# 自由软件，自由社会

理查德 · 斯托曼选集（第三版）

**[美] Richard M. Stallman 著**
**北京 GNU/Linux 用户组 译**[2]

2019 年 04 月

# 译者序



https://beijinglug.club

《自由软件，自由社会：理查德 · 斯托曼选集》（Free Software, Free Society: Selected Essays of Richard M. Stallman）是一本指导了无数人的书籍，它将理查德 · 斯托曼的自由软件哲学和思想，阐述得淋漓尽致，此书第一版首发于 2002 年。也是在 2002 年，北京 GNU/Linux 用户组（BLUG）成立了，经过十多年的发展壮大，已经成为中国推广自由软件的先锋力量。2015 年，《自由软件，自由社会》第三版发布。到同年冬天，BLUG 成员中已经有很大一部分成为了自由软件基金会（FSF）和电子前哨基金会（EFF）成员。2016 年春，北京 GNU/Linux 用户组正式成为中国第一个被 GNU 工程认可的 GNU 用户组（GNU User Group）[1]。

成为 GNU 用户组，意味着 GNU 和自由软件基金会认可了北京 GNU/Linux 用户组，十多年来在推广自由软件哲学和技术上的贡献。同时更是一种督促，督促我们需要为推广自由软件做更多贡献。所以您手上这本书，就是我们为 GNU 工程和自由软件做出的新的贡献。

翻译这本书的想法，来源于人民邮电出版社 2015 年出版的邓楠、李凡希翻译的理查德 · 斯托曼传记《若为自由故：自由软件之父理查德 · 斯托曼传》。看到此书出版，让我们意识到仅有他个人的传记是远远不够的，

[1]相关信息可见 https://savannah.gnu.org/projects/blug

传记只能让大家理解和懂得理查德 · 斯托曼的人生经历，却完全不足以理解他对自由软件哲学的思考和阐述，因此我们决定将他对自由软件的阐述和哲学思想的选集——《自由软件，自由社会》一书翻译出来，以自由文档许可证公开对外发布。

自由软件来到中国，已经有二十多年的历史，从 1998 年引入 GNU/Linux 到现在很多公司基于 GNU/Linux 开发的所谓 “国产操作系统”，虽然走过了二十多年的道路，但我们开发了多少真正尊重用户的软件？最近几年，国家大力推进 “自主可控” 的信息产业建设，虽然涌现出了很多优秀的国产 “开源” 软件，然而这些软件是真正尊重用户 “自主可控” 权力的吗？这些让我们深深意识到，只有社会中每一个人都能对运行在自己计算机（当然也包括智能手机、智能穿戴设备等）上的软件 “自主可控”，整个国家的信息产业才能真正实现 “自主可控”，毕竟这个国家是由每个如你我这样的普通公民组成的。

翻译《自由软件，自由社会》一书，也希望可以团结国内自由开源软件社区和高校社团，启发所有参与者找回初心，合力推动自由软件在中国落地生根，推动中国的自由软件开发更合规，融入世界潮流。

选择翻译此书第三版，有很多考量。除了因为第三版是 2015 年发行的最新版，更是因为此版本有至少三分之一都是新文章，特别是理查德 · 斯托曼最近几年在《纽约时报》和《连线》杂志上发表的重要文章，反映了最新的技术革新和社会变化：比如《如今自由软件更加重要》 (p. 32) 一文则提到了随着移动互联网、所谓 “云计算” 技术的发展，新时代自由软件的重要性比之前更加重要；又如《自由硬件和自由硬件设计》 (p. 53) 一文就覆盖了近年来热门的 “创客” 运动和自由硬件开发；还比如《民主可以承受多少监控?》 (p. 323) 一文就反映了自 2013 年 NSA 前雇员爱德华 · 斯诺登披露美国大规模监控丑闻以来，自由软件社区在 “大数据” 时代需要面对的机遇和挑战。第三版中有大量篇幅阐述软件专利和版权的内容，而这些内容是前两版较少涉及的部分，更是中国的自由软件（也包括开源软件）用户和开发者很少关注和理解的，我们认为将这一部分翻译出来，会极大推动国产自由软件的合规化。

此书的翻译，本着尊重原文，力求精品的原则。很多人认为理查德·斯托曼的观点偏激，甚至是极端的，我们翻译时在理解他论点的前提下，只直译了原文的文本，除非必要很少意译，以防曲解甚至误解作者原意，力求传达出理查德·斯托曼阐述自由软件哲学原始的本真的意思。这样也给读者留下自行判断的空间：斯托曼的观点是不是偏激极端。为了帮助读者理解，翻译时我们还加入了必要的“译者注”，以降低阅读门槛。

本书的翻译参考了既有的一些中文译版，特别感谢 GNU 网站中文化团队以及“哲思社区”曾翻译过此书的第一版和第二版的部分文章，他们高质量的翻译为本书的翻译提供了很多参考。术语和专业性内容我们也参考了国内和港澳台专家撰写的相关论文、书籍，还有中文维基百科。同时为了减少重复劳动，以合规性为前提，我们也将几篇既有的优秀中文翻译（主要是许可证文本）加入其中，感谢他们曾做出的巨大贡献。

参与本书翻译和校对的成员有（按贡献量排序）：tonghuix、nadebula、Hagb、mytbk、MandyMY、persmule、wxy、biergaizi、liushuyu、IceCatX 和 Artoria2e5。其中 nadebula 提供了 25 篇高质量翻译，特别是翻译了大量版权和专利相关的文章，而这一部分则是此前很少发表的内容；其他成员则参与了翻译和校对；感谢 mytbk 帮助解决了此书 PDF 及 EPUB 格式生成的问题；liushuyu 重新制作了“自由软件之歌”中文版乐谱，并可以通过自由的乐谱软件 MuseScore 播放聆听[1]；Hagb 志愿帮助将此书新翻译的文章向 GNU 官网反馈，并解决了多年未解决的交叉引用问题。

本书的翻译依然有不完善之处，我们非常欢迎各界朋友雅正。本书的翻译项目托管在 https://github.com/beijinglug/fsfs-zh，您可以直接在那里提交 Issue 或 Pull Request 来帮助我们改正。此处亦附上此书英文原版下载地址 https://www.gnu.org/doc/fsfs3-hardcover.pdf，以便对照。

北京 GNU/Linux 用户组（https://beijinglug.club）
2019 年 4 月于北京

[1] 参见本书附录 C 的《自由软件之歌》 (p. 347)

# 第三版序言



给理查德·斯托曼的情书，作者雅各·阿贝尔鲍姆（*Jacob Appelbaum*）

我们生活在由机器构成的信息社会里。

软件和硬件对我们的信息时代和互联网一样重要。自由软件被称为是革命性的政治理论，就如同我们可以控制自己的生活，我们也应该能够控制那些延伸我们身体的机器。这个被自由软件基金会支持的理论，已经成为过去三十年来数万人的实践和传统。

自由软件作为一种政治理论承认了软件和硬件系统在我们社会中所扮演的角色。批判之前和现在的制度是必要的。我们也许会发现自己无法修改既存制度。我们成为了别人制造的那些不公道制度的牺牲品，而他们本身就是不公道的。而这些制度的产出并非总是显而易见，特别是当我们被迫接受这些制度安排，尤其是当它们已变成标准化和品牌标准的时候。自由软件并不仅仅只是批评：它可以成为提供自由的替代品，依赖于自由的标准，自由地为大众使用。

自由软件提供了一种转变的范例，我们可以自由的从前人那里理解并学习，自由地成长和分享，从错误中学习，从学习中受益，并分享这些益处给每一个人。当我们用 Copyleft 的时候，我们能确保以后的用户都能从我们的作品中获得同样的自由。自由软件确保了我们的子孙后代也能解码数据的全部历史。这不仅确保了我们的自由权益，同样也包括他们的。

在当今这个大规模监控的时代，自由软件通过它的可验证性机制带来了必须的透明性。自由软件让我们可以通过加密，来确保数据完整性、身份认证以及匿名我们自己。在一个不断深入私有化的世界里，我们发现自由软件给出了通向自由社会的一个公共支柱。我们无法列举自由软件的好处，因为我们无法道尽自由本身的好处。推广自由软件的事业是不会终了的，就如同为正义的声索，并且需要时刻保持警惕。推广自由软件的事业是困难的，我们需要秉持排除万难的坚定意志来倡导和推动自由软件。

投身于自由软件不仅仅是对于知识，同时也包括授权：授权学习，授权修改，授权分享，以及授权使之能够被分享。争取信息时代的自由需要拒绝在自由软件核心原则上妥协，并且要有牺牲精神。很多人可能拒绝这些负担，他们只想获得更多的财富；而其他人却在努力增加人类知识的深度和广度。通过实现自由软件，我们能发现一种面向长期愿景的可持续性发展模式，不仅增加了知识，同时切实可行地将这些知识自由地分享给所有人。这份崇高的事业以其无差别的关怀泽及我们每一个人；从现实到超现实，自由软件都被涉及。

理查德 · 斯托曼是一位革命家和理论家，他为这个世界推出了自由软件。而他撰写文章中涉及的论题，被信息时代和之后创立制度及系统的人们广泛阅读和诠释，已成为数十年来必读的经典。他一生致力于人性解放事业，同时这本书也介绍了如何为这项事业提供帮助。

雅各 · 阿贝尔鲍姆[1]
JACOB APPELBAUM

[1]雅各 · 阿贝尔鲍姆（Jacob Appelbaum），美国独立记者，计算机安全研究员，艺术家和黑客（Hacker）。受雇于华盛顿大学，曾是 Tor 项目的核心开发者。——译者注

# 第一版序言



每一个时代都有引领时代发展的哲学家——也许是一个作家亦或一个艺术家。有时这些人家喻户晓，有些则是需要几代人的努力将其理念变为现实。无论是否为人熟知，一个时代会为这些抒发理想的人留下印记，他们或是盈盈细语于诗歌，或是揭竿而起于政治运动。

我们这个时代有一位哲学家。他不是艺术家，也不是专职作家。他是一位程序员。理查德 · 斯托曼（Richard Stallman）是 MIT 前程序员以及操作系统架构师。作为程序员和架构师，他创立了一项在这个日益由 “代码” 决定的世界里争取自由的运动，从此开始了他在大众生活舞台上的职业生涯。

“代码” 是一项使计算机运行的技术。无论是写在软件里或者烧在硬件上，都是由一系列指挥机器运行并写成文字的指令组成。这些机器——计算机日益定义并控制了我们的生活。它们决定了电话如何接通，电视上播什么节目；它们决定了视频是否能够通过视频流传到计算机上；它们也控制了计算机汇报给其制造商哪些信息。这些机器为我们服务，而代码控制这些机器。

我们应该如何控制这些代码？怎么理解它们？启用控制时应该有什么自由与之匹配？有什么样的权力？

这些问题也挑战着斯托曼的生活。通过他及他的作品，让我们明白了代码 “自由” 的重要性。这里的 free 并非表示写代码的人不能得到报偿，而

是意味着程序员将代码透明给所有人，任何人都可以控制代码，并按照自己的需求修改。这就是“自由软件”；“自由软件”为这个由代码构成的世界做了回答。

“Free”，斯托曼在解释这个词的时候颇有怨言。然而这并没有什么可悲叹的，模糊的概念迫使人们去思考，“Free”这个词着实足够启发思考。对当代美国人而言，“自由软件”听起来像是乌托邦，完全无法实现的，空洞的，甚至连午餐，也是免费的。那些控制着世界上最重要机器的代码怎么能是“免费的”。一个理性的社会怎么能主张这样的理想？

但“free”一词的奇怪意思只是对我们而言，而不是这个词汇本身。Free一词有很多意思，只有一个是表示免费。更多的则表示是自由。斯托曼说，把它称为“言论自由”，或者更准确地说是“自由劳动”较为恰当，并不是表示没有花费，而是表示没有被他人控制和限制。

