unix 编程

艺术

OCR 识别 + 精确校对版

Eric S. Raymond[1] | 万泽[2]

版本：1.0

[1]作者：埃里克·斯蒂芬·雷蒙（Eric Steven Raymond，著名黑客。）
[2]编者：万泽，湖南常德人氏。邮箱：a358003542@gmail.com。

目 录

哲学

> Those who do not understand Unix are condemned to reinvent it,poorly.
>
> 不懂 Unix 的人注定最终还要重复发明一个蹩脚的 Unix。
>
> Usenet 签名，1987 年 11 月
>
> —Henry Spencer

文化？什么文化

这是一本讲 Unix 编程的书，然而在这本书里，我们将反复提到“文化”、“艺术”以及“哲学”这些字眼。如果你不是程序员，或者对 Unix 涉水未深，这可能让你感觉很奇怪。但是 Unix 确实有它自己的文化；有独特的编程艺术；有一套影响深远的设计哲学。理解这些传统，会使你写出更好地软件，即使你是在非 Unix 平台开发。

工程和设计的每个分支都有自己的技术文化。在大多数工程领域中，就一个专业人员的素养组成来说，有些不成文的行业素质具有与标准手册及教科书同等重要的地位（并且随着专业人员经验的日积月累，这些经验常常会比书本更重要）。资深工程师们在工作中会积累大量的隐形只是，他们用类似禅宗“教外别传”[译注[1]]

[1]禅宗用语，不依文字、语言、直悟佛陀所悟之境界，即称为教外别传。

的方式，通过言传身教传授给后辈。

软件工程师算是此规则的一个例外：技术变革如此之快，软件环境日新月异，软件技术文化暂如朝露。然而，例外之中也有例外。确有极少数软件技术被证明经久耐用，足以演进为强势的技术文化、有鲜明特色的艺术和世代相传的设计哲学。

Unix 文化便是其一。互联网文化又是其一——或者，这两者在 21 世纪无可争议地合二为一。其实，从 1980 年代早期开始，Unix 和互联网便越来越难以分割，本书也无意强求区分。

Unix 的生命力

Unix 诞生于 1969 年，此后便一直应用于生产领域。按照计算机工业的标准，那已经是好几个地质纪元前的事了——比 PC 机、工作站、微处理器甚至视频显示终端都要早，与第一块半导体存储器是同一个时代的古物。在现今所有分时系统中，也只有 IBM 的 VM/CMS 敢说它比 Unix 资格更老，但是 Unix 机器的服务时间却是 VM/CMS 的几十万倍；事实上，在 Unix 平台上完成的计算量可能比所有其他分时系统加起来的总和还要多。

Unix 比其它任何操作系统都更广泛地应用在各种机型上。从超级计算机到手持计算机到嵌入式网络设备，从工作站到服务器到 PC 机到微型计算机。Unix 所能支持的机器架构和奇特硬件可能比你随便抓取任何其他三种操作系统所能支持的总和还要多。

Unix 应用范围之广简直令人难以置信。没有哪一种操作系统能像 Unix 那样，能同时在作为研究工具、定制技术应用的友好宿主机、商用成品软件平台和互联网技术的重要部分等各个领域都大放异彩。

从 Unix 诞生之日起，各种信誓旦旦的预言就伴随着它，说 Unix 必将衰败，或者被其他操作系统挤出市场。可是在今天，化身为 Linux、BSD、Solaris、MacOS X 以及好几种其它变种的 Unix，却显得前所未有的强大。

> Robert Metcalf[以太网络的发明者] 曾说过：如果将来有什么技术来取代以太网，那么这个取代物的名字还会叫“以太网”。因此以太网是永远不会消亡的[2]。Unix 也多次经历了类似的转变。
>
> —Ken Thompson

至少，Unix 的一个核心技术——C 语言——已经在其他系统中植根。事实上，如果没有无处不在的 C 语言这个通用语言，还如何奢谈系统级软件工程。Unix 还引入了如今广泛采用的带目录节点的树形文件名字空间已经用于程序间通信的管道机制。

Unix 的生命力和适应力委实令人惊奇。尽管其它技术如蜉蝣般生生灭灭，计算机性能成千倍增长，语言历经嬗变，业界规范几次变革——然而 Unix 依然巍然屹立，仍在运行，仍在创造价值，仍然能够赢得这个星球上无数最优秀、最聪明的软件技术人员的忠诚。

性能—时间的指数曲线对软件开发过程所引发的结果，就是每过 18 个月，就有一半的知识会过时。Unix 并不承诺让你免遭此劫，只是让你的知识投资更趋稳定。因为不变的东西很多：语言、系统调用、工具用法——它们积年不变，甚至可以用上数十载。而在其他操作系统中则无法预判什么东西会持久不变，有时候甚至整个操作系统都会被淘汰。在 Unix 中，持久性知识和短期性知识有

