今晚我不会找他们，但我明天会的。”人们对这种情况似乎反应良好。

00:19:26 - Saron Yitbarek:

是的。当你维护一个项目时（尤其是作为一个新的开发者时），我一直想知道的是，是不是这意味着你必须是整个存储库贡献者中最聪明的人？你实际上在评估、评判并审查其他人的代码。你是否遇到过，自己懂的东西不如提交 PR 的人那么多的情况？你是如何处理的？

00:19:55 - Shannon Crabill:

好问题。我认为，“噢，我需要成为有史以来最聪明最好的开发者”，这样的想法会成为障碍。我觉得自己很幸运，当我进入这个领域的時候，我没有这样想，所以我能像这样说，“让我们先开始，并且期待未来会发生什么吧。”

00:20:12:

你无需成为一个有 20 年经验的高级开发人员。你只需要有一个好点子，知道如何使用相应软件，然后愿意去学习自己不知道的东西。

00:20:22:

肯定有一两个 PR 为我的项目添加了一些很酷的功能，说实话，如果它们崩溃了，我真不知道该如何修复。我可能会看着代码然后想道，“是的，它崩溃了。”我不知道该怎么做才能从头构建它们。

00:20:34:

我认为这就是它酷的地方。如果只有我一个人在做贡献，我可能就做了一些基本的工作，但没法像其他拥有丰富经验的人所作出的贡献一样，将项目变得那么酷。

00:20:45 - Saron Yitbarek:

作为一个维护者，在使项目更易于访问，更加友好，更加易于贡献的过程中，你学到了些什么呢？

00:20:55 - Shannon Crabill:

哦，确实有件我认为很有帮助，并且我希望我一开始所做的事，那就是尽可能的建立模板，以及文档、文档、文档。

00:21:07:

我的确在我的 README 文件里加了很多东西，并且我认为在项目开始时有一个 README 文件确实是很重要的一步。这件事情意味着你对其他人大声说，“嘿，请查看我们的贡献指南。”我认为我制作了一个 PR 模板、一个议题模板，“单击这里查看当前议题”，这样人们才不会多次提交同样的内容。

00:21:31:

我认为让项目尽可能简单，或尽可能对用户友好，这是你作为维护者需要迈出的一大步。

00:21:38 - Saron Yitbarek:

绝对是这样，我经常看到 README 文件，经常听说他们有多么多么重要。我觉得在 README 文件里还可以做很多事情。归根结底，它就像是一个空白文档，告诉别人去阅读它。你应该在文档里写什么呢？你该如何构建它，来使得它真正与希望做出贡献的人产生关联呢？

00:22:00 - Shannon Crabill:

我在我的 README 文件里放了很多 gif。

00:22:03 - Saron Yitbarek:

很好。

00:22:05 - Shannon Crabill:

我有 gif，也有链接。

00:22:07 - Saron Yitbarek:

我一开始就知道 Shannon 已经认识到了合作关系的重要性。她早就知道，如果有人被邀请来协作，这项目就会变得耀眼，大家也会因此度过美好的时光。

00:22:20 - Shannon Crabill:

有人用开源项目作出很棒的事情，我也认为这很有趣，而且有个有意思的项目，“嘿，我制作了这些很酷的蝙蝠，每次你单击页面的时候，它们都会随机生成。”

00:22:33 - Saron Yitbarek:

我也喜欢人们做不同的事情。如果你真的很喜欢艺术性的东西，你可以做蝙蝠生成功能。如果你想让项目变得整洁，你也可以做。如果你想说，“我会坚持贡献文档，我将花时间为其他的贡献者们打造更干净，更舒适的环境”，那你也可以选择做这个。

00:22:56 - Shannon Crabill:

是的。我已经说得很清楚了，无论你想贡献的是什么，都没有问题。如果你发现了一个拼写错误并且想要纠正它，很好。如果你发现某个链接损坏并且想要修复它，也很好。如果你只是想帮着注释这份代码，使得它更易于阅读和理解，那也将很有帮助。

00:23:12 - Saron Yitbarek:

我认为，你在社区中获得了积极的反馈真的很棒，因为我听说过很多并不是这样美好的故事。人们在网络上并不十分友好，也不太热情善良，尤其是对我们这一些，可能会问出比他们想象中更简单问题的新手。

00:23:33:

你认为，是什么使得你的社区，比起其他社区更加友好呢？

00:23:41 - Shannon Crabill:

只是因为在我们社区，贡献是一件很随意的事。如果你想做出贡献，你可以，这很酷。如果你不想，那也很酷。我理解有人认为开源是一件令人恐惧的大事，你需要具备所有这些经验，并且了解所有这些复杂的语言，或是前后端以及其他的一切，才能够做出贡献。

00:24:03 - Saron Yitbarek:

绝对是这样。做 Hacktoberfest，它是怎样改变了你现在对开源的想法？

00:24:11 - Shannon Crabill:

它肯定对我造成了积极的影响。就像我说的，我有过很棒的经历，而且我希望每一个以某种方式参与我项目的人也能有很好的经历。这毫无疑问促使我想去尝试更多类似的事情，即使它们没有进一步的发展。现在这看起来这些项目更加平易近人了。

00:24:32 - Saron Yitbarek:

真是个好消息。

00:24:34:

这儿有个信息，来自数百家公司的数千人，以及一些根本没在公司工作的人，为 Linux 内核做出过贡献。这意味着基本上运行着互联网的 Linux 是由一群日常英雄在维护的。如果你渴望为开源作出第一次贡献，或许你会想了解有关 Fedora 社区的更多信息，我们有大量的资料正等着来帮助你。请查阅 redhat.com/commandlineheroes 以获得更多信息。

00:25:20 - Saron Yitbarek:

快速提醒一下，这一季我们将构建我们自己的开源《代码英雄》游戏。欢迎你以对你来说合适的方式来做出贡献。快来了解如何成为其中一员吧。我们希望你可以通过 redhat.com/commandlineheroes 和我们一起开发这款游戏。

00:25:42 - Saron Yitbarek:

下一集，我们将讨论残酷的达尔文式错误以及开源开发中失败的美丽之处——它如何困扰我们，指导我们，并使我们变得更好。

00:25:57 - Saron Yitbarek:

《代码英雄》是红帽的原创播客。你可以在 Apple Podcast、Google Podcast 或是其他你喜欢的途径免费收听。我是 Saron Yitbarek，坚持编程，下期再见。

什么是 LCTT SIG 和 LCTR LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 LCRH SIG 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-2/ready-to-commit>

作者: Red Hat 选题: bestony 译者: JonnieWayy 校对: acyanbird, wxy

本文由 LCRH 原创编译，Linux中国 荣誉推出

《代码英雄》第二季（4）：更好的失败

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频博客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《代码英雄》系列播客第二季（4）：更好的失败的音频脚本。

导语：失败是探索时的心跳。我们会在尝试新事物时会多次跌倒。其中秘诀是放弃快速失败，取而代之的是，更好地失败。

本期节目关注在科技领域如何拥抱失败。（对于科技领域来说）以好奇和开放的态度来对待失败是过程中的一部分。Jennifer Petoff 分享了 Google 是如何建立起一种从失败中学习和改进的文化；Jessica Rudder 通过视角的转变，展示了拥抱错误如何能带来意想不到的成功。而 Jen Krieger 则介绍了敏捷框架如何帮助我们为失败做计划。

失败未必是终点。它可以是迈向更伟大事物中的一步。

00:00:00 - Saron Yitbarek:

如果你没有听过这个笑话——两个工程师在编译他们的代码。新手举手喊道：“哇，我的代码编译好了！”；老手则会眯着眼睛喃喃道：“唔，我的代码居然编译好了”。

00:00:18:

如果你已经做过一段时间编程，当你开始思考失败这件事，对有些事情的看法可能就会有所不同。那些过去无法解决的问题，如今开始看起来像一个更大的解决方案中的一个正常组成部分。那些你曾经称之为“失败”的东西，现在看起来像是变相的成功。

你开始希望你的代码无法通过编译。你希望可以一路摆弄和实验它们，调试和修订和重构这些代码。

00:00:37:

你正在收听的是红帽公司的原创播客节目《代码英雄》。我是主持人 Saron Yitbarek。

fail fast

老实说，那句“快速失败”的口号经常被用来作为通往成功的捷径。但是，如果我们不是告诉彼此加快速度并快速失败，而是鼓励彼此更好地失败呢？

00:01:20:

《代码英雄》的第二季将介绍的是开发工作中真实的体验：“当我们生活在代码中，到底感觉如何？又是如何变化的？这也是为什么我们要用一整集的时间来讨论失败，因为正是这些失败时刻促使我们适应它。我们称之为“失败”的东西，是进化的心跳，而开源开发者正在拥抱这种进化。当然，这说起来容易做起来难。

00:01:59:

想象一下，如果一首全新的莎士比亚的十四行诗被发现了。网络上会兴起一阵热潮，每个人都想去搜索它。但这时，有个小小的设计缺陷导致了所谓的“文件描述符耗尽”。这会造成一连串的失败。突然之间，这所有的流量都在越来越少的服务器之间流动。很快，在 Google 上的“莎士比亚”搜索崩溃了，并崩溃了一个多小时。

00:02:33:

现在，你丢掉了 12 亿次搜索查询。这是一场莎士比亚式的悲剧，所有的一切，在网站可靠性工程师（SRE）四处补救的同时上演。

00:02:45 - 配音：

还有你吗，布鲁特？那就倒下吧，凯撒！

00:02:54 - Saron Yitbarek:

不好意思，我打断一下。但上面说的这个莎士比亚事件其实并不存在。事实上，这是一本书《SRE: Google 运维解密》中一系列灾难性场景的一部分。从这本书中学到的重要的一课就是你必须超越灾难本身。这就是我的意思。

00:03:13:

在这个莎士比亚的例子中，当流量被集火到一个被牺牲的单独集群时，这个死亡查询问题就解决了。这为团队赢得了扩充容量的足够时间。但你不能就此止步。尽管这个问题很糟糕，但解决它并不是真正的重点所在。因为失败不一定以痛苦告终，失败也可以引导你的学习。

00:03:38 - Jennifer Petoff:

嗨，我是 Jennifer Petoff。

00:03:41 - Saron Yitbarek:

Jennifer 在谷歌工作。她是 SRE (站 点 可 靠 性 工 程) 团队的高级项目经理，领导谷歌的全球 SRE 教育计划，她也是这本描述了莎士比亚场景的书的作者之一。对于 Jennifer 来说，钻研这样的灾难才能使事情变得更好，但前提是你需要有一个拥抱错误和意外的文化。

00:04:08:

所以，让我们再拿莎士比亚举例子。有一个直接的办法，减少负载可以让你免于这种连锁故障。但，真正的工作将在一切恢复正常之后开始，重点在于事后分析报告。

00:04:25 - Jennifer Petoff:

事件解决后，我们会创建一个事后分析报告。谷歌的每一个事件都需要有一个事后分析和相应的行动项目，以防止将来再次出现问题，以及更有效地检测和缓解未来出现类似事件或整类问题的可能。

00:04:42 - Saron Yitbarek:

这是一个关键的区别。不仅仅是解决这个特定事件，而是看到这个事件告诉你的一系列问题。真正有效的事后分析，不只是告诉你昨天哪里出现了问题。而是让你对今天所做的工作以及对未来的计划有深刻的见解。这种更广泛的思想，灌输了对所有这些事故和失败的尊重，使它们成为日常工作生活中至关重要的一部分。

00:05:12 - Jennifer Petoff:

所以，一个真正好的事后分析不仅仅要解决手头的单个问题，它还解决了整个问题。事后分析的重点是什么地方作对了，什么地方做错了，在何处幸运的解决了问题，以及可以采取哪些优先行动来确保这种情况不会再次发生。如果你不采取行动，历史必将重演。

00:05:32 - Saron Yitbarek:

在谷歌，人们关注的是无 责 任 的 事 后 分 析，这就造成了根本的不同。如果出了问题而没有人要责怪，那么每个人都可以诚实地挖掘错误，真正地从错误中吸取教训，而不必掩盖任何线索，也不必争吵。这些无责任的事后分析已经成为谷歌文化的一个重要组成部分，其结果是一个不必害怕失败的工作场所。这是一种正常情况。

00:06:01 - Jennifer Petoff:

谷歌如何看待失败？100% 的在线时间是一个不可能的目标。如果你认为这是可以实现的，那就是在自欺欺人。所以，失败会发生只是时间和方式的问题。在谷歌，失败是值得庆祝的，因为我们可以从中吸取教训，而事后分析也会在团队中广泛分

享，以确保学到的东西可以广泛使用。

00:06:23 - Jennifer Petoff:

错误是不可避免的，但你永远不想以同样的方式失败两次。犯错是人之常情，但反复犯错是可以避免的。

00:06:34 - Saron Yitbarek:

听到 Jennifer 讨论失败的方式，这真是太有趣了，因为就像她在犯那些错误一样。比如，当事情出错的时候，这意味着你已经走到了一个可以挖掘价值的地方。

00:06:50 - Jennifer Petoff:

你会现场处理这种情况，但事后花时间把发生的事情写出来，让别人可以从中学习。发生任何事件时，你都需要付出代价。如果你不写出事后分析，并真正从这个经验中吸取教训，你就不会重新收回解决问题所花费的成本。在我看来，这是至关重要的一课。在谷歌，我们坚信无责任文化。你不会因为指责别人而获得任何好处，那只会让人们去掩盖失败，而失败，总是会发生。

00:07:27 - Saron Yitbarek:

这里非常重要的一点是，要记住 Jennifer 之前说过的一些话，没有错误的工作是一种幻想，总会有出错的地方。归根结底这是思想的转变。我们可以抛弃那种认为只有一个单一的、可确定的最终目标，即一切最终都会按照我们想象的方式发展的想法。我们没有人试图达到这一目标，事实证明，这是非常强大和积极的东西。

谷歌拥抱失败的做法很有意义。超级实用。但我想知道，这只是口头上的么？我们是否有一些具体的让事情变得更好的失败例子，或者这只是一种当我们进行第 200 次编译时，让我们感觉更好的一种方法。

00:08:26:

事实证明，有人可以回答这个问题。

00:08:29 - Jessica Rudder:

我的名字叫 Jessica Rudder。我是 Github 的软件工程师。

00:08:33 - Saron Yitbarek:

Jessica 在 Github 经历过失败。从某种意义上说，这是一个失败的舞台，在这一过程中，她收集了一些关于失败是通往巨大成功的故事。比如这个：

00:08:50 - Jessica Rudder:

90 年代有个游戏开发公司正在开发一款全新的游戏。从本质上说，这是一款赛车游戏，但他们的转折之处在于将其改为街头赛车。所以当赛车手在街道上飙车时，他们不仅是在互相飙车，而且他们也是与在追赶他们的警车（非玩家角色）赛车。如果一辆警车抓住了你，它会让你靠边停车，然后你就输掉了比赛。然后他们把这些代码衔接起来，然后开始运行，他们发现他们完全调校错了算法：警车只是尖叫着从侧街冲出来，直接撞向玩家的车，而不是追赶玩家的车。

00:09:37:

所以这里简直是一团糟。他们想，不要惊慌，让我们继续前进，看看人们如何看待它的，这样我们就知道该怎么调整算法了。所以他们把它交给了游戏测试人员，他们发现游戏测试人员在逃离警察并试图躲避被这些流氓暴力警车抓捕的过程中获得

了更多乐趣。而事实上，它是如此的有趣，以至于开发团队改变了他们为游戏打造的整个理念。

00:10:17 - Saron Yitbarek:

你能猜出这是怎么回事吗？

00:10:21 - Jessica Rudder:

Grand Theft Auto

所以我们才有了《侠盗猎车手》。我的意思是，它确实是有史以来最畅销的电子游戏，它能存在的全部原因都是因为当时他们没有使用正确的算法时所导致的失误，他们想，好吧，让我们来试试：看看我们得到了什么，看看我们能从中学到什么。

00:10:41 - Saron Yitbarek:

很神奇吧？但这里有个技巧，《侠盗猎车手》团队在遭遇失败时必须保持宽容；他们必须保持好奇心。

00:10:52 - Jessica Rudder:

所以，如果这些开发者没有开放的思想，并决定从这个错误中去学到什么，我们将永远不会有《侠盗猎车手》，我们只能玩一些无聊的、普通的街头赛车游戏了。

00:11:07 - Saron Yitbarek:

Silent Hill

让我们再就游戏主题讨论一分钟，类似的事情也发生在《寂静岭》的制作过程中。这是一个大型的、3A 级的大制作游戏。但他们遇到了严重的弹出问题。局部景观的处理速度不够快，因此突然之间，你会突然发现一堵墙或一条小路突然冒出来。这是一个破坏性的问题，而且他们的开发已经到非常后期。他们是怎么做的？完全放弃游戏，举手投降？还是将错就错。

00:11:42 - Jessica Rudder:

他们所做的就是让这个世界充满了非常浓郁、诡异的雾气。因为事实证明，雾对处理器来说非常容易渲染，而且不会有任何延迟。而且另外，雾使你看不到远处的东西，所以在现实中，那些建筑物仍然会突然出现，但由于雾遮挡了你的视线，你看不到它们。所以当它们进入视野时，它们已经被渲染了，看起来它们是从雾中出来的。

00:12:15 - Saron Yitbarek:

雾是变得如此受欢迎，它基本上被认为是《寂静岭》系列中的一个特点。它限制了玩家的视野，使游戏变得更加恐怖。甚至当处理器的速度快到不需要再掩盖那些弹出的时候，他们也保留了雾气。

00:12:33 - Jessica Rudder:

你无法在没有雾的情况下玩《寂静岭》。而这些雾最初所做的一切都是在掩盖一个错误。

00:12:40 - Saron Yitbarek:

我喜欢这个故事！他们拥抱失败而不是逃避失败，从而挽救了一个重大的发展。这条关于不怕失败的原则也适用于个人的小事，而不仅仅是全公司的决策。从容面对失败是我们一点一点地变得更好的方法。

00:13:01 - Jessica Rudder:

很多时候人们脑子里想的太多了，他们认为失败意味着我不擅长某样东西。并不是代码坏了我还不知道如何修复它，而是“我不知道如何编写 JavaScript”。而且，你永远不会通过说“我不知道如何编写 JavaScript”来学习所需的知识。但是如果你能确定，“哦，我不知道如何在 JavaScript 中实现这个循环”，那么你可以通过 Google 找到答案，而且效果很好。我是说，你仍然需要努力，但你遇到的麻烦会少的多。

00:13:36 - Saron Yitbarek:

因此，无论你是新开发人员还是大型工作室的负责人，我们的错误将我们推向更大的领域，那些实验，那些失败，那些英勇的尝试，它们占据了旅程的大部分。在我所熟悉和喜爱的开源社区里，这是最真实的情况了。失败在开源中可能是一件美好的事情，这就是我们接下来的故事。

00:14:14:

我们在前面看到了失败是如何带来惊喜——那些我们甚至不知道自己想尝试的事情。在最好的情况下，开源开发文化正好符合这一点。它让失败变得正常。为了理解这种愿意失败的想法是如何被引入开源开发的，我和 Jen Krieger 聊了聊。她是 Red Hat 的首席敏捷架构师。我们讨论了对开源失败的态度，以及这些态度是如何塑造无限可能的。请听：

00:14:47:

我想谈谈这个口号，我觉得这也许是一个很好的表达方式。“
fail fast and break things
快速失败，打破现状”，这几乎是为我们社区所设计的一个巨大的召集口号。你怎么看？

00:15:04 - Jen Krieger:

我对此有很多想法。

00:15:06 - Saron Yitbarek:

我也觉得你会有。

00:15:06 - Jen Krieger:

快速失败，在失败中前进，所有这些都是一个意思。所以，在我刚刚参加工作的时候，我在一家没有失败空间的公司工作。如果你做错了什么事情，你就可以准备辞职了。任何人都不能做错事，没有任何空间、途径允许你犯错。这令人们困扰。你绝对没有失败的余地，导致我们几乎陷入一场文化运动。愿意的话，这会催生出一个很棒词——敏捷，以及催生出另一个很棒的词——DevOps。当我看到这些词的时候，我看到的是我们只是要求团队做一系列非常小的实验，帮助他们修正方向。

00:16:02:

这是个，哦，你已经做出了选择，而这实际上是一件积极的事情。你可能会做一个冒险的决定，然后你赢了，因为你做出了正确的决定。或者反之，就是你做了错误的决定，然后你明白了，那不是正确的方向。

00:16:18 - Saron Yitbarek:

是的，这是有道理的。所以，当你把“快速失败，打破现状”当成这个运动的时候，感觉在如何失败，如何以正确的方式失败上还是有一些方式，有一些最佳的实践的。那么，如何以一种正确的方式失败，有哪些最佳实践和原则呢？

00:16:44 - Jen Krieger:

我总是喜欢告诉工程师，他们需要尽早和尽可能多地破坏构建。如果他们正在破坏他们的构建，并且他们意识到他们已经破坏了构建，他们在当下还有机会真正修复 feedback loops 它。而这一切都围绕着“反馈循环”这个概念，并确保你在工作中得到的反馈循环尽可能小。

00:17:08:

所以在开源开发中，我提交了一个补丁，然后有人说，“出于这九个原因，我不会接受你的补丁”，或者“我认为你的补丁很棒，继续吧”。或者，你提交了一个补丁，但是机器人告诉你它失败了，因为它没有正确构建。有各种不同类型的反馈。

00:17:25:

然后在开源开发中，你可能会遇到更长的反馈循环，你可能会说，“我想设计这个新功能，但我不确定所有的规则应该是什么。有人能帮我设计吗？”因此，你进入了一个漫长的过程，在这个过程中，你要进行长时间和详细的对话，而人们参与进来，提出最好的想法。

00:17:45:

所以有各种各样的反馈循环可以帮助你完成这个。

00:17:50 - Saron Yitbarek:

Jennifer 认为，每个公司的反馈循环看起来都不一样。它们是可定制的，人们可以使它们以 100 种不同的方式工作。但重点是，她甚至没有把它们称为失败或错误。她只是称它们为“反馈循环”。这是一个有机系统，这是一种思考整个过程的健康方式。

00:18:11:

与此同时，对这些小毛病的另外一种态度却产生了完全相反的效果。

00:18:18 - Jen Krieger:

有些组织所做的事情是完全错误的。

00:18:23 - Saron Yitbarek:

嗯是啊。

00:18:24 - Jen Krieger:

让你的领导团队（或者，在一个很高的层面上，比如组织）认为，羞辱做错事情的人，或者在绩效结果方面灌输恐惧；就像是，“如果你工作做得不好，就拿不到奖金”或者“如果你工作做得不好，我会把你列入绩效计划。”这些都是会产生敌意的事情。

00:18:50 - Saron Yitbarek:

她描述的是一个不正确的失败。不能接受失败就是失败。她也在呼应 Jennifer Petoff 的态度，对吧？就是我们在这集开头提到的那个无责任的事后分析？

00:19:07:

是的，这很有趣。就好像如果我们在如何一起工作上要求更严格一点，或者只是更用心，更有目的性的在一起工作，我们几乎就会被迫在失败中表现得更好。

00:19:23 - Jen Krieger:

是的。有一些公司已经学会了这一点，而且他们很久以前就学会了，丰田就是一个很好的例子，它接受了这种不断学习和改进的理念，这是我在其他公司很少看到的。就是这样一种想法，任何人在任何时候都可以指出某些东西不能正常工作。不管他们是谁，在公司的哪个级别。在他们的文化中，认为这是对的。这种持续学习和改进的环境，我想说，是一种领先的实践，这是我希望公司能够做到的事情，能够适应失败并允许它发生。

00:20:06 - Saron Yitbarek:

嗯，没错。

00:20:07 - Jen Krieger:

如果你问的是为什么事情进展不顺利，而不是指责或试图隐藏事情，或责怪别人，这就会造成完全不同的情况。那就是改变对话方式。

00:20:23 - Saron Yitbarek:

这很有趣，因为你之前提到过“快速失败，打破现状”这句话是这种文化，这种对过去做事方式的反击。但这听起来似乎是一种口头禅，也许也创造了一种在公司内部、技术团队内部的不同的团队工作方式。再给我讲讲这个问题，它是如何改变了开发人员看待自己角色的方式，以及他们与公司其他人互动的方式？

00:20:55 - Jen Krieger:

我早期和工程师一起工作的时候差不多是这样的，工程师们都坐在一个小区域，他们互相交谈。他们从未真正与任何商业人士进行过交流。他们从来没有真正理解他们的任何需求，我们花了很多时间真正专注于成功所需的东西，而不一定是企业实际完成工作所需的东西。所以，它更像是，“我是一个工程师，我需要什么才能编写这个功能片段？”我观察到，今天在几乎每一个和我一起工作的团队中，对话方式已经发生了巨大的变化，“作为工程师我需要什么才能完成工作”变成了“客户是谁，或者用户需要什么才能真正感觉到这我做的这块功能对他们来说是成功的？他们如何使用产品？我该怎样做才能让他们更轻松？”

00:21:56:

很多这样的对话已经改变了，我认为这就是为什么如今公司在提供有意义的技术方面做得更好的原因。我还想说的是，我们发布的速度越快，我们就越容易知道我们的假设和决定是否真正实现了。所以，如果我们对用户可能想要什么做了假设，在此之前，我们需要等待，比如，一年到两年才能确定这是不是真的。

00:22:25:

而现在，如果你看看亚马逊或奈飞的模式，你会发现，他们每天会发布数百次假设的客户需求。他们从使用他们的应用程序的人们那里得到的反馈，会告诉他们他们是否在做用户需要他们做的事情。

00:22:46 - Saron Yitbarek:

是的，这听起来需要更多的合作，因为即使是你之前提出的关于构建、破坏构建、经常破坏它的建议，这就需要工程团队或开发人员与 DevOps 保持步调一致，以便他们能够破坏它，并了解尽早发布并经常发布是什么样子的。听起来这需要双方更多的合作。

00:23:15 - Jen Krieger:

是的，对于拥有敏捷教练这个头衔的人来说，或者以我作为首席敏捷架构师看来，总是很有趣，因为《敏捷宣言》的初衷是让人们从不同的角度来考虑这些事情。我们通过开发和帮助别人开发来发现更好的开发软件的方法。它确实是敏捷所要做的核心、根本和基础。因此，如果你将 10 年，15 年以上的时间快速推进到 DevOps 的到来，并坚持我们需要持续进行集成和部署。我们有监控，我们开始以不同的方式思考如何将代码扔出墙外。

00:23:56:

所有这些东西都是我们最初开始讨论敏捷时应该想到的。

00:24:03 - Saron Yitbarek:

嗯。绝对是的。所以，不管人们如何实践这种失败的理念，我认为我们都可以接受失败，将失败规范化只是过程的一部分，是我们需要做的事情，是我们可以管理的事情，是我们可以用“正确的方式”做的事情，这是一件好事。它对开源有好处。跟我说说这个新运动的好处，这种接受失败是过程的一部分的新文化的一些好处。

00:24:36 - Jen Krieger:

看着这个过程发生是一件美妙的事情。对一个人来说，从一个他们害怕可能发生事情的环境，到一个他们可以尝试实验、尝试成长、尝试找出正确答案的环境。真的很高兴，就像它们已经盛开花朵。他们的士气提高了，他们真正意识到他们可以拥有的是什么，他们可以自己做决定，而不必等待别人为他们做决定。

00:25:05 - Saron Yitbarek:

失败即自由。啊，我喜欢! Jen Krieger 是红帽公司的首席敏捷架构师。

00:25:19:

并不是所有的开源项目都像 Rails、Django 或 Kubernetes 那样声名鹊起。事实上，大多数都没有。大多数都是只有一个贡献者的小项目，解决一小群开发人员面临的小问题的小众项目，或者它们已经被抛弃，很久没有人碰了。但它们仍然有价值。事实上，很多这样的项目仍然非常有用，可以被回收、升级，被其他项目蚕食。

00:25:54:

而另一些人通过他们的错误启发我们，教导我们。因为在健康的、开放的舞台上，失败会带给你比胜利更好的东西。它给了你洞察力。还有一点。尽管有那些死胡同，尽管有各种冒险的尝试和惊呼，但开源项目的数量每年都在翻倍；我们的社区正在繁荣，事实证明，尽管因失败我们没有繁荣，但因失败我们正在繁荣。

下一集预告，DevOps 世界中的安全性如何变化。持续部署意味着安全正在渗透到开发的每个阶段，这正在改变我们的工作方式。同时，如果你想了解更多关于开源文化的知识，以及我们如何改变围绕失败的文化，请访问 redhat.com/commandlineheroes，免费资源等着你。

00:26:54 - Saron Yitbarek:

《代码英雄》是红帽的原创播客。你可以在 Apple Podcast、Google Podcast 或是其他你喜欢的途径免费收听。我是 Saron Yitbarek，坚持编程，下期再见。

什么是 LCTT SIG 和 LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 LCRH SIG 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-2/fail-better>

作者: Red Hat 选题: bestony 译者: bestony 校对: wxy

本文由 LCRH 原创编译，Linux中国 荣誉推出

《代码英雄》第二季（5）：关于 **DevSecOps** 的故事

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频播客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《代码英雄》系列播客第二季（5）：关于 DevSecOps 的故事的音频脚本。

导语：不良的安全和可靠性实践会导致影响数百万人的中断。现在是时候让安全加入 DevOps 运动了。并且，在 DevSecOps 的世界中，我们可以创造性的提升安全性。

每月发现一个漏洞曾经是常态。而现在，由于敏捷流程和 DevOps 团队，软件开发的进展迅速。Vincent Danen 告诉我们，这如何导致被认为是漏洞的东西急剧增加。前亚马逊灾难主管 Jesse Robbins 介绍了公司如何为灾难性故障和漏洞做好准备。而 Elastic 的产品安全主管 Josh Bressers 则展望了科技领域安全的未来。

我们不应该把安全团队当成脾气暴躁的妖怪。听听 DevSecOps 团队如何将英雄们聚集在一起，以实现更好的安全。

00:00:01 - 众议院小组委员会代表：

1991 年 6 月 26 日，在华盛顿特区，马里兰州和西弗吉尼亚州的大部分地区，以及我的家乡的大部分地区都因公共电话网络的大规模故障而瘫痪了。然而，随着技术变得越来越复杂，网络系统越来越相互依存，反复发生故障的可能性也会增加。似乎并没有警告说会发生这种情况。

00:00:23 - Saron Yitbarek:

在 20 世纪 90 年代初，有 1200 万美国人遭受了大规模的电话网络故障。人们不能给医院打电话，企业不能给客户打电话，父母不能打电话给托儿所。对于一个基础设施严重依赖于万物互联的计算机系统的国家来说，这是一场混乱也是一记警钟。这些计算机网络变得越来越大，然后当它们出现故障时，故障时间就会很长。

00:01:01:

电脑故障会导致电话系统崩溃。在今天代码中的一行小错误的后果比以往时候都要严重。

00:01:15:

我是 Saron Yitbarek，这里是红帽公司的原创播客节目《代码英雄》。

00:01:24:

因此，软件安全性和可靠性比以往任何时候都重要。传统的瀑布式开发方法，安全性只是一个附加流程而已，已经不再适用。我们生活在一个 DevOps 的世界里，一切都变得更快、更敏捷、扩展性更强，这在电话网络崩溃时是他们无法想象的。这意味着我们的安全和可靠性标准必须不断改进，以应对这些挑战。

00:01:55:

在本集中，我们将研究如何将安全性集成到 DevOps 中，我们还将探索在运营中构建可靠性和弹性的新方法。即使在介绍了所有这些之后，我们知道还有很多东西可以讨论，因为在 DevSecOps 的世界里，对于开发人员和运营人员来说，事情都在快速变化。这些变化意味着不同的事情，这取决于你的立场，但这是我们的看法。我们也很想听到你们的消息——所以如果你认为我们错过了什么，不要害羞——在网上联系我们。

00:02:34:

好了，让我们开始探索这个全新的领域吧。

00:02:43:

事情就是这样，让安全性和可靠性跟上时代的步伐，并为 DevOps 世界做好准备，这意味着我们必须对工作方式进行一些关键的调整。第一，我们必须拥抱自动化。我的意思是，想想双因子认证的逻辑。想想那些难以想象的艰巨任务吧。很显然，你不能仅仅通过增加员工来解决问题，所以第一点就是拥抱自动化。

00:03:15:

然后，第二点，这个可能不是那么明显，那就是它真的改变了文化，使安全不再是一个祸害。稍后我将解释我所说的改变文化的含义。但是让我们一个一个的解释这两点。首先，拥抱自动化。

00:03:42:

以前，应用程序的部署在每个单独的发布之前都涉及到一个人为的安全审查，我不知道你是否注意到了，但是人为的审查可能会有点慢。这就是为什么自动化是在 DevOps 构建安全性的关键部分。以 Verizon 最近的数据泄露报告为例。他们发现，81% 的与黑客相关入侵涉及密码被盗或者弱密码。从表面上看，这是一个非常简单的问题，但是规模却很大。就像我之前所提及到的，你不能用工作人员去解决 3000 万个密码问题，对吧？问题在于解决大规模问题，而每次的答案都是一样的。那就是自动化，自动化。

00:04:36 - Vincent Danen:

如果你等待人参与进来，那么规模就不会扩大。

00:04:41 - Saron Yitbarek:

Vincent Danen 是红帽公司产品安全部门的主管，在过去的 20 年里，他见证了 DevOps 的快速发展。安全团队不得不竞相追趕。

00:04:56 - Vincent Danen:

刚开始的时候，每个月都有漏洞，后来变成了每隔一周，然后是每周都有。现在，每天都能找到几百个漏洞。

00:05:08 - Saron Yitbarek:

有趣的是，Vincent 说，随着安全团队的发展，实际上会出现更多的漏洞，而不是更少。

00:05:17 - Vincent Danen:

我们永远不会说，哦，我们现在安全了，我们做完了，我们的工作结束了。安全审计会一直存在，就像呼吸一样，这是必须要有的。

00:05:27 - Saron Yitbarek:

事实证明，对于安全性和可靠性团队来说，细节的问题变得越来越重要。

00:05:35 - Vincent Danen:

当我们在寻找这些漏洞时，我们会发现更多的东西，而且这个趋势还将继续。因为你会发现新的漏洞类型和一些我们可能认为不太重要的东西，或者以前甚至不知道它们存在的东西。我们会发现这些东西发展的速度很快，而且数量更多，因此规模爆炸性增长。知识、软件的数量、消费者的数量都促进了该领域安全性以及漏洞的增加。

00:06:06 - Saron Yitbarek:

一旦你将安全视为一个不断发展的问题，而不是随着时间的推移而“得到解决”的问题，那么自动化的理由就会变得更加充分。

00:06:18 - Vincent Danen:

嗯，我认为有了自动化，你可以以一种非常快的方式将这些东西集成到你的开发流水线中，这是其一。其二，你不需要人类来做这些工作，对吧？计算机不需要睡觉，所以你可以在处理器允许的情况下以最快速度浏览代码，而不是等待人类通过一些可能相当乏味的命令行来查找漏洞。

00:06:44:

然后通过模式匹配和启发式方法，甚至在开始编写代码的时候，你就可以知道代码中那些地方是易受攻击的。如果在你编写代码的时候，在你的 IDE 或者工具中有一个插件，它能告诉你。嘿，这看起来有点可疑，或者你刚刚引入了一个漏洞。在你提交代码之前你都可以纠正这些可疑点或者漏洞。

00:07:08 - Saron Yitbarek:

安全在进步。这真是一笔巨大的奖励。

00:07:12 - Vincent Danen:

每一天，甚至每一小时，都会有很多东西涌现出来。通过持续集成和持续部署，你写了代码，10 分钟后它就被部署了。因此，在代码被推送之前自动进行验证是非常关键的。

00:07:32 - Saron Yitbarek:

我们可以使用各种各样的工具来完成这个任务，不管是静态代码分析，还是 IDE 的插件，或者是一大堆其他选项。我们将在 redhat.com/commandlineheroes 上分享一些我们最喜欢的片段。

00:07:53:

一旦我们有了这些工具，它们将帮助我们把安全放在首位。结果就是，DevOps 被重新定义为 DevSecOps。安全被纳入到流程中。

00:08:08 - Vincent Danen:

就像开发人员和运维人员结合的方式一样，你将这两个规则合成到了一个规则。现在，你有了 DevOps，并将安全这第三个组件与开发和运维集成到一起，我认为这非常重要。因为事后才考虑安全性，这会使安全性变得非常被动、昂贵以及可能会损害消费者。当你一开始就把安全代入其中，你就可以完成开发工作，从头到尾进行安全检查并开始运作。

00:08:44 - Saron Yitbarek:

当然，就像我们在这一集的开头提到的，自动化只是一个大蛋糕的一半，而 Vincent 也明白这一点。

00:08:53 - Vincent Danen:

并不仅仅是一部分。不能仅仅在你的 CI/CD 流水线中随便引入一个工具就期望一切都变好。为了达到我们希望看到的最终有益结果，需要使用各种技术和行为。

00:09:15 - Saron Yitbarek:

自动化确实让我们做到了一半，但我们必须记住另一部分——稍微模糊一点的那一部分。让我们一起来说，那就是文化部分，让开发者和运维人员都一起参与进来，这样这些问题就不再是可怕的问题。

00:09:33:

我们必须改变一种文化，而有些人正在学习以一种最不痛苦的方式，通过游戏的方式来做到这一点。

00:09:44:

现在让我们来看看事情的另一面。如今建立庞大的基础设施很容易，但这并不意味着我们应该做粗制滥造的工作。我们仍然应该努力改进我们的系统，确保可靠性，未雨绸缪。这就是 **Jesse Robbins** 正在努力实现的。

00:10:08:

如今，**Jesse** 是 **Orion Labs** 的 CTO，但在此之前，他因在亚马逊被称为灾难大师而名声大噪。在那里，**Jesse** 特别是在让大家至少意识到这些问题这件事上几乎是个奇才。他通过一个叫做“游戏日”的活动来做到这一点。让其中可能涉及成千上万的员工进行故障演练，通过灾难演练来习惯系统被破坏并了解发生的原因和方式。

00:10:39:

下面是 **Jesse** 和我在讨论，尤其是在运营方面如何建立可靠性和弹性。

00:10:47:

大家都知道你做了很多非常酷的事情，其中之一就是你在亚马逊做的活动——“游戏日”。那是什么？是什么游戏？

00:10:58 - Jesse Robbins:

“游戏日”是我创建的一个项目，通过大规模破坏来测试最脆弱系统的运行情况。如果你是 **Netflix** 的“混乱猴子”的粉丝，“游戏日”则是我的一个可以实现类似的所有事情的东西。实际上，它非常专注于建立一种卓越的运营文化，建立大规模测试系统的能力，当系统崩溃时能了解它们是如何崩溃的以改进它们。然后还要建立一种文化，能够对事件做出反应并能恢复。它是按照事故指挥系统建模的，这是世界各地的消防部门用来处理任何规模事故的系统。

00:11:56:

它的诞生源于...

00:11:58 - Saron Yitbarek:

旁白，**Jesse** 早在 2005 年就经过训练成为一名消防员。在那儿，他了解了这个事故指挥系统，最终激发了“游戏日”的灵感。因此，所有做这些故障演练的开发人员，都要感谢 **Jesse** 对消防和应急管理的激情。好了，回到我们的谈话。

00:12:22 - Jesse Robbins:

弹性是一个系统的能力，这包括人和这些人建立的适应变化、应对失败和干扰的能力。而建立这种文化最好的方法之一——建立一种文化，能够对这种类型的环境做出反应，并真正理解这些环境是如何工作的——就是提供人员培训演习。这些演习可以很简单，比如重启服务器，也可以很复杂，比如关闭整个数据中心造成大

规模故障等等。所以，“游戏日”首先是一个过程。在这个过程中，你通过让整个组织聚集在一起，讨论系统如何发生故障，并思考人类对故障发生的预期。而这个演习本身就是“游戏日”开始时最有价值的部分之一。

00:13:24:

但是，当你实际对系统做了一个或大或小的破坏后。当你这样做的时候，你就可以看到每个人是如何反应的。你看到系统崩溃了，可能是之前安全的东西崩溃了，一个很容易理解的组件或者是某个东西暴露了一个潜在的缺陷。这些问题隐藏在软件、技术或者大规模的系统中，只有当你遇到极端或者意外事件时，我们才能发现。“游戏日”的目的是为了训练员工并且建立系统让你了解他们如何在压力下工作。

00:14:12 - Saron Yitbarek:

所以当我听到“游戏日”的时候，我就会想，“这是对某个特定事件的回应吗？它是从哪儿来的？”

00:14:20 - Jesse Robbins:

因此，“游戏日”刚开始的一段时间内，因为我知道自己的角色以及作为消防员和应急管理者的背景，因此将文化方法从注重预防失败的观念转变为拥抱失败非常重要，接受失败发生。激发我这样做的部分原因是我自己的经历，你知道，了解系统，比如建筑是如何倒塌的，市政基础设施是如何倒塌的，以及灾难是如何发生的，以及灾难给人们的压力。所以说，如果环顾我所在工作场所所具有的复杂性和运营规模就会知道，想要真的构建成一个高可靠性、持续在线环境的唯一办法就是拥抱消防服务的方法。我们知道失败会发生，这不是如果的问题，而是什么时候的问题。就像我之前的消防队长说的，不是你选择时机，而是时机选择你。你只需要在它发生的时候准备好即可。

00:15:28 - Saron Yitbarek:

哦，这个不错。所以当你第一次开始做“游戏日”并思考如何为灾难场景做准备时，每个人都同意了吗？你得到任何反对意见了吗？

00:15:40 - Jesse Robbins:

每个人都认为我疯了。因此，肯定有人反对。有趣的是，有一种非常简单的方法可以克服这种抵制，那就是首先创造出我称之为“冠军”的东西。你要教一小群人，如何以非常安全的方式工作，然后你能够使用一些信服的指标。你能够说，看，让我们只需衡量发生了多少分钟的中断，我的团队经过了这种培训并以这种方式进行操作的停机时间有多少分钟。相反，你的团队没有这个，并且似乎认为进行这种类型的培训和练习没有价值或者不重要。

00:16:25 - Jesse Robbins:

你一旦完成了这种事情，基本上就会有我所说的引人注目的事件。因此，经常会有断电或其他事情让组织突然意识到：哦，我的天哪，我们不能再像以前那样继续做事了。这就是你用来说服怀疑论者的方法。你一方面使用数据和性能数据，再结合指标，然后讲故事，然后等待一个大的故障或者可怕的事情发生。然后，你可以说，如果我们要在 web 规模或者互联网规模上运维，整个组织都需要这种应变能力。

00:17:06 - Saron Yitbarek:

嗯嗯。所以我喜欢它的原因是它不只是停留在亚马逊内部。相反，它在传播。很多其他公司也在这么做。很多人最终接受了要为故障做好准备这个知识和过程。那下一步是要做什么？我们如何将从“游戏日”中学到的知识继续运用到未来的项目和公司中？

00:17:31 - Jesse Robbins:

我喜欢把它称为趋同进化。每个在 web 上运行的大型组织现在都采用了我提倡的事件管理基础的一个版本，并创建了他们自己的“游戏日”测试。比如，Netflix 将其称为“混乱猴子”。谷歌有他们的 Dirt 计划。

00:17:57 - Saron Yitbarek:

那么你对未来的“游戏日”有什么寄望呢？

00:18:00 - Jesse Robbins:

首先让我感到兴奋的是，我们可以看到人们从闭门造车思维的转变。系统从根本上是相互联系，相互依赖的，而且由世界各地试图有所成就的聪明人构建和运行的。

00:18:22:

几年前，当我刚参加工作时，对运维工作毫不关心，我觉得那非常无趣。然后突然的，我们发现自己能够传播这样一种理念：开发人员和运营人员一起工作是在互联世界中构建和运行有意义的技术的唯一途径。

00:18:44:

所以我对未来的希望是，第一，我们看到越来越多的人接受这些想法并学习它。明白了当你建造了人们依赖的东西时，你有义务确保它是可信赖的、可用的、可靠的，它是人们可以作为日常生活的一部分来使用的东西。

00:19:05:

而且我们也看到了一种新的学科的诞生。“游戏日”的这种思维模式正在被研究，也有博士正基于这个撰写博士学位论文。它正在不断建立中。

00:19:16 - Saron Yitbarek:

这真的是太棒了。

00:19:16 - Jesse Robbins:

也有写这方面的书，但是包含这些新资源的没有。只有少数人在会议上谈论他们认为世界应该怎么运转。所以我的那种鼓舞人心的希望是，你要明白如果你正在构建软件和技术，那么你真的成为了社会基础设施的一部分。所以作为一名消防员，我所努力贡献的一系列技能和正在出现的技术，这些技术将使它走得更远，它们是建造人们日常生活所依赖的东西的基础的一部分。

00:19:53 - Saron Yitbarek:

很好。这是一个很好的结束方式。Jesse，谢谢你抽出时间来。

00:19:56 - Jesse Robbins:

是的，谢谢。

00:11:59 - Saku Panditharatne:

我认为所有这些因素都不利于采用最佳软件。

00:20:02 - Saron Yitbarek:

在 Jesse 看来，像“游戏日”或者“混乱猴子”这样的演习是我们不断发展的科技文化的重要组成部分，但它们对于整个社会也至关重要。我很喜欢他很重视这个，因为他是对的。我们的世界取决于我们所做的工作。早在 90 年代，当电话网络开始崩溃时，这一点就很明显了。

00:20:26 - 众议院小组委员会代表:

我们所知道的现代生活几乎陷于停顿。

00:20:31 - Saron Yitbarek:

这是一种伴随的责任。我们有责任关心安全和可靠性，关心我们所建造东西的弹性。当然，当谈到在 DevOps 中的构建安全性时，我们才刚刚开始。

00:20:53 - Saron Yitbarek:

这是 Josh Bressers。他是数据搜索软件公司 Elastic 的产品安全主管。对 Josh 来说，尽管计算机行业已经成熟了半个世纪左右，但我们在讨论的安全问题却让人觉得它是刚刚才出现的。

00:21:11 - Josh Bressers:

实际上，就像我想说也行作为一个专业，安全仍然是非常新的东西，有很多事情我们还不是很了解。

00:21:19 - Saron Yitbarek:

但我们确实明白，在 DevSecOps 的世界中，有一些非常好的机会可以创造性的思考安全能达到什么成就。

00:21:29 - Josh Bressers:

我最近和一些人讨论了一个概念，他们利用用户行为来决定用户是否应该能够访问系统。每个人都有特定的行为，比如他们来自哪里，他们访问系统的时间，他们打字的方式，他们移动鼠标的方式。所以我认为，如果我们做得好，他们的这些行为，可以产生一些非常强大结果，如果我们能做到这一点，我们可以注意到有人在做什么。然后假设我表现的很奇怪，因为我刚刚扭伤了手臂。但你知道，另一端并不知道。

00:22:05 - Josh Bressers:

因此，它可能会说，这种行为就有些奇怪，我们就会希望你使用双因子认证登录，并且还会向您发送条短信或其他内容，对吧？这就从用户名和密码变成了更有趣的东西。所以我认为用新的和独特的方式来对待这些问题将是关键。在很多情况下，我们只是还没到那一步。

00:22:27 - Saron Yitbarek:

实现这一目标需要我们所描述的两大步骤。第一步，就是自动化，这很重要，因为.....

00:22:35 - Josh Bressers:

人类很不擅长重复地做同一件事。

00:22:38 - Saron Yitbarek:

很公平。然后，我们有了第二步，就是文化，无论我们的职称是什么，我们所有人都有不安全感和责任感。

00:22:49 - Josh Bressers:

当大多数人想到安全团队时，他们不会认为那是一群快乐的好好先生，对吧？一般来说，这些人都很可怕，脾气暴躁，令人讨厌，如果他们出现了，就会毁了你的一天。没有人想要这样，对吧？

00:23:10 - Saron Yitbarek:

但我认为我们可以克服这种偏见，因为我们必须这样想——每天都有更多的安全威胁发生，而且 IT 基础设施每天都在变得更大、更强。把这两个事实放在一起，你最好可以生活在一个被安全环绕的世界里。一个非常 DevSecOps 的世界，在这个世界里，开发人员和运营人员都在提升他们的安全，提高他们的可靠性。我所谈论的是一个自动化被整合到每个阶段的未来，每个人对这些问题的态度变得更加全面。这就是我们保护未来系统安全的方法。这是我们保持电话响，灯开，所有现代生活健康强壮的方法。如果你查一下《福布斯》全球 2000 家公司的名单，也就是前 2000 家上市公司，你会发现其中整整四分之一的公司都采用了 DevOps。集成的敏捷工作场所正在成为规则。并且在几年之内，关于 DevSecOps 的思考可能会成为第二天性。我们希望尽可能快，但是当团队中的每个成员都齐心协力时，长距离比赛实际上会更快。

00:24:40 - Saron Yitbarek:

Zettabyte

下一集，我们将面临数据的大爆炸。人类已经进入了泽字节时代。到 2020 年，我们将在服务器上存储大约 40 泽字节的数据，而这些信息大部分甚至现在还不存在。但是我们该如何让这些数据有用呢？我们如何使用高性能计算和开源项目让我们的数据为我们所用呢？我们会在 Command Line Heroes 第 6 集中找到答案。

00:25:13 - Saron Yitbarek:

提醒一下，我们整季都在致力于《代码英雄游戏》的开发。这是我们自己的开源项目，我们很喜欢看着它的诞生，但是我们需要你来帮助我们完成。如果你点击 redhat.com/commandlineheroes，你可以发现如何贡献。你也可以深入了解我们在这节课中讨论过的任何内容。

00:25:39 - Saron Yitbarek:

《代码英雄》是红帽原创播客。你可以在 Apple Podcast、Google Podcast 或任何你想做的事情上免费收听。我是 Saron Yitbarek。坚持编程，下期再见。

什么是 LCTT SIG 和 LCTT LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 LCRH SIG 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-2/the-one-about-devsecops>

作者: Red Hat 选题: bestony 译者: mrpingan 校对: bestony, wxy

本文由 LCRH 原创编译, Linux 中国 荣誉推出

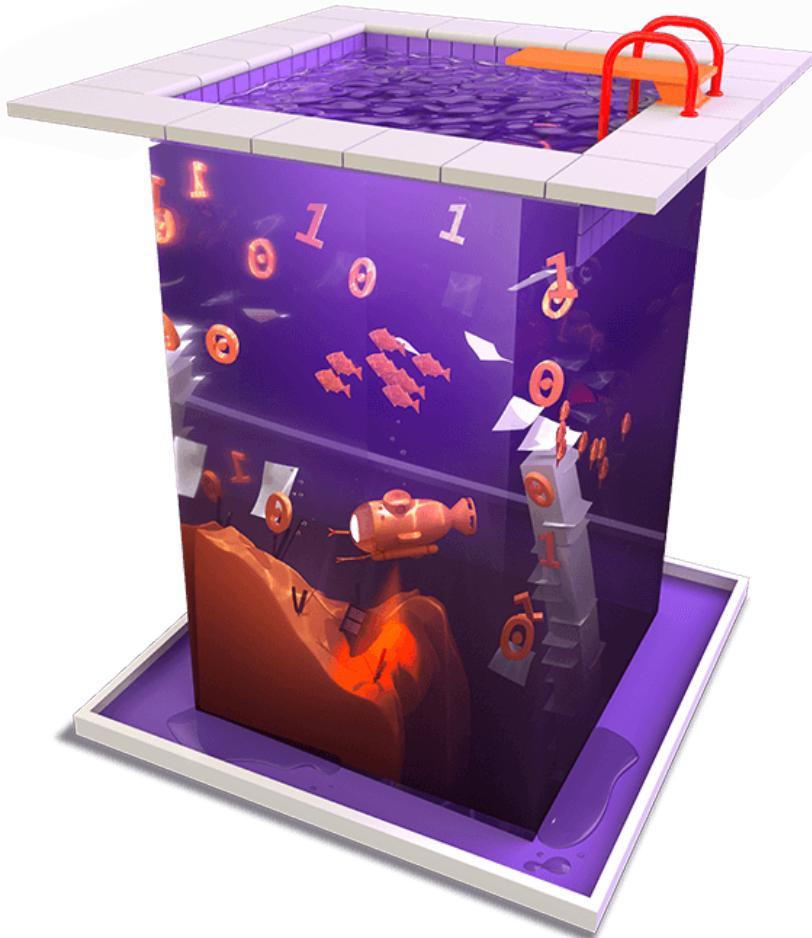
《代码英雄》第二季（6）：数据大爆炸

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频博客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《代码英雄》系列播客第二季（6）：数据大爆炸的音频脚本。

导语：大数据将有助于解决大问题：我们如何种植粮食、如何向需要的人运送物资、如何治疗疾病。但首先，我们需要弄清楚如何处理它。

现代生活充满了相互联系的组件。我们现在一天产生的数据比几千年来的数据还要多。Kenneth Cukier 解释了数据是如何发生变化的，以及它是如何开始改变我们的。Ellen Grant 博士告诉我们波士顿儿童医院是如何使用开源软件将堆积如山的数据转化为个性化治疗方法。Sage Weil 则分享了 Ceph 的可扩展和弹性云存储如何帮助我们管理数据洪流。

收集信息是了解我们周围世界的关键。大数据正在帮助我们拓展永不停歇的探索使命。

00:00:03 - Saron Yitbarek:

如果你把从人类历史早期到 2003 年创建的所有数据计算在内，你将得到大约 500 万 GB 的数据。我们昨天创建了多少 GB 数据？

00:00:15 - 问卷调查答案 1:

哦，天哪，10 万。

00:00:21 - 问卷调查答案 2:

可能是 500 万 GB 数据。

00:00:23 - 问卷调查答案 3:

我们在昨天一天之内创建了多少 GB 的数据？1000 万 GB 数据？

00:00:31 - 问卷调查答案 4:

我不太知道，可能是 200 万 GB 数据？

00:00:36 - 问卷调查答案 5:

也许一天就有 100 万 GB 数据？

00:00:40 - Saron Yitbarek:

答案是？超过 25 亿 GB 数据！

00:00:44 - 问卷调查答案 1:

哇哦。

00:00:44 - 问卷调查答案 2:

25 亿？

00:00:45 - 问卷调查答案 3:

所以，我们已经打破了世界纪录。

00:00:45 - 问卷调查答案 4:

那可真是很多个 G 啊。

00:00:45 - 问卷调查答案 5:

我都不敢相信那有这么多的数据。

00:00:52 - Saron Yitbarek:

在 2016 年，我们的年度在线数据流量首次超过了 1ZB。一个 ZB 是 1000^7 字节。好，记住这个数字了吗？现在把它乘以 3，因为那是我们将在 2021 年拥有的数据量大小。

00:01:10:

我知道，大脑不是以 ZB 为单位进行思考的，但请你至少暂时记住这个数。我们的 IP 流量将在五年内翻三番。这是数据的洪流，而我们正处于其中。

00:01:24:

在刚过去的一分钟里，人们发出了 1600 万条短信；与此同时，在我说出这句话的时间里，谷歌处理了 20 万条搜索。

00:01:37:

如果我们能在数据洪流来临时做好准备、站稳脚跟，那么隐藏在其中的模式、答案和秘密可以极大地改善我们的生活。

00:01:50:

我是 Saron Yitbarek，这里是《代码英雄》，一款红帽公司原创的播客节目。浪潮近在眼前。这里是第二季第六集，数据大爆炸。

00:02:17:

我们如何处理如此大量的数据？采集到这些数据后，我们如何利用它们？大数据将为我们解决一些最复杂的问题：

00:02:29:

如何管理交通、如何种植粮食、如何向需要的人提供物资，但这一切的前提是，我们必须弄清楚该怎么使用这些数据、以及该怎么在短得不能再短的时间内完成对它们的处理。

00:02:43 - Kenneth Cukier:

通过获取更多的数据，我们可以深入到这些子群体、这些细节，而这是我们以前从来没有过的方式。

00:02:53 - Saron Yitbarek:

The Economist
Kenneth Cukier 是《经济学人》的高级编辑，他也和我都在科技播客《Babbage》里。

00:03:01 - Kenneth Cukier:

这并不是说我们以前无法收集数据。我们可以，但这真的、真的很昂贵。真正的革命性突破是，我们使数据搜集变得十分容易。

00:03:10:

现在收集数据的成本极低，而且处理起来也超级简单，因为都是由电脑完成的。这已经成为我们这个时代巨大革命，它可能是现代生活最显著的特征，在未来几十年甚至下个世纪都会如此。这也是大数据如此重要的原因。

00:03:33 - Saron Yitbarek:

历史可以提醒我们这种变化是翻天覆地的。想想看，4000 年前，我们把所有的数据都刻在了干泥板上。

00:03:46 - Kenneth Cukier:

这些黏土盘很重。黏土盘被烤干后，刻在其中的数据就无法更改。从古至今，信息处理、存储、传输、创建的方式都发生了变化，对吗？

00:04:04 - Saron Yitbarek:

时代巨变。大约在 1450 年，印刷机的发明带来了第一次信息革命。今天，我们也迎来了一场革命。

00:04:16 - Kenneth Cukier:

现在的存储介质很轻巧。信息的修改变得极度简单，借助现有的处理器，我们只需要使用删除键就能修改我们所拥有的信息实例，无论那是储存在磁带上还是晶体管里。我们可以以光速传输数据，不用携带什么黏土盘。

00:04:37 - Saron Yitbarek:

在 15 世纪，借助印刷机，大量的数据得以传播。这些知识提升了人们对事物的认识，并促成了启蒙运动。

00:04:45:

今天，大数据可以再次提升我们的知识水平，但我们必须要想办法充分利用这些数据。唯有修好大坝、备好轮机，才能让浪潮为人所用。

00:05:00 - Kenneth Cukier:

当下，人们远没有做到对数据的充分利用。这一点非常重要，因为我们已经看到，数据中存在这种潜在的价值，而收集、存储和处理数据的成本在近百年来，乃至于近十年来，已经显著地降低了。

00:05:22 - Kenneth Cukier:

这很振奋人心。但问题是，我们在文化上、在我们的组织流程上，甚至我们的 CFO 和 CIO 们拨给相关方面的预算中，并不重视这种价值，

00:05:35 - Saron Yitbarek:

想着这种事肯定让人极度沮丧。启蒙运动在敲门，却无人应答。然而，我们不回答的部分原因是：门后到底是谁？这些数据能带来什么？

00:05:51:

Kenneth 认为，某些公司不采用大数据，乃是因为它太过于新奇。

00:05:56 - Kenneth Cukier:

一旦你收集了大量数据后，你能拿它干什么？我就直说吧，只有傻子才会以为自己知道。你绝对无法设想你今天收集到的数据明天能拿来做些什么用。

00:06:12:

最重要的是要有数据，并以开放的思想对待所有可以使用的方式。

00:06:18 - Saron Yitbarek:

如果我们按照 **Kenneth** 说的那样以正确的方式对待大数据，我们将会发现一切的全新可能性。这将是一个人 —— 而不只是数据科学家 —— 都能洞察趋势、分析细节的世界。

00:06:33 - Kenneth Cukier:

如果我们能意识到，这个世界是可以通过收集经验证据来理解、改变和改善的，并且可以用一种自动化的方式进行改善，我们将会得到看待它的全新角度。我个人认为，现如今，在世界各地，上至政策制定者、下至星巴克咖啡师，都在经历这种引人深思的文化上或心理上的变化。

00:07:00:

各行各业的人都有点数据基因，就像是被感染了似的。现在，无论他们专注于什么方面，他们都以大数据的方式思考。

00:07:15 - Saron Yitbarek:

Kenneth Cukier 给我们讲了一个简短的故事来展现这种新数据思维方式的力量。微软的一些研究人员开始着手研究胰腺癌问题。

00:07:27:

人们发现胰腺癌往往为时已晚，早期发现可以挽救生命。因此，研究人员开始询问这些患者，在开始搜索有关胰腺癌的信息之前几个月，他们搜索了什么？而早在发现前数年，他们又搜索了什么？

00:07:46:

研究人员开始寻找埋藏在所有搜索数据中的线索和模式。

00:07:54 - Kenneth Cukier:

他们有了重大发现。通过分析患者在最终开始搜索“胰腺癌”之前的这段时间中所搜索的关键词，他们识别出了一套规律，可以非常准确地预测搜索者是否患有胰腺癌。

00:08:09:

在这里，我们能学到一点：想象力与数据中潜在规律的结合，是可以挽救生命的。他们现在所要做的就是找到一种方法，通过方法来解释这一发现，这样当人们在搜索这些术语时，他们可以以一种微妙的方式干预，说，“你可能要去诊所检查一下。”

00:08:29:

像这样使用数据，就能救人于水火之中。

00:08:37 - Saron Yitbarek:

研究人员偶然发现的是一种新的癌症筛查方式，通过这种方法，患者可以提前一个月得知自己可能患癌。利用数据不仅仅是一个利润或效率最大化的问题。

00:08:52:

它的意义远不止于此。对于人类而言，这些数据中确确实实存在着大量的潜在利好。抗拒使用大数据可能只是自欺欺人。接下来，我们要关注的是，这场将数据投入工作的持久战。

00:09:18:

哈佛医学院的波士顿儿童医院去年完成了 26000 多台手术，进行约 25 万人次的儿童放射检查。

00:09:31:

医护人员的表现令人称道，但有一个巨大的障碍挡在他们面前。

00:09:37 - Ellen Grant:

在医院的环境中，尤其是作为医生，我们经常会遇到难以获取数据的问题。

00:09:45 - Saron Yitbarek:

这位是 Ellen Grant 医生，她是波士顿儿童医院的儿科神经放射科医生，她在诊疗时依靠访问数据和分析医学图像。

00:09:56 - Ellen Grant:

如果没有专门设置的环境，想要从 `packs` 里存储的图像进行额外的数据分析绝非易事。当你在一个只提供了普通的医院电脑的读片室里时，要做到这一点并不容易。

00:10:14:

获取数据实际上是有障碍的。

00:10:17 - Saron Yitbarek:

其实许多医院都会大量抛弃数据，因为存储它们的成本实在过于高昂。这部分数据就像这样丢失了。像 Grant 这样的放射科医生可能是第一批因为数据实在太多而感到沮丧的医务人员。

00:10:33:

当医院走向数字化后，他们开始创造大量的数据，很快，这个量就大到无法处理了。

00:10:41 - Ellen Grant:

我，作为一名临床医生，在读片室里的时候希望能将所有复杂的分析工作在研究环境中做完。但我无法随便就从 `packs` 中拿出来图像，拿到一些可以进行分析的地方，再拿回到我手里。

00:10:59 - Saron Yitbarek:

顺便说一句，`packs` 就是医院存储其图像的数据仓库。Grant 医生知道有一些工具可以让这些图像 `packs` 发挥更大的功能，但成本太高。

00:11:12 - Ellen Grant:

随着机器学习和 AI 时代的到来，数据的生产量将会日渐加大，我们会需要更多计算资源来进行这类大规模的数据库分析。

00:11:27 - Saron Yitbarek:

数据已经堆积如山了，但处理能力却没有相称的增长。在这一前提下，对数据的彻底处理将变得遥不可及。而复杂、昂贵的超级计算机并不是医院的选择。

00:11:41:

Grant 医生深感沮丧。

00:11:44 - Ellen Grant:

我们能不能想出一个更好的办法，让我把数据拿到这里来，分析一下，然后放回去，这样我就可以在会诊的时候，一边解释临床图像，一边把分析做完，因为我希望可以在会诊上展示数据，在此同时进行快速分析。

00:11:56:

我可不想在不同的电脑和存储器之间把这些数据挪来挪去，这不是我的工作。我的工作是理解非常复杂的医学疾病，并把相关的事实在脑子里。

00:12:10:

我想专注于我的技术领域，在此同时利用计算机领域的新兴技术；而不必这方面过于深入钻研。

00:12:21 - Saron Yitbarek:

Grant 和世界各地的放射科医生们需要的是一种方法，只要点击图像就能运行详细分析，并让这一切都发生在云端，这样医院就不必建立自己的服务器场地，也不必把医务人员变成程序员。

00:12:40:

他们需要一种方法来使他们的数据尽可能地拯救生命。这正是 Grant 医生和几位代码英雄决定去做的事。

00:12:55:

Grant 在波士顿儿童医院的团队正在与红帽和马萨诸塞州开放云（MOC）合作。关于 MOC 的更多内容稍后再说。首先，我们需要请出 Rudolph Pienaar，他是医院的一名生物化学工程师，来描述一下他们的解决方案。它是一个开源的、基于容器的成像平台。

00:13:15 - Saron Yitbarek:

它完全是在云端运行的，所以你不受医院本身计算能力的限制。他们称这一作品为 ChRIS。

00:13:24 - Rudolph Pienaar:

ChRIS 有一个后台数据库，其实就是一个 Django Python 机器。它可以跟踪用户，并跟踪这些用户使用过的数据以及分析结果。

00:13:35:

围绕这个数据库，有大量的服务群，这些服务都是作为自己的实例存在于容器中。它们处理与医院资源的通信，比如与医院数据库的通信。这些服务从资源中提取复杂的数据，将其推送给云端的、或者另一个实验室的、或者别的什么地方的其他服务处理。在计算数据的地方，有 Kubernetes 之类的编排服务，以及你需要使用的分析程序。数据处理结束之后，结果就会被发送回来。

00:14:11 - Saron Yitbarek:

对于 Grant 医生来说，ChRIS 成像平台是一种让数据活起来的方法。更重要的是，这种数据处理方式能让她成为更好的医生。

00:14:21 - Ellen Grant:

优秀的医生之所以优秀，是因为他们在一生中积累了丰富的从业经验。如果我们能把这一点融入到数据分析中，以此来获得更多的信息，我们就能知道得更多，并更有效地整合这些经验。

00:14:39:

例如，我对特定病患的特定受伤方式的认识，取决于我的从医经验和对这些经验的整体理解。

00:14:52:

现在，我可以根据真实数据创建受伤症状分布的概率图，并将其公之于众；我也可以寻找有相似模式的患者，并告诉他们在接受治疗时，什么对他们最有效，以便更接近精准医疗。

00:15:10:

整合大量的数据，尝试探索我们过去的知识，并尽你所能，点明治疗病人的最佳方式。

00:15:21 - Saron Yitbarek:

这对被送到医院的孩子意味着什么？Grant 医生说，ChRIS 平台能提供更有针对性的诊断和更个性化的护理。

00:15:31 - Ellen Grant:

如果我们拥有更复杂的数据库，我们就能更好地理解信息之间繁杂的相互作用，因此就能更好地指导每个患者。我认为 ChRIS 就像是我进入超级大脑的接口，它能让我比平时更聪明，因为我不能一次把所有数据保存在我的大脑中。

00:15:53 - Saron Yitbarek:

当赌注如此沉重时，我们要突破人类大脑的极限。这位是 Máirín Duffy。她是红帽团队中的设计师，她让 ChRIS 成为现实，而根据个人经验，她知道这件事其中的风险。

00:16:15 - Máirín Duffy:

我父亲中风了，所以我一直作为病人家属等待医疗技术诊断，因为当一个人中风并被送到医院之后，医务人员必须弄清楚是哪种类型的中风。根据中风类型，有不同的治疗方法。

00:16:31:

如果使用了错误的治疗方案，就可能发生极其糟糕的事。所以，在这种情况下，你能越快地把病人送来作核磁共振，就能越快地得到治疗方案。速度越快就有可能挽救他们的生命。

00:16:43:

想想看，仅仅是把图像处理从云端推送出来，并行化处理，就能让它快很多。这样就能将这个过程从几小时、几天，缩短到几分钟。

00:16:55 - Saron Yitbarek:

医学可能正迎来一个新的拐点。一个不是由药理学驱动，而是由计算机科学驱动的拐点。另外，想想像 ChRIS 这种东西的拓展性。

00:17:08:

发展中国家的医生也可以受益于波士顿儿童医院的专业知识和数据集。任何有手机服务的人都可以通过网络访问能够拯救生命的数据和计算结果。

00:17:24:

除了医学，很多其他领域也可能出现类似的拐点。但前提是，人们得知道如何从自己的数据中找到隐藏信息。为了做到这一点，他们需要探索一个全新的计算领域。

00:17:46:

世界各地的人们都在学习如何利用数据。就像在波士顿儿童医院一样，将数据洪流导向目标。

00:17:56:

换句话说，我们在处理这些数据。但我们之所以能做到这一点，是因为新一代的云计算使之成为可能。

00:18:11:

对于像 ChRIS 这样的平台来说，一个关键因素是基于云计算的新型存储方式。请记住，很多医院都会把收集到的数据扔掉，因为他们根本无法容纳所有数据。

00:18:25:

这就是我想重点讨论的数据泛滥的最后一块拼图：存储解决方案。对于 ChRIS 来说，存储解决方案是一个叫 Ceph 的开源项目。它使用的马萨诸塞州开放云，就基于 Ceph。

00:18:45:

我和 Ceph 的创建者 Sage Weil 聊了聊，想了解更多关于像波士顿儿童医院这样的地方是如何在闪电般的时间内处理海量数据的。以下是我与 Sage 的对话。我认为，第一个重要问题是，什么是 Ceph，它能做什么？

00:19:05 - Sage Weil:

当然，Ceph 是一个由软件定义的存储系统，它允许你提供可靠的存储服务，并在不可靠的硬件上提供各种协议。

00:19:14:

它的设计从开始就是满足可扩展性，所以你可以拥有非常非常大的存储系统、非常大的数据集。于此同时，系统对硬件故障和网络故障有优秀的容忍性，所以即使出现了一些这类问题，存储中的数据仍然不会变得难于访问。

00:19:29 - Saron Yitbarek:

现在，数据太多了。

00:19:31 - Sage Weil:

是的。

00:19:33 - Saron Yitbarek:

如此大的工作量。要处理的东西实在是太多了。你认为这个解决方案出现得是时候吗？

00:19:39 - Sage Weil:

是的，肯定是这样。在当时，行业中这方面的严重不足是显而易见的。没有开源的解决方案可以解决可扩展的存储问题。所以，我们显然得造个轮子。

00:19:53 - Saron Yitbarek:

考虑到我们每天要处理的数据量，以及它将来只会越来越多、越来越难管理的事实，你认为当今该怎么做才能解决这种日益增长的需求？

00:20:09 - Sage Weil:

我认为有几方面。一方面，有令人难以置信的数据量正在产生，所以你需要可扩展的系统。它不仅可以在硬件和数据规模上进行扩展，而且，它的管理成本应该是一样的，至少应该基本固定。

00:20:25 - Saron Yitbarek:

嗯。

00:20:26 - Sage Weil:

你不会想就为每多 10PB 存储空间或类似的东西就多雇一个员工吧？我认为这套系统在运维上也必须可扩展。

00:20:33 - Saron Yitbarek:

是的。

00:20:35 - Sage Weil:

这是其中的一部分。我认为，人们利用存储空间的方式也在改变。一开始，都是文件存储，然后我们有了虚拟机的块存储，我觉得对象存储在某种程度上是行业的重要趋势。

00:20:51:

我认为，下一个阶段的目标并不局限于提供一个对象存储端点，并将数据存储在集群中；我们需要将解决方案进一步升级，好让它能管理集群的集群，抑或是对分布于不同地理位置的云空间及私有数据中心储存空间中的数据进行管理。

00:21:13:

比如说，你现在将数据写入一个位置，随着时间的推移，你可能会想将数据分层到其他位置，因为它更便宜、或者服务器离你更近；或者，一旦数据太老、不会频繁使用了，你就需要将其移动到性能更低、容量更大的层次上，以保证存储的成本较低。

00:21:27:

你可能也会为了遵循地方法规而移动数据。在欧洲的一些地区接收数据时，数据来源必须保持在特定的政治边界内。

00:21:39:

在某些行业，像 HIPAA 这样的东西限制了数据的移动方式。我认为，随着现代 IT 组织越来越多地分布在不同的数据中心、公有和私有云中，统一地、自动化地管理它们的能力正变得越加重要。

00:21:58 - Saron Yitbarek:

当你想到未来我们要如何管理和存储数据，以及如何处理数据的时候，开源在其中扮演了怎样的角色？你曾提到，你之所以要创建一个开源的解决方案，是因为你个人的理念和你对自由和开源软件的强烈感情。

00:22:16:

你如何看待开源对未来其他解决方案的影响？

00:22:21 - Sage Weil:

我认为，特别是在基础设施领域，解决方案正在向开源靠拢。我认为原因是基础设施领域的成本压力很大，特别是对于构建软件即服务（SaaS）或云服务的人来说，低成本的基础设施很重要，从他们的角度来看，开源显然是一个非常好的方法。

00:22:48:

第二个原因更多地是社会因素，在这个快速发展的领域里有如此多新的工具、新的框架、新的协议、新的数据思维方式，这个领域中有这么多创新和变化，有这么多不同的产品和项目在相互作用，所以很难以传统方式做到这一点，比如说，让不同的公司互相签订合作协议，共同开发。

00:23:20:

开源可以消除此事上的所有阻力。

00:23:28 - Saron Yitbarek:

Sage Weil 是红帽公司的高级咨询工程师，也是 Ceph 项目的负责人。我要绕回到《经济学人》的 Kenneth Cukier，以从一个更整体的视角上进行讨论，因为我希望，我们能够记住他关于人与数据之间关系的看法，以及我们从泥板，到印刷机，再到像 Sage 打造的云端奇迹的进步历程。

00:23:55 - Kenneth Cukier:

这关乎人类的进步，关乎我们如何更好地理解世界，如何从现实中总结经验，以及如何改善世界。这进步也是人类一直以来的使命。

00:24:08 - Saron Yitbarek:

使命永无止境。但是，与此同时，学会处理我们收集到的数据并将其投入使用，是 Oak Ridge National Laboratory 整整一代人的开源任务。我们将在田纳西州的橡树岭国家实验室短暂停留，结束我们的数据之旅。它是世界上最快的超级计算机 Summit 的所在地，至少在 2018 年是最快的超级计算机。

00:24:43:

这台机器每秒能处理 20 万亿次计算。换个计量单位，就是 200 petaflops。这样的处理速度，对于医院、银行或者今天所有受益于高性能计算的成千上万的组织来说并不现实。

00:25:04:

像 **Summit** 这样的超级计算机更多的是留给强子对撞机的领域。不过话说回来，我们曾经在泥板上记录的只是 100 字节的信息。

00:25:16:

在数据存储和数据处理的领域中，非凡的壮举不断成为新的常态。有一天，我们或许能将 **Summit** 级别的超级计算机装进口袋。想一想，到时候我们能够搜索到的答案。

00:25:42:

下一集，我们聊聊无服务器。第 7 集将会讲述我们与基于云的开发之间不断发展的关系。我们将会探究，在我们的工作中有多少可以抽象化的部分，以及在这个过程可能会失去的东西。

00:25:58 - Saron Yitbarek:

同时，如果你想深入了 **ChRIS** 的故事，请访问 redhat.com/chris，了解它是如何构建的，以及如何为项目本身做出贡献。

00:26:12 - Saron Yitbarek:

《代码英雄》是一款红帽公司原创的播客。你可以在 Apple Podcast、Google Podcast 或任何你想做的事情上免费收听。

00:26:24 - Saron Yitbarek:

我是 **Saron Yitbarek**。坚持编程，下期再见。

什么是 LCTT SIG 和 LCTT LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 **LCRH SIG** 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-2/the-data-explosion>

作者: Red Hat 选题: bestony 译者: TimeBear 校对: Northurland, wxy

本文由 **LCRH** 原创编译, **Linux中国** 荣誉推出

《代码英雄》第二季（7）：无服务器

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频博客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《[代码英雄](#)》系列播客第二季（7）：[无服务器](#)的音频脚本。

导语：无服务器 Serverless</rt>到底意味着什么？当然，总得有服务器存在——构建网络的基本架构不会改变。不过在将服务器交给一小部分人运维之后，开发者们会发生什么变化呢？

无服务器编程让初学者们可以更加轻松简单地部署自己的应用程序，让工作更有效率，这是它的优点。Andrea Passwater 跟我们分享了抽象底层架构会给我们带来多大的便利。不过凡事必有代价，无服务器化也有很多问题。

Rodric Rabbah 解释了无服务器意味着你将部署和回应问题的能力拱手献出——这就是为什么他帮助创建了 Apache OpenWhisk，这是一个开源的无服务环境框架，同时 Himanshu Pant 也来分享了他对于何时应该使用无服务器服务的观点。

empowerment

无服务器编程应该是为开发者们赋能授权的。我们也应该对于全局场景保持关注——尽管我们精简了我们的工具箱。

00:00:03 - Al Gore 档案：

现如今，当然了，在全美乃至全世界，互联网正在彻头彻尾地改变着我们的生活。

00:00:13 - Saron Yitbarek:

那是 1998 年。Google 才刚刚雇佣了它的第一名员工，副总统 Al Gore 在接受媒体采访。

00:00:22 - Al Gore 档案：

Bill Clinton

这项技术还处于起步阶段。当我和比尔·克林顿总统入主白宫时，只有 50 个网站。现在看看，我在生日那天收到了一束虚拟鲜花。

00:00:37 - Saron Yitbarek:

好的。我已经感觉到你的眉毛皱起了。为什么我要向你展现某些 20 年前的互联网史？这是因为我想提醒你，互联网的基础仍然是相同的。

00:00:51:

当然，现在有不止 50 个站点了，我知道。但是，我们仍然在发送虚拟鲜花。从开发人员的角度来看，如果你将我们所有的惊人进步剥离开来，你得到的仍然是相同 **client-server** 的“客户端 - 服务器”（C/S）模型，这就是一切的开始。一个提供了分布式网络的客户端 - 服务器模型。

00:01:16:

Serverless

如今，开发人员谈论了很多有关无服务器的问题，这听起来像是 Al Gore 谈到的客户端 - 服务器模型被废弃了。而且，如果我们不小心，我们能够抽象出太多的基础架构，以至于忘记了仍然有服务器在那里做着它们的工作。

00:01:37:

但是，无服务器真的意味着没有服务器吗？真的吗？还是开发人员与服务器之间的关系正在变化？在这一集中，我们将与来自世界各地的人们交谈，以探索这种被称为“无服务器”的东西。

00:01:54:

我是 Saron Yitbarek，这里是《代码英雄》，一档来自红帽的原创播客节目。

00:02:03 - Andrea Passwater:

你知道无线网络在某些地方还有线缆吗？

00:02:06 - Saron Yitbarek:

Andrea Passwater 在一家名为 嗯..... “无服务器” 的公司工作。他们创建了一款流行的开源框架来开发无服务器应用程序。Andrea 注意到了各个组织是多么渴望抽象化基础架构的方法，而这正是神奇的“无服务器”一词始终给予人希望的地方。

00:02:28 - Andrea Passwater:

我认为这一术语主要是为了传达这样一个事实，即如果你是从事无服务器应用方面工作的开发人员，你不必考虑那些服务器。你只需要写代码并将代码部署到云提供商即可，而不必担心管理。这就是无服务器的真正含义。

00:02:49 - Saron Yitbarek:

对于 Andrea 来说，无服务器的吸引力很明显。

00:02:53 - Andrea Passwater:

倘若你以无服务器的方式开发应用程序，则可以不必去考虑部署和维护该应用程序的日常工作。这意味着你可以专注于它的商业价值，你可以专注于发挥创造力。

00:03:12 - Saron Yitbarek:

而无服务器的另一大好处是，你不太可能发现自己在重复造轮子。

00:03:18 - Andrea Passwater:

当有像 Auth0 这样可以直接使用的服务存在时，为什么要创建自己的身份验证方法呢？归根结底，无服务器就是为开发人员提供机会，使得他们能够更加轻松快速地构建起能把他们脑子里的所有的主意带到世界上的程序。

00:03:41 - Saron Yitbarek:

我明白了！

00:02:27:

想象一下，你拿了满手的东西，正跌跌撞撞地走向一扇门。这扇门滑开了，以简单、友好.....

00:03:50:

(让我来说)

00:03:51:

.....的方式。这就是无服务器。它为你打开了大门，使得开发工作不再那么繁琐。事实上，随着各个组织趋向于混合云环境，以及无服务器运动的发展，开发的障碍正在消失。

00:04:09:

Andrea 听说过很多有关非开发人员进行开发的话题。

00:04:15 - Andrea Passwater:

这是传统上认为自己写不了代码，而现如今由于无服务器而得以投身于软件工程的人的故事，并且能够开发这些使得他们自己的工作流程和类似的东西自动化的工具。这与你做什么工作都没关系。

00:04:31:

你在工作中要做的一些事情是如此的呆板无聊，比如你每天都在做的例行事情，和那些你会想“难道计算机不能为我做这件事吗？”的事情。我开始有了这种感觉的时候，我碰巧在一家名为“无服务器”的公司工作，他们像这样：“你意识到我们制作的产品可以为你提供帮助，对吗？”

00:04:50 - Saron Yitbarek:

Andrea 认为，不久之后，许多从未将自己视为开发人员的人将意识到他们能够自己构建简单的应用程序，基本上免费。

00:05:02 - Andrea Passwater:

借助 Lambda，我从不需要为自己制作的任何小型应用程序付费。我可以让这些机器人为我做一部分工作，是的，我可以提高工作效率。但是，我也不必再做这些无聊的工作了。我可以做些更有趣的事情。

00:05:17 - Saron Yitbarek:

即使是对于专业开发人员来说，这种自动门效果在满手杂物的世界里也是很诱人的。

00:05:25 - Andrea Passwater:

我认为人们对于能够让一两个人的团队，在短时间内就让原型工作起来很感兴趣。在几天时间内，他们就可以启动并运行原型。我认为这使得人们开始意识到，他们可以专注于驱动他们的应用、产品和公司中的商业价值的部分。这非常让人兴奋，他们可以专注于商业价值。

00:05:54 - Saron Yitbarek:

Functions-as-a-service

我要再抛出一个术语给你。准备好了吗？功 能 即 服 务 (FaaS)。就像是 AWS Lambda 或 Apache OpenWhisk 之类的无服务器产品。“功能即服务”意味着，只有在被触发时程序才会执行某个功能，这效率更高。

00:06:15:

此外，这让我对计算能力和运行时间的担心少了很多。最终，无服务器可能会成为一个相当不错的基础配置。事实上，有些人甚至开始怀疑，我们是否要完全使用无服务器？它可以替代容器吗？

00:06:34 - Michael Hausenblas:

我理解这这种想法。

00:06:35 - Saron Yitbarek:

Michael Hausenblas 是 Red Hat OpenShift 团队的开发倡导者。

00:06:41 - Michael Hausenblas:

如果你看一下我们现在拥有的这些东西，包括 OpenShift 和 Cloud Foundry 和一些其他东西，你实质上就拥有了抽象化。基本上，Heroku 或多或少地倾向于向这个想法。对吗？这是非常简单的方式，无需担心程序会如何运行，无需担心它是什么样的。只需要给我们代码，我们来处理剩下的工作。

00:07:03 - Saron Yitbarek:

no-ops

是的。听起来相当不错。这听起来有点儿像是梦想中的“无运维”环境，一切都自动化并且抽象化了，就像是开发者版本的极简主义室内设计。很棒、很干净的界面。

00:07:21:

但是，Michael 想要让你了解你一些现实情况。

00:07:25 - Michael Hausenblas:

没有运维！是吗？你知道，它只是神奇地以某种方式消失。你可以在 HackerNews 和 Twitter 以及其他任何地方看到这些笑话。无服务器？当然有服务器！我知道，当然有。而且也肯定有运维。

00:07:39:

总得有人去做这些，总得有人去架设服务器、给操作系统打补丁、去创建容器镜像，因为，你猜猜这些功能会在哪里执行？当然是在某种计算机上。

00:07:54 - Saron Yitbarek:

这不是零和博弈。功能即服务无法直接取代容器，而是在工具箱中增加了一个工具。我还有更多的事情要告诉你。通过使用这种新工具，转变成无服务器并不只是意味着运维就是其他人的事情，你仍然需要自己考虑自己的运维。

00:08:14 - Michael Hausenblas:

你会看到在基础架构侧有一点运维工作。但是，也有一点是开发人员的事情。如果你处在一个极端情况之下，比如说使用 Lambda，那么你是没有任何任何类型的管理员权限的，对吧？

00:08:29:

你不能简单地致电或是短信给一名基础架构管理员。显然，你组织之中的某一个人就必须得做这件事。但是，我担心许多组织只看到了它是如此简单而便宜。我们无需动这个，还有这个、这个，然后忘记了谁在待命，谁是真正地在待命？你对此有什么策略吗？

00:08:52:

如果没有的话，那么你可能会想要在进行之前，先制定一个策略。

00:09:00 - Saron Yitbarek:

需要有人处于待命状态。即使选择了“无服务器”，你仍然需要在头脑中萦绕更大的场景，你仍然需要让你的运维有序进行。

00:09:24:

在我早先时候抛出那个“功能即服务”术语时，你是不是有过些许畏缩？过去，基于云的开发为我们带来了大量的“xx 即服务”的术语。我们有基础架构即服务（IaaS）、平台即服务（PaaS）、软件即服务（SaaS）、数据即服务（DaaS）、数据库即服务（DBaaS）……等等。

00:09:48:

如果你难以理解它们的不同，那你并不孤单。这就是我们找来了 **Himanshu Pant** 的原因。他是位于印度德里的苏格兰皇家银行的技术主管。他花了多年时间来分析其中的差异。

00:10:04 - Himanshu Pant:

这些其他的计算范例在名称上和无服务器听起来是如此的相似，以至于人们往往会忘记，或者困惑于为什么没有将它们称之为无服务器，或者为什么这个被称为无服务器。

00:10:20 - Saron Yitbarek:

因此，无服务器与容器不同，无服务器也不是平台即服务。但是 **Himanshu** 希望将其明确一下。功能即服务能够提供什么？又不能提供什么？

00:10:35:

他与我们分享了两件轶事，有两次他弄清楚了什么时候该使用无服务器，什么时候应该放弃。第一次来自一个 24 小时黑客马拉松。**Himanshu** 当时正试图开发一个聊天机器人。

00:10:49 - Himanshu Pant:

有各种各样的指标会影响它的选择。例如逻辑覆盖率、可能产生的成本以及可伸缩性。而我选择在无服务器环境下完成这项工作。

00:11:04:

当我在开发它的时候，我意识到成本是一个层面，而这确实是我所青睐的技能。因此，即使其他所有的参与者都有更好的……我想说的是，覆盖率，或者说是逻辑覆盖率。这里讲的是 NLP 语境或其场景。

00:11:19:

但是，就成本和可伸缩性而言，我是手操胜券的，因为借助无服务器，这完全取决于人们在该聊天机器人上所进行调用的次数，并相应的触发该功能。这是一个我十分乐意采用无服务器的用例，因为成本——没有成本。以及更快的开发时间，而且老实说，当时还并不完全是生产规模级别的工作负载。

00:11:45:

我可以使用平台上的某些新兴工具。这对我而言是一次胜利。

00:11:52 - Saron Yitbarek:

很好。那时无服务器才有了意义。但是，在 **Himanshu** 目前供职的银行里，人们正在将他们的系统从旧版迁移到云端。而这提出了不同的目标。

00:12:07 - Himanshu Pant:

我们正在尝试查看哪些工作负载适用于哪些范例。比如 IaaS、BaaS、FaaS，这显然是属于企业领域的。你要看到这些方面，比如说第一，可靠的供应商难以寻找，以及第二，该技术应该得到广泛的验证。这对于像是银行业这样的规避风险的行业而言更是如此。

00:12:30:

这就是平台即服务（PaaS），但是仍然需要更好的证明、更好的功能，以及它们比传统工具更优越的地方。

00:12:40 - Saron Yitbarek:

Himanshu 正在研究自己的需求以及他自己的舒适区，并且研究哪些工作负载在哪种云计算规范中是有意义的。

00:12:49 - Himanshu Pant:

假设某个听众在一家贸易商店工作，他想构建某种东西，比如说一个入口。对于他或者她来说，无服务器可能并不是真正合适的选择，因为在那种在特定机器的应用程序中，延迟可能是不该出现的。

00:13:05 - Saron Yitbarek:

归根结底，这是一种有节制的做法，而不是将所有东西都丢进一个桶里。当我们思索哪一种基于云的架构是真正我们所想要做的工作时，还有一件事情需要考虑。所有这些抽象的东西，所有解放你双手的东西，最终如何改变的不仅仅是我们的工作方式，还改变了完成工作本身。

00:13:31:

抽象掉一部分工作负载可能意味着更少的自定义选项。想象一下你购买的一辆车。它能工作，它能开。但是接着想象一下你自己组装的一辆车，这辆车会按照你决定的方式工作。

00:13:48 - Rania Khalaf:

这是有代价的。

00:13:50 - Saron Yitbarek:

Rania Khalaf 是 IBM 研究部门的 AI 工程总监。

00:13:56 - Rania Khalaf:

在使用这些无服务器应用程序的过程中，你可能无法完全控制所有正在发生的事情。你无法控制全盘计划，或是程序何时何地运行。

00:14:06 - Saron Yitbarek:

这是一种权衡，对吗？当你使用无服务器时，细粒度控制可能会失误。

00:14:13 - Rania Khalaf:

它对于终端用户而言，抽象化了如此之多的东西，以至于你想要拥有更多的控制权、不同的规划、更多的检查与平衡、功能可以运行多长时间的不同值，等等等等。那么，如果你真的希望能够进入系统并修补，也许你可以创建你自己的部署环境。

00:14:32 - Saron Yitbarek:

不过，这将需要一些新东西，一类新的无服务器环境，开源社区已经在为自己打造了它。Rania 和她的 IBM 团队参与了该运动。

00:14:44 - Rania Khalaf:

我们首先研究是一种语言……它基本上是 JavaScript 的扩展，可以让你创建这些多线程交互服务的组合，以此作为起点，为你提供一种更加轻量级服务的方式。大约在同一时间，云和微服务以及平台即服务开始真正兴起。

00:15:08:

仅仅是将这两种趋势结合起来，就可以用可能来自于你，也可能来自其他人的许多小部件，构建更加高阶的功能。

00:15:18 - Saron Yitbarek:

Rania 和她的团队正在构建 Apache OpenWhisk，一款开源的功能平台。

00:15:23 - Rania Khalaf:

对于 OpenWhisk，我们从一开始就开源了。其中很大的原因是，为了让社区和我们一起参与进来。但是同时也是为了揭掉外包装，将控制权交给想要运行自己的无服务器计算环境的人们，以便他们能够根据自己的需求对其进行自定义，也许将它们置身于自己的控制之中，看看它实际上是如何运行的，以对其进行更好的控制。

00:15:54:

而且，我们还可以提供更加精细的控制，如果仅仅是普通服务，人们就不会有这种控制。

00:16:03 - Saron Yitbarek:

将控制权交还给想要运行自己的无服务器运行环境的人。这是下一阶段的无服务器。加入 OpenWhisk，你将获得像是 Fission 和 Gestalt 之类的其它开源平台。我们开始看到无服务器领域正在演变得比原先更具适应性，而且功能更为强大。

00:16:31:

为了真正了解为什么开源版的无服务器很重要，我与 OpenWhisk 的创始人之一进行了谈话。

00:16:39:

嗨，Rodric。最近好吗？

00:16:40 - Rodric Rabbah:

很好。你好吗？谢谢你邀请我参与节目。

00:16:42 - Saron Yitbarek:

Rodric Rabbah 是构思并创立 OpenWhisk 的三位开发者之一。以下是我们的谈话。

00:16:54 - Rodric Rabbah:

别人可能会很困惑，也可能会窃笑，因为人们可能会想：“倘若没有服务器，你要怎么做计算呢？”

00:17:02 - Saron Yitbarek:

是的。服务器就在某处，只是我不必去费心考虑它。

00:17:05 - Rodric Rabbah:

完全正确。这就是这个模式的真正美妙之处。当你开始以无服务器的方式进行开发时，你就再也不想回到过去了。你知道的，如今我已经置身其中接近 4 年了，并且已经开发了一些达到生产质量的应用程序。

00:17:19:

这是我如今惟一的开发方式。如果你告诉我必须要配置一台计算机并且安装操作系统，这对我而言完完全全是陌生的。我甚至都不确定我是不是还知道该怎么做。

00:17:29 - Saron Yitbarek:

是的。当你这样说的时候，听起来像是减轻了很大的负担。你知道吗？当最初听说无服务器时，至少我会想：“伙计，我必须要去学习的事又多了一件。”

00:17:38:

但是，当你这样说的时候，听起来不错。

00:17:41 - Rodric Rabbah:

这确实听起来很棒。然而，你应该已经意识到你必须要从这幻想的气泡中抽出一点儿空气。它不是万能药。

00:17:50 - Saron Yitbarek:

有哪些令人惊讶的风险或问题是人们在起步时可能没有看到或意识到的呢？

00:17:58 - Rodric Rabbah:

我认为缺乏透明度可能是最大的问题。这有点儿让我想起了新语言问世时出现的，那些提高了计算机抽象水平的技术。在当今的无服务器环境中，这是一种类似的令人震惊的效果。

00:18:16:

在这个过程中，你通常会写一个功能，然后只需部署这个功能即可。它可以立即运行，比如在 web 上作为 API 点。它可以大规模伸缩。我的意思是你无需自己做任何工作即可运行数千个实例。

00:18:32:

但是，倘若哪里出了问题，那应该如何调试呢？或者我实际上是想要检查我的功能失败的上下文环境。通常，这些功能在进程内运行，与你隔离——你甚至无法登录计算机查看你的代码在何处运行。它们可能在封闭的容器环境之中运行，你不知道里面有什么。

00:18:53:

获得一点儿透明度对你而言变得很困难。这是工具最终将提供帮助的地方。但是，工具的缺乏某种程度上会让人们陷入一个相当大的陷阱。

00:19:05 - Saron Yitbarek:

这真的很好。那么让我们回到 OpenWhisk。请给我说说关于它的事情。

00:19:11 - Rodric Rabbah:

该项目在 Amazon Lambda 宣布推出产品的那一刻就开始了，这实际上是无服务器开始成为术语，并且开始在该领域获得关注的时刻。当我们看到 Lambda 时，我们开始思索：“这里有许多技术需要开发。不仅仅是在新的云计算的基础层上，而且在置于其上的编程模型之上，这实际上都使得它更易于被程序员访问。”你知道，由于出自 IBM 研究所，我们拥有相当强大的技术，不只是编程语言设计、编译器专业知识以及运行时的专业知识的技能。

00:19:54:

我们的一个小团队，基本上三个人……

00:19:57 - Saron Yitbarek:

哇。

00:19:57 - Rodric Rabbah

……聚集在一起，做了最初的开发和原型，最终成为 OpenWhisk，带有命令行工具，这是现如今无服务器实际上的编程接口。编程模型概念，然后是它必须支持的实际架构，本质上，是服务器模型的这个功能，提供了无服务器所支持的所有好处。

00:20:22:

请注意，真正的起源是 Amazon Lambda 的出现，并可以说这是一种新的计算模型。

00:20:28 - Saron Yitbarek:

那么花了多长时间？或者说是第一个版本什么时候出现的。

00:20:30 - Rodric Rabbah:

实际上很快。事实上，当 IBM 宣布……好吧，那时还是 IBM OpenWhisk。从我们第一次提交到现在才一年。

00:20:39 - Saron Yitbarek:

哇。我的天哪。

00:20:41 - Rodric Rabbah:

这着实令人激动。

00:20:43 - Saron Yitbarek:

这确实很令人印象深刻。事实上，当它第一次启动时，它不叫 OpenWhisk，而是 Whisk。对吗？

00:20:49 - Rodric Rabbah:

Whisk 是内部名称，没错。我取的这个名字。这个名字背后的意思是迅速而又灵活地行动。

00:21:00 - Saron Yitbarek:

很好。

00:21:01 - Rodric Rabbah:

你“搅拌”了一个功能，就可以了。你可以将其放入烤箱中烘焙。

00:21:07 - Saron Yitbarek:

太好了，因为当我看到它，我肯定想的是鸡蛋。我在想，让我们“搅拌”一些鸡蛋。
whisk

00:21:12 - Rodric Rabbah:

对。我们对该名称进行了一些正面和负面的评价。当我们开源一项技术，并将其放到 GitHub 上时，我们会在上面加上 `open` 前缀，以强调该技术与开源一样开放，可以自由使用、自由下载、自由贡献。

00:21:32 - Saron Yitbarek:

是的。

00:21:33 - Rodric Rabbah:

我们将其开源的目的，实际上是一定程度上提升可以在当今无服务器平台上执行标准。对我们来说，重要的是要建立一个平台，不仅可以用于生产环境，还可以与全世界共享，而且还可用于学术研究或一般性研究。也许因为出自 IBM 的研究机构，我们有点儿在意这个。

00:22:00:

但是，这是有所回报的，我知道一些大学——从 Princeton 到 Cornell——在他们的研究中使用 OpenWhisk。我去过 Brown、Williams College、MIT、CMU 等几所大学，并且进行了一些讲座，目的是鼓励学生真正地去研究围绕无服务器以及服务器功能的问题、工具、编程模型，并且使他们对技术感兴趣。

00:22:26:

向他们显示，如果他们真的为开源项目做出了贡献，那么有一条路可以让 IBM 的云功能接受它并在生产环境中运行，通常在一周之内。

00:22:34 - Saron Yitbarek:

哇。这么快。

00:22:36 - Rodric Rabbah:

这让一些人感到惊讶。

00:22:38 - Saron Yitbarek:

这是一个非常高效的过程。

00:22:41 - Rodric Rabbah:

这确实证明了我们是如何在开源环境下开发许许多多技术的。这不是一个开放核心模式，有些组件有所保留。实际上在 IBM 云之中所运行的就是 Apache OpenWhisk 项目。

00:22:56 - Saron Yitbarek:

当你思索无服务器的未来，以及我们所选择的前进道路时，你觉得它们将是不可避免地奔向开源吗？

00:23:08:

我认为最近对于开源的价值存在一场激烈的争议，尤其是在云计算领域。

00:23:13 - Saron Yitbarek:

是的，没错。

00:23:15 - Rodric Rabbah:

如果你在思考为什么人们会转向云计算，或者为什么他们可能会厌恶投身于云计算领域，这就是供应商锁定的问题……丧失透明度。开源在一定程度上缓解这些问题，发挥了重要的作用。然后再看看类似 **Kubernetes** 的服务，它只是在容器和系统管理方面吞噬了云，就那么的成功！

00:23:41 - Rodric Rabbah:

如果你正在做的事情接触到了容器，鉴于它的主导地位，保持闭源与否还值得讨论吗？我倾向于认为开放性是有帮助的，从开发人员的角度来看，这很有吸引力。

00:23:57 - Saron Yitbarek:

当你考虑未来的无服务器生态及其工具、项目以及服务时，你看到了什么呢？对你来说，无服务器的未来是什么样的？

00:24:08 - Rodric Rabbah:

我认为，你会开始越来越少思考底层技术，而变得越来越多地考虑编程体验以及围绕它的工具：用于调试的、用于部署管理的、用于性能分析的、用于安全性。

00:24:26:

我认为，所有这些都非常重要。你如何运行你的功能的底层机制——无论它们是在容器中还是在一些未来的技术下运行，也无论你是将它们运行在一个云上或是多个云上——我认为，它们都不重要。有点儿像是 **Kubernetes** 在容器以及容器管理方面所做的事情。

00:24:46:

类似的还有一层要放在上面，就是服务分层的功能，可以实现那种无服务器概念。然后，它实际上的表现取决于你在其中安放的中间件。你如何赋能授权给开发者，让他们真正利用这款新的云端计算机，以及你要在它周围投入的辛劳，让他们的体验愉快。

00:25:07 - Saron Yitbarek:

是的。这种赋能授权看起来怎么样？

00:25:13 - Rodric Rabbah:

一言以蔽之，就是高效。这是一种能力，可以只专注于对于我，作为一名开发人员而言有价值的东西，或者如果我在公司工作的话，对于我公司的价值。这样能够更快地规避问题，因为你释放了你的脑细胞，而不用去考虑基础设施和如何伸缩，以及在硬件层面如何保障事物的安全性。

00:25:38:

现在，你可以真正地创新，将脑力重新投入到更快的创新中去，从而为你的终端用户带来更多的价值。我想把这一切都归结于更高的效率。

00:25:55 - Saron Yitbarek:

Rodric Rabbah 是 OpenWhisk 的一位创始人。还记得我在这一期节目开始的时候所说的吗？互联网所基于的那种老旧的客户端 - 服务器模型并不会消失。改变的是，我是说彻底改变的是我们对服务器的思考方式。

00:26:19:

在所谓的无服务器世界之中，希望我们专注于代码本身，而不用担心基础架构。但是，我们所选择的抽象等级，以及如何保持对于未被抽象的工作的控制，才是使得无服务器世界变得真正有趣的地方。

00:26:40:

无服务器最终应当是为开发人员赋能授权的。免于打补丁、进行扩展和管理基础设施。但是，与此同时，即使我们抽象化了一些任务，依然必须对整体如何工作保持关注。我们会问，我要放弃的是哪些控制权，而我要收回的又是哪些控制权？

00:27:07:

下一集是我们史诗般的第二季的大结局，《代码英雄》将要前往火星。我们将会了解 NASA 的火星探测器如何开始自己的开源革命。并且我们将与 NASA 喷气推进实验室的 CTO 进行交流，了解开源是如何塑造太空探索的未来的。

00:27:39 - Saron Yitbarek:

与此同时，如果你想要更加深入地研究无服务器开发的问题，或是在这一季里我们所探索过的任何主题，请访问 redhat.com/commandlineheroes，查看免费资源。当你在那里时，你甚至可以为我们自己的代码英雄游戏作出贡献。

00:28:00 - Saron Yitbarek:

我是 Saron Yitbarek。感谢收听，编程不已。

什么是 LCTT SIG 和 LCTT LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 LCRH SIG 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-2/at-your-serverless>

作者: Red Hat 选题: bestony 译者: JonnieWayy 校对: acyanbird, wxy

本文由 LCRH 原创编译，Linux中国 荣誉推出

《代码英雄》第二季（8）：开源好奇号

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频播客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《代码英雄》系列播客第二季（8）：开源好奇号的音频脚本。

导语：那些最棒的、最聪明的人用袖珍计算器的计算能力把我们带到了月球。现在他们要带我们走得更远——而且是用我们整季播客都在谈论的技术来实现。开源将带我们去火星。

第二季的结局是将我们带到美国宇航局（NASA）的 Jet Propulsion Laboratory 喷气推进实验室（JPL）。Tom Soderstrom 分享了 JPL 通过拥抱开源获得了多少好处。Hila Lifshitz-Assaf 解释说，NASA 正在用开源软件和众包解决一些最大的问题。Dan Wachspress 介绍了与 NASA 合作意味着专有商业需要做出一些牺牲——但他们可以参与到世界上最具创新性的项目中。

远疆的探索者们选择在开放的环境中工作——火星是他们的目的地。那么下一步呢？

00:00:12 - Saron Yitbarek:

rover Curiosity

2012 年 8 月 6 日，一枚汽车大小的漫游车，好奇号，从火星大气的顶层降落到了这颗星球的地表。降落花了 7 分钟，下降的过程仿佛永无止境。漫游车携带着贵重的货物：500,000 行代码，76 台尖端科学仪器，以及另一些工具，用以开展一些前所未有的实验。先是超音速降落伞展开，接着是隔热护盾分离，然后，反冲发动机启动，甚至在半空中部署天车——最终，好奇号平稳着陆。

00:00:59 - NASA 工程师：

现在，是时候看看“好奇号”会将我们走向何方了。

00:01:13 - Saron Yitbarek:

听见那声音了吗？那就是火星车成功着陆后，满屋子工程师的欢呼声。几天后，他们将收到来自贝拉克·奥巴马总统的贺电。但现在是只属于他们的时刻。这群人一同完成的这件事，是他们中的任何一人永远都无法独自做到的。

00:01:39:

我是 Saron Yitbarek，这里是代码英雄，来自红帽的一档播客节目。本季以来，我们见证了开源技术对我们工作方式的重塑，及其对人们能力的拓展。作为线索，社区这个要素贯穿了整季节目。这些故事告诉我们该怎么团队协作，该怎么向行家学习，以及，如何在这同时学会听取新人的意见——一言以蔽之，如何更好地与人交流。

00:02:11:

在第二季的终章，我们在记住所有这些道理的同时，我们将探索开源是如何为那些伟大的项目赋能的。*insight* 你也许在去年的十一月观看了 NASA 的洞察号登陆器抵达火星地表。彼时四周皆是欢呼与拥抱。事实证明，这项向星星——这颗红色的星球发射漫游车的任务，只有把宝押在众人合作上才能实现。

00:02:45:

我刚才说过了吗？那台火星车有它自己的推特账号。它还有 4 百万粉丝呢。这还不算什么。最近它给地球人发送了一条推文，是一封邀请函，邀请人们使用 NASA 喷气动力实验室开源的说明和代码，建造一辆属于自己的漫游车。我们跟踪采访了一个民间的组织，叫 SGVHAK 小组，他们是第一批自己建造漫游车的人。

00:03:13 - SGVHAK 小组:

现在我要试试从坎上开过去。

00:03:15:

看看她（LCTT 译注：指漫游车）这样能不能行。

00:03:15:

上来了——不，不行。

00:03:15:

不行。

00:03:20:

她以前很容易就能开过去的。不过我们现在对轮子做了一些改动，她就有点不乐意了。

00:03:27 - Saron Yitbarek:

这个小组的成员包括 Roger Cheng、Dave Flynn、Emily Velasco 以及 Lan Dang。

00:03:34 - SGVHAK 小组:

Dave 包揽了所有机械设计，我想我应该算是负责组装的。Roger 是我们的程序员，Dave 是机械设计师，Lan 是我们无畏的领导。

00:03:45:

你也是一个，你是队里的机械制造专家。

00:03:49:

大概吧，应该算是。

00:03:51:

跟他们讲讲伺服电机架子的事。

00:03:53:

噢，好的。

00:03:54:

是你把电机凑起来的，现在可以让它运行，演示一下——

00:03:58:

我们现在就拿——

00:04:00 - Saron Yitbarek:

打造这样一部漫游车可不是玩乐高玩具。它需要真正的火箭科学家花 200 小时来完成。所以，让我们给这些家伙几分钟的准备时间，稍后再回来。

00:04:19:

与此同时，我想了解更多关于为什么一开始 NASA 要将漫游车的技术和代码向全世界开源。我刚刚找到了回答这个问题的最佳人选。

00:04:29 - Tom Soderstrom:

Jet Propulsion Laboratory

我叫 Tom Soderstrom，我是 NASA 喷气推进实验室 (JPL) 的 IT 部门的 technology and innovation officer
首席技术与创新官。

00:04:37 - Saron Yitbarek:

JPL 总共有大约 6000 人。它是 NASA 的研究中心之一，由联邦政府提供资金，专注于研究如何用无人机械进行太空探索。

00:04:48 - Tom Soderstrom:

我们寻找地外生命，我们也探究宇宙的起源和未来。除此之外，考虑到可能的星际移民需求，我们也在寻找地球 2.0。也就是具有类似地球的环境的、另一个适宜人类居住的行星。

00:05:33 - Saron Yitbarek:

是的，他们肩负着伟大的使命。不过，事情是这样的。Tom 的团队并不是一群秘密进行研究、与外界全然不接触的工程师。他们与下一代的科学家们有着深厚的关系。他们常常采用全新的方式激发大众的创造性思维。事实上，这是他们工作中至关重要的一部分。漫游车开源计划释放出了部分设计，这样一来，NASA 之外的民间组织就能组建他们自己的漫游车了。这是 NASA 促进创新的宏观战略的一部分。很自然地，Tom 从开源漫游车开始，向我讲述了为何开源 NASA 的成果是如此重要。

00:05:46:

有很多人可能像我一样；发现这个网站时，我想，天啊，也许我可以自己打造一个漫游车呢。这真是太激动人心了。这太令人神往了。

00:05:55 - Tom Soderstrom:

我们打造这个开源网站，是为了能让公众和学校能够从中学习。而且我们发现，围绕漫游车开源形成的社区是非常优秀的实验平台。所以当我们想测试新鲜事物的时候，这里就是最佳的演练场，非常简单，非常快速，我们能很快就把想要的技术应用到真正的漫游车上。所以我们希望人们能够把像太阳能面板、加速度传感器、科学实验仪器，以及先进的人工智能程序等结合起来，我们只是希望这些经验能够扩散，在这些方面感兴趣的人最终也能对太空感兴趣。因为太空真的是太酷了。

00:06:32 - Saron Yitbarek:

所以除了刚才提到的这些之外，人们在这个项目中还做过什么让你印象深刻的，或者是让你觉得激动的事儿？

00:06:42 - Tom Soderstrom:

降价是一项非常令人印象深刻的工作，而现在有一堆人在试着用人工智能干这个事儿。这是目前我见到的最有趣的事情之一，我很想看到有人能为其添加一个机械臂。

00:06:55 - Saron Yitbarek:

真是太酷了。

00:06:55 - Tom Soderstrom:

也是我们正在思考的问题。

00:06:58 - Saron Yitbarek:

这个项目是全方位开放的吗？我是指，对于硬件部分，你们给出了一份推荐的零部件清单，但参与者如果愿意，也能自行替换，换成更便宜的，或者换成更贵的；而软件部分就完全是开源的了。这中间有任何非开源的部分吗？

00:07:17 - Tom Soderstrom:

没有，完全是开源的。

00:07:19 - Saron Yitbarek:

如果你告诉我高中生都能通过这个项目自行制作漫游车，我会觉得非常震惊。那听起来太难了；你知道吗，听起来太高大上了，高到只有 NASA 才会去做的那种。对吧？真正地组装火星车会是件容易的事吗？

00:07:36 - Tom Soderstrom:

我们并不想将这做成一个玩具。这个项目是货真价实的，能给参与者带来实际的经验。加利福尼亚州的几所学校已经将其纳入他们的科学、工程与编程课程中，比如 **Mechatronics** 所谓的机械电子学，就是上述几方面的结合体。这让我们非常高兴，因为这样，在不久的将来就能有一批训练有素的专业人才来到 NASA。

00:08:04 - Saron Yitbarek

但这似乎不仅有利于未来的 NASA 工程师、未来的科学家，也有利于你们（LCTT
译注：指目前在 NASA 工作的科学家、工程师）。跟我多说说这方面吧。

00:08:14 - Tom Soderstrom:

嗯……突然间，我们就有了一个非常简单易用的实验平台。我能想象到开源的漫游车在 JPL 附近行驶，还向人间好。在于艰苦环境中验证某些技术之前，我们能在后院里对它们快速地进行试验。它成为了我们可以实验的移动平台。话又说回来，我认为人工智能会是这些技术中非常有趣的一样，因为它发展得太快了，而且，向它加入新的机器人部件也很容易。

00:08:50 - Saron Yitbarek:

所以，理论上来说，你们也可以创建一个你们自己的移动试验场，可以进行试验，也不用将其开源，对吧？它本来可以成为一个内部项目。但它现在是开源的。这能让你们多学到什么呢？

00:09:08 - Tom Soderstrom:

嗯，这是一个好问题。为什么要开源？对我们来说，开源才是更困难的选择，不过那也是挑战所在。我们希望，通过将其开源，学校以及其他民间组织能够用上代码，并且将漫游车造出来。编写一本不是机器人学专家的人也能看懂的手册带来了大量额外的工作。但起码，当我们最终将一些东西真正开源的时候，它变得更整洁、更紧凑、文档更完善了；因为我们知道，其他人会来使用这些东西，所以也必须具备一定的可扩展性。因此，项目的质量提高了。我曾经和 Netflix 的人谈过，他们也有同样的感觉，被开源的代码普遍更加整洁、质量更好。

00:09:54 - Saron Yitbarek:

你是怎样看待开源漫游车的？五年，十年，甚至将来的二十多年后，你觉得它们能做什么，而人们能够创造出什么？你认为这些能为身处地球的我们带来什么潜在利好？

00:10:11 - Tom Soderstrom:

能带来多方面的利好。现在的漫游车被设计成使用轮子行驶。我可以说，它其实也可以被设计成采用步行，也可以对它进行扩展，加上跳跃的功能。我们已经送了一架直升飞机上火星，它现在还在飞着。所以说，这些漫游车并不仅仅局限于现在我们所说的漫游车。有了全世界各地人们的实验，我们能走得更快，更远，不断探寻可能性的上限，并提出一些坦率地说我们可能没有想到的想法。未来到底会是什么样的？我迫切地想了解，不过我不知道。

00:10:49 - Saron Yitbarek:

通过创造这个平台，你们已经为许多人提供了打造自己的“玩具”的机会。接下来就让我们拭目以待吧。

00:10:57 - Tom Soderstrom:

是的，这是很重要的一点。因为我发现，我能想到的主意都已经有人做过了。但我们需要找到这些人。无论我们最开始做的是什么，一旦到最终用户手上，他们就能以我们无法想象的方式将其改进十倍。所以，向大众开放一些这样的“玩具”，任他们自由发挥，他们会因此而得到锻炼，这对将来使用更先进技术的任务也大有益处。

00:11:23 - Saron Yitbarek:

这真是太棒了。

00:11:29:

如果你想了解开源漫游车项目，你可以访问 opensourcerover.jpl.nasa.gov。你能在那儿玩个尽兴。我们稍后再与 Tom Soderstrom 进行更多讨论。但首先，让我们来更深入地挖掘 NASA 与开源之间的关系。

00:11:51 - Hila Lifshitz-Assaf:

大家好，我是 Hila Lifshitz-Assaf，纽约大学商学院的教授。

00:11:57 - Saron Yitbarek:

Hila 正在研究 NASA 是如何打破知识壁垒的。

00:12:02 - Hila Lifshitz-Assaf:

如果你能够回到过去，回到 15 世纪，看看那些孤独的发明家，像莱昂纳多·达芬奇和其他人，他们某种程度上只在自己所在的本地社区活动。之后，有了工业革命，有了实验室。此后的两百多年里，人们都在实验室里工作。最大的变化发生在 21 世纪，数字革命之后我们有了开源的概念，而它打破了之前这类组织结构的边界。

00:12:34 - Saron Yitbarek:

在 NASA 工作的日子里，Hila 见证了这种巨大变革的典型例子。

00:12:39 - Hila Lifshitz-Assaf:

NASA 最吸引我的一点在于，他们就某种意义上来说是最勇敢的，因为他们勇于接
research and development
受战略性的研 究 与 开 发 (R&D) 事业中的挑战，他们的科学家、工程师，最优秀的一批人才为此而努力工作，并使这些挑战向大众开放。而且，我必须说，直到现在，很多其他的组织在做开源科学或是众包的时候，他们不会把最核心的、战略性的难题交给社区，而是只开放一些次要的任务。无论成功还是失败，这都不会让他们的组织蒙受多大损失。而 **NASA** 做的事情，一旦成功了，就真的改变了一切。

00:13:14 - Saron Yitbarek:

自 2009 年起，**NASA** 就开始使用像 **Innocentive** 和 **Topcoder** 这样的开源创新平
台。而且，他们不是闹着玩的。就像 **Hila** 所提到的，他们甚至把他们核心的 R&D
难题都放了上去。

00:13:29 - Hila Lifshitz-Assaf:

这些问题上传之后，甚至没过一年，我们就已经陆续开始看到来自众包平台的解决
方案。

00:13:34 - Saron Yitbarek:

这确实很快。我将在此举出一项研究成果，它是 **NASA** 开源科研的诸多喜人硕果之
一。**Hila** 向我们将讲述，他们是如何将太阳耀斑的预测技术提升一个台阶的。

00:13:46 - Hila Lifshitz-Assaf:

嗯……太阳耀斑的预测是学术界研究了数十年的一项非常困难的太阳物理学问题。
总的来说，他们已经完成初步的构想，以便更多的人可以参与这方面的研究。而且
他们非常在意这一点。这让我觉得很惊奇，因为他们确确实实地希望能从太阳物理
学圈子外的人那里找到解决方案。**Bruce Kragan**，一名来自新罕布什尔州乡村的、
半退休的无线电工程师，只花了三个月，就给我们带来了太阳耀斑的预测方法。当
NASA 验证模型时，他们发现，这种预测方法的准确率有 80%。用传统方式需要花
费数百万美元及数年时间才能获取的成果，在三个月内就被收获了，花费大概在三
万和四五万美元之间。

00:14:36 - Saron Yitbarek:

你也许已经猜到了，这样的变化需要 **NASA** 内部的文化作出改变。

00:14:44 - Hila Lifshitz-Assaf:

他们中的有些人选择邀请外来的解决方案提供者进入项目组，还有一些选择提供实
习或合作机会；他们采用了很多有意思的方法来交流知识，使其不至于淤塞。

00:14:59 - Saron Yitbarek:

仔细想想，这其实是很美好的。就如我们所知，很多组织仍然抵制开源活动，不愿
放弃专有软件。但在这里，你可以找到这个星球上最具创新性，愿景宏大的一群科
学家；而他们说，嗨，我们一起来。这样的态度有绝大的影响力。

00:15:22 - Hila Lifshitz-Assaf:

我们已经看到了开源给软件行业带来的变革。我们目前尚未在科学与技术方面见到
太多改变，不过，我认为，现在就是最好的时机。计算物理学、计算生物学越是崛
起，这就越有可能实现。我认为，像这样，就有更多人能够参与到科研中，协助处
理不同的任务。以这种方式，科学技术的发展速度或许甚至能够超过应用开发。

00:15:53 - Saron Yitbarek:

Hila Lifshitz-Assaf 是一位来自纽约大学斯特恩商学院的教授。

00:16:00:

NASA 在开放他们的研究课题的道路上受益良多，但他们也通过
Small Business Innovation Grant Program
中小企 业 创 新 资 助 计 划——这个项目鼓励私营经济中的创新——
blue sky projects
打造了另一种社区。所有大 胆 的 计 划 都有风险，可能付出代价，但有时回报也
尤其丰厚。

00:16:22 - Dan Wachspress:

大家好，我叫 Dan Wachspress，我是一名航空工程师，在 Continuum Dynamics
集团工作。

00:16:28 - Saron Yitbarek:

Dan 的公司专注于研究与开发旋翼飞行器，也就是直升机、旋翼飞机这类的飞行
blue sky
器。这可以说从是字面上看就很“蓝 天”（LCTT 译注：此处呼应之前提到的 blue
sky projects）了。他们一直与 NASA 合作，致力于解决垂直升降的问题，让飞行
器能变得像是 Jetson 的“空中出租车”（LCTT 译注：典出《The Jetsons》，一部
始播于 1962 年的系列动画，这里的 Jetson 指动画主角 George Jetson）那样。

00:16:50 - Dan Wachspress:

总体的构思是，只要用电动机代替车辆上的燃气涡轮发动机，你就可以安装更多推
进器。它们会比现有的更安静、更安全，直升机所有恼人的问题统统消失，而我们
可能会得到这样的未来：你打个的就能从达拉斯去到沃斯堡（LCTT 译注：均为美
国得克萨斯州的城市），而那是一辆准乘 5 人、无人驾驶的电动飞的，而非一辆汽
车，在汽车里面你还得跟堵车较劲。这就是我们的愿景。

00:17:22 - Saron Yitbarek:

有很多公司，比如 Uber，对这种空中出租车的设想非常感兴趣。而 NASA 在这之
中扮演的角色是伟大的：它打破了不同公司本要各自为营死磕研究的壁垒。

00:17:38 - Dan Wachspress:

那些公司必定不想走漏风声，他们想保持商业机密，不愿共享成果。NASA 的目标
就是把尽可能多的研究成果送到他们手里，让他们尽可能地获得能力。而且我敢说
你如果去问他们任何一家公司，他们肯定会说，没有 NASA 长久以来的技术支持，
他们没法做到这么快地开发。

00:18:13 - Saron Yitbarek:

我想这并不影响 NASA 拥有一些非常棒的风洞。从火星漫游车到飞行汽车，你有感
觉了吗？我们谈论的是创新，天穹尚且不是我们的极限。而且这一切都归功于我们
全地球级别的合作，而正是开源观念让这一切成为可能。

00:18:41 - Saron Yitbarek:

我们再回来和 NASA 的 Tom Soderstrom 聊聊。Tom 认为，我们两季以来探讨的
开源的“魔力”已经形成了一股巨大的推动力。他称之为第四次工业革命。

00:18:59 - Tom Soderstrom:

创新总是与科技的潮流挂钩。当今，有好几波科技浪潮同时袭来，它们的合流将引起巨大的海啸。它们逼近得越来越快，所有事物都会为之改变，因此，我称其为第四次工业革命。

00:19:21 - Saron Yitbarek:

我将简单几句带过那些浪潮，虽然就它们中的每一样都可以独立讲一期节目。**Tom** 所说的“即将到冲击我们生活的潮流”包括了大规模的网络安全问题、量子计算之 **software-defined** 类，也包括了各类“软件 定义 的”东西。但这些只是其中的一小部分。他还提到 **ubiquitous computing natural interfaces** 了普 适 计 算、自然 用户 界面和物联网。

00:19:49 - Tom Soderstrom:

这些技术都将是那场海啸的一部分，生活将因为无处不在的内 **built-in intelligence** 置 智 能而大为改观。

00:19:54 - Saron Yitbarek:

想象一下，如果这些技术合流，产生了 1+1 大于 2 的效应，“海啸”发生，情况会变成什么样？

00:20:04 - Tom Soderstrom:

我不认为这种变化会是突然发生的；不会有像是一个人站起来感叹“你看这个”这样的情况。就如我刚才所说，内置智能会渗透进生活的各个方面。“智能”，对吧？比如智能电视，或者智能会议室。我们生活中的事物会变得越来越智能，越来越聪明。对于企业来说，它意味着，你可以对着一个房间提出问题，然后程序会为你在数千种不同的数据源、好几 PB 的数据里找出答案。而这背后，就是自然语言处理、深度学习、机器学习这样的技术；而我们不会突然说，“哇，我们已经发展到这种地步了”。不会的，这是一个循序渐进、潜移默化的过程。

00:20:49 - Saron Yitbarek:

这第四次工业革命，又会如何影响你在 **JPL** 的工作呢？

00:20:57 - Tom Soderstrom:

我认为首先是实验，下一场工业革命可以帮助我们更快地完成实验，并且利用更好的组件，包括软件和硬件，也许我们不需要把所有的东西都造出来，但是我们可以更智能地使用它。然后是开源。开源真是一项颠覆我们工作方式和工作内容的事业。

00:21:23 - Saron Yitbarek:

此话怎讲？

00:21:24 - Tom Soderstrom:

说到开源这东西，我的资历足够老，我当年也经历过开源战争。当时开源还只是个玩笑。那时的开源很糟糕，开源软件似乎比商业软件低一等似的。不过，那些都已经过去了，至少在喷气动力实验室是这样的。现在，开源的方式更适合我们。开源的研究与开发更加经济划算，也可以更快地做实验验证。另外，因为开源软件的存在，现在我们再也不用全部从头造轮子了。我们可以自己开发；在此之后，如果我们选择将之开源，就能从社区获得帮助，对成果进行改进。开源的软件和科研也能为我们招贤纳士。这就很有趣了。

00:22:05 - Saron Yitbarek:

确实很有意思。

00:22:07 - Tom Soderstrom:

我认为现在的人们，特别是新一代，可以将他们的项目开源，以此获得更多的影响力，得到尽可能多的星标。你可以看见他们的简历上写着，我开发的软件得到了多少星标。

00:22:21 - Saron Yitbarek:

对你来说，让你们的工作方式和内容发生变化的开源，并不是什么新鲜事物，对吧？这点让我很感兴趣。你说你经历过开源战争。你也见证了开源在这么多年来的
发展。那么，当今的开源有哪些优势是十年、十五年、二十年前没有的呢？

00:22:47 - Tom Soderstrom:

有两点。一个就是云计算。我们不用签署一份大合同，购买一堆软件然后使用许多年。我们可以在云端简单地做个实验。这点相当重要。还有另外一点，就是开源并
religious
不比闭源的安全性差。这早就不是——请原谅我的用词——一个信 仰的问题了。这更多地只是一个实用性和经济性的问题。

00:23:15 - Saron Yitbarek:

开源显然在你的工作当中扮演了一个重要的角色；尤其是当你谈及喷气动力实验室的未来，以及你期望它以怎样的方式前行的时候，开源似乎仍将是故事的重要组成部分。说到开源社区式合作开发，你认为这种方式在最理想的情况下能够导向什么样的成果？而对于人类，它的意义又是什么？

00:23:42 - Tom Soderstrom:

好问题。我认为你刚刚已经说出了正确答案——“人类”。它的理想结果是能让每个人都参与到这份工作中来，而它的意义也在于此。你知道，总有一天我们会把人送上火星。我们要探索更广袤的深空，去寻找地球 2.0。我们还要再开展一次载人登月。这些都需要来自世界各地的人们积极参与。

00:24:15 - Saron Yitbarek:

我被这场变革深深的吸引了。Tom Soderstrom 是 NASA 喷气动力实验室的首席技术与创新官。

00:24:29:

从地球 2.0 的话题，回到地球 1.0。我们应当铭记，“第四次工业革命”的起源，也是艰苦朴素的。开源项目固然可以变得宏大无比，但千里之行毕竟始于足下；一项伟大的事业，在其伊始，或许只是几名极客试图让漫游车正常工作。