这种机制通过一种“copyleft”模式的GPL许可证得以体现。基于copyleft条款赋予的权力，“自由软件”不仅仅是开放的，更保证了修改自由，且其他使用自由软件（技术上称为“派生”）的也必须是自由的。如果你使用了一个自由软件程序，并公开发布了修改版，那么修改版也和原版一样是自由的。这是必须的，否则会违反版权法。

“自由软件”与自由社会一样，也有其敌人。微软发起了针对GPL的战争，警告所有人GPL是一种“危险的”许可证。事实上，它说的危险性很大程度上是欺骗性的。还有反对者认为GPL“强迫”修改版也必须是自由的。光看这一条不能判断是否强迫。微软拒绝允许用户发布修改过的Office软件而不需要支付（假设）数百万的费用，如果这还不是强迫的话，那么GPL坚持的修改自自由软件的软件也必须是自由的，也不是强迫。

有些人认为斯托曼的言论太过极端，然而并不极端。事实上，显而易见的，斯托曼的作品只是将代码出现之前世界的自由简单翻译过来了。“自由软件”能确保代码世界的秩序与之前的传统是一样“自由的”。

比如：一个“自由的社会”是由法律来规范的。然而任何自由社会的法律对这些都有限制：没有自由社会可以用秘密法律。在传统上没有政府可以对其治理的对象隐藏规范。法律能够起效，只在明显公平正义的情况下

才可以。只有当法律的条款是可以被其监管的人或其管理的管理者（律师，立法机构）知悉和控制时，法律才是可见的。

法律的这种条件超出了立法者的工作。来想想美国法院的司法实践吧。当事人雇用律师来提高利益，有时候这种利益通过诉讼得以提升。在诉讼过程中，律师写出诉讼状，这些诉讼状反过来影响法官写下的意见。这些意见决定谁能够在案件中获胜，或者某法律条文是否符合宪法的精神。

这个过程中的所有材料都符合斯托曼所说的自由。诉讼简报是公开的，别人可以自由使用。论据是透明的（不等于说就是好的），论证过程也无需原律师的允许就可以拿走。法官意见可以在之后的诉讼摘要里引用，并可以复制或综合到别的诉讼简报或意见中。美国法律的“源代码”在设计上和原则上，对任何使用它的人都是开放和自由的。拿律师来说，借由重复使用之前案件的材料，律师发挥了创造力，可以作出最佳的诉讼简报。原始资料是自由的，创新和经济都是在其上建立的。

自由代码（这里指法律代码）的经济并没有扼杀律师。尽管任何人都可以使用和复制他们做成的东西，律师事务所仍然有足够动机来作出好的诉讼简报。律师也是手艺人，他或她的产品是公开的。但是这种工艺不是慈善行为。律师能得到收入，公众没有不给他们工作报酬。相反，这种经济能够蓬勃发展恰是因为之后的工作可以弥补前面的。

我们来设想一种不同的法律实践——案情和证据都是保密的，判决只有结果而没有论证，法律只由警方把持而其他人看不到，法规条令也不经解释如何执行。

我们尽可以设想这样的社会，但却不能称其为“自由”。无论是否有更好的动机，或更有效率的分配，这样的社会都无法认为是自由的。自由社会中的自由生活理想并不仅仅只是高效的分配。而是在开放和透明的前提下建立法律的限制，不能因为对领导人有利就加入相应的选项。由软件代码治理的生活也不能比这个更差。

编写代码不是诉讼。它更好，更丰富也更有建设性。创造性和动机并不依赖于对创作产品的完全控制，法律就是一个明显的例子。类似爵士乐、小说或者建筑，法律条文是基于既有的基础而建立的。这种增补和改

变一直是创新的体现。一个自由的社会必须保证其最重要的资源一直是自由的，这才像话。

这本书以一种使之更微妙且权力更明确的方式，收录了理查德·斯托曼的著作和演讲。这些文章跨度很大，从版权到自由软件运动的历史，包含了许多不为人知的论点，其中尤其是对数字世界中版权受到怀疑的变化的情况的深刻见解。这些文章是理解他思想的源泉，理解他的激情和他的正直，即使其他方面都不及此。这些文章将会启发那些接受他的理念并将之发扬光大的人们。

我并不十分了解斯托曼，但我很清楚他是一个严厉的人。他求胜心切，常常不耐烦。他对朋友可以像对敌人一样火冒三丈。他很强硬，不屈不挠，但却也很有耐心。

当我们的世界最终理解了代码的威力和危险，终于认识到代码和法律、政府一样，必须透明才能获得自由时，不妨回首看着这位强硬且不屈不挠的程序员，会发现他终身奋斗的景象终于实现：自由和知识可以在编译器下幸存。我们会认识到没有人能像斯托曼这样，可以为社会获得自由而奋斗和发声。

然而我们尚没有获得自由。我们在捍卫自由的时候也许会失败。然而无论胜败，这些作品都展现了自由的景象。创作这些作品的生活中，会启发那些就像斯托曼一样，为自由而战的人们。

劳伦斯·莱斯格[1]
LAWRENCE LESSIG

[1]劳伦斯·莱斯格（Lawrence Lessig），是一位美国学者暨学术与政治的行动主义者，哈佛法学院法学教授。他还是知识共享（Creative Commons）发起委员、软件自由法律中心（SFLC）委员、阳光基金会咨询委员与电子前哨基金会（EFF）前任委员。——译者注

# 前言

此第三版《自由软件，自由社会》（Free Software, Free Society）将第二版的很多文章更新了，并加入了很多新文章，其中三分之一都是新文章。

与前几版一样，此书首先介绍了自由软件的原则和哲学。讲述了软件为何必须自由，并解释了我们的原则是如何指导实践活动的，以及解决了有关硬件自由设计的问题。

如何称呼和框定一个问题会影响我们对这个问题的思考。企业通过术语来推广他们的理念，接受了相应术语也就支持了他们。因此，在这一版中，我们加入了新的文章来讲解 FSF（自由软件基金会，Free Software Foundation）是如何以及为什么这么称呼这些术语。

“版权” 这一部分节录了讨论版权相关法律问题的演讲，并说明如何修改这些法律。

“专利” 这一部分设想了一个解决计算机领域专利问题的方案。我将专利和版权分开讨论，是因为这两点是不能混淆在一起的。

“许可证” 这一部分大体并没有改变，依旧讲述了 GNU 的各个许可证，还有 Brett Smith 讲述的各个许可证的历史变迁，并用一篇文章解释了为什么软件项目必须升级到 GNU 通用公共许可证第三版。

这一版继续阐述自由软件社区需要面对和解决的各种危险和陷阱，包括了当下的非自由游戏、电子书和持续威胁的电子监控。

我希望此书可以告诉你，我们很容易就会失去自由，教会你如何保护，并启发你珍视自由。

要感谢 Jeanne Rasata 管理此项目，编辑书目，排版文字，并建立目录。还要感谢 Karl Berry 协助解决 Texinfo 技术问题，以及 Kyle Winfree 设计和排版此书的封面。

理查德 · 斯托曼
RICHARD STALLMAN

# 目录

# 第 1 部分

# GNU 计划和自由软件

## 什么是自由软件？


此文最初于 1996 年发表在 http://gnu.org。

### 自由软件的定义

自由软件的定义给出了一个标准，即一个特定的软件是否有资格被称为自由软件。我们时不时修改这一定义，以澄清和解决与之相关的各种细微问题。关于我们对自由软件定义所作的修改记录，请参阅 http://gnu.org/philosophy/free-sw.html “历史” 部分。

“自由软件”（Free Software）表示的是那些尊重用户和社区自由的软件。粗略的说，它赋予**用户运行、复制、分发、学习、修改并改进软件的自由**。因此，“自由软件” 是有关用户的自由权益（liberty），而不是指免费（free）的价格。为了理解这个概念，你需要将 “free” 一词理解成 “言论

自由” 中的 “自由”，而不是 “免费啤酒” 里的 “免费”。有时我们会称之为 “Libre Software”，以避免带有免费的意思。

我们争取这些自由，是因为每个人都应该拥有它。有了这些自由，用户（包括个体和团体）就可以控制程序为己所用。当用户无法控制程序时，我们称这样的软件为 “非自由”（Nonfree）或 “专有”（Proprietary）程序。非自由的程序控制了用户，而开发者控制着程序：这就让程序成为了非正义权力的帮凶[1]。

如果一个软件的用户拥有以下四项基本自由，那么该软件可以被称为自由软件：

- 基于任何目的，按你的意愿运行软件的自由（自由之零）。
- 学习软件如何工作的自由，按你的意愿修改软件以符合你的计算的自由（自由之一）。可访问源代码是此项自由的先决条件。
- 分发软件副本的自由，因此你可以帮助他人（自由之二）。
- 将你修改过的软件版本再分发给其他人的自由（自由之三）。这样可以让整个社区有机会共享你对软件的改动。可访问源代码是此项自由的先决条件。

如果一个软件可以充分授予用户所有这些自由，它就是自由软件，否则就是非自由软件。即使我们可以依据它们与这四项自由有多大的差异来区分不同的非自由发行方式，我们依然认为这些非自由软件是不符合伦理的。

在任何情况下，这些自由都必须应用于所有要使用的代码，或引导其他人使用这些软件。例如，程序 A 会自动调用程序 B 来处理一些事情，如果我们独立发布 A 也意味着用户还需要 B，因此我们需要判断 A 和 B 是否都是自由的。如果我们修改了 A 使得它不再需要 B，那么只需要 A 是自由的，就可以忽略 B。

[1] 可参见《如今自由软件更加重要》 (p. 32) 一文。

本文余下的内容，将会澄清关于特定自由是否适用的一些关键点。

再分发软件的自由（自由之二和自由之三）表示你可以在任何地方发布软件的副本给任何人，无论是否有过修改，无论是免费或收取费用。自由地再分发意味着（排除其他因素）你不需要必须事前征得任何发行许可或为此支付额外费用。

你应该有修改软件的自由并将软件用于私人的工作或娱乐，甚至不需要提到你的修改。如果你发布了自己的修改版，你不应被要求通知特定的人或以特定的方式发布。

运行软件的自由，即赋予任何个人或组织在任何计算机系统上，基于任何工作方式或任何目的运行软件，而无需与任何开发者或特定实体沟通的自由。在这个自由中，重要的只是*用户*的目的，而非*开发者*的目的；你作为用户有基于任何目的运行软件的自由；如果你将软件发布给了其他人，则他也有按自己目的运行软件的自由，但你不能将自己的目的强加给他。

以你的意愿运行软件的自由也表示你不能被要求禁止或停止运行该程序，这无关软件的功能，或对你来说它是否有用。

再分发软件副本的自由，表示分发该程序的二进制或可执行格式时必须附带源代码，包括修改过的和未修改版（以可运行格式发布程序可以方便在自由的操作系统上安装）。对特定程序也可以不发布和产生二进制可执行格式（因为一些编程语言不支持这个特性），但你仍有发布这些格式，或开发一种方法来产生这些格式文件的自由。

为了实现自由之一和自由之三（修改软件并发布修改版的自由），你必须能够访问程序的源代码，因此有权力访问程序的源代码对自由软件来说是一个必要条件。而混淆过的 “源代码” 不能算真正的源代码[1]。

自由之一允许用你修改的版本来替代原始的版本。如果一个程序从产品设计的角度发布只能用其他人修过的版本而不能用你的修改版——例如

[1]这里指的是 “混淆代码”（Obfuscated code）的做法，是将计算机程序的代码，转换成一种功能上等价，但是难于阅读和理解的形式的行为。——译者注，摘自维基百科

所谓的 “Tivo 化”[1]或 “锁定”，或者（用行业内的话来说）“安全启动”（Secure Boot）——这样自由之一就被架空而无法实现，即便这些软件的二进制是从自由的源代码编译而来，也不能算是自由软件。