[2]事实上，以太网已经两次被不同的技术所取代，只是名字没有变。第一次是双绞线取代了同轴电缆，第二次是千兆以太网的出现。

着明显的区别，人们在一开始学习的时候，就能提前判断（命中率约有九成）要学的知识属于那一类。这些便是 Unix 有众多忠实用拥趸的原因。

Unix 的稳定和成功在很多程度上归功于它与生俱来的内在优势，归功于 Ken Thompson, Dennis Ritchie, Brain Kernighan, Doug McIroy, Rob Pike 和其他早期 Unix 开发者一开始作出的设计决策。这些决策，连同设计哲学、编程艺术、技术文化一起，从 Unix 的婴儿期到今天的成长路程中，已经被反复证明是健康可靠的，而 Unix 才得以有今天的成功。

反对学习 Unix 文化的理由

Unix 的耐用性及其技术文化对于喜爱 Unix 的人们、以及技术史家来说肯定颇为有趣。但是，Unix 的本源用途——作为大中型计算机的通用分时系统，由于受到个人工作站的围剿，正迅速地退出舞台，隐入历史的迷雾之中。因而 Unix 究竟能否在目前被 Microsoft 主宰的主流商务桌面市场上取得成功，人们自然也存在着一定的疑问。

外行常常把 Unix 当作是教学用的玩具或者是黑客的沙盒而不屑一顾。有一本著名的抨击 Unix 的书——《Unix 反对者手册》（Unix Hater’s Handbook）[Garfinkel]，几乎从 Unix 诞生时就一直奉行反对路线，将 Unix 的追随者描写成一群信奉邪教的怪人和失败者。AT&T、Sun、Novell，以及其他一些大型商业销售商和标准联盟在 Unix 定位和市场推广方面不断铸下的大错也已经成为经典笑柄。

即使在 Unix 世界里，Unix 的通用性也一直受到怀疑，摇摆在

危崖边。在持怀疑态度的外行人眼中，Unix 很有用，不会消亡，只是等不了大雅之堂：注定只能是个小众的操作系统。

挫败这些怀疑者的不是别的，正是 Linux 和其他开源 Unix（如现代 BSD 各个变种）的崛起。Unix 文化是如此的有生命力，即使十几年的管理不善也丝毫未箝制[3]它的勃勃生机。现在 Unix 社区自身已经重新控制了技术和市场，正快速而有效地解决着 Unix 的问题（第 20 章将有详述）。

Unix 之失

对于一个始于 1969 年的设计来说，在 Unix 设计中居然很难找到硬伤，这着实令人称奇。其他的选择不是没有，但是每一个这样的选择同样面临争议，无论是 Unix 爱好者，还是操作系统设计社群的人们。

Unix 文件在字节层次以上再无结构可言。文件删除了就没法恢复。Unix 的安全模型公认地太过原始。作业控制有欠精致。命名方式非常混乱。或许拥有文件系统本身就是一个错误。我们将在第 20 章讨论这些技术问题。

但是也许 Unix 最持久的异议恰恰来自 Unix 哲学的一个特性，这一条特性是 X window 设计者首先明确提出的。X 致力于提供一套“机制，而不是策略”，以支持一套极端通用的图形操作，从而把工具箱和界面的“观感”（策略）推后到应用层。Unix 其他系统级的服务也有类似的倾向：行为的最终逻辑被尽可能推后到使用端。Unix 用户可以在多种 shell 中进行选择。而 Unix 应用程序通常会提供很多的行为选项和令人眼花缭乱的定制功能。

[3]同抑制，而钳制有胁迫之意。

这种倾向也反映出 Unix 的遗风：原本是技术人员设计的操作系统；同时也表明设计的信念；最终用户永远比操作系统设计人员更清楚他们究竟需要什么。

> 贝尔实验室的 Dick Hamming 在 1950 年代便树立了此信条：尽管计算机稀缺昂贵，但是开放式的计算模式，即客户可以为系统写出自己的应用程序，这一点势在必行，因为“用错误的方式解决正确的问题总比用正确的方法解决错误的问题好”。
>
> —Doug McIlroy

然而这种选择机制而不是策略的代价是：当用户“**可以**”自己设置策略时，他们其实是“必须”自己设置策略。非技术型的终端用户常常会被 Unix 丰富的选项和接口风格搞得晕头转向，于是转而选择那些伪称能够给他们提供简洁性的操作系统。