00:24:47 - SGVHAK 小组:

我们再看看这样它能不能跑起来。我们会——再加一样东西，方便它从坎上过去。
过去了！耶！

00:24:59 - Saron Yitbarek:

耶。

00:25:02 - SGVHAK 小组:

啊呀，我下不来了，它被卡在花坛里了。

00:25:14:

我们把它抬出来就行了。

00:25:16:

这又不是在火星。你可以直接走过去，把它拿起来。

00:25:20 - Saron Yitbarek:

我觉得他们有所进步。

00:25:25:

让我们和这些代码英雄说再见吧，他们要继续学习、探索、投入他们的工作了。他们会知道，通过开源，我们总有一天可以到达天空之外的高度。

00:25:40:

如果你想让你自己的开源游戏项目上一个台阶，不要忘了我们正在开发《代码英雄》游戏，开发已经持续了一整季的时间，现在你仍然可以贡献代码。

00:25:52 - Michael Clayton:

嗨，我是 Michael。

00:25:52 - Jared Sprague:

嗨，我是 Jared。我们是《代码英雄》游戏的开发人员。

00:25:58 - Saron Yitbarek:

我们请来了红帽的 Jared Sprague 和 Michael Clayton，看看情况怎么样。

****00:26:03 - Michael Clayton_**

我有点惊讶于我们这么快就引来了这么多的感兴趣的人。反响非常好，提交的拉取请求蹭蹭上涨。

00:26:17 - Saron Yitbarek:

你认为是什么让大家这么激动？

00:26:18 - Michael Clayton:

我认为我们播客的许多听众都受到了鼓励，因此他们看到这个开源项目就迫不及待地想试试看。尤其是自从我们开始征求不限类型的贡献，让任何想要贡献的创意人员、任何类型的工程师都能够参与项目后，听众就找到事儿做了。

00:26:39 - Saron Yitbarek:

那么接下来希望社区能给大家带来什么？这游戏还在开发中。你最想见到什么？

00:26:47 - Jared Sprague:

我个人非常享受开发过程中的进入节奏的感觉，我们让大家贡献美术素材和音乐以及音频效果、故事线、代码，所有这些东西都可以并行不悖。一旦每个人都进入节奏，我们可以一起开发，共同看着游戏被开发出来，那真是太美妙了。

00:27:14 - Saron Yitbarek:

顺带一提，我们的游戏的公测版将会在今年的五月七日到九日，在波士顿开办的红帽峰会发布。届时，数千名像你一样的代码英雄会前来进行为期三天的创新与教育之旅。访问 redhat.com/summit 以查看详情。

00:27:34:

最后一件事，这或许是第二季的最终章，但这不是真正的告别。第三季已经在筹备中！与此同时，我们也会为大家带来一期彩蛋。我们开展了一次圆桌会议，与你们最喜爱的思想家们探讨开源的未来。请锁定一月份。别忘了，如果你不想错过最新一期节目，记得点关注哦。就点一下，点一下又不会怎么样，这样你才能在第一时间收到新内容的通知哦。

00:28:09:

我是 Saron Yitbarek，感谢你聆听本季节目，生命不止，编码不休。

什么是 LCTT SIG 和 LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 LCRH SIG 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-2/open-curiosity>

作者: Red Hat 选题: bestony 译者: Colin-XKL 校对: Northurland, wxy

本文由 LCRH 原创编译, [Linux中国](#) 荣誉推出

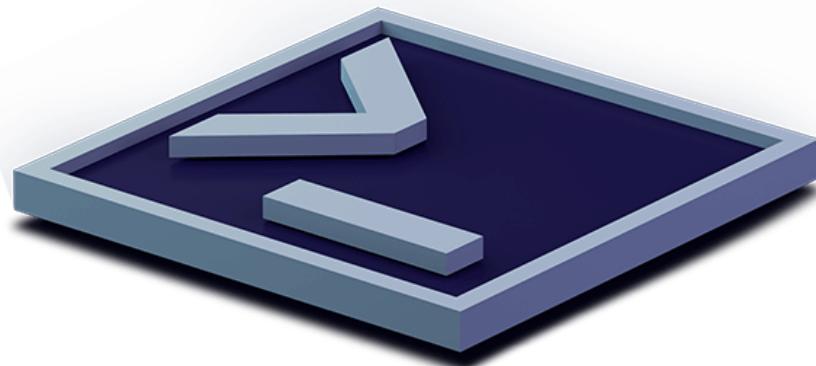
《代码英雄》第 2 季（9）：特别篇 - 开发者推广大使圆桌会议

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频播客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《代码英雄》系列播客第二季（9）：特别篇 - 开发人员拥护者圆桌会议的音频脚本。

developer advocate

导语：开发者推广大使在开源社区中起到的作用是举足轻重的。我们邀请了几位这样的推广大使来到本期节目中，以向我们揭示他们的工作方式，并阐述这份工作背后的意义。

来自 Mozilla 的 Sandra Persing、Twilio 的 Ricky Robinett 与 红帽的 Robyn Bergren 将在此接受 Saron 的采访，分享他们的工作内容、他们支持社区的方式，以及他们对 2019 年的展望。

00:00:06 - Saron Yitbarek:

大家好，我是 Saron Yitbarek，这里是红帽的原创播客节目

Command Line Heroes

《代 码 英 雄》的特别篇。我们的节目在第 2 季去了很多地方。我们探索了[编程语言](#)、[数据大爆炸](#)、[安全危机](#)以及[无服务器](#)时代的到来，我们甚至去了[火星](#)。

00:00:28:

但是在结束了第 2 季之后，我们还有一个地方要去。我们要走进推广大使和思想领袖们的认识当中，他们帮着塑造了开发人员所做的全部工作。有时，他们被称为 **developer advocate**
开发者推广大使，或者说他们担任开发者关系的职位，或开发者布道师。

00:00:50:

从开发人员的角度来看，无论他们的具体头衔是什么，他们似乎都做着许多相同的事情。你在会议上见过他们发表主题演讲。你在播客上听过他们接受采访，就像在本期节目中一样。你可能还阅读过他们的博文。但是他们是谁呢？他们到底在用自己的声音做了些什么？

00:01:10:

为了迎接 2019 年的到来，我们为大家召集了一个优秀人物的圆桌会议。尽管他们的头衔各不相同，但他们的目的是一致的。他们来到这里，是为了帮助开发人员，并确保其声音和需求能被大众听到。这些人都是典型的代码英雄。

00:01:29:

Bay Area Global Strategist
来自湾区的 Sandra Persing 是 Mozilla 的全球策略师，也是 DevRel 峰会的创始人。

00:01:38 - Sandra Persing:

嗨， Saron。

00:01:39 - Saron Yitbarek:

你好。还有同样来自旧金山的 Ricky Robinett。他是 Twilio 的开发人员网络总监。

00:01:47 - Ricky Robinett:

你好。

00:01:49 - Saron Yitbarek:

从凤凰城外和我一起来的是 Robyn Bergeron，Red Hat 的社区架构师。

00:01:55 - Robyn Bergeron:

嗨！你好吗？

00:01:56 - Saron Yitbarek:

我很好。

00:01:57 - Robyn Bergeron:

看到你真高兴。

00:01:57 - Saron Yitbarek:

Developer Advocate
在不同的地方，这份工作的践行者被冠以不同的头衔。有开发者推广大使、
Developer Evangelist Developer Relations
开发者布道师、开发者关系；对于新事物来说，基本定义非常重要。因此，我认为一个好的起点是来定义这些东西是什么，尤其是定义你可以在这个领域做什么。所以，你能告诉我你的头衔以及这个头衔实质上的含义吗？
Ricky，让我们从你开始吧。

00:02:22 - Ricky Robinett:

Director of the Developer Network

好的，听起来很棒。我的头衔是开发者网络总监。我有幸供职于一个开发者关系专家团队。我们有一个布道师团队、一个教育团队和一个社区团队。因此，这绝对是一个大杂烩，有你听说过的各种不同的头衔，我们将其汇集在这个团队之中。

00:02:43 - Saron Yitbarek:

厉害啊。Sandra，你呢？

00:02:45 - Sandra Persing:

Global Strategist

我在 Mozilla 担任活动和赞助的全球策略师，并与 Mozilla 新兴技术小组的开发者拓展团队合作。我通常会将我的日常活动解释为大量的研究、交流和探索，而这都将最终影响我们评估与决策应当如何投资我们的资源：我们的时间、金钱、纪念品、演说者等等，以回馈开发者社区，同时也从开发者社区得到反馈。这份工作既有布道的一面，也有倡导的一面。

00:03:24 - Saron Yitbarek:

很棒。Robyn，该你了。

00:03:26 - Robyn Bergeron:

Community Architect

你好。好吧，我的正式头衔是社区架构师。人们就我这个头衔有过很多问
Community Manager Developer Advocate
题。我曾被称为社区经理，也曾被称为开发者推广大使，甚至还在过
Operations Advocate
去的一份工作中做过运维推广大使。但是我想我所做的是……嗯，你知道，
像是“社区经理”之类头衔所暗示的“你在管理着社区中发自内心为项目做出贡献的
人”的这种想法，其实相当愚蠢，因此，我喜欢把我所做的事情看作是构建一个框架，
人们可以真正顺利地参与其中，确保过程中没有障碍，并确保他们可以完成所有他们想要做的事情。

00:04:02 - Saron Yitbarek:

Ricky，由于你基本上是负责整个开发者网络的总监，你不仅仅是置身于推广大使
团队、布道团队，也是社区团队之中，这有点儿像是你在运行着整个项目。你是怎
样理解的这个的，是向开发者推广还是推广开发者？

00:04:20 - Ricky Robinett:

是的，这是个很棒的问题。Phil Nash 是我们团队中的一名布道师，他对此有一个
很好的认识，而我要将它剽窃过来。我们可以从很多不同的角度来解释这份工作，
但说到底，我们的工作其实大体上就是帮助他人。

00:04:33 - Saron Yitbarek:

我喜欢这一点。是的。

00:04:35 - Ricky Robinett:

我们帮助开发人员，有时候这些帮助看起来就像是在 Stack Overflow 上回答问题，
有时看起来像是构建一款新工具，有时看起来像组织一个活动，有时看起来又像是
发起一次产品的内部改动。因此，我认为这就是我所听说过的最好的认识——我

们的责任是帮助。

00:04:57 - Saron Yitbarek:

的确。Sandra，你的职位让我感兴趣的一点是，你不光是策略师，还是全球策略师。让世界各地的开发人员组织起来，并且帮助他们，这是你在 Mozilla 的工作的一部分。在全球范围内的推广倡导会是什么样的呢？在不同的国家、不同的大洲之间，这份工作的含义会有所不同吗？

00:05:20 - Sandra Persing:

对，确实存在一些不同。我们今年刚刚在新加坡举办了第三届 DevRel 年度峰会。在过去的两年间，我们在西雅图主办了这项活动，而到了新加坡，我们看到了不同的视角。当我们走出湾区时，就连某些最基本的组织方式都需要做出改变。比如，如何确保一切都能在线下运转，因为连接性是一个大问题；再比如，我们该怎样使一名工作于印尼的开发人员能在峰会过程中与社区充分互动，无论他是在线上参加聚会，还是来到现场。

00:05:59 - Sandra Persing:

我们总会发现，所谓的“基础”，一些对我们来说如此寻常的事，比如在湾区举办一次简单的聚会——这再常见不过了，对吧？你高呼一声“我要参会”，就会发现到处都有聚会可以参加。然而对于，比如，越南的开发者来说，此事可以是具有特殊意义，且对他们而言相当重要的。这种聚会很宝贵，丰富了他们的开发者生活。

00:06:27 - Saron Yitbarek:

我注意到的一件事是，大家都以某种方式提到了社区。而我自己也运营着一个社区，Code Newbie，而且我经常被一些公司聘用为他们的布道师或社区经理。有一件事我一直都有点儿担心，也许有点畏惧；我在想，在过去的三四年里，我一直在尽我所能，尽心尽力地负责着这个社区。

00:06:55:

但如果我为一家公司工作，我就必须要牺牲这些吗？我是不是必须要将公司置于社区的需求之上？我该如何平衡这种关系呢？

00:07:06:

Community Architect

所以我在想，Robyn，或许我们可以问问你这个社区架构师，你是如何区分这两者的呢？或者说，你是如何平衡这两者的？

00:07:16 - Robyn Bergeron:

好吧，这当然是一个有趣的平衡。我的意思是，我以前的一个工作实际上是担任 Fedora 项目负责人。而你知道，Fedora 是 Red Enterprise Linux 的上游，在这里，你角色的一部分确实是某种找寻平衡的行为，对吧？两者之间的平衡是让社区里的人们开心，让公司对社区所做的事情高兴，同时要确保每个人都是快乐大家庭的一员。

00:07:41:

而且，你知道，我想当你在这个职位上做得最好的时候，你肯定会时不时地激怒公司决策层的某些人。但是你知道的，最终还是要用结果来证明，对吧？

00:07:59:

人们总是问我，你是如何平衡 Ansible 和红帽之间的关系的，你知道，当 Ansible 被红帽收购时，就像是——哦我的天哪，红帽会接管它，然后对它做些糟糕的事情，并摆脱 Ubuntu 的支持吗？

00:08:12:

而这就像是，拥有整个项目的全部目的就是为了破坏它，就像是为了不要吸引 4000 名贡献者而故意变得糟糕一样。

00:08:22:

确保你的管理层信任你，并始终与人们保持清晰的沟通以了解实际发生的事情，并确保沟通通道的两侧都不会出现意外，这是成功与否的部分原因，也许并不总是会成功，但肯定会让让人感到惊讶。

00:08:41 - Saron Yitbarek:

嗯嗯。确实。Ricky，你呢？当你在做这么多不同事情的时候，你是如何看待这种关系中的平衡的？

00:08:49 - Ricky Robinett:

我认为你必须相信公司和技术。你必须相信，你所带给开发人员的东西将会对他们的生活、职业生涯以及公司产生影响。

00:09:03:

另一方面，你必须让高管们相信这种方法。因此，我们非常幸运的一点是我们的 CEO 是一名开发人员，而且在很多方面，他是我们和开发者社区打交道的原动力。我们的布道师们的使命是激励并装备开发人员。因此，有时候我们会说这能激励并装备他们吗？因为如果不能的话，我们就不应该这么做，因为这超出了我们的职权限范围。

00:09:36 - Saron Yitbarek:

嗯嗯。Sandra，我觉得你有点儿优势，因为 Mozilla 是一个非营利性组织，对吧？所以我觉得或许——

00:09:44 - Sandra Persing:

我刚想说。

00:09:46 - Saron Yitbarek:

跟我讲讲这个吧。

00:09:47 - Sandra Persing:

Mozilla 的历史就是我们是一家叛逆的公司。我们一直在反抗企业家，对吗？出走 Netscape，并与我们的创始人之一 Mitchell Baker 一起走过的整段历史，确保了 Web ——

00:10:01:

——乃至互联网是面向所有人的开放而自由的资源。我的意思是，我们仍然，我们每个人，每一个 Mozilla 人，都信奉这一口号，我们对此深信不疑。因此，这绝对是一家令人惊叹的、100% 拥抱了社区的公司。

00:10:22 - Saron Yitbarek:

确实。所以，Ricky，我还清楚地记得那件红色的运动夹克，我还记得你亲自出马做的那些很棒的演示，感觉那种联系并帮助开发人员的方式非常新颖。这个想法是怎么在 Twilio 上产生的？

00:10:41 - Ricky Robinett:

嗯，你这么说真是太好了。我们确实相信我们是站在巨人的肩膀之上。你会想到 Apple 公司的 Guy Kawasaki，有人在我们之前就采用了这种营销方式。我认为我们很幸运能在正确的时间把它带给开发人员。

00:11:03:

而且有这么多的人进来，带着我们如何能够做到这一点以及如何能够不断提高方法水平的想法。但是我实际上不知道是谁发明了红色运动夹克，因此现在我需要去探寻历史——

00:11:19 - Saron Yitbarek:

你一定得找到答案。

00:11:19 - Ricky Robinett:

——有关这是何时开始的历史。

00:11:20 - Saron Yitbarek:

这是件很棒的夹克。

00:11:21 - Ricky Robinett:

知道了，我今天下午就去找。

00:11:25 - Saron Yitbarek:

而我想知道的是，布道和推广的想法在 Twilio 是如何随着时间而发生变化的。你知道，曾经你们只是一家小小的初创公司，一家叛逆的初创公司，然后现如今成了一家大公司。随着公司本身的变化，布道的形式又是如何变化的呢？

00:11:46 - Ricky Robinett:

hackathon

是的，开始时，我觉得我本可以一年 365 天都在黑客马拉松上渡过，而在纽约，每个周末你都要从五到六个黑客马拉松之中抉择一个。我们在布道方面所做的许多事情都是黑客马拉松场景，而现在，场景不同了。确实，最大的变化在于公司外部而不是内部。

00:12:11:

因此，我谈到了布道师的激励和装备。所以令人高兴的是，这些年来这个任务并没有发生任何变化，他们激励和装备的方式一直在变化，但任务本身并没有什么改变。

00:12:26 - Saron Yitbarek:

那么 Robyn，随着 DevOps 和 DevSecOps 的兴起，推广大使在你和社区架构师看来如何呢？会有运维推广大使吗？

00:12:39 - Robyn Bergeron:

嗯，实际上那是 我的意思是说，孩子们，那是我辍学后的第一份工作。我不建议这样做，不要在家做这个事情。

00:12:47 - Saron Yitbarek:

留在学校里继续上学。

00:12:48 - Robyn Bergeron:

SysAdmin

听着。你看，小姑娘，我的第一份工作实际上是在 Motorola 担任系统管理员多年，而当我从担任 Fedora 负责人跳槽时，我在 Elastic Search 工作了一段时间，你知道，那个头衔是开发者推广大使，因此我花了几个月在大学里攻读 C。但是我的心一直都在运维这边。我开始对自己真的是一名开发者推广大使感到疑惑？我觉得我只是在推广运维人员而已。我开始自称是一名运维推广大使，每个人都为之注目。

00:13:22:

每个人都说，嗯，这真的是一个很酷的头衔。我的意思是，你知道，我基本上只是在公司内部倡导其他人在做什么。

00:13:33 - Saron Yitbarek:

所以，Sandra，我们谈到了推广大使和布道师在世界不同地区看起来如何不同，但是我想知道，随着我们变得越来越全球化，联系越来越紧密，更大范围和形式的布道对你来说是不是也在随着时间流逝而变化？

00:13:52 - Sandra Persing:

你知道什么是开发者关系吗？我们是在推销我们的产品吗？你知道，而我注意到的是，即使是大公司也正在远离这种策略。要明白的是，做一个真实的人，真正留心倾听并回应开发人员的需求是最为重要的，而不是去推销产品。

00:14:17:

我总是回过头来和我在 Mozilla 的团队分享一个理念，开发者们实际上是我们所合作过的最聪明、最具有创造力的客户群体之一。他们可以大老远就闻到商业销售的气味。因此，我们必须对于如何共享信息保持精明，就像它必须是一群有着才华和智慧的人聚集在一起，成为我们与开发者社区进行沟通的创新方式。

00:14:48 - Saron Yitbarek:

嗯，真的，我很喜欢这样的想法，需要将不同的技能，我想，还有不同的背景，集合在一起，才能很好地服务于开发者，也能帮助开发者自助。

00:15:01:

当我想起开发者推广大使的兴起时，在我看来，它与开源的兴起息息相关。这感觉几乎就像是开源贡献者越多，开源就变得越重要，重视它的大公司也越多，他们几乎必须与这些开源贡献者、这些开发人员建立起更好的关系，并且我感觉这二者真的是紧密相关的。

00:15:27:

因此我很好奇你的想法。那么 Robyn，让我们从你开始吧。是这样的吗？推广大使的这个想法与开源的兴起有所联系吗？

00:15:37 - Robyn Bergeron:

如果你是一家销售软件或是销售许可证和长期支持的公司，你知道，无论你的开源公司的业务模型是什么，如果你没有这种反馈回路，或者你没有真正关注人们在说什么，那你最终会犯下一个众所周知的错误。我认为，真正能在全世界范围内每天都做到这一点，就是成功和失败之间的区别，没有人愿意花时间去做错误的事情。那是个坏主意。

00:16:10 - Saron Yitbarek:

是的，这通常是个坏主意。很好。因此，我想知道你们每个人各自都在关注什么，都真正在思考什么。因此，Ricky，让我们从你开始吧。在你的 Twilio 角色中，你试图为开发者文化带来什么样的改进呢？

00:16:27 - Ricky Robinett:

如果我要说的我与开发者交谈时听到过最多的一个短语，那就是“我不是开发者，但是……”，而这可能是一直萦绕在我脑海中的最大的事情之一——开发者定义的扩大化。

00:16:45:

对我们很多人来说，“冒牌综合症”是一个非常真实的现象。令人惊讶的是，即使是你所认识的一些最好的开发者也在为此努力。而对我来说，我们所有人能在我们的文化中做的最重要的事情之一，就是让人们说，“你知道吗？我是一名开发者。我用代码解决问题。”

00:17:08:

因此，我最喜欢的故事之一是，我们社区有一个名为 Doug McKenzie 的成员，他是个魔术师。他自学了 PHP，以便在魔术中使用技术。

00:17:23 - Saron Yitbarek:

噢，很棒。

00:17:23 - Ricky Robinett:

Doug 之所以如此之酷，是因为他超级谦虚，就像是：“哦，我不是一名开发者。”而突然之间你发现，他正在编写比许多人所见过的更为复杂的代码，可以完成令人震惊的事情。因此，我感觉世界上有许多 Doug McKenzies 这样的人，他们都在用代码做着伟大的事情，我们有机会让他们成为社区的一份子，并且在这项工作中拥有自己的身份。

00:17:51 - Saron Yitbarek:

我喜欢这个故事，因为它让我想起了我曾经为 Code Newbie 播客采访过的人，她是一名作家而且有很多写作任务，必须学习 Git 才能撰写有关 Git 的文章，她写了很多这方面的文章，又逐渐转向了其他的编程主题，最终，你知道，几年过去之后，她实际上已经成为了一名开发者却不自知。

00:18:13:

在我采访她的时候，我说：“嘿，你知道自己是一名开发者吗？”她说：“不，我是个作家。”而我说：“你可以二者兼备。它们并不互相排斥。”但是，是的，这种说法换个角度说，“嘿，我实际上是在编程和创造，因此现在我可以自称是一名码农了”，这种认识对人们来说很难，要做到这样需要一段时间。

00:18:32 - Ricky Robinett:

是的，当然是这样。

00:18:33 - Saron Yitbarek:

是的。因此，Robyn，对你来说，最近几年来你一直在努力推动的最积极的变化是什么？

00:18:41 - Robyn Bergeron:

只是确保在我们的成长过程中，我们不会失去对大局观的把握。我们的主要目标是使人们易于使用、易于贡献、易于实际完成生活中的事情，确保我们不会偏离这个目标，或者在实际项目的某些结构层面获得更多的工程帮助，并确保我们保持所有这些东西都井井有条。我们在这方面做得很好，我不知道是不是，但我认为这很重要。我不知道自己会不会因此而获得诺贝尔和平奖，但我知道这对许多贡献者都非常 important。

00:19:19 - Saron Yitbarek:

这对我来说很有用。Sandra，你呢？作为全球策略师，你近些年来所推动的最积极的变化是什么？

00:19:26 - Sandra Persing:

在我脑海中真正突出的两件事是扩展开发者角色这个定义，这一点非常重要。我们想要发展我们的社区，对吗？那么为什么要限制开发者的定义和描述呢？

00:19:43:

当我们在 Sundance 与 Reggie Watts 和 Chris Milk 等著名人物进行座谈时，我们给电影制作人、制片人、决策者一个机会，让他们说：“哦，我们也能做到。我们不需要受限于作为创意电影制作人能做些什么。”

00:20:00:

我们可以采纳技术，也可以成为开发者，而这相当令人耳目一新。我们在 Mozilla 经历的另一个时刻是让一位芝加哥的灯光艺术家 Ian Brill 与我们合作进行一个项目，我们把这个项目标记为 Arch，我们为今年两个重要的 JavaScript 开发者活动带来了这种巨型的塑料灯，LED 灯，带着七个可编程的树莓派结构。这是为了邀请更多的程序员（无论他们是否自称为开发者）来尝试 Mozilla 今年大力倡导的两种语言：WebAssembly 和 Rust。

00:20:49:

因此，我们创建了两个简单的模板，说，“尝试一下吧”，但这不是我们所想要推动的编程。这不是编程，而是“是的，有几行代码。它是编程语言，但你要做的是创建几行代码，现在可以将它们转化为艺术”，而这让许许多多的新人进入到了我们的群体之中，也就是把他们带到我们的桌子旁边，然后他们会编写他们的代码行，在 Arch 之下走过，去查看目前正在结构中循环着的灯光表现。这真是太神奇了。

00:21:22 - Saron Yitbarek:

哇。这听上去很美，确实美。因此，我想知道的是，当我们谈论到为开发者做推广大使时，我们谈了很多有关社区的话题，也谈论了那个最终的想法，无论我们的职业头衔是什么，我们真的只是想要帮助人们，开发者需要什么方面的帮助？

Ricky，让我们从你开始吧。开发者们说他们需要从你这里得到什么呢？

00:21:44 - Ricky Robinett:

哇。是的，这是个好问题。我认为，我们所发现的其中一件事是，科技变化如此之快，我们被问到的很多问题都是，“我从哪里开始？我首先要做什么？我怎么知道我走的路是正确的呢？”对我们来说，这可能是我们投入最多的领域之一，我们称之为帮助人们发现他们用代码改变世界的能力。

00:22:16 - Saron Yitbarek:

噢，这很美妙。我赞成。

00:22:18 - Ricky Robinett:

嗯，谢谢你。是的，这令人兴奋。因此我们构建了一款名为 TwilioQuest 的工具，以帮助人们发现这种力量，帮助他们知道从何处开始。但我只是感觉，你知道，人们一直在寻找自己的身份或获得允许使用该身份的方式，而对每个人而言这就已经是写代码，是用代码或软件解决问题。有许多人想这么做，然而却不知道从哪里开始。所以这是我们经常思考的一件事。

00:22:51 - Saron Yitbarek:

是的。因此，Robyn，对于你们红帽来说，红帽的开发者所寻求的是什么呢？

00:23:00 - Robyn Bergeron:

很多时候，是有人来找你，他们遇到了一些障碍，也许是“不知怎么的，我的公关人员被机器人遗漏了”，但是很多时候，也有人会说，“嘿，我有一个很酷的想法。或许它不太适合这里，但是我认为它可能会改善社区的运行方式，或者可能会是其它我们正在研究的东西的不错的辅助工具。我该怎么办呢？”这就像是，“好吧，我该如何帮你入门呢？”你知道吗？“我能怎么做呢？比如说，你只是需要有人同意吗？因为我在这儿整天都基本上是对任何事情点头，让人们知道，这行得通，你当然有权力这样做”，所以这是我所认为最好的事情是，你可以做。至少在我看来，就是确保人们在自己的前进道路上没有阻碍，或者如果阻碍他们前进的一件事是正等着某人说可以，我一直在重申，你不需要别人的许可，但是如果有人需要，那么看在上帝的份上，请给他们许可。

00:23:58 - Saron Yitbarek:

所以，最后一个问题，我们需要总结一下，我将向你们逐一询问。你在 2019 年将要倡导的最重要的事情是什么呢？如果你有魔杖，那么你想要改变的下一件大事是什么呢？Sandra，让我们从你开始吧。

00:24:15 - Sandra Persing:

嘿，我就知道你会这么做。好吧，我们将在 2019 年进行的最大挑战和最为激动人心的项目是真正兑现我们的承诺，使 Web 成为最棒、最大、最易于访问的平台之一。我们总是告诉开发者这是你应当为之构建并部署的地方，但是我们知道 Web 本身非常复杂，而且有着多个浏览器供应商，有时候这并不是一个正确的说法，这对我们而言是一个长期的挑战，尤其是在 Mozilla，在这里我们想要保持 Web 开放、自由，并且能被所有人访问。我们想要继续确保能够兑现对开发人员的承诺，确保网络确实是开放的、可访问的，对所有人都开放。

00:25:10 - Saron Yitbarek:

嗯，爱了。Ricky，你呢？

00:25:13 - Ricky Robinett:

只需要确保我们为开发人员提供服务，使得他们能够在全世界范围内进行线上或是线下的聚会。更容易专注于你所看到的内容，并且无需记得开发人员是在何处，即使你没有看到他们。因此，我会挥舞我的魔杖，并在全世界更多的地方，找出我们如何能够帮助那里的开发人员。

00:25:36 - Sandra Persing:

dark matter developer

我只想说，我喜欢 Jarod 有关暗 物 质 开 发 者的演讲。实在是太棒了。

00:25:42 - Ricky Robinett:

当你第一次听到它的时候，感觉像是一个了不起的概念，就像是，“哇，真的是一回事儿。”

00:25:50 - Saron Yitbarek:

那就给我们讲讲吧，什么是暗物质开发者呢？

00:25:52 - Sandra Persing:

本质上，有一些开发者，他们不参与你的聚会，不参与 GitHub 和在线社区，不为 Stack Overflow 做贡献。这些开发者仍在努力工作并作出贡献，但我们不知道。我们知道他们就在那里，但我们看不到他们。我们无法认出他们，而这些人实际上是开发者社区中非常重要的一部分，我们往往会忽略他们，但不能这样。忽略那些不愿发表意见的社区将损害我们与开发者的关系，我们需要更加积极主动地寻找宇宙中那些暗物质开发者。

00:26:34 - Saron Yitbarek:

哦，我喜欢。这很酷。是的，Jarod 实际上在 Twilio 工作，对吗？

00:26:38 - Sandra Persing:

是的。是的，他负责运营亚太区的 DevRel。

00:26:42 - Robyn Bergeron:

我从前在红帽的同事之一 Chris Grams，现在在一家名为 Tidelift 的公司里工作，
Dark Matter Matters
他曾经有一个名为暗 物 质 很 重 要的博客，因为它有点儿像是 ——

00:26:52 - Saron Yitbarek:

嗯嗯。

00:26:54 - Robyn Bergeron:

你知道，你所看不到的东西实际上仍然很重要，所以 ——

00:26:58 - Saron Yitbarek:

确实是这样。Robyn，你呢？你会用你的魔杖做什么呢？

00:27:03 - Robyn Bergeron:

哦，用我的魔杖，有很多事情，但是根据这次谈话，我想我会更好地管理彼此之间的依赖关系，而且也许并不令人惊讶，尤其是当我们之中有许多人在 OpenStack、OPNFV、Ansible 以及所有这些建立在彼此之上的东西工作时。只是确保我们项目

之间的关系比你筋疲力尽时所能看到的关系更加明显。因此，我非常期待来年的事情，因为我们在事物上越来越受关注。现在这非常令人兴奋。

00:27:39 - Saron Yitbarek:

吸引力总是那么好。它是如此令人兴奋。好吧，我想感谢你们所有人，非常感谢你们今天加入我们并分享你们的思维、想法和故事。非常感谢大家。那么现在说再见？

00:27:51 - Robyn Bergeron:

再见，各位。

00:27:54 - Sandra Persing:

很好。非常感谢你，Saron。能上这期节目很荣幸。

00:27:58 - Ricky Robinett:

它已经开始了，游戏工作室对于开源的态度已经开始转变。

00:27:59 - Saron Yitbarek:

确实。今天的圆桌会议包括红帽的社区架构师 Robyn Bergeron、Mozilla 的全球策略师 Sandra Persing 以及 Twilio 的开发者网络总监 Ricky Robinet。我认为自己很幸运，能有这样的平台让我分享对于我们社区将何去何从的愿景，无论是这档播客节目还是在其他地方，但我要指出的是，你无需拥有自己的播客就能够成为推广大使。成为推广大使只是意味着睁大眼睛并代表他人大声疾呼。这确实是每个人的工作。因此，我希望 Robyn、Sandra 和 Ricky 给了你一点儿有关倡导你觉得重要的事情的启发。

00:28:50 - Saron Yitbarek:

同时，《代码英雄》第 3 季已经在制作中了。当新剧集在今年春天推出时，你可以成为最早了解新剧集的人之一。如果你尚未注册，请在 Apple Podcasts、Google Podcasts 或任何你获得播客节目的地方进行订阅。只需点击一下，100% 免费。我是 Saron Yitbarek。非常感谢你的收听，在第 3 季到来之前，继续编程。

什么是 LCTT SIG 和 LCTT LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 LCRH SIG 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-2/developer-advocacy-roundtable>

作者: Red Hat 选题: bestony 译者: JonnieWayy 校对: windgeek, Northurland, wxy

本文由 [LCRH](#) 原创编译, [Linux中国](#) 荣誉推出

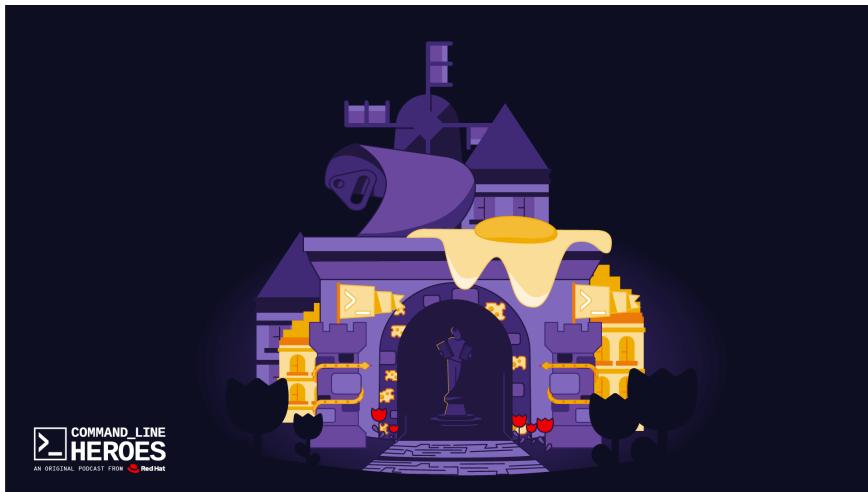
《代码英雄》第三季（1）：Python 的故事

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频播客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《[代码英雄](#)》系列播客第三季（1）：[Python 的故事](#)的[音频](#)脚本。

benevolent dictator for life

导语：一位“仁慈的终身独裁者（BDFL）”的辞职，永久改变了 Python 语言的命运，Guido van Rossum 那个名为《移交权利》的备忘录，让大家的注意力集中到了语言发展这件事情上。

在这一期里，Emily Morehouse 将 Python 技术层面的拓展性和它的包容性社区联系在了一起。Michael Kennedy 解释了 Python 是如何做到在简单易学的同时，又强大到足以担当的起像 Youtube 和 Instagram 这样的大项目。而 Diane Mueller 则强调了 Python 社区是如何率先在科技领域传播了许多包容性的实践——包括社区主导的决策的兴起。

有时候，一位仁慈的终身独裁者可以让一个语言开始发展，但 Python 证明了，是社区让语言兴旺起来的。

00:00:06 - Saron Yitbarek:

在 2018 年 7 月 12 日的早晨，世界各地的 Python 社区成员起床之后，他们拿起一杯咖啡，打开了自己的电脑。随后一个接着一个地，看见了一条来自他们仁慈的独裁者的消息。

00:00:24:

Guido van Rossum, 世界上最伟大的编程语言之一 Python 的创造者, 也许没有之一。他面向 Python 社区的全体成员写下这条消息, 我们不难想象出 Python 的粉丝们阅读这条消息时的画面。

00:00:38 - 多个配音演员:

我没想到在我已经如此努力的情况下, 仍有这么多人对我的决策不满。我想把自己从决策过程中完全抽离出来, 给自己一个永久的假期, 让自己一辈子都不做仁慈的独裁者。你们都要靠自己了, 我不会指定继任者。那么你们接下来要如何做呢, 创立民主制度吗? 或者保持无政府状态?

00:01:00 - Saron Yitbarek:

在这条简短却惊人的消息发布之后, **Guido van Rossum**, 这个被 Python 社区追随了几十年的人……好吧, 他基本上就这样退出了。这条消息被命名为
Transfer of Power
《移交权利》, 它将永久的改变 Python 的格局。

00:01:19:

不过除此之外, 它又引出一个问题, 那就是不光是 Python, 所有的这些编程语言在未来要怎样衍变与壮大呢? 是否应该由一个仁慈的独裁者把控, 赋予它们形状和一致性? 又或者, 在开源世界里, 编程语言的发展与变化是否应该更像口语, 取决于所有语言使用者的行为? Python 社区, 这个成长速度远超其他语言的社区, 需要找到这个问题的答案。

00:01:56:

我是 **Saron Yitbarek**, 这里是《代码英雄》第三季, 一个来自红帽的原创播客。在上一季的《代码英雄》里, 我们探索了广袤天地, 从[游戏主题](#)出发, 到[失败的艺术](#), 再到[无服务器开发](#), 最后我们甚至追踪了一台在火星表面, 来自 NASA 的[火星车](#)。但是真正抓住每个人想象力的一集, 似乎是[Grace Hopper 的故事](#)。她参与开发的编译器创造出了世上第一个高级编程语言, COBOL。我们随后意识到, Grace Hopper 的故事不过是万千个塑造了软件开发与运维的世界的故事之一。新的编程语言使得我们可以连通人类与机器, 它们为我们打开了一扇通往无限可能的大门。

00:02:51:

因此, 第三季将全部围绕这些语言展开。我们会谈论 JavaScript、BASIC、Go 以及 Perl, 当然, 还有 Python。我们的旅程将从 Python 开始, 顺着 Python 的故事, 我们会学到一个与所有编程语言未来相关的至关重要的事实。

00:03:16:

在 Python 的仁慈的独裁者放弃王座之后, 整个 Python 社区……呃, 有些迷茫。
Presbyterian church
你要如何在独裁者退位之后组织工作呢? 有人提议采用 长老会 结构, 不过没能奏效。要弄清楚 Python 是如何重新找到领导方式的, 并了解这对其它语言的未来有什么意义, 我们必须要回到 Python 故事的源头。

00:03:46 - Guido van Rossum:

好吧, 让我用 C 来完成全部代码, 这事情变得有些枯燥。

00:03:51 - Saron Yitbarek:

本尊驾到，Guido van Rossum，Python 的仁慈的独裁者。Van Rossum 曾在为阿姆斯特丹著名的 Centrum Wiskunde & Informatica 工作数年，在那里他协助开发了 ABC 编程语言。现在，他将描述他使用 C 工作时，发现对一个全新编程语言产生需求的时刻。

00:04:13 - Guido van Rossum:

仍然感觉还是有着很多 bug，并且开发速度缓慢。我在想，呃，如果这里支持 ABC 语言，我可以在十五分钟内写出整个登录程序，然后我可以接着开发账户管理之类的功能，但是用 C 进行开发，却要花掉我一整个星期。我开始设想，要提出一种可以在 Amoeba 环境中使用 ABC 语言特性的方法。

00:04:47 - Saron Yitbarek:

在研究编程语言的历史时我们发现，没有什么东西是全新的。它们全都是为了拼凑出解决方案而从旧语言借鉴来的。语言会不断的变化、发展、开枝散叶。当 van Rossum 被种种可能性弄得迷茫时，他设想出一个可以弥合 C 与 Shell 编程两者间隙的编程语言。一些场景下使用 C 语言往往是牛刀杀鸡，与此同时，用 Shell 脚本又常常让人感到繁琐，二者之间的最佳结合点，正是 Python 的栖身之处。当 van Rossum 在 1991 年首次发布 Python 时，它给大家都带来了启发，尤其是对系统管理员而言。这是一种前无来者，功能全面的脚本语言。

00:05:35 - Emily Morehouse:

第一次使用 Python 时，我立即就爱上了它。

00:05:39 - Saron Yitbarek:

这是 Emily Morehouse，目前 Python 担任核心开发者的五位女性之一。

00:05:44 - Emily Morehouse:

我认为当你从接触到的像 C++ 这样的第一门语言跨越到 Python 时，发现二者之间如此显著的差异，会真的会意识到语言与其设计的优美之处。你不必去应付内存管理的毛糙的底层实现细节，它可以既快又好的构建一些项目，更不必说它还有着如此广泛的应用领域。

00:06:18 - Saron Yitbarek:

Python 吸引力的关键之处，就在于它的可扩展性。举个例子，像 ABC 这样的语言，在设计上是固化单一的，无法形成一个能够真正帮助改善它的社区。反观 Python，van Rossum 在设计之初就希望 Python 是开放的、易于扩展的。

00:06:37 - Emily Morehouse:

进行软件设计时，你往往需要用到一些现有的软件或系统，并且要让它们能够协同工作，其实软件设计的核心价值之一就在于确保它的可扩展性。

00:06:58 - Saron Yitbarek:

听起来不怎么费脑子，但并非每一个语言都像 Python 这样，有着与生俱来的强可扩展性。事实上，一门不具备可扩展性的语言，大概率会随着自身的发展而崩溃。

00:07:16 - Emily Morehouse:

Python 有一个非常有趣的设计，它允许在其内核上进行扩展。实际上，你可以在运行时环境上对系统的不同片段打补丁，假如你想要改变模块导入的方式，或者你想改变你的字符串类型或整数类型，**Python** 允许你用相当容易的方式去实现这些事。

00:07:44:

Python 可扩展性的核心是所谓的 **C** 扩展，或者说是 **C** 模块。因此，**Python** 实际上被设计出来的意图，是为你提供对其他语言的切入点。本质上讲，如果你可以写出 **C** 扩展或 **C** 模块，可以用它去桥接其它数百种语言，那么你在某种角度上算是破解了 **Python** 的设计，再造了一个它。

00:08:16 - Saron Yitbarek:

这完全取决于用户根据自身需求调整语言的能力。所以说 **Python**，按照 **Guido van Rossum** 的设想，绝不会局限于独裁者的个人视野。他的《移交权利》备忘录由来已久。**Van Rossum** 明白社区影响力所能产生的力量，这是一种能够使所有人都聚集于同一顶“大帐篷”之下力量。没错，他最终被冠以独裁者称号，但却是一名仁慈的独裁者。

00:08:44 - Emily Morehouse:

我认为 **Python** 社区变得如此多元化，原因之一就是 **Guido**。**Python** 现在能够拥有女性核心开发者，正是因为 **Guido** 希望有这样的改变，并一手推动其实现。

00:09:01 - Saron Yitbarek:

Python 软件基金会的主席 **Naomi Ceder** 曾在一次演讲中说：“**Python**，因语言而来，为社区而留。”这可能正是 **Guido van Rossum** 留下的最棒的礼物。不仅仅是 **Python** 语言，他更是为整个 **Python** 社区都创造了空间。他不仅在语法意义上为 **Python** 提供了可扩展性，某种角度来讲，他也在社会意义上提供了扩展性。永远都为新来者留有位置。

00:09:29 - Emily Morehouse:

Python 有着大量不同的应用领域，而社区又如此多元化。。所以它真的、真的扩展了社区的边界。

00:09:51:

Emily Morehouse 是 **Python** 核心开发者与 **Cuttlesoft** 的工程总监。

00:09:59:

Python 降世之后，它就开始了前所未有的增长。我看到一个 **Stack Overflow** 上的图表，统计了每门语言被提及的次数，对应着 **Python** 的那条线仿佛坐上了火箭。2018 年，在 **Google** 搜索 **Python** 的人数比搜索金-卡戴珊的还多。另一个令人振奋的消息是，**Python** 开始和 **Java**、**C** 以及 **C++** 之类语言争夺最常用语言的头衔。

00:10:26:

那么，这些人们对 **Python** 的偏爱究竟来源何处呢？为了找到答案，我找到 **Michael Kennedy**，他活跃在 **Python** 时代的中心。他主持着两个专注于 **Python** 的播客：**Talk Python to Me** 和 **Python Bytes**。我们会在展示内容里放上对应的链接以便大家查看。**Michael** 将要和我一起讨论 **Python** 是如何真正迈出大步的。

00:10:52 - Michael Kennedy:

如果你看过分析报告或总结报告一类的东西，会发现 2012 年是一个重要拐点，而发生在 2012 年左右最具意义的事情莫过于，数据科学社区换掉了 R 之类的东西，开始真正专注于 Python。这件事发生以后，Python 社区就有了更充足的劲头，也有了更多的机器学习库。像许多流行的机器学习库，都会首先考虑使用 Python 来实现，此后才会考虑其他语言。

00:11:22 - Saron Yitbarek:

嗯，我也这么认为。当我想起 Python 时，我知道它可以被用作 Web 开发，我也认识许多仍然在使用 Python 来写 Web 应用的人。但我感到 Python 如今真正的核心更多是在数据科学领域。你认为是什么导致了这件事情发生？为什么数据科学社区抛弃了那些，或者我不应该叫抛弃，而是远离了 R 之类的工具。

00:11:44 - Michael Kennedy:

对，正是如此。

00:11:45 - Saron Yitbarek:

从何而来呢？

00:11:46 - Michael Kennedy:

我认为这种转变中是有两件事在起作用。其中之一就是 Python 作为一种“真正的”——引号起来的——编程语言，它既可以写一些简单的东西，例如构建图形或数据分析工具等等，但是，它也可以用来构建 Instagram 和 YouTube 以及其他类似的复杂应用……

00:12:08 - Saron Yitbarek:

是的，显而易见。

00:12:09 - Michael Kennedy:

相对于 R ……之类的东西而言。对，这很显而易见。很多应用都是由 Python 来编写的。所以他们之前在使用一些其它的编程语言时，例如 R，这门专精数据科学工作的，科学统计类的编程语言，如果他们想要写一个 Web 应用来展示他们的结果，那么，他们要用什么来实现呢？Node 或者 Python 还是……这导致他们没办法一直使用那些语言。

00:12:31 - Saron Yitbarek:

对，说的很好。

00:12:31 - Michael Kennedy:

所以说 Python 拥有相当棒的能力 …… 基本上，它是一种真正的编程语言，这是其一。第二点是，Python 十分独特，被我称为……全频谱语言。我所谓的全频谱的意思是指，假如我是一名生物学家、天文学家或别的什么，当我想要处理一些数据，想要加载一个 CSV 文件并运行一些指令得到图像时，我无需理解什么是类、静态方法、主函数、编译或者链接。你不需要像一些编程语言那样，为了入门而去学会这些。我只用写几行代码，键入指令，程序就可以运行。

00:13:14:

但是，你也可以用它去搭建像 Instagram 之类的软件。它可以成长为绝对专业的系统，你能使用，但是你不会在一开始就被逼迫着，去理解大型应用的一切深层抽象，你可以根据自身需要来使用它。这有意义吗？

00:13:30 - Saron Yitbarek:

对，对。说的非常贴切。刚才我们谈到 2012 年的拐点时，我搜索了一些关于 Python 的资料，Python 的确是世界上被 Google 次数最多的编程语言。

00:13:42 - Michael Kennedy:

哇。

00:13:43 - Saron Yitbarek:

你感觉它现在确实被大家青睐，并且仍在成长之中吗？

00:13:47 - Michael Kennedy:

我认为它真的被青睐并处于成长之中。在我们谈到的过去几年里，有相当多的企业集团都选择使用 Python。而他们曾经的选择是 .NET、Java，甚至是 C。这就是回答，并且现在 Python 开始走出属于自己的路，我认为它在某些场景下被格外倚重，关于这个，我指的是那些数据科学家们。就像，显然我们用到的 Jupyter Lab 那类炫酷笔记本应用，这些东西，全都是 Python 造就的。

00:14:17:

数据科学没有祖传代码的负担。如果我想开发一个新的项目，用来探索广告活动或科学结果，就不会对一堆陈旧的东西，保持着庞大的依赖关系。模型和数据都会过时，因此对于数据科学世界而言，切换技术栈或保持最新状态要容易得多。

00:14:37 - Saron Yitbarek:

说的不错。

00:14:39 - Michael Kennedy:

嗯，谢谢。

00:14:42 - Saron Yitbarek:

嗯，听上去 Python 不会很快停止成长，而是保持增长的趋势，甚至这种增长的势头仍在向前发展。你认为对这种增长与前进的势头影响最大的事情是什么呢？

00:14:52 - Michael Kennedy:

我感觉这就像滚雪球。Python 有如此多我们可以使用的库和软件包。现在更是多到难以置信。一两年前，这个数字大概只有 10 万。而现在我们已经有了 17 万个包或项目。

00:15:10 - Saron Yitbarek:

哇。

00:15:10 - Michael Kennedy:

你可以仅仅写几行代码，然后说，噢，我要做机器学习。有人在会议上向我们展示了一个例子，她们训练一个机器学习系统，给它提供大量人脸的图像，并告诉系统他们拥有什么类型的眼睛。是圆眼睛？或者椭圆形眼睛之类的东西。似乎这会帮助

你决定妆容之类的。

00:15:30 - Saron Yitbarek:

噢，哇。

00:15:31 - Michael Kennedy:

这位女士的演讲十分精彩，然后她说，这是训练模型的代码，让大家提问。而代码从头到尾只有十五行。

00:15:40 - Saron Yitbarek:

哇。

00:15:40 - Michael Kennedy:

接着你看，就像她说的那样，你提供一张照片，模型就会知道你的眼睛像什么。

00:15:44 - Saron Yitbarek:

噢，天哪。

00:15:45 - Michael Kennedy:

这种类型的项目势头十足。这些简洁，却又极具能量的东西，你可以直接通过一些包来导入，这真是厉害极了。

00:15:53 - Saron Yitbarek:

哇，这好酷。

00:15:54 - Michael Kennedy:

是不是很疯狂。

00:15:56 - Saron Yitbarek:

好吧。让我们暂停一下对话。我们会在稍后听到更多 Michael 的观点，但我现在想要回来强调一些事情。使 Python 这些令人惊叹的特质成为可能的源头，Python 社区。Python 成功的一个决定性因素就是这个庞大的、反应迅速的社区。

00:16:21:

此时此刻，正如我们在 van Rossum 离开时看到的那样，社区的规模已经庞大到让人难以接受。想象一下你不得不背着整个语言包袱的样子。某种角度来说，吸引了如此庞大的社区，让保有一个终生独裁者的主意变得站不住脚。Van Rossum 未曾设想他的语言会收到如此巨大的反响，但是，几乎完全自发的，社区成员将 Python 的邮箱列表、新闻组、网站，以及最终通过 PEP 讨论语言变化的流程汇集在一起。PEP 是 Python 增强建议的缩写。

00:17:05:

所以，尽管有着独裁者的名号，van Rossum 仍致力于搭建一个用户可以真正传达意见，参与构建的语言。我敢打赌，尽管他在离开的那一刻感到沮丧，但 van Rossum 明白，一个活跃的社区给他的语言带去的好处，要远胜他离去而折损的。

00:17:25 - Diane Mueller:

我叫 Diane Mueller。

00:17:27 - Saron Yitbarek:

Diane 是红帽云平台的社区开发总监。在过去的 30 年里，她目睹了开源社区的强劲发展，尤其是 Python 社区，给她留下了极为深刻的印象。

00:17:42 - Diane Mueller:

Python 社区做的太棒了……他们带来了大量行为准则的概念，关于会议、多样性奖金，所有这类东西。通过引入不同的声音以及不同的观点，我们得到了一个更好、更具创新性的项目，它必定能够留存得更久，并有望为更多的人带去更好的体验。

00:18:03:

即便他们犯错了，也会开诚布公的进行处理。在看到这种精神弥漫进硅谷和初创公司的兄弟文化中之后，通过与社区的合作，Python 就像带我回到了我创业的地方，围绕着它的社区也像是回到了往日。它真的相当鼓舞人心，非常惊艳。

00:18:33 - Saron Yitbarek:

能够鼓舞如此多的人，缘由是 Python 在最初就重新定义了成为社区一员的含义。我讲过 Guido van Rossum 即使是在引退之际，仍倡导了在社区里对女性的拥护。此外他也在更宽泛的意义上帮助了社区的拓展。

00:18:50 - Diane Mueller:

个人为集体带来的远远不止是代码贡献。绝大多数社区管理者或是项目主管，都把精力集中在如何促进大家为他们的项目做出贡献。而在 Python 社区里，大家会鼓励你参与文档工作，或是帮助会议的举办，以及对多样性的促进。这里有许多不同类型的事让你参与，都可以使你成为 Python 社区的一份子。

00:19:19:

所以这个想法，即社区贡献不仅仅局限于代码，像参与活动、相互学习和教育，以及大量的文档工作，这些对大多数人而言都是融入 Python 社区的方法。

00:19:35 - Saron Yitbarek:

当然，我们还有许多路要走。

00:19:37 - Diane Mueller:

精英阶层仍然注重于技术。没人会怀疑这个。但我想你也看见社区管理和社区管理者的理念了……我们都是社区重要的一份子，而不是被雇佣来干事的。

00:19:55:

对 Diane 而言，van Rossum 正式放弃他独裁者角色的决定，是全局变化的一部分。这代表着编程语言的建设开始摆脱陈旧与单一。

00:20:07:

我想我们可能已经摆脱了这种模式，尽管在一段时间里，我经常听见人说：“是的，我终生都是这个项目的仁慈独裁者。”但我并不认同。

00:20:21:

Diane Mueller 是红帽社区发展总监。

00:20:28:

在 **Guido van Rossum** 发出那份令人瞠目结舌的《移交权利》之后，**Python** 社区自身便成为了权力中心。随着项目的成熟发展，出现新的治理模式是正常的，并且正如我们所见，这些人已经做好准备，要来管理自己的语言了。但我仍然好奇，整件事最终是如何收场的？**van Rossum** 退场之后究竟发生了什么。让我们回到与 **Michael Kennedy** 的对话中来寻找一些答案吧。

00:20:59:

.....他离开了 **Python**，社区在没有他的情况下过得怎么样呢？

00:21:05 - Michael Kennedy:

嗯，社区的状态其实还不错，但我们陷在一个.....制高点，有点像是卡住了。运行时和语言基本上陷停滞不前。有一些有趣的提案，有些比较麻烦，有些又挺简单。比如，嘿，要不然我们每一年发布一次 **Python**，而不是每 18 个月，这样跟年度会议绑在一起，语言变化会变得更有可预测性。诸如此类的事情。

00:21:33 - Saron Yitbarek:

噢，嗯。

00:21:33 - Michael Kennedy:

这些都是无法做出定夺的。因为他引退之后，还没有方法去做决策。他基本上只留下一句，我去度假了，全靠你们了，你们必须得弄清楚怎么才能继续运行下去。我不会告诉你们怎样去做决策或怎么去运营。麻烦现在是你们的了。

00:21:48 - Saron Yitbarek:

哇。这听起来非常有戏剧性，但仔细一想。还记得那些 **Python Enhancement Proposals**

Python 增强建议 (PEP) 吗，那些用于接受社区反馈的 **PEP**？它们可以拯救大伙。

00:22:02 - Michael Kennedy:

有一系列他们试图为 **Python** 社区确立的新治理模型。好吧，最大的新闻是他们最 **Steering Council** 终决定选择其中之一，叫做指 导 委 员 会，委员会由五人组成——我相信他们都有着平等的投票权——他们最近选举出了五个人。。所以，这不是一个人的责任，而是所有人的责任。

00:22:23:

我感到十分高兴的一件事是，**Guido van Rossum** 也是他们的一员。他引退后说，我不能作为.....所有这些人们想要改变和反馈的压力的唯一的发泄口。但是他还 在，他没有完全抛弃这门语言。他仍然是一个核心开发者，并且加入了指导委员会。因此他仍然保留了发言权，但无需再独自承受一切，这真是非常棒。

00:22:47 - Saron Yitbarek:

我很好奇，这一切在现实世界里是怎么奏效的？我感觉如果我是指导委员会的一员，和语言的创造者坐在一起，我可能会倾向于赞同他所说的任何意见。

00:22:58 - Michael Kennedy:

对，确实如此。在平局的情况下，最终取决于 **Guido**。

00:23:03 - Saron Yitbarek:

嗯，的确。

00:23:03 - Michael Kennedy:

我不确定。但我认识指导委员会的一些人，他们一直以来都是社区的贡献者和开发者，可能超过了……在代码水平上，比 Guido 还多十五年。他们也都是非常深入的参与者，并且相当有主见，所以……

00:23:23 - Saron Yitbarek:

……而且相当投入。

00:23:25 - Michael Kennedy:

对，投入巨大。所以我认为这是行之有效的。另外，我觉得 Guido 的态度是，我仍然想参与其中，但不愿把自己的意志强加于人，否则一切都和原先没什么两样……我认为他可能倾向于选择一个更轻松的立场。

00:23:43 - Saron Yitbarek:

好的。我想知道，你是否认为在语言的初创阶段，为了项目能够启动运行，以便语言可以变得激进，从而得到那些突破性的好处，拥有一个
benevolent dictator for life

终生仁慈独裁者（BDFL）的管理模型是必要的？

00:24:00 - Michael Kennedy:

我认同。我有考虑过，大多数由委员会进行的设计，并不惊艳。在早期，有太多决策需要进行，语言如何工作？要不要用分号？怎么做这，怎么做那？所有这些内容都很难由委员会来决定，但是 Python 至今已经有 25 年的历史了。它有这么多人参与其中。我认为现在，它执行的是一个非常好的模型。

00:24:29 - Michael Kennedy:

他们还有过辩论，是否应该换上一个替补的 BDFL，这次我们又要选出谁来做我们的领袖呢？好在他们最后决定反对这个提议。

00:24:37 - Saron Yitbarek:

好。如果 BDFL 的位置是极其重要的，我想知道，社区需要依赖他多久呢？听上去 Guido 是自行做出决定：嘿，太过分了，这不是一个可以延续的状态，我不干了。但是，假如他没有做出这样的决策，我想知道，是否有一个最佳的时机让这个人退出，让大家走向一个更民主的状态呢？

00:25:01 - Michael Kennedy:

嗯，一定会有的，对吧？我认为这个时机应该存在。一个人难以与社区、技术的脉搏以及新的趋势一直保持紧密联系，说个数，大概 40 年。这是件极其困难的事，因此一定要有这个转换。我不能确切的说究竟在什么时候，但我认为必须得等到其他人相比 BDFL 做出了更多的贡献。随着核心贡献者和开发者越来越多，然后你就，嗯，我在度假，看看这些新的事情发生了，它还能活下来。类似这样的事情。

00:25:39 - Saron Yitbarek:

嗯。就好像是社区在准备好后会自己告诉你。

00:25:42 - Michael Kennedy:

对，正是如此。

00:25:48 - Saron Yitbarek:

由于 Python 社区仍在自己的生命历程中，因此这里就是我们暂时告一段落的地方。Michael Kennedy 的两个播客会持续追踪 Python 之后的历程。欢迎订阅 Talk Python to Me 和 Python Bytes。

00:26:07 - Saron Yitbarek:

Solon

你听说过被称为古代雅典立法者的梭伦的故事吗？他是个很酷的家伙。在梭伦为 Athenian democracy

民主雅典建立宪法之后，他选择了自行流放。因为他清楚，继续执政只会增加他成为暴君的风险。我觉得 Guido van Rossum 就像是当代梭伦，为我们提供了数十年的标准实践，有点像是一部宪法。他建立起一个出色的编程语言，一个真正由开源社区自己创作的语言。然后他给予他们一个权力转移的时刻，他在那时告诉他们，你们由自己掌控，我不再是你们的独裁者了。

00:26:54 - Saron Yitbarek:

他确保了一定是由社区，而非他本人，来推动 Python 前行。某种意义上，Guido van Rossum 的“移交权利”是开源世界中所有编程语言的共同宣言，因为任何语言随着其社区的发展，终将面临唯有社区才可以解决的挑战。

00:27:19 - Saron Yitbarek:

在《代码英雄》的第三季中，我们会对编程语言的世界进行深入的挖掘。语言影响力来源，正是它们如何通过强力的新方法去解决新的问题。在本季的剩余时间里，我们会探索 JavaScript、Perl、COBOL、Go，以及更多语言所具备的超能力。在下一集，我们会学习 BASIC 的故事，此外还会谈到母语究竟教会了我们什么。

00:27:47 - Saron Yitbarek:

如果你想更深入地研究 Python 或你在本集里听到的任何内容，请转至 redhat.com/commandlineheroes。最后，我是 Saron Yitbarek。直到下期，请坚持编程。

什么是 LCTT SIG 和 LCTT LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 LCRH SIG 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-3/pythons-tale>

作者: Red Hat 选题: bestony 译者: caiichenr 校对: acyanbird, wxy

本文由 LCRH 原创编译, Linux中国 荣誉推出

Guido van Rossum, 他是世界上最伟大的编程语言之一, **Python** 的创造者, 或许说是最伟大的那个也不为过。他面向 **Python** 社区的全体成员写下这条消息, 我们不难想象出 **Python** 的粉丝们阅读这条消息时的画面。

****00:00:38 - Multiple voice actors_**

我没想到在我已经如此努力的情况下, 仍有这么多人对我的决策不满。我情愿给自己放永远的长假, 并完全从决策的流程当中退出, 以免我毕生都作为仁慈的独裁者。你们好自为之, 我不会指定继任者。那么你们接下来要如何做呢, 创立民主制度吗? 或者保持无政府的状态?