修改软件的一个重要方法是合并可用的自由子程序或模块。如果一个程序的许可证让你不能将一个合理授权发布的现有模块融合进来——例如要求你是你所添加进的代码的版权所有者，那么该许可证过于严苛以致不能被视为自由的。

自由之三允许你以自由软件的方式发布修改版。一个自由许可证可允许以其他形式发布；也就是说，这不一定是 copyleft 许可证。不过如果一个许可证要求以非自由的方式发布修改版，则是非自由的。

为了让这些自由得以实现，在你遵守许可的条件下，这些条款必须是永久且不可撤销的。在你遵守许可的情况下，如果软件的开发者有权力撤销，或者添加限制性条款，那么该软件就不是自由的。

然而，一些对自由软件发布方式的限制规则是可以接受的，前提是与核心自由不冲突。例如，copyleft（最简单地说）规定你不能在分发程序的时候添加限制性条款以拒绝其他人的核心自由。这一规则与核心自由并不冲突，反而是在保护核心自由。

在 GNU 工程中，我们用 copyleft 合法保护每个人的四项基本自由，我们相信使用 copyleft 是非常重要的。然而非 copyleft 的许可证也是合乎伦理的。有关 “自由软件”、“copyleft 软件” 和其他类型软件的话题可以参见本书《自由与非自由软件的分类》 (p. 80) 一文。

“自由软件” 并不意味着 “非商业”。一个自由软件允许商业使用，商业开发以及商业发布。自由软件的商业开发已经不是一件特殊的事情。这对自由软件是非常重要的，你可能需要为获取自由软件的副本而付费，也可能不需要付费。但是无论如何获得副本，你都有复制并修改软件的自由，甚至有重新销售的自由。

[1] “Tivo” 是美国一大有线电视机顶盒产品，“Tivo 化”（Tivoization）是指该产品包含了以 GPL 许可证发布的软件，但实际上用户不能修改，因为一旦设备发现软件经过修改就会自动关机。相关内容可参考本书《为何升级到 GPLv3》 (p. 244) 一文。

对程序的修改是不是改进程序，这只是一个主观判断。如果你对软件的修改权实际上仅限于作出被他人认作改进的修改，该程序并不自由。

不过，规范修改版打包行为的条款是可以接受的，因为实质上这没有限制你发布修改版的自由，或者私下使用的自由。因此，许可证的一些条款是可以接受的，比如要求你改变修改版的名字，移除徽标（LOGO），或者修改版为你自己所有。只要这些要求不那么繁冗以至于影响了正常发布，都是可以接受的。既然你都已经做过一些修改，也就无所谓再多做一些了。

“如果你的版本在这种方式下可用，也必须能够在另一种方式下可用”，类似这样的条款也是可以接受的。同样的，比如有规则要求如果你发布了修改版，也必须给前一个开发者发送一份，也是可以接受的（注意这样的条款仍然让你有自由选择是否公开发布的权力）。要求你为公开发布版的程序附带源代码，也是可以接受的。

这里有个特殊问题，如果许可证要求你改变程序被其他程序调用时的名字。这样做实际上阻碍了你发布修改版以便在其被其它程序调用时取代原始版。这种情况仅当通过一种别名机制来标示原始程序名是修改版的别名时才可被接受。

有时政府的出口管制或者贸易制裁会影响你国际范围内发布软件的自由。软件的开发者没有权力消除或覆盖这些限制，但他们可以做的是拒绝强制要求使用程序时接受这些条件。这样，这些限制就不会影响到那些政府管辖以外的国家和人民的活动。因此自由软件许可证为了行使这些必要的自由，不能要求服从任何这些出口限制条款。

我们仅仅只是提及这些出口限制条款，而没有将这些条款作为软件许可证的条件，因为这样并没有限制用户，所以也是可以接受的。如果该出口限制条款对自由软件来说是非常普遍的，将其作为需要的条件也不是一个问题。然而这会产生一个潜在问题，出口限制法规若之后有一定修改，可能会让软件变成非自由的。

自由的许可证不能要求遵守非自由程序的许可证。所以，如果一个许可证要求你必须遵守 “你使用的所有程序” 的许可证，那么运行非自由程序的用户在这种情况下意味着需要遵守非自由程序的许可证，那么这将使

得该许可证变成非自由的。

自由软件许可证可以指定与软件相关的诉讼需要遵守哪个地区的法律，或需要在哪里提起诉讼。

大多数自由软件许可证是基于版权的，而何种要求能通过版权施加是有限制的。如果一个基于版权的许可证尊重如上文所述的自由，那么也就不会发生我们意料之外的问题（尽管这也许会发生）。然而有些自由软件许可证是基于合同的，而合同可以增加更大范围的限制。这样就有更大可能性使得该许可证出现无法接受的限制或非自由。

我们无法将这些可能的情形一一列出。如果一个基于合同的许可证以不同于基于版权许可证的方式限制了用户，此处也没有表明其为合法，那么我们就需要考察这个许可证，并且很可能会将之视为非自由的。

在谈到自由软件的时候，尽量不要用“赠予”（Give Away）或“免费”（For Free），因为这些词都是暗示价格上的免费而非自由的。一些常用语比如“盗版”也体现了一些我们希望你不会认同的意见。相关详情参见本书《应避免使用（或慎用）的词语，由于它们是不公正的或者引起混淆的》(p. 107) 一文。我们同时还根据不同语言列出了“自由软件”（Free Software）一词的翻译，请参见本书附录 B (p. 341) 。

最后，这些有关自由软件的标准需要谨慎的解读。判断一个软件许可证是否是自由软件许可证，标准就是看它是否符合自由软件精神以及用语是否确切。如果一个许可证包含了不合理的限制，即便是此文中我们没有预料到的问题，我们也会拒绝它。有些时候一个许可证会带来新的问题和思考，这需要我们与律师咨询以后，共同思考和研判。我们最终得出结论以后，将会更新这份标准以使其能更容易体现特定的许可证是否符合标准。

如果你对特定的许可证感兴趣，可以参考我们的许可证列表，位于 http://gnu.org/licenses/license-list.html。如果里面没有列出你所关心的，可以发邮件询问我们：licensing@gnu.org。

如果你打算起草一份新的许可证，请首先通过上面这个邮箱联系自由软件基金会。不同的自由软件许可证意味着用户需要花费更多精力来理解；

我们也许可以帮你从现有的自由软件许可证里找到符合你需要的。

如果这样依然不行，你真的需要一份新的许可证，有了我们的帮助也可以确保许可证符合自由软件标准并避免一些现实问题。

## 软件之外

软件手册必须是自由的[1]，就如同软件必须是自由的一样，因为手册会对软件产生一部分影响。

同理可证其他各种具有实用功能的作品，也就是说，任何体现知识可用性的作品也应该如此，比如教育资源和参考资料。维基百科（Wikipedia）就是最著名的例子。

任何领域的作品都可以是自由的，对自由软件的定义已经扩展为了对自由文化的定义，可应用于任何领域的作品[2]。

## 开放源代码?

另一些用户使用 “开放源代码”（“开源”，Open Source）一词来表示与 “自由软件” 相近（但不尽相同）的意思。我们倾向于使用 “自由软件” 一词，因为一旦你理解它表示自由而不是价格，这么称呼可以表达自由。“开放”（Open）一词并不能表达自由之要义[3]。

[1] 可参见《为什么自由软件需要自由的文档》 (p. 46) 一文

[2] 可参见 http://freedomdefined.org

[3] 可参见《为什么说开源漏掉了自由软件的要点》 (p. 90) 一文

# GNU 工程

本文最初以标题 “GNU 操作系统和自由软件运动” 发表于由 Chris DiBona 等人编写的《开源软件文集：开源革命之声》（Sebastopol: O’Reilly Media, 1999）[1]。尽管我不是 “开放源码” 的支持者，我还是贡献了这篇文章，这样自由软件运动的思想不会在那本书中完全失声。

## 第一个软件分享社区

当我在 1971 年开始在 MIT（麻省理工学院）的人工智能实验室工作的时候，我成为了那里一个已存在数年之久的软件分享社区的一员。在我们这个特别的社区里，分享软件不受任何限制；这和计算机的历史一样悠久，正如分享菜谱的行为乃是和做饭的历史一样久远的。但我们分享得比大多数人更多。

人工智能实验室使用一个叫 ITS（不兼容分时系统）的分时操作系统，由实验室的黑客[2]员工们设计，并以 Digital PDP-10——当年的大型机之一——的汇编语言写成。作为这个社区和人工智能实验室系统黑客员工的一员，我的工作便是改进这个系统。

那时我们并不称我们的软件为 “自由软件”，因为那个词尚不存在；但它们实际上就是。只要其他大学和公司的人想要移植和使用我们的程序，我们都十分欢迎。要是你看谁在用一个没见过而有趣的程序，你总可以提出要看看源代码，以便阅读，改动，或者吸收其部件以创造新的程序。

[1] 《开源软件文集：开源革命之声》一书中译本已由中国电力出版社于 1999 年 12 月出版，洪峰译。——译者注

[2] 一部分大众传媒混淆地将 “黑客” 一词用来表示 “安全破坏者”。我们作为黑客拒绝认可这个含义，并继续用这个词表示 “那些喜爱编程、享受有趣的才智，或两者兼备的人。” 见拙作 “On Hacking”，于 http://stallman.org/articles/on-hacking.html。

## 社区的解体

80 年代初，Digital 关停了 PDP-10 系列，与此同时，形势发生了剧变。PDP-10 在 60 年代优雅而强大的架构无法自然地扩展到 80 年代开始可用的更大的地址空间上。这意味着几乎所有组成 ITS 的程序都要作废。

人工智能实验室的黑客社区不久前就解体了。在 1981 年，附属的 Symbolics 公司几乎雇走了人工智能实验室的所有黑客，而严重减员使得社区已无法自持（由 Steve Levy 撰写的《黑客》一书记述了这些事件，并给出了一幅社区全盛时期的清晰图景[1]）。人工智能实验室在 1982 年买了一台新的 PDP-10，而其管理员打算使用 Digital 的非自由分时系统取代 ITS。

那时的新计算机，如 VAX 或 68020，都自带着操作系统，但没有一个是自由软件：仅仅为了得到一份可以运行的副本，你就得签保密协议。

这意味着使用计算机的第一步就是要你下保证不会去帮助他人。协作的社区被禁止了。私有软件所有者订立的规则是：如果和他人分享，你就是海盗（在英语中也用该词表示 “盗版者”——译者注）。想要改动，就来求我们吧。

说私有软件的社会制度——不许分享和改造软件的制度——是反社会的、是不道德的、是完全错误的，可能令不少读者吃惊。但是对于一个建立在分裂群众并保持用户无助的基础之上的制度，我们还能说些什么呢？对上述观点吃惊的读者可能已经将私有软件的社会制度视为理所当然，或用带有私有软件行业暗示的词语来判断。软件出版商花了大量的力气和时间去使人们相信对这个问题只有一种看法。

当软件出版商谈论 “行使” 他们的 “权利” 或 “停止盗版” [2] 时，他们实际 “说” 的是次要的。这些声明真正传达的是他们将未阐明的假设视为理所当然；公众被要求不加审视地接受这些。还是让我们来仔细审视一番吧。

其中一个假设就是，软件公司对拥有软件有着毋庸置疑的自然权利并

[1] 《黑客：计算机革命的英雄》一书已由机械工业出版社于 2011 年出版，赵俐、刁海鹏、田俊静译。——译者注

[2] 参见《应避免使用（或慎用）的词语》一文的 “盗版（Piracy）” 一节 (p. 121) 查看 “盗版” 一词的更多错误用法。

因此有权置身所有用户之上。(如果这真是一个自然权利，那么无论它对公众有多大害处，我们也不能反对。) 有趣的是，美国宪法和法律惯例排斥这种观点：版权不是自然权利，而是一个政府强加的、限制用户自然地复制权的人为垄断。