只看眼前的话，Unix 的这种自由放纵主义风格会让它失去很多非技术性用户，但从长远考虑，最终你会发觉这个“错误”换来至关重要的优势：策略相对短寿，而机制才会长存。现今流行的界面观感常常会变成明日进化的死胡同（去问问那些使用已经过时的 X 工具包的用户，他们会有一肚子苦水倒给你！）。说来说去，只提供机制不提供方针的哲学能使 Unix 长久保鲜；而那些被束缚在一套方针或界面风格的操作系统，也许早就从人们的视线中消失了。

Unix 之得

最近 Linux 爆炸式的发展和 Intemet 技术重要性的渐增，都给我们充足的理由来否定怀疑者的论断。其实，退一步说，就算怀疑

者的断言正确，Unix 文化也同样值得研习，因为在有些方面，Unix 及其外围文化明显比任何竞争对手都出色。

开源软件

尽管“开源”这个术语和开源定义 (the Open Source Definition) 直到 1998 年才出现，但是自由共享源码的同僚严格复审的开发方式打从 Unix 诞生起就是其文化最具特色的部分。

最初十年中的 AT&T 原始 Unix，及其后来的主要变种 Berkeley Unix，通常都随源代码一起发布。下文要提到的 Unix 的优势，大多数也由此而来。

跨平台可移植性和开放标准

Unix 仍是唯一一个在不同种类的计算机、众多厂商、各种专用硬件上提供了一个一致的、文档齐全的应用程序接口（API）的操作系统。Unix 也是唯一一个从嵌入式芯片、手持设备到桌面机，从服务器到专门用于数值计算的怪兽级计算机以及数据库后端都腾挪有余的操作系统。

Unix API 几乎就可以作为编写真正可移植软件的硬件无关标准。难怪最初 IEEE 称之为“可移植操作系统标准”（Portabje Operating System Standard）的 POS 很快就被大家加了后缀变成了“POSIX”[译注：缩写为 POSIX 是为了读音更像 Unix]。确实，只有称之为 Unix API 的等价物才能算是这种标准比较可信的模型。

其它操作系统只提供二进制代码的应用程序，并随其诞生环境的消亡而消亡，而 Unix 源码却是永生的。至少，永生在数十年不断维护翻修它们的 Unix 技术文化之中。

Internet 和万维网

美国国防部将第一版 TCP/IP 协议栈的开发合同交给一个 Unix 研发组就是因为考虑到 Unix 大部分是开放源码。除了 TCP/IP 之外，Unix 也已成为互联网服务提供商 (Intemet Service Provider) 行业不可或缺的核心技术之一。甚至在 1980 年代中期 TOPS 系列操作系统消亡之际，大部分互联网服务器（实际上 PC 以上所有级别的机器）都依赖于 Unix。

在 Intemet 市场上，Unix 甚至面对 Microsoft 可怕的行销大锤也毫发无伤。虽然成型于 TOPS-10 的 TCP/IP 标准（互联网的基础）在理论上可以与 Unix 分开，但当应用在其它操作系统上时，一直都饱受兼容性差、不稳定、bug 太多等问题的困扰。实际上，理论和规格说明人人都可以获取，但是只有 Unix 世界中你才见得到这些稳固可靠的现实成果。

互联网技术文化和 Unix 文化在 1980 年代早期开始汇合，现在已经共生共存，难以分割。万维网的设计——也就是互联网的现代面孔，从其祖先 ARPANET 所得到的，不比从 Unix 得到的更多。实际上，统一资源定位符 URL(Uniform Resource Locator) 作为 Web 的核心概念，也是 Unix 中无处不在的统一文件名字空间概念的泛化。要作为一个有效的网络专家，对 Unix 及其文化的理解绝对是必不可少的。

开源社区

伴随早期 Unix 源码发布而形成的社群从未消亡——在 1990 年代早期互联网技术的爆炸式发展之后，这个社群新造就了整整一代的使用家用机的狂热黑客。

今天，Unix 社区是各种软件开发的强大支持组。高质量的开源开发工具在 Unix 世界极为丰富（在本书中我们会讲到很多）。开源的 Unix 应用程序已经达到、或者超越它们专属同侪的高度[Fuzz]。整个 Unix 操作系统连同完整的工具包、基本的应用套件，都可以在互联网上免费获取。既然能够改编、重用、再造，节省自己 90% 的工作量，为什么还要从零开始编码呢？

通过协作开发与代码复用路上艰辛的探索，才耕耘出代码共享的传统。不是在理论上，而是通过大量工程实践，才有了这些并非显而易见的设计规则：程序得以形成严丝合缝的工具套装，而不是应景的解决对策。本书的一个主要目的就是阐明这些原则。