00:01:00 - Saron Yitbarek:

在这条简短却惊人的消息发布之后, **Guido van Rossum**, 这个被 **Python** 社区追随足有数十年的人...算是基本退场了。这条消息被命名为“权力交接”, 它必定会对 **Python** 带来永久的改变。

00:01:19 - Saron Yitbarek:

不过除此之外, 它又引出一个问题, 那就是我们所有的这些编程语言在未来要怎样衍变与壮大呢? 是否应该由一个仁慈的独裁者把控, 来为语言塑形? 又或者, 在开源世界里, 编程语言的发展与变化应该更像口语, 取决于所有语言使用者的行为? **Python** 社区, 这个成长速度远超其他语言的社区, 需要找到这个问题的答案。

00:01:56 - Saron Yitbarek:

我是 **Saron Yitbarek**, 这里是代码英雄第三季, 一个来自 **Red Hat** 的原创播客。在上一季的代码英雄里, 我们探索了广袤天地, 从游戏主题出发, 到失败的艺术, 再到无服务器开发, 最后我们甚至追踪了一台在火星表面, 来自 **NASA** 的漫步者。但是真正抓住每个人想象力的一集, 似乎是 **Grace Hopper** 的故事。她参与开发的编译器创造出了世上第一个高级编程语言, **COBOL**。我们随后意识到, **Grace Hopper** 的故事不过是万千个塑造了软件开发与运维的世界的故事之一。新的编程语言使得我们可以连通人类与机器, 它们为我们打开了一扇通往无限可能的大门。

00:02:51 - Saron Yitbarek:

因此, 第三季将全部围绕这些语言展开。我们会谈论 **JavaScript**、**BASIC**、**Go** 以及 **Perl**, 当然, 还有 **Python**。我们的旅程将从 **Python** 开始, 顺着 **Python** 的故事, 我们会学到一个与所有编程语言未来相关的至关重要的事实。

00:03:16 - Saron Yitbarek:

在 **Python** 仁慈的独裁者放弃王座之后, 整个 **Python** 社区...呃, 有些迷失。你要 **Presbyterian church** 如何在独裁者退位之后组织工作呢? 有人提议采用长老会教堂结构, 不过没能奏效。要弄清楚 **Python** 重新组织的过程, 并了解这对其它语言的未来有什么意义, 我们必须要回到 **Python** 故事的源头。

****00:03:46 - Guido van Rossum_**

好吧, 我用 **C** 来完成全部代码, 这让事情有些枯燥。

00:03:51 - Saron Yitbarek:

本尊驾到, **Guido van Rossum**, **Python** 仁慈的独裁者。**Van Rossum** 先前数年都在为阿姆斯特丹著名的 **Centrum Wiskunde & Informatica** 工作, 在那里他协助 **ABC** 编程语言的开发。现在, 他将描述他使用 **C** 工作时, 发现对一个全新编程语

言产生需求的时刻。

****00:04:13 - Guido van Rossum_**

仍然感觉它有着大量的 bug，并且发展缓慢。我在想，呃，如果我们现在实现了 ABC，我可以在十五分钟内写出整个注册程序，然后我可以接着开发账户管理之类的功能，但是用 C 进行开发，却要花掉我一整个星期。我开始设想，要提出一种可以在 Amoeba 环境中使用 ABC 语言特性的方法。

00:04:47 - Saron Yitbarek:

在研究编程语言的历史时我们发现，没有什么东西是全新的。它们全都是为了拼凑出解决方案而从旧语言借来的。语言会不断的变化、发展、开枝散叶。当 van Rossum 被种种可能性弄得迷茫时，他设想出一个可以弥合 C 与 Shell 编程两者间隙的编程语言。一些场景下使用 C 语言往往是牛刀杀鸡，与此同时，用 Shell 脚本又常常让人感到繁琐，二者之间的甜点区，正是 Python 的栖身之处。当 van Rossum 在 1991 年首次发布 Python 时，它给大家都带来了启发，尤其是对系统管理员而言。这是一种前无来者，功能全面的脚本语言。

00:05:35 - Emily Morehouse:

首次使用 Python 的那一刻，我立即就爱上了它。

00:05:39 - Saron Yitbarek:

这是 Emily Morehouse，目前 Python 五位女性核心开发者之一。

00:05:44 - Emily Morehouse:

我认为当你从接触的首个类 C++ 语言跨越到 Python 时，发现二者之间如此显著的差异，会自然而然地意识到语言与其设计的优美之处。你不必去应付内存管理毛糙的底层实现细节，它可以既快又好的构建一些项目，更不必说它还有着如此广泛的应用领域。

00:06:18 - Saron Yitbarek:

Python 吸引力的关键之处就在于它的可扩展性。举个例子，像 ABC 这样的语言，设计坚若磐石，就无法形成一个能够真正帮助改善语言的社区。反观 Python，van Rossum 在设计之初就希望 Python 是开放，并易于扩展的。

00:06:37 - Emily Morehouse:

进行软件设计时，你往往需要用到一些现有的软件或系统，并且要让它们全都能够协同工作，其实软件设计的核心价值之一就在于确保它的可扩展性。

00:06:58 - Saron Yitbarek:

听起来不怎么费脑子，但并非每一个语言都像 Python 这样，有着与生俱来的强可扩展性。事实上，一门不具备可扩展性的语言，大概率会随着自身的发展而引发崩溃。

00:07:16 - Emily Morehouse:

Python 有一个非常有趣的设计，它允许在其内核上进行扩展。你可以在运行时上对系统的不同片段进行拼接，假如你想要切换如何导入模块的方式，或者你想切换你的字符串类型或整数类型，Python 会允许你用相当容易的方式去实现这些事。

00:07:44 - Emily Morehouse:

Python 可扩展性的核心是所谓的 C 扩展，或者说是 C 模块。因此，Python 实际上被设计出来的意图，是为你提供对其他语言的切入点。本质上讲，如果你可以写出 C 扩展或 C 模块，可以用它去桥接其他数百种语言，那么你在某种角度上算是破解了 Python。

00:08:16 - Saron Yitbarek:

这完全取决于用户根据自身需求调整语言的能力。所以说 Python，按照 Guido van Rossum 的设想，绝不会局限于独裁者的个人视野。他的“权力交接”停在备忘录上有相当长的一段时间。Van Rossum 明白社区影响力的力量，这是一种能够使所有人都聚集于同一顶“大帐篷”之下的力量。没错，他最终得名独裁者，但却是一名仁慈的独裁者。

00:08:44 - Emily Morehouse:

我认为 Python 社区变得如此多元化，原因之一就是 Guido。Python 现在能够拥有女性核心开发者，正是因为 Guido 希望此事发生，并一手推动其实现。

00:09:01 - Saron Yitbarek:

Naomi Ceder，Python 软件基金会的主席，她曾在一次演讲中说：“Python。因语言来，为社区留。”这可能正是 Guido van Rossum 留下的最棒的礼物。不仅仅是为 Python 语言，他更是为整个 Python 社区都留下了空间。他不仅在语法意义上为 Python 提供了可扩展性，某种角度来讲，他也在社会意义上提供了扩展性。永远都为新来者留有位置。

00:09:29 - Emily Morehouse:

Python 有着大量不同的应用，而社区又如此多元化。这实在是，充分扩展了社区的边界。

00:09:51 - Saron Yitbarek:

Emily Morehouse 是 Python 核心开发者与 Cuttlesoft 的工程总监。

00:09:59 - Saron Yitbarek:

Python 降世之后，成长便空前绝后。我看到一个 Stack Overflow 上的图表，统计了每门语言被提及的次数，对应着 Python 的那条线仿佛坐上了火箭。2018 年，在 Google 搜索 Python 的人数超过了搜索 Kim Kardashian 的。另一个令人振奋的消息是，Python 开始和 Java, C 以及 C++ 之类语言争夺最常用语言的头衔。

00:10:26 - Saron Yitbarek:

那么，这些偏爱究竟来源何处呢？为了找到答案，我找到位于 Python zeitgeist 中心的开发者 Michael Kennedy。他主持着两个专注于 Python 的播客：Talk Python to Me 和 Python Bytes。我们会在展示内容里放上对应的链接以便大家查看。

Michael 将要和我一起讨论 Python 是如何真正迈出大步的。

00:10:52 - Michael Kennedy:

如果你看过分析报告或总结报告一类的东西，会发现 2012 年是一个重要拐点，而发生在 2012 年左右最具意义的事情莫过于，数据分析社区换掉了 R 之类的东西，开始真正专注于 Python。这件事发生以后，Python 社区就有了更充足的劲头，也有了更多的机器学习库。像许多流行的机器学习库，都会首先考虑使用 Python 来实现，此后才会考虑其他语言。

00:11:22 - Saron Yitbarek:

嗯，我也这么认为。当我想到 Python 时，我知道它可以被用作 web 开发，我也认识许多仍然在使用 Python 来写 web 应用的人。但我感到 Python 如今真正的核心更多是在数据分析领域。你认为是什么导致了这件事情发生？为什么数据分析社区抛弃了那些，或者不应该叫抛弃，替换掉了 R 之类的工具。

00:11:44 - Michael Kennedy:

对，正是如此。

00:11:45 - Saron Yitbarek:

从何而来呢？

00:11:46 - Michael Kennedy:

我认为这种迁移的发生，是有两件事在起作用。其中之一就是 Python 作为一种“真正的”编程语言，它既可以写一些简单的东西，例如构建图形或数据分析工具等等，但是，它也可以用来构建 Instagram 和 YouTube 以及其他类似的复杂应用...

00:12:08 - Saron Yitbarek:

是的，显而易见。

00:12:09 - Michael Kennedy:

像 R...之类的东西。对，相当的显而易见，它们都是由 Python 来编写的。所以他们之前在使用一些其它的编程语言时，例如 R，这门专精数据科学工作的，科学统计类的编程语言，如果他们想要写一个 web 应用来展示他们的结果，那么，他们要用什么来实现呢？Node 或者 Python 还是...这导致他们没办法坚持使用那些语言。

00:12:31 - Saron Yitbarek:

对，说的很好。

00:12:31 - Michael Kennedy:

所以说 Python 拥有相当棒的能力...基本上，它是一种真正的编程语言，这是其一。第二点是，Python 十分独特，被我称为...全光谱语言。我所谓的全光谱的意思是指，假如我是一名生物学家、天文学家或别的什么，当我要处理一些数据，想要加载一个 CSV 文件并运行一些指令得到图像时，我无需理解什么是类、静态方法、主函数、编译或者链接。我不必在刚开始就学会这些...其它语言必须要做的一系列事。我只用写几行代码，键入指令，程序就可以运行。

00:13:14 - Michael Kennedy:

但是，你也可以用它去搭建像 Instagram 之类的软件。它可以成长为你所使用的绝对专业的系统，但是你不会在一开始就被逼迫着，去理解大型应用的一切深层抽象，你可以根据自身需要来使用它。这样说的通吗？

00:13:30 - Saron Yitbarek:

对，对。说的非常有道理。刚才我们谈到 2012 年的拐点时，我搜索了一些关于 Python 的资料，Python 的确是世界上被 Google 次数最多的编程语言。

00:13:42 - Michael Kennedy:

哇。

00:13:43 - Saron Yitbarek:

你感觉它现在确实被大家青睐，并且仍在成长之中吗？

00:13:47 - Michael Kennedy:

我认为它真的被青睐并处于成长之中。在我们谈到的后几年里，有相当多的企业集团都选择使用 Python。而他们曾经的选择是 .NET, Java, 甚至是 C。这就是回答，并且现在 Python 开始走出属于自己的路，我认为它在某些场景下被格外倚重，关于这个，我指的是那些数据科学家们。就像，显然我们用到的 Jupyter Lab 还有那些炫酷笔记本之类的东西。全都是 Python。

00:14:17 - Michael Kennedy:

数据科学没有祖传代码的负担。如果我想开发一个新的项目，用来探索广告活动或科学结果，就不会对一堆陈旧的东西，保持着庞大的依赖关系。模型和数据都会过时，因此对于数据科学世界而言，切换技术栈或保持最新状态要容易得多。

00:14:37 - Saron Yitbarek:

说的不错。

00:14:39 - Michael Kennedy:

嗯，谢谢。

00:14:42 - Saron Yitbarek:

嗯，听上去 Python 不会在近期停止成长，而是保持增长的趋势，甚至这种增长的势头仍在向前发展。你认为对这种增长与前进的势头影响最大的事情是什么呢？

00:14:52 - Michael Kennedy:

我感觉这就像滚雪球。Python 有如此多我们可以使用的库和软件包。现在更是多到难以置信。一两年前，这个数字大概只有 10 万。而现在我们已经有了 17 万包或项目。

00:15:10 - Saron Yitbarek:

哇。

00:15:10 - Michael Kennedy:

你可以仅仅写几行代码，然后说，噢，我要做机器学习。有人在会议上向我们展示了一个例子，她们训练一个机器学习系统，给它提供大量人脸的图像，并告诉系统他们拥有什么类型的眼睛。是圆眼睛？或者椭圆形眼睛，之类的东西。似乎这会由你的妆容或别的什么来驱动。

00:15:30 - Saron Yitbarek:

噢，哇。

00:15:31 - Michael Kennedy:

这位女士的演讲十分精彩，然后她说，这是训练模型的代码，让大家提问。而代码从头到尾只有十五行。

00:15:40 - Saron Yitbarek:

哇。

00:15:40 - Michael Kennedy:

接着你看，就像她说的那样，你提供一张照片，模型就会知道你的眼睛像什么。

00:15:44 - Saron Yitbarek:

噢，天哪。

00:15:45 - Michael Kennedy:

这种类型的项目势头十足。这些简洁，却又极具能量的东西，你可以直接通过一些包来导入，这真是厉害极了。

00:15:53 - Saron Yitbarek:

哇，这好酷。

00:15:54 - Michael Kennedy:

这难道不疯狂吗？

00:15:56 - Saron Yitbarek:

好吧。让我们暂停一下对话。我们会在稍后听到更多 Michael 的观点，但我现在想要回头来强调一些事情。使 Python 这些令人惊叹的特质成为可能的源头，Python 社区。Python 成功的一个决定性因素就是这个庞大的，有影响力的社区。

00:16:21 - Saron Yitbarek:

此时此刻，正如我们在 van Rossum 离开时看到的那样，社区的规模已经过于庞大。想象一下你不得不背着整个语言包袱的样子。某种角度来说，吸引了如此庞大的社区，让保有一个终生独裁者的主意变得站不住脚。Van Rossum 未曾设想他的语言会收到如此巨大的回应，但是，几乎完全自发的，社区成员将 Python 的邮箱列表、新闻组、网站，最后还有通过 PEPs 讨论语言变化的流程汇集在一起。这些 Python Enhancement Proposals 内容共同组成了 Python 增强建议。

00:17:05 - Saron Yitbarek:

所以，尽管有着独裁者的名号，van Rossum 仍致力于搭建一个用户可以真正传达意见，参与构建的语言。我敢打赌，尽管他在离开的那一刻感到沮丧，但 van Rossum 明白，一个活跃的社区给他的语言带来的好处要远胜他离去而折损的。

00:17:25 - Diane Mueller:

我叫做 Diane Mueller。

00:17:27 - Saron Yitbarek:

Diane 是 Red Hat 云平台的社区开发总监。在过去的 30 年里，她目睹了开源社区的强劲发展，尤其是 Python 社区，给她留下了极为深刻的印象。

00:17:42 - Diane Mueller:

Python 社区做的太棒了... 他们带来了大量行为准则的概念，关于会议、不同奖金、所有这类东西。通过不同的声音以及不同的观点，我们得到了一个更好，更具创新性的项目，它必定能够留存得更久，并有望为更多的人带去更好的体验。

00:18:03 - Diane Mueller:

即便他们犯错了，也会开诚布公的进行处理。在看到这种精神弥漫进硅谷和初创公司的兄弟文化中之后，通过与社区的合作，Python 就像带我回到了我刚起步的地方，围绕着它的社区也像是回到了往日。它真的相当鼓舞人心，非常惊艳。

00:18:33 - Saron Yitbarek:

能够鼓舞如此多的人，缘由是 Python 在最初就重新定义了成为社区一部分的含义。我讲过 Guido van Rossum 即使是在引退之际，仍倡导了在社区里对女性的拥护。此外他也在更宽泛的意义上帮助了社区的拓展。

00:18:50 - Diane Mueller:

个人为集体带来的远远不止是代码贡献。绝大多数社区管理者或是项目主管，都把精力集中在如何促进大家为他们的项目做出贡献。而在 Python 社区里，大家会鼓励你参与文档工作，或是帮助会议的举办，以及对多元性的促进。这里有许多不同类型的事让你参与，都可以使你成为 Python 社区的一份子。

00:19:19 - Diane Mueller:

所以这个想法，即社区贡献不仅仅局限于代码，像参与活动、相互学习和教育，以及大量的文档工作，这些对大多数人而言都是融入 Python 社区的方法。

00:19:35 - Saron Yitbarek:

当然，我们还有许多路要走。

00:19:37 - Diane Mueller:

精英阶层仍然着重于技术。没人会怀疑这个。但我想你也看见社区管理和社区管理者的理念了...我们都是社区重要的一份子，而不是被雇佣来干事的。

00:19:55 - Saron Yitbarek:

对 Diane 而言，van Rossum 正式放弃他独裁者角色的决定，是全球运转的一部分。这代表着编程语言的构建过程开始摆脱陈旧与单一。

00:20:07 - Saron Yitbarek:

我认为我们会从这个模式中前进，尽管在一段时间里，我经常听见人说：“是的，我终生都是这个项目的仁慈独裁者。”但我并不认同。

00:20:21 - Saron Yitbarek:

Diane Mueller 是 Red Hat 社区发展总监。

00:20:28 - Saron Yitbarek:

在 Guido van Rossum 发出那条令人瞠目结舌的消息之后，Python 社区自身便成为了权力中心。随着项目的成熟发展，出现新的管理模式是正常的，并且正如我们所见，这些家伙已经做好准备，要来管理自己的语言了。但我仍然好奇，整件事最终是如何收场的？van Rossum 退场之后究竟发生了什么。让我们回到与 Michael Kennedy 的对话中来寻找一些答案吧。

00:20:59 - Saron Yitbarek:

... 离开了 Python，社区在没有他的情况下过得怎么样呢？

00:21:05 - Michael Kennedy:

嗯，社区的状态其实还不错，但我们陷在一个...制高点，有点像是卡住了。运行时和语言基本上陷停滞不前。有一些有趣的提案，有些比较麻烦，有些又挺简单。比如，嘿，要不然我们每一年发布一次 Python，而不是每 18 个月，这样跟年度会议绑在一起，语言变化会变得更有可预测性。诸如此类的事情。

00:21:33 - Saron Yitbarek:

噢，嗯。

00:21:33 - Michael Kennedy:

这些都是无法做出定夺的。因为在他们引退之后，还没有方法去做决策。他基本上只留下句，我去度假了，全靠你们了，你们必须得弄清楚怎么才能继续运行下去。我不会告诉你们怎样去做决策或如何去运营。麻烦现在是你们的了。

00:21:48 - Saron Yitbarek:

哇。这听起来非常有戏剧性，但仔细一想。还记得那些

Python Enhancement Proposals

Python 增 强 建 议 吗，那些用于接受社区反馈的 PEPs？它们可以拯救大伙。

00:22:02 - Michael Kennedy:

这里有一系列他们试图为 Python 社区确立的新治理模型。大新闻是他们最终决定 **Steering Council**

选择其中之一，叫做指 导 委 员 会，委员会由五人组成——我相信他们都有着平等的投票权——他们最近选举出了五个人。所以，相比于站在一个人的肩膀上，现在取而代之的方法是站在所有人的肩膀上。

00:22:23 - Michael Kennedy:

我感到十分高兴的一件事是，Guido van Rossum 也是他们的一员。他引退后说，我不能作为唯一的发泄口...所有这些人们想要改变和反馈的压力。但是他还在这里，他没有完全抛弃这门语言。他仍然是一个核心开发者，并且加入了指导委员会。因此他仍然保留了发言权，但无需再独自承受一切，这真是非常棒。

00:22:47 - Saron Yitbarek:

我很好奇，这一切在现实世界里是怎么奏效的？我感觉如果我是指导委员会的一员，和语言的创造者坐在一起，我可能会倾向于赞同他所说的任何意见。

00:22:58 - Michael Kennedy:

对，确实如此。所有事情制衡的情况下，最终取决于 Guido。

00:23:03 - Saron Yitbarek:

嗯，的确。

00:23:03 - Michael Kennedy:

我不确定。但我认识指导委员会的一些人，他们一直以来都是社区的贡献者和开发者，可能超过了...代码层面上，超过了 Guido 十五年。他们也都是非常深入的参与者，并且相当自以为是，所以...

00:23:23 - Saron Yitbarek:

而且相当投入。

00:23:25 - Michael Kennedy:

对，投入巨大。所以我认为这是行之有效的。另外，我觉得 Guido 的态度是，我仍然想参与其中，但不愿把自己的意志强加于人，否则一切都和原先没什么两样...我认为他可能倾向于选择一个更轻松的位置。

00:23:43 - Saron Yitbarek:

好的。我想知道，你是否认为在语言的初创阶段，为了项目能够启动运行，以便语言可以变得激进，从而得到那些突破性的好处，拥有一个
benevolent dictator for life
终生仁慈独裁者（BDFL）的管理模型是必要的？

00:24:00 - Michael Kennedy:

我认同。我有考虑过，大多数由委员会进行的设计，并不惊艳。在早期，有太多决策需要进行，语言如何工作？要不要用分号？怎么做这，怎么做那？所有这些内容都很难由委员会来决定，但是 Python 至今已经有 25 年的历史了。它将太多人卷入其中，我认为现在，它执行的是一个非常好的模型。

00:24:29 - Michael Kennedy:

他们还有过辩论，是否应该换上一个替补的 BDFL，这次我们又要选出谁来做我们的领袖呢？好在他们最后决定反对这个提议。

00:24:37 - Saron Yitbarek:

好。如果 BDFL 的位置是极其重要的，我想知道，社区需要依赖他多久呢？听上去 Guido 是自行做出决定：嘿，太过分了，这不是一个可以延续的状态，我不要再这样做了。但是，假如他没有做出这样的决策，我想知道，是否有一个最佳的时机让这个人退出，让大家走向一个更民主的状态呢？

00:25:01 - Michael Kennedy:

嗯，一定会有的，对吧？我认为这个时机应该存在。一个人难以与社区、技术的脉搏以及新的趋势一直保持紧密联系，说个数，大概 40 年。这是件极其困难的事，因此务必要进行这个转换。我不能确切的说究竟在什么时候，但我认为必须得等到其他人相比 BDFL 做出了更多的贡献。随着核心贡献者和开发者越来越多，然后你就，嗯，我在度假，看着发生的新鲜变化，它们都幸存了下来。这种效果。

00:25:39 - Saron Yitbarek:

嗯。就好像是社区在准备好后会自己告诉你。

00:25:42 - Michael Kennedy:

对，正是如此。

00:25:48 - Saron Yitbarek:

由于 Python 社区仍在自己的生命历程中，因此这里就是我们暂时告一段落的地方。Michael Kennedy 的两个播客会持续追踪 Python 之后的历程。欢迎订阅 Talk Python to Me 和 Python Bytes。

00:26:07 - Saron Yitbarek:

你听说过被称为古代雅典立法者的梭伦的故事吗？他是个很酷的家伙。在梭伦为 **Athenian democracy**

民主雅典建立宪法之后，他选择了自行流放。因为他清楚，继续执政只会增加他成为暴君的风险。我觉得 **Guido van Rossum** 就像是当代梭伦，为我们提供了数十年的标准实践，有点像是一部宪法。他建立起一个出色的编程语言，一个真正由开源社区自己创作的语言。然后他给予他们一个权力转移的时刻，他在那时告诉他们，你们由自己掌控，我不再是你们的独裁者了。

00:26:54 - Saron Yitbarek:

他确保了一定是由社区，而非他本人，来推动 **Python** 前行。某种意义上，**Guido van Rossum** 的“权力转移”是开源世界中所有编程语言的共同宣言，因为任何语言随着其社区的发展，终将面临唯有社区才可以解决的挑战。

00:27:19 - Saron Yitbarek:

在代码英雄的第三季中，我们会对编程语言的世界进行深入的挖掘。语言影响力来源，正是它们如何通过强力的新方法去解决新的问题。在本季的剩余时间里，我们会探索 **JavaScript**, **Perl**, **COBOL**, **Go**, 以及更多语言所具备的超能力。在下一集，我们会学习 **BASIC** 的故事，此外还会谈到母语究竟教会了我们什么。

00:27:47 - Saron Yitbarek:

如果你想更深入地研究 **Python** 或你在本集里听到的任何内容，请转至 redhat.com/commandlineheroes。最后，我是 **Saron Yitbarek**。直到下期，请坚持编程。

什么是 LCTT SIG 和 LCTT LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 **LCRH SIG** 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-3/pythons-tale>

作者: [Red Hat](#) 选题: [bestony](#) 译者: [caichenr](#) 校对: [acyanbird, wxy](#)

本文由 [LCRH](#) 原创编译, [Linux中国](#) 荣誉推出

《代码英雄》第三季（2）：学习 BASIC

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频播客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《代码英雄》系列播客第三季（2）：学习 BASIC 的音频脚本。

导语：以前，成为程序员需要博士学位，还要能够接触到像服务器之类有着举足轻重地位的硬件。直到 1965 年，一群工程师有了个激进的想法：让编程变得更容易、亲民。

像 BASIC 这样专为初学者设计的语言为许多人打开了编程世界的大门。Tom Cormen 和 Denise Dumas 回忆了 BASIC 是如何改变一切的。Avi Flombaum 和 Saron 则会向这个软件开发新时代的编程新手们提供关于挑选第一门语言的建议。最后，Femi Owolade-Coombes 和 Robyn Bergeron 会向我们讲述，新一代的程序员们，是如何通过游戏来接触编程的。

这些初学者语言的诞生，让每个人都有机会踏入编程的大门。同时，这也让整个产业欣欣向荣。

00:00:02 - Saron Yitbarek:

1965 年是美国历史上发生重大变化的一年。《民权法案》就是在那年签署的。

00:00:09 - 新闻档案:

国会通过了史上影响最为广泛的《民权法案》，并将它写进了法律里。

00:00:12 - Saron Yitbarek:

作战部队被第一次被派往越南，男人们烧掉了他们的征兵证。

00:00:19 - 新闻档案:

征兵名单被放在波士顿法院的台阶上，一群高中男生挥拳抗议。

00:00:25 - Saron Yitbarek:

在纽约，披头士乐队举行了世界上第一场体育馆内的音乐会。但在距离这些大新闻很远的地方，一场与众不同的革命正在酝酿。

00:00:39:

John Kemeny

Dartmouth

5月1号，凌晨4点。约翰·凯梅尼教授和一名本科生正在达特茅斯的GE 225型电脑上忙碌着。他们运行了一个方才写好的程序。电传打字机的三行输出，永远地改变了计算机编程领域。

00:01:04:

欢迎来到红帽原创播客《代码英雄》的第三季第2集。我是主持人，Saron Yitbarek。在这一集，我们将继续“编程语言的过去与未来”的旅程。在第1集中，我们深入研究了Python，并了解了开源社区如何在其领导阶层的大动荡中生存下来。比起上一期节目，本次节目则会更注重于个人。我会先聊聊我邂逅入门语言时的体验，这种体验我们应该都有。我将带领你们领略我学习的第一门语言Ruby，以及某些新手语言是如何用游戏的形式，来到我们身边的。

00:01:50:

不过，我得先举个例子，好让大家都明白“新手语言”到底是什么意思。这个例子就是BASIC，它诞生于之前提到的达特茅斯实验室的三行输出。在1965年的那个黎明，约翰·凯梅尼作为创造者之一，见证了BASIC的诞生。

00:02:14:

Beginner's All Purpose Symbolic Instruction Code

BASIC的意思是“初学者的泛用符号指令代码”。这里面的第一个词“初学者”正是其不凡之处所在。在那时候，即使是少量的编程工作，也需要一名博士来完成。这是一种极高的要求，需要有人接受专业教育，更不用说那些高级Kemeny的硬件了。但是凯梅尼教授和他的团队希望改变现状。他们想知道：如果要编写一种所有人都可以使用的编程语言，那会是什么样的？当美国各地的青年们剧烈地抗议，要求变革时，达特茅斯团队提出了一种另类的革命。

00:02:57:

他们的秘密武器是一个房间那么大的大型电脑——GE 225。它重达2000磅，和这份不得了的重量相匹配的是，它具备某些全新的功能。那就是分时功能(LCTT译注：也就是时间片，实现了多用户多进程快速切换的技术）。突然间，编程不再需要复杂的打孔卡系统或庞大的开关墙了。分时意味着程序们都可以同时执行，因为系统会根据不同用户的需要来回切换其处理内容。

00:03:31:

Dartmouth

分时意味着使用者可以用惊人的新方式来访问计算机。这是来自达特茅斯的些许录音，记录了 BASIC 刚诞生的时候，65 班的一位本科生 John McGeachie 学习的实况。

00:03:47 - John McGeachie:

我们把这台相当贵的、原本只能一次让一个人用的计算机变成了可以……让超过 30 人使用的东西。30 名本科生可以同时在这上面编程，还能快速得到输出。它是史无前例的简洁直接。

00:04:15 - Saron Yitbarek:

John Kemeny **Thomas Kurtz**
约翰·凯梅尼联合另一位计算机文化的布道师托马斯·卡茨共同开发了 BASIC 语言。他们发现，分时让计算机使用变得更亲民，因此准入门槛降低了许多。而剩下的工作则是编写一门简单直白的编程语言。一门像 BASIC 那样的语言。

00:04:39:

他们开发了像 `HELLO` 和 `GOODBYE` 这样的指令，来代替 `LOG ON` 和 `LOG OFF` 命令。事实上，初版 BASIC 只有 14 个简单的指令，比如条件判断 `IF`，和 `GOTO` 之类的简单选项。

00:04:54 - Tom Cormen:

Dartmouth
我是 Tom Cormen。我是达特茅斯大学的一名计算机科学教授。

00:04:59 - Saron Yitbarek:

Cormen 教授将与我们谈论所谓“初学者的革命”，以及 BASIC 的出现如何成为时代精神的一部分。它带来了一个新世界，在其中，计算机技术对更多人来说不仅不再遥不可及，甚至还是激动人心的——编程甚至变得有点时髦了。

00:05:14 - Tom Cormen:

据说在 1960 年时，男学生会在这里（计算机中心）约会，我真是不能想象如果发生在现在会是什么样子。但在那个时候，计算机中心一度是约会的最佳场所。

00:05:24 - Saron Yitbarek:

这种在校园兴起的编程热潮，是“初学者语言”诞生所带来的直接结果。

00:05:31 - Tom Cormen:

就如同我们过去说的那样，BASIC 为那些计算机爱好者降低了准入门槛。不想学习 FORTRAN 这类编程语言的社会科学家可以使用 BASIC。从事人文艺术的人们可以用它做文本处理，甚至用来创作艺术作品，这些完全可以用 BASIC 来完成。

00:05:55:

没过几年，人们就开始写电脑游戏，也时常在编程中实现 GUI。

00:06:03 - Saron Yitbarek:

对于所有那些认为自己不太可能成为程序员的孩子们，编程领域的的大门，突然间以 **Grace Hopper** 一种近乎直观的方式打开了。它使得我想起来葛丽丝·哈伯（LCTT 译注：参见上一季。元祖级程序员，美国海军准将，COBOL 的重要编写者。她也是世界上第一个在计算机领域使用“bug”这个词的人。）所追求的那种世界。当我们上一季谈到

Hopper

哈 伯时，我们谈到了她的语言创新如何将编程带到更多人眼前。BASIC 的此时此刻就像是哈 伯梦想的延续。

00:06:29 - Tom Cormen:**Grace Hopper**

我非常确信，如果葛丽丝·哈伯能看到更多人开始写代码，她会非常高兴。她可能会喜欢 BASIC 和 COBOL 的不同之处，因为这种不同为人们提供了更多的选择。他们可以写 COBOL，可以写 FORTRAN，可以写 BASIC，可以写 ALGOL，可以选择当时任何流行的语言。

00:06:54 - Saron Yitbarek:**Dartmouth**

Tom Cormen 是达特茅斯计算机科学系的一名教授。当初，计算机技术中的几大变化催生了新一代的程序员。分时功能使并发工作成为可能，而 BASIC 使入门编程变得简单了。这两个因素结合在一起，创造出了改变游戏规则的星星之火。很

Bill Gates

快，编程就不仅是大型机构的专利。这一代程序员中有像比尔·盖茨和

Steve Wozniak

史蒂夫·沃兹尼亚克这样的特立独行者，也有在个人电脑上每天默默耕耘的开发者。

00:07:32 - Denise Dumas:

BASIC 能够在程序出错时立即给使用者以反馈。这让它的使用体验变得不同，它将你吸在一起，使你继续前进；这种交互是有意义的，就因为它是一种交互。

00:07:50 - Saron Yitbarek:**Red Hat® Enterprise Linux®**

这位是 Denise Dumas，她带领“红帽企业版操作系统”工程师团队。

00:07:55 - Denise Dumas:

我认为 BASIC 使编程变得民主了许多，因为 BASIC 把它自身交予学生、大众，当时的其他语言——像是 COBOL、FORTRAN 之类——根本做不到这一点。当 CRT（阴极射线管）流行的时候，BASIC 更为火爆，因为在你输入了信息之后，输出不再是被打印在一张纸上，而是显示在你面前的屏幕上。

00:08:26 - Denise Dumas:

我们现在有时候会嘲笑命令行，对吧？一切都图形化了，命令行还是一如往常，但它比命令行之前的时代有了很大的改进。这一重大改进就是，你输入指令能够立即得到回应了。我认为 BASIC 降低了准入门槛，虽然我是那种控制狂，我喜欢汇编，但我还是会这么说的。BASIC 使编程变得更加平易近人。

00:08:59 - Saron Yitbarek:

BASIC 在 1980 年代带来的这份激情一直延续到了今日。现在，有许多语言能被初学者用作进入编程世界的光明大道。但如今发生了另一些重要的改变：编程入门不再只有进入大学学习一途。如果将入门的方式比作道路，那么，今日，条条大路通罗马。

00:09:29:

Grace Hopper

尽管 BASIC 很神奇，但葛丽丝·哈伯的梦想不会止步于此。如今，初学者有上百种方法可以开始编程。就我而言，我是在纽约市一家名为 Flatiron 的编程学校学习的第一门语言。我为那个项目准备了几个月，我阅读了所有可能的编程资源和教程，以确保它们也许能让我在训练营中获得最大收益。我想了解当今人们是如何接触自己的入门语言的。所以，今天我采访了自己当初的那位老师。

00:10:03 - Avi Flombaum:

我是 Avi Flombaum。

00:10:04 - Saron Yitbarek:

Avi 是 Flatiron 学校的创始人之一。对我来说，探究所有有关入门语言的思想，以及我们对入门语言的方法与 BASIC 的旧时代相比有什么变化，是一种享受。

00:10:26:

当你教别人该如何编程时，先从哪里下手呢？我记得当我学习第一门编程语言时，我觉得这一切可真是太奇怪了。我在大学读的是英语专业，在学校里我很喜欢数学，也喜欢有机化学。我喜欢很多不同类型的科目，各种各样的东西，但是没有什么是和编程有关的。因此，我也没办法将编程比作什么。它自成一套，就像一个自己的小世界。所以，作为老师，你会从哪里开始呢？

00:10:49 - Avi Flombaum:

我认为生活中你所熟悉的任何事物，曾经都是陌生的。只有经历过，真正花时间去做，才会让你对它感到习惯。这是不断重复练习的问题，只关乎你投入的时间的多少。我认为编程之所以困难，之所以让人感觉如此陌生，是因为它用极其特殊的语法来表达一些东西。没有任何歧义，没有错误的余地。要么成功，要么失败。当我们互相交流的时候，我们一般没有那种要求。一切都必须完全正确。一个逗号丢了，一个括号缺了，就全坏了。

00:11:32:

我总是说，计算机是愚蠢的，为了使计算机理解我们的意思，我们必须做到完美。

00:11:37 - Saron Yitbarek:

你说的这一点我喜欢。我同意，这很棒。

00:11:43 - Avi Flombaum:

是啊。

00:11:44 - Saron Yitbarek:

对我来说，我喜欢的入门语言是 Ruby。根据经验，我可以告诉你，这是一种非常可爱的入门语言。当你阅读代码时，感觉就像你在阅读英语。它有很多不错的库；

Yukihiro Matsumoto

这里插播一则知识补充。Ruby 的创造者 松 本 行 弘 是通过学习 BASIC 进入编程领域的。

00:12:06:

如今，许多新手可能会改用 Python 或 JavaScript，而我想知道，Avi 你是否认为有最理想的入门语言。

00:12:18:

我想知道，是否有首选的入门语言？如果某人没有技术背景，没有编程经验，也没有计算机科学学历而进入这个领域，那么他们就是从一开始，甚至说从零开始……随着时间的推移，有没有更好的初始语言冒出来呢？

00:12:39 - Avi Flombaum:

好吧，首先，我要说的是每个人都沒有背景。沒有谁是天生的程序员，因此，无论你是有计算机科学学历，还是在社区中心里学习，抑或读书自学，每个人都是从初学者开始的。然后，就初学者的首选语言而言，我认为你最先看到的语言就是最好的。我对初学者的建议始终是，选择一种语言并学习它，不要改变。我认为，初学者最容易犯错的是，我先学习 Python，然后对它感到沮丧，说 Python 很烂，现在我要去用 JavaScript 了。他们学习 JavaScript，然后对 JavaScript 感到沮丧，然后又换了起来。

00:13:22:

如你所知，我个人认为 Ruby 是一种很棒的初学者语言。我认为它的语法真的很漂亮。它的价值，明确地就是要使你（程序员）感到高兴。我不知道是否有其他任何使程序员开心的语言。我知道人们发明了许多语言来使机器开心，但我认为 Ruby 确实很自然。

00:13:44 - Saron Yitbarek:

这一点我们可以达成共识。但是同样，这里没有错误的答案。毕竟，入门语言就是这样。它只是开始。我们所有人都有一生的时间用来学习，而且，选择一种语言而不是另一种语言，并不会阻止你成为出色的程序员。

00:14:05:

我喜欢画画，但我不会为了一支笔而死去活来。我不会因为我使用的笔而大动干戈，我为我所创造的东西而激动。那么从你的经验来看，为什么会这样呢？在大家面前辩护，让大家相信你对一种编程语言的看法是正确的，这样的一种思潮是从何而来的呢？

00:14:23 - Avi Flombaum:

我也不知道。因为我很喜欢你关于笔与作品的比喻，我首先想到的是……在我的设想中，人们死死地抓住工具，恐怕是因为作品并不吸引人。

00:14:41 - Saron Yitbarek:

哈哈，那真好笑。

00:14:42 - Avi Flombaum:

如果你做的东西就是不厉害，不伟大，对这个世界也没有太大的价值，你还想捍卫你的手艺，你唯一能指出的就是，是啊，看我把那把锤子挥得多好。当然，房子虽然倒了，但那把锤子，那把锤子真的很棒，我知道怎么用。作为一个人……我也觉得建造出的东西比你建造它的方式更重要。

00:15:09 - Saron Yitbarek:

说得好。现在，初学者不止可以选择 BASIC 或者 FORTRAN，我们已经拥有一整套的入门语言清单，但总有一种危险，那就是你忘记了这些语言仍然是达到目的的一种手段。语言是工具，不是你要制作的东西。

00:15:27 - Avi Flombaum:

我认为技术的价值在于，为人们创造有意义的事物。说到底，如果你确实是一名非常非常好的程序员，但是你做的东西没有人需要，没有人喜欢……它没有为世界增加价值，但是你真的非常擅长这门手艺，只是找不到能产生影响的方法，我想你可能会深入研究并讨论所使用的工具。

00:15:56 - Saron Yitbarek:

好吧，非常感谢 Avi，感谢你分享你的所有经验以及对编程语言的想法。那我们就谈到这里？

00:16:01 - Avi Flombaum:

是的，这非常好玩，很高兴再次跟你交流，我希望大家都找到自己喜欢的语言，每天都能用它来工作。

00:16:10 - Saron Yitbarek:

听起来不错。

00:16:13:

Avi Flombaum 是 Flatiron 学校的联合创始人之一。

00:16:20 - Saron Yitbarek:

编程学校可以将教育和知识获取的精神提升到一个全新的高度。这是我们在 Dartmouth 达特茅斯看到的一切的延续；但如今，在一个多样化的开发生态系统中，新的、更加自然的入门方式将会不断涌现，初学者们已经有了更理想的手段来开始他们的编程“游戏”；有时，他们甚至会以玩游戏的方式入门。

00:16:45 - CO.LAB 参与者:

你试过其他键吗？试试 G 键。

00:16:52 - Saron Yitbarek:

现在，在这暂停。这不是一群年轻的程序员在默默地背诵 O'Reilly 的课本，也不是 Tate Modern 在 Flatiron 学校的讲座。这其实是红帽在伦敦泰特现代美术馆里运行的一个小实验室。那些孩子呢？他们在学习写代码。对于新一代来说，编程的乐趣就是他们首先体会到的。

00:17:13 - Femi Owolade-Coombes:

好的。嗨，我叫 Femi，也叫 Hackerfemo。

00:17:17 - Saron Yitbarek:

Femi Owolade-Coombes 只有 13 岁，但他已经是新一代程序员中的一名领袖。

00:17:26 - Femi Owolade-Coombes:

我八岁的时候第一次接触到编程，那时我去约克参加一项活动。活动本身其实是数学主题的，不过我在那里看到了个很酷的东西——合法地黑入 Minecraft。作为八岁的孩子，那时候的我认为那真的很酷。所以我就这样喜欢上了编程。

00:17:47 - Saron Yitbarek:

他并不孤单。Minecraft 已经将一代人引入了编程领域，而且它做到了没有过去几代人所经历的痛苦和枯燥的学习。游戏的魔力正在消除障碍。Minecraft 是基于 Java™ 的，它也为该语言注入了新的活力，催生了一大群新的 Java 粉丝。

00:18:11:

但不一定是 Java。对于 Femi 来说，Python 是他在 Minecraft 中发现的语言。

00:18:16 - Femi Owolade-Coombes:

Raspberry Pi

当你使用树莓派版的 Minecraft 时，你可以用 Python，因为树莓派上的 Minecraft 是特殊编写的版本。它很酷，只要导入这个库，你就可以黑进去，到处放置爆炸性的 TNT；你可以在自己身后创建方块，也可以创造建筑物。你可以拿它做各种各样的事。

00:18:42:

当我第一次玩它时，我发现可以添加很多类似 mod 的东西（LCTT 译注：一类加载于电子游戏上的插件，玩家可以通过 mod 改变游戏的行为方式），这很酷。mod 这东西本身就有点像黑进游戏一样，但 mod 的存在让我意识到，我们也可以用正规的方法改变游戏，让它以你希望的方式行事。我认为这真的很酷。

00:18:57 - Saron Yitbarek:

Femi 打开了一个编程世界，而通往世界的大门是他最喜欢的游戏。然后，他做了一件了不起的事情。他开始向其他孩子分享那扇门。

00:19:10 - Femi Owolade-Coombes:

嗯，我想与同伴分享我的知识。因为我觉得，你知道吗？他们会非常喜欢的，我也会因此度过愉快的时光。我想与其他所有人共享这些，这样他们就能了解它，甚至可以参与编程。

00:19:30 - Saron Yitbarek:

South London Raspberry Jam

Femi 将此事贯彻到底，创办了南伦敦树莓酱（LCTT 译注：这个名称典出“树莓派”；jam 也有即兴演奏的意思，常用于一些赛事。），在那里，他已经看到了全新一代的编程者，他们正挑战人们以往对编程初体验的预判。除了那些 Minecraft 黑客，像 Scratch 或 Fruit 这样的可视化语言也让越来越年轻的用户拥有基础知识可以实现编程。

00:19:54 - Femi Owolade-Coombes:

我确实喜欢玩游戏这个点子，比起在课堂上学习代码，我最喜欢的是能够控制游戏中发生的事情，而代码是背后的魔法，代码给了你那种非常酷的能力，让游戏做你想要的事情。

00:20:15 - Saron Yitbarek:

Femi 的妈妈告诉我们，当她发现玩游戏并非一种弊大于利的追求时，她有多么高兴。我喜欢他的故事，因为这故事是从游戏开始的，却没有结束于游戏。他建立了一个属于年轻程序员的、了不起的社区；Femi 自己的代码生涯也在起飞，不再是围绕着 Minecraft 了。他在 HTML、JavaScript 和 CSS 领域工作，建立网站，做网页设计。他甚至在用 Unity 打造自己的游戏。

00:20:44 - Femi Owolade-Coombes:

每个人都应该有权利学习编程，因为这是未来。

00:20:53 - Saron Yitbarek:

Minecraft 真的是一所巨型编程大学吗？明日的程序员，是否会通过游戏和玩耍来吸收新的语言呢？只通过环境渗透能否真正地学习语言？

00:21:06 - Robyn Bergeron:

嗨，我叫 Robyn Bergeron。

00:21:08 - Saron Yitbarek:

community architect

Robyn 是 Red Hat 的 Ansible® 社区架构师，她有几个孩子，这些孩子们偶然间接触到了编程。

00:21:18 - Robyn Bergeron:

那是晚饭时刻。我在做饭，每个人对这件事情都印象深刻。我的女儿来到厨房，她说：“妈妈，我在 Minecraft 中提交了一个错误！”我从事软件工作，我看到过很多很多错误报告，而且我真的很好奇，在那个宇宙中，这意味着什么。是说我在 Twitter 上和别人聊了聊，说它坏了，还是什么？我让她给我看看，于是她打开了电脑，她已经在 Mojang 的系统中创建了一个 JIRA 帐户。

00:21:53 - Saron Yitbarek:

我们应该注意到，罗宾的女儿在当时只有 11 岁。

00:21:57 - Robyn Bergeron:

她把表格填得像模像样。我见过很多内容不完整的错误报告，也见过很多口吻过于尖锐的错误报告。但这是一个完美的报告……这份报告里有说“发生了什么事”，“我预期会发生什么事”，以及“如何重现错误”。对于很多人来说，这是他们与项目的第一互动，无论是商业的、专有的软件项目，比如一个电子游戏，还是一个开源软件项目。我很自豪，我告诉她，之后我们完全可以去参加 Minecraft 大会，因为他们一直很想去。

00:22:33 - Saron Yitbarek:

Robyn 意识到，当我们其他人从事日常工作时，孩子们却开始了一场革命。以下是在那场 Minecraft 大会上发生的事情。

00:22:43 - Robyn Bergeron:

我们去参加主题演讲，我说，就算我们在最后一刻去也会没事的，会在第二排找到位置。但我的设想完全错了，我们坐得就跟房间后面那 10 个大屏幕一样靠后。不过这并没有减少孩子们的热情。大会的其中一天，全体开发人员也在会议上出场了。当工程师们出来的时候，所有在场的孩子都站起来尖叫。如果你看过披头士乐队表演的视频，那这时候就像披头士乐队来美国时一样。我不能相信他们就在我们眼前，这是难以置信的一幕！在会议期间，人们都在试图得到他们的签名，这是……我和我的孩子坐在那里，我在想，我开发操作系统，连接互联网，这样你们才能在一起玩游戏吧？我们做错了什么，才会如此默默无闻呢？

00:23:36 - Robyn Bergeron:

但是孩子们就像，我长大后要成为使用 JavaScript 的人！是的，那个活动带来的热烈氛围令人着迷，但……这是一个电子游戏。

00:23:51 - Saron Yitbarek:

在 70 年代的某一段时间，每个人的入门语言都是 **BASIC**；然后可能是 **C**。近来，人们开始使用 **Java** 或 **Python**，但是可视化语言编程和游戏正在催生我们鲜有设想的编程未来。

00:24:10 - Robyn Bergeron:

尽管对于已经从事多年编程工作的人来说，这似乎微不足道，但我开始游玩的那一刻，我甚至没有意识到，我其实是在学习一种可以让我受益终生的东西。

00:24:23 - Saron Yitbarek:

community architect

Robyn Bergeron 是红帽的 Ansible 社区架构师。

00:24:32:

BASIC 邀请大学生进入编程世界，而 **Minecraft** 等游戏则邀请小学生进入当今编程世界。但是从某种意义上说，创造的动力并没有改变。那些大学生在学习 **BASIC**？是的，他们经常用它来编写游戏。最常见的似乎是足球题材。

00:24:54:

创新精神是驱使我们学习编程语言的第一推动力。这种驱动力让我们使世界变得更好，或变得更加有趣。

00:25:08:

下次，在第 3 集中，我们会讲述，全新的编程语言究竟从何而来？我们会了解到，巨大挑战是如何推动着开发人员从过去的语言中走出来，并在今天创造出新事物的。

00:25:26:

《代码英雄》是 Red Hat 的原创播客。如果你想更深入地了解 **BASIC** 的起源或在本集中听到的其他内容，请转至 redhat.com/commandlineheroes。

00:25:37:

我是 Saron Yitbarek。在下期节目到来之前，也请继续编程。

什么是 LCTT SIG 和 LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 LCRH SIG 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-3/learning-the-basics>

作者:Red Hat 选题:bestony 译者:erlinux 校对:acyanbird, Northurland, wxy

本文由 LCRH 原创编译, Linux中国 荣誉推出

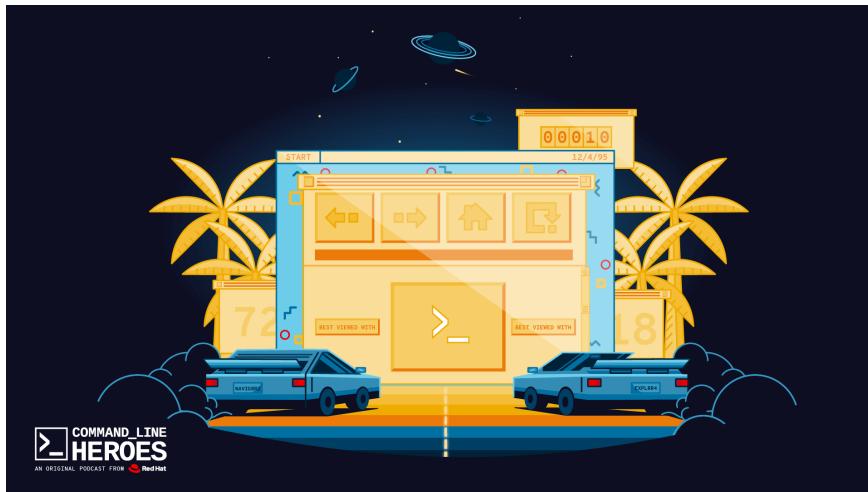
《代码英雄》第三季（3）：创造 JavaScript

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频播客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《代码英雄》系列播客第三季（3）：创造 JavaScript 的音频脚本。

导语：一个在 WWW 初期就确立了它的发展方向的任务，在 10 天内完成，结果呢？它成了一种改变了一切的、不可或缺的语言。

JavaScript 是战胜了一切困难的弱者。Clive Thompson 回顾了浏览器大战，以及这场战争对互联网未来的影响。Charles Severance 解释了 JavaScript 是如何从一个几乎不太可能实现的任务变成默认的 Web 开发语言的。Michael Clayton 坦言，他和其他许多人一样，低估了 JavaScript。而 Clint Finley 则描述了一个没有它的阴暗的互联网。

00:00:00 - Saron Yitbarek:

嗨，大家好。我们回来了。我们很高兴能推出《代码英雄》第三季。我们要感谢你们中很多人在这个节目中讲述的故事，因为每一季都源于我们与开发人员、SIS 管理员、IT 架构师、工程师以及开源社区的人们讨论你最感兴趣的的主题和技术。现在，我们正在进一步开放这种方式。我们希望大家都能参与进来，帮助塑造《代码英雄》的未来。你可以通过我们的简短调查来做到这一点。你喜欢这个节目的什么地方？你还希望我们多谈论哪些内容？亲爱的听众，我们想进一步了解你。你是开

发人员吗？你是在运维部门工作，还是在做一些与技术完全无关的工作？请访问 commandlineheroes.com/survey，以帮助我们提升第四季及以后的播客内容。现在，让我们进入第三季。

00:01:00:

Netscape

Brendan Eich 34 岁时，在网景公司总部任职。他正致力于一场为期 10 天的大规模编码冲刺。一种新的语言，一种全新的编程语言，将在短短 10 天内诞生。那是在 1995 年，编程语言的世界即将永远改变。

00:01:26:

我是 Saron Yitbarek，这里是《代码英雄》，一个来自红帽的原创播客。整季节目中，我们都在探索编程语言的威力和前景，探索我们的语言是如何塑造开发世界的，以及它们是如何推动我们的工作的。这一次，我们将追踪 JavaScript 的创建历程。也许你以前听过 Brendan Eich 的故事，但是像 JavaScript 这种计算机语言是如何真正创造出来的呢？其中肯定也是来自 Brendan 的努力，但是这个故事还有更多的内容。

00:02:02:

我们的 JavaScript 故事始于一场战争，一场浏览器之战。20 世纪 90 年代的浏览器大战似乎已经成为历史，但它的影响无疑是巨大的。战场的一方，是与 Sun Microsystems 升阳微系统结成了联盟的网景公司；另一方，你看到的是软件巨头，微软。他们争夺的战利品是什么？赌注已经大得不能再大了，因为这是一场决定谁将成为互联网看门人的对决。

00:02:40:

为了真正了解浏览器之战是如何进行的，我找来了我最喜欢的科技史学家之一，作家 Clive Thompson。他的最新著作是——

00:02:50 - Clive Thompson:

Coders: The Making of a New Tribe and The Remaking of The World
《编 码 者 : 新 部 落 的 形 成 和 世 界 的 重 塑》。

00:02:54 - Saron Yitbarek:

Clive 和我谈论的是浏览器之战，让我来为你做个背景铺垫吧。你会看到网景公司意识到浏览器将会是人们用来上网的关键软件。还有微软，他们的整个商业模式就是将东西打包到 Windows 中。直到 20 世纪 90 年代，他们才真正对浏览器感兴趣，微软意识到也许他们一直在打瞌睡。世界正在向网上转移，而微软的 Windows 内没有任何东西可以帮助他们实现这一目标。但是有些人已经在这么做了，一家名为网景的公司，他们正在提供一个通往互联网的入口。突然之间，微软在整个行业的主导地位看起来并不是那么绝对了。浏览器之战始于那一刻，微软意识到了互联网的力量，并开始打量他们新竞争对手。好了，这就是我的铺垫。这里我和 Clive 讨论接下来发生的事情。

00:04:03 - Clive Thompson:

这场战争是抢夺谁将成为上网的主要入口。你需要意识到，在 20 世纪 90 年代初期，没有人真正的在线。当 Mosaic 浏览器出现并最终变成网景浏览器时，它们是第一款任何人都可以下载的并让人能够浏览 Web 的浏览器。它们于 1994 年 12 月

上线。所以突然之间，成千上万的人能够以这种图形方式使用互联网。他们获得了巨量的下载和大量的新闻报道。基本上每个人都在说：“是的，网景是这种被称为互联网的事物的未来。”

00:04:40:

所以在西雅图，你可以看到微软非常警惕地关注着这件事，因为他们几乎忽略了互联网。他们只专注于销售 Windows，而实际上并没有对这种被称为互联网的疯狂新事物给予任何关注。因此，他们不得不加入到一场急速追赶游戏当中。近一年后，他们才推出自己的浏览器。在 1995 年秋天，他们的浏览器问世了，这实质上是浏览器大战的开始，微软也正在努力成为人们上网的门户。

00:05:13 - Saron Yitbarek:

好吧，花费一年的时间才让浏览器面世听起来不算太糟，对吧？时间不算太长，对吧？这似乎是一个合理的时间。

00:05:21 - Clive Thompson:

是的，没错。这听起来好像不是很长时间，但那时是发展的是如此之快。而且人们有一种强烈的先发优势意识，那就是第一家能够以自己的品牌形象作为你上网的方式的公司将是多年甚至是永远的赢家。我还记得当时的开发速度有多快。我的意思是，网景公司每两三个月就会推出一款新的浏览器版本。他们会说，“哇。现在，我们已经将电子邮件集成到浏览器中了。现在，我们在顶部有了一个小小的搜索栏。”它一直在变得越来越好。你可以在某种程度上看到，可以在网上做的所有事情都进入了视线，因为他们可以快速迭代并快速将其推出。

00:06:01:

微软习惯于非常缓慢的开发模式。可以是长达四年的开发过程，它是我们可以买到的没有 bug 的版本，把它封盒，投放到商店去，然后四年都不发布新版本。现在网景出现了，它是第一家说，“不，我们将推出一款不怎么合格的产品，但它运行得足够好，我们将在三个月、三个月又三个月内推出一个新的版本供你下载。”这完全破坏了微软的稳定的步骤。

00:06:30 - Saron Yitbarek:

好吧。如果我是微软，我可以看着它说，“哦，天哪。这就是未来。我需要迎头赶上。我需要与之竞争。”或者我可以说，“啊，这只是一时流行而已。”那么浏览器到底是什么呢？它让微软选择了第一个选项。它让微软说，“哦，天哪。这是个值得的东西。我要与之竞争。”

00:06:51 - Clive Thompson:

浏览器本身具有大量的文化传播和积淀作用。你在互联网上可以做的第一件事，一般是获得像文化之类的乐趣。你可以突然进入某个乐队的网页，查看他们的帖子和他们的照片。你可以培养你的爱好，找到佛罗里达州所有的人偶模型。在此之前，关于互联网的一切都看起来很呆板。电子邮件、文件传输、诸如此类。我的意思是，突然之间，浏览器使互联网看起来像一本杂志，像一个有趣的互动对象。报纸、CNN 和杂志前所未有的以这种非常激动人心的方式对此进行了报道。就在这一刻，科技相关的新闻被从《纽约时报》上排在后面的商业版块移动到了报纸的头版。

00:07:41 - Saron Yitbarek:

那么，对于开发人员而言，网景浏览器甚至说一般的浏览器能有什么吸引力呢？他们为什么如此着迷呢？

00:07:48 - Clive Thompson:

为此我拜访过很多开发人员。突然间，随着浏览器的出现，互联网出现了，有一个 Web 页面，上面写着：“下载我那酷酷的软件吧。”因此，它开启了我们今天看到的软件打造的整个世界。

00:08:04 - Saron Yitbarek:

我在这里应该提一下，起初微软实际上提出要收购网景公司。他们出价很低，网景公司拒绝了他们。因此，微软不得不打造自己的浏览器，他们称自己的浏览器为 Explorer 探索者（IE）。

00:08:21 - Clive Thompson:

微软花了一年的时间疯狂地开发浏览器，并于 1995 年秋天将其推出。他们做的事情与网景差不多。他们快速推出了一些东西，并不担心它是否完美，因为它会越来越好。但是，在 20 世纪 90 年代后半叶真正出现的一场战争是谁的浏览器最有趣、最具交互性、最功能复杂。

00:08:53 - Saron Yitbarek:

请记住，网景在这方面绝不是占了上风。

00:08:57 - Clive Thompson:

微软拥有非常强大的地位。当全球的 80% ~ 90% 的计算机都安装了 Windows 时，很容易就可以把软件设置为默认软件。而这正是他们所做的。所以你可以看到 Internet Explorer（IE）的不断崛起。

00:09:16 - Saron Yitbarek:

在某种程度上，可怜的老网景在这场战斗中一直处于劣势。但问题是，在战斗结束 Hail Mary 之前，他们做了一个孤注一掷的选择，事实证明，这将成为整个编程世界的一个令人难以置信的成绩。

00:09:35 - Clive Thompson:

这就是 JavaScript 创建过程中迷人而怪异的故事。

00:09:43 - Saron Yitbarek:

所有围绕 Web 的热议，围绕浏览器生活的热议，都非常清楚地表明了一件事。我们需要一种新的编程语言，一种远远超出 HTML 的语言。我们需要一种为所有新的基于 Web 的开发量身定做的语言。我们想要一种不仅能在网上生存，而且在那里蓬勃发展的语言。

00:10:10 - Clive Thompson:

如何为浏览器创建编程语言呢？

00:10:15 - Saron Yitbarek:

我的朋友，这是一个价值数十亿美元的问题。在网景看到微软与他们竞争的时候，他们开始关注 Java™。Java 会成为 Web 开发的语言吗？Java 是一种丰富的编译语言。它表现得和 C++ 一样好。但它仍然需要编译。开发人员确实想要一些更轻量级的东西，一些可以解释执行而不是编译的东西，一些可以吸引所有涌入 Web 的非专业程序员的东西。毕竟，那些新的程序员想要直接在网页上工作。那是我们的梦想。

00:11:05 - Saron Yitbarek:

网景公司需要一种可以在浏览器内部运行的编程语言，让开发人员能够让这些静态网页动起来。他们想，如果他们能在发布 Netscape 2.0 测试版的同时，发布一种新的轻量级语言，为 Web 编程创造奇迹，那不是很棒吗？只是有一个问题，他们只有 10 天的时间来创造一门新的语言。实际上，只给了一个叫 Brendan Eich 的人 10 天的时间。他就是那个负责完成这件事的人。毫无疑问，如果有人能做到这一点，那就是他。在 Brendan 还是伊利诺伊大学的学生时，他常常为了好玩而创造新的语言，只是为了摆弄一下语法。

00:11:57 - Charles Severance:

Brendan Eich 的关键在于，在构建 JavaScript 时，Brendan Eich 已经是编程语言狂热分子了。

00:12:05 - Saron Yitbarek:

为了了解 Eich 到底取得了什么成果，我们联系了 University of Michigan School of Information 密歇根大学信息学院的教授 Charles Severance。

00:12:14 - Charles Severance:

JavaScript 在某种程度上是在 Java 被视为未来的环境中创建的，在 1994 年，我们认为它（Java）将解决一切问题。一年后，那个真正能解决一切的东西即将出现，但它不能说，“嘿，我已经解决了一切”，因为每个人，包括我自己，就像都相信 1994、1995 年的我们已经看到了未来一样，这个未来就是 Java 编程语言。他们必须建立一种看似无关紧要、看似愚蠢、看似毫无意义，但却是正确的解决方案的语言。

00:12:56 - Saron Yitbarek:

但是 Eich 提供的可不仅仅是一种玩具语言。它以隐藏的方式进行了复杂处理，并从以前的语言中汲取了主要灵感。

00:13:07 - Charles Severance:

如果你看一下基本语法，很明显它的灵感来自于带有花括号和分号的 C 语言。一些字符串模式取自 Java 编程语言。但面向对象的底层模式取自名为 Modula-2 的编程语言，它有头等函数的概念，对我来说，这确实是使 JavaScript 成为如此强大以及可扩展语言的最令人惊叹的选择之一，即函数、函数的主体、构成函数本身的代码也是数据。

00:13:41:

另一个真正的灵感来源于 HyperCard。JavaScript 总是在浏览器中运行，这意味着 Document Object Model 它有文档对象模型（DOM）的基本数据上下文，文档对象模型是网页的面向对象表示。它不像传统的编程语言。JavaScript 代码不是从一开始就有的，最

初它是一个网页，最终演变成了这种面向事件的编程。

00:14:12 - Saron Yitbarek:

1995 年 11 月 30 日，当 JavaScript 与网景的 Navigator 2.0 一起发布时，所有的 America Online 魔力都被植入到一粒强大的语言小种子中。包括美国在线 (AOL) 和 AT&T (美国电话电报公司) 在内的 28 家公司同意将其作为一种开放标准的语言使用。当它发布时，有一些老派的专业人士对 JavaScript 嗤之以鼻。他们认为这只不过是一种新手的语言。他们忽略了它革命性的潜力。

00:14:46 - Charles Severance:

这些超级先进的概念来自不太知名但又非常像高级面向对象的语言当中，Brendan 决定将所有这些概念融入其中。所以 JavaScript 就像一只特洛伊木马。它在某种程度上潜入了我们的集体意识，认为它很傻、像个玩笑、简单、轻巧。但是几乎从一开始它就建立了一个功能强大的、深思熟虑的编程语言，它几乎能做计算机科学中的任何事情。