另一个未阐明的假设是，对软件唯一重要的是它能让你做什么工作——我们计算机用户不应关心我们应当拥有什么样的社会。

第三个假设是，如果我们不授予软件公司凌驾于程序的用户之上的权利，我们将没有软件可用 (或者决不会有一个程序来做这个或那个特定的工作)。这个假设，在自由软件运动展示了我们能够制造丰富的有用的软件而不在上面拴任何锁链之前，也许看似是有理的。

如果我们拒绝接受这些假设，并基于普通常识下的道德把用户放在首位来判断这些问题，我们会得到非常不同的结论。计算机用户应当有自由去修改程序以适应他们的需求，并自由地共享软件，因为帮助别人是社会的根基。

篇幅所限，结论背后的展开叙述不在此详述，读者请参阅文章《为何软件不应有主》于 http://gnu.org/philosophy/why-free.html，以及《如今自由软件更加重要》 (p. 32) 。

## 严酷的道德抉择

没了社区，继续像从前那样是不可能了。与此相反，我面临着一个严酷的道德抉择。

最简单的选择是加入私有软件世界，签署保密协议并下保证不去帮助黑客同仁。很可能我也会开发在保密协议下分发的软件，因而压迫他人并迫使他们也去背叛他们的伙伴。这样我可以挣钱并在写代码时乐在其中。但我知道当我在事业的尽头，回望筑墙分化人们的岁月时，我将发现我竟耗费了生命以使这个世界变得更糟。

当有人拒绝把我们的打印机控制程序 (该程序缺少部分特性使得打印机极其难用) 的源代码交给我和 MIT 人工智能实验室时，我已经尝过了当保密协议接受端的滋味。所以我无法对自己说保密协议是无辜的。当他拒

绝和我们分享代码时我很生气，己所不欲，毋施于人。

另一个选择是离开计算机领域，直截了当但令人不快。这样我的技艺不会被滥用，但它们会荒废。我将不会因分化限制计算机用户而被指责，但这种事仍然会发生。

所以我求索一条路线让程序员可以做一些好事。我扪心自问，有没有可能有什么项目或程序可以由我来写，以再重新成立一个社区？

答案很清晰：我们首先需要的是一个操作系统。它是开始使用一台计算机的至关重要的软件。有了操作系统你可以做很多事；没有操作系统你就完全无法使用计算机。有了自由的操作系统，我们就可以再次拥有互助的黑客社区——并邀请任何人加入。而且任何人都能使用计算机而毋须图谋去剥夺他们的朋友。

作为一个操作系统开发者，我拥有适当的技艺。故我觉上天降此大任于我，纵不觉注定成功。我决定将系统做成和 Unix 兼容以使其可移植，而且这样 Unix 用户可以很容易切换到它。追随黑客传统，我选了 GNU 这个名字，即 “GNU’s Not Unix”（意为 “GNU 不是 Unix”）的递归缩写。

一个操作系统并不意味着仅仅是一个内核，这样仅仅足够来运行其他程序。在 20 世纪 70 年代，能称得上操作系统的软件都包含了命令处理器、汇编器、编译器、解释器、调试器、文本编辑器、邮件程序，以及其它许多程序。ITS、Multics、VMS 和 Unix 都有这些。GNU 操作系统也得有。

后来我听到希勒尔的这些话[1]：

> 我不为我谁为我？我只为我我为何？此时不作更待何时？

启动 GNU 工程的决定正是基于类似的情怀。

## Free 是 “freedom” 的 “free”

“free software” 一语常常引发误解——它无关价格、它关乎自由。以下是自由软件的定义：

一个软件对你，一个特定用户而言是自由软件，当：

[1] 作为无神论者，我不追随任何宗教领袖，但我有时发现我钦佩他们说过的一些话。

- 不论目的为何，有使用该软件的自由。
- 有修改软件以符合自身需求的自由。（为使得该自由有效行使，你必须能够访问源代码，因为在没有源代码的情况下修改程序非常困难。）
- 有重新分发该软件副本的自由，既可免费亦可收费。
- 有发行该软件的修改版的自由，这样社区将从你的改进中受益。

因为 “free” 指的是自由，而不是价格，所以在销售副本和自由软件之间没有矛盾。事实上，销售副本的自由是很重要的：以 CD-ROM 形式卖出的自由软件集对社区很重要，并且出售它们是为自由软件开发筹集资金的重要方法。因此，无法给予人们这些自由的程序就不是自由软件。

因为 “free” 一词的二义性，人们用了很长时间找寻替代词，但更好的词汇尚未找到。英语比其它语言有更多的单词和词汇间的微妙差别，但它缺少一个简单而明确的单词用来表示 “自由”，就像在 “freedom” 一词中——最接近此含义的单词是 “unfettered”。诸如 “liberated”，“freedom” 和 “open” 等替代都有错误的含义或一些其它缺点。

## GNU 软件和 GNU 系统

开发一个系统是个非常大的工程。为了达成目标，我决定尽可能适配并使用现有的自由软件。例如，一开始我就决定用 TEX 来做主要的文本排版器；若干年后，我决定使用 X 窗口系统而不是为 GNU 再写一个窗口系统。

由于上述决定，和其他类似的决定，GNU 系统不同于所有 GNU 软件的集合。GNU 系统包含非 GNU 的软件，这些程序是由其他人或项目为了他们自己的目的而开发的。我们之所以能用它们是因为它们是自由软件。

## 工程启动

我在 1984 年 1 月辞去了 MIT 的工作而开始编写 GNU 软件。离开 MIT 是必要的，这样 MIT 就无法干涉我将 GNU 作为自由软件发行。如果我还在职，MIT 可能会要求拥有这些作品，并强加他们自己的发行条款，甚至

将它们变成一个私有软件包。我不希望做大量工作却只是看到它背离其初衷：创建一个新的软件分享社区。

尽管如此，Winston 教授，后来的 MIT 人工智能实验室的领导，友善地邀请我继续使用实验室的设施。

## 第一步

GNU 工程开始前不久，我听说了自由大学编译器工具包，又称 VUCK（荷兰语的“自由”一词以 *V* 开头）。该编译器设计成支持多种编程语言，包括 C 和 Pascal，并支持多种目标机器，我曾写信给其作者询问 GNU 是否可以使用它。

他带着嘲弄回答了，说大学是自由的而编译器不是。因此我决定首先为 GNU 工程而写的程序就是一个支持多种语言，多平台的编译器。

我希望能避免只靠自己编写整个编译器，因此我要来了 Lawrence Livermore 实验室开发的多平台编译器 Pastel 的源码。它支持一种适合系统编程的 Pascal 语言的扩充版本，并由该语言写成。我给它加上了一个 C 语言前端，并开始将其移植到 Motorola 68000 计算机。但当我发现该编译器需要数兆字节的栈空间，而可用的 68000 Unix 系统仅允许 64k 时，我只得放弃。

随后我了解到 Pastel 编译器的工作方式是分析整个输入文件得到一个语法树，将整个语法树转化为一条“指令”链，再产生整个输出文件，整个过程不释放任何内存空间。到此为止，我总结我只得从头开始写一个新的编译器。那个新编译器现在叫 GCC：其中没有一点 Pastel 编译器的内容，不过我仍努力把我之前写的 C 前端适配上去并用上。但那是几年后的事了：首先，我做出了 GNU Emacs。

## GNU Emacs

我从 1984 年 9 月开始写 GNU Emacs，从 1985 年初它就开始可用了。这使得我开始可以使用 Unix 系统编辑文件；因为没有兴趣学用 vi 或 ed，

在那之前我在其他类型的机器上编辑文件。

这时候，人们开始想用 GNU Emacs，因此出现了该如何发行它的问题。当然，我把它放在了我在 MIT 时用的计算机的匿名 ftp 服务器上（那台计算机 `prep.ai.mit.edu`，因而成了主要的 GNU ftp 发行站点，当它一年后退役时，我将其域名转移到我们的新 ftp 服务器上）。但在那时，不少感兴趣的人们并不在互联网上因而无法通过 ftp 得到副本。所以问题是，我该跟他们说什么？

我可以说，“找个上网的朋友帮你下载一个”。或者我可以像对原来 PDP-10 Emacs 那样做：跟他们说，“寄一盘磁带和 SASE（贴足邮资写明发信人）来，我会把 Emacs 写到磁带上寄回去。” 但我没有工作，而我正在寻找通过自由软件挣钱的方法。所以我宣布我会寄一盘磁带给任何想要的人，要价 150 美元。我以这种方式启动了发行自由软件的事业，那是今天发行整个 GNU/Linux 系统的公司们的先驱。

## 是一个对任何用户都自由的程序吗？

如果一个程序离开作者的手时是自由软件，这并不一定意味着它对拥有其副本的每一个人都是自由软件。例如，公有领域的软件[1]（没有版权的软件）是自由软件；但任何人都可以制作由它修改而来的私有版本。类似的，不少自由程序是被版权保护但按照一个简单的、允许私有修改版的宽容性协议发行。

这个问题的典型例子是 X 窗口系统。这是一个由 MIT 开发并以一个宽容性协议发行的自由软件，它很快被各个计算机公司接受。它们把 X 以仅有二进制的形式加到它们的私有 Unix 系统中，并被同样的保密协议控制着。这些 X 的副本和 Unix 一样，已经不再是自由软件。

X 窗口系统的开发者们并不认为这是一个问题——他们期望并有意使其发生。他们的目标不是自由，而仅仅是 “成功”，那种定义为 “有许多用户” 的成功。他们不在意这些用户是否拥有自由，只是希望他们人数众多。

这导致一个矛盾的情形，两种不同的自由度计算方法对同一个问题

[1] 参见《自由与非自由软件的分类》中的更多公有领域软件的相关内容 (p. 82) 。

“这个程序自由吗？”给出不同的回答。如果你基于 MIT 许可证的发行条款给出的自由作判断，你就会说 X 是自由软件。但是如果你以一般 X 用户的自由来衡量，你就只能说它是私有软件。大多数 X 用户当时正在使用的是随 Unix 系统而来的私有版本，而不是自由版本。

## 左版（Copyleft）和 GNU GPL

GNU 的目标是给与用户自由，而不仅仅是流行。所以我们需要使用可以阻止 GNU 软件被转变成私有软件的发行条款。我们使用的方法叫“左版”[1]。

左版使用版权法，但使它的作用与之通常的作用相反：它成为一种保持程序自由的手段，而不是限制程序的手段。

左版的核心思想是给与任何人运行程序、复制程序、改写程序，和发行改写后的程序的许可——但那些人无权添加自己的限制。这样一来，定义“自由软件”的关键自由藉由每个拥有副本的人得以保证，这些自由成了不可剥夺的权利。

对于一个有效的左版，修改版也得是自由的。这确保了建立在我们的工作基础上的作品发布后将有用于社区。当以程序员为业者志愿改进 GNU 软件时，左版能防止他们的雇主说：“你不能分享那些改进，因为我们要用它们来做一个我们的私有版本。”

如果我们要确保程序的每个用户的自由，就需要做出修改保证必须的自由。那些私有化 X 窗口系统的公司通常作了一些修改以将其移植到它们的系统和硬件。这些改动与 X 的大规模扩展相比而言是较小的，但是它们并非微不足道。如果进行修改是拒绝用户自由的一个借口，任何人来利用这一借口都是非常容易的。

一个有关的问题涉及将自由的程序和非自由的代码结合到一起。这样

[1]在 1984 或 1985 年，Don Hopkins（一位非常有想象力的朋友）寄给我一封信。在信封上他写下了不少有趣的话，包括这句：“Copyleft——逆转一切权利。”我用“copyleft”一词来命名我正在开发的发行理念。（本文采用了贾星克、李极光发表在云南师范大学学报的《论自由软件运动》一文中对 copyleft 一词的译法，即“左版”。——译者注。）