今天，方兴未艾的开源运动给 Unix 传统注入了新的血液、新的技术方法，同时也带来了新一代年轻而有才华的程序员。包括 Linux 操作系统以及共生的应用程序如 Apache、Mozilla 等开源项目已经使 Unix 传统在主流世界空前亮眼与成功。如今，在争相对未来计算基础设施进行定义的这场竞争中，开源运动似乎已经站在了胜利的边缘——新架构的核心正是运行在互联网上的 Unix 机器。

从头到脚的灵活性

许多操作系统自诩比起 Unix 来有多么的“现代”，用户界面又是多么的“友好”。它们漂亮外表的背后，却是以貌似精巧实则脆弱狭隘难用的编程接口，把用户和开发者禁锢在单一的界面方针下。在这样的操作系统中，完成设计者（指操作系统）预见的任务很容易，但如果要完成设计者没有预料到的任务，用户不是无计可施就是痛苦不堪。

相反，Unix 具有非常彻底的灵活性。Unix 提供众多的程序粘合手段，这意味着 Unix 基本工具箱的各种组件连纵开合后，将收到单个工具设计者无法想象的功效。

Unix 支持多种风格的程序界面（通常也因为给终端用户增加了明显的系统复杂度而被视为 Unix 的一个缺点），从而增加了它的灵活性：只管简单数据处理的程序而无需背上精巧图形界面的担子。

Unix 传统将重点放在尽力使各个程序接口相对小巧、简洁和正交——这也是另一个提高灵活性的方面。整个 Unix 系统，容易的事还是那么容易，困难的事呢，至少是有可能做到的。

Unix Hack 之趣

那些夸夸其谈 Unix 技术优越性的家伙一般不会提到 Unix 的终极法宝、它赖以成功的原因：Unix Hack 的趣味。

一些 Unix 的玩家有时羞于认同这一点，似乎这会破坏他们的正统形象。但是，确实如此，同 Unix 打交道，搞开发就是好玩：现在是，且一向如是。

并没有多少操作系统会被人们用“好玩”来描述。实际上，在其它操作系统下搞开发的摩擦和艰辛，就像是有人比喻的“把一头搁浅的死鲸推下海”一样费力不讨好；或者，最客气的也就是“尚可容忍”、“不是太痛苦”之类形容词。与之成鲜明对比的是，在 Unix 世界里，操作系统以成就感而不是挫折感来回报人们的努力。Unix 下的程序员通常会把 Unix 当作一个积极有效的帮手，而不是把操作系统当作一个对手还非得用蛮力逼迫它干活。

这一点有着实实在在的重要经济意义。趣味性在 Unix 早期的

历史中开启了一个良性循环。正是因为人们喜爱 Unix，所以编制了更多的程序让它用起来更好，而如今，连编制一个完整商用产品级的开源 Unix 操作系统都成了一项爱好。如果想知道这是多么惊人的伟绩，想想看你什么时候听说过谁为了好玩来临摹 OS/360 或者 VAX VMS 或者 Microsoft Windows 就行了。

从设计角度来说，趣味性也绝非无足轻重。对于程序员和开发人员来说，如果完成某项任务所需要付出的努力对他们是个挑战却又恰好还在力所能及的范围内，他们就会觉得很有乐趣。因此，趣味性是一个峰值效率的标志。充满痛苦的开发环境只会浪费劳动力和创造力；这样的环境会在无形之中耗费大量时间、资金，还有机会。

就算 Unix 在其它各个方面都一无足处，Unix 的工程文化仍然值得学习，它使得开发过程充满乐趣。乐趣是一个符号，意味着效能、效率和高产。

Unix 的经验别处也可适用

在探索开发那些我们如今已经觉得理所当然的操作系统特性的过程中，Unix 程序员已经积累了几十年的经验。哪怕是非 Unix 的程序员也能够从这些经验中获益。好的设计原则和开发方法在 Unix 上实施相对容易，所以 Unix 是一个学习这些原则和方法的良好平台。

在其它操作系统下，要做到良好实践通常要相对困难一些，但是尽管如此，Unix 文化中的有益经验仍然可以借鉴。多数 Unix 代码（包括所有的过滤器、主要脚本语言和大多数代码生成器）都可以直接移植到任何只要支持 ANSI C 的操作系统中（原因在于 C 语

言本身就是 Unix 的一项发明，而 ANSI C 程序库表述了相当大一部分的 Unix 服务）。

Unix 哲学基础

Design

附录

缩写术语表如 API 之类的请自行搜索之，这里就省略了。参考文献部分也省略了，如果有需要请查看原始 pdf 文档。