00:15:17 - Saron Yitbarek:

其结果是成为了一种浏览器原生语言，可以随着我们在线生活的发展而不断进化。没过多久，JavaScript 就成为了事实上的 Web 开发选择。

00:15:29 - Charles Severance:

JavaScript 是一种不二之选的编程语言，我只能学习它，事实上学习 JavaScript 的人通常别无选择，因为他们会说，“我想构建一个浏览器应用程序，我想让它有交互元素。”答案是你必须学习 JavaScript。如果你想象一下，比如说，你最喜欢的编程语言是什么，那么这个问题的答案几乎就是某某加上 JavaScript，对吧？有人可能会说，“我喜欢 Python 和 JavaScript”，或者“我喜欢 Scala 和 JavaScript”，因为它就像是每个人都需要学习的语言。

00:16:05 - Saron Yitbarek:

University of Michigan School of Information
Charles Severance 是密歇根大学信息学院的教授。他说，网景公司一开始非常强大，他们在浏览器之战中奋力拼搏，但最终……

00:16:22 - Clive Thompson:

网景浏览器作为一款严肃的产品就这样消失了。

00:16:27 - Saron Yitbarek:

微软在整个行业的主导地位是一股压倒性的力量。尽管在浏览器竞争上晚了一年，但他们还是能够力挽狂澜，赢得了今天的胜利。但你知道，网景公司最后一击，它创造的 JavaScript，是成功的，在战斗结束很久之后，这种从浏览器战争中诞生的语言瑰宝，将有一个改变一切的后世。

00:17:01:

如果你是最近才开始编程的，很可能会理所当然地认为，你可以开发可更改和更新的交互式 Web 页面，而无需从服务器拉取页面的全新副本。但是，想像一下，当这样做成为一种全新的选择时会是什么样子的。我们有幸请到红帽公司的软件工程师 Michael Clayton 帮助我们了解那是一个多么巨大的转变。

00:17:28 - Michael Clayton:

我想说，在 2004 年 Google Mail 发布了。Gmail，据我所知，它是第一个真正将 JavaScript 带到更高水平的 Web 应用程序，它使用 JavaScript 来动态地切换你正在查看的内容。

00:17:49 - Saron Yitbarek:

假设你正在查看收件箱，然后单击了一封电子邮件。在过去，你的电子邮件查看器会在你的浏览器中加载一个全新的页面，仅仅是为了向你显示那封电子邮件。当你关闭该电子邮件时，它会重新加载整个收件箱。

00:18:05 - Michael Clayton:

这造成了很多延迟。当你在视图之间来回切换时要等待很多时间，Gmail 改变了这一切。他们使用 JavaScript 在后台获取你想要查看的内容，然后将其展现在你面前，而无需等待全新的页面视图。

00:18:23 - Saron Yitbarek:

这节省了大量的时间和精力。但是仔细想想，它改变的不仅仅是速度。它改变了我们工作的本质。

00:18:35 - Michael Clayton:

所以，Web 开发者作为一种职业，已经从类似幕后角色的服务端走到了离用户仅薄薄一层之隔的位置，因为他们直接在浏览器中编写代码，而用户也正是通过浏览器查看 Web 页面。

00:18:52 - Saron Yitbarek:

它改变了一切。事实上，你完全可以把引领 Web 2.0 革命的功劳都归功于 JavaScript。任何有 Web 浏览器的人都突然之间拥有了一个摆在他们面前的开发环境。但是，正如我之前提到的，老保守派对民主性并不一定感到舒服。

00:19:16 - Michael Clayton:

早期反对 JavaScript 的人当中，我也是其中的一员。我有个阻止 JavaScript 运行的浏览器扩展。我认为它是一种无用的玩具语言，每当我访问一个网页，该网页的某些关键功能需要 JavaScript 时，我都会感到愤怒。我想，“你应该在没有 JavaScript 的情况下以正确的方式构建你的网站。”

00:19:43 - Saron Yitbarek:

然而，很快，Brendan Eich 仅仅用 10 天创建的语言，它所蕴含的美和潜力对每个人来说都变得显而易见了。现在，它不仅征服了浏览器，也征服了服务器。有了 Node.js，这种小众语言的全新领域已经打开。

00:20:03 - Michael Clayton:

当我听说 JavaScript 打算在服务器上运行时，我想，“为什么会有人想这么做？”那时，我已经是一名专业的 JavaScript 开发人员了。我每天都写很多 JS，但我还是不太明白为什么它可以归属到服务器端，事实证明，像很多听众都知道的那样，Node.js 现在是这个行业的一支巨大的力量。我认为这是有充分理由的。

00:20:32:

Node.js 如此成功的原因之一，就是它拥有庞大的前端 **JavaScript** 开发人员和客户端开发人员社区。他们写代码，他们在用 **JavaScript** 为浏览器编写代码。这么多的开发者，现在又可以用同样的语言来为服务器端编程，这让他们立刻就拥有了大量的可以立即开始为服务器端做贡献的人员。这个工具已经在你的工具包中，你只需将其拿出来，安装上 **Node.js**，然后就可以加入到编码竞赛中去了。

00:21:11 - Saron Yitbarek:

先是在浏览器中，然后又在服务器上。**JavaScript** 是这种朴实无华、暗自芬芳，有时候也会有点古怪的编程语言。这个浏览器战争中的幸存者，被大家低估了。

00:21:25 - Michael Clayton:

JavaScript 算是编程语言中的灰姑娘故事，它始于基本上是在 10 天内拼凑起来的初态。中间经历了来自其他编程社区的许多嘲笑，然而仍以某种方式继续取得成功和增长。最后到现在稳居世界上最流行的编程语言中排名第一、第二的位置。

JavaScript 基本上无处不在。在网页内部运行的能力意味着 **JavaScript** 和 Web 一样普及、非常普遍。

00:22:08 - Saron Yitbarek:

Michael Clayton 是红帽公司的工程师。**JavaScript** 吞噬了世界吗？它是否搭上了 Web 的顺风车，才成了一种主流语言？我想找出 **JavaScript** 的实际边界在哪里。

00:22:25 - Klint Finley:

嗨，我叫 **Klint Finley**。我是 [Wired.com](#) 网站的撰稿人。

00:22:28 - Saron Yitbarek:

Klint 对同样的事情也很好奇。他越是关注今天 **JavaScript** 的运行方式，就越发意识到它已经渗透到他的在线生活的每一个环节。

00:22:40 - Klint Finley:

在你还没来得及决定是否要让所有这些不同的应用程序在你的电脑上运行之前，**JavaScript** 已经成为一种可以增强整个应用程序能力的工具。它们就那么运行了，它们参与了广告或促进广告商使用的跟踪。所以，在你的浏览器中，有很多事情在无形中发生，你甚至可能根本不知道，也不希望发生。

00:23:07 - Saron Yitbarek:

因此，**Klint** 决定做一个小实验。

00:23:10 - Klint Finley:

我决定试着在没有 **JavaScript** 的情况下使用 Web 一段时间。我决定试一试，花一周时间禁用浏览器中的 **JavaScript**。

00:23:21 - Saron Yitbarek:

听起来很简单，但是放弃所有 **JavaScript** 产生了一些令人惊讶的效果。因为 **JavaScript** 已经变得如此之大，如此之全，这种以轻量级著称的语言现在实际上占用了大量的空间和能源。当 **Klint** 屏蔽了那种语言时才发现……

00:23:39 - Klint Finley:

总体而言，这在很多方面都是一种更好的 Web 体验，比如页面加载更快，页面更干净，我电脑的电池续航时间更长，并且我对电脑上发生的事情有了更多的控制感，因为没有这些奇怪的、看不见的随机程序在后台运行。

00:24:02 - Saron Yitbarek:

想象一下第一次过上没有弹出式广告的生活是多么幸福。

00:24:07 - Clint Finley:

很多东西很大程度上依赖于 JavaScript 来加载。所以网页变得简单多了，广告少了，干扰也少了。

00:24:17 - Saron Yitbarek:

不过，这种整洁的 Web 体验并不是全部。如果你拔掉 JavaScript 的插头，Web 的某些部分就完全不能工作了。

00:24:26 - Clint Finley:

很多内容都不能正常运行了。Gmail 把我重定向到了一个为旧手机设计的不同版本。Facebook 也一样，很多流畅的互动没有了，它变得更像是一系列的网页。因此，Netflix 无法正常工作。YouTube 无法正常运行。是的，任何非常依赖互动的东西都不能运行了。拿掉了 JavaScript，有好处也有坏处，最终我不得不做出抉择，有 JavaScript 总比什么都没有要好。

00:25:05 - Saron Yitbarek:

Klnt Finley 是 Wired.com 的撰稿人。大多数人预测 JavaScript 只会继续主导移动和桌面应用程序开发。像基于浏览器的游戏、基于浏览器的艺术项目等等，它们的复杂程度正在飞涨。不断增长的 JavaScript 社区正在最大限度地利用这一潜力。

00:25:34:

值得回想一下，就在 1995 年，就在几十年前，Brendan Eich 坐在一个房间里，设计出一门新的语言。今天，这种语言渗透到我们所做的每一件事中。也许说一串新的代码会改变世界听起来有点陈词滥调，但它确实发生了。一位代码英雄将他对语言的所有热爱汇聚到 10 天的冲刺中，世界的 DNA 也将永远改变。

00:26:10:

我们可以为 Google Docs、YouTube 和 Netflix 而感谢 JavaScript。但是你知道，“能力越大，责任越大”，随着 JavaScript 的影响力在大量开源库的推动下不断增长，责任不再仅仅落在一个人身上了。一个更广泛的社区已经接过了责任。

SlashData 最近估计 JavaScript 开发人员的数量为 970 万，在 GitHub 上，
Pull Requests
JavaScript 有比任何其他语言都多的 PR（拉取请求）。JavaScript 在全世界代码英雄们的力量加持下，正在走向美好未来。

00:26:59:

下一期的《代码英雄》，我们将遇到另外一种 Web 语言，我们将探索 Perl 是如何在一个广阔的新领域蓬勃发展的。

00:28:04:

最后，有听众在网上分享了我们上一季的 **Hello World** 那一期，在该期中我们也谈到了 **Brendan Eich** 和 **JavaScript**。在那一期，有嘉宾说，在那 10 天里，**Brendan** 可能没有睡过多少觉，如果有的话，也是很少。好吧，**Brendan** 在推特上回应说，他确实在那次冲刺过程中睡过觉。想要更多地了解这 10 天发生了什么，请查看 Devchat 对 **Brendan** 的采访播客。我们会在我们的节目记录里加个链接。我是 **Saron Yitbarek**。下期之前，编码不止。

什么是 LCTT SIG 和 LCTT LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 **LCRH SIG** 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-3/creating-javascript>

作者: Red Hat 选题: bestony 译者: gxlct008 校对: windgeek, FineFan, wxy

本文由 **LCRH** 原创编译, **Linux中国** 荣誉推出

《代码英雄》第三季（4）：深入 Perl 语言的世界

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频播客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《[代码英雄](#)》系列播客《代码英雄》第三季（4）：深入 Perl 语言的世界的音频脚本。

导语：语言来了又走。只有少数几种语言具备了登峰造极的能力，而能留在那里的则更少。Perl 有过一次惊人的崛起，也有过一次平静的低迷，现在已经在编程世界中找到了自己的位置。

Perl似乎注定要统治网络。Michael Stevenson 和 Mike Bursell 描述了 Perl 的设计如何使它成为早期 web 的理想选择。我们从 Conor Myhrvold 那里听到了它的座右铭：“实现它的方法不止一种”。Elizabeth Mattijsen 分享了，尽管 Perl 有优势，但漫长的开发周期如何减缓了 Perl 的发展。虽然它不再是最顶级的网络语言，但 John Siracusa 指出，Perl 作为一种小众工具仍然存在。

00:00:02 - Saron Yitbarek:

想象一下 Perl 语言的创建者 Larry Wall 在 1999 年的一次会议上站在麦克风前，留着他标志性的浓密胡子和梳理过的刘海，感觉相当好，因为他所发明的语言正在越来越受欢迎。

00:00:19 - 配音演员：

声音测试。

00:00:19 - Saron Yitbarek:

Perl 语言轻而易举地超越了 COBOL、Visual Basic，而 Python 呢？Python 仍然是一个不入流的竞争者，是 Larry Wall 的几个玩笑素材。Wall 展示了 dice.com 报告中的人群数据，在那时来看，Perl 语言的未来是非常、非常光明的。然而之后 Perl 的未来就不再如此了。20 年之后，dice.com 在 2018 年夏天将 Perl 列为最可能灭绝的语言之一。短短 20 年之间发生了什么？

00:00:59 :

我是 Saron Yitbarek，这里是《代码英雄》，一档红帽公司的原创播客。这一季是关于我们编程语言的力量和前景的。在上一集我们追踪了搭上了互联网顺风车的 JavaScript 的疯狂崛起。

00:01:19:

并不是每一种语言都有一个不停地成长和成功的故事。大多数语言的出现，在当时非常特殊的生态系统中发挥了它们的作用，然后当我们的编程生活里需要其他类型的工具时它们就开始消退。

00:01:37:

本集我们将深入了解 Perl 世界。是什么造就了它早期的成功，为什么它的成功突然就被颠覆了？我们所处的环境，我们的基础设施，我们的硬件，各种因素都会决定哪些语言会繁荣，哪些会开始萎缩。这就是 Perl 语言的故事的魅力所在。

00:02:08:

我们知道 Perl 并没有统治世界，但是退回到上世纪 90 年代的时候，却看不到这一 World Wide Web 点。Tim Berners-Lee 在 1991 年发明了万维网（WWW），它迅速创造了一个全新的基于 Web 的开发领域。谁也说不准会是哪种编程语言在这个新领域取得成功。

00:02:31 - Michael Stevenson:

在 Web 出现的时候，所有人都等待着会有什么事情发生。那个时候整个世界都是令人兴奋的。

00:02:39 - Saron Yitbarek:

University of Amsterdam
Michael Stevenson 是阿姆斯特丹大学媒体研究的副教授。他为我们描述了 America Online 早期的 Web。人们见过 Usenet，也见过 ARPANET。他们看到了美国在线，还有一些其它的东西。但是直到 Web 出现，互联网的全部发展潜力才真正得到体现。突然之间，你就可以通过鼠标和点击进入了这个巨大的互联世界。这是一个出乎意料的事情。

00:03:09 - Michael Stevenson:

你要是记得 1993 年，也就是 Web 开始崭露头角的那一年，也是 Wired Magazine 《连线杂志》开始出版的那年。在那之前类似《Mondo 2000》这类的杂志真的让电脑看起来像神秘知识的来源，让电脑看起来很酷。

00:03:32:

因此从这个意义上说，Web 也到达了一个相当特定的时期，人们已经准备好以这样的方式迎接技术的兴起。

00:03:43 - Saron Yitbarek:

故事在这个时候开始了：Larry Wall 在 1987 年创建了 Perl，4 年后万维网才开始兴起。Larry Wall 给世界带来的 Perl 最初只是一种通用的 Unix 脚本语言。当然，它确实很有用，但同时 Perl 还有一些秘密元素，这些元素使它成为即将到来的 Web 开发世界的理想语言。

00:04:14 - Michael Stevenson:

比较有名的是 Perl 是 Larry Wall 在参与的一个美国国家安全局（NSA）的秘密项目中发明的，基本上他所做的就是创建一个类似黑客版的新闻栏目，运行在 Usenet 上，因此 Perl 语言从一开始就是操作文本和将数据从一个地方移动到另一个地方，这完全符合 Web 的需求。而 Perl 作为一种易于使用的脚本语言，更加接近于自然语言。它可以用来快速开发东西，所有这些都让 Perl 成为了一个完美的契机，不仅适合专业用户，也适合新加入的业余爱好者。

00:05:09 - Saron Yitbarek:

很偶然的是，当 Web 出现时，Perl 已经为 Web 做好了准备。Larry Wall 不可能知道 Web 即将出现。但当它出现时正好相互吻合。但我认为还有一点很关键：Perl Free Software Foundation 是一种自由语言。Larry Wall 在自由软件基金会开发的 General Public License GPL（通用公共许可证）下发布了它。

00:05:37:

Larry Wall 让他的语言自由开源的决定，这完全符合刚刚开始出现的基于 Web 的新思维方式，同时 Perl 在其他方面也很有前瞻性。

00:05:50 - Michael Stevenson:

Perl 的开放性令人难以置信，在某种意义上，它总是愿意整合一些其他的东西，新的东西。这和 Larry Wall 的身份很相配。他是个很开放谦虚的人，总是仔细考虑别人的想法，并试着设身处地为别人着想。对我来说 Perl 作为一种语言，以及作为一个社区的性格，在很长一段时间里，真的很符合这一点。

00:06:27 - Saron Yitbarek:

同样 Perl 非常适合 Web 早期的、狂野的西部阶段，也就是黑客时代。实际上 Perl There is more than one way to do it. 的座右铭之一就是“实现它的方法不止一种”。

00:06:39 - Michael Stevenson:

在 Perl 的鼎盛时期和这个试验性的开放 Web 的鼎盛时期之间，存在着一种近乎浪漫的联系，在它变得如此被几个平台所控制之前。

00:06:56 - Saron Yitbarek:

记得上世纪 90 年代是互联网历史上的一段启蒙时期，那时人们还在争先恐后地想知道还有哪些是可能的。那时对编程的需求是巨大的，每个人都需要有一个网页，这意味着一群全新的开发人员，他们都对新的做事方法持开放态度。问题变成了“好吧，我们有了一个全新的领域，但用什么语言来完成这项工作呢？”

00:07:26:

Perl 虽然并不是这个问题的最终答案，但对于很多人来说 Perl 是他们首选的答案。

00:07:34 - Michael Stevenson:

我并不是说我更愿意看到加载速度超慢的网页、也没有谷歌搜索引擎的时代，但我确实认为那个时代有一些特别的东西，当时有一些人，在他们的宿舍里，创建了一个类似 slashdot 的东西。而现在随着 Web 已经变得如此主流、专业化和被集中在几个大公司中，我们确实怀念那个时代。对我来说，Perl 比其他任何早期的故事都更能象征这一点。

00:08:15 - Saron Yitbarek:

Michael Stevenson 是阿姆斯特丹大学媒体研究的副教授。

00:08:24 - Saron Yitbarek:

之后，随着 90 年代的到来，Perl 作为一种能适应早期互联网发展潜力的语言出现了，它是一个时代的语言。Larry Wall 和他所创造的 Perl 正好明白了 Web 的本质。

00:08:40 - Mike Bursell:

在网上你可以随意搜索，也可以随时创建网页，这是互联网的美丽新世界，你可以随时做这些事情。

00:08:52 - Saron Yitbarek:

这位是 Mike Bursell，红帽公司的首席安全架构师。Mike 是 90 年代中期发现和使用 Perl 的黑客之一。

00:09:00 - Mike Bursell:

对于 Web 来说，Perl 是许多人的起点。Java™ 语言还处于早期阶段，它在文本输入输出方面并不是很出色。如果你想进行查询和生成页面，Perl 是人们都在使用的工具。

00:09:22:

Perl 非常适合，因为它擅长获取文本，并使用文本做其他事情，而这正是 Web 所需要的。

00:09:31 - Saron Yitbarek:

顺便需要说的是，Larry Wall 有语言学背景，这就解释了为什么 Perl 有强大的文本解析能力。正如 Mike Bursell 提到的，这是一笔巨大的好处，因为在早期，Web 主要是一种基于文本的媒介，因为那时候人们没有足够的带宽来处理图像。

00:09:51 - Mike Bursell:

它很容易使用，也很容易复制。人们在分享方面非常开放，而且它的输出速度很快，这些都是好东西。

00:10:02:

哦，当然还有一点，就是你可以用管道使用它。所以，这是非常多的人所习惯的，而且非常容易测试，甚至离线测试，这都是非常有用的。

00:10:13 - Saron Yitbarek:

尤其对那些在 Web 的世界中重新规划自己生活的系统管理员来说非常有用。

00:10:21 - Mike Bursell:

Perl 是给系统管理员们的真正礼物。即使在那个年代，如果你做一些事情，你会得到很多日志。管理这些日志，分解它们，搜索它们，并能够以不同的方式显示它们，或获取任何其他大型文本语料库（主要就是日志），甚至可以对它们进行调试。除非你要在命令行里用管道方式传输 `orc`、`sed` 之类的东西，以及 `ed` 命令，那样的话你很快就会变得非常痛苦，而 Perl 正好适合让你去处理这些事情。

00:10:55 - Saron Yitbarek:

到上世纪 90 年代后期，Perl 5 已经聚集了一个强大的用户群体。像 Fortran 和 C 这样的旧语言依赖于庞大而昂贵的硬件，而 Perl 更有生命力，也更容易移植。在这样一个硬件成本急剧下降的世界里，Perl 的设计使得它得到了蓬勃发展，Perl 让所有的新程序员快速、轻松地工作。这是因为 Larry Wall 以牺牲 CPU 和内存的需求为代价，使 Perl 语法更人性化。所有这些元素组合在一起，使 Perl 成为一种很受新开发社区欢迎的语言。

00:11:36 - Mike Bursell:

在一个正在壮大的社区里，你可以去和他们聊聊社区里的事情，然后 PerlMonks 论坛出现了，那里是一个讨论的好地方，能在论坛里知道正在发生的事情。

00:11:50 - Saron Yitbarek:

这个社区确实拥有 Web 所能提供的最好的东西。他们发现了一个巨大的软件模块库，一个叫做 CPAN 的存储库，这些系统管理员都很喜欢它。它给 Perl 提供了更多的灵活性，许多人都可以部署由少数几个编程大师编写的代码。

00:12:15 - Mike Bursell:

它有很多库可以做你想做的任何事情，如果你找不到你想要的库，你可以去问一下，然后就会有好心人写出你想要的库。

00:12:21 - Saron Yitbarek:

Mike Bursell 是红帽公司的首席安全架构师。

00:12:28 - Saron Yitbarek:

正是由于 Perl 是免费的，它受到不断增长的模块库的支持，它是可移植的，而且它有一个蓬勃发展的社区。一切看起来都很好。Perl 可以在上世纪 90 年代 Web 开发新天地中发挥所有优势，但就在 90 年代即将结束的时候，互联网的发展前景又一次发生了变化，时代来了一个大的转变。

00:12:57 - Alan Greenspan:

但我们又能如何知道非理性繁荣何时已过度推高了资产价值，进而导致了意料之外的长期经济收缩？

00:13:12 - Saron Yitbarek:

“非理性繁荣”是时任美联储主席 Alan Greenspan 在 1996 年对

American Enterprise Institute

美国企业协会所说的话。他那句“非理性繁荣”是对 90 年代人人都经历过的网络泡沫的警告。所有早期使用 Perl 的 Web 开发人员都在那个泡沫中乘风破浪，但正如 Greenspan 预测的那样，泡沫在 2000 年破裂了。

00:14:11 - Conor Myrvold:

大家好，我是 Conor Myrvold。在过去的五六年里我一直从事编程，现在在技术领域为 Uber 工作。

00:14:20 - Saron Yitbarek:

2000 年代初，当 Conor 还在高中的时候，Perl 仍然是一个非常重要的东西。但他越来越意识到一种与之竞争的语言，叫做 Python。

00:14:31 - Conor Myrvold:

Python 所追求的是一种更结构化的语言，做很多不同的事情都有一种更显然的方式，它就是那样设计的。而 Perl 则喜欢做某件事有不止一种方法，但这让很多初学者感到困惑。

00:14:49 - Saron Yitbarek:

There is more than one way to do it.

Perl 有这样一句座右铭“实现它的方法不止一种”。而 Python 的理念实际上是相反的，Python 为每个问题都提供了一个明显的解决方案，这意味着查看别人的 Python 代码很容易；而另一方面查看其他人的 Perl 代码可能会令人困惑。Perl 作为一个程序员的第三或第四种语言是有意义的，而正是因为它是一种脚本语言，而脚本是互联网连结的基础。

00:15:23:

但是 Python 是一种你可以真正深入研究的语言，即使你是一个新手。Perl 有一套特定的优势，比如在搜索文本和生物信息学上。但是 Python 就是这样一种简单通用的语言。Python 获得了越来越多的信任，成为人们首先想学的语言，这是一件大事。

00:15:47 - Conor Myrvold:

越来越多的人开始上网，越来越多的人开始学习如何编程。尤其是相对于 Perl 而言，Python 受益于它本身相对容易学习，因为它更结构化。而这样结果是，在一个快速增长的世界里，如果你能得到更多增长的份额，那将最终意味着有更多的教程和更多的东西可供你使用。

00:16:10 - Saron Yitbarek:

在之前提到过的 CPAN，它是 Perl 用户可以使用的强大的中央存储库。这在 90 年代是一大亮点，但 CPAN 的价值也在变化。

00:16:24 - Conor Myrvold:

这也不能真正帮助你学习一门语言，因为你是在“复制粘贴”，只是用最少的方式替换你需要的东西。从长远来看这是一个劣势，因为如果你让人们通过自己进行原始开发来学习如何使用一种编程语言，即使这需要花费更长的时间，他们也会觉得自己对它投入了更多，而且他们也了解在这中间发生了什么。

00:16:48 - Saron Yitbarek:

Python 没有像 CPAN 那样的集中式存储库，但是对于那些在新千年时代来到这里的开发人员来说，在一个互联网搜索功能如此强大的世界里，存储库并没有那么大的价值。

00:17:05 - Saron Yitbarek:

最终 Python 有了大量的教程，当然现在也有了像 GitHub 这样的平台。

00:17:13 - Conor Myrvold:

最终发生的事情是 Perl 拥有的许多优势，是来自一个已经过时的时代的网络效应。

00:17:24 - Saron Yitbarek:

Conor Myrvold 是 Uber 的一名工程师。

00:17:30 - Saron Yitbarek:

然而语言的兴衰很少是由外部力量单独决定的，而 Perl 的内部问题是，在它的发展过程中它似乎遇到了障碍。Python 正在以一种相当有序的方式发布新的迭代，而正如我们在本季度第一集中所了解到的，Perl 在 2000 年互联网泡沫破裂之时，Python 开始获得更多新开发人员的青睐。

00:17:59 - Saron Yitbarek:

每个人都期待着 Perl 6 的发布，人们都很兴奋。他们等啊，等啊，等啊……他们等了 14 年。

00:18:15 - Elizabeth Mattijsen:

人们提出了大约 300 多件 Perl 6 应该能够完成的事情，当然其中很多事情基本上都是相互排斥的。

00:18:26 - Saron Yitbarek:

这是 Elizabeth Mattijsen，她是 Perl 6 的核心开发人员。2000 年，Elizabeth 参加 Monterey 了在蒙特雷举办的 Perl 会议。那时开发者认为他们已经停滞不前了，所以 Perl 6 是必要的。Larry Wall 同意了，但是如果说 Perl 5 是他对 Perl 的重写，那么他希望 Perl 6 是由社区来对 Perl 进行重写。由于团队合作可能需要更长时间，甚至用了 14 年，对于那些开发者来说，这是一条漫长而艰难的道路。

00:19:01 - Elizabeth Mattijsen:

我们可以说当前实施的 Perl 6 项目实际上是实现它的第三次尝试。

00:19:07 - Saron Yitbarek:

按照 Elizabeth 的说法，在这 14 年里有过多次尝试。中间经历了漫长而痛苦的深度的尝试。开发者们心力交瘁；人们陷入了死胡同。到 2015 年圣诞节那天 Perl 6 终于问世时，世界上的许多地方已经开始了新的发展。而需要注意的是 Perl 6 并没有给予成为某种革命性的新事物，从而实现对 Python 的反击。Perl 6 是对原版进行了深思熟虑的重新设计。

00:19:43 - Elizabeth Mattijsen:

State of the Onion

我认为 Larry Wall 在他的“洋 葱 状 态”演讲中使用了一个很好的比喻。对他来说，Perl 5 就像《霍比特人》，而 Perl 6 就像《指 环 王》。如果你仔细看过《霍比特人》和《指环王》的故事，你会发现它们基本上是同一个故事。只是《霍比特人》比《指环王》小得多，情节漏洞也更多，没有《指环王》那么宏大的背景。我认为这很好地描述了 Perl 5 和 Perl 6 之间的区别。它基本上是同样的想法，同样的思路，同样的环境，只是对它的重新构想。

00:20:26 - Saron Yitbarek:

Elizabeth Mattijsen 是 Perl 6 的核心贡献者。

00:20:32:

如今，Perl 甚至可能不在前 20 种语言之列。在外部竞争和内部拖延之间，它还没有向大多数新开发人员证明自己。但这提出了一个大问题，我们真的应该根据一种语言的流行度来判断我们的编程语言吗？或者说我们应该根据其他方面来判断一种编程语言的价值？当昔日的超级巨星成为陪衬时，这到底意味着什么呢？

00:21:06:

在世纪之交时互联网泡沫破裂时，Perl 的统治地位开始衰退时，Larry Wall 发表了一个有趣的声明。他认为尽管 Perl 永远不会再成为世界上最流行的编程语言，但它可以成为较小类别中的领先者。Larry Wall 说那才是真正的目标。成为同类中最好的，而不是世界上最好的。正如他所说的，SUV 永远不会和赛车竞争。

00:21:38 - Saron Yitbarek:

我想深入研究这个想法，我想了解在细分类别中做到最好对编程语言的真正含义。

00:21:48 - John Siracusa:

我是 John Siracusa，我是一个程序员，也是一个播客主。

00:21:53 - Saron Yitbarek:

John 实际上共同主持了三个播客：《Accidental Tech Podcast》、《Reconcilable Differences》和《Robot or Not?》。我们刚刚聊到了 Perl 在当今世界的地位。

00:22:06:

Perl 在当今世界排名如何？它仍然是最好的计算机语言吗？

00:22:10 - John Siracusa:

Perl 6 具有其他语言没有的、其他语言应该具有的东西，我一直在等待其他语言偷学它。例如语法是将常见任务概念化的一种好方法，而在我看来，使用语法来解决解析问题比使用现有的工具更令人愉快、更干净、更好。

00:22:31:

在 Perl 中，对象系统的许多部分看起来很琐碎而无关紧要，但我完全期待其他语言最终会采用它，就像许多语言最终采纳了 Perl 5 中的许多思想一样。因此我认为 Perl 6 在许多方面都是最好的。遗憾的是，很少有人有机会使用它。

00:22:52 - Saron Yitbarek:

你认为 Perl 6 社区的发展需要做些什么？想让人们更多地参与到 Perl 6 中，需要做些什么？

00:23:00 - John Siracusa:

这有点像 Perl 6 的故事，就像它一直在寻找一个真正奇妙的实现一样。这是第二系
second-system syndrome
统问题的一部分……第二系统综合症，我想他们是这样称呼……

00:23:11 - Saron Yitbarek:

哦。

00:23:12 - John Siracusa:

Perl 6 的，人们希望修复世界上的所有问题。他们想要解决的问题之一是运行时环
境。是什么在运行我们的代码？运行 Perl 5 和之前的 Perl 4 的东西是一个巨大的
C 程序，这是由具有独特编码风格的开发者编写的。还有大量的宏，它是一种相当
难以理解的东西。

00:23:33:

Perl 6 的想法是让我们不要再那样做了，让我们不要制造大量的 C 代码。相反，让
我们使用一个虚拟机，这在当时是一种时尚，有很多关于如何实现它的伟大想法。
最终我们得到了几个中规中矩的虚拟机实现版本，有时这些还会相互竞争，但没有
一个达到真正交付语言使用时需要的性能、稳定性和特性。

00:24:01 - Saron Yitbarek:

现如今 Perl 到底发生了什么？你对此有什么看法？

00:24:06 - John Siracusa:

Perl 5 绝对像是在走下坡路，因为与 Perl 5 同时代的所有其他语言都采纳了它的大
部分最佳思想，并获得了更多的支持。也就是说，因为它在很长一段时间内都是王
者，所以有很多 Perl 5 代码在运行一些大型的、重要的站点，人们需要维护和扩展
这些代码。

00:24:29:

这需要很长时间才能消失。只要看看现今仍然存在的 COBOL，人们怎么还在雇佣
人在 COBOL 上做维护吧？

00:24:35 - Saron Yitbarek:

嗯。是这样。

00:24:36 - John Siracusa:

你刚才问 Perl 是不是一门垂死的语言，我提到了 COBOL，这听起来并不乐观。
Perl 6 本身会成为主流语言吗？看起来可能性不大。现在对其他语言有非常多的关
注，如果 Perl 6 现在还没有得到开发者的关注，我不知道将会需要发生什么变化来
让它流行起来。

00:24:54 - Saron Yitbarek:

如果你是这样想的，你对 Perl 有什么期望？你希望在 Perl 5 或 Perl 6 中看到什
么，以及希望看到将来发生什么？

00:25:04 - John Siracusa:

我对 Perl 5 的希望是人们不要忽视它，因为尽管有其它更流行的语言，但今天许多公司仍然采用 Perl 5 做为解决问题的最佳方案。通常这些都是胶水类型语言的问题。如果你发现自己曾经编写过 shell 脚本，并且可能会说：“好吧，我不会用我的‘真正的编程语言’来做这件事。”不管是 Python，还是 Ruby，还是别的什么。但是 shell 脚本可以让我把一堆东西连接起来（胶水类型语言）。Perl 是完成这项工作的更好工具。编写正确的 Perl 脚本要比编写正确的 shell 脚本更容易。

00:25:40 - Saron Yitbarek:

我认为归根结底 Perl 可能不再是一个适合入门的语言，但对于经验更丰富的多语言开发人员来说，它是那个你永远不希望扔掉的工具箱中的小工具，而且特定的工具有时让你提升水平的工具。

00:25:58 - John Siracusa:

有时我为 Perl 6 感到难过和沮丧，认为它不会有任何进展，有时我想“好吧，这是一个不错的小社区”。每个社区都不需要称霸世界，也不需要成为整个行业的主导语言。也许可以就这样一直走下去，就是，无限期地走下去。这就是开源，和编程语言的伟大之处。没人可以阻止你，你可以像以前一样继续开发 Perl 6。

00:26:27 - Saron Yitbarek:

John Siracusa 是一名程序员，也是三个科技播客的联合主持人。

00:26:34:

语言都是有生命周期的。当新的语言出现时它们能够精确地适应新的现实，像 Perl 这样的选择可能会占据更小的、更小众的领域，但这并不是一件坏事。我们的语言应该随着我们需求的变化而扩大或缩小它们的群体。在互联网开发的早期历史中，Perl 是一个至关重要的角色，它以各种方式与我们联系在一起，只要看一看它的历史就会发现它的存在。

00:27:11:

下次在《代码英雄》中，我们将讨论：是什么将一种语言变成了标准？以及在基于云的开发世界中，新的标准将如何出现？

00:27:26:

《代码英雄》是红帽的原创播客。如果你想深入了解 Perl 的故事，或者任何我们在第三季中探索的编程语言，请访问 redhat.com/commandlineheroes。我们的网站里有许多精彩内容等你去探索。

00:27:49:

我是 Saron Yitbarek。下期之前，编码不止。

什么是 LCTT SIG 和 LCTT LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 [LCRH SIG](#) 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-3/diving-for-perl>

作者: [Red Hat](#) 选题: [bestony](#) 译者: [Mikedkmilk](#) 校对: [Northurland, wxy](#)

本文由 [LCRH](#) 原创编译, [Linux中国](#) 荣誉推出

《代码英雄》第三季（5）：基础设施的影响

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频播客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《[代码英雄](#)》系列播客《代码英雄》第三季（5）：基础设施的影响的音频脚本。

导语：用在 IT 基础设施中的语言是没有有效期的。COBOL 已经存在了 60 年——而且不会很快消失。我们为大型机维护了数十亿行经典代码。但我们也正在用 Go 等语言为云构建新的基础设施。

COBOL 是计算机的一次巨大飞跃，让各行各业变得更加高效。Chris Short 介绍了学习 COBOL 是如何被视为安全的长期投注的。60 年后的今天，还有数十亿行不容易被替换掉的 COBOL 代码——懂这门语言的专家也很少。Ritika Trikha 解释说，有些事情必须改变。要么更多的人必须学习 COBOL，要么依赖 COBOL 的行业必须更新他们的代码库。这两个选择都很困难。但未来并不是用 COBOL 编写的。今天的 IT 基础架构是在云端构建的，其中很多是用 Go 编写的。Carmen Hernández Andoh 分享了 Go 的设计者是如何想要设计一种更适合云计算的语言。Kelsey Hightower 指出，语言通常都是超专注于一种任务。但它们正变得越来越开放和灵活。

00:00:00 - Saron Yitbarek:

1904 年，纽约市地铁首次开始运营时，它被惊叹为现代的一个奇迹。但是……当今天的通勤者仍依赖一个多世纪前设计的基础设施时，会发生什么？列车挤满了人，而且常常晚点。纽约每年有 20 亿人次地铁出行，再也没有人为此感到惊叹了。我们还在依赖昨日的过时基础设施，我们必须找到新的好办法，让它发挥作用。

00:00:44:

过去，基础设施项目通常是些可见的大而具体的事物，例如地铁。而且由于这种物理可见性，它们损坏时也显而易见。高速公路开裂、电线杆倒下，我们很清楚这些东西何时需要维修。为了使我们的生活与老化的基础设施保持协调，大量的工作是必不可少的。

00:01:12:

但是事物并不总是那么一是一、二是二。如今，我们还拥有 IT 基础设施，在偏僻地区嗡嗡作响的服务器农场，跨越海洋的光纤电缆，还有软件基础设施。而像遗留的操作系统或没人敢替换的 shell 脚本，这些 IT 基础设施变得陈旧和破旧时，我们是无法看出的。但是，造就了今日发展的基础设施却正在老化，就像旧的地铁轨道一样。这可能会扰乱我们的现代生活。如今命令行英雄们正努力确保我们不会被过去束缚，因此出现了大量新的挑战。

00:02:02:

这是我们第三季进入编程语言世界探索的第 5 期。我们将研究两种编程语言，它们与最初设计的目标基础设施密切相关。**COBOL** 是一种大型机的原生语言，而 **Go** 是云计算的原生语言。它们都深受其起源的影响。理解这一点可能会让明天的开发者不至于像纽约人一样被塞进宾夕法尼亚州的车站。

00:02:33:

我是 Saron Yitbarek，这里是红帽的原创播客，《命令行英雄》的第三季。

Grace Hopper

00:02:43 - 格蕾丝·赫柏：

我们面前有很多事情需要去做，但是我们首先需要大量相关的且易于访问的信息。我们才刚刚开始。

00:02:53 - Saron Yitbarek:

Grace Hopper

海军上将格蕾丝·赫柏

在 20 世纪 40、50 年代率先开发了高级编程语言。而她之所以能够实现这种巨大的飞跃，是因为当时的基础设施，大型计算机。

00:03:08 - Chris Short:

嗨，我叫 Chris Short。

00:03:10 - Saron Yitbarek:

Chris 是红帽的首席产品营销经理，而且他也是一位历史爱好者。

00:03:17 - Chris Short:

上世纪 40 年代的赫柏上将创造了 FLOW-MATIC，这在当时是革命性的，她被广泛认为是 COBOL 的祖母。因此，她能够坐在那里说：“嘿，只需将其放在大型机上”，或“嘿，只需将其存储在大型机上”即可。

00:03:31 - Saron Yitbarek:

这是一个重大的游戏规则改变。突然，你有了这种机器无关的语言，即 COBOL，它是大型机环境特有的。可能性开始逐步打开。

00:03:42 - Chris Short:

大型机和 COBOL 真正使得每个组织能够说，它们不再需要充满了带着铅笔、纸、计算器和滑尺的人的办公室，他们可能只需要半个办公室来安装大型机。然后，他们可以雇人用 COBOL 来编写一些应用程序来完成整个财务团队做的所有的数学、逻辑运算以及账目。因此，你需要的财务团队人数少了，仅仅是因为更多的输入可以被数字化，而不是全手动操作。

00:04:17 - Saron Yitbarek:

如果你是那些新来的 COBOL 程序员之一，你会觉得你有了一份终身的工作。因为你的工作所基于的基础设施——所有那些大型机——始终会在那里。

00:04:30 - Chris Short:

那时候摩尔定律还未出现，所以你可以整整十年都在同一个大型机上工作，对吧？就像你不用去考虑下一个操作系统，或者下一个类型的容器编排器，又或者下一个出现 AI 之类的东西一样。你可能会整个职业生涯都在从事 COBOL。而且你知道自己的生活将会非常稳定。

00:04:55 - Saron Yitbarek:

但是，摩尔定律最终还是来了。新的基础设施也出现了。现如今，程序员不太可能去学习一种半个世纪前的旧语言。但实际上，那些老旧的大型机其实并没有消失。这意味着我们对 COBOL 开发人员的需求也没有消失。

00:05:17 - Chris Short:

寻找 COBOL 开发者变得越来越困难。最终会发生的事情是这些大型机可能已经存在了 50 年。这些仍然可以编写出色 COBOL 程序的开发人员将获得巨额收入，以帮助项目运行并完成大型机中的数据重组。而且，该技能肯定会绝迹，并且正在成为一个利润丰厚的职业领域，如果你……现在写 COBOL 绝对可以赚很多钱。

00:05:49 - Saron Yitbarek:

尤其是在制造业和金融业。你无法超越几十年前建立的所有基础设施。遗留代码遍及全球。忽略这些老旧的基础设施及其相关的语言，将是一个巨大的错误。

00:06:08 - Chris Short:

有两千亿行代码摆在那里，要重构这些代码真的很难。不，我不认为在有生之年我们会看到它消失，真的。

00:06:21 - Saron Yitbarek:

Chris Short 是红帽的首席产品营销经理。

00:06:28:

我想花一秒钟解释一下 Chris 的观点。你想想看，95% 的 ATM 交易中都有 COBOL 代码，那就是我们与这种语言的联系。但是，COBOL 程序员的平均年龄并不比该语言年轻多少。他们 45 岁，或许 55 岁。新人们并不感兴趣这门语言。这就是为什么我想向你介绍一个人。

00:06:56 - Ritika Trikha:

嘿，我是 Ritika Trikha。

00:06:59 - Saron Yitbarek:

Ritika 是一名技术编辑，曾在 HackerRank 工作。她对 COBOL 的这个问题着迷：人们认为 COBOL 是后大型机时代无意义的残留品。

00:07:12 - Ritika Trikha:

如今的开发人员根本不会考虑 COBOL 了，见也没见过，想也没想过。

00:07:17 - Saron Yitbarek:

但这可能是灾难的根源。

00:07:21 - Ritika Trikha:

如今，仍然有大量的 COBOL 代码在驱动企业的业务。每年至少新增 15 亿行 COBOL 新代码。我认为当你看特定行业时，真的很有意思。就像美国国税局有 5000 万行代码。社会保障局有 6000 万行代码。因此，这些单位和实体正在处理一些如今最敏感、重要的信息，如果我们不继续为这些大型机提供支持和维护，就会造成很大的破坏。

00:08:04 - Saron Yitbarek:

因此，如果我们无法摆脱旧的基础设施，又无法挥舞魔杖来重建整个大型机业务，我们该怎么办？编码人员有时候仅考虑未来，该如何接受过去？我们首先需要直面该问题。

00:08:25 - Ritika Trikha:

你知道，年轻一代将不得不重拾这些技能。或者，必须对这些大型机进行某种现代化改造。无论是哪种方式，这个问题都不会消失。这就是为什么 COBOL 还活着的原因。

00:08:35 - Saron Yitbarek:

这并不容易。Ritika 认为我们已经忽略这个问题太长时间了。

00:08:42 - Ritika Trikha:

这非常昂贵、艰巨，并且替换数十亿行 COBOL 代码的风险也非常高。它是用于关键任务的代码，比如社会保障和金融信息。COBOL 是专门为此类大量交易而设计的。因此，它由格蕾丝·赫柏在 60 年代为商业交易而设计。自上世纪 60 年代以来，一直存在“如果没坏，为什么要修复它”的说法，现在我们处于这样一个关头，即延续了数十年的大量的高价值数据运行在 COBOL 上。

00:09:22 - Saron Yitbarek:

从某种意义上说，Ritika 在呼吁一种文化的转变。改变对 "进"与 "退"的态度。由于发展的世界慢慢有了越来越久的历史，我们会更加地接触到自己的历史。你无法摆脱老化的基础设施。这意味着你也不能忽略编程语言的历史。

00:09:47 - Ritika Trikha:

有些事情必须得做。当我在 HackerRank 时，我亲眼看到了多少银行和金融机构对 COBOL 开发人员的伤害，几乎是绝望的。这不是一个会被解决的问题，我认为要么必须有某种现代化的系统，要么我们继续培训人员并激励他们。我个人认为将会有 COBOL 再次出现的一天。真的，当所有拥有 COBOL 知识的开发人员退休，并且没有新一代的开发人员学 COBOL 时，将会发生什么？总得做点什么，对吧？所以，当从 COBOL 转向新的基于云的基础设施时，需要有更多的系统化和制度化的改变。

00:10:37 - Saron Yitbarek:

Ritika Trikha 是一名旧金山的技术作家。

00:10:49 - Saron Yitbarek:

那么 Ritika 提到的那些基于云的基础设施呢？我们今天建立的基础设施是否会将后代绑定到特定的语言，像我们仍绑定找 COBOL 上一样？

Amazon Web Services

亚马逊 Web 服务（AWS）可能是最大的单一云基础设施，于 2006 年推出。

Google Cloud Platform

Google 云 平 台（GCP）于 2008 年问世，微软 Azure 于 2010 年问世。Go 语言以并发为重点，定位于在新的云基础设施上蓬勃发展。这是这个时代的语言。

00:11:26 - Carmen Andoh:

嗨，我叫 Carmen Andoh，我是谷歌 Go 团队的项目经理。

00:11:34 - Saron Yitbarek:

Carmen 对 Go 语言与今天的基础设施有怎样的联系有深入的理解。这要从 Go 的创作者和编程语言历史的紧密联系说起。

00:11:47 - Carmen Andoh:

Robert Pike、Robert Griesemer 和 Ken Thompson。这些名字算是从上世纪 60 年代就开始出现了。Ken Thompson 发明了 B 语言，然后他在夏天的假期继续发明 UNIX 操作系统。Rob Pike 发明了字符串编码 UTF-8，他还发明了 ASCII。他帮助 Ken Thompson 共同编写了 UNIX 编程环境。所以，这两个人是很多、很多年前的同事，他们一直在研究和发明用以前的编程语言编写的操作系统，这些语言包括 Ken Thompson 最终帮助 Dennis Ritchie 一起编写的 C 语言。

00:12:28 - Saron Yitbarek:

Pike、Griesemer 和 Thompson 在 Google 工作之后，他们发现了一个严重的问题。并没有出现大规模的并发。人们等待了几个小时编译出来。他们使用的是 C++，并且必须得编写所有这些回调和事件调度器。那是在 2009 年，我们的基础设施再次发生了变化。诸如 C++ 之类的语言越来越不适应这种新的现实。

00:12:59 - Carmen Andoh:

多核处理器、网络系统、大规模计算集群和 Web 编程模型等正在引入这些问题。而且，还有这个行业的增长，程序员数量在 2010 年就会达到成千上万。因此，直到那时，所有的编程语言都是在规避问题而不是在正面解决问题。

00:13:24 - Saron Yitbarek:

最终，将达到一个临界点，必须开始改变。

00:13:30 - Carmen Andoh:

嘿，我们讨厌 C++，我说：“好吧，让我们看看我们是否能发明些新的东西。”

00:13:37 - Saron Yitbarek:

这种新语言需要完美地适应我们最新的基础设施。

00:13:43 - Carmen Andoh:

2005 年云技术到来以后，你不再需要自己的计算机，在某种程度上在其他地方租用它，你就可以得到一个分布式系统。但是在分布式系统中，以及在云计算中，存在并发消息传递问题。你需要确保采用异步对你来说没有问题。Go 缺省就是异步的编程语言。基本上，这意味着你执行的每个操作（例如将所有这些不同的消息发送给系统中的另一个计算机）都无需等待另一个机器对你的响应即可完成。因此，它可以在任何给定时间处理多个消息。

00:14:28 - Carmen Andoh:

就是说，云计算是分布式的。因此 Go 的开发就是来满足这一确切需求。Go 早就成为进行这种分布式计算的标准方法之一。这就是为什么我认为它立即引起了开发人员的广泛关注。Go 绝对是云基础设施的语言，无论是其设计，还是所有云基础设施工具，以及在过去十年中如雨后春笋般出现的模块的生态。

00:15:06 - Saron Yitbarek:

很快，诸如 Kubernetes 之类的关键应用都用 Go 编写了。谷歌还创建了 Go Cloud，这是一个开源库和一系列工具，使得 Go 更具吸引力。很显然，它是新生态系统的语言。它是云的语言。而且，它的创造者们因开发生命力持久的语言而享有声誉，这绝对没有坏处。

00:15:33 - Carmen Andoh:

我认为业界的其他人会说：“嘿，我认为这不会很快消失。”这种语言的发明者恰巧也发明了语言有 50、60 年了。

00:15:47 - Saron Yitbarek:

Carmen Andoh 是谷歌 Go 团队的项目经理。

00:15:54:

因此，我们有了一种新的语言 Go，旨在提供云基础设施必需的并发性。听起来不错。Go 的设计师倾向于创造可以持续半个世纪的语言。这也很棒。但是我的问题是，从现在起，50 年后，当 Go 更像是 COBOL 时，这到底意味着什么？当世界上充满了只有老开发人员才能理解的旧版 Go 代码时，这又意味着什么？在当今的云基础设施老化的时候，我们是否会做好准备？我们是否从 COBOL 和大型机领域吸取了教训，可以帮助我们为 Go 和云设计更美好的未来？

00:16:40:

幸运的是，我找到了问所有这些问题的合适人选。这就是下面这位。

00:16:51:

我们如何使我们的语言能面向未来？我们知道他们与当今的基础设施息息相关。我们也知道，随着数十年的发展，新的基础设施必将取代旧的基础设施。那么，我们今天做些什么以确保将来能平滑演进？

00:17:10 - Kelsey Hightower:

我是 Kelsey Hightower，我在谷歌，是一名开发人员推广大使，我致力于引入开放性技术并将它们应用于谷歌云上的产品。

00:17:19 - Saron Yitbarek:

Kelsey 花了大量时间思考编程的未来。我很好奇，是否有一天我们将陷于握有 Go 语言技能的是另一批老龄化的程序员的问题，就像我们现在缺少 COBOL 的引导一样。我们是否在为这个长远的未来做计划？因此，我和 Kelsey 坐下来进行讨论。

00:17:42 - Kelsey Hightower:

...等等。但是，如果你考虑到今天面临的一些新的挑战，如应对互联网，这种网络，你将面临许多用户，成千上万的并发用户，以及不同的机器和架构类型的组合。考虑到这些新的场景，因此你通常希望有一种新的语言来解决。例如，JavaScript 是用于 Web 的，你不会想改造 COBOL 以便可以用它来进行 Web 编程。最终，我们今天已经有了数百种相当完善的语言，而且它们都非常专注于他们的优势。

00:18:15 - Saron Yitbarek:

那么，在那种情况下，我们是否需要积极推动人们走向 COBOL？如果我们正在为这些新问题开发新语言，并且它们是高度定制化的，而 COBOL 仍在坚持不谢幕，我们是否需要鼓励人们选择它，以便我们可以维护我们的旧代码？

00:18:32 - Kelsey Hightower:

好吧，我认为这将对企业是个挑战吧？所以，你已经在 COBOL 上投入了 10 到 20 年，没有人会主动想学习一些新的 COBOL。或者你不会像刚从大学毕业那么“我要加倍努力……”。

00:18:45 - Saron Yitbarek:

没错。

00:18:46 - Kelsey Hightower:

“...这种语言比我父母的年龄都大。”因此，在那个领域，你必须问自己，继续使用 COBOL 的风险是什么？未来是否仍然有意义？我认为它仍然有益于某些类型的工作任务，但是我们必须问自己一个问题，是时候推进了吗？是时候进化一点了吗？因此，如果你仍然有数十亿行的 COBOL 代码，那么你将必须寻找所有剩余的 COBOL 人才，并将其招进来，但也许我们该开始考虑其他语言能从 COBOL 学习些什么，并将某些功能和库加入到其他语言中。

00:19:26 - Saron Yitbarek:

COBOL 终止以后的日子，将会是一个巨大的基础设施项目。用我对纽约地铁的比喻，就像是要替换每条地下隧道。因此，展望未来，我想知道我们是否可以预见到这些问题，甚至将来对自己也能有所帮助。

00:19:48:

如果我们将今天的云计算与大型机进行比较，我们是否会在最终到达同一条船上，有着这些旧代码库，使用着旧的但非常稳定的语言，而且我们也到了必须做出选择的时候，我们应该继续前进还是保持不变？

00:20:05 - Kelsey Hightower:

云有点不同的是它不是来自一个制造商，对吗？许多云提供商通常提供了一系列技术集合，你就可以选择编程语言、编程范式，无论你是要事件驱动还是基于 HTTP 的全 Web 服务。这意味着你可以选择编程的方式，然后只需专注于要解决的问题。因此，数据进进出出，但是你来选择处理数据的方式。

00:20:36:

大型机通常只有一个主界面，对吗？就像你编写一份任务一样，这就是你提交任务的方式，这就是你监控任务的方式，这就是结果。这本身就是一个限制。如果你考虑一些较新的大型机，它们也会支持些较新的技术，因此即使在大型机领域，你也会开始看到可用于运行任务的编程语言扩展。

00:20:58:

那么我们开始问自己，好吧，既然我有了新的选项，该什么时候离开这种特定的编程范式继续前进呢？我认为我们不会陷入困境。

00:21:08 - Saron Yitbarek:

正确。

00:21:08 - Kelsey Hightower:

我认为新的分布式机器很不错，可能起步成本更低，你不必购买整个大型机即可开工。但是我们仍然希望易用性和之前一样：给你我的工作，你为我运行它，完成后告诉我。

00:21:24 - Saron Yitbarek:

绝对是这样，你觉得发生的事情，或者说 COBOL 所发生的事情，会发生在今天的任何一种语言上吗？以 Go 语言做例子，你看到我们在努力地改进 Go 从而吸引人们在 30 年后还想用 Go ？

00:21:38 - Kelsey Hightower:

我认为所有语言都会遭受这种命运吧？如果你仔细想一下，其实 Python 已经存在很长时间了。我想差不多 20 年了，甚至更久。因此，我认为会 Python 重新流行起来了，它现在是机器学习的基础语言。

00:21:53 - Saron Yitbarek:

是的。

00:21:54 - Kelsey Hightower:

对于 **Tensorflow** 之类的库，如果我们仅用时间来衡量，我认为这可能不是看待它的正确方式。还应该有社区是否活跃？语言的适配意愿是否活跃？我认为 **Python** 做得确实非常出色……社区看到了使其他语言更易于使用的能力。例如，**Tensorflow** 底层有很多 **C++**，使用这种语言编程可能没有 **Python** 这样的用户友好性。你可以用 **Python**，并用它来生成人们正在使用的一些东西，例如 **Tensorflow**。现在机器学习非常热门，人们将 **Python** 引入了这个新领域，那么你猜怎么着？**Python** 仍然是活跃的，并且在未来的一段时间内都是活跃的。对于 **Go** 来说，这同样适用。如果 **Go** 能够继续保持活跃度，那么，它就像许多基础设施工具和许多云库的基层一样，它也将保持活跃。因此，我认为都是这些社区确保了它们将来占有席之地，并且当未来出现时，确保那里仍有它们的位置。

00:22:58 - Saron Yitbarek:

是的。那么，我们如何让我们的语言面向未来呢？就是说，我们如何有意地设计一种语言，使其能持续存在，并在 20、30 年内都保持活跃呢？

00:23:10 - Kelsey Hightower:

使用语言的人，我认为这在开源世界中确实是独一无二的。现在，我们已经不再使用商业语言了，对吧，过去曾经来自微软的语言或太阳微系统的如 **Java™**，那时候，每个人都依赖于供应商来尽心尽力来对语言能干什么以及对运行时环境进行新的改进。现在，我们看到的诸如 **Go**、**Node.js**、**Ruby** 之类的东西，所有这些都是社区支持的，并且社区专注于运行时环境和语言。这样任何人都可以添加新库，对吧？有一个新的 **HTTP** 规范，对，**HTTP/2** 几年前问世了，每个社区都有贡献者添加了那些特定的库，猜猜现在怎么样？所有现在这些语言，都兼容于大部分的未来网站。

00:24:01:

我认为现在是个人真正地拥有了更多的控制权，如果他们想让语言适用于新的用例，只需要自己贡献即可。因此，我们不再受限于一两家公司。如果公司倒闭，那么运行时环境可能会因此而消亡。我们不再有这个问题了。

00:24:23 - Saron Yitbarek:

我们之前在这个播客上已经说过了。未来是开放的。但是，令人着迷的是考虑怎样能做到再过几十年，过去也将是开放的。它们将继承能够变形和演进的基础设施和语言。

00:24:39 - Kelsey Hightower:

太棒了，感谢你的加入，我期待人们的工作，而且大型机仍然有意义。它们是经典技术，因此我们不称其为遗产。

00:24:47 - Saron Yitbarek:

哦，我喜欢，经典，非常好。

00:24:51:

Kelsey Hightower 是 **Google** 的开发者推广大使。

00:24:57:

我正在想象一个充满经典编程语言以及尚未诞生的新语言的未来。那是我为之兴奋的未来。

00:25:07 - 演讲者:

请离关着的门远一点。

00:25:12 - Saron Yitbarek:

要知道，2017 年 Andrew Cuomo 州长宣布纽约市地铁进入紧急状态。他的政府拨出 90 亿美元投资于老化的基础设施。这应该提醒我们，迟早我们得注意遗留的系统。你不仅需要继续前进，面向未来。你还背着历史包袱。

00:25:37:

在开发的世界中，我们倾向于偏向未来。我们认为我们的语言仅在它们流行时才有用。但是，随着信息基础架构的不断老化，开发的历史变得越来越真实。事实证明，过去根本没有过去。记住这一点是我们的工作。

00:26:05:

你可以前往 redhat.com/commandlineheroes，以了解有关 COBOL 或 Go 或本季我们要介绍的其他语言的更多信息。那里有很多很棒的材料在等你。

00:26:19 - Saron Yitbarek:

下一集是关于 Bash 的。我们将探索 shell 脚本的起源以及自动化的关键。

00:26:30 - Saron Yitbarek:

《命令行英雄》是红帽的原创播客。我是 Saron Yitbarek。下期之前，编码不止。

什么是 LCTT SIG 和 LCTT LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 LCRH SIG 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-3/the-infrastructure-effect>

作者: Red Hat 选题: bestony 译者: messon007 校对: wxy

本文由 LCRH 原创编译，Linux中国 荣誉推出

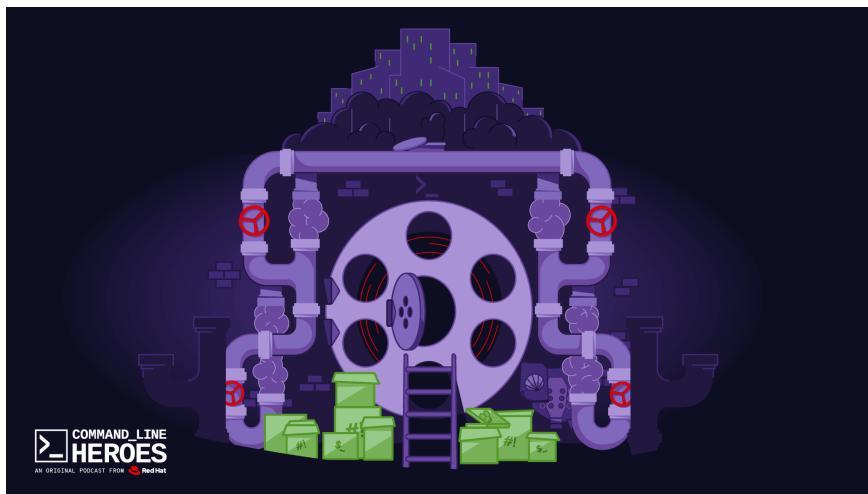
《代码英雄》第三季（6）：Bash Shell 中的英雄

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频播客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《[代码英雄](#)》系列播客《代码英雄》第三季（6）：Bash Shell 中的英雄的[音频脚本](#)。

导语：Shell 使得大规模 IT 成为可能。它们是现代计算的必要组成部分。但 Free Software Foundation

是，如果没有自由软件基金会一位名叫 Brian Fox 的开发者的辛勤工作，它可能不会变成这样。现在，世界上几乎每台电脑都有 Bash shell。

Bell Labs

在上世纪 70 年代，贝尔实验室希望将重复的、复杂的命令序列自动化。

Chet Ramey 描述了贝尔实验室是如何开发出几个 shell 的——但 UNIX 只能有一个官方支持的 shell。获选的是 Bourne shell。尽管 Bourne shell 是这些之中最好的一个 shell，但它也有其局限性。而且它只有在受到限制的 UNIX 许可证下才能使用。Brian J. Fox 讲述了他在自由软件基金会的工作，他需要创建一个自由的 Bourne shell 版本。它必须兼容但不使用任何原始源代码的元素。这个 Bourne-Again Shell，即 Bash，可能是这个星球上使用最广泛的软件。而 Taz Brown 描述了它是如何成为一个开发者可以学习使用的最重要的工具之一。

00:00:07 - Saron Yitbarek:

那是 1987 年。里根总统治下的美国正蓬勃发展，一个怀揣远大梦想的人正驱车前往他位于圣巴巴拉的新家。这个人名叫 Brian Fox，27 岁，是高中辍学生。在他车的后备箱里，有两盒巨大的磁带，里面载满了他当时正在编写的代码。

00:00:28:

Fox 多年来一直以程序员的身份工作在所谓的自由软件运动中。他相信他锁在这个后备箱里的代码，可以带来一场革命，这是一种全新的软件范例。他的社区正在一点一点地使之成为现实。

00:00:49:

Richard Stallman Free Software Foundation

那年，理查德·斯托曼（RMS）的自由软件基金会的一组程序员，正在想尽办法给计算机界带来自由。他们想要构建一个 UNIX 的替代品，以取代自从 70 年代以来就主导编程的 UNIX 操作系统。他们的 GNU（表示 GNU's not UNIX）将成为公众的操作系统，任何人都可以使用它，无需担心许可费用或版权问题。

00:01:18:

多年以来，基金会一直在努力制造这个崭新的系统。那么 Brian Fox 汽车后备箱里的那两盒装着代码的巨型磁带是什么？它们存储着这个系统一个至关重要的组成部分。这是一个自由的，而且可更改的 shell，它能够使 GNU 操作系统变得完整。这是 Brian Fox 送给自由软件运动的礼物。他称之为 Bash。