的结合体将不可避免变得不自由；任何一个在非自由部分上缺失的自由也将在整体上缺失。允许这样的结合将会打开足以沉掉一艘船的缺口：任何添加或结合到左版程序上的东西必须使得更大的结合版也是自由和左版的。

我们明确为大多数 GNU 软件使用了左版的 GNU 通用公共许可证，或简称 GNU GPL。我们在特定场合下有其它种类的左版可以使用。GNU 手册也是左版的，但使用一个非常简化的左版类型，因为 GNU GPL 的复杂性对手册是不需要的[1]。

## 自由软件基金会

随着使用 Emacs 的兴趣的增长，开始有其他人加入 GNU 工程，我们感到再次筹集资金的时候到了。所以我们在 1985 年建立了自由软件基金会（FSF），一个发展自由软件的免税慈善机构。FSF 也接手了发行 Emacs 磁带的工作；后来通过将其他自由软件（既有 GNU 的也有非 GNU 的）加到磁带上，和卖软件的自由手册扩展了该业务。

FSF 的大部分收入曾经来自销售自由软件的副本和其他相关服务（源代码的 CD-ROM、二进制文件的 CD-ROM、精心印刷的手册，都有着再发行和修改的自由），以及豪华的发行版（我们为客户选择的平台定制的完整的软件集）。今天 FSF 仍然销售手册和其他部件[2]，但大部分的资金来自成员的会费。你可以通过 http://fsf.org/join 来加入 FSF。

自由软件基金会的雇员已经编写并维护了大量的 GNU 软件包。两个值得注意的是 C 库和 shell。GNU C 库是运行于 GNU/Linux 系统上的任一程序用于和 Linux 通信的组件，由自由软件基金会的成员之一，Roland McGrath 所开发。用于大部分 GNU/Linux 系统的 shell 是 BASH，“Bourne Again Shell”[3]，由 FSF 雇员 Brian Fox 编写。

[1] 我们现在对文档使用 GNU 自由文档许可证。

[2] 于我们的在线商店中可见，位于 http://shop.fsf.org。

[3] “Bourne Again Shell” 是对 “Bourne Shell”——Unix 上普遍使用的 shell 玩的文字游戏。（Bourne 与表示出生的 born 谐音——译者注）

我们资助了这些程序的开发是因为 GNU 工程并不仅仅与工具和开发环境有关。我们的目标是一个完整的操作系统，而该目标需要这些程序。

## 支持自由软件

自由软件的哲学抵制一种特定的分布极广的商业实践，但它不反对商业。当商业尊重用户的自由时，我们祝愿它们成功。

销售 Emacs 的副本展现了一种自由软件的生意。当 FSF 接手了这项生意之后，我需要另一种方法谋生。我在销售与我所开发的自由软件相关的服务中找到了它。它包括教人们诸如对 GNU Emacs 编程，定制 GCC，和主要是移植 GCC 到新平台的软件开发的课题。

今天，每个此类的自由软件生意被许多公司实践着。有的通过 CD-ROM 发行自由软件集；其他的则销售服务，在从回答用户问题，到改正程序错误，以至增加大的新功能等不同层次上。我们甚至开始看到基于发起新的自由软件产品的自由软件公司。

值得特别注意的是，不少公司尽管将它们自己与 “open source” 一词联系在一起，实际上它们的生意是基于与自由软件一起工作的非自由软件。它们不是自由软件公司，它们是私有软件公司，其产品诱惑用户远离自由。它们称此为 “增值包”，反映了它们希望我们接受的价值观念：便利在自由之上。如果我们更珍惜自由，我们应该称它们为 “去自由的” 包。

## 技术目标

GNU 的首要目标是作为自由软件。即使 GNU 对 Unix 没有技术优势，它也有允许用户合作的社会优势；它还有道德优势：尊重用户的自由。

但是将已知好的实践检验过的标准应用到工作上是很自然的——例如，动态地分配数据结构以避免武断地固定大小限制，并在任何有意义之处处理所有可能的 8 位代码。

另外，我们放弃了面向小内存的 Unix 设计，决定不支持 16 位机器（显然 32 位机器在 GNU 系统完成时将成为主流），并且不为了减少内存使

用而作任何努力，除非超过了一兆字节。在处理非常大但不是至关紧要的文件的程序中，我们鼓励程序员们将整个文件读入内存，然后扫描其内容而不必顾虑输入输出的问题。

这些决定使得不少 GNU 程序在可靠性和速度上超越了在 Unix 上的对应的程序。

## 捐来的计算机

当 GNU 工程声名鹊起，人们开始给该工程捐赠运行 Unix 的计算机。这非常有用，因为开发 GNU 组件最轻松的方法就是在一个 Unix 系统上做，然后一个一个地替换掉 Unix 系统上的组件。但这引发了一个道德问题：我们拥有 Unix 的副本从根本上说是不是正当的。

Unix 以前是（并且现在还是）私有软件，而 GNU 工程的哲学说我们不该用私有软件。然而，应用那些与推论出“自卫的暴力是正当的”相同的理由，我的结论是：在开发用于帮助他人停用私有软件包的自由软件代替品的关键时刻，使用私有软件包是合理的。

但是，尽管这是一个可以合理化的罪恶，它仍然是罪恶。今天我们已经不再持有任何 Unix 的副本，因为我们已经用自由的操作系统取代了它们。如果我们不能将一台计算机的操作系统换成自由的，我们就把整台计算机都换掉。

## GNU 任务清单

随着 GNU 工程的进行，以及越来越多的系统组件被找到或开发，最终使得整理一份未完成工作的清单变得很有用。我们用它来招募开发者来编写缺失的部分。它被称为 GNU 任务清单。除了尚未完工的 Unix 组件外，我们列出了额外的各种各样的其它有用的软件和文档项目，我们认为，这些是一个真正完整的系统所应当拥有的。

如今[1]，留在 GNU 任务清单中的 Unix 组件除了一些无关紧要的之外已经几乎没有了——它们都已经被完成了。但清单中充满了可以被称为“应用程序”的项目。将任何不止能吸引一小部分用户的程序加到操作系统中都是有益的。

就连游戏都在任务清单里——而且从一开始就有。Unix 包含游戏，所以 GNU 自然也该包含。但兼容性对游戏来说不是问题，所以我们没有跟着 Unix 已有的游戏列表走。作为代替，我们列出了用户可能会喜欢的一系列不同种类的游戏。

## GNU 库 GPL

GNU C 库使用一种特别的左版，称作 GNU 库 GPL（LGPL）[2]，允许将私有软件链接到该库上。为什么需要这样的特例？

这不是个原则问题：没有哪个原则说私有软件产品有资格包含我们的代码。（为什么要为一个严辞拒绝与我们分享的工程作贡献呢？）为 C 库，或任何库使用 LGPL，是个策略问题。

C 库做的是通用的工作：每个私有系统或编译器都附带 C 库。因此，让我们的 C 库只能为自由软件所用不会为自由软件带来任何优势——这只会吓阻人们使用我们的库。

有一个系统是这个的例外：在 GNU 系统（包括 GNU/Linux）中，GNU C 库是唯一的 C 库。所以 GNU C 库的发行许可决定了是否可以为 GNU 系统编译私有程序。没有道德理由允许私有应用在 GNU 系统中运行，但从战略上来看禁止它们会更多地吓阻人们使用 GNU 系统，而不是鼓励开发自由应用。所以使用库 GPL 对 C 库是个好策略。

对于其他的库，策略性的决定需要具体问题具体分析。当一个库做的

[1]写于 1998 年。2009 年我们不再维护长任务列表。社区的自由软件开发得很快，我们没法全部追踪到。取而代之，我们有一个高优先级项目的列表——一个更短的列表，列举了我们十分想要鼓励人们去写的项目。

[2]这个许可证现在叫 GNU Lesser General Public License（GNU 宽通用公共许可证），以免给出所有库都该使用它的印象。更多信息见《为什么你不应该在下一个函数库中使用宽 GPL》于 http://www.gnu.org/philosophy/why-not-lgpl.html。

是一种能帮助编写特定种类程序的特殊工作时，那么将其用 GPL 发行，限制其只能被用于自由软件，是一种帮助其它自由软件开发者的方法。这给了他们面对私有软件的一个优势。

想一下 GNU Readline，一个被开发用来为 BASH 提供命令行编辑功能的库。Readline 是用普通的 GNU GPL 而不是库 GPL 发行的。这可能确实减少了 Readline 的使用量，但这对我们没有损失。与此同时，至少有一个可用的应用特地为了能够使用 Readline 而变成了自由软件，那是社区真正的收获。

私有软件有金钱提供的优势；自由软件开发者则要相互取得优势。我希望有朝一日我们能拥有大量受 GPL 保护的、没有可用的私有替代的库，提供作为新自由软件的砖石的有用模块，并为进一步的自由软件开发添加巨大的优势。

## 搔到痒处?

Eric Raymond[1] 说 “每一个优秀的软件作品都从搔到开发者个人的痒处（意为 ‘解决开发者个人的问题’——译者注）开始[2]。” 也许有时是这样。但不少 GNU 软件的关键部分是为了一个完整的自由操作系统而开发的。它们来自愿景和计划，而不是冲动。

例如，我们开发了 GNU C 库因为类 Unix 系统需要一个 C 库，开发了 BASH 因为类 Unix 系统需要一个 shell，还有 GNU tar 因为类 Unix 系统需要一个 tar 程序。我们自己的程序也同样如此——GNU C 编译器，GNU Emacs，GDB 和 GNU Make。

有些 GNU 程序是为应对我们的自由的特定威胁而开发的。为此，我们开发了 gzip 来取代因 LZW 专利而从社区流失的 Compress 程序。我们发现有人开发 LessTif，后来我们还开始了 GNOME 和 Harmony，来解决因某些私有软件库（见下文）所带来的问题。我们正在开发 GNU 隐私卫士（即

[1] Eric Raymond 是开源的主要拥护者；参见《为什么说开源漏掉了自由软件的要点》(p. 90)。

[2] 埃里克 ‘雷蒙德（Eric. S. Raymond），《大教堂与集市》；此书已由机械工业出版社于 2014 年出版，卫剑钒译。

GnuPG 简称 GPG。基于 OpenPGP 协议实现的自由的用于加密、数字签名及产生非对称密钥对的软件。——译者注）来取代流行的非自由加密软件，因为用户不应该在隐私和自由之间作出选择。

当然，写这些程序的人们变得对这项工作感兴趣，许多人为了自己的兴趣和需要给它们添加了很多功能。但那并不是这些程序存在的原因。

## 出乎意料的发展

在 GNU 工程刚刚开始的时候，我觉得我们将开发整个 GNU 系统，然后整个发行。而这并没有发生。

因为 GNU 系统的每一个组件都是在 Unix 系统中实现的，每个组件早在一个完整的 GNU 出现以前就都可以在 Unix 系统中运行。这些程序有的变得流行，而用户们开始扩充并移植它们——到各种互不兼容的 Unix 版本上，有时也会移植到其他系统。

这个过程使得这些程序更加强大，且为 GNU 工程引来了资金和贡献者。但或许也使得一个最小可用系统延迟了数年，因为 GNU 的开发者们把时间投入到维护这些移植版和为已有组件增加特性，而不是去编写一个个缺失的组件上。

## GNU Hurd

到了 1990 年，GNU 系统几乎已完成了：唯一主要的缺失部分是内核。我们已经决定将我们的内核实现为一组运行在 Mach 上的服务进程。Mach 是一个由卡内基梅隆大学，而后在犹他大学开发的微内核；GNU Hurd 是运行于 Mach 之上一组服务（正如一群牛羚——GNU）（Hurd 与 herd 谐音，而 herd 有 “群” 的意思——译者注），负责 Unix 内核的各种任务。开发的启动有所延误，因为我们要等 Mach 像它承诺的那样作为自由软件发行。

