《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 都只是贝尔实验室（Bell Labs）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 Multics 的分时（time-sharing）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在西海岸，聚焦于
Homebrew Computer Clum
自制计算机俱乐部这样的组织，以及从其中发展而出的企业；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：

“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：

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：

Jon Gertner 是《The Idea Factory 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吗？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：

这位是 Joy Lisi Rankin，她是《A People's History of Computing in the United States 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：

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scientific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（Ken Thompson）、丹尼斯·里奇（Dennis Ritchie）、道格拉斯·麦克劳伊（Doug McIlroy）、约瑟夫·欧桑纳（J.F. Ossanna）。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：Uni- 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：

Joy Lisi Rankin 是《美国计算机人物史（A People's History of Computing in the United States）》一书的作者。

00:13:29：

Dennis Ritchie fellowship

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，这是他们一直用来进行 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，他们在一个他们称之为“太空旅行”的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：

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（Ken Thompson）编写。

00:15:30 - Andy Tanenbaum：

这是 BCPL（基础综合编程语言，Basic Combined Programming Language）的一种衍生语言。

00:15:33 - Saron Yitbarek：

这位是 安德鲁·塔能鲍姆（Andy Tanenbaum）。他是阿姆斯特丹的一位计算机科学教授，也是许多书籍的作者，包括经典教材《计算机网络》（Computer Networks）。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 语言对应的汇编代码仍然比 B 语言编译器的线程代码（threaded-code）技术 [1] 要快。

00:16:31 - Andy Tanenbaum：

BCPL 和 B 语言只有一种数据类型，就是双字节（word）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（byte）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，包含其它数据类型，比如说字符(char)类型
、整数(int)类型和长整数(long)类型等等。

00:17:47 - Saron Yitbarek：

因此，在 1971 年至 1973 年之间，丹尼斯(Dennis)·里奇(Ritchie)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 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。

GNU Compiler Collection

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，本质上来讲是一种 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：

安德鲁·塔能鲍姆（Andy Tanenbaum）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 MINIX 事实上也是林纳斯•托瓦兹（Linus Torvalds）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肯·汤普逊（Kenneth L. Thompson）与丹尼斯·里奇（Dennis M. Ritchie）的团队。

00:23:40 - Saron Yitbarek：

回望上世纪 60 年代，这四位贝尔实验室的员工——肯·汤普逊（Ken Thompson），丹尼斯·里奇（Dennis Ritchie），道格拉斯·麦克劳伊（Doug McIlroy）和约瑟夫·欧桑纳（J.F. Ossanna）——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**
>
> LCTT SIG 是 LCTT Special Interest Group 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

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。 ↩