00:01:46:

Command Line Heroes Red Hat

我是 Saron Yitbarek，这里是 代码 英 雄，一档来自红 帽的原创播客节目。在这一集中，我们将来看看 Bash shell 中的英雄们。我们将探索 shell 的历史，以及它们为什么对我们如今的工作如此重要。大家可以将 shell 看作要给演员的剧本。它们提供了完整的命令序列，然后 shell 可以快速地运行，就像演员可以一行接一行地读她的台词一样。这是对于实现重复且复杂的代码的，是最终的解决方案，也是自动化的关键。你可能会说，shell 脚本是我们开发的一大助力。但是，是否可以编写一个，能给所有人带来帮助的 shell？这就是挑战所在。

00:02:38 - Ken Thompson:

让我们回到 1969 年。那时候贝尔实验室的几位计算机科学家，正在根据自己的需求开发程序。

00:02:48 - Saron Yitbarek:

这位是代码英雄先驱 Ken Thompson。由贝尔实验室设计的 UNIX 操作系统，在一开始确实是供他们个人使用的。最初，它只是一个内部系统。UNIX 鼓励程序员之间进行密切的交流，不过目的并不是要改变整个世界，而是改变贝尔实验室。

00:03:13 - Ken Thompson:

到现在，几乎整个贝尔实验室都在使用这个系统。我们公司拥有近两万个计算机终端，其中大多数使用 UNIX 系统。

00:03:25 - Saron Yitbarek:

一款由 Ken Thompson 所设计的 UNIX shell 在 1971 年发布。虽然 Thompson shell 被设计为命令行解释器，但是它却不能很好地支持脚本。所以直到六年后的 1977 年，脚本才开始兴起。

00:03:44 - Chet Ramey:

Shell 参数、特殊参数以及我们如今认为理所当然的变量，起源于 Steve Bourne 和 Bourne shell。

00:03:57 - Saron Yitbarek:

这位是 Chet Ramey，Case Western Reserve 大学的 IT 架构师。Chet 致力于维护 Bash，他也为我们讲述了许多 Bash 的起源故事。他描述了贝尔实验室当时研究 UNIX shell 样子时的情景。

00:04:13 - Chet Ramey:

我们如今使用的编程结构起源于 Steve Bourne，他的 shell 赢得了这场比赛。当时有大量使用 Mashey shell 的用户社区，也有大量用户开始使用 Bourne shell。那时候成立了一个委员会来决定哪一个将会获胜，哪一个将会成为从那时候起得到官方支持的 UNIX shell，Bourne 的 shell 赢了。而其他的 shell，正如他们所说，成为了历史。

00:04:54 - Saron Yitbarek:

不过，这还不是历史的终结。当然，Bourne shell 是一个巨大的飞跃。它打开了一扇通向更高自动化水平的大门。但是尽管有一段时间 Bourne 占据了上风，但是 Bourne shell 并不能解决我们所有的脚本需求。

00:05:14 - Chet Ramey:

Bourne 撰写自己的 shell 时所受到的限制，几乎是现在的你我难以想象的。显然，当你遇到这些限制时，你不得不放弃很多东西，Bourne 就放弃了很多。考虑到他所处理的空间、内存和 CPU 限制，他能够让 Bourne shell 包含那么多东西，这相当了不起。

00:05:42 - Saron Yitbarek:

请记住，Bourne shell 仍然是贝尔实验室 UNIX 系统的一部分。它仍然与 UNIX 许可证绑定。这意味着它不是自由的，不是开放的。这款 shell 是私有的。

00:05:55 - Chet Ramey:

如果你不在大学里，获取 UNIX 源码将会非常困难。显然，这对 Berkeley UNIX 的普及产生了影响。Berkeley UNIX 始于大学中，在大学社区中成长，并走了一条阻力最小的道路。因此，如果你在正确的地方，访问到 Bourne shell 的源码并不困难，但是总的来说，这并不是大众都能够认可的方案。

00:06:36 - Saron Yitbarek:

Chet Ramey 是 Bash shell 的维护者。

00:06:41:

因此，我们有了 shell 的雏形，可以着手写这些关键的组成部分，但是目前为止，最好的 shell 的许可证却有个大大的问题，它是闭源的。对于理查德·斯托曼和他的自由软件基金会而言，这是绝对无法接受的事情。我们所需要的是一个不与任何公

司绑定的 shell，一个面向所有人的 shell。

00:07:05:

但这就带来了问题。这意味着我们需要编写某种，能做到 Bourne shell 所能做到的一切，而又不会侵犯到版权的东西。如果逐字复制 Bourne shell 的代码，你会被起诉。

00:07:20:

为了使人们摆脱 Bourne shell 的束缚，你必须找到一位能在没看过 Bourne shell 任何源代码的情况下，编写这款复杂程序的程序员。你必须找到这样的一位局外人天才。而理查德·斯托曼找到了完成这项工作的程序员。

00:07:46:

Brian Fox 是一名 20 来岁的高中辍学生，比贝尔实验室的大多数人更懂代码。他从来没有见过任何 Bourne shell 的源代码，这使得他非常适合手头的任务。

00:08:02 - Brian Fox:

我是 Brian Fox。

00:08:04 - Saron Yitbarek:

为什么不直接问问这个年轻人，这个故事是什么样的呢？现如今，Fox 是一位开源倡导者以及 Opus Logica 的 CEO。但是早在 80 年代后期，他只是一个信仰开源软件运动的年轻人。我们聊了聊过去的日子，以及 Bash 是如何从那时演变过来的。

00:08:23:

所以那时候理查德·斯托曼请你为 UNIX 开发一款 shell。那将会是一款自由的 shell，并且是 Bourne shell 的替代品。你是如何回应的呢？

00:08:38 - Brian Fox:

“我们就不能做个更棒的吗？”

00:08:41 - Saron Yitbarek:

我喜欢这个。再多跟我说说。

00:08:45 - Brian Fox:

我为斯托曼所做的第一件事，其实就是编写个信息技术文档系统。我让理查德惊讶于我做这种编程的速度。他是个优秀的程序员而且工作的很快，但是他不认为其他人也能写得那么快。

00:09:00 - Brian Fox:

因此，在第一周内，我完成了一款名为 GNU Info 的程序的第一版实现，理查德对此有点儿震惊。我说：“我的下一个项目是什么？我的下一个项目是什么？”他说：“好吧，现在给它做个编译器吧。”我就做了，一周时间之内就完成了。然后我说：“我的下一个项目是什么？我的下一个项目是什么？”他说：“好吧，另一个家伙一直在研究那个 shell，但他还没有太多进展。”我说了“好的”，九个月后，Bourne shell 的替代品完成了。

00:09:29 - Saron Yitbarek:

九个月，哇。再多告诉我一些。为什么它如此具有挑战性？

00:09:33 - Brian Fox:

这真是个有趣的问题。它之所以如此具有挑战性，是因为我们必须忠实地模仿 Stephen Bourne 最初的 Bourne shell 的所有行为，同时对其进行扩展，让它成为人们能使用的、更好的工具。

00:09:51:

那时候，我和 Korn shell 的作者 David Korn 私下进行了秘密争论。POSIX 委员会，也就是规定了什么是标准 UNIX 的委员会，他们也参与了进来，并说：“哦，很好，我们需要知道 shell 到底要包含些什么。”而这方面最重要的两个人是我和 David Korn。David Korn 已经写了一个名为 KSH 的 shell。对于他所加入到 KSH 中的每一个功能，他都说：“这应该是一个标准功能。”是这样吗？对他来说这比起拥有最完美的 POSIX shell 要容易得多，如果这仅仅是他的 shell 的话。

00:10:31:

其中的一些功能并不是很好的功能，不是很好的选择，而且使得这款 shell 与 Bourne shell 有些不兼容，或者我觉得缺少功能，对此我们进行了一些讨论和争论，因此构建一个兼容 POSIX 的 shell 与过去为 Bourne shell 所编写的每个 shell 脚本都完全兼容花了超过 3 个月时间。

00:10:54 - Saron Yitbarek:

因此，如果你正在设计的产品不仅可以取代 Bourne shell，而且还试图模仿 Bourne shell 的每个部分，听起来你可能会遇到一些版权问题。你是如何处理的？

00:11:08 - Brian Fox:

为了构建真正开源而自由的软件，你必须得在一个干净的空间里，开始做这项工作。你不能从查看别人的代码开始然后重新实现它。因此，我从未见过与任何贝尔的系统、UNIX 或者甚至 Berkeley UNIX 相关的任何软件，也从未见过这些东西的源代码。

00:11:29:

当我开始构建 Bash shell 时，我使用了一个名为 Bison 的解析器，理查德已经将其整合到自由软件基金会里，并且与之前任何的其他程序完全不同。因此，我已经知道我所要构建的东西，绝对不会侵犯任何先前构建的东西的版权。

00:11:55 - Saron Yitbarek:

创建 Bash 的工作有很多小插曲，对于那些硬核的代码英雄来说，这只是其中一个例子。

00:12:03 - Brian Fox:

globbing

有一次，我正致力于在 shell 中实现通配扩展。举例来说，这是允许你匹配大量文件的通配符扩展。你可以给出 `*.c`，而这会匹配所有带有 `.c` 扩展名的文件。

00:12:17:

因此我在通配扩展上忙活了几个小时，并且使其生效了，对此我感到很兴奋。这是一个很好的实现。而在创建这一版实现的过程中，我在我的目录里创建了一个名为 `*.c` 的文件，然后我想：“好吧，我应该删掉这个文件”，然后我输入了 RM、空

格、引号、星号点 C、闭合引号，在现代 shell 中当你使用了括号，这意味着“不要扩展这个”，然后我按下了回车，提示符过了很长时间才重新出现，因为我们正在使用 Sun 350s，运行缓慢。我意识到，之所以花了很长时间是因为它要删除这个目录里的所有源文件。

00:12:58 - Saron Yitbarek:

哦，不！

00:12:59 - Brian Fox:

是的。所以我当时删掉了 Bash 的源代码。

00:13:01 - Saron Yitbarek:

哦，不要。

00:13:04 - Brian Fox:

这——

00:13:05 - Saron Yitbarek:

哦我的天哪，嗯。

00:13:06 - Brian Fox:

这件事让我笑了很久，笑的很大声。我甚至没有感到一丝沮丧。然后在接下来的几天里，我重新输入了全部。这份代码在我脑海里是完全是崭新的。

00:13:20 - Saron Yitbarek:

哇。

00:13:20 - Brian Fox:

问题解决了。只需将其记录到文件中即可。

00:13:25 - Saron Yitbarek:

好的。因此，大多数人会在那一刻完全惊慌失措。而你笑了，只是说：“哦，我想我必须重新做一遍了。”为什么你当时那么冷静呢？

00:13:35 - Brian Fox:

这让我感到疯狂很荒唐，也非常好笑，我正在打造这个工具，而要确保自己能搞好，确保该工具正常工作，你得在构建它的过程中就使用它。但是该工具无法正常工作。我还没有实现引号，并且因为我还没实现引号，所以我输入的命令没有按照我所预期的去执行，我觉得这真的很滑稽。

00:14:06 - Saron Yitbarek:

太神奇了。

00:14:08:

不过，甚至是关于错误的这个故事也能说明 Fox 的才华。他们说莫扎特在头脑中完成了交响曲，然后只需要在完成后写下来即可。Fox 也有类似的天赋。

00:14:23:

因此，当你最终完成并交付 Bash 时，感觉如何呢？

00:14:27 - Brian Fox:

呵，其实感觉很壮观。那么这里有一个故事，其实我一般不讲的。构建这款 shell 花了大约 8 个月的时候，当时我知道，我大概还需要大约一个月时间才能完成工作，然后另一个 shell 发布了——ASH，一个开源的 shell 被发布了，我很沮丧，因为我们还没有向任何人发布 Bash shell，所以只有少数人在使用它。我知道这还需一个月的工作量，于是我想：“哦，这太糟糕了。我投入的全部能量和精力都不会得到赞许，甚至可能都不会被看见。”所以我非常沮丧。这次我没有笑。

00:15:13 - Saron Yitbarek:

然而，布丁好不好，吃了才知道。GNU 的 Bash 发布于 1989 年并且变成了 Linux 的默认 shell。如今，它是计算机中不可或缺的一部分。

00:15:25:

它无处不在。如此多的人每天都在使用它。它遍布于每一台计算机上。作为 Bash 的作者感觉如何？

00:15:34 - Brian Fox:

大多数时候，我甚至都没有注意到 Bash 是比工具更加重要的东西。我真的没有经常想这件事。每隔一段时间，我会走进一家苹果商店，环顾四周然后想：“哇，这里的每台计算机不仅运行着我 27 年前编写的软件，甚至上面还包含有我的名字。”然后我想：“互联网上的每台计算机、每台服务器都在运行着 Bash shell，并且其中包含有我的名字。”然后 Windows 在去年还是前年推出了 Power shell，就是 Bash，当时我想：“哦，天哪。我的名字遍及地球上的每台计算机了。”

00:16:21 - Saron Yitbarek:

不过，我想让你们能仔细听听 Fox 接下来告诉我的内容，因为它是很重要。他从未想过，它的程序会这样统治全球。他试图提供帮助，试图帮助他所置身其中的编程文化。

00:16:37 - Brian Fox:

我并没有打算去实现出现在每个人的计算机上这样的宏伟目标。我对此一点都不感兴趣。我想制作一款有用的软件，我希望它有典型的 3 到 5 年软件寿命，而不是像现在这样疯狂的 30 年的寿命。

00:16:58 - Saron Yitbarek:

难道你对于你在计算机领域有如此巨大影响力的事情，一直反应那么平淡吗？

00:17:06 - Brian Fox:

我为自己写了 Bash 而感到骄傲，而且它让我意识到了我的价值，所以有时候我会做一些事情，诸如接受播客邀约谈论 shell 之类的事情。

00:17:14 - Saron Yitbarek:

非常感谢你。

00:17:15 - Brian Fox:

谢谢。但这不是存在于我日常生活中的东西。幸运的是，我只是一个默默无闻的人，对吧？的确，我的软件正运行在每家每户的计算机上，不过也确实没有人知道这一点，对吧？因此我保持了许多个人隐私，而这个 shell 以及某个住在圣芭芭拉的人编写了它这一事实正越来越广为人知，我开始在生活中越来越多地注意到它。人们有时候来看我演奏音乐，然后告诉我说：“你是写了 shell 的那个家伙。”我感觉有点儿像 Keanu Reeves。

00:17:54 - Saron Yitbarek:

很酷。所以你说过你不指望 Bash 出现在每台计算机上。你打算做的是什么呢？你对 Bash 有什么期望？

00:18:04 - Brian Fox:

一个有用的替代工具，成为 GNU 项目的一部分，并帮助创建这个自由的开源操作系统。我实际上以为一旦我们完成了该开源操作系统的创建，该系统上的软件就可以升级，并且我将有机会创建自己想要创建的那种 shell，以帮助人们在某种程度上促进计算机科学的发展。

00:18:35 - Brian Fox:

我最终意识到，Bash 被创建的原因实际上是与已经存在的 UNIX 世界向后兼容，并且这种势头使其保持了活力，这是另一个独一无二的地位，你的工具如此基础，几乎是一副不可或缺的螺母和螺栓。

00:19:01 - Saron Yitbarek:

确实是这样。

00:19:01 - Brian Fox:

知道我创造了世界上某种有价值的、别人仍然还在使用的东西，这真的是一种很棒的感觉。然后当我注意到这是怎么回事时，我意识到，更重要的是，“自由软件”和“开源”这些词存在于日常英语和全世界的日常语言之中了，而最初并不是这样的。这是我和理查德·斯托曼还有其他人所投入努力的产物。作为这一运动的一部分，我很幸运能这么早参与，但让我回过头来看时，也感到非常满意，我想：“哇，开源软件已经存在，而且我就是其中的一部分。”

00:19:50 - Saron Yitbarek:

Brian Fox 是 Bash shell 的创建者和 Opus Logica 的 CEO。

00:20:01 - Steve Bourne:

事实上，我确实听说过 Bash。

00:20:03 - Saron Yitbarek:

这位就是被 Brian Fox 的工作所替代的 Bourne shell 的创建者 Steve Bourne。我们想知道 Bourne 对 Fox 的工作有何看法。他是否将重生的 shell Bash 视为自己作品的开源复制品？我的意思是，他觉得 Bash 怎么样？

00:20:20 - Steve Bourne:

有一天，写了 Bash 的那个人在一次会议上找我，给了我一件 T 恤，前面印着“Bourne again”的字样。

00:20:26 - Saron Yitbarek:

那就是 Brian Fox。

00:20:29 - Steve Bourne:

那是一种友好的情绪，当时是：“好吧，希望您不介意，但我只是重写了您的 shell”，而我说：“听起来不错”，然后他给了我一件 T 恤。

00:20:38 - Saron Yitbarek:

如果我在编程领域学到了一件事，那就是每个人都喜欢意外之喜。事实证明，Stephen Bourne 认为 Bash 是他和其他人在贝尔实验室所做工作的必要扩展。一点儿都不为此苦恼。

00:20:52 - Steve Bourne:

曾经有一些人们想要，但是我没做的特性，例如变量替换和字符串管理，但是这些都被加入到了 Bash 中，现如今人们经常会用到。Bash 和原始 shell 之间的关系，我当时的印象是，它只是对语言的重新实现，并且随着时间的推移，它确实添加了功能，因此它确实取得了超越我所写作品的进步，当然是在字符串管理领域。我现在一直在用它。

00:21:21 - Saron Yitbarek:

Steve Bourne 是 Bourne shell 的创建者和 Rally Ventures 的 CTO。

00:21:32 - Saron Yitbarek:

自从 Bash 在前往圣芭芭拉的长途车程中被塞进 Brian Fox 的卡车以来，已经过去了许多年。2019 年，版本 5.0 被发布，就像 Fox 提到的那样，Bash 现在被内置进了 Linux 中、macOS 中，甚至微软 Windows 中。Bash 已经成了开源世界中脚本编写的基石。这是我们自动化的基础。

00:22:02 - Taz Brown:

随着组织规模的扩大，使用能够使我们更快完成工作的工具变得至关重要。它成为了必需品。

00:22:16 - Saron Yitbarek:

Taz Brown 是 Red Hat 的资深 Ansible 自动化顾问，因此她非常了解 Bash 的价值。

00:22:24 - Taz Brown:

我绝对认为人们在职业生涯初期就应该使用 Bash。与其使用 GUI 或者说是图形用户界面，不如将自己视为管理员或 DevOps 人员。

00:22:39 - Saron Yitbarek:

而这是因为作为一名 Bash 程序员，你将会掌握能让你晋升的核心价值。

00:22:45 - Taz Brown:

学习写脚本有一定的价值，因为这可以让你从自动化的角度，为程序的长期运行做打算。你可以看到脚本的运行方式，然后可以说：“好吧，我可以做到，我可以使这项任务自动化执行。”它开始使你成为与之前不一样的思想家和技术专家。

00:23:09 - Saron Yitbarek:

对于运维而言，自动化已经变得不可或缺。复杂的程序、应用和工具均由优雅的 Bash 代码实现。

00:23:21 - Taz Brown:

如果你愿意的话，你不必重复造轮子。你可以从 GitHub 库或是其他任何你存储这些特定文件的地方拉取它们。Bash 允许你这么做。Bash 允许你执行这些常见任务，并且可以从 10 台服务器扩展到 1000 台服务器。

00:23:42:

关于自动化的伟大之处在于，一旦你制定了计划，就可以以一种非常有效的方式执行。它允许你执行那些，无法手动执行的操作。

00:23:56 - Saron Yitbarek:

最近 Taz Brown 所从事开发的 Ansible® 这样的最新产品可以始终与 Bash 集成在一起，完成了工作。

00:24:04 - Taz Brown:

虽然时代在不停前进，但是我认为 Bash 永远都会是管理员会去选择使用的工具，特别是他们想要快速自动化的情况下。

00:24:14 - Saron Yitbarek:

最后，这一切的成功，都可以追溯到它是一个自由的、允许所有人加以改进的软件这件事上。它是 Brian Fox 提供给世界的，某种没有许可证和限制的东西。满足了人们一直的需求，所以是 Bash 成功的关键。实际上，他甚至已经不再主管 Bash 开发已经很长一段时间了。这位是 Chet Ramey，他维护了 Bash 数十年。

00:24:38 - Chet Ramey:

我想，Brian 在发布 1.05 版本后就已经决定了他想要继续去从事其他工作。他曾在自由软件基金会负责过其他任务，他想做除了 Bash 以外的事情，而我是 Bash 最活跃的贡献者。他和我一起开发了许多新功能。我们共同努力解决了许多 bug，因此当到了需要其他人接手时，我是最佳人选。

00:25:16 - Saron Yitbarek:

就像 Fox 一样，Ramey 也必须继续努力，因为 Bash 比任何一位维护者都重要。

00:25:25 - Chet Ramey:

我是从 23 岁开始贡献的，有点儿像是我和 Bash 共同成长。在某些时刻，我会需要征集一个团队。我需要征集那些愿意并且有能力投入时间推动 shell 发展向人们。

00:25:46 - Saron Yitbarek:

Bash，这款再次降生的 shell 明年将迎来 30 岁（LCTT 译注：Bash 发布于 1989 年，至本译文发表时，已经 31 岁了），并且没有衰老的迹象。Bash 乘着自由软件浪潮，然后是开源浪潮，直到传播至编程世界的每个角落。但曾经，它只是存储在 Brian Fox 汽车后备箱里磁带上的代码。它只是一些程序员，想要带给大家的 shell 语言。几乎偶然的，Brian Fox 在此过程中成为了一名伟大的代码英雄。

00:26:23:

顺便说一句，有些事情始终困扰着我， Brian Fox 驱车将所有 Bash 代码载到了 Santa Barbara。为什么要转移呢？我的意思是，他在某家科技公司找到了新工作吗？

00:26:34 - Brian Fox:

我想要继续我的音乐生涯，而我认为做到这一点的最佳去处就是气温总在 72 华氏度左右、天空没有乌云、海滩很美的地方。

00:26:45 - Saron Yitbarek:

很好，我更喜欢这个理由。

00:26:49:

现在让我们向 Wayne A. Lee 致敬，是他向我们建议了这一集标题《Bash Shell 中的英雄》。干得好， Wayne。

00:26:57:

在下一集中，我们对于自动化的兴趣，将提升到一个全新的高度，并且着眼于 AI 语言，特别是 John McCarthy 创造的 LISP。

00:27:11:

《代码英雄》是 Red Hat 的原创播客节目。如果你访问节目的网站 redhat.com/commandlineheroes，你将更深入了解到有关 Bash 或是我们本季所介绍的任何编程语言的故事。

00:27:28 - Saron Yitbarek:

我是 Saron Yitbarek。下期之前，坚持编程。

什么是 LCTT SIG 和 LCTT LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 LCRH SIG 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-3/heroes-in-a-bash-shell>

作者: Red Hat 选题: bestony 译者: JonnieWayy 校对: acyanbird, wxy

本文由 LCRH 原创编译，Linux中国 荣誉推出

《代码英雄》第三季（7）：与机器对话

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频博客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《代码英雄》系列播客《代码英雄》第三季（7）：与机器对话的音频脚本。

导读：创造一台会思考的机器在 20 世纪 50 年代似乎是科幻小说。但 John McCarthy 决定把它变成现实。他从一种叫做 LISP 的语言开始。Colin Garvey 介绍了 McCarthy 是如何创造出第一种人工智能语言的。Sam Williams 介绍了早期人们对思考机器的兴趣是如何从学术界蔓延到商业界的，以及在某些项目没有兑现承诺之后，漫长的人工智能寒冬是如何最终到来的。Ulrich Drepper 解释说，人工智能的梦想超出了当时硬件所能提供的范围。

但硬件每天都在变得更强大。Chris Nicholson 指出，今天的机器有足够的处理能力来处理人工智能的资源需求——以至于我们正处于人工智能研发的革命性复苏之中。最后，Rachel Thomas 确定了 LISP 之外的人工智能语言——证明了人工智能现在正准备做的不同类型的任务。

00:00:05 - 播音员：

卡斯帕罗夫非常慌乱。虽然已经拼尽全力，但是他意识到，他已经输了。

00:00:12 - Saron Yitbarek:

Garry Kasparov

1997 年的春天，国际象棋冠军加里·卡斯帕罗夫输给了计算机程序“深蓝”。对于机器智能来说，这是历史上的关键时刻。对于某些人来说，这是一次生存危机，是对人类智慧至上的挑战。但是，对于世界上的技术专家来说，这是另一种意味上的里程碑，是人工智能（AI）领域的一次飞跃。这表明，想要真正的智慧机器诞生，或许不是太过于疯狂的梦想。

00:00:47 - 播音员：

去赋予一台机器思考的能力仍然还是梦想，距离实现梦想还需要很多年的努力。此外，还需要做出不少惊人的突破。

00:00:56 - Saron Yitbarek:

我们是如何抵达这一步的？是什么导致了卡斯帕罗夫那场著名的失败？我们从哪里来，又该到哪里去？我是 Saron Yitbarek，这里是《代码英雄》，一档来自红帽的原创播客节目。在本季，我们都在探索编程语言的奥秘，揭开他们的历史和发展潜力。这一期让我们关注人工智能。你会使用什么语言，来让你的机器拥有自己的思想呢？我们的编程语言如何帮助我们抵达“深蓝时刻”，甚至是更远的地方？什么样的编程语言能最适配会思考的机器？这是我们半个多世纪以来一直在尝试解决的问题。因此，我们的故事要追溯到 1940 年代，那时，人工智能这个词汇还没有被创造出来。

00:01:59:

回顾二战结束是如何结束的，我们有这样一种感觉，那就是技术帮助同盟国赢得了战争。人们乐观地认为，科技可以成就一切。整整一代人都相信计算机的力量。在这一代人中诞生了人工智能的教父，一位出色的数学家——约翰·麦卡锡，他从根本上改变了我们与机器交谈的方式。我想知道他的思想的起源，以及范式的转变是如何发生的。我的意思是，对于初学者来说，当麦卡锡和他的同事们在想象智能机器的未来时，他们到底想了什么？

00:02:43 - Colin Garvey:

哇，这是一个好问题。

00:02:46 - Saron Yitbarek:

Rensselaer Polytechnic Institute

我和 Colin Garvey 聊了聊。他是伦斯勒理工学院科学与技术研究系的历史学家。以下是我们的一些聊天内容：

00:02:58 - Colin Garvey:

麦卡锡对提出 AI 究竟是什么这件事情非常谨慎。不过，举个例子来说，他可能在他最著名但并未实现的程序中描述过（这只是一个思路），人工智能是一个 advice taker

接受建议者。接受建议者这个说法由麦卡锡于 1960 年的名为

Programs with Common Sense

《具有常识的程序》的论文中提出的。在开始的时候，你觉得接受建议者可能是一个会学习的机器人，这是它被发明的本意，它算是一个可以接受建议的家用机器人。你说：“不，你这样做是错误的，要那样做。”它就会明白你的意思。

00:03:44 - Saron Yitbarek:

那很有帮助。

00:03:45 - Colin Garvey:

可能会有帮助。那时候接受建议者要实现的目标是，从办公室导航到机场。换句话说，将从办公室去机场这个建议，正式化成接受建议者机器人能够接受的程序以达成目的，得出从办公室前往机场的一系列程序。它把人类的语言正式化为一系列的合乎逻辑的陈述，即需要根据现在的情况进行合适的抉择，以达到把当前情况转变为理想情况的目标。他将这些想法伪代码化，这实际上是 LISP 的起源。然后在接下来的几年里，LISP 作为他想法的实现，或者说实现他想法的语言出现了。

00:04:39 - Saron Yitbarek:

麦卡锡的 LISP 语言改变了游戏规则。它不仅能帮助机器理解命令，还能帮助机器理解一些常识性的逻辑。麦卡锡发现，他可以编写条件表达式，即体现规则的代码，而不是仅仅是直白的命令。实际上，LISP 有一堆重大的编程突破，条件表达式、垃圾回收、递归等等。LISP 在代码和数据中都使用了列表。这是一个简单但具有深远的影响的更改。这是语言范式的转变，所有这些为麦卡锡本人称之为“人工智能”的整个领域打开了大门。想象一下，在你和机器说话的时候，并不需要把每一个细节都给编写好。你可以让那台机器自己来进行推断和推理。麦卡锡希望通过他的 LISP 语言，给机器一种智慧。好了，回到我和 Colin Garvey 的聊天。

00:05:41 - Colin Garvey:

LISP 是人们编写高级计算机语言的尝试的过程中，诞生的精华。

00:05:47 - Saron Yitbarek:

这很有意思。因为我的下一个问题是 LISP 和人工智能之间的关系。接受建议者第一次描述了 AI 的功能，它听起来也像是 LISP 的开始。能跟我聊聊更多 LISP 和人工智能之间的关系吗？

00:06:04 - Colin Garvey:

当然。这些早期的人工智能研究人员面临着一件事，那就是他们还在使用打孔卡进行编程。也许这些早期的编程人员，知道如何进行机器码级别的编程，这是非常困难且耗时的工作。因此，你需要一个更高级的编程语言，以便你可以用更加接近人类语言的方式编程。所以，像 LISP 这样，用列表的形式给出指令的语言——这也是其名称的来源，基于列表的处理（LCTT 译注，LISP 全称：“LISt Processing”）。从某种意义上来说，这样用列表给出的指令更加接近于人类语言，
formal logic 因为他们基本上都可以用通顺的逻辑去理解。因此，如果你可以读懂形式逻辑，那么基本上可以看明白 LISP 程序，或者任何基于逻辑的编程语言的程序，并对于代码中发生的事情有着更好的了解。

00:07:10 - Saron Yitbarek:

所以 LISP 真的帮了大忙……嗯，我的意思是它帮助我们将我们的想法给予人工智能，让我们能够朝着实现人工智能努力，有朝一日总能达成这个梦想。不过这也让我想知道在那个时期，“智能”到底意味着什么？所以，如果我们回到 50 年代，在那个时间点上，什么是“智能”？人们是如何定义它的？因为 LISP 最初为 IBM 704 开发的，它每次做的其实只是一件事。看起来并不是很“智能”。当时的人们是如何看待“智能”的呢？

00:07:43 - Colin Garvey:

有关“智能”的定义，这自然非常有争议的。就我个人，从一个普通大众的角度而言，他们的概念是非常狭窄的。但是在当时，广泛认可的定义是：能够执行被认为有智力的举动的能力。但持这个观点的这些人实际上是数学家、逻辑学家和计算机程序员。我的意思是，说句粗俗的话，下棋的能力被认为是智力的明确标志。这一代早期的 AI 人更喜欢回避这个问题，并说，“嗯，哲学家们还没有就什么是‘智力’达成一致，但如果我们将制造出一台能下棋的电脑，我想我们都同意这是‘智能’。”

00:08:40 - Saron Yitbarek:

不管怎样，这是一个起点。就像婴儿的第一步。

00:08:45 - Colin Garvey:

麦卡锡有一个梦想，让机器可以具有常识，并像人一样聪明。本质上来说，你可以和它们交谈。他开始创造一种程序语言来实现这个梦想。这就是 LISP。它模仿了人类思想的某些方面，尤其是逻辑思维能力，并使得利用计算机放大或扩展这些思想特征成为可能。所以从数学家的角度来看，他已经在实现智能机械的道路上走得很久远了。

00:09:32 - Saron Yitbarek:

Colin Garvey 是伦斯勒理工学院科学技术研究系的历史学家。

随着 LISP 的推进，发展人工智能的新机会开始出现，LISP 是这一新领域的标准语言。麦卡锡离开麻省理工学院去斯坦福大学工作后，麻省理工学院的其他工程师继续从事 LISP 工作。他们把麦卡锡的语言改进成一种叫做 MacLisp 的方言版本，甚至开发了 LISP 操作系统。看起来麦卡锡对人工智能未来的梦想正在成为现实。

David Levy

1968 年，麦卡锡甚至和苏格兰国际象棋大师大卫·利维打赌。麦卡锡打赌，10 年后，电脑将能够在国际象棋比赛中赢过利维。但是，如果事情进展得那么顺利，那就不是《代码英雄》的故事了。那时候麦卡锡还不知道，人工智能的冬天就要来了。

00:10:45 - Saron Yitbarek:

当麻省理工学院的那个基于 LISP 的操作系统，从学院衍生到公司后，情况开始发生变化。其中的一家公司 Symbolix 推出了 LISP 机器，甚至从 MIT 的 AI 实验室雇用了 14 名员工，而且更不切实际的研究不再是重点。

AI Lab

00:11:25 - Sam Williams:

这些公司原本都来自 AI 实验室。

00:11:31 - Saron Yitbarek:

这是记者 Sam Williams，他写了一本名为《Arguing AI》的书。

00:11:36 - Sam Williams:

Symbolix 可能是最突出的一个。它得到了最多的关注、最多的风险投资。约翰·麦卡锡和他的门徒们推动了所有的创新。我想，这只是一个案例。在 70 年代末期，人们普遍认为：“好吧，我们已经在学术界做了我们能够做到的事情，让我们在商界脱颖而出，让私营企业为我们投资。”

00:12:01 - Saron Yitbarek:

这在当时的环境下可以理解。那时 AI 似乎即将起飞，有利可图。

00:12:07 - Sam Williams:

那时候的算力便宜到中型企业也可以购买。因此，我认为许多公司看到了一个机会：“那里有一个富有潜力的市场，我们可以进入这个市场，我们可以把关于人工智能的产出卖给那些想要具有全球竞争力的企业。”他们让投资者为这个想法大肆投资。在 80 年代初期，大量的资金涌入该领域。

00:12:29 - Saron Yitbarek:

我们可以讨论一下为什么这个热潮后来干涸了。但是，我现在可以肯定的一点是，事情已经被大肆炒作作了。

00:12:39 - Sam Williams:

大量资金流入。人们认为市场非常成熟。

00:12:44 - Saron Yitbarek:

从一定来说，大量热钱涌入人工智能领域造成的繁荣，就像世纪之交时网络泡沫的前奏。

00:12:52 - Sam Williams:

这在技术发展史上并不少见，人们过度投资，但最终公司未能兑现他们的希望。所以，资金供应陷入困境。公司必须通过老式的方式赚钱——去拉客户，去生产东西。

00:13:08 - Saron Yitbarek:

所以那时，麦卡锡的梦想轰然崩溃，回归了现实。Williams 描述了随后的人工智能冬天。

00:13:16 - Sam Williams:

你会看到像 Symbolix 这样的公司市值上升到 1 亿美元，然后短短几年后就申请破产保护。在那时候有很多这样的人：“在 10 年内，我们将做到这一点，在 5 年内，你会看到这个。5 年后，你就会看到这个。”这些家伙是投机者，或者是想要从五角大楼或任何其他地方获取更多资金的学者。因为这种危险的炒作，过度炒作，让人工智能走上歧途。

00:13:48 - Saron Yitbarek:

人们对于人工智能的兴趣崩塌了。基础设施不到位，或许计算速度也不够高，这使一切努力都变成了徒然。但无论如何，这之中的指导思想是正确的，它只是生在了错误的时代。到了 80 年代末，人工智能研究日渐减少。

00:14:06 - Sam Williams:

我认为人们脱离人工智能这个领域的原因是，它失去了最初吸引他们进入其中时的冒险性。它曾处于技术领域的最前沿。在西海岸，很多人的精力被吸引到制造个人电脑上。从 40 年后的今天看来，试着让电脑进入美国每个家庭，比帮助财富 1000 强的企业做出更好的决策，是一个更引人注目的概念。

00:14:39 - Saron Yitbarek:

Sam Williams 是《Arguing AI》的作者。我们在这里描述的是一个计算机编程的世界，它被一分为二：一方面是约翰·麦卡锡这样的人，他们试图使用 LISP 等新型语言来探究智能的本质；另一方面，也无可避免地存在着另一群人，他们更专注于解

决实际问题，试图让那些愿意为其服务付款的企业轻松一些。现实情况是，世界上的那些约翰·麦卡锡们，那些抽象的梦想家们，并没有太多机会在人工智能的可能性上进行深入探索。他们所拥有的设备甚至还比不上我们今天使用的手机。他们的愿景对于手头的硬件现实来说，真的太大了。

00:15:36 - Ulrich Drepper:

当初，我们在轰轰作响的庞大计算机上工作，和数不清的人、数不清的工作组抢夺它有限的性能。

00:15:46 - Saron Yitbarek:

这是红帽的杰出工程师 **Ulrich Drepper**。他解释了早期的人工智能梦想之所以失败，只是因为没有成功所需的基本工具。

00:15:55 - Ulrich Drepper:

我们有 PDP 10, PDP 11 之类的计算机（LCTT 译注：Digital Equipment Corporation 公司的大型计算机产品），但每一台都可能有另外 20 个人在同时使用。即使没有这些人将计算力分流走，它们也是真正的低能力机器，其性能不足以在支持 LISP 系统的必要功能的同时，运行一套机能齐全的用户界面。

00:16:19 - Saron Yitbarek:

随着时间的推移，新开发的硬件对 LISP 也不再友好。

00:16:26 - Ulrich Drepper:

我们不仅走了台式机的道路，也走了服务器的道路。在我们今天已经习惯的处理器类型中，我们的实现的一切都是基于 8 位的处理器，而我们可能有了 16 位处理器、32 位处理器之类的。想要在这样的机器上高效实现 LISP 系统，就必须绕不少圈才能成功。

00:16:56 - Saron Yitbarek:

因此，计算世界的外部现实对这个脆弱的人工智能领域产生了意想不到的抑制作用。

00:17:06 - Ulrich Drepper:

和大厂商生产的 CPU 相比，定制硬件的成本极为高昂，其发展也极为缓慢。

00:17:21 - Saron Yitbarek:

通过创建自定义的硬件解决方案，继续使用 LISP 朝着 AI 梦想前进并非不可能。但是，事实是，令人兴奋的新硬件将很多人的注意力推向了其他工作领域。

00:17:35 - Ulrich Drepper:

你只需要等待处理器的下一个版本，那个普通的处理器获得的改进，比你自己开发硬件求解，比如说实现一个 LISP 系统，所能达到的效果和收获要大得多。最终，硬件变得越来越复杂，自行定制硬件成为了一件近乎于不可能的事。

00:18:03 - Saron Yitbarek:

那么，约翰·麦卡锡的远景梦想是慢慢死去，还是只在那漫长的人工智能冬天里沉睡，等待时机？在 21 世纪的第一个十年里，约翰·麦卡锡的语言 LISP 的使用者不断地流失。但是，在同一时间，发生了一件惊人的事。尽管 LISP 本身逐渐消逝，

与这门语言相连的梦想却被重新点燃了。人工智能的梦想重新活了起来。

00:18:34 - Chris Nicholson:

我想说的是，这次不一样。

00:18:36 - Saron Yitbarek:

这是 Chris Nicholson，开源 AI 软件公司 Skymind 的创始人兼 CEO。Chris 认为，在经历了大起大落之后，人工智能或许终于要进入一个可持续发展的长夏了。

00:18:52 - Chris Nicholson:

我们正处在人工智能的夏天，许多人都对 AI 可能的应用而激动不已。但是，这种兴奋是基于研究的真正进展，而这些进展是基于其他领域的进展。我们的硬件比当初好了太多太多，人们能用 GPU 替代 CPU，也能在大规模集群中使用这些 GPU 来训练巨大的机器学习模型，从而产生真正准确的预测。为了做到这一点，他们也使用了前所未有的巨大数据集，这样的数据量在 80 年代、90 年代或者之前都是闻所未闻、见所未见的。

00:19:30 - Saron Yitbarek:

对于 AI 语言，巨大的数据集的兴起是关键，因为它们代表了我们理解智能的方式 **deep learning** 的关键变化。AI 研究的重点转移到了深度学习方面，而深度学习似乎成为了 AI 研究的全部。

00:19:47 - Chris Nicholson:

LISP 是为了操纵符号而设计的。在 LISP 诞生的时代，AI 意味着符号推理。当时，人们认为人类的思维和智力本身就是符号的操作。现代的人工智能，也就是让我们如此着迷的人工智能，远不止于此。现在的 AI，就机器学习而言，是一种可以从非结构化数据中学习的大型数据处理机器。因此，你可以让机器学习算法一点一滴地拾起文本和图像，在这个过程中，随着时间的推移，这些算法将变得越来越智能。LISP 不是那样设计的。它被设计为仅在人工干预的基础上发展。然而，对于现在的机器学习算法而言，其中的学习部分意味着它们会根据所接触的数据进行自我调整。

00:20:40 - Saron Yitbarek:

这就是让机器以它们自己的方式实现智能化。但是，你知道，不仅仅是我们的机器在学习以强大的新方式思考；开源工具也让人们提高了自己的水平。克里斯认为，开源的开发方式对防止另一个人工智能冬天的到来有很大的帮助。

00:21:03 - Chris Nicholson:

人工智能冬天发生的一个原因是，当创意在放缓时，网络就会崩溃，资金就会枯竭。有了现在这些免费的开源工具，我们看到的不是创意的放缓，而是一种加速。创意得以被及时地测试，得以被迅速地分享。所以，这不仅仅是工具，而是预先训练好的机器学习模型，一个研究小组可以与另一个研究小组分享，或者与广大公众分享。而创意本身也被发表在 ARXIV 这样的平台上。ARXIV 是由 Cornell University 康奈尔大学主办的。所有这些因素汇合在一起，加快了创意流动的速度。对我来说，这是对人工智能冬天的一种保险。这些因素的交汇使得人工智能成为一个超级令人兴奋的领域，它们意味着人工智能的发展速度比炒作更快。

00:22:03 - Saron Yitbarek:

Chris Nicholson 是 Skymind 的创始人兼 CEO。

所以，开源能将人工智能领域从旧的专有模式所催生的寒冬中解冻出来。AI 的新时代伴随着 GPU 的巨大改进而到来。与之一并到来的还有算法上的突破，它帮助我们训练 AI 学习数据中的模式。然后，数据本身，海量的数据集正在为今天的 AI 提供动力。这一点，比任何事情都重要，这就是为什么新语言得以继续 LISP 未能完成的征途。我们向 Fast AI 的联合创始人 Rachel Thomas 询问，哪些语言适合下一代 AI。

00:22:50 - Rachel Thomas:

我们目前正在探索将 Swift 用于 TensorFlow 的可能性。Swift 是 Chris Lattner 开发的一种语言，我知道也有人在使用 Julia，他们正促进着 Julia 的开发与发展。但是，就目前而言，Python 在深度学习领域有着无可比拟的巨大优势。

00:23:03 - Saron Yitbarek:

这是因为 Python 非常适合处理大型数据集。

00:23:08 - Rachel Thomas:

我认为过去 20 年中发生的最重要的改变是，数据量的增长和人们对于数据的越发重视。我认为，数十年前有关 AI 的许多早期工作更多地围绕一种符号系统，即某种抽象符号系统。我要说的是，目前 AI 领域中的许多工作都围绕着数据展开，例如识别图像里的某样东西，或者辨认一条影评是褒还是贬。这些都是基于数据的问题。所以，我认为，Python 成为赢家 —— 至少是早期的赢家 —— 的原因之一，是因为 NumPy、SciPy 和 pandas 的生态系统，以及 Python 中所有数据处理用的库都足够完善。

00:24:02 - Saron Yitbarek:

在约翰·麦卡锡的团队还在麻省理工学院工作时，人工智能事业确实只有世界级的精英学者才能参与。但是自那时以来，语言和计算环境已经发生了巨大的发展，时至今日，每个人都能参与到 AI 的领域中来。我们在本季的第 1 期中听说过，Python 是许多初学者的第一语言。它使新手程序员能够加入 AI 的世界。

00:24:29 - Rachel Thomas:

我只想告诉大家，实际上你只需要一年的编码经验。你不需要诸如“鬼才”这样的称号，也不必具有真正的声望或权威。但是，我们需要来自各个背景的人。实际上，不同背景的人才能够为这个领域提供许多新鲜的创意，你的创意也是我们所需要的。

00:24:52 - Saron Yitbarek:

Rachel Thomas 是 Fast AI 的联合创始人，也是旧金山大学数据研究所的教授。我喜欢她为我们提供的信息。

AI 的故事是发现截然不同的人们之间的共同语言的故事。这是我们真正的任务，为充满 AI 的世界开发语言解决方案，开发能使所有人共同努力以实现下一个突破的解决方案。

00:25:22 - 播音员 1:

各位，你们看到了今天在这里创造的历史。AlphaGo 以伟大的风格赢得了比赛。

00:25:32 - 播音员 2:

是的，它甚至向我们展示了它的代码。

00:25:34 - 发言者 7:

甚至向我们展示了它的代码。我想要祝贺这个计划。

00:25:40 - Saron Yitbarek:

Garry Kasparov

2015 年，就在加里·卡斯帕罗夫在国际象棋比赛中输给深蓝的几十年后，围棋世界冠军李世石被谷歌的 AlphaGo 击败。深蓝显示出的智力威力大半依托于蛮力，而 AlphaGo 的胜利之所以让人感到惊讶，是因为它依靠神经网络和强化学习赢得了那场比赛。换句话说，AlphaGo 已经成功地朝着真正的智慧迈出了下一步。如果没有开源对人工智能的推动，这一切都不可能发生。新的语言正朝着麦卡锡的梦想而逐渐发展。我们的语言正越来越适合于最大限度地提高机器的智能。好消息是，在开源世界中，我们可以共同实现这份梦想，共同应对它所带来的挑战。

00:26:39 - Saron Yitbarek:

《代码英雄》是红帽的原创播客节目。如果你希望更深入的了解 LISP 或者人工智能，你可以访问节目的网站 redhat.com/commandlineheroes。在这里，你也能看到每一集的额外内容。下一期将是本季的最终集。在下一期中，我们将对一门重量级语言进行探究，并揭露与之有关的许多惊人事实；这门语言对本季中迄今为止讨论过的几乎所有语言都产生过影响。这是我们对 C 的深入研究。我是 Saron Yitbarek，下期之前，编程不止。

什么是 LCTT SIG 和 LCTT LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 LCRH SIG 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-3/talking-to-machines>

作者: Red Hat 选题: bestony 译者: bestony 校对: acyanbird, Northurland, wxy

本文由 LCRH 原创编译，Linux中国 荣誉推出

《代码英雄》第三季（8）：C 语言之巨变

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频播客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《[代码英雄](#)》系列播客《代码英雄》第三季（8）：C 语言之巨变的[音频脚本](#)。

导语：C 语言和 UNIX 是现代计算的根基。我们这一季介绍的许多语言都与 C 语言有关，或者至少受到 C 语言的影响。但是 UNIX 和 C 都只是 Bell Labs 贝尔实验室的几个开发人员作为秘密计划项目创造出来两个成果而已。

贝尔实验室是二十世纪中期的一个创新中心。Jon Gertner 将其描述为一个“创意工厂”。他们在二十世纪 60 年代最大的项目之一是帮助建立一个名为 time-sharing Multics 的分时操作系统。Joy Lisi Rankin 博士解释了当时关于分时系统的一些宣传，它被描述为有可能使计算成为一种公共服务。大型团队投入了数年的精力来构建 Multics —— 但这并不是他们所希望的成果。贝尔实验室在 1969 年正式远离了分时系统。但正如 Andrew Tanenbaum 所描述的那样，一个由英雄组成的小团队还是坚持了下来。C 语言和 UNIX 就是这样的结果。他们并不知道他们的工作会对技术的发展产生多大的影响。

00:00:00 - 发言人 1:

我们掀起了新一波的研究浪潮。我们的创造力正在延伸。

00:00:10 - 发言人 2:

噪音、噪音。

00:00:13 - 发言人 1:

这些人是贝尔电话实验室的设计工程师。

00:00:16 - Saron Yitbarek:

在上世纪 60 年代，坐落于新泽西州默里山的贝尔实验室，是科技革新的中心。在那里，我们的未来科技迈出了第一步。在那里，贝尔实验室发明了激光与晶体管，它还是信息论的摇篮。在 1968 年，贝尔实验室的四名程序员创造了一种极具开拓性的工具，它根本地改变了我们世界运行的方式，也标志着贝尔实验室的种种创新达到了新的高峰。

00:00:53:

我是 Saron Yitbarek，这里是《代码英雄》——一款来自红帽公司的原创播客。在一整季的节目中，我们追寻着编程语言世界中最具影响力的一些故事，现在，我们终于迎来了这一季的结尾。我认为，我们把最好的故事留到了最后。这个故事中的编程语言使本季中提到的其他编程语言成为了可能。在正好 50 年以前，C 语言在贝尔实验室被设计出来，这是一种非常基础的通用程序设计语言。它是如此的基础，以至于我们有时候都会忘记，C 语言的发明是多么意义深远的成就。

00:01:35:

为了得到事件的全貌，我们需要回到上世纪 60 年代，正好在 C 语言的诞生之前。那是一个一切似乎都有可能的时代。

00:01:46 - Jon Gertner:

在上世纪 60 年代，贝尔实验室简直是研究人员的世外桃源。在今天，已经很难找到与贝尔实验室相似的企业研发实验室了。

00:01:56 - Saron Yitbarek:

这是 Jon Gertner，他是
The Idea Factory: Bell Labs and the Great Age of American Innovation
《创 意 工 厂： 贝 尔 实 验 室 与 美 国 革 新 大 时 代》的作
者。我们采访了 Jon，让他大家解释当时贝尔实验室的情况。为什么在上世纪 60
年代，贝尔实验室能够成为他所说的“创意工厂”呢？

00:02:15 - Jon Gertner:

我想今天我们都相信——“竞争带来伟大的科技革新”，但是我不能确定这种观点的正确性，并且，其实，贝尔实验室的成就是在一定程度上是与这种观点相悖的。贝尔实验室的工程师和科学家们并没有特别大的压力，但是与此同时，由于贝尔实验室在诸多的研究实验室中的地位，它又确实可以雇佣到最优秀、最聪明的研究者，并给他们足够的时间去研究感兴趣的问题，同时提供大量的资助。如果你能证明你的研究项目符合这家电话公司（LCTT 译注：指 AT&T）的目标和理念，你就能够得到经费。

00:03:00 - Saron Yitbarek:

而 Jon 强调，虽然贝尔实验室是一个商业公司的产物，但它的价值观还是比较接近学术界的。通过让员工自行决定工作方式及内容，贝尔实验室实践了类似于开源社区的开放式领导原则。

00:03:19 - Jon Gertner:

这是诸如苹果、谷歌与微软这样的大公司出现前的时代。计算机的历史更多的聚焦
Homebrew Computer Club
在西海岸，聚焦于自 制 计 算 机 俱 乐 部 这样的组织，以及从其中发展出的企
业；但是我认为贝尔实验室和那些企业一样重要。贝尔实验室坐落于一个现在看
来几乎不可思议的地方：新泽西的郊区。但是，这里曾聚集了对科技突破做出了巨
大贡献的科学家、研究者和计算机工程师，他们的研究成果对全世界都有惊天动地
般的显著影响。

00:03:54 - Saron Yitbarek:

time-sharing
“分 时”就是这些惊天动地的项目之一。它的核心概念很简单，实现难度却极
大。他们能构建一个能够同时由成百上千的用户使用的操作系统吗？这样的发明将
会使当时的计算机领域为之震动。从 1964 年起，贝尔实验室的天才们，与
General Electric
通 用 电 气 和 麻 省 理 工 学 院 (MIT) 合 作，试 图 集 体 推 进 这 项 工 作 的 进 展。实
Project MAC
际 上，麻 省 理 工 学 院 在 一 年 前 已 经 有 了 相 关 的 研 究 项 目，即 MAC 计 划；但 是 现
在，所 有 这 些 顶 级 团 队 已 经 团 结 起 来，开 始 着 手 钻 研 大 型 主 机 分 时 操 作 系 统 的 构 建
方 式。

00:04:40:

John McCarthy
实际上早在 1959 年，约 翰 · 麦 卡 锡 就 提 出 了 分 时 操 作 系 统 的 概 念。请 收 听 我 们
第七集 的 节 目 获 知 更 多 细 节。他 设 想 了 一 种 可 以 在 多 个 用 户 之 间 切 换 其 “ 注意 力 ” 的
大 型 计 算 机。麦 卡 锡 认 定，这 样 的 一 种 机 器 有 潜 力 极 大 地 拓 展 现 有 的 计 算 机 文 化。
让 我 们 来 设 想 一 下 吧，如 果 一 千 名 用 户 能 够 同 时 在 一 台 机 器 上 工 作，你 就 完 成 了 对
整 个 编 程 与 计 算 机 世 界 的 民 主 化。现 在，这 支 群 星 荟 萃 的 团 队 准 备 着 手 将 麦 卡 锡 的
梦 想 变 成 现 实，并 为 他 们 想 象 中 的 操 作 系 统 起 了 一 个 名 字 —— Multics (LCTT 译
注：Multi- 前 缀 代 表 “多 人 的”)。

00:05:23:

Multics 团队为分时操作系统进行了多年的工作，但是该项目遇到了严重的资金困难，并且在十年之后，项目仍然看不到尽头。雪上加霜的是，项目的领导者 Bill Baker 是一个化学家，对贝尔实验室的计算机科学部门并不感兴趣。除此之外，我们仍然能找到一个新的问题——自尊心问题。

00:05:46 - Jon Gertner:

在贝尔实验室，人们每天习以为常的一件事情就是独自工作。我的意思是，贝尔实验室的人们有一种感觉：他们认为自己拥有一切他们所需要的人才和构思，并且拥有最先进的科技，当他们遇到值得解决的问题时，他们有能力去解决这样的问题。这种看法可能有一定合理性；但是 **Multics** 项目在贝尔实验室没有进展，在某种程度上也可能是因为像这样更加复杂的、合作性的工作在贝尔实验室的体系中运转不良，也不能让那里的高管们满意。

00:06:20 - Saron Yitbarek:

The Idea Factory

Jon Gertner 是《创意工厂》一书的作者，他刚刚发表的新书是
The Ice at the End of the World
《世界尽头的冰》。

00:06:32:

贝尔实验室于 1969 年四月正式宣布放弃 **Multics** 项目，但这是否就是故事的结尾呢？就贝尔实验室而言，分时操作系统 **Multics** 之梦已经破灭。但是这个梦真的结束了么？结果却并不是这样，并非所有贝尔实验室的研究人员都放弃了分时操作系统。四个顽固的拒不退让者在所有人都已放弃之后，继续坚持这一梦想，而那就是下一个故事了。

00:07:08:

说实话，有些梦想太美好了，这样的梦想是很难被人抛弃的。

00:07:12 - Joy Lisi Rankin:

这是一件大事业。

00:07:14 - Saron Yitbarek:

A People's History of Computing in the United States

这位是 Joy Lisi Rankin，她是《美国计算机人物史》一书的作者。Joy 将会和我聊聊分时操作系统的理想，以及为什么分时操作系统如此不可或缺。

00:07:27 - Joy Lisi Rankin:

开发分时操作系统是一件重要且极富雄心壮志的事，直到该项目开始之前，大部分上世纪 60 年代早期的分时系统在一台主机上都约有 40 至 50 个终端。因此，提升终端的数量是重要性很高的一件事，贝尔实验室的雄心很可能超出了大部分人的认知，这也是这个项目在实现最初目的的过程中碰到了不少困难的原因。但是，尽管如此，分时操作系统继续以不同的形态发展，并真正地走向繁荣；分时操作系统不仅仅在麻省理工学院得到发展，也走向了其他的地方。

00:08:09 - Saron Yitbarek:

是啊。那么，当我们谈起上世纪 60 年代，是谁在推动分时操作系统的需求？你提到了麻省理工学院、通用电气公司和贝尔实验室。那么我们的关注点是商业还是学术团体？谁才是真正的推动者？

00:08:23 - Joy Lisi Rankin:

scientific

我认为学术团体和商业团体共同推动了发展的进程，除此以外，一些科学团体也参与了这项事业，因为，正如我之前所说，分时操作系统是一种更加一对一、富有互动性的计算体验。但是从另一个角度来看，我也会说教育工作者也同样在推动这件事的发展。并且，从国家的层面上讲，当时也在进行关于创建全国性计算设施的对话。那么，基本上来说，所谓的全国性计算设施指的就是全国性的分时操作系统网络。真的，美国的思想领袖们也有这样的言论，他们认为这样的系统会是与供电、电话、供水一样的基础性服务。

00:09:08 - Saron Yitbarek:

哇哦。

00:09:08 - Joy Lisi Rankin:

对啊，我知道的！这确实很.....