选择这种设计的一个原因是为了避免此工作中看起来似乎是最困难的一部分：在没有一个源码层调试器的条件下调试内核程序。这部分工作在 Mach 中已被完成，因此我们期待能将 Hurd 服务作为用户程序来调试，用

GDB。但这花了很长时间才做到，而互相发消息的多线程服务群实际上非常难以调试。这使得让 Hurd 能够稳定工作的进程延长了很多年。

## Alix

GNU 的内核原本并未打算叫 Hurd。它原本的名字是 Alix——以我当时的恋人命名。她，作为一个 Unix 系统管理员，指出她的名字是多么符合 Unix 系统版本的一般命名模式；作为玩笑，她跟朋友说："有人一定得用我的名字给一个内核起名。"我什么都没说，但打算用一个叫 Alix 的内核让她吃一惊。

事情并没有保持不变。Michael（现在叫 Thomas）Bushnell，内核的主要开发者，钟意 Hurd 这个名字，并重新定义了 Alix 来表示内核的某个特定部分——用于捕获系统调用并向 Hurd 服务器发消息以处理的那个部分。

后来，Alix 和我分手了，她还改了名字；与此独立地，Hurd 的设计改变了，以至于 C 库直接向服务器发消息，而这使得那个 Alix 组件从设计中消失了。

但在这些事情发生以前，她的一个朋友在 Hurd 的源码中偶然见到了 Alix 这个名字，并告诉了她。所以她确实有机会发现有一个内核以她命名。

## Linux 和 GNU/Linux

GNU Hurd 尚不适合用于生产，我们也不知道它还能不能适合。这个基于能力的设计有着直接来源于设计灵活性的问题，而且不知道有没有解决方案。

幸运的是，有另一个内核可用。在 1991 年，Linus Torvalds 开发了一个 Unix 兼容的内核并称之为 Linux。它一开始是私有的，但在 1992 年，他使其成为自由软件；整合 Linux 和尚不完整的 GNU 得到了一个完全自由的操作系统（当然，整合工作自身也很重要）。正是因为 Linux，我们今天终于能运行 GNU 系统的一个版本。

我们称这个版本的系统为 GNU/Linux，以表达他是由 GNU 系统和作

为内核的 Linux 整合而成的。不要实用主义地称整个系统为 “Linux”，因为那将我们的工作归于他人。请同等程度地提及我们[1]。

## 未来的挑战

我们已经证实了我们开发多种类型的自由软件的能力。但这并不意味着我们是不可战胜、不可阻挡的。一些挑战使得自由软件的未来变得不确定；与它们会战将需要坚实的努力和耐力，有时要奋战数年。这将会需要那种当人们珍惜他们的自由，并且不让任何人将其夺走时所显示的决心。

下边四个段落将讨论这些挑战。

### 机密硬件

硬件厂商愈发倾向于对硬件规格保密。这使得编写让 Linux 和 XFree86 能支持新硬件的自由驱动程序变得很难。现在我们有了完整的自由操作系统，但是如果不能支持明天的计算机，我们将会在明天失去它们。

有两种方法来应付这个问题。程序员可以采取逆向工程的手段来了解如何支持这些硬件。其他的人则可以选用被自由软件支持的硬件；随着我们的人数增加，规格保密将成为一个自取灭亡的策略。

逆向工程是件大工作；我们会有程序员具备足够的决心去担负这件工作吗？是的——如果我们已经建立了认为自由软件是个原则问题，而私有驱动程序不可容忍的坚定信念。我们中的大多数人会额外花钱，或甚至额外花时间，从而我们可以使用自由驱动程序吗？是的，如果拥抱自由的决心被广泛传播。

[2008 脚注：这个问题同样延伸到 BIOS. 有一个自由的 BIOS，LibreBoot[2]（coreboot 的一个发行版）；这个问题对计算机变得重要起来，因为 LibreBoot 可以不用非自由的 “Blob” 就能支持它们。]

[1] 更多信息见 “GNU/Linux FAQ”，于 http://gnu.org/gnu/gnu-linux-faq.html，和《Linux 和 GNU 操作系统》 (p. 75)。

[2] 参见 http://libreboot.org。

## 不自由的库

运行于自由操作系统上的非自由库表现得就像针对自由软件开发者的陷阱一般。库的诱人特性是诱饵；如果你用了这个库，你就掉进了陷阱，因为你的程序无法有用地成为自由操作系统的一部分（严格来说，我们可以包含你的程序，但没有了该库它就无法运行）。更糟糕的是，如果一个使用私有库的程序流行起来，它可引诱其他没有怀疑的程序员们落入陷阱。

这样的程序的第一个实例是 80 年代的 Motif 工具箱，尽管那时还没有自由操作系统，但是很显然 Motif 以后会对自由操作系统引发什么问题。GNU 工程通过两种方式回应这个问题：通过请求个别的自由软件工程在支持 Motif 的同时也支持自由的 X 部件工具箱，并请求一些人编写替代 Motif 的自由软件。该工作花费许多年时间；由匈牙利程序员们开发的 LessTif，在 1997 年才变得足够强而得以支持大多数 Motif 应用程序。

在 1996 到 1998 年间，另一个不自由的 GUI 工具箱库，叫 Qt，被用在 KDE 桌面这一包含大量自由软件的集合中。

自由的 GNU/Linux 系统不能使用 KDE，因为我们不能使用那个库。但是一些不严格坚持自由软件的 GNU/Linux 系统商业发行者将 KDE 加入到它们的系统中——而产生了有更强能力，和更少自由的系统。KDE 小组积极地鼓励更多的程序员们使用 Qt，成百万的新 “Linux 用户” 从来都不知道有这样一个问题的存在。情形相当糟糕。

自由软件社区以两种方法应对这个问题：GNOME 和 Harmony。

GNOME，GNU Network Object Model Environment（译为 “GNU 网络对象模型环境”，现在已经不再用这个说法——译者注），是 GNU 的桌面项目。从 1997 年开始，由 Miguel de Icaza 在红帽软件的支持下开发，GNOME 开始提供类似的桌面工具，但排它地只使用自由软件。它也有技术上的优势，如支持多种语言，而不仅仅是 C++。但是它的主要目的是自由：不需要使用任何非自由软件。

Harmony 是一个兼容的替代库，设计为使得无需 Qt 运行 KDE 软件成为可能。

在 1998 年 11 月，Qt 的开发者声明改动许可证，当其实施后，应该会

使得 Qt 成为自由软件。虽然没办法确信，但是我想这应该部分归功于社区对 Qt 是非自由软件时所造成的问题的坚定回应（新的许可证既不方便也不公正，所以仍旧值得去避免使用 Qt）。

[后记：2000 年 9 月，Qt 按 GNU GPL 发行，实际解决了这个问题。]

我们将如何应对下一个诱人的非自由库呢？整个社区会明白要远离陷阱吗？或者我们中的许多人将为了方便而放弃自由，从而产生一个大问题？我们的未来将取决于我们自身的哲学。

### 软件专利

我们面对的最恶劣的威胁来自软件专利，它可以对自由软件加上算法和功能的限制多达二十年。LZW 压缩算法的专利申请于 1983 年，而我们仍然无法发行能够生成适当压缩过的 GIF 的自由软件。[直到 2009 年这些专利才过期。]1998 年，由于专利诉讼威胁，一个用于生成 MP3 压缩音频的自由程序被迫从发行版中移除。

仍有办法对付专利：我们可以寻找证据以证明一个专利是无效的，也可以寻找替代方法来完成工作。但是这每一种方法只是偶尔才起作用；当它们都失败时，一个专利可能会迫使所有的自由软件都缺少某些用户想要的功能。当这种事发生时我们可以做些什么呢？

我们中因自由而重视自由软件的人们无论如何都将与自由软件共进退。我们将设法不用专利保护的功能而完成工作。但是那些认为自由软件技术出众而重视它的人们，有可能在专利抑制自由软件时认为这是自由软件的失败。因而，虽然讨论软件开发的“市集”模式的实用效力和一些自由软件的可靠性和能力是有用的，但我们决不能止步于此。我们必须探讨自由和原则。

### 自由文档

我们自由操作系统的最大不足不是在软件中——而是缺乏可以包含在我们系统中的优秀自由手册。文档资料是任何软件包的要害部分；当一个重要的自由软件包没有与优秀的自由手册一起出现，那就是一个重大缺

陷。今天我们有许多这样的缺陷。

自由文档，像自由软件一样，是自由问题，不是价格问题。自由手册的标准几乎与自由软件完全相同：它是为了给予所有用户某种自由。必须允许重新发布（包括商业销售），不论是在线还是书面形式，因而手册能够伴随每个程序的每个副本。

改动的许可同样至关重要。作为一个普遍规律，我不相信人们有必要拥有修改所有种类文章和书籍的许可。例如，我不认为你或我应该感激被给予修改像本文这样描述我们行为和我们观点的文章的许可权。

但是有一个特殊的原因说明为什么修改自由软件文档资料的自由是要紧的。当人们行使他们修改软件的权利，并且增加或改变其功能时，如果他们是尽职的，则他们也会同时修改文档和资料——因而他们能随着修改过的程序一起提供正确和可用的文档资料。一个不允许程序员们尽职并完成该工作的手册不符合我们社区的要求。

关于修改应该如何完成的一些限制并不会造成问题。例如，保持原作者的版权声明，发行条款，或作者列表的要求是正当的。要求修改后的版本包括它们是修改版本的声明也是没有问题的，即使有整个章节不能删除或修改，只要这些章节处理的是非技术主题。这些类型的限制不是问题，因为它们不阻止尽职程序员修改手册以适应修改过的程序。换种说法，它们不妨碍自由软件社区完全利用该手册。

然而，必须能够修改手册中的所有技术性内容，并将结果以所有常规介质，通过所有常规渠道分发；否则，这些限制将对社区造成阻碍，手册将变得不自由，而我们需要另一手册。

自由软件开发者们会有觉悟和决心去生产全系列的自由手册吗？再说一次，我们的未来取决于哲学。

## 我们必须谈论自由

估计当今有数千万的用户使用诸如 Debian GNU/Linux 和红帽 “Linux” 这样的 GNU/Linux 系统。自由软件已经发展到了这样实用的优势，使得用户纯粹为了实用原因而聚集到它身边。

这种现象的好结果是明显的：更多人有兴趣开发自由软件，更多用户参与到自由软件产业，以及更能鼓励公司开发商业自由软件而不是私有软件产品。

但是对软件的兴趣增长快于对其指导哲学的了解，这带来了一定麻烦。我们面对上述挑战和威胁的能力依赖于坚决主张自由的意志。为了确定我们的社区拥有这个意志，我们需要在新用户来到社区时向他们传播这样的思想。

但是我们正在这点上失败：吸引新用户加入社区的努力大大超越了教育他们成为我们社区的好公民的努力。我们需要做这两件事，而且我们也需要保持这两个努力的平衡。

## “开放源代码”

当 1998 年一部分社区决定停止使用术语 “自由软件” 并改为说 “开源软件” 时，教导新用户有关自由的观念变得更加困难。

一些喜欢该术语的人想要避免 “自由” 与 “免费” 的混淆——这个目标是正当的。其他人却打算将激励了自由软件运动和 GNU 工程的原则精神抛到一边，反而迎合行政和商业用户，而这些用户中的许多人持有一种将利润置于自由，社区和原则之上的意识形态。因而，“开源” 的花言巧语集中在制作高质量，强有力软件的潜能上，但是避开自由，社区和原则的思想。

《Linux》杂志是一个清晰的例子——其中充满了利用 GNU/Linux 运行私有软件的广告。当下一个 Motif 或 Qt 出现时，这些杂志将警告程序员们远离它还是为它登载广告呢?

