Brian Behlendorf
Apache

Scott Bradner
Internet Engineering Task Force

Jim Hamerly
Netscape

Kirk McKusick
Berkeley Unix

OPENSOURCES

Open Source革命之声(影印版)

Tim O'Reilly
O'Reilly & Associates, Inc.

Tom Paquin
mozilla.org

Bruce Perens
Open Source Initiative

Eric Raymond
Open Source Initiative

Richard Stallman
Free Software Foundation

Michael Tiemann
Cygnus Solutions

Linus Torvalds
Linux

Paul Vixie
Bind

Larry Wall
Perl

Bob Young
Red Hat

Chris DiBona, Sam Ockman & Mark Stone 编

清华大学出版社

OPENSOURCES

Open Source 革命之声（影印版）

Chris DiBona, Sam Ockman, Mark Stone 编

O REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

O'Reilly & Associates, Inc. 授权清华大学出版社出版

清华大学出版社

图书在版编目（CIP）数据

OPENSOURCES：Open Source革命之声：/（美）蒂波娜（DiBona, C.），（美）奥克曼（Ockman, S.），（美）斯通（Stone, M.）等编. —影印版. —北京：清华大学出版社，2003.7
书名原文：OPENSOURCES: Voices from the Open Source Revolution
ISBN 7-302-06591-8

Ⅰ.O… Ⅱ.①蒂… ②奥… ③斯… Ⅲ.①电子计算机工业－企业家－生平事迹－美国－英文 ②电子计算机工业－企业管理－经验－美国－英文 Ⅳ.K837.126.16 ② F471.266

中国版本图书馆CIP数据核字（2003）第031690号

北京市版权局著作权合同登记
图字：01-2003-1930号

书　　名/ OPENSOURCES：Open Source革命之声（影印版）
书　　号/ ISBN 7-302-06591-8/TP · 4934
责任编辑/ 常晓波
封面设计/ Edie Freedman，张健
出版发行/ 清华大学出版社（www.tup.tsinghua.edu.cn）
地　　址/ 北京清华大学学研大厦（邮政编码 100084）
经　　销/ 各地新华书店
印　　刷/ 北京艺辉印刷有限公司
开　　本/ 787毫米×980毫米　16开本　17.75印张
版　　次/ 2003年7月第一版　2003年7月第一次印刷
印　　数/ 0001-3000册
定　　价/ 28.00元（册）

Open Sources: Voices from the Open Source Revolution

Eric Raymond
esr@thyrsus.com
6 Karen Drive
Malvern, PA 19355

Bruce Perens
bruce@pixar.com
c/o Pixar Animation Studios
1001 West Cutting #200
Richmond, CA 94804

O'Reilly & Associates 公司介绍

O'Reilly & Associates 公司是世界上在 UNIX、X、Internet 和其他开放系统图书领域具有领导地位的出版公司，同时是联机出版的先锋。

从最畅销的*The Whole Internet User's Guide & Catalog*（被纽约公共图书馆评为20世纪最重要的50本书之一）到GNN（最早的Internet门户和商业网站），再到WebSite（第一个桌面 PC 的 Web 服务器软件），O'Reilly & Associates 一直处于 Internet 发展的最前沿。

许多书店的反馈表明，O'Reilly & Associates 是最稳定的计算机图书出版商 —— 每一本书都一版再版。与大多数计算机图书出版商相比，O'Reilly & Associates 公司具有深厚的计算机专业背景，这使得O'Reilly & Associates形成了一个非常不同于其他出版商的出版方针。O'Reilly & Associates 所有的编辑人员以前都是程序员，或者是顶尖级的技术专家。O'Reilly & Associates 还有许多固定的作者群体 —— 他们本身是相关领域的技术专家、咨询专家，而现在编写著作，O'Reilly & Associates 依靠他们及时地推出图书。因为O'Reilly & Associates 紧密地与计算机业界联系着，所以 O'Reilly & Associates 知道市场上真正需要什么图书。

出版说明

计算机网络与通信技术的成熟与广泛应用，以及Internet与Web的迅速发展，为人类的工业生产、商业活动和日常生活都带来了巨大的影响。网络与通信技术在我国的很多领域也已经广泛应用，并且取得了巨大的效益。然而，该领域的技术创新的速度之快也是有目共睹的。为了帮助国内技术人员和网络管理人员在第一时间掌握国外最新的技术，清华大学出版社引进了美国O'Reilly & Associates公司的一批、在计算机网络理论和Opensource方面代表前沿技术或者在某专项领域内享有盛名的著作，以飨读者。本套丛书采用影印版的形式，力求与国外图书“同步”出版，“原汁原味”地展现给读者各种权威技术理论和技术术语，适合于相关行业的高级技术人员、科研机构研究人员和高校教师阅读。