00:09:09 - Saron Yitbarek:

那可真是一项大事业。

00:09:11 - Joy Lisi Rankin:

那是一项非常大的事业。

00:09:13 - Saron Yitbarek:

Joy 让我想起了一件事。尽管这一期节目主要聚焦于创造了 C 语言和 UNIX 操作系统的团队，但是在贝尔实验室之外，对分时操作系统的推动是一项运动，比任何一个团队都大。将计算机视为公共设施是一个非常有意义的想法，在这项事业中，有许多优秀的人物，可惜我们不能请他们来到这里，比如 Bob Albrecht 和 Martin Greenberger，以及其他的一些杰出人物。

00:09:37:

好的，在进行了一些预先说明之后，让我继续和 Joy 的对话吧。

00:09:41 - Joy Lisi Rankin:

那么，当约翰·麦卡锡在麻省理工大学的演讲上首次公开的谈论分时操作系统时，他明确的将其与电力进行了比较，并说：“这是一个让所有人都能使用计算机的方式，不仅仅是在学校里和商业活动中，还在每个人的家里。”回首过去，再阅读当时的文章与档案，许多人都确信，未来会出现一种能够被规范化管理的计算公共设施。因此，人们对这种全国性的分时基础设施充满了信心和支持。

00:10:22 - Saron Yitbarek:

非常有趣的一点是，在 1970 年，IBM 实际上已经退出了分时操作系统这一产业。即使是通用电气也出售了他们的大型主机部门，不过他们还仍然保留了一部分分时操作系统相关的业务。让我们简单地谈一谈这些吧，1970 年发生了什么？

00:10:39 - Joy Lisi Rankin:

我认为 1970 年已经一定程度上已经成为某种标志，这也许是人为假想的标志，这一年标志着公共计算设施与分时操作系统产业的失败。从某些角度上来说，这种观点是错误的。我认为在上世纪 60 年代末期，麻省理工和 Multics 项目明显在创建

一个支持上千个终端的分时操作系统上遇到了困难，而这是一个知名度极高、影响力很大的项目。在同一时期，数十个基于分时计算模型的商业项目在美国兴起并繁荣发展。这是一个科技泡沫。随后，对于分时操作系统的热情走向衰落。这不完全是因为通用电气出售了他们的计算主机业务，他们在上世纪 70 年代至 80 年代间一直保留着他们的分时计算业务，并且这一业务盈利状况良好。除此以外，当时的大学，例如麻省理工学院，也继续运行着他们的分时操作系统，直到上世纪 80 年代。

00:11:52:

因此，依我之见，“分时系统只是一个在上世纪 70 年代破碎的科技泡沫”的公共记忆之所以产生，一定程度上是因为人们过多地关注了 Multics 的困境。然而，事实上来说，如果我们回到过去，看一看当时的人们如何使用分时操作系统，以及分时操作系统赢得了多少利润，了解一下分时操作系统的成功，我们就会发现，其实上世纪 70 年代正是分时系统繁荣的年代。

00:12:17 - Saron Yitbarek:

现在让我们把眼光放回到贝尔实验室，由四位技术专家组成的小组想要创造他们自己的分时操作系统。他们是肯·汤普逊、丹尼斯·里奇、道格拉斯·麦克劳伊、J.F. Ossanna 约瑟夫·欧桑纳。不过他们并不想完成 Multics，他们想要越级跳过 Multics，制作一个不受过往拖累、功能更为强大的操作系统，他们称之为 UNIX（LCTT 译注：这个前缀代表“单一的”）。

00:12:39 - Joy Lisi Rankin:

我认为 Multics 是 UNIX 的灵感来源，其原因在于，许多在 Multics 上工作的程序员是如此享受分时操作系统在编程上的优点，以至于在 Multics 陷入困境时，他们便想要创造一个属于他们自己的分时环境。这些来自贝尔实验室的程序员，他们决定构建他们自己的编程框架与分时操作系统，这就是 UNIX 的起源。

00:13:20 - Saron Yitbarek:

A People's History of Computing in the United States
Joy Lisi Rankin 是《美国计算机人物史》一书的作者。

00:13:29:

Dennis Ritchie
丹尼斯·里奇将自己和其他三名同事称为一个团队。他们几个开发者想要作为一个紧密的四人小团体而工作，并且他们需要一种能够协调他们程序设计的硬件。但是贝尔实验室已经放弃了分时操作系统的梦想，即便它是一个学术研究的世外桃源，给已经放弃的项目拨款这件事也超出了他们的底线。因此他们拒绝了使用新硬件的提议。为此事购买新的硬件太过昂贵了，为什么要冒险呢？但研究员们还是坚持了下来。

00:14:05:

汤普逊和里奇要求得到一种类似 GE645 的机器，这是他们一直来进行 Multics 相关工作的型号。当他们得知无法得到经费时，他们刚刚在纸上潦草地写下一些关于文件系统的想法。最后，他们在《太空旅行》游戏中成功地实现了他们的一些想法，这个游戏运行在 PDP7 机型上，这种机型基本上与

Commodore 64 是同一个级别的。没有贝尔实验室的支持，他们的开发是缓慢的，至少开始是这样的，是一个字节、一个字节地前进的。这四人组复活了分时操作系统之梦，以他们称之为 **UNIX** 的形式。

00:14:47:

不过这里就是问题所在了：**UNIX** 操作系统是用汇编语言写成的。也就是说，他们用纸带向 **PDP7** 传输文件；你可以想象到，他们在缺乏理想的工具与上级的支持的情况下，努力构建这个开创性的操作系统时所遇到的困难。**UNIX** 已经获得生命，但还没有一种合适的编程语言能够让它歌唱。

00:15:23:

Ken Thompson

开发者们初次尝试为 **UNIX** 设计的语言称为 **B** 语言，由肯·汤普逊编写。

00:15:30 - Andy Tanenbaum:

Basic Combined Programming Language

这是 BCPL（基础综合编程语言）的一种衍生语言。

00:15:33 - Saron Yitbarek:

Andy Tanenbaum

这位是安德鲁·塔能鲍姆。他是阿姆斯特丹的一位计算机科学教授，也是许多书籍 **Computer Networks** 的作者，包括经典教材《计算机网络》。让我们听听他讲解汤普逊的 **B** 语言背后的故事。

00:15:48 - Saron Yitbarek:

所以说，**B** 语言是 BCPL 的一种衍生物？

00:15:51 - Andy Tanenbaum:

BCPL 源于一种构建 CPL 编译器的企图，这种语言编写的编译器确实能够起到作用，而 CPL 基于 ALGOL 60，ALGOL 60 语言又源于 ALGOL 58。ALGOL 58 则源于对 Fortran 进行改进的尝试。

00:16:01 - Saron Yitbarek:

搞明白了吗？现在的问题就是，**B** 语言有许多历史包袱。**B** 语言和它的这些前身相比，并没有太多的突破性改变，因此，**B** 语言不能完成让 **UNIX** 歌唱的挑战。**B** 语言中没有变量类型，对于初学者来说这是一个问题。除此以外，**B** 语言对应的汇编

threaded-code
代码仍然比 **B** 语言编译器的线程代码技术¹ 要快。

threaded-code

¹. 线程代码技术：一种通过把一系列调用指令转换成一完整的地址表，然后使用恰当的方式调用的技术。线程代码最初被用来减少代码的占用空间，提高代码密度。通俗地讲，这种技术有点类似于在 C 语言中把一系列的 switch-case 语句转化为用函数指针数组实现的形式。 ↩

00:16:31 - Andy Tanenbaum:

word

BCPL 和 B 语言只有一种数据类型，就是双字节类型

。双字节类型在基于双字节类型开发的 IBM 的 704 和 709、7090、7094 机型上效
byte

果不错，但是从 360 和其它所有的小型电脑开始的机型都是基于单字节类型的。在这种情况下，双字节类型就不是一个好主意了，它和现代计算机的匹配程度极其糟糕。因此，显然 B 语言无法解决现有的问题。

00:16:57 - Saron Yitbarek:

那么，该团队之前工作使用过的所有机器都是基于双字节类型的，但是在基于单字节对象的操作上，这种类型的机器就不够好用了。幸运的是，在这个时间点上，贝尔实验室的领导们又回来加入了 UNIX 项目，他们意识到了这个团队中正在产生令人激动的进展。他们资助了一台价值 65000 美元的 PDP-11，并且这台机器不是基于双字节类型的，而是面向单字节的。现在，装备上了 PDP-11，丹尼斯·里奇能够在处理编程语言的难题时更进一步。

00:17:36 - Andy Tanenbaum:

丹尼斯，以及在肯的少量帮助下，决定编写一种更加结构化新编程语言，包含其它

char int long

数据类型，比如说字符类型、整数类型和长整数类型等等。

00:17:47 - Saron Yitbarek:

Dennis Ritchie

因此，在 1971 年至 1973 年之间，丹尼斯·里奇一直在调整 B 语言。他增加了一种字符类型，并且构建了一个新的编译器，这样就不需要再使用线程代码技术了。

两年结束时，B 语言已经变成了一种崭新的语言，这就是 C 语言。

00:18:08:

C 语言是一种功能强大的语言，结合了高级功能和底层特性，能够让使用者直接进行操作系统编程。它的一切都是如此的恰到好处。C 语言从机器层次中进行了足够的抽象，以至于它也可以移植到其他的机型。它并非一种只能用来写应用的语言。它几乎是一种通用的编程工具，无论是在个人电脑还是超级计算机上都十分有效，而这一点极其重要，因为个人电脑革命当时已经近在眼前。

00:18:49:

团队的成员们在确定了 C 语言就是正确的道路之后，就立刻用它重写了 UNIX 内核和许多 UNIX 组件。因此，只要你想使用 UNIX，你就必须使用 C 语言。C 语言的成功与 UNIX 的成功紧密的结合在了一起。

00:19:06 - Andy Tanenbaum:

C 语言的流行，其实主要不是因为它是一门比 B 语言更优秀的语言——当然它确实比 B 语言优秀——而是因为，它是编写 UNIX 的语言，并且当 UNIX 广泛发行的时候，它自带了一个 C 语言编译器；甚至最后它还配备了两个 C 语言编译器。那么，UNIX 受到了广泛欢迎，每个使用它的人都有了 C 编译器，而且 UNIX 的一切都是由 C 语言写成的。而 C 语言是一种相当不错的语言，它又是与 UNIX 共同出现的，那为什么还要找其他的编程语言呢？

00:19:33 - Saron Yitbarek:

从这里开始，C 语言的价值开始显现。

00:19:35 - Andy Tanenbaum:

由于 **UNIX** 是用 **C** 语言写成的，并且带有一个 **C** 语言编译器，**C** 语言与 **UNIX** 从一开始就在一定程度上互相依赖，因此，它们也共同成长。在一个关键的时间点，**C** 语言在 **UNIX** 系统中已经足够流行时，像 **Steve Johnson** 这样的人开发了可移植的 **C** 语言编译器，这种编译器可以为其他型号的计算机产生机器码。最终，出现了面向其他操作系统的 **C** 语言编译器，人们开始用 **C** 语言编写各种各样的软件——从数据库系统到……天知道什么奇奇怪怪的玩意儿，因为 **C** 语言在各种环境下都可用，并且十分有效，效率很高。

00:20:07 - Saron Yitbarek:

因此，不久以后，人们也开始用 **C** 语言编写与 **UNIX** 无关的程序，因为这门语言的优点是显而易见的。**Andy** 将为我们讲述，**C** 语言如何完全接管了整个编程世界。

00:20:20 - Andy Tanenbaum:

我想说的是，**C** 语言在正确的时间出现在了正确的地点。在上世纪 70 年代，计算机的普及范围远比现在要小。普通人不会拥有计算机，并且对计算机一无所知，但是在大学和大企业所拥有的计算机中，有许多都使用了 **UNIX** 操作系统以及随之而来的 **C** 语言，也就是说，这些大学和大企业都在使用 **C** 语言。这些大学与大企业发布了大量的软件，也产生了大量的程序员。如果一个企业想招聘一名 **C** 程序员，发布招聘广告后一定会有人来应聘。如果想招聘一名 **B** 语言程序员，没人会来面试。

00:20:49 - Saron Yitbarek:

在 **C** 语言的世界中，有许多基础设施——软件、函数库、头文件等，这一切编程工具都构成了一个完美的闭环。

00:20:59 - Andy Tanenbaum:

因此，**C** 语言变得越来越流行。

00:21:02 - Saron Yitbarek:

现在，互联网的兴起导致了人们对 **C** 语言安全性的关注，这些问题在变种中得到了部分解决，比如 **C#**。有些时候我们会觉得，好像所有的兴奋点都在 **Python** 或 **Go** 等新语言上。但是我们希望能在播客中试图做的一件事就是让大家回忆起当下的我们与历史的紧密关联，而 **C** 语言的影响至今仍然是不可思议的。

00:21:29:

C 语言在现代最出名的产物就是 **UNIX** 的教子——**Linux**，而 **Linux** 的绝大部分都是用 **C** 编写的。就连 **Linux** 项目使用的标准编译器
GNU Compiler Collection
GCC（**GNU** 编译器集合），也是用 **C** 语言写成的。虽然这一点可能不太引人注意，但是今天所有聚集在 **Linux** 上的开源编程者，都与一种在半个世纪以前的语言相联系，而 **C** 语言的统治也在年复一年的增强。

00:22:02 - Andy Tanenbaum:

以上这些事情的结果就是世界上占支配地位的两种操作系统的诞生。一个是运行在 **Linux** 操作系统上的安卓，而 **Linux** 是重写 **UNIX** 操作系统的产物。而 **iOS**，本质上讲是一种 4.4 版的 **Berkeley UNIX**。因此，安卓和 **iOS** 从本质上说都是 **UNIX**。我怀疑几乎所有的服务器都是运行在 **UNIX** 或 **Linux** 的某个版本上的。这些服务器在幕后发挥着巨大的作用，并且任何运行 **UNIX** 的系统都源于 **C** 语言，为 **UNIX** 所编写的一切程序都使用了 **C** 语言。**C** 语言确实是无处不在的。

00:22:41 - Saron Yitbarek:

Andy Tanenbaum

安德鲁·塔能鲍姆是一名计算机科学教授，他是《计算机网络》一书的作者。说点有趣的题外话吧，他同时也是 MINIX，一个免费、开源版本的 UNIX 的作者，而 Linus Torvalds

MINIX 事实上也是林纳斯·托瓦兹开发 Linux 的灵感来源。当然，Andy 使用 C 语言编写 MINIX。

00:23:03 - Saron Yitbarek:

今天，C 语言存在于我们生活中的任何一个角落，从火星上的漫游车到台式电脑上的浏览器。它影响了许多我们在本季节目中提到的语言，例如 Go、Javascript 和 Perl。由于 C 语言与 UNIX 密不可分的联系，C 语言很可能是分布最广泛的编程语言。

00:23:28 - 发言人 7:

1998 年美国国家科学奖的获得者是——来自朗讯科技公司贝尔实验室的

Kenneth L. Thompson Dennis M. Ritchie

肯·汤普逊与丹尼斯·里奇的团队。

00:23:40 - Saron Yitbarek:

Ken Thompson

回望上世纪 60 年代，这四位贝尔实验室的员工——肯·汤普逊，

Dennis Ritchie Doug McIlroy J.F. Ossanna

丹尼斯·里奇，道格拉斯·麦克劳伊和约瑟夫·欧桑纳——他们那时还不得不向上级乞求关注和资助。但是在 1998 年，汤普逊和里奇就收到了美国国家科学奖，这是为了表彰他们在 C 语言和 UNIX 上的工作。他们也共享了一百万美元的图灵奖奖金。历史的眼光是公正的。

00:24:10:

在一整季的节目中，我们一直在追寻那些我们最喜爱的编程语言的发展沿革与魅力。无论它们像 C 语言一样搭上了操作系统发展的便车，又或者是像 Go 语言一样在一种新的基础架构上发展，有一件事是永恒不变的：编程语言有它们自己的生命。它们是活着的。它们出生，成长，走向成熟。有时，编程语言也会变老，走向消亡。我们越多的了解这些语言，我们越会发现编程语言是一股重要的力量，它们总是在不断地变化，以切合时代的需要。我们的职责就是意识到这些变化，并且加以回应。我们的语言一直都是构建我们想要的世界的最佳工具。

00:25:00:

以上就是我们所有第三季的《代码英雄》节目。我希望大家喜欢收听我们的节目。节目的第四季已经在制作中，即将推出，敬请期待。

00:25:13:

《代码英雄》是来自红帽公司的原创播客。

00:25:18:

如果你想深入了解 C 语言或者本季节目中我们提到的任何其他编程语言的故事，欢迎访问 redhat.com/commandlineheroes。我是 Saron Yitbarek，下期之前，编程不止。

什么是 LCTT SIG 和 LCTT LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 [LCRH SIG](#) 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-3/the-c-change>

作者: [Red Hat](#) 选题: [bestony](#) 译者: [QwQ2000](#) 校对: [Northurland, wxy](#)

本文由 [LCRH](#) 原创编译, [Linux中国](#) 荣誉推出

《代码英雄》第四季（1）：小型机——旧机器的灵魂

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频播客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《[代码英雄](#)》系列播客《代码英雄》第四季（1）：小型机——旧机器的灵魂的[音频脚本](#)。

minicomputer

导语：它们不适合放在你的口袋里。但在当时，小型机比之前的房间大小的大型机的尺寸小了一个数量级。它们为可以装进包里的 Personal Computer 个人电脑（PC），以及最终你口袋里的手机铺平了道路。

16位小型机改变了20世纪70年代的IT世界。他们让公司的每个工程师都有机会拥有自己的计算机。但这还不够，直到32位版本的到来。

Data General

Carl Alsing 和 Jim Guyer 讲述了他们在数据通用公司创造革命性的新32位计算机的工作。但他们如今被视作传奇的工作是在秘密中完成的。他们的机器代号为“Eagle”，其设计目的是为了与自己公司的另一个团队正在制造的机器竞争。这些工程师们回忆了为使项目继续进行而做的公司政治和阴谋，以及他们如何将限制转化为优势。Neal Firth 则讨论了如何在一个令人兴奋但要求很高的项目中的生活。在这个项目中，代码英雄们一起工作，只是因为他们想这样做，而不是期望获奖或成名。这三个人都讨论了这个故事

The Soul of a New Machine

如何在Tracy Kidder的非虚构工程经典《新机器的灵魄》中所成就的不朽。

00:00:03 - Saron Yitbarek:

那是1978年，小型机业界激战正酣。就在一年前，
Digital Equipment Corporation

数字设备公司（DEC）发布了其32位的VAX 11 780计算机。它比市面上的16位机器要强大得多。VAX的销售迅速地超越了那些步履缓慢的竞争

Data General 对手。其主要竞争对手数据通用公司迫切需要一台新机器来和VAX竞争。他们需要一台属于自己的32位计算机，而且要够快才行，但数据通用公司和DEC之间的竞争并不是唯一的战斗。数据通用公司内部还正酝酿一场地盘争夺战，而这两场战斗的战利品都是在令人难以置信的环境下创造令人难以置信的机器。一台13英寸的笔记本电脑大概有3磅重。如今，我们认为计算机的便携性及方便的尺寸是理所应当的，但是在20世纪70年代，大多数计算机仍然有着整个机房大小的大型机、重达数吨，而且价值数百万美金。而当硬件成本急剧下降后，开发更小、更快、更便宜的机器的竞争就开始了。小型机为工程师和科学家打开了拥有自己的终端机的大门。正是小型机引领我们到了今天的样子。

00:01:37:

在上一季的《代码英雄》中，我们深入探究了对软件开发至关重要的领域——编程语言的世界。诸如JavaScript、Python和C、Perl、COBOL以及Go之类的语言，我们审视了它们的历史、它们所解决的问题，以及它们是怎样随着时间推移而演变的。在这一季中，也就是第四季，我们将再一次深入探索，这一次是我们软件运行于其上的硬件。我们将为大家讲述七个特别的故事，讲述那些敢于改变硬件规则的人和团队。你桌上的笔记本电脑、口袋里的手机，如今你所拥有的每一件硬件设备，以及它们的前代产品，都是代码英雄们全身心投入的结果。他们的激情打造、他们的大胆举动，让我们的硬件成为现实，彻底改变了我们现如今的编程方式。

00:02:36:

Red Hat

我是Saron Yitbarek。这里是代码英雄，一档来自红帽的原创播客节目。

00:02:45:

在我们本季的首播中，将讲述一个工程团队竞相设计、调试和交付下一代计算机的故事。他们工作的故事变成了 1981 年 Tracy Kidder 获得了普利策奖的畅销书 **The Soul of a New Machine** 《新机器的灵 魂》的主题。在这本书中讲述了这一集中你将听到的众多嘉宾的故事。

00:03:07:

让我们说回到数据通用公司。该公司主席 **Ed de Castro** 制定了与 DEC 竞争的计划。他拆分了工程部门，将一支团队从其位于马萨诸塞州韦斯特伯勒的总部 **North Carolina** 迁移到了北卡罗来那州的新办公室。他们的任务是开发一款领先的 32 位的机器设计，以粉碎 VAX。他们将项目命名为 “**Fountainhead**”。**Ed de Castro** 为这支团队提供了几乎无限的支持和资源，他将 **Fountainhead** 视为他公司的救星。而剩下的几名工程师被留在了马萨诸塞州，他们觉得自己被严重轻视了。他们认为自己能够开发一个干掉 VAX 的杀手，可能会比 **Fountainhead** 项目所能开发的更好，但是 **Ed de Castro** 不会给他们机会。因此，这个小组的负责人 **Tom West** 决定自己动手。**Tom West** 是一名自学成才的计算机工程师，他负责数据通用公司的 **Eclipse** 部门。**Eclipse** 是数据通用公司最成功的 16 位小型机产品线。**Tom** 能造机器，也能卖货，而且他知道市场需求是什么。**Fountainhead** 项目成立以后，**Ed de Castro** 让剩下的工程师们继续致力于优化去年的产品线。而 **Tom** 和其他人都对此不以为然。

00:04:31 - Carl Alsing:

我们对此一点都不满意。我们中有些人离开去做其他工作，而另一些人则感到十分沮丧，担心自己的事业，而且感觉没意思。而且我们觉得另一组人肯定会失败。

00:04:46 - Saron Yitbarek:

Carl Alsing 是数据通用公司微编程小组的经理。他是 **Tom** 的副手。他们决定提出自己的项目计划。

00:04:56 - Carl Alsing:

这将是使用最新技术进行的全新设计，构建一个能够击败 DEC 的 VAX 的 32 位计算机。所以我们为此提出了一项建议，并去找了主席 **Ed de Castro**，他说：“不，没门儿。北卡罗来那州的小组在负责这项工作。你们不必操心。”因此，我们感到十分灰心，回去提出了另一个名为 **Victor** 的计划。我们研究了如何使去年的老产品更好的方法。我们在里面设置了一个小开关，即系统里的一个模式位，当你打开它时，它将使得计算机能够像现代 32 位小型机一样运行，尽管比较慢。然后我们向 **Ed de Castro** 提出了这个功能。最后他说：“你们在这里有个模式位。我不想再看到其他任何带有模式位的设计。北卡罗来那州那边正在负责新设计。”于是乎又一次，我们深感沮丧，我想就是在那会儿 **Tom West** 决定要做一些秘密的事情。

00:06:06 - Saron Yitbarek:

Tom 想出了两个故事。一个是讲给 **Ed de Castro** 的：他们将会对旧的 **Eclipse** 产品线进行加强，使它运行得更快一点，增加几个新按钮，并且换个新颜色。**Tom** 把它说成保险措施，以防万一北卡罗来那州那边出了什么问题。**Ed de Castro** 同意了。然后 **Tom** 给他的团队讲了另一个更棒的故事。

00:06:32 - Carl Alsing:

Tom West 向我们团队中的一些人提议，我们要开发一款真正优秀的现代机器，它对以前的机器完全兼容，并且采用我们所需的所有最新高科技的东西，有虚拟内存、32 位和纠错代码，以及所有这类东西：多任务、多处理、大量内存。“伙计们，我们将打造出最新的、能在市场上大杀四方的新机器。”

00:07:04 - Saron Yitbarek:

这款极具市场杀伤力的新机器的代号是“**Eagle**”。现如今，人们觉得我们的电脑中的内存是没有任何限制的，但是在那时候，当从 16 位转换到 32 位时，发生了重大的突破。突然之间，你的地址空间就从能够存储 65000 字节的信息变成了 40 多亿字节。随着这一增长，软件也可以处理更加大量的数据。这给计算机公司带来了两个基本的挑战：从 16 位过渡到 32 位，这是肯定的，但是他们还得让使用旧软件的老客户感到满意。因此，他们必须得开发一款能够让旧软件继续运行的机器，即一款向后兼容的 32 位计算机。**VAX** 尽其所能也没有找到第二个问题的完美解决方案，但是 Tom 坚信他的 **Eagle** 可以做到。

00:08:14:

Eagle 的总部位于韦斯特伯勒 14 号楼 AB 的地下室。Tom 指派 Carl 负责 **micro coding** **coder** **Micro Kids** 微 码。Carl 指派 Chuck Holland 来管理编码员，他们被称为微码小子。同时，Ed Rasala 将负责硬件。他委派了 Ken Holberger 负责管理团队，这个团队被 **Hardy Boys** 恰当地称为哈迪 男孩。（LCTT 译注：《哈迪男孩》是一部 1977 年的美国电视剧。）Tom 有一个盟友，就是工程副总裁 Carl Carman。Carman 也对 Ed de Castro 有意见，因为 Ed de Castro 拒绝让他来负责北卡罗来那州的小组。

00:08:51 - Carl Alsing:

Carl Carman 知道我们在干什么，却什么都没有对他的老板说。所以他给我们提供了资金，但我们需要保持较低的薪水，并且需要一些非常聪明的工程师。因此，我们决定去招收大学毕业生。这样做好处之一是他们不知道什么是做不到的。他们以为你无所不能。

00:09:15 - Saron Yitbarek:

那时 Jim Guyer 刚从大学毕业两年，在数据通用公司工作时被分派到了哈迪男孩。

00:09:21 - Jim Guyer:

北卡罗来那州那边正在开发的机器更多是高端计算，本质上几乎是大型机。而且，嗯，我的意思是，这在当时确实是投入到与 IBM 以及其他大型机公司的竞争中去的相当重要的一步。我们认为我们有优势，因为我们想做的事情并不那么雄心勃勃，而且我们真的、真的专注于一种简洁、干净、简单的实现方案，用最低的成本、最少的组件等等。

00:09:51 - Saron Yitbarek:

firmware 成本低廉，设计简单。他们意识到他们需要使用固 件来控制一切。与硬件相比，把越多的功能置于固件控制之下，所开发的机器就越便宜、越灵活。

00:10:03:

而且它们能够根据需求进行修改。当然，现代计算机都是以这种方式构建而成的，但在 1978 年，这种方法是全新的。

00:10:15 - Carl Alsing:

我们所做的设计非常简约。我们正在研究能够使事情简单明了的方法。因为我们知道，它不可能变成一个庞大而又昂贵的机器。它必须只是几块板子、几个电路，这是让使它快速发展的一个优势。设计一款安全的、无风险的产品，和设计一款用于制胜的产品是有区别的。而我们并不担心风险，我们在意的是取胜。我们希望它既快速又便宜，我们希望快速地开发它。因此，我们只用了三、四块板子，这是最低限度的硬件，我们通过固件来弥补这些。

00:11:06 - Saron Yitbarek:

Eagle 团队面临着很多严苛的限制。**VAX** 是这个星球上（当时）性能最高的 32 位计算机。**Eagle** 必须与之匹敌。但最重要的是，它还必须兼容他们的 16 位架构。要用比其他团队更少的金钱和时间来做到所有这一切，这使得 **Eagle** 感觉像是在赌博。但 Tom West 的团队全力以赴投身于其中。

00:11:32 - Jim Guyer:

有两套无时无刻都在运行着的系统，我们有两班工程师为之工作。我们所有人都必须全盘掌握。因此，我们不得不学会其他每一个人的岗位所要做的工作。这对我而言既具有挑战性又极其富有教育意义。但是我们大家都参与其中努力着，“要解决这个问题下一步该做什么？我们需要着眼于何处？”每个人都仔细钻研原理图和其他文档，试图找出办法，“看看这个信号，看看那台计算机的状态，看看微码正在执行的步骤顺序。它在正常运转吗？哦等等，它出错了。呃，为什么它会这样运行呢？”

00:12:13 - Carl Alsing:

这是件严肃的事情，这就是工作态度。小组里的工作很紧张。有时候人们会对于采用何种方式去做某件事情而发生争论。可能有一种方法会花费稍微多一点的钱，而另一种方法则更便宜，但是可能速度稍慢或效率稍低。为了达成共识，可能会开展激烈的讨论或会议。但我们还是做出了这些选择，然后我们协作努力。

00:12:44:**prototype**

我们没日没夜地工作，在原型上付出了很多时间。我们只有两个原型，两个团队都能在这两个原型上工作着实很重要。因此，晚班和白班都有人在工作，人们都很疲惫，但这让人感到非常兴奋——这是值得的。所以没有人过多地抱怨工作条件。

00:13:11 - Saron Yitbarek:

工作条件——据传当时 Tom West 为了让团队完成他所期望的东西，实行了一种 **mushroom management** 被称为“蘑菇管理”的方法。喂养它们然后看着它们成长。在狭窄而炎热的工作空间里，时间很漫长，日程安排也不切实际。Tom 本人被形容为神秘、冷酷、无情的人。有位工程师甚至称他为“黑暗王子”。Tom West 是否如此渴望取胜以至于剥削了他的团队吗？他是否为了得到完美的机器而牺牲了微码小子们和哈迪男孩们的福祉？

00:13:56 - Jim Guyer:

Tom 是个有趣的家伙。他对你期望很高，但不会给你很多指导。他希望你可以自己弄清楚需要做什么，而如果你不能做到的话，你就会被踢出团队。

00:14:10 - Saron Yitbarek:

指导来自于 Carl 或是 Ed，他们是 Jim 和团队其他成员每天都与之工作的部门经理。但这些年轻的工程师也为了取胜而参与其中，他们喜欢自己所获得的机会，愿意自己去搞清楚。

00:14:26 - Jim Guyer:

我个人获得了第一届微码小子通宵荣誉奖。我不知道是什么理由，也许我们都是能力超群、豪气冲天、无知无畏的年轻后浪。我们很自信，我们觉得自己相当聪明，可以解决问题，我们相互依靠，也许这就是自负。我乐在其中。我认为我们大部分人都乐在其中。

00:14:56 - Saron Yitbarek:

Carl 不同意“蘑菇管理”这一说法。他说情况恰恰相反。他们都确切地知道正在发生什么，以及预期是什么。反而是高层管理人员不知道。同时，Tom West 正在承受着来自多条战线的巨大压力，而这种压力也传递给了他的团队。

00:15:18 - Carl Alsing:

Tom 对这个项目的真实性质保持着低调。因此，他并没有对工程师们说很多，他保持着超然的态度，当然他也告诉他们，他们不能在团队之外或是家中讨论该项目。甚至不能使用 “Eagle”一词。因此，我们还传达了这个项目非常紧急，我们必须在一年之内完成，竞争已在市场中展开，如果我们要通过这个东西来占据市场之巅，我们必须现在就把它完成。因此他们承受着很大的压力，并且团队期望他们在夜晚和周末也参加工作，没有时间留给他们去和家人野餐或是做其他任何与工作不相关的事情。

00:16:06 - Saron Yitbarek:

我想知道在 14 号楼 AB 的战壕里奋战是怎样的感受。所以我邀请 Neal Firth 来到身边。他是微码小子中的一员，刚毕业就加入了团队。

00:16:20:

和 Tom West 共事的感觉如何？你和他有过很多互动吗？

00:16:24 - Neal Firth:

不一定。他是那种幽灵般的人物。我们能看到他在身边。他一般也不会不干预，以便我们能够领导自己、实现目标。我们正在做的事情是全新的，他不想把上一代处理器要做的工作强加给我们。

00:16:49 - Saron Yitbarek:

这听起来像是一个工作十分紧张的地方，在这里你真切地想不断前进并完成工作。你是怎样应对没有太多时间这一事实的？

00:16:57 - Neal Firth:

老实说，这并不是问题。想要时间充裕实际上并没有什么问题。我们可能会花费一些时间来实现结果。这需要家里人非常支持与理解，因为她们并不一定会立马同意。你可以将其等同于当时的一些硅谷人，或是像乔布斯、沃兹尼亚克之类的人，

我们投身其中并搞定它。我们确实不全是“住 在 同 一 间 公 寓”或“code-on-the-floor
在地板上写代码”那样的人，但具有其中的一些特征。

00:17:35 - Saron Yitbarek:

在那段时间里，是什么让你坚持了下来？为什么你这么有动力？

00:17:39 - Neal Firth:

坦率地说，就是在解决问题。我一直是一个喜欢思考并善于解决问题的人。事实上，团队里的大部分人都是这样的。我们所有人都有着类似的背景，并且我们都很享受解决问题这件事。就是，解决那些难题，找到一种前所未有的方式去做事。

00:18:01 - Saron Yitbarek:

那么在这个项目中，你最难忘的时刻是什么？

00:18:05 - Neal Firth:

当时，项目已经进行了相当长的时间，我们正在运行微码模拟器，它实际上是被提议当作生产模拟器来运行的，已经运行了 10 到 12 个小时了。突然，字母 E 出现在了控制台上，然后我们等了一会儿，又是一个字母，接着又是一个字母，然后我们突然意识到我们运行的是测试代码，是正在设计运行的诊断程序。因此，微码模拟器正在模拟运行这份微码的硬件，并且它开始打印字母，就好像它真的在运行一样。所以当它真正问世并运行时，可能比实际上要慢了十万倍，但这仍是我最难忘的时刻之一。

00:19:02 - Saron Yitbarek:

现在回过头想想，你觉得自己当时有被剥削吗？

00:19:07 - Neal Firth:

没有。我可以意识到正在发生着什么。我知道正在发生什么。所以，没有，我没觉得被剥削。实际上，这是我大学毕业时候的期望，否则我永远都不可能参与这么重大的项目，或是有机会在这样的一个项目中扮演那么重要的角色。

00:19:31 - Saron Yitbarek:

我想知道你如何看待发明的牺牲，因为如果你要考虑所有我们所创造的伟大事物，通常来说我们不得不放弃一些东西，是这样吗？必须舍弃一些东西来做出真正惊人的东西。这样的情况你遇到过吗？如果有的话，你不得不放弃的东西是什么呢？

00:19:48 - Neal Firth:

我不会说我放弃某些东西是有一个思想驱动的过程，我认为更重要的是，我更能适应自己正在做的事情以及所做的事对周围人影响。

00:20:03:

但我从来没把它看作是一种牺牲，而与我亲近的人们，他们生活在就是这样的一个世界里。我听说过一些可怕的故事，如果你愿意的话，在今天，在这里都是，你醒来，插上你的咖啡机，拿一些披萨或点心，然后你开始写代码，最后你在你的键盘上睡着。然后你在下一个早晨醒来，重复这个过程。

00:20:35:

我们当然远没有达到那种牺牲的程度，我仍然有妻子，我仍然有朋友，我们仍然在 nine-to-five

一起。这当然不是份朝九晚五的工作，但是它给我带来了许多个人层面与技术层面的成就，我能把这些和我的妻子、姐妹、父亲、母亲以及岳父分享，这些人可以欣赏这些。

00:20:59 - Saron Yitbarek:

是啊。那么，你认为使某件事情变得真正伟大的关键是什么呢？

00:21:06 - Neal Firth:

使某件事情变得真正伟大的关键——有趣的问题——我认为这取决于参与其中的人员，因为他们想要这样做，而不是因为他们对成就、财富或是名望的渴望。因为那些东西都转瞬即逝，而且永远无法使人满足。但是如果你要努力地去实现一个目标，而且你和一群人共同努力并去实现它，那么当你最终实现目标的时候，这确确实实是能令人心满意足的。

00:21:42 - Saron Yitbarek:

Neal Firth 是 Eagle 项目中微码小子中的一员。他目前是一家名为 VIZIM Worldwide 的软件公司的总裁。

00:21:57:

正如 Tracy Kidder 书中所记载的那样，Tom West 的超然和距离感是刻意为之的。这是他试图在日常交谈中保持头脑清醒，从而能使 Eagle 的目标保持原样。不过更重要的是，他想保护团队，将他们与周遭发生的政治与企业边缘化相隔绝。他也保护了微码小子们和哈迪男孩们不受先入为主的观念影响。

00:22:28:

1980 年，Eagle 完成了。比 Tom 所承诺的要晚了一年，但至少还是完成了，不像 Fountainhead。就像资深团队所预料的那样，Fountainhead 团队失败了，他们的项目遭到了搁置。这位是 Bill Foster，当时的软件开发总监，他谈到了 Fountainhead 的挣扎。

00:22:50 - Bill Foster:

我认为所犯下的最大错误在于没有对其设置任何限制。或多或少地，就是让我们去开发世界上最好的计算机。嗯，它应该在什么时候被完成？哦，我们对此确实没有个明确的期限。它应该花费多少成本？好吧，我们也不确定。我不得不让 Ed 失望了。他没有在程序员和工程师之间设置足够的界限。

00:23:15:

而且如果你让一群程序员和工程师放任自流，猜猜怎么着，他们将使事情变得复杂，设计出过分庞大的东西，以至于永远不能完成。

00:23:26 - Saron Yitbarek:

让我们回忆一下。从 Tom 和他的团队决定秘密开发 Eagle 开始，这已经进行了两年了。整个过程中，公司总裁并不知道正在发生了什么。当现在已经正式命名为 Eclipse MV/8000 的这款机器准备交付时，市场营销负责人去找了 Ed de Castro，为营销活动放行。Carl Alsing 将对此作出解释。

00:23:53 - Carl Alsing:

市场营销负责人说道：“好吧，我们已经准备好为 **Eagle** 进行推广了，我们需要数千美元。我们将在全球六个不同的城市举行新闻发布会。我们将进行一个巡演，去到很多城市，我们将拍摄一部影片来展示它，这将轰动一时。”

00:24:14 - Carl Alsing:

然后 **Ed de Castro** 说：“我不明白。你们为什么要那样做？”这只不过是在 **Eclipse** 那边的另一个包而已，表面工夫的工作而已。市场经理说：“不，这是一款全新的机器。这是一款 32 位的机器，有虚拟内存，具备兼容性。它将击败 **VAX**。所有你要的东西都在这里了。”

00:24:37 - Carl Alsing:

Ed de Castro 着实有点儿困惑。他以为我们在北卡罗来那州遭遇了失败，将成为事情的终结，但我们拯救了他。因此，是的，他邀请了我们所有人，我们举行了一次小型的午餐会。午餐会上有三明治和苏打水，他说：“嗯，你们做得很好，我很吃惊。我此前并没有意识到你们在做这个，但是我们会推广它，我知道将会安排一部影片、一些巡游，而且你们将成为这其中的一部分，所以，感谢你们，请吃你们的三明治吧。”

00:25:19 - Saron Yitbarek:

Computer World

现如今被命名为 **MV/8000** 的 **Eagle** 出现在了《计算机世界》杂志的封面上。推出期间，媒体的炒作使得这支原本秘密的、深居地下室的团队成员们变成了小名人。毕竟，他们拯救了数据通用公司。

00:25:38:

但好景不长。**Tom West** 再也无法让团队免受公司内部政治的影响。团队面对敌意毫无准备。公司里的其他人都嫉妒他们的成就，并对他们能在秘密项目中逃离如此之久感到震惊。

00:25:57:

不久后，一个新的工程副总裁取代了他们的盟友 **Carl Carman**。这个新来的人在第一台 **MV/8000** 被售出之前就拆分了 **Eagle** 小组，并把 **Tom** 分配到了数据通用公司的日本办事处。

00:26:13 - Jim Guyer:

我们认为我们开发出了花钱所能够买到的最好的 32 位超级小型机，我认为这对数据通用公司来说是一件很棒的事情，我认为它将能把 **DEC** 稍微踢开一点，而不是我们把世界从他们身边夺走。那时候的竞争太艰难了，在高科技领域很难成为赢家，但我认为我们已经做了一些有价值的事情。

00:26:42 - Saron Yitbarek:

当 **Eagle** 发行时，它确实拯救了数据通用公司，但在市场份额被 **DEC** 夺走三年后，该公司从未真正恢复过来，而行业却发展了。小型机不再是大势所趋。
microcomputer **personal computer**
微型计算机的竞争已经开始了，这为个人计算机革命铺平了道路。

00:27:04 - Carl Alsing:

数据通用公司继续开发了其他的版本，并在其他的型号上进行了改进，这样进行了一段时间，取得了一些成功。但是世事无常。市场发生了变化，他们自己转型成了一家软件公司，然后最终被其他人收购了。如今，我认为他们在马萨诸塞州 Hopkinton 霍普金顿某家公司的文件抽屉里。

00:27:36 - Saron Yitbarek:

一年后，Eagle 团队中的许多人离开了数据通用公司。有人感到精疲力竭。有些人准备好要去开发一些不一样的东西。一些人前往西部的硅谷，热衷于寻找下一个创意火花。无论如何，在一个不承认他们为拯救公司所做的一切的公司里，继续待下去似乎没有什么意义。1981 那一年 Tracy Kidder 的《新机器的灵魄》出版了。现如今，全世界都知道 Eagle 是如何构建起来了。

00:28:14 - Carl Alsing:

如果你问我新机器的灵魂是什么，我想我会说是他们所经历的人和事，他们所做出的牺牲，他们所做出的努力，他们对此所感到的兴奋，以及他们所希望得到的满足感。也许得到了，也许没有，但他们为之而努力。

00:28:35 - Jim Guyer:

从某种意义上说，这款机器是有点儿个性。但真正有个性的是这些有勇气的人。

00:28:47 - Saron Yitbarek:

在我们有关硬件的全新一季的下一集里，我们会将时光倒回大型机世界，讲述另一群叛逆员工的故事。他们制造的计算机催生了一门改变世界的编程语言。

00:29:04 - Saron Yitbarek:

Command Line Heroes Red Hat

《代码英雄》是红帽的一档原创播客节目。这一季，我们正在为你整理一些出色的研究，以便你可以更多地了解到我们正在谈论的硬件的历史。请前往 redhat.com/commandlineheroes 深入了解 Eagle 及其背后的团队。我是 Saron Yitbarek。下集之前，编码不止。

什么是 LCTT SIG 和 LCTT LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 LCRH SIG 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-4/minicomputers>

作者: Red Hat 选题: bestony 译者: JonnieWayy 校对: windgeek, wxy

本文由 LCRH 原创编译, Linux中国 荣誉推出

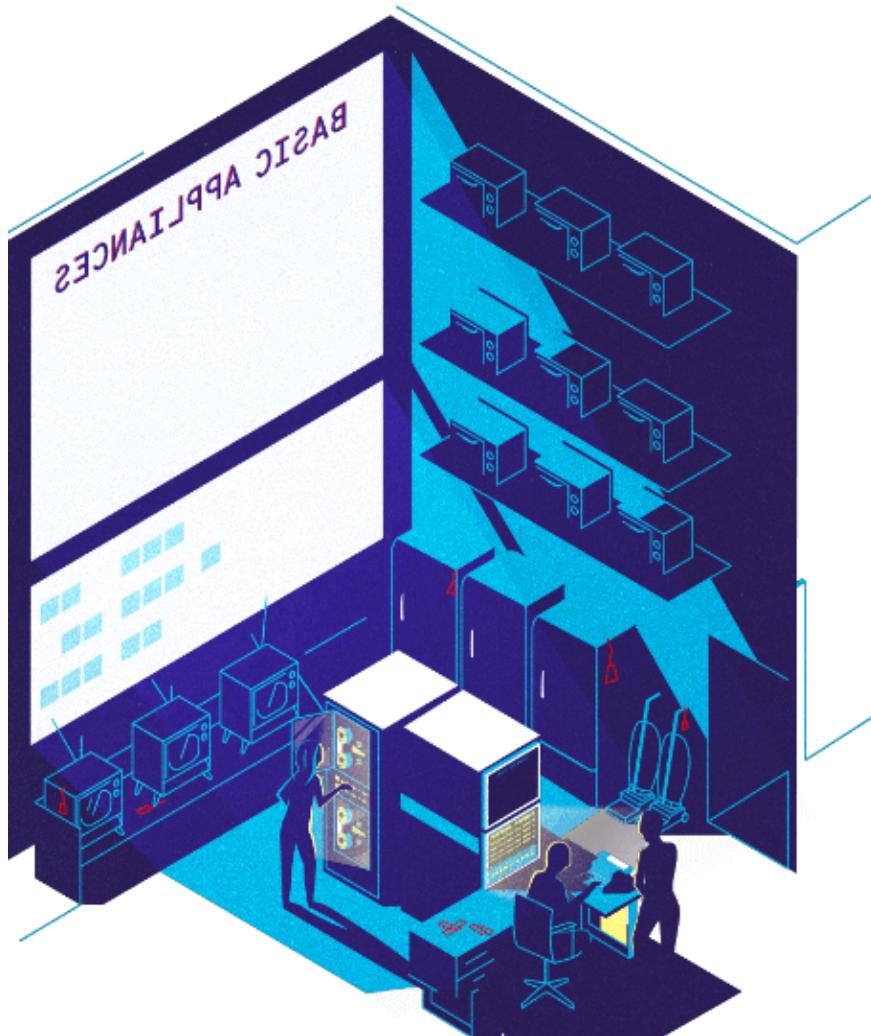
《代码英雄》第四季（2）：大型机 GE-225 和 BASIC 的诞生

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频播客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《[代码英雄](#)》系列播客《[代码英雄](#)》第四季（2）：大型机: GE-225 和 [BASIC 的诞生](#)的[音频](#)脚本。

General Electric

导语：二战后，计算机行业开始蓬勃发展。通用电气（GE）的CEO拒绝进入这个市场。但一小队反叛的员工不顾规则，秘密进取。他们创造了GE 225。这是工程上的一次巨大飞跃，它将计算从一个小众市场推向了主流，为今天的科技行业播下了种子。

mainframe

在创建通用的大型机之前，计算机通常是为了执行单一功能而制造的。

William Ocasio 回忆了 GE 的第一台专用计算机 ERMA 是如何帮助银行每天处理成千上万的交易的。John Joseph 讲述了 GE 的几名关键员工是如何蒙骗他们的 CEO 建立一个计算机部门的。Tomas Kellner 解释了他们的工作如何产生了一台革命性的机器：GE 225。而 Joy Lisi Rankin 则介绍了 Dartmouth College

time-sharing

达特茅斯学院的工程师们如何对 GE 225 用于分时计算，并利用它创建了 BASIC —— 使计算变得更加方便的重要里程碑。

00:00:05 - Saron Yitbarek:

让我们回到几十年前，回到 40 年代末、50 年代初，当时计算机行业有“白雪公主

General Electric

和七个小矮人”的说法。众所周知，“白雪公主”指的是 IBM，而通用电气公司便是“七个小矮人”之一。这是一家偶尔生产定制机器，但从没在公开市场售卖计算机的公司。

00:00:32:

通用计算机是 IBM 的地盘，而 IBM 是 GE 的第二大客户（第一是美国政府）。

IBM 经常采购 GE 生产的真空管、电机、变压器和开关等设备，GE 时任总裁

Ralph Cordiner 对此非常满意。所以，每当 GE 的部门主管将转向计算机业务的计划书提交到总裁办公室时，收到的答复都是封面上大写的橙色字母：RJC，Ralph Cordiner 一次又一次地拒绝他们。

00:01:19:

事实上，在 Cordiner 担任 GE 总裁兼首席执行官的 13 年时间里，GE 的态度从未改变。即使研究指出计算机是电子工业中增长最快的领域，但 Cordiner 总是用愤怒回应挑战他底线的员工们。然而，一小群叛逆的员工看到了制造大型机的机会，他们不想错失良机。然而他们不知道的是，这台大型机将拯救银行业，打开分时系统的大门，并孕育出新的编程语言。在上一季，我们听了 John Kemeny 和

Dartmouth College

Thomas Kurtz 在达特茅斯学院创造 BASIC 的故事，了解到因为 BASIC 是解释型语言，早期的计算机没有足够的资源运行它。像 BASIC 这么好的点子，正等待着像 GE-225 这样合适的设备出现，让它大放异彩。这一章，让我们一起揭开那台差点被扼杀在摇篮中的大型机，那鲜为人知的故事。这台房间大小的机器打开了 Steve Wozniak Bill Gates

新世界的大门，它鼓舞了像史蒂夫·沃兹尼亚克和比尔·盖茨这样有远见的代码英雄，鼓舞他们推动个人电脑革命，它的创造在今天仍然意义非凡。我是 Saron Command Line Heros

Yitbarek，欢迎收听代码英雄，Red Hat 的原创播客第四季：硬件设备。

00:03:05 - 话音 1:

Adams 先生觉得他的新支票账户很好用。与其他公司直接从
Elmvale National Bank
埃姆维尔国家银行兑现不同，用这个账号可以从自家银行兑现。与其他
Federal Reserve Bank
银行一样，自家银行把支票送到联邦储备银行去执行兑现。联邦储备
银行正是为处理每天来自数百家银行的数千张支票而设立的。

00:03:29 - Saron Yitbarek:

1947 年，支票开始流行，银行工作人员的时间耗费在无穷无尽的支票当中。战后
经济的蓬勃发展，更是让银行被铺天盖地的支票所淹没。为了腾出时间手工填写账
簿，他们被迫下午 2 点就早早关门，但即使这样，他们仍然赶不上进度。他们迫切
的希望，能有一台快速、强大的机器，让银行跟上商业发展的步伐。当时
Bank of America
美国银行每天要处理数百万张支票。下面有请
Kellogg School of Management at Northwestern University
西北大学凯洛格管理学院的教授 William Ocasio。

00:04:12 - Will Ocasio:

难以想象，银行如果没有电脑可用会是什么样子。银行本身就是繁重、文书密集型
的行业，美国银行分支机构又那么多。有这么多的信息，他们想要能够快速的传递
和处理这些信息。这对于这样一个大公司来说真的很重要。他们知道计算机才是未
来的方向。

00:04:39 - Saron Yitbarek:

Stanford Research Institute
所以，1950 美国银行与斯坦福研究院 (SRI) 签约，希望找到自动处
理这些支票的方法。SRI 花了五年时间制造了一台原型机，并将其命名为
electronic recording machine accounting
电子记录会计机简称 ERMA。ERMA 有超过 100 万英尺的
电线，8000 个真空管，重约 25 吨，每天可以处理 5 万笔交易。

00:05:11:

美国银行希望 ERMA 马上投入生产。于是向电子制造商们发出
request for proposal
招标请求 (RFP)，让它们竞标。当然，所有人都认为赢家将是行业巨
头，被称作“白雪公主”的 IBM。Baker 博士是通用电气电子部门的副总裁，他知道
他的老板 Cordner 不想涉足 IBM 的领域，也知道计算机是公司的禁区，但当听到
美国银行 RFP 的风声时，Baker 看到了其中的机会。他找到了 GE 位于
Palo Alto
帕洛阿尔托的微波实验室的经理 Barney Oldfield。在这个离 SRI 最近的地方向
University of California, Irvine
Oldfield 提出了一个建议。下面有请加州大学欧文分校的战略副教授
John Joseph。

00:06:09 - John Joseph:

我认为他是一位成功、进取的企业家，也是精明的经理和商人，他认为这是部门发
展的巨大机会。

00:06:27 - Saron Yitbarek:

Baker 和 Oldfield 成功说服了他们的老板 Cordiner，这是一台定制的生产控制系统，不是通用计算机，生产它不会让 IBM 不快。并且向他保证，GE 不会涉足计算机行业。

00:06:45 - John Joseph:

我觉得 Cordiner 最终屈服的原因是，他给他们提出了一个附加条件：仅此一份合同，不要继续深入商用机市场，只能参与这一次竞标。如果能做到，那你们去竞标吧。

00:07:08 - Saron Yitbarek:

尽管 Cordiner 不对竞标抱有任何希望，但还是让他们着手进行 RFP。就让他们发泄一下自己的创造力吧。随后 Oldfield 把他们的提案送到旧金山的银行办公室，等待着他们的答复。

00:07:26:

出人意料的事情发生了，IBM 忽然放弃竞标，更出乎意料的是，GE 的提案从所有制造商中脱颖而出。这个不被看好的提案赢得了百万美元的合同。美国银行董事会在 1956 年 4 月 9 日正式接受了这个方案。Baker 在没有经过他的老板 Cordiner 审核的情况下签下了这份价值 3100 万美元的合同，把不可能变成了可能。Oldfield 可以找个地方生产 ERMA 了，当然，他得先成立一个实际的计算机部门。

00:08:19 - John Joseph:

接下来轮到他们大显身手了。首先，他们确实成立了计算机部门，虽然听上去只是发一份声明就能搞定的事情。但是在这么大的公司内，把公司的资源和人力调动起来成立一个新部门，真的是一件很了不起的事情。

00:08:46 - Saron Yitbarek:

Barney Oldfield 成为计算机部门的主管。这个新部门很像 GE 的另一个制造定制机器的部门：军事系统部。挑战 IBM 之前，两个部门要先分出胜负。

00:09:06 - Saron Yitbarek:

新成立的部门想要低调行事，而 GE 的分权管理方式，刚好适合这样偷偷摸摸的搞事情。只要部门是盈利的，就不会有太多的监管。没人知道他们在做什么。

00:09:26 - John Joseph:

当时的情况是，想要在 GE 发展你的小领地，就必须走出去寻找发展的机会。公司没有任何这方面的计划，他是个有冲劲的人，看到了这个机会。他干劲十足，想引领公司走出这重要的一步。

00:09:59 - Saron Yitbarek:

更大的挑战还在后头，在哪建立工厂好呢？Palo Alto 的团队想要搬到 Stanford 的工业园区，无奈加州劳动法太严和税收太高，所以这不是个好点子。最终他们选择了 Phoenix。虽然 Phoenix 不是吸引资深工程师的最佳地点，但自有它的优势。

00:10:26 - John Joseph:

GE 的总部远在纽约，选择 Phoenix 能让他们远离 GE。在这里，他们可以避开高层的监督，野蛮生长。事关大把钞票，远离 Cordiner 才能获得更大的自主权。

00:10:55 - Saron Yitbarek:

Oldfield 在 Phoenix 组建了一支可靠的工程师团队。团队成员有：Bob Johnson、George Snively、Gene Evans 还有 George Jacobi 等人。还有 John Pivoden 负责硬件、Henry Harold 是逻辑设计师、Jay Levinthal 是系统架构师。在这个与世隔绝的桃园胜地，团队氛围非常融洽。他们不仅能担起秘密制造 ERMA 的大任，还能幽默的看待自己的工作。我们找到了他们表演过的一个小短剧，他们称之为“进步的前沿”，这个小短剧某种程度上展示了项目的进展。下面大家一起欣赏一段摘要：

00:11:39 - 话音 2:

好了，我们到凤凰城了。

00:11:41 - 话音 3:

你终于来了。欢迎来到计算机部门。

00:11:45 - 合音:

啥部门？

00:11:46 - 话音 3:

计算机部门。

00:11:47 - 话音 4:

计算机是啥？

00:11:49 - 话音 3:

哦，有点像是带有圣诞树灯的涡轮机，可以播放音乐。

00:11:53 - 话音 5:

是一种快速执行运算的机器。

00:11:56 - 话音 2:

我们没必要用它记账，对吧？

00:11:58 - 话音 3:

不，但我们得给 Van 一台让他玩，假装我们在用。

00:12:01 - 话音 4:

噢，计算机是啥？

00:12:08 - Saron Yitbarek:

对美国银行而言，计算机是可以一天处理 55000 笔交易的机器，这台机器需要对各种大小和状况的支票进行分拣和分发，需要更新客户帐户和余额，需要能识别支票。他们要的不是 1 台，而是 36 台计算机。

00:12:34:

项目初期，团队就决定 GE 版本的 ERMA 将使用晶体管来实现。在 50 年代，虽然晶体管比真空管更昂贵，但体积小，与逻辑板的连接器也更简单。原型机的每个真空管和触发器被两个晶体管所取代，使用额外的电阻和电容将它们连接在一起。将

ERMA 设计成软件编程设备，而不是硬件编程设备，是对原型机的另一个重大改变。这样可以简化机器设计，方便以后轻松修改。鉴于大多数 GE 的开发人员都从事硬件工作，他们得再聘请一名程序员。他们选择了一位几年前从纳粹德国逃出，以难民身份来到美国的男子。这名男子名叫 Joseph Weizenbaum。

00:13:34:

Weizenbaum 曾在一家名为 Bendix 的公司为 G-15 电脑编程。他甚至为它开发了一种名为 Intercom 100 的伪机器编程语言。尽管除了兑现支票之外，Weizenbaum 没有任何银行相关经验，但他还是领导小小的编程团队，开始编写支持晶体管硬件的软件。该团队还为所有的外围设备编程，包括支票排序器，还有被他们称做 magnetic ink character recognition MICR 阅读器的东西。MICR 的意思是“磁性墨水字符识别”。你知道支票底部的那行数字吗？那就是 MICR。这行数字由三部分组成，分别表示银行账户、路由号码和支票号码。直到今天，支票上仍有 Weizenbaum 和他的团队在凤凰城的杰作。

00:14:28:

值得一提的是。Weizenbaum 后来被认为是 AI（人工智能）的奠基人之一。1958 年 12 月 28 日，在 GE 赢得合同近三年后，美国银行圣何塞分行实装了第一台 ERMA 机器。虽然这台机器每天只能处理 100 笔交易，但这是朝着正确方向迈出的一步。接下来，他们要兑现每天处理 55000 笔交易的承诺。

00:15:04:

到了次年 3 月，该团队不仅兑现了 5.5 万交易的承诺，还增加了分拣机和打印机，使整个系统每天可以处理 200 万笔交易。美国银行兴奋不已。位于 Phoenix 的电脑部门交付了 32 台命名为 GE-100 的机器，还有更多的订单正在准备中，是时候庆祝了。

00:15:32 - Will Ocasio:

美国银行邀请 Cordiner 参加计算机的揭幕仪式。他们甚至邀请了曾在通用电气工作的罗纳德·里根作为他们的电视发言人，这太不起了。然而，Cordiner 来到揭幕式后，忽然想到：“等一下，这跟之前说的不一样”，然后他生气的开除了 Barney Oldfield。

00:16:00 - Saron Yitbarek:

额，Oldfield 就这么被开除了？Phoenix 的团队的明明超额完成了任务，完成了不可思议的壮举，他们的领袖得到的奖励却是被开除？更可气的是，Cordiner 重新任命了部门的负责人，Baker 博士的继任者 Harold Strickland 对电脑并不感兴趣。因为担心会惹 IBM 生气，Cordiner 给 Strickland 下了明确的指示，要他务必管好计算机团队。一个叫 Claire Lasher 的公司职员接替了 Oldfield 的工作。可 Cordiner 不知道的是，Lasher 骨子里也是个反叛者。

00:16:47:

通用电气总裁 Cordiner 想要远离商用计算机行业，他从一开始就不想进入这个行业。他想让通用电气尽快回到老样子。他允许生产已有的订单，但用他的话来说：“下不为例！”。后来，当 Cordiner 听到自己银行界的朋友向他称赞 GE-100 的创新时，他的态度发生了转变。所以好吧，他们可以继续在自己创造的沙盒中自由发挥，唯一的限制就是：“不要和 IBM 正面交锋”。

00:17:24:

Claire Lasher 的专长是市场营销。他看到了通用计算机市场的蓝海，借鉴了 Oldfield 的经验，制定了自己的计划。那就是两用计算机 200 系列。这个系列既是 The Big Look 定制机器，又是通用机器。Claire 将他的商业计划命名为 大 观，他为 Phoenix 团队招募了更多的工程师，其中包括一位曾在纽约从事 GE-312 和 412 过程控制计算机的人，他的名字叫 Arnold Spielberg，是新团队的领导。

00:18:05:

看完技术规格后，Arnold 说：“嘿，如果我们调整一下硬件设计方案，我们就能造出通用机器界的大杀器”。于是，Arnold 增加了读卡器、打印机和磁带处理程序等外围设备。仅用 5 个月就完成了原型机，这款新机器被称为 GE-225。有趣的是， Steven Spielberg

Arnold Spielberg 是史蒂文·斯皮尔伯格的父亲。为了了解更多关于 Spielberg，以及他在创造 GE-225 这台高层从来不想有的机器中所扮演的角色，我采访了通用电气的首席故事官 Tomas Kellner。

00:18:51 - Tomas Kellner:

Arnold 和他的同事 Chuck Prosper 是这台电脑的设计者，他们一起制造了它。有趣的是，和 GE 以前的计算机不同，GE-225 是商用计算机，它实际上有一个存储系统，并且能够处理数据的输入和输出。

00:19:15 - Saron Yitbarek:

GE-225 的哪些技术进步可以归功于 Arnold 呢？

00:19:21 - Tomas Kellner:

最有趣的是，这台电脑有自己的内存，能够记录和输出信息。这种存储器可以存储 20-bit word 8000 到 16000 个 20 位字。它还有一个辅助存储器，大约可以存储 32000 个 20 wire software 位字。他之前也用过一些硬件编程设备，它们真的很难使用。这些设备只能编程一次，然后就不能再修改了。GE-225 的进步之处在于，有了数据存储的能力。

00:19:58 - Saron Yitbarek:

GE-225 长什么样子？

00:20:02 - Tomas Kellner:

说实话，GE-225 看起来不怎么好看，它像一堆盒子。它有存储信息的磁带，有输入终端和输出终端。尽管它被称为小型计算机，但它能占满整个地下室。

00:20:26 - Saron Yitbarek:

它能完成哪些其他计算机无法完成的任务呢？

00:20:30 - Tomas Kellner:

GE-225 计算机的新特性是支持分时操作。支持多个远程终端访问计算机，多个用户能够同时在上面工作、写代码。据我所知，当时其他的计算机没这种能力。

00:20:58 - Saron Yitbarek:

那么谁买了 GE-225？GE-225 的目标客户是哪些人呢？

00:21:02 - Tomas Kelner:

通用电气公司内部肯定使用了这些计算机，但全国各地的一些银行也使用了它们，
Cleveland Browns

还有克利夫兰布朗队也用它们来管理季票销售。有人甚至用其中一台电脑预测了一场全州范围的选举，当然，预测的很准。人们似乎对这台机器非常着迷。

Cordiner 让团队在 18 个月内退出计算机行业，但是因为这台计算机的成功，他们搁置了这个计划。

00:21:43 - Saron Yitbarek:

重点是，GE-225 不仅是一个银行解决方案。还记得 BASIC 的创始人 *John Kemeny* 和 *Thomas Kurtz* 吗？BASIC 就是他们在 GE-225 上创造的。还有另一位代码英雄，他发现了 GE-225 上的编程漏洞。

00:22:03:

尽管那时他还在上高中。接下来让 *Tomas* 告诉你，GE-225 在 BASIC 的开发中起到了什么作用？

00:22:14 - Tomas Kelner:

GE-225 上开发了很多有趣的项目，达特茅斯学院的科学家们开发的 BASIC 语言便是其中之一。当时，他们想发明一种使程序员在不同的终端上同时工作的工具。这个工具，就是后来的 BASIC。关于 BASIC 另一件趣事是，通用电气以最快的速度从达特茅斯学院获得了 BASIC 的授权，并开始在内部使用它和预装在 GE-225
Steve Wozniak

上。史蒂夫·沃兹尼亚克就是通过 GE-225 接触到 BASIC。当他接触到了一个连接到 GE-225 计算机上运行 BASIC 的终端时，他瞬间就爱上了这个工具，实际上他的处女作就是在上面完成的。

00:23:13 - Saron Yitbarek:

那么当你和 *Arnold* 交谈时，他意识到他对计算机世界的影响了吗？

00:23:18 - Tomas Kelner:

对 *Arnold Spielberg* 的采访真是令人难以置信。那时，他已经 99 岁了。

00:23:23 - Saron Yitbarek:

哇哦。

00:23:24 - John Joseph:

他记得所有的事情，我们谈到了互联网，我清楚的记得，他说，在 1960 年代，畅想过计算机的未来，但是他从没想到，有一天人们可以通过家用电脑和智能手机，连接到互联网这个庞大的网络中。没想到人们可以随时随地获取信息，航空公司通过计算机控制飞机，控制机器工作。一方面，他对该领域的发展非常感兴趣。另一方面，他也非常谦虚地承认，该领域的发展超过了他在 20 世纪 60 年代的想像。

00:24:12 - Saron Yitbarek:

Steven Spielberg

你认为这一切对史蒂芬·斯皮尔伯格和他的事业有什么影响？

00:24:18 - Tomas Kelner:

史蒂芬还记得曾拜访过 GE 在他们的家乡亚利桑那州 Phoenix 建的工厂，他的父亲带他来工作的地方参观。他当时一头雾水，他爸试图向他解释电脑是什么，能做什么。他的原话是：“这一切对我来说都像是希腊语。”可见他当时是真的听不懂。我问 Arnold 对自己儿子的印象。他说：“我想让他学习工程学，培养他的技术热情，但他却只对电影感兴趣”。

00:25:00 - 话音 6:

随后银行业进入电子时代。今天，这种磁性油墨计算机系统让银行能够提供世界上最快、最有效的服务。保险公司、百货公司和公用事业公司也陆续开始使用类似的系统。事实上，所有文书工作都开始使用计算机。但谁知道呢，也许将来，会有更好的解决方案。

00:25:34 - Saron Yitbarek:

到 1962 年，GE-225 全面投产，并在推出一年后，迅速成为公司的重磅产品。它不仅为公司盈利，还在商界赢得了很高的声誉。在之后的几个月里，Cordiner 收到了来自全国各地的祝贺信息，他最终改变了他的想法，打算投身计算机行业。他终于正式认可了通用电气计算机部门。

00:26:13:

让我们回到与 Tomas 对话中提到的一些事情，我们在上一季的《C 语言之巨变》那一期中也有提到。当达特茅斯学院使用 GE-225 开发一个工具，让程序员可以在不同的终端上同时工作——换句话说，分时系统——GE 并没意识到分时的潜力。

00:26:38 - Joy Lisi Rankin:

达特茅斯学院使用 GE-225 和 GE DATANET 30 实现了分时系统，此前通用电气从未考虑使用这两种设备来实现这一功能。

00:26:53 - Saron Yitbarek:

Joy Lisi Rankin 是一名技术历史学家。

00:26:57 - Joy Lisi Rankin:

分时系统的关键在于电脑需要某种方式来停止自己的时钟。分时不是指人们在计算机上共享时间，而是指计算机共享自己的时间来处理多个计算请求。达特茅斯学院的师生们决定，使用 DATANET 30（这是一台 GE 的通信计算机）和 GE-225 共同开发分时系统。

00:27:32 - Saron Yitbarek:

因为大型机在 60 年代非常昂贵，批量运行是使用大型机最高效的方法。当时的人编写程序，然后做成打孔卡片以运行程序，他们将卡片交给操作员，然后他们就得等着它和其他程序一起被分批运行。有时要等几个小时，甚至几天。

00:27:58 - Joy Lisi Rankin:

在社交电脑和社交网络出现之前，在 Facebook 出现之前，分时系统、BASIC 还有达特茅斯学院和 GE，对开启个人电脑时代发挥了重要作用。通用电气从达特茅斯学院建立的分时系统中吸取经验，将其应用到自己的业务中，迅速建立起了全球分时服务行业。1970 年的某个时候，仅欧洲就有 10 万份时用户。在 20 世纪 70 年代到 80 年代，分时是他们的主要业务。

00:28:44 - Saron Yitbarek:

尽管 GE-225 和随后的 200 系列计算机取得了成功，通用电气公司还是在 1970 年将其大型机部门卖给了 Honeywell。但他们仍然决定保持分时共享业务，并在未来几年保持盈利。

00:29:08:

Ralph Cordiner 的故事讲完了，就像我们在上一期，数据通用公司发明小型机的故事中看到的，下一代伟大机器，它往往需要一个由顽固的、有远见的叛逆者和一些前瞻的执行官组成的团队来建造。人算不如天算，集思广益往往会有意想不到的收获。

00:29:41 - Saron Yitbarek:

下一期，我们将从 GE-225 结束的地方开始，来谈谈大型机如何启发新一代程序员展开个人计算机革命，谈一谈他们对我们的启发。《代码英雄》是红帽的原创播客。访问我们的网站 redhat.com/commandlineheroes 了解更多有关 GE-225 的资料。我是 Saron Yitbarek，下期之前，编码不止！

什么是 LCTT SIG 和 LCTT LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 LCRH SIG 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-4/mainframes>