商业支持能以许多方式为社区作贡献；其它种类的支持也都一样，它是有益的。但是为了赢得他们的支持而减少对自由和原则宣讲可能损失惨重；它使得前述 “软件推广和公民意识教育” 之间的失衡变得愈加糟糕。

“自由软件” 和 “开放源码” 或多或少地描述了同一个软件的类型，但对软件本身和价值观的描述是不同的。GNU 工程继续使用 “自由软件” 这个术语，来表达自由是重要的思想，而不仅仅是技术。

## 尝试!

犹达大师的名言“没有‘尝试’”听似优雅，但对我不起作用。我已经完成了我的大部分工作，与此同时担心我能不能做，且不确定如果我做了，我的努力够不够完成目标。不过无论如何我尝试了，因为在敌人和我的城池之间除了我没有任何人。令我吃惊的是，我有时会成功。

有时我失败了；我的一些城池陷落了。随后我发现另一个受到威胁的城池，并为另一场战斗做好准备。天长日久，我学会了寻找威胁并将我置于城池和威胁之间，并召集其他黑客和我联合。

今天，我不是在单打独斗。当我看见一个团的黑客挖战壕坚守战线时的感觉是一种安慰和乐趣，我意识到，目前，这个城池也许能幸存。但是危险与年俱增，并且现在 Microsoft 已经明确地将目标对准我们的社区。我们不能把自由的将来视为天命。别把它当作天命！如果你想要保持你的自由，你必须备战以保卫它。

# GNU 操作系统的初始公告

这是 GNU 工程的原始通告，由理查德 · 斯托曼（Richard Stallman）于 1983 年 9 月 27 日发表。

纵观历史，可以发现 GNU 工程在很多地方都与这份初始通告有很多差异。比如实际是拖延到了 1984 年 1 月才开始。而自由软件的很多哲学理念也是数年之后才得以厘清。

```
From mit-vax!mit-eddie!RMS@MIT-OZ
 From: `RMS%MIT-OZ@mit-eddie`
 Newsgroups: net.unix-wizards,net.usoft
 Subject: new Unix implementation
 Date: Tue, 27-Sep-83 12:35:59 EST
 Organization: MIT AI Lab, Cambridge, MA
```

> 自由的 Unix!
>
> 今年的感恩节我要去写一个完整的类 UNIX 软件系统，命名为 GNU（Gnu's Not Unix），并以自由[1]的方式开放给所有人使用。非常欢迎大家贡献时间、金钱、程序和设备来参与其中。
>
> 首先，GNU 将是一个内核加上编写和运行 C 程序所需的所有工具：编辑器，Shell，C 语言编译器，链接器，汇编器，以及一些其它的东西。在此之后，我们将添加一个文本排版工具，一个 YACC，一个帝国的游戏，电子表格，以及数百种其它的东西。我们希望最终可以提供一切通常和 UNIX 系统一起出现的有用的东西，并包括一份在线的和印刷版的文档。
>
> GNU 可以运行 UNIX 程序，但不会与 UNIX 完全相同。我们会基于在其它系统上的经验完善之以使其更舒适。特别的，我们

[1]这里用词没有注意，初衷是想说没有人需要为使用 GNU 系统而索要授权。然而这个词没有说的很清晰，结果人们经常以为获取 GNU 的副本需要很少或者免费。而这从来都不是目的。

计划使用更长的文件名，文件版本号，抗崩溃的文件系统，可能还有文件名自动补全，独立于终端的显示支持，以及一个基于 Lisp 的窗口系统，最终所有 Lisp 程序和 Unix 程序都可以共享同一个屏幕。C 语言和 Lisp 都可以作为系统编程语言。还会有基于 MIT 的 chaosnet 协议的网络软件，会远远优于 UUCP。我们还会有某些东西来兼容 UUCP。

我是谁?

我是理查德 · 斯托曼，备受模仿的原版 EMACS 编辑器的发明者，现在供职于 MIT（麻省理工大学）的人工智能实验室。我广泛从事过编辑器、编译器、调试器、命令解释器、不兼容分时系统、Lisp 机器操作系统。我率先在终端独立显示支持 ITS。此外我还实现了一个可靠的文件系统和两个 Lisp 机器的窗口系统。

我为什么必须写 GNU

我信奉的一条金科玉律就是如果我喜欢一个程序，那么我必须与其他人一起分享。我不能凭良心签署保密协议或软件许可协议。

因此我不能继续使用那些破坏我原则的电脑，因此我决定将所有自由软件组合在一起，这样我就可以不使用任何不自由的软件了。

如何贡献其中

我正在向计算机厂商索要捐献一些机器和金钱。还向一些个体索要程序和作品。

其中一家厂商已经提供了我一台机器。但我们可以用更多。如果你能捐助更多机器，那么 GNU 将会更早地在上面运行。机器最好能够在一个住宅区内进行操作，并且不需要复杂的冷却或供电。

独立程序员可以写一些 Unix 工具的兼容复制品并将其贡献给我。对大多数项目而言，如此大规模的分布式项目通过兼职很难协作，独立完成的部分可能很难组合在一起。然而对替换 Unix 的任务而言，这个问题并不存在。大多数接口规格已经通过 Unix 兼容固定下来了。如果每个贡献者的作品都可以和 Unix 的剩余部分工作在一起，那么多半一样可以和 GNU 的剩余部分一起工作。

如果我获得了捐助的金钱，我可能需要聘用一些人全职或者兼职工作。薪水可能不高，但是我希望能找到这样的人，对他们而言认识到自己工作是帮助人类和赚钱一样重要。我将其看作一种让有献身精神的人们无需按它法谋生，而将他们的全部精力投入到 GNU 的工作上的途径。

联系我以获取更多资讯。

Arpanet 邮件：

```
RMS@MIT-MC.ARPA
```

Usenet：

```
...!mit-eddie!RMS@OZ   ...!mit-vax!RMS@OZ
```

美国邮政：

```
Richard Stallman
166 Prospect St
Cambridge, MA 02139
```

# 如今自由软件更加重要



自 1983 年以来，自由软件运动一直在为计算机用户的自由而战——用户控制他们使用的软件，而不是相反。当一个程序尊重用户和社区的自由时，我们把它称为“自由软件”。

我们有时也称之为“libre software”以便澄清我们关注的是自由，而不是价格。一些专有（非自由）软件比如 Photoshop，非常昂贵；而其他比如 Flash 播放器则免费提供——所以价格并不是大问题。无论高价还是免费，它们都将程序开发者的权力凌驾于用户之上，而这种权力是不应该存在的。

同时这些软件还有个共同点：他们都是恶意软件。也就是说从功能上都是设计用来虐待用户的。今天的专有软件往往是恶意软件，因为开发者的权力腐化了它们[1]。而自由软件，用户控制着程序，既是单独用户控制同时也是群体控制。这样他们就控制着计算机做的事情（假设这些程序都是忠实完成用户的指令）。

专有软件程序控制着用户，另一些（开发者或“所有者”）则完全控制着程序。因此专有程序给了开发者凌驾用户的权力。而这本身就是不公平的，并且诱使开发者用其他方式虐待用户。

自由意味着你控制着自己的生活。如果你使用程序来打理你的生活，你的自由就取决于你如何控制着这些程序。你应该拥有对程序的控制权，更何况这些程序控制着你生活中重要的事情。

用户对程序的控制需要四项基本的自由[2]。

1. 基于任何目的，按你的意愿运行软件的自由。

[1] 参见 http://gnu.org/proprietary/proprietary.html 可知这些不断增加的威胁。

[2] 自由软件的完整定义可参见《什么是自由软件?》 (p. 1) 一文。

2. 学习软件“源代码”并修改的自由，这样可以让程序执行你想做的事情。程序是由程序员使用编程语言编写的（比如结合英语和代数），这种形式称为“源代码”。任何熟悉编程，并能以源代码形式编程的人，都可以读源代码，懂得其逻辑，并可以修改之。当你只能得到可执行格式，也就是对计算机来说能理解，但对人类极难读懂的一系列数字时，读懂并修改该形式的程序难如登天。

3. 将当前副本重新分发的自由（这不是一种义务，这样做是你的选择。如果程序是自由的，并不意味着别人有义务提供给你一份副本，或者你有义务为其他人提供副本。分发程序而没有自由，会虐待用户。然而如果不分发软件——只是私下使用——则不会虐待任何人）。

4. 随时分发你修改过的版本的自由。

前两个自由表示用户可以实际控制程序。而剩下的自由，表示任何用户团体都可以集体控制程序。具备这四个自由，用户就可以完全控制程序。这四个自由缺一不可，否则程序就是专有的（非自由的），并且不道德。

任何实用性的作品都可以适用，包括烹饪菜谱，教育作品比如教科书，参考书比如字典和百科全书，显示文章的字体，硬件设计的电路程序，3D打印用的（不只是装饰品）模型文件。因为这些并不是软件，所以自由软件运动严格上来说并不包括它们，但同理可证：这些作品也需要符合上面这四项自由。

一个自由的程序允许你按照自己的想法去改造。对于将专有软件看成密封盒子的人来说，改造软件看起来很荒谬，然而在自由的世界里则是非常普遍，并且对学习编程非常有利。传统美国消费者对汽车的改造并不顺畅，恰是因为汽车包含非自由的软件。

## 专有化的不公

如果用户没有控制程序，那么程序就在控制用户。对专有软件，有一些实体比如开发者或者程序“所有者”，控制着程序，将权力凌驾于用户之上。一个非自由的程序就像个操纵杆，是一把操纵不公权力的工具。

令人发指的是（这种情况很常见）专有程序设计用来窥视、限制、审查甚至虐待用户[1]。例如苹果的 iThings 操作系统做了所有这些，同样的基于 ARM 芯片移动设备上的 Windows 系统也是如此。Windows 手机固件以及 Google Chrome 的 Windows 版包含了通用后门，可以让一些公司不经过用户同意就远程修改程序。亚马逊的 Kindle 则通过后门删除用户的电子书。

“物联网”（internet of things）产品上使用的非自由程序会将物联网变成“骗联网”（internet of telemarketers）[2]或“窥联网”（internet of snoopers）。

为了结束非自由程序的不公，自由软件运动开发了自由的程序，这样用户可以解放自己。我们首先在 1984 年开始开发自由的操作系统 GNU。现在数以万计的计算机运行着 GNU，主要是 GNU/Linux 结合体[3]。

分发非自由的程序给用户是残害用户的行为；然而如果不分发程序则不会伤害任何人。如果你写了一个程序并私下使用，不会伤害到任何人。你也许会失去做好事的机会，但这与做错事是不一样的。因此，我们说所有程序都应该自由，意思是所有副本都应该遵循这四个自由，但并不意味着别人有义务提供给你一份副本。

## 非自由软件与 SaaSS

非自由软件是公司控制人们电脑的首选方案。今天，还有另外一种方式，称为“替代软件的服务”（SaaSS，Service as a Software Substitute）。这意思是让别人的服务器去做你自己的计算任务。

SaaSS 并不意味着运行在服务器上的程序是非自由的（虽然大多数确实是非自由的）。然而，使用 SaaSS 会导致与使用非自由程序一样的不公：殊途同归。就拿 SaaSS 翻译服务为例：用户发送文本到服务器，服务器将其翻译好发回用户（比如从英文翻译为西班牙文）。那么翻译的工作是由

[1] 参见《自由与非自由软件的分类》 (p. 80) 一文中关于专有软件的脚注 1。

[2] 参见 Marcelo Rinesi 于 2015 年 8 月 6 日发表的文章 “The Telemarketer Singularity”http://ieet.org/index.php/IEET/more/rinesi20150806

[3] 关于 GNU 操作系统的历史可参见《GNU 工程》 (p. 8) 一文，以及 “GNU/Linux FAQ” http://gnu.org/gnu/gnu-linux-faq.html。

服务器运营商控制的，而不是用户。

如果你使用 SaaSS，服务器运营商控制着你的计算过程。它需要委托所有数据到服务器运营商那里，而这些数据也可能会被迫出让给国家——毕竟谁是服务器真正服务的人[1]？