本批图书包括以下几种：

- 《802.11安全手册（影印版）》
- 《构建Internet防火墙（影印版）》
- 《Java技术手册（影印版）》
- 《Open Sources（影印版）》
- 《WWW信息体系结构（影印版）》
- 《LINUX RAID管理（影印版）》
- 《Peer-to-Peer（影印版）》
- 《Java实例技术手册（影印版）》
- 《Free As In Freedom（影印版）》
- 《Unix操作系统（影印版）》

Acknowledgments

No book like this happens without the help and counsel of a number of people, so I thank my Mom, Dad, Trish, Denise, Neil, and Mickey. I'd also like to thank the folks at the Coffeenet, who have been there for me. And of course my thanks to the contributors who have wasted valuable coding time to work on this book; I appreciate it!

To the people at VA Research Linux Systems, I couldn't have hoped for a better collection of smart, dedicated people; thanks for putting up with me during the writing of this book.

This book would not have happened without the continual support and dedication of Mark Stone. He is the true hero behind this book's creation. He has said that "A book could be written about how this book was written," which I'm sure he means in the nicest way possible. I'm sure one day he will look back and laugh at this time—right, Mark? Mark?

In the initial brainstorming period for the book, a number of people contributed ideas and support that eventually led to Open Sources. These people include Paul Crowley, Paul Russell, Corey Saltiel, Edward Avis, Jeff Licquia, Jeff Knox, Becky Wood, and the guy whose site (*http://slashdot.org*) acted as the catalyst, Rob Malda. Thanks to all; I hope this book is everything you had wished it to be.

Finally, I could not have completed this work without the morale of Christine Hillmer, who selflessly unpacked our apartment while I toiled away at my keyboard. You are all that I ever could have wished for and I am reminded each day how lucky I am.

—*Chris DiBona*

I'd like to thank the following people for the many ways that they have supported me: Joe McGuckin, Peter Hendrickson, Jo Schuster, Ruth Ockman, Allison Huynh, and Nina Woodard. I'd also like to thank my favorite hackers: David S. Miller and H. Peter Anvin.

—Sam Ockman

I'd like to thank Sam and Chris for bringing this crazy idea to me in the first place. I'm sure they had no idea what they were getting into, but I'm equally sure it has been worth it. I'd also like to thank each of the contributors; each is a creative spirit with more ideas than time to execute them, but each understood the importance of this project and the need to make time for it.

For every book there are people behind the scenes who work their magic to make the book a success. On this book especially, some of these quiet heroes deserve acknowledgment: Troy Mott, Katie Gardner, Tara McGoldrick, Jane Ellin, Robert Romano, Rhon Porter, Nancy Wolfe Kotary, Sheryl Avruch, Mike Sierra, and Edie Freedman. Of course I'd also like to thank my friends and family who have supported me through yet another of Dad's crazy book projects. Finally, my thanks to the good people at Lytton Coffee Roasting Company, my "office away from home."

—Mark Stone

Table of Contents

Introduction

Chris DiBona, Sam Ockman,
and Mark Stone

Prologue

Linux creator Linus Torvalds reports that the name "Linus" was chosen for him because of his parents' admiration for Nobel laureate Linus Pauling. Pauling was the rarest of men: a scientist who won the Nobel Prize not once, but twice. We find a cautionary tale for the Open Source community in the story of Pauling's foundational work that made possible the discovery of the structure of DNA.

The actual discovery was made Francis Crick and James Watson, and is famously chronicled in Watson's book *The Double Helix*. Watson's book is a remarkably frank account of the way science is actually done. He recounts not just the brilliance and insight, but the politics, the competition, and the luck. The quest for the secret of DNA became a fierce competition between, among others, Watson and Crick's lab in Cambridge, and Pauling's lab at Cal Tech.