作者: Red Hat 选题: bestony 译者: 2581543189 校对: wxy

本文由 LCRH 原创编译, Linux中国 荣誉推出

《代码英雄》第四季（3）：个人计算机 —— Altair 8800 和革命的曙光

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频播客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《代码英雄》系列播客《代码英雄》第四季（3）：个人计算机 —— Altair 8800 和革命的曙光的音频脚本。

导语：因为 Altair 8800，我们今天才能在大多数家庭中拥有计算机。它最初是为业余爱好者设计的，但是一些有远见的人看到了这个奇怪的小机器的巨大潜力，并努力使其他人也看到。他们创造的东西所带来的影响远远超出了任何人的想象。

Forrest Mims 告诉了我们他的联合创始人 Ed Roberts 是如何计划拯救他们陷入困境的电子公司的。他的想法是什么？一台为业余爱好者制造的微型计算机。那台计算机让比尔·盖茨和保罗·艾伦打了一个决定性的电话。Dan Sokol 和 Lee Felsenstein 回顾了 Altair 8800 在自制计算机俱乐部的揭幕，以及它如何激发了史蒂夫·沃兹尼亚克的 Apple I 的灵感。然后，我们在 John Markoff 那里听到了一个臭名昭著的软件抢劫案，该案为代码是否应该是专有的辩论创造了条件。最后，Limor Fried 回顾了这个故事如何继续影响今天的开源硬件运动。

00:00:04 - Saron Yitbarek:

1974 年 12 月一个严寒结霜的下午，两个年轻人走在哈佛广场上，他们可能正在谈论着他们最感兴趣的计算机话题。

00:00:19:

Popular Electronics

当他们经过报摊，看到了《大众电子》杂志从其他杂志中露出的蓝色字体，他们停下来往下看了一下，杂志封面上是一个金属盒的照片，在它的正面有十几个开关和指示灯。标题上写着：“世界上第一台小型计算机套件，Altair 8800。”这个盒子看上去不太像样，裸露着金属，就像是给业余爱好者和修理工们准备的。但对这两个人来说却不是这样，更像是他们一直在等待的机器，因为他们正好有适合这种新硬件的完美软件。同时，他们也有一些忐忑，如果别人也有这种想法并已经开始实施的话，那该怎么办呢？必须尽快行动起来了。这两位代码英雄是谁呢？

Bill Gates Paul Allen

比尔·盖茨和保罗·艾伦。此时他们并不知道，Altair 8800 将会是打开个人计算机革命大门的机器，它将永远的改变我们的生活。这台设备还做到了另一件事，在一个神秘小偷的帮助下，它将引发自由软件和专有软件之间的争论，我们稍后会讲到。

00:01:50:

在硬件这一季的第一集，我们了解了 Eagle 这样的分时小型机。在第二集，我们了解了 GE-225 大型机。但它们仍然受制于自身的尺寸、价格以及处理能力。而这一 micro computer 集讲的是，缩小到微型计算机的所有东西。这一切，都始于邮寄给业余爱好者的 DIY 套件，就像是《大众电子》的那张划时代的封面里面的一样。

00:02:23:

这些简单的套件，激发出了一种革命性的想法：计算机可以放在你的家里。这台计算机是属于你的，你可以用来做实验。一个全新的、面向大众的硬件产品 —— personal computer 个人计算机（PC）——诞生了。我是 Saron Yitbarek，这里是《代码英雄》，一款红帽公司的原创播客。

00:02:51:

让我们回到上世纪 60 年代末，在新墨西哥州的沙漠里，Altair 8800 诞生了。一个名叫 Ed Roberts 的人与人合伙创立了一家小型电子零件公司 MITS（意即“Micro Instrumentation and Telemetry Systems”微型仪器和遥测系统）。通过爱好者杂志，他们可以将这些小玩意卖给痴迷于无线电遥控飞机和火箭模型的新市场。

00:03:21:

到 1971 年，Ed 独立经营着公司。他决定将重心转向电子计算器，这在当时是一个全新的市场。MITS 准备提供第一台爱好者计算器，但是 Ed 失算了。这位是他最初的联合创始人，Forrest Mims。

00:03:42 - Forrest Mims:

像索尼、夏普和德州仪器这样的公司，他们正在制造专业的袖珍计算器，并以低于 Ed 的价格出售。这是一场灾难，Ed 濒临破产，并且不知道该怎么办。

00:03:57:

有一天，他了解到英特尔公司开发了一种新型的微处理器，并在 Electronics Magazine 《电子杂志》上刊登了广告。我仍然记得那则广告，那款微处理器叫做 Intel 8080。

00:04:09 - Saron Yitbarek:

Ed 大量购买了英特尔微处理器，因为当时没有人购买它。他围绕这个微处理器设计了一台完整的计算机。

00:04:23 - Forrest Mims:

有一天晚上，他给我打电话说：“我有一个新玩意儿想让你看看。”于是我骑自行车去了 MITS。桌上有一个蓝色的盒子。他说，“看这个。”我说，“这是什么？”他说，“这是一台微型计算机。”我说，“你在开玩笑吧。”他说，“不，这是一台微型计算机，《大众电子》杂志已经认可了，并且想要刊登一篇关于它的文章。”

00:04:43 - Saron Yitbarek:

Ed 的目标是创造一个非常基本的计算机套件，同样提供给那些模型飞机和火箭的爱好者市场。他通过《大众电子》杂志来邮购销售这个套件。当你收到这个套件时，会获得一个装满金属零件的袋子，里面有一个装有最重要的 Intel 8080 微处理器芯片的特殊管子。Ed 把整个套件的价格定在 400 美元左右。

00:05:14 - Forrest Mims:

他在考虑一个问题，“你觉得能卖出多少台？”他问道。基于之前我们通过《大众电子》杂志销售东西的经验，我说，“好吧，Ed，顶天了也就几百台吧。”我这么说的时候，他看起来很难过。直到今天我都为此感到内疚。在《大众电子》杂志刊登了这个套件之后，他的小楼前的停车场里挤满了汽车。最后竟然卖了 5000 台这样的计算机。它被命名为 Altair 8800。当时 Ed 不知道该怎么称呼它，这个名字还是《大众电子》杂志的工作人员想出来的。

00:05:50 - Saron Yitbarek:

Altair 8800 是 Ed Roberts 为了拯救他的公司而做的拼死一搏，这是他做过的最好的决定。他做了一件真正有意义的事情，但他当时并没有意识到。通过将 Altair 以一个合适的价格投放到市场，他让自己的机器展现给了比铁杆电子爱好者更大的受

众群体面前。他的 Altair 开拓了一个全新的市场——那些从未想过能拥有自己计算机的消费者群体。

00:06:28:

更重要的是，他们可以修理自己的计算机。这是一个大时代的开端，但它还缺少一个部分，软件。这个硬件需要正确的软件才能活起来。

00:06:51:

Massachusetts

回到马萨诸塞州的剑桥，比尔·盖茨和保罗·艾伦刚刚在最新一期《大众电子》杂志的封面上看到了 Altair 8800。在他们走过哈佛广场的那段决定性路程之前，保罗一直在向比尔抱怨那些 Intel 8080 的新芯片，他在寻找使用这些芯片创建软件的方法。

00:07:16:

比尔和保罗使用 BASIC 编程。正如我们在上一集中知道的，如果没有 GE-225 主机，BASIC 永远不会诞生并流行起来。对于比尔和保罗来说，BASIC 的易用性使得它成为了理想的语言，可以提供给那些受限于内存和处理能力的硬件，比如 8080。

00:07:38:

当他们看到杂志封面上封装着 8080 芯片的 Altair 时，他们意识到可以用 BASIC 编写软件来支撑这个机器。他们很快联系了 MITS 的 Ed Roberts。Forrest Mims 还记得那个电话。

00:07:56 - Forrest Mims:

保罗说，“我们得给这个公司打个电话，告诉他们我们已经有 BASIC 了。”盖茨同意了，但他不想亲自打这个电话，因为他的声音实在太年轻了，而且他看起来也像个青少年。所以保罗·艾伦打电话给 Ed：“我们已经为你的 Altair 准备好了 BASIC。”Ed 说，“每个人都告诉我他们已经搞好了 BASIC。如果你弄好了它，就把它送过来，合适的话，我会考虑的。”

00:08:17:

他们并没有 BASIC。他们花了一个月的时间在麻省理工学院或哈佛大学都不知情的情况下借用了计算机时间，为从没有见过的 Altair 开发了 BASIC 软件。

00:08:27 - Saron Yitbarek:

比尔和保罗知道他们有能力为 Altair 编写代码。但实际上，他们还没有开始编写。所以他们日以继夜地为这个机器改写 BASIC。使用公布的规范，他们在 DEC PDP-10 主机上创建了一个仿真器，以此来仿真 Altair，然后开发了一个 BASIC 解释器。

00:08:53:

没有 BASIC 解释器的 Altair 8800 本质上就是一个带有开关和指示灯的金属盒子，并没有什么用。然而随着 BASIC 解释器的加入，这台机器突然有了全新的未来。

00:09:10:

仅仅几个星期后，也就是 1975 年 3 月，代码就准备好了。保罗飞往 Albuquerque 阿尔伯克基，准备亲手将演示程序交给 Ed Roberts。前一天晚上比尔一夜没睡，以确保代码没有任何的错误。他把最终的代码打在纸带上，并在航班起飞前交给保罗。在三万英尺的高空，保罗突然意识到，他们忘了一件事。

00:09:39 - Forrest Mims:

Coder bootstrap logger

他意识到，他们没有开发出编码器，或者说引导记录器，来帮助计算机读取程序。他在飞机上写了那个代码。保罗·艾伦就是这么才华横溢。

00:09:53 - Saron Yitbarek:

现在他已经准备好了。在第二天进行演示的时候，保罗将首次在一台真正的 Altair 上测试他们的代码。1994 年比尔·盖茨在一段旧影片中，谈到保罗·艾伦在 MITS 装上纸带时所发生的事。

00:10:13 - 比尔·盖茨:

第一次，由于某些原因，代码并没有工作。当第二次加载它时，它就顺利的工作了。然而这只是仿真器，速度非常的慢，需要大量的指令才能完成一条指令。因此，实际上，即使是一个非常小的微处理器，真实的机器也比我们的 PDP-10 仿真器要快，大约快五倍。

00:10:32:

所以，对保罗来说，当它最终出现并显示出“READY”提示符时，保罗输入了一个程序，“打印二加二”，它成功了。然后他让它打印出类似于平方和求和之类的结果。他和这家公司的负责人 Ed Roberts 一起坐在那里都惊呆了。我的意思是，保罗惊讶于我们的那部分能够工作，而 Ed 则惊讶于他的硬件能够工作。

00:10:55 - Saron Yitbarek:

保罗·艾伦和比尔·盖茨的 BASIC 解释器在 Altair 上工作得非常棒，这是一个惊人的成就。

00:11:02:

Ed Roberts 对此印象非常的深刻，以至于……

00:11:08 - Forrest Mims:

他当场聘请了保罗·艾伦担任他的软件开发副总裁。

00:11:13 - Saron Yitbarek:

保罗·艾伦在那之后很快就搬到了新墨西哥州，开始了他的新工作。至于比尔，他回到了哈佛，毕竟他还是个学生。

00:11:23 - Forrest Mims:

但是保罗·艾伦说服了盖茨在 1975 年的夏天回来，并开始用 BASIC 工作。他们一起开了一家公司，叫做 Micro-soft，带一个连字符。后来他们去掉了连字符。

00:11:36 - Saron Yitbarek:

MITS 成为了比尔和保罗的第一个客户，他们授权了他们的 **BASIC** 解释器给 **MITS**，并同意随机器分发他们的代码。他们称这套软件为 **Altair BASIC**，它成为了与个人计算机捆绑的第一款软件。现在他们只需要卖掉它就好了。

00:11:57 - Forrest Mims:

Ed 做了什么呢？嗯，他们买了一辆大型房车，把它做成一个移动销售设备，一个销售计算机的移动办公室。他们把它开到全国各地，在不同的城市停下来，举行演示，这吸引了大量的人。

00:12:12 - Saron Yitbarek:

它被称为“移动 **MITS**”，把巡回路演带到了西部。在加利福尼亚海岸沿岸，会议室里挤满了好奇的工程师和修理工。**MITS** 团队展示了 **Altair** 和 **Altair BASIC**。然而，在 **Palo Alto** 一个特别拥挤的酒店会议室里，发生了一件意想不到的事情。这件事改变了整个软件历史的进程。

00:12:46:

Bay area
让我们先等一下。在开始说这件意想不到的事情之前，我们先来了解一下**湾 区的 Homebrew Computer Club**
电子爱好者和业余爱好者的情况。他们自称为**自 制 计 算 机 俱 乐 部**。他们互相交换零件、电路和想法，并在探索个人计算机的新世界里相互扶持。

00:13:11 - Dan Sokol:

这些人都对此感到好奇。他们中大多数都是某个领域的工程师。其中只有两三个人对计算机编程完全没有了解。当时做了一项调查，在座多少人拥有一台计算机，而又有多少人计划购买一台。这个调查十分有趣，所有人都想拥有一台计算机，但是实际上只有一两个人拥有它们，他们会把计算机带到俱乐部聚会上展示它们。我记得那时最令人尴尬的问题是，“你打算用它做什么？”而没有人知道。

00:13:46 - Saron Yitbarek:

这位是 **Dan Sokol**，自制计算机俱乐部最初的成员之一。因为每个人都想看一看 **Menlo Park**
Altair 8800，在门罗公园的一个车库里他们举行了第一次集会。

00:14:08 - Lee Felsenstein:

在 1975 年 3 月 5 日的一个雨夜，有 30 个人来到了这个车库。这里有一台为他们展示的 **Altair**。它是启动着的，但没有连接任何东西，也没有做任何事情。

00:14:22 - Saron Yitbarek:

这位是 **Lee Felsenstein**，俱乐部的另一个初始成员。

00:14:27 - Lee Felsenstein:

我们在房间里走来走去，尝试学到一些什么。我们从 **Steve Dompier** 听到了更多的报告，他订购了一台 **Altair** 计算机，它是在一月份的《大众电子》杂志上发布的。他实际上开车去了 **Albuquerque** 核实了他的订单，然后向我们报告了他的发现。

00:14:48 - Dan Sokol:

Dompier 带来了一台 **Altair**，他坐在那儿，通过前面板的开关进行编程，使它播放音乐。在大约尝试了一个小时后，有人不小心把电源线踢掉了，这使得他不得不重新开始。但在当时看来，这是“看一台计算机，而且是你能买得起的那种。”

00:15:08 - Saron Yitbarek:

在举行的聚会上还有一个人。当他看到 Altair 的时候，让他大吃一惊，但不是因为他不相信有这样一台机器存在。恰恰相反，因为有类似功能并比它好的多的机器已经存在了，他已经制造了它。那个人就是年轻的 史蒂夫·沃兹尼亚克。他的一个朋友劝说他去参加那个聚会，以便让史蒂夫展示他制造的视频终端。但是每个人都被 Altair 所吸引。在此之前，史蒂夫从未听说过 Altair，也没听说过使它工作起来的英特尔 8080 微处理器。他带了一份数据表回家，这件事带来了一个不可思议的惊喜。

00:16:01:

这是史蒂夫·沃兹尼亚克（“沃兹”）早在 2013 年自制计算机俱乐部聚会上的发言。

00:16:10 - 史蒂夫·沃兹尼亚克:

我把它带回家研究了一下，然后发现，“天哪，这些微处理器是一个芯片。”竟然能卖到 400 美元，这太疯狂了。这就是在我高中时在纸上设计的计算机。实际上，五年前我自己也制造了一个，当时我必须自己制造一个处理器。因为那时还没有微处理器。

00:16:31 - Saron Yitbarek:

在上一集中，我们了解了沃兹高中时是如何在 GE-225 计算机上开始用 BASIC 编写他自己的软件的。嗯，从高中开始，沃兹就想拥有一台属于自己的计算机。但要做到这一点，他必须包括一种编程语言，这意味着至少需要 4K 内存以及用于人工输入输出的功能来输入程序。他的机器有 256 字节的固态 RAM，而此类芯片非常的昂贵，所以他设计了一个处理器，还在高中的时候就不断地改进它。就像 Altair 一样，它有输入 1 和 0 的开关。但现在他意识到 Altair 的微处理器是他梦寐以求的。

00:17:24 - 史蒂夫·沃兹尼亚克:

你按下按钮 1、0、1、0、1、0，然后按下一个按钮，它就进入一个地址寄存器，在按下一个 1 和 0，然后写入内存。你写满了内存，在俱乐部听到了 Altair 播放音乐，是如此的兴奋。但对我来说，这都不算什么，我想要的是一台可以使用的机器，现在我要做的是输入数据直接写入内存。这太容易了，我说，“我的梦想就是拥有一台自己的计算机。”那天晚上，我看到了这种方法。

00:17:56 - Saron Yitbarek:

那天晚上，Apple I 的形象突然出现在了史蒂夫·沃兹尼亚克的脑海中。他可以通过在终端机上添加一个微处理器，几乎不用做什么就可以得到一台适合使用的计算机。他的想法是：当计算机启动时，会运行一个程序去接收输入的数据，就像打字机一样。而不再需要拨弄 1 和 0 了。再加上他制造的视频终端机，让程序员可以看到自己输入的内容，一台感觉更人性化的计算机就诞生了，这是一台对普通人有用的个人电脑。下面是 Lee Felsenstein 的发言。

00:18:42 - Lee Felsenstein:**TV typewriter**

他已经开发出一种小型的视频终端适配器，通用术语是电视打字机。可以把它连接在电视上。他当时接触了一种只需要 25 美元的处理器芯片，并意识到，“如果我把它放在带有内存的主板上，我也能在主板上放上电视终端，这样我就会拥有一台

具有视频显示的计算机。”他就这样做了，在聚会时就在为此做准备，当我们搬到 Stanford Linear Accelerator Auditorium

斯坦福直线加速器礼堂时，他占住了唯一有电源插座的座位。他总是比别人先到那儿，他正在为 Apple I 编写 BASIC 程序。在那里他开创了苹果计算机系列。

00:19:34 - Saron Yitbarek:

每次沃兹完成了他的计算机制作，他就会很兴奋地向俱乐部的每个人展示。他解释了如何使用几块芯片制造出一台价格低廉的个人计算机。沃兹是信息自由、分享知识以帮助建立更好的系统和社会的理念的主要倡导者。这与当时俱乐部的开放价值观和社会解放运动相呼应。

00:19:59:

因此，在会议结束的时候，他拿出了他的设计、硬件和软件的蓝图，免费传给大家。他认为他们每个人都可以利用他的方案来制造自己的 300 美元的计算机。但沃兹的朋友兼商业伙伴，一个名叫史蒂夫·乔布斯的人，很快就终止了他的这个想法。乔布斯一直在外奔波，并没有意识到沃兹会把 Apple I 的设计送给别人。乔布斯并不认同沃兹的黑客思维方式，他更注重专利。很快，乔布斯说服了沃兹，他们决定改为出售计算机。

00:20:42:

这种自由和专有技术之间的道德斗争，曾经不止一次发生在自制计算机俱乐部。事实上，在那次让大家对 Altair 瞠目结舌的首届俱乐部大会之后的几个月，还有一次聚会点燃了这场辩论的导火索。它发生在斯坦福直线加速器中心礼堂里。聚会结束时，数十名与会者冲上台去想要获取一份纸带程序，这是微软公司的 Altair Basic 的最新副本，是一款让所有人都很感兴趣的软件。

00:21:21:

为什么会有这些副本呢？这个软件还没有正式发布，那么它是如何在那个自制俱乐部聚会上出现的呢？原来，原始的纸带已经被偷了。这是那个时代最大的软件抢劫案。

00:21:44:

好吧，让我们具体了解一下这一切是如何发生的。还记得前面提起过关于移动 MITS 到西部去展示 Altair 和 Altair BASIC 的事吗？

00:21:54 - John Markoff:

1975 年 6 月 10 日，该公司在 Palo Alto 演示他们计算机和搭载的软件。

00:22:03 - Saron Yitbarek:

这位是 John Markoff，纽约时报的记者。

00:22:06 - John Markoff:

这家旅馆叫 Rickeys。请记住，在当时个人计算机行业实际上并不存在，对个人计算机感兴趣的大多数人也并不是真正的商人，因为那时并没有商业软件，所以他们向一个广泛的团体展示计算机。当时在新硅谷，有很多人是电气工程师，他们都是程序员。有各种各样的人对技术、对计算机感兴趣。

00:22:39 - Saron Yitbarek:

在那次演示过程中，MITS 的工作人员在将软件加载到机器上时遇到了一些麻烦。在当时，软件是打孔在纸带上的，纸带必须通过一个机械阅读器才能安装程序。当那名员工因此而慌乱时，房间里所有的目光都盯在闪闪发亮的新 Altair 上，然而人群中的某个人发现了一些别的东西，在旁边的一个纸板箱，在那个箱子里是一卷卷 Altair 的纸带，这是一个千载难逢的机会。他把手伸进箱子里并用手指缠住了一卷长长的纸带。把它装进口袋带走了。没人看见这些。

00:23:36 - John Markoff:

不知怎么回事，那卷纸带最终被一位半导体工程师得到了，他在一家名为 Signetics 的公司工作，他的名字叫 Dan Sokol，他的技术能力很强，也是参加过最初的自制计算机俱乐部聚会的人。所以 Dan 有机会接触到一台相对高速的纸带复印机，他用它做了一堆副本。直到今天 Dan 仍然坚称，他不是拿走原始纸带的人，他只是拿到了一份原纸带的副本，然后把它带到自制计算机俱乐部的下一次聚会上，并与那里的会员们分享。

00:24:17 - Dan Sokol:

由于我不道德的行为，我被称为世界上第一位软件盗版者，这是有其道理的。我是那个复制了 MITS BASIC（即微软 BASIC）纸带的人。当时有人在自制计算机俱乐部里站出来说，“谁有能力复制纸带吗？”我说我可以，就这样，我最终得到了那盘纸带并复制了它。

00:24:45 - Saron Yitbarek:

当 Dan 分发他的盗版副本时，Lee 也正在那个自制计算机俱乐部会议上。

00:24:51 - Lee Felsenstein:

所以发生的事情就是 Dan Sokol 做了 10 份副本，在那次会议上，我们拿到了副本并对他说，“这是 Altair BASIC 的副本。现在带回来的拷贝会比你拿过来的多。”

00:25:02 - John Markoff:

当时的约定是，如果你得到一个副本，你必须自己做一个副本，并与朋友分享。这是一个未知的领域。当时还没有个人计算机软件公司，所以这真的是一种狂野的西部，当时人们只是卖计算机，而共享软件。

00:25:19 - Saron Yitbarek:

在 1975 年，软件只是你用来让计算机工作的东西。个人计算机是一个全新的概念。当时的黑客们并没有与这个词联系在一起的所有想法。他们只是想分享他们的工作，通过思想和软件的自由交流来建立一个开放的社区。这次抢劫和赠品事件为一场至今仍能引起反响的争论创造了条件。软件应该自由共享还是应该被买卖？对此，比尔·盖茨一定有自己的看法，当他发现自己的软件发生了什么时，他非常愤怒。

00:26:03 - John Markoff:

当他意识到他的 BASIC 编程语言正被业余爱好者们广泛分享时，他给他们写了一封愤怒的信，指责他们窃取了他的软件，削弱了他的谋生能力。

00:26:18 - Lee Felsenstein:

我们收到了那封信。在聚会上阅读了这封信，里面有一句话：“我们花费了大量的金钱去开发它。我们用了近 4 万美元的计算机时间。”房间里的每个人都知道，那样的计算机美元是假的，这只是一个会计把戏。你没有为它们支付真正的钱，而我们也知道这一点，所以我们想，“继续抱怨吧。我们会继续做我们正在做的事情。”

00:26:45 - Dan Sokol:

他叫我们海盗和小偷。我们试图理智和理性地向他解释，你不能以 400 美元的价格出售一台价格为 400 美元的电脑的软件。在今天这个时代，很难回过头并试图解释他们当时的心态，那就是小型计算机的思想，小型计算机被用于工业，而我们只是一群用套件来制造自己计算机的爱好者。唯一的功能性软件就是这个 BASIC 解释器，它几乎充当了一个操作系统。早在 1974 年，个人计算机里还没有操作系统，我们无法与他沟通，也无法向他解释，“把手册以 100 美元卖给我们，让我们随便用软件。”他没有听这些，多年来微软的软件价格过高，被盗版，而且盗版严重。

00:27:51 - Saron Yitbarek:

俱乐部成员选择无视这封信。因为早在 1975 年，当时的版权法并没有涵盖软件。将软件从一个纸带复制到另一个纸带上不会有任何的惩罚。这种情况在 1977 年会发生变化，但在那几年里，这种做法并没有违反法律。

00:28:12 - John Markoff:

具有讽刺意味的是，比尔·盖茨并没有创造 BASIC 语言。他只是简单地创建了它的副本，是从原始设计者达特茅斯大学的一位教授那得到的，他基本上是做了一个副本，然后把它卖掉，所以这一切的根源在于分享。

00:28:31 - Saron Yitbarek:

抢劫、纸带、分享、愤怒的信件。所有这些都导致了新兴的软件业和那些被他们视为盗版者的人们之间长达数十年的战争，但在这场战争中，开源软件运动也随之兴起。它的核心价值观与那些点燃个人计算机革命的早期爱好者是一脉相承的，因为这些业余爱好者意识到，个人计算机未来的关键点在于释放软件的潜能。

00:29:07 - Lee Felsenstein:

传递纸带、互相鼓励和互相借鉴彼此成果的过程确实是使个人计算机行业成功的原因。

00:29:17 - Saron Yitbarek:

现在，我们再也没有说过最初的小偷是谁。谁偷了那条珍贵的纸带，至今仍是个谜。那些知道答案的人们也更愿意保留这个谜。

00:29:32 - Dan Sokol:

至于它是如何被“解放”的，如果你想用这个词的话，我知道是谁干的，但是我不会说，因为那个人很久以前就要求保持匿名，因为这样更安全，我尊重这种选择，并且我也会继续尊重下去。所以，我当时不在 Rickey 旅馆，但纸带却找到了传递给我的方法。

00:30:01 - Saron Yitbarek:

随着时间的推移，个人计算机革命让位于硅谷和众多风投支持的科技创业公司，但那些修理工、那些电子爱好者和业余爱好者们却从未消失。事实上，他们比以往任何时候都更强大。

00:30:20 - Limor Fried:

嗨，我叫 Limor Fried，是 Adafruit 工业公司的首席工程师兼创始人。

00:30:26 - Saron Yitbarek:

Adafruit 是一家开源硬件公司，是过去几年开始的那场新革命的一部分，即 open source hardware movement

开 源 硬 件 运 动，这场运动与那些早期的爱好者有着同样的价值观。但它变的更好一些。

00:30:43 - Limor Fried:

自制计算机俱乐部，我认为，人们带着他们的计算机加入进来是有这样一种信念
hack 的，这就像，“来看看我做的这个很酷的魔改吧”，然后每个人都会说，“天哪，这太酷了。好吧，下个月我会带来一个更棒的。”这是一个积极的反馈循环，带来了真正好的技术创新。我认为黑客哲学仍然存在，人们只是有了更多的背景知识，所以他们认为作为一个很酷的黑客，我想说的确有所进步，但它实际上已经泛化了，我认为这很好。我认为分享的价值仍然存在，相互帮助，共同努力工作与合作。这个理念贯穿始终。它存在于整个开源社区。

00:31:32 - Saron Yitbarek:

我们将用一整集来讲述开源硬件运动的兴起，这样就可以看到我们是如何进步的，并为 Limor Fried 这样的现代制造商创造空间。请继续关注几周后的第六集。下一集，是改变了世界的磁碟——软盘。

00:31:56 - Saron Yitbarek:

代码英雄是红帽的原创播客。请到 redhat.com/commandlineheroes 了解一些关于个人计算机革命的伟大研究。这里有一个美丽的轶事，你可以读到关于比尔•盖茨在 Ed Roberts 临终前拜访他的故事，如果你想知道在那次 PC 革命期间发生了什么，请查看我们最初的几期命令行英雄节目，[操作系统战争](#)。我是 Saron Yitbarek，下期之前，编码不止。

附加剧集

Forrest Mims 对 Ed Roberts 有很多话要说。听听有关 Ed 与保罗•艾伦和比尔•盖茨会面，以及他们开始合作的故事。

音频

什么是 LCTT SIG 和 LCTT LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 LCRH SIG 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-4/personal-computers>

作者: Red Hat 选题: bestony 译者: linitok 校对: Northurland, wxy

本文由 LCRH 原创编译, Linux中国 荣誉推出

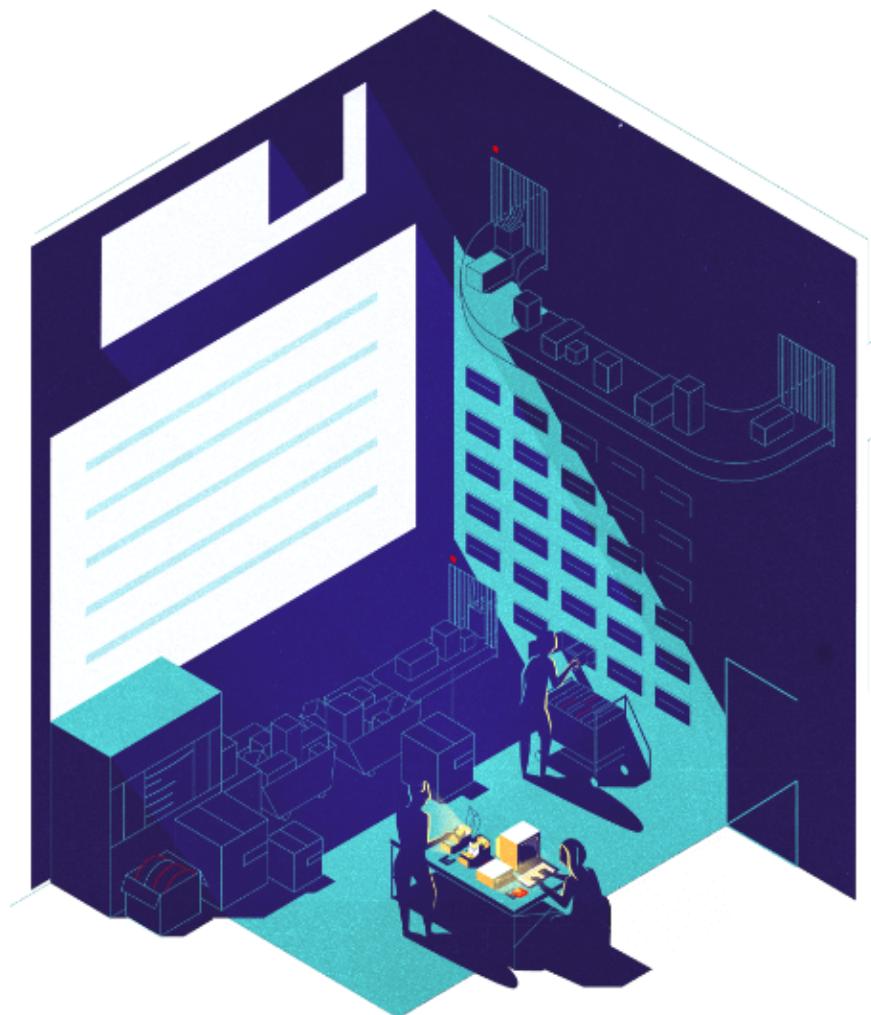
《代码英雄》第四季（4）：软盘——改变世界的磁盘

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄 是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频播客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。



本文是《代码英雄》系列播客《代码英雄》第四季（4）：软盘——改变世界的磁盘的[音频脚本](#)。

导语：软盘是计算机领域最伟大的突破之一。它以一种盛行几十年的格式助推了软件行业的发展。在某些情况下，它保存了曾经被认为是永远失去了的珍宝。

在软盘出现之前，计算机背负着打孔卡和磁带缓慢前行。Steven Vaughan-Nichols 描述了软盘所带来的巨大变化。Dave Bennet 解释了对永久存储（和便于邮寄）的需求，如何导致了第一批 8 英寸驱动器的出现。George Sollman 回顾了他是如何受命制造出更小的软盘的，以及哪些意想不到的来源激发了他的下一个设计的灵感。而当 Sollman 把它展示给 HomeBrew Computer Club 自制计算机俱乐部时，这一季的几位常客请他展示更多的内容。接下来发生的事，就众所周知了。

Matthew G. Kirschenbaum 指出，软盘在一些意想不到的地方仍然在使用。Jason Scott 和 Tony Diaz 告诉我们他们是如何将一些源代码从“跑腿网络”中带到云端的。

00:00:00 - Saron Yitbarek:

Karateka The Prince of Persia
Jordan Mechner 是个收藏爱好者。他是《空手道》和《波斯王子》等游戏的开发者。他精心保存下了开发过程中的一切细节，比如日记、素描和情节提要等所有的一切。因此，当他找不到自己保存的某样东西时，多少会让他有点吃惊，而这也确实是一件大事。

00:00:26:

Prince of Persia: The Sands of Time
早在 2002 年，Mechner 就在做《波斯王子：时之沙》的开发。程序员们希望将该游戏的经典版本作为复活节彩蛋加入到他们的 PlayStation 2 版本中。因此，他们向他索要原始的源代码。但是当 Mechner 查看他的归档时，他找不到这份源代码了。他四处寻找。这份源代码是在他的老旧 Apple II 上写的，他肯定自己保存过的这些代码，消失了。

00:00:58:

快进 10 年。Mechner 的父亲正在打扫房间，有一个看上去很破旧的鞋盒藏在壁橱后面，里面装着一堆满是灰尘的 3.5 英寸旧软盘。其中一张被标记为“《波斯王子》的源代码。版权所有 1989 年”，括号里用全大写字母写着“原版”一词。这份丢失了很长时间的代码，终于被找到了。它在那个盒子里待了四分之一世纪，然后像考古发现一样被发掘出来。

00:01:36:

但那是 2012 年。他要怎样把信息从那些旧磁盘上取出来，数据是否还完好无损？事实上，现在去挽救他的工作已经为时已晚了吗？

00:01:54:

挽救（保存）我们的工作，现如今这个过程经常是自动发生的。随着程序定期向云端推送内容，我们再也无需费心去手动保存东西了。事实上，全新的一代人甚至不知道那个保存图标表示什么意思。旁注：这不是自动售货机。

00:02:16:

但是几十年来，保存、存储和传输我们的数据都必须使用某种物理介质。当个人计算机革命开始后（我们在有关 Altair 8800 的上一集中讲过），有一项技术成为了保存的代名词——软盘。如今看起来这是如此的简单。但是软盘改变了我们的历史进程，因为它们帮助将微型计算机转变成了个人电脑。

00:02:50:

Red Hat

我是 Saron Yitbarek。这里是《代码英雄》，一档来自红帽的原创播客节目。

00:02:58:

让我们暂且把 Jordan Mechner 发现软盘的故事搁在一边。我们之后会回过头来谈论它。首先，我想先了解一下软盘最初是怎样诞生的，以及近 40 年以来它是如何成为技术界如此至关重要的一部分的。

00:03:18:

我们的第一站是 1890 年。在电子计算机出现之前，就已经有了机械的电子计算设备。数据输入输出的方法是通过一美元钞票大小的打孔卡实现的。电子计算机在上世纪 50 年代问世时，IBM 用 80 列 12 行的打孔卡进行了标准化。所打的孔就会形成一种字符，没有打孔则意味着另一种。

00:03:50:

长期以来，这些打孔卡是数据输入的主要方式。但是为更大、更复杂的程序处理数以百计的打孔卡非常麻烦。必须要有一种更好的方法来保存和传输数据。

00:04:08:

紧接着是 20 世纪 50 年代问世的纸带。要了解纸带如何在个人计算机的起源中扮演了核心角色，请收听我们的上一集节目。纸带与打孔卡具有同样的用于读取数据的打孔方式。但是因为这都在一卷纸带上，人们没必要担心卡片会混起来。它可以携带更多的数据，并且使用起来更快。但是随着微型计算机容量的增加，存储程序需要越来越多的纸带。就像打孔卡一样，纸带最终遇到了它的瓶颈。

00:04:47:

让我们进入到磁带阶段。其关键成分是聚酯薄膜，一种坚韧、有弹性的材料，表面涂有磁性氧化物来使得磁带能够记录。每盘磁带的九条磁道最多可以存储 175 MB。这在上世纪 50 和 60 年代是一件大事。10.5 英寸卷轴的磁带驱动器成了企业的标准配置。

00:05:11 - Steven Vaughan-Nichols:

但是磁带的问题在于，尽管其很方便将大量数据从一个位置移动到另一个位置，但要在磁带上搜索以找到某些特定的东西着实不易。我们会使用磁带在微型计算机和大型机上安装软件，但是对于小型便携的数据或者涉及数据交互的事情，磁带确实没那么好用。

00:05:40 - Saron Yitbarek:

这位是 CBS Interactive 的特约编辑 Steven Vaughan-Nichols。当然，磁带可以存储更多的数据。但是它太大、太慢了。确实，这仅仅在大型机领域可行。又一次，需要有一个更好的存储方法了。

00:05:58:

disk drive

更好的方法出现在 1956 年，当时 IBM 推出了其首个磁盘驱动器——IBM 350 磁盘存储单元。它是 305 RAMAC 大型计算机的组成部分，这台机器整整占满了一个房间。这位是前 IBM 磁盘和存储产品工程师 **Dave Bennet**。

00:06:23 - Dave Bennet:

存储在核心内存中。事实上，RAMAC 中的磁盘存储设备是第一个允许随机访问给定的记录的存储设备，而不是磁带驱动器。

00:06:39 - Saron Yitbarek:

很有趣。那款磁盘驱动器差点没有问世，因为它威胁到了 IBM 的打孔卡生意。但是这个项目最终还是获得了批准。问题在于，该驱动器包含由固体金属制成的磁碟。RAMAC 重达一吨，它必须由叉车搬运，并且用大型货运飞机运输。这不是最方便的存储方式，但除此之外，还有更好的解决方案。

00:07:10 - Dave Bennet:

floppy disk

尽管软 盘最初是为了满足一个新需求而开发的。原因在于有一种中间类型的存储方式。最初的存储方式是计算机代码，然后是计算机内存，即工作内存。但是随着 System 360 的出现，它们之间出现了一类新的内存，它们称之为固 件。在 System 360 中，有着独特的各种形式的固件技术，要么是一种特殊的打孔卡，要么是一种所谓的转 换 器 只 读 存 储。

00:07:51:

但是，新的需求是在半导体技术瞬息万变的时代，从这些技术转向半导体技术。这意味着在切断电源后，半导体中所存储的内容都会消失。因此，必须要有一种再次充入的方式，在电力恢复后将程序存回到内存中，以加载所谓的微 程 序 或 中间存储器。

00:08:28:

对这种设备的需求是导致软盘驱动器发展的原因。

00:08:37 - Saron Yitbarek:

因此在 1967 年，一小队由 David Noble 领导的工程师开始着手开发一款廉价的系统，用于将这些微程序载入到大型计算机。他们项目的代号是“Minnow”。

00:08:53 - Dave Bennet:

Noble 能想到的所有东西，包括各种形式的打孔卡、盒式磁带，他都亲自实验了一遍。我不知道他还实验过什么。但是他想到了使用基于柔性磁盘的廉价磁盘驱动器，和成本非常廉价的只读机制。

00:09:19 - Saron Yitbarek:

Minnow 团队希望能够把他们的微程序邮寄到需要加载它的各个地方。因此这款用于发送该程序的产品必须足够耐用，才能够在不损坏其数据的情况下实现邮寄。这是某种外壳包装方式。

00:09:38 - Dave Bennet:

现在，为了使它能够被邮寄，他们实际上必须要做的是，把它放进一个相当坚固的塑料容器里。实际上，他们会在磁盘还放在这个塑料容器（像一个信封、塑料信封）里的时候读写该磁盘。当磁盘上有涂层而且有一个刚性的磁头时，必定会有磨损。而一旦有磨损，就会产生磨损颗粒。

00:10:06:

他们所遇到的问题是，随着磨损颗粒的堆积，会引起雪崩效应。这些颗粒会充当额外的磨料。然后很快，颗粒散在那里，记录轨道会被磨损，就不再能用了。

00:10:28:

参与该项目的一个名叫 **Herb Thompson** 的家伙非常聪明，他提出了一项方案，该方案基于 **3M** 卖给家庭主妇用于家具除尘的家用除尘织物。他在信封和磁盘之间放了这样的一张布。这种材料吸收了磨损颗粒，并将其嵌入到除尘织物中，从而防止了雪崩效应，真正解决了这个问题。

00:11:00 - Saron Yitbarek:

floppy disk

柔性 8 英寸聚酯薄膜磁盘，甚至可以称它们为软 盘。它完全被装在一个很薄但是很坚固的信封中，并带有再利用的除尘装置。总之，它很容易被拿起，也很容易邮寄。每张新的软盘有 80 KB 的存储容量，可以容纳与 3000 张打孔卡相当的数据量。这是存储容量上一个巨大的飞跃。

00:11:29:

IBM 在 1971 年发布了他们的 8 英寸软盘。**Minnow** 团队做得很好，但他们不知道有多好。他们的突破即将演变成一种完全改变游戏规则的技术。

00:11:49:

IBM 的直接访问存储产品的负责人是一个名叫 **Alan Shugart** 的人。**Minnow** 项目是他的宝贝。**Shugart** 很有个性，天生具有企业家精神。他的日常制服是夏威夷衬衫，从来不穿夹克和领带。在发布 8 英寸软盘不久后，**Shugart** 离开了 **IBM**，并于 1969 年加入了 **Memorex**，然后在 1973 年成立了自己的公司 **Shugart Associates**，专门从事计算机外围设备。

00:12:23:

Don Massaro 从 **IBM** 就跟随 **Shugart**，他成为了 **Shugart Associates** 的运营副总裁。在那工作了几年之后，**Massaro** 收到了一个名叫王安的人的需求。他是 **Wang Laboratories** 王 安 电 脑 公 司 的负责人，这是当时领先的计算机公司之一。王安想出了一种磁芯存储器的方法，这是计算机在未来 20 年内使用的方法。

00:12:51:

当 1975 年王安接触 **Massaro** 时，他给了 **Massaro** 一个挑战。当时 **Shugart** 的一名工程师 **George Sollman** 还记得这件事。

00:13:03 - George Sollman:

王博士说：“我真的很想做小型商业系统和文字处理器，但是现在的磁盘驱动器——你的 8 英寸大的磁盘驱动器太大了。我们需要几个小型的存储设备。它们可以被安置在 **CRT** 显像管旁边，但我们不能用 8 英寸大的，因为磁场会破坏图像。”因此，他认为我们可以采用 8 英寸的软盘并且把它缩小。我们知道必须设计

出低成本而且可行的东西来。我们整理了一张普通老套的挂图来描述它是什么，然后拿回来和王博士见了面。王博士说：“我喜欢它，但是你不能在里面使用交流电动机，因为这会扭曲图像。”

00:13:52 - George Sollman:

所以他说：“为什么不找找看谁在制造磁带机的电动机，比如通用汽车公司？”因此，我们回过头来，实现了一些很小的电动机，这些电动机适合微型软盘，可以驱动皮带，并转动软盘。

00:14:10 - Saron Yitbarek:

George Sollman 必须确定这种新软盘的规格，它应该变得有多小，以及应该容纳什么。

00:14:19 - George Sollman:

我们所做的是，查看了当时所有的磁带机，并计算了它们的平均尺寸。完成了全部的工作后，最终选择了 5.25 英寸的软盘尺寸。我们知道我们可能只有一次机会。我记得那是在王博士把 Don 和我拉进他办公室时说的：“我给你们看一下，你们的磁盘驱动器装在我们的新文字处理器上。”他们还想向我们订购大量的磁盘驱动器，有 10000 个。因此这就好像，哇，我们成功了。

00:14:54 - Saron Yitbarek:

现如今，在历史上的技术传说中，有关 5.25 英寸这一尺寸是如何形成的，有几个不同的说法。一种说法是让软盘比口袋的尺寸大，以避免不幸的弯曲和折断。最受欢迎的民间版本是，Al Shugart 在酒吧里喝了几杯酒，然后当他弄脏了一块恰巧 5.25 平方英寸的酒吧餐巾纸时，灵光一动。这位是 Teri Shugart，Al 的女儿。

00:15:26 - Teri Shugart:

他喜欢喝酒，我可以告诉你的是，他所创办的所有公司的大部分计划都确实是在酒吧里做出的。因此，这件事并非不可能，实际上很可能就是这样。

00:15:43 - Saron Yitbarek:

但是，真正改变了 Shugart 和他团队的游戏规则的，坦率地说也是改变了所有计算机历史的关键时刻，是 George Sollman 和 Don Massaro 决定在自制计算机俱乐部展示他们的 5.25 英寸软盘。

00:16:00 - George Sollman:

有一个自制计算机俱乐部的会议在斯坦福的线性加速器中心的会议室举行，离我们住的地方大约有一英里。因此，我们带了我们的微型软盘过去并做了演示，观众里 Steve Wozniak 有个名叫史蒂夫·沃兹尼亚克的家伙，他想就我们的产品和我谈谈，并说：“我得让某些人关注到它。”

00:16:24 - Saron Yitbarek:

演示之后，George 和 Don 回归了他们的常规事务，但是几天后，在办公室里，Don 把他叫到了一边。

00:16:33 - George Sollman:

并说道：“你是营销人员，你应当保持前厅整洁，George。”然后我说：“嗯，Don你想说什么？”他说：“我们的前厅里有个流浪汉，我们应该把他弄出去。”所以我走了出去，和这个家伙握了握手。他有着一双最为锐利的眼镜，我又和他聊了一会儿，他的谈话非常有趣，他说：“我想和你去实验室看看 Steve 说的这个东西是什么。”我不知道这样是否合法，但他是我所遇到过的最有趣的人之一，很显然他有 Steve Jobs
Steve Jobs
着很高的智商和极大的好奇心，他就是史蒂夫·乔布斯。

00:17:19 - Saron Yitbarek:

史蒂夫·乔布斯讨厌苹果早期计算机中的盒式磁带驱动器。它们总是出现故障，要花很长时间大费周折获取信息。对于 Apple II，他想要一个软盘驱动器，史蒂夫·沃兹尼亚克想要制造它，但尺寸是关键因素。必须缩小软盘的尺寸，从而使得这款计算机更能吸引消费者。因此，在 1997 年圣诞节假期的两周时间里，沃兹尼亚克靠着天天吃麦当劳开发了原型。在新一年的拉斯维加斯 Consumer Electronics Show 消费电子展之前及时完成了。

00:17:57:

沃兹尼亚克很喜欢开发那款软盘驱动器，并将其描述为他参与过的最好的工程。乔布斯雇佣了 Shugart 来制造 Apple 软盘。Disk II 成为了 Apple 的主打产品，助推了 Apple II 计算机的销售并改变了行业。这位是 Steven Vaughan-Nichols。

00:18:20 - Steven Vaughan-Nichols:

如果你买了一台 Apple II，Disk II 是一个很快就会大受欢迎的选择。确实，你可以将史蒂夫·乔布斯誉为将软盘驱动器引入到个人计算机领域的人。人们正在运行名为 CPM 80 的操作系统，实际上还有数十家小型制造商和一些像 DEC 之类的大型制造商在制造 CPM 80 计算机，而所有这些计算机都使用 5.25 英寸的软盘驱动器。然后，当 IBM 在 1981 年推出第一台 IBM PC 时，每个人都清楚这不再是一个业余市场了。

00:19:04:

这是一个真正的商业市场，而这将由新流行的 5.25 英寸软盘驱动器驱动。

00:19:14 - Saron Yitbarek:

软盘和个人计算机革命是共生的。它们一个驱动着另一个，反之亦然。它们一起进化。由于有了软盘，公司可以编写程序，将程序复制到磁盘里，然后通过邮寄或在商店里出售。它对早期个人电脑的作用就像应用商店对智能手机的作用一样，为开发人员提供了便利。事实上，软盘使得软件行业成为可能，并且随着这些软件程序 Sony 变得越来越大、越来越复杂，软盘再一次发生了变化。1981 年，索尼推出了将软盘缩小到 3.5 英寸的新版本。

00:19:59:

较小的同类产品用硬塑料包裹，并内置金属遮板，以防止灰尘进入。它们更加坚固，可以存储更多的数据。有趣的是，尽管它们的外壳很硬，但它们仍被称为软盘，而当苹果公司在 1984 年发布其新 Macintosh 计算机时，又一次使得 3.5 英寸软盘成为新标准。

00:20:25 - Steven Vaughan-Nichols:

Steve Jobs Steve Wozniak

好吧，我们都知道史蒂夫·乔布斯和史蒂夫·沃兹尼亚克。像 Shugart 这些软盘驱动器的创造者，却不是家喻户晓的名字，但是他们所做的工作绝对至关重要。如果没有他们的话，个人计算机被采用的速度将会慢得多。我无法想象，如果没有这些早期的驱动器，个人计算机革命会如何发生。它使得数据输入输出计算机比其它方式更容易。

00:21:06 - Saron Yitbarek:

由于软盘尺寸适中且易于分享，因此它们具有一种社交性，就像早期的社交媒体形式一样。你可以和人们见面并交易软盘。这是在人们可以轻松访问调制解调器、互联网尚未出现之前的事情。人们分享装满了程序的软盘，就像分享装满 Cyndi Lauper 曲目的混合磁带一样。而且这种数据传输的方式甚至有个名字，即 Sneakernet 跑腿网络，从字面上看，你必须用你的双脚将数字信息从一台机器传输到另外一台，而人们在这些分享的磁盘上所存放的内容可以是任何东西，当然有软件，但是还有更多，尤其是随着新兴的数字创作工具的出现。

00:21:55:

Jaime Levy 是上世纪 80 年代后期那些新兴的数字创作者之一。她如今是 the University of Southern California 南加州大学 的教授，也是《UX Strategy》的作者。

00:22:07 - Jaime Levy:

我当然也认为这很神奇，你可以在软盘上制作出交互式的、非线性的东西。我当时 diskzine 正在制作后来被称之为磁盘杂志的杂志。那是在桌面出版的同一时期。我想当时 PageMaker 已经问世。人们刚购买了 Macintosh，正逐渐被吸引到数字技术上来。

00:22:32 - Saron Yitbarek:

此前从来都没人听说过交互式的杂志，甚至连 Jamie 也没有，但是她结合了动画、艺术、声音以及交互式界面，并使得它们适合放在软盘上。她制作了一堆副本，将它们打包并运送到了当地的一家书店。

00:22:48:

在大多数情况下，人们没有 Macintosh 甚至不能看它们的内容，他们不知道上面有什么。所以我说，在这里放 10 个副本，你拿一个回家看看，如果喜欢它的话，就把它们卖出去。他们会把它们放在前台，因为它们没法很好地被放在架子上，然后立即就有人买了它们。只要有 Mac 的人走进那里，看到这个东西只要 6 美元，他们就会说：“是的，我想要那东西。”

00:23:15:

书店里不断售空。Jaime 收到了来自世界各地的来信，并且开始引起了全国媒体的注意。不久后，她通过邮购贩卖她的磁盘做生意。然后，她作为界面设计师的职业生涯开始了。她的故事是软盘和跑腿网络力量的证明。曾经有一段时间，你可以使用单个 160 KB 的软盘驱动器运行一个完整的操作系统，但是到了上世纪 90 年代中期，你需要多张软盘才能运行任何东西，文件变得越来越大，而把软盘从你的计算机来回放入和取出实在是很烦人。

00:23:57:

1998 年，iMac 成了第一款不带软盘驱动器的个人计算机，这是一个不祥之兆。当调制解调器变得越来越好，互联网更加易于使用，人们从软盘转移到了更新的存储技术，比如 CD ROM，从 CD 我们又转到了 DVD、SD 卡、USB 闪存驱动器。回过身来，我们的身后有一整个废弃的存储技术坟墓。现如今，我们有了云。可怜的老软盘，它们不再有机会了。但是，重要的是，软盘仍然苟延残喘。它们有持久的生命力。首先，仍然有供初学者使用的“保存”图标。

00:24:43:

人们仍然有着装满了它们的鞋盒。实际上，事实证明，软盘是最具弹性的技术之一。

你可能会惊讶地发现它们仍然被用来维护

the US Intercontinental Ballistic Missile System

美国洲际弹道导弹系统等遗留机器。直到最近，该系统一直

the University of Maryland

依赖于 8 英寸的软盘驱动器。Matthew Kirschenbaum 是马里兰大学的英语和数字研究教授。

00:25:17 - Matthew Kirschenbaum:

因此，有一个关于美国空军从其核指挥和控制系统中逐步淘汰 8 英寸软盘的故事正在流传。伴随着的是人们在导弹发射井中将软盘插入某种老式大型计算机中的照片。我认为着实令人惊讶，因为看到某些东西，比如核武器仍然通过 8 英寸软盘控制着。政府、军方最终将淘汰这些系统。我当然认为那是一件好事，但也要记住，作为存储介质，软盘往往非常耐用且具有弹性。它们的面积密度低，这意味着与当今的硬盘驱动器相比，它们相对粗糙或低保真。实际上，这使得它们更具弹性，也更可靠。由于年代久远，它们也是那些计算机系统的过时之处。具有讽刺意味的是，它们可以更好地保护我们免受当今以病毒、黑客攻击和类似形式出现的网络威胁。
air gap 人们所使用的术语是“气隙”，软盘没有以任何方式连接到互联网，并且可以将其视为非常理想的安全功能。即使这使得人们有些许不安。但它们被使用了这么长时间，并不完全是没道理的。

00:26:53 - Saron Yitbarek:

当然。现在看软盘，它们看起来很脆弱，而且有点儿可笑，但是借助正确的设备，几十年前的磁盘如今仍然可以被读取。谁知道在那些数据被检索时能发现什么样的宝藏呢。

00:27:09 - Matthew Kirschenbaum:

几年前，有新闻说发现了十几张 Andy Warhol 的图片。实际上这些图片是 20 世纪 80 年代 Andy Warhol 在一台 Amiga 计算机上创造的。

00:27:26:

他得到了一台 Amiga，并进行了实验，开始将其用于他的一些艺术作品，所得到的 Pittsburgh Carnegie Mellon University 图像被保存到位于匹兹堡 Warhol 博物馆的软盘上。卡内基·梅隆大学的一个团队抢救了这些文件，所以现在我们又有 Andy Warhol 的十几件艺术品。

00:27:53 - Saron Yitbarek:

说到宝藏，是时候了解一下 Jordan Mechner 和他丢失多年的源代码到底发生了什么了。前情回顾，Jordan 丢失了《波斯王子》的源代码。多年之后，他的父亲在壁橱后面发现了一个鞋盒，里面装着一堆满是灰尘的 3.5 英寸软盘。其中一

张被标记为“源代码”，但是源代码真的在那张磁盘上吗？他能够恢复他
Prince of Persia
《波斯王子》作品吗？

00:28:23:

为了找到答案，我们和 Jordan 招募的拯救源代码小组的两个成员进行了交谈。

Internet Archive

Jordan 联系的第一个人是 Jason Scott，他是互联网档案馆的放养档案管理员。

00:28:38 - Jason Scott:

这可能行得通，但也可能不行，但是你总是希望能对最坏的结果做好准备。我认识一个人，他的名字叫 Tony，而 Tony 与苹果公司的关系非常密切。他对他们无所不知。我称他为“Apple II 耳语者”。

00:28:59 - Saron Yitbarek:

通过一个长期运行的 IRC 频道，Jason 找到了 Tony Diaz，并招募了他来担任这份工作。Jordan 邀请了这对计算机考古学家到他位于 Hollywood Hills 的家中。Tony 将一堆装备装进了卡车的后部，准备从 Oceanside 长途开车到洛杉矶。

00:29:19 - Tony Diaz:

我拿了几套闲置的 Apple IIe 系统，打包装箱，各种东西装满了半个皮卡车斗，并且还有用于读取困难的软盘的改装磁盘驱动器和各种类似的工具。磁盘驱动器控制器的功能有所不同，还有一些更为现代化的计算机上的东西，有些可以帮助处理软盘（如果有必要的话）。是的，把它们全部装载起来然后上路。

00:29:43 - Saron Yitbarek:

在 Jordan 的起居室里。Tony 建立了一个令人印象深刻的由老式 Apple II 计算机和 3.5 英寸磁盘驱动器组成的阵列。然后 Jordan 把一个装在塑料袋里的鞋盒拿了出来，就仿佛它是个圣杯一样。这个圣杯周围有橡皮筋，以防止鞋盒盖子掉落。

00:30:03 - Tony Diaz:

我曾多次遇到这种情况，刚刚打开旧的磁盘盒，它们都有同样的那种霉味。我不知道该怎么说，这很难描述，但是对于那些曾经在海军舰船上的人来说，它们气味相同，和你去到飞机后面闻到的都一样。软盘有它自己的独特的，那个词怎么说来着，光泽。

00:30:25 - Saron Yitbarek:

Tony 取出了几个磁盘，看看它们的状况。如果需要的话，他准备着取下保护套，并用 Joy 洗碗皂清洗它们。这些磁盘已经很久没有转动过了。因此，他把手指放在中间，摇晃了一下并旋转了一下，以检查是否发霉。然后他和 Jason 分成了两队。

00:30:49 - Jason Scott:

flux reading
我要去做磁通量读数，也就是磁信息，从软盘上拉取下来每一个磁性波动信息，这会产生非常巨大且难以解码的软盘镜像。这样的想法是，如果发生其他任何错误或者我们在某种单一方式受到阻碍，那么未来我们能够使用这些信息。

00:31:16:

Tony 知道这些磁盘上的信息是什么，他在计算机上使用实用工具来读取磁盘，就像早在 1990 年时候那样，这些数据对他来说是有意义的。我们去找那些很明显是标准文件副本的磁盘。试图确保在我们知道面对的是什么之前，我们不会处理任何只有单张的磁盘。这些最初的简单磁盘上的一些是诸如《Death Bounce》之类的东西。这是他制作的一款从未问世的游戏，还有 Asteroids Clone。它们能运行，我们能够在机器上玩它们，他看着他几十年来都没想起来过的游戏，它们正运行良好。

00:32:06 - Saron Yitbarek:

Prince of Persia

到目前为止，一切都很顺利。现在轮到被标记着《波斯王子》的那张磁盘了。Tony 开始分析它，并且意识到它是一个硬盘驱动器的备份，被分成了五份放在了五张软盘里。他需要把整个东西重新整合在一起。

00:32:23 - Tony Diaz:

因此，要想还原这些镜像，你必须有一个与写入时大小相同的硬盘驱动器卷。当然，我带来了硬盘，但是怎么可能正好有完全相同的大小呢？因为你并不见得总能这么巧。没关系，我将在我的卡上创建了一个 10M 的分区，并制作了这些磁盘的副本，然后告诉它是从 10M 的驱动器上进行的备份。我们继续还原它，然后得到了一个完美的硬盘驱动器，装满了待查看的源代码文件。

00:32:50 - Saron Yitbarek:

重大的发现出现在当他们完成了第一个目录并查看文件的时候。

00:32:55 - Tony Diaz:

是的，就是它。Merlin 文件，它们都以 ".s" 结尾，还有 OBJ 文件，用于编译或汇编时——都在这里了。哦，我的天哪，它有如此之多的不同版本，我们不得不把这些弄清楚，但是所有人的头基本上转向了右后方的显示器和计算机上，多少有点儿沉默，好吧，让我们来瞧瞧这上面有什么？那个呢？我记得这个。

00:33:20 - Jason Scott:

当他开始这么做，并且我们开始滚动浏览那张磁盘上的文本文件，Jordan 立即识别出来它们是他的原始源代码。因此，即使我们知道我们有了重大发现，我们也还是继续浏览所有的磁盘，以确保我们不会错过一些其他的版本。在我们发现的东西中，有《波斯王子》中其他正在进行的工作，他曾尝试过不同的图形等等。

00:33:48 - Saron Yitbarek:

令人惊讶。当团队看到可识别的源代码文件那一刻，长长地松了一口气。《波斯王子：时之沙》已经从时间的流沙中获救。不过他们的工作还没有完全完成。

00:34:09 - Saron Yitbarek:

Jason 将抢救回来的代码上传到了 GitHub，世界各地的粉丝立刻就能获取到它。这件事的消息已经散布出去，人们登录并等待着。

00:34:22 - Tony Diaz:

我们那天的主要目的是尽快将其上传到 GitHub 上的一个仓库里。我们一直都连接在同一个 IRC 聊天室，有各种各样的人问我们问题。“这是什么？你看到的是什么？你获得了什么？”而且我们在进行时得到了现场反馈。

00:34:38 - Jason Scott:

Doom Quake

曾开发过《毁灭战士》和《雷神》的 John Romero 说，他在看源代码时学到了一些技巧，而其他一些人当然完全搞不懂他们在看什么，因为在 20 世纪 80 年代后期所写的代码和今日相比相当不一样。事实上他逐个字节地移动内存，试图清理空间以使得“精灵”看起来像预期的一样。那是完全不同的一个世界。

00:35:09 - Saron Yitbarek:

自从 2012 年那重大的一天以来，《波斯王子》的源代码一直被研究、分享、评论和珍藏。这里是 Matthew G. Kirschenbaum 的一些总结。

00:35:23 - Matthew Kirschenbaum:

这是另一个例子，有关我们会认为是数字文化、有点像当今艺术作品的方式，我们当下的创造性表达方式。很多东西都被锁在了这些陈旧过时、满是灰尘的软盘里，但是凭借一点运气和工程学工作，发生了一些我们如何把东西弄回来的非常了不起的故事。

00:35:49 - Saron Yitbarek:

如今，“保存”图标是我们中的一些人离软盘最近的地方，但是当你们看到它时，我希望你们能够记住，它是这段神奇历史的一部分，我们不应该将其视为理所当然。这是一段有关分享和拯救我们所创造的东西的历史，一段有关保存的历史。

00:36:09 - Saron Yitbarek:

《代码英雄（Command Line Heroes）》是红帽（Red Hat）的一档原创播客节目。请到 redhat.com/commandlineheroes 查看我们的软盘研究笔记。顺便说一句，如果我们保存了这一集（大约 51.5 MB），我们估计它会占用 36 张 3.5 英寸的软盘。我是 Saron Yitbarek，直到下一次，继续写代码吧。

附加剧集

从打孔卡和纸带到软盘并不是简单的事情。听听王安的故事，他推动了计算机存储技术的发展。

音频

什么是 LCTT SIG 和 LCTT LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 LCRH SIG 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

via: <https://www.redhat.com/en/command-line-heroes/season-4/floppies>

作者: Red Hat 选题: bestony 译者: JonnieWayy 校对: wxy

本文由 [LCRH](#) 原创编译, [Linux中国](#) 荣誉推出