Watson describes with obvious unease the way in which Pauling came to know that Watson and Crick had solved the mystery, and created a model of DNA's helical structure. The story here centers on Max Delbruk, a mutual friend who traveled between Cambridge and Cal Tech. While sympathetic to Watson and Crick's desire to keep the discovery secret until all results could be confirmed, Delbruk's allegiance ultimately was to science itself. In this passage, Watson describes how he learned that Pauling had heard the news:

> Linus Pauling first heard about the double helix from Max Delbruk. At the bottom of the letter that broke the news of the complementary chains, I had asked that he not tell Linus. I was still slightly afraid something would go wrong and did not want Pauling to think about hydrogen-bonded base pairs until we had a few more days to digest our position. My request, however, was ignored. Delbruk wanted to tell everyone in his lab and knew that within hours the gossip would travel from his lab in biology to their friends working under Linus. Also, Pauling made him promise to let him know the minute he heard from me. Then there was the even more important

consideration that Delbruk hated any form of secrecy in scientific matters and did not want to keep Pauling in suspense any longer.

Clearly the need for secrecy made Watson uncomfortable. One of the poignant themes that runs throughout the book is Watson's acknowledgment that competition kept parties from disclosing all they knew, and that the progress of science may have been delayed, if ever so slightly, by that secrecy.

Science, after all, is ultimately an Open Source enterprise. The scientific method rests on a process of discovery, and a process of justification. For scientific results to be justified, they must be replicable. Replication is not possible unless the source is shared: the hypothesis, the test conditions, and the results. The process of discovery can follow many paths, and at times scientific discoveries do occur in isolation. But ultimately the process of discovery must be served by sharing information: enabling other scientists to go forward where one cannot; pollinating the ideas of others so that something new may grow that otherwise would not have been born.

What Is Free Software and How Does It Relate to Open Source?

In 1984, Richard Stallman, a researcher at the MIT AI Lab, started the GNU project. The GNU project's goal was, simply put, to make it so that no one would ever have to pay for software. Stallman launched the GNU project because essentially he feels that the knowledge that constitutes a running program—what the computer industry calls the source code—should be free. If it were not, Stallman reasons, a very few, very powerful people would dominate computing.

Where proprietary commercial software vendors saw an industry guarding trade secrets that must be tightly protected, Stallman saw scientific knowledge that must be shared and distributed. The basic tenet of the GNU project and the Free Software Foundation (the umbrella organization for the GNU project) is that source code is fundamental to the furthering of computer science and freely available source code is truly necessary for innovation to continue.

Stallman worried how the world would react to free software. Scientific knowledge is often in the public domain; it is one function of academic publishing to put it there. With software, however, it was clear that just letting the source code go into the public domain would tempt businesses to co-opt the code for their own profitability. Stallman's answer to this threat was the GNU General Public License, known as the GPL (see Appendix B).

The GPL basically says that you may copy and distribute the software licensed under the GPL at will, provided you do not inhibit others from doing the same, either by charging them for the software itself or by restricting them through further licensing.

The GPL also requires works derived from work licensed under the GPL to be licensed under the GPL as well.

When Stallman and others in this book talk about free software, they are really talking about free speech. English handles the distinction here poorly, but it is the distinction between *gratis* and *liberty*, as in "Free as in speech, not as in beer." This radical message (the freedom part, not the beer part) led many software companies to reject free software outright. After all, they are in the business of making money, not adding to our body of knowledge. For Stallman, this rift between the computer industry and computer science was acceptable, maybe even desirable.

What Is Open Source Software?

In the spring of 1997, a group of leaders in the free software community assembled in California. This group included Eric Raymond, Tim O'Reilly, and VA Research president Larry Augustin, among others. Their concern was to find a way to promote the ideas surrounding free software to people who had formerly shunned the concept. They were concerned that the Free Software Foundation's anti-business message was keeping the world at large from really appreciating the power of free software.

At Eric Raymond's insistence, the group agreed that what they lacked in large part was a marketing campaign, a campaign devised to win mind share, and not just market share. Out of this discussion came a new term to describe the software they were promoting: Open Source. A series of guidelines were crafted to describe software that qualified as Open Source.

Bruce Perens had laid much of the groundwork for the Open Source Definition. One of the GNU project's stated goals was to create a freely available operating system that could serve as the platform for running GNU software. In a classic case of software bootstrapping, Linux had become that platform, and Linux had been created with the help of GNU tools. Perens had headed the Debian project, which managed a distribution of Linux that included within the distribution only software that adhered to the spirit of GNU. Perens had laid this out explicitly in a document called the "Debian Social Contract." The Open Source definition is a direct descendant of the "Debian Social Contract," and thus Open Source is very much in the spirit of GNU.

The Open Source Definition allows greater liberties with licensing than the GPL does. In particular, the Open Source Definition allows greater promiscuity when mixing proprietary and open-source software.

Consequently, an Open Source license could conceivably allow the use and redistribution of open-source software without compensation or even credit. As an example you can take great swaths of the Netscape browser source code and distribute it with another, possibly proprietary, program without even notifying Netscape. Why would

Netscape wish this? For a number of reasons, but the most compelling is that it gets greater market share for their client code, which works very well with their commercial offerings. In this way, giving away source code is a very good way to build a platform. This is also one of the reasons why the people at Netscape did not use the GPL.

This is not a small issue in the community. Late in 1998, there was an important dispute that threatened to fracture the Linux community. This fracture was caused by the advent of two software systems, GNOME and KDE, each of which aims to build an object-oriented desktop interface. On the one hand, KDE utilized Troll Technology's Qt library, a piece of code that was proprietary, but quite stable and mature. On the other hand, the GNOME people decided to use the GTK+ library, which was a completely free library, though not as mature as Qt.

In the past, Troll Technology would have had to choose between using the GPL and maintaining their proprietary stance. The rift between GNOME and KDE would have continued. With the advent of Open Source, however, Troll was able to change their license to one that met the Open Source definition, while still giving Troll the control over the technology they wanted. The rift between two important parts of the Linux community appears to be closing.

The Dark Side of the Force

Though he may not have realized it at the time, Watson stood at the threshold of a new era in biological science. At the time of the discovery of the double helix, science in biology and chemistry was essentially a craft, a practical art. It was practiced by a few men working in small groups, primarily under the auspices of academic research. The seeds of change had already been planted, however. With the advent of several medical breakthroughs, notably the polio vaccine and the discovery of penicillin, biological science was about to become an industry.

Today organic chemistry, molecular biology, and basic medical research are not practiced as a craft by a small body of practitioners, but pursued as an industry. While research continues in academia, the vast majority of researchers, and the vast majority of research dollars, belong to the pharmaceutical industry. This alliance between science and industry is an uneasy one at best. While pharmaceutical companies can fund research at a level undreamed of in academic institutions, they also fund research with a vested interest. Consider: would a pharmaceutical company rather put major funding into research for a cure for an illness that is therapy-based or medication-based?

Computer science, too, must exist in an uneasy alliance with industry. Once new ideas came primarily from academic computer scientists; now the computer industry drives innovation forward. While the rank and file of Open Source programmers are still the many computer science undergrads and graduate students around the world, more and more Open Source programmers are working in industry rather than academic settings.

Industry has produced some marvelous innovations: Ethernet, the mouse, and the Graphical User Interface (GUI) all came out of Xerox PARC. But there is an ominous side to the computer industry as well. No one outside of Redmond really thinks that it is a good idea for Microsoft to dictate, to the extent they do, what a computer desktop should look like or have on it.

Industry can have a negative impact on innovation. The Graphical Image Manipulation Program (GIMP) languished incomplete for a year at beta release 0.9. Its creators, two students at Berkeley, had left school to take jobs in industry, and left their innovation behind.

Use the Source, Luke

Open Source was not an idea decreed from the top. The Open Source movement is a genuine grass roots revolution. While evangelists like Eric Raymond and Bruce Perens have had great success changing the language around free software, that change would have been impossible if the conditions were not right. We have reached the stage where an entire generation of students who learned computer science under the influence of GNU is now at work in industry, and have quietly been bringing free software in through the back doors of industry for years. They do so not from altruistic motives, but rather to bring better code to their work.

The revolutionaries are in place. They are the network engineers, system administrators, and programmers who have thrived on open-source software throughout their education, and want to use open-source software to thrive professionally as well. Free software has become a vital part of many companies, often unwittingly, but in some cases quite deliberately. Open Source has come of age: there is such a thing as an Open Source business model.

Bob Young's company, Red Hat Software, Inc., thrives on giving away its core product: Red Hat Linux. One good way to deliver free software is to package it as a full-featured distribution with a nice manual. Young is primarily selling convenience, as most do not want to have to bother with downloading all the pieces that make up a full-featured Linux system.

But he is not the only one doing this. So why does Red Hat dominate the U.S. market? Why does SuSE Linux dominate Europe? Open-source software is a commodity market. In any commodity market, customers value a brand they can trust. Red Hat's strength comes from brand management: consistent marketing and community outreach that makes the community recommend them when their friends ask them which distribution to use. The same is true for SuSE, and the two companies own their respective markets mostly because they were first to take brand management seriously.

Supporting the community is essential. Red Hat, SuSE, and other companies in the Linux space understand that to just make money off of Linux without giving anything back would cause two problems. First, people would consider such a company a freeloader and would recommend a competitor instead. Second, a company must be able to differentiate itself from competitors. Companies like CheapBytes and Linux Central merely provide low-cost distribution, selling CDs for as little as a dollar. For Red Hat to be perceived as offering greater value than these budget distributors, Red Hat must give something back. In a wonderful irony of the Open Source model, Red Hat can afford to charge $49.95 for their distribution only because they support the development of new code and return that code to the community at large as Open Source.

This kind of brand management is new to Open Source, but an old-fashioned model of simply providing good service has been a part of the Open Source business model for a long time. Michael Tiemann helped found Cygnus on the idea that though the world's best compiler, GCC, was freely available, companies would still be willing to pay for support of and enhancements to that compiler. Co-founder John Gilmore's description of Cygnus is apt: "Making free software affordable."

In fact this model of giving away the product and selling the support is proliferating rapidly in the Open Source world now. VA Research has been making and supporting high-quality Linux systems since late 1993. Penguin Computing offers similar products and services. LinuxCare does full, soup-to-nuts support for Linux in all of its flavors. Sendmail creator Eric Allmen has now created Sendmail Inc. to provide service and enhancements for the mail server software that holds about 80% of the market share. Sendmail is an interesting case because they have a two-tiered approach to the market. It has the proprietary Sendmail Pro, and the Free Software Sendmail, which is one year behind Sendmail Pro's development cycle.

Along those same lines, Paul Vixie, the president of Vixie Enterprises and a contributor to this book, enjoys a practical monopoly through his program BIND. This unassuming program is used every time you send an email or go to a web site or download a file via ftp. BIND is the program that handles the conversion of addresses like "www.dibona.com" to their actual IP address (in this case, 209.81.8.245). Vixie enjoys a thriving consultancy derived from his program's ubiquity.

Innovation Through the Scientific Method

The most fascinating development in the Open Source movement today is not the success of companies like Red Hat or Sendmail Inc. What's intriguing is to see major corporations within the computer industry, companies like IBM and Oracle, turn their attention to Open Source as a business opportunity. What are they looking for in Open Source?

Innovation.

Science is ultimately an Open Source enterprise. The scientific method rests on a process of discovery, and a process of justification. For scientific results to be justified, they must be replicable. Replication is not possible unless the source is shared: the hypothesis, the test conditions, and the results. The process of discovery can follow many paths, and at times scientific discoveries do occur in isolation. But ultimately the process of discovery must be served by sharing information: enabling other scientists to go forward where one cannot; pollinating the ideas of others so that something new may grow that otherwise would not have been born.

Where scientists talk of replication, Open Source programmers talk of debugging. Where scientists talk of discovering, Open Source programmers talk of creating. Ultimately, the Open Source movement is an extension of the scientific method, because at the heart of the computer industry lies computer science. Consider the words of Grace Hopper, inventor of the compiler, who said, in the early 60s:

> To me programming is more than an important practical art. It is also a gigantic undertaking in the foundations of knowledge.

Computer science, though, differs fundamentally from all other sciences. Computer science has only one means of enabling peers to replicate results: share the source code. To demonstrate the validity of a program to someone, you must provide them with the means to compile and run the program.

Replication makes scientific results robust. One scientist cannot expect to account for all possible test conditions, nor necessarily have the test environment to fully test every aspect of a hypothesis. By sharing hypotheses and results with a community of peers, the scientist enables many eyes to see what one pair of eyes might miss. In the Open Source development model, this same principle is expressed as "Given enough eyes, all bugs are shallow." By sharing source code, Open Source developers make software more robust. Programs get used and tested in a wider variety of contexts than one programmer could generate, and bugs get uncovered that otherwise would not be found. Because source code is provided, bugs can often be removed, not just discovered, by someone who otherwise would be outside the development process.

The open sharing of scientific results facilitates discovery. The scientific method minimizes duplication of effort because peers will know when they are working on similar projects. Progress does not stop simply because one scientists stops working on a project. If the results are worthy, other scientists will follow up. Similarly, in the Open Source development model, sharing source code facilitates creativity. Programmers working on complimentary projects can each leverage the results of the other, or combine resources into a single project. One project may spark the inspiration for