## 主要和次要的不公

当你使用专有软件或者 SaaSS，首先你对自己不好，因为这给了别的实体可以用不公的权力凌驾你之上。所以为你着想，你必须远离非自由的程序。如果你承诺不分享程序，你就在对他人不好。这个承诺本身就是罪恶，不那么罪恶的是打破承诺；最好的做法就是你应该完全不作出这样的承诺。

有一些非自由的程序会直接给用户施压。Skype 就是一个例子：当用户使用了一个非自由的 Skype 客户端软件，它会要求其他人使用同样的软件——这样两个人都将自由放弃了（Google Hangouts 也有同样的问题）。建议使用这样的软件也是同样的错误。我们必须坚决拒绝它们，即使是短时间，即使是在别人的电脑上。

另一个使用非自由程序和 SaaSS 的害处是会嘉奖肇事者，鼓励他们开发更多这样的程序或“服务”，导致更多人掉入公司的陷阱。

所有这些间接伤害发生在公共实体或学校的时候会更加显著。

## 自由软件与国家

公共机构是为人民服务的，而不是为他们自己。在计算机领域也是如此。他们有义务完全控制计算过程以确保恰当地为人民服务（也就是所谓的国家计算主权）。他们决不能让计算机的控制权落入私人之手。

为了维持对为人民服务的计算过程的控制，公共机构必须不能使用专有软件（一个由国家之外的实体掌控的软件），同时不能委托国家机构以外的实体编写或运行服务，因为这会是 SaaSS。

[1] 参见《服务器真正是在为谁服务?》 (p. 302) 一文。

专有软件在非常时期是没有安全可言的——因为无力抵抗其开发者。甚至开发者会帮助其他人攻击。微软会在修复 Windows 的 bug 之前将其展示给 NSA[1]（美国数字间谍机构）。我们不知道苹果是不是也这样做，但他们同样受到与微软一样的政府压力。如果其他国家政府使用这样的软件，会危害国家安全[2]。你会希望 NSA 攻入你自己国家政府的计算机吗?

## 自由软件与教育

学校（包括所有这类教育活动）通过他们的教学会影响社会的未来。为了做善事他们必须只教自由软件。教授专有程序会产生依赖性，这与教育的使命是相悖的。通过培训使用自由软件，学校会将社会的未来转向自由，并帮助天才的程序员掌握这门手艺。

他们同时也教育了学生协作的习惯，帮助其他人。每个班级都要有这样的规则：“对学生而言，这个班级是分享知识的地方，如果你将一个软件带来，不仅仅是你自己用，同时你必须将副本分享给班里其他人——包括源代码——以备其他人想要学习。因此，不允许将专有软件带到课堂，除非学习逆向工程。”

专有软件的开发者会惩罚那些好心分享软件的学生，并阻挠学生企图修改的好奇心。这是很烂的教育[3]。

## 自由软件：不止 “优势”

经常有人让我描述自由软件的 “优势”。然而 “优势” 这个词对自由而言太弱了。没有自由的生活是一种压迫，无论是我们平时的生活还是计算机领域。我们必须拒绝让软件或计算机服务的开发者控制我们的计算机。这才是我们要做的事情，虽然有自私的因素；但不仅仅是自私的考量。

[1]参见 Sean Gallagher 于 2013 年 6 月 14 日发表的文章 “NSA Gets Early Access to Zero-Day Data from Microsoft, Others” http://arstechnica.com/security/2013/06/nsa-gets-Eearly-access-to-zero-day-data-from-microsoft-others/

[2]关于我们建议的政策可参见《政府推动自由软件的措施》 (p. 41) 一文

[3]有关自由软件在学校的讨论可参见 http://gnu.org/education

自由包括与其他人协作的自由。拒绝人们的这项自由意味着让人孤立，会成为对人压迫的开始。在自由软件社区，我们深刻的意识到与他人协作的重要性因为我们的工作正是有组织的协作。如果一个朋友看到你在用一个程序，她也许会要一份副本。而禁止人们再分发，或者说你“不应该”这么做的程序，是反社会的。

在计算机领域，协作包括向用户再分发原始副本，也包括分发你修改过的版本。自由软件鼓励所有这些协作，而专有软件禁止这些。专有软件禁止再分发副本，并拒绝提供给用户源代码，封锁人们对软件的修改。SaaSS 也有同样的效果：如果你的计算过程是由互联网上其他人的服务器，其他人的程序副本做出的，你不能看到或碰触到这些计算用的软件，因此你就不能再分发或修改了。

## 结论

我们应该控制我们自己的计算机，如何赢回控制权？可以拒绝使用我们自己或平时所用计算机上的非自由软件，拒绝 SaaSS。对于我们这些程序员而言，可以开发自由软件[1]，还可以拒绝开发或者推广非自由软件或 SaaSS，并广泛散播这些理念给其他人[2]。

我们以及上千用户从 1984 年开始就这么做，这样才有了现在我们使用的自由的 GNU/Linux 操作系统，所有人——无论是否是程序员——都可以使用。以一个程序员或活动家的身份，加入我们的事业。让我们一起解放所有计算机用户吧。

[1] 关于许可证的建议可参见《如何为你的作品选择一份许可证》 (p. 211) 一文中的推荐许可证。

[2] 各种帮助的方式可参见 http://gnu.org/help

# 为什么学校应该只使用自由软件



教育性的活动（包括学校里的）有道义责任只教授自由软件。

所有计算机用户都应该坚持使用自由软件：这会让你获得控制你自己计算机的权利——专有软件更愿意做其拥有者或开发者想要做的事情，而不是用户想让它们做的事情。自由软件还会赋予用户自由协作的权利，这将会引领一种积极向上的生活方式。这些理由适用于学校，而且也适用于每个人。但是，本文的目的不仅限于此，还会讲到专门适用于教育领域的特殊原因。

自由软件可以帮助学校节省经费只是一个连带好处。节省经费是因为自由软件赋予学校，如同赋予其他个人用户那样，自由复制和分发软件的权利。教育系统可以给每一所学校一份副本，而且每一所学校都可以将其安装在学校的每一台电脑上，这样做无需支付任何费用。

尽管这项好处很有用，但我们坚定地拒绝将其置于首位，因为与道义层面性命攸关的重要问题相比，它显得微不足道。推动学校使用自由软件，不只是让教育变得稍微“好一点”的一个方法，也是一件让好教育替代烂教育的大事。所以，我们来探讨一下这个更深层次的问题。

学校有一项社会使命：要教导学生成为一个坚强、有才能，独立、相互协作并且自由的社会中的一名公民。学校应该像推广保育和选举一样推广自由软件的使用。通过教授自由软件，学生能成为自由的数字化时代的合格公民。这项工作能帮助社会从整体上脱离大集团公司的统治。

相比之下，如果我们教学生使用非自由软件，就等同于培养依赖性，这将违背学校的社会使命，学校应该极力避免这种事情的发生。

为什么呢？毕竟一些专有软件开发者可以免费给学校的提供非自由程序的副本。那是因为，就像烟草公司免费向学生发放香烟一样，他们希望

利用学校给学生灌输这种对他们产品的依赖性[1]。

当学生毕业之后，他们就会终止这种免费行为，而且，他们也不会为毕业生就职的公司或机构提供免费服务。你一旦对这些软件形成了依赖，你很可能就会付费，并且之后的升级可能会很昂贵。

自由软件可以让学生了解和掌握这些软件的工作原理。有些学生，或者那些天才程序员，青少年时代就对计算机和软件充满好奇心，他们急切地想要获知他们想要知道的一切。他们非常渴望阅读那些他们每天都在使用的软件的源代码。

专有软件阻抑了学生对知识的渴望：他们被告知，“你想要获得的知识是一个秘密，禁止学习！”专有软件是教育精神的敌人，所以，除非作为逆向工程的目的，学校应该拒绝使用它们。

自由软件鼓励每个人学习。自由软件社区反对“高高在上的技术”——这会让大众对技术的基本原理敬而远之。我们鼓励任何年龄层次学生或个人阅读源代码，而且，我们希望他们学得越多越好。

使用自由软件的学校将会成为那些喜爱编程学生的乐土。你知道喜欢编程的学生是怎样成为优秀的程序员的吗？他们必须要阅读和理解人们真实使用的软件的源代码。通过阅读和编写大量的代码，你才有可能写出高质量清晰的计算机程序。目前来看，只有自由软件允许你这样做。

如何学习为大型软件项目编写代码？最好的办法就是为现有的大型项目编写大量代码。自由软件鼓励你这样做，但专有软件会禁止如此。任何学校都可以给学生提供这种掌握编程技术的机会，但只有使用自由软件的学校才真正有这个可能。

学校使用自由软件最深层次原因是可以进行德育教育。我们希望学校教给学生基本的知识和有用的技巧，但是这只是学校工作的一部分。学校最根本的一项职能，就是培养好公民——其中包括助人为乐的习惯。在计算机领域，这就意味着我们需要教授学生分享软件的精神。从幼儿园开始，

[1]参见《雷诺烟草公司因向儿童发放卷烟样品，于 2002 年被罚 1500 万美元》（RJ Reynolds Tobacco Company was fined $15m in 2002 for handing out free samples of cigarettes at events attended by children）http://bbc.co.uk/worldservice/sci_tech/features/health/tobaccotrial/usa.htm

学校就应该告诉学生，“如果你把软件带到学校来，你应该与其他同学一起分享。一旦有人想要学习一下它的工作原理，你就应该把源代码展示给大家。所以说，非自由软件最好不要带到学校来，除非出于反向工程的目的。”

当然，学校必须履行其诺言：在课堂上只允许自由软件出现（除非为了逆向工程的目的），在分享软件的同时，源代码也应该一并分享，这样的话，学生们就能复制它们，把它们带回家，甚至还可以将它们再次分享给其他人。

教授学生使用自由软件，让他们参与自由软件社区，就是一门生动鲜活的公民教育课程。这门课程教授将会把学生培养成为具有公共服务精神的行为模范，而不是垄断企业的超级大亨。所有学校都应该使用自由软件。

如果你和学校有一定联系——如果你是一名学生、一名教师、一位员工、一名管理者、一位捐赠人，或者这些人的父母——你有义务和责任向学校推广自由软件。如果私人请求无法解决这个问题，可将这个问题公开出来提交到社区。这样就能让更多的人意识到这件事情的重要性。你可以寻找同盟，让大家一起发起这项推广自由软件的运动。

# 政府推动自由软件的措施



本文提出了一些推动自由软件在国内发展的强力政策，并引领国内其他方面转向软件自由。

国家的使命是通过整合社会资源为人民自由谋福祉。这个使命在计算机领域的一个方面就是鼓励用户使用自由软件：代表用户自由的软件[1]。而专有（非自由）软件则践踏了人民的自由；这是一个国家应该努力去根除的社会问题。

从国家的计算主权角度来说，国家需要坚持自由软件的主张（国家控制其计算过程）。所有用户理所应当控制自己的计算过程，但国家有代表人民对计算过程维持控制的责任。现在大多数政府活动都依赖计算过程，而对这些活动的控制则依赖于对计算过程的控制。而在任务很关键的机构里失去这种控制将会极大危害国家安全。

将国家机构转向自由软件同时也会提供额外的好处，比如节省经费和鼓励本地的软件支持行业等等。

本文中，“国家实体”指代各级政府，以及公共机构包括学校，公共—私立伙伴关系，主要由国家出资的活动比如特许公立学校，以及由国家控制或由国家特许经营或注资的“私营”企业。

## 教育领域

教育领域的政策非常重要，因为会影响国家的未来

- 只能教授自由软件

  教育活动或者至少这些国立教育实体，必须只教授自由软件（因此，他们决不能引导学生使用非自由程序），并教育学生坚持使用自由软

[1] 参见《什么是自由软件?》 (p. 1) 一文了解自由软件的定义

件背后的公民原因。教授非自由程序就是在教授依赖性，这与学校的使命是相悖的。

## 国家与公众

这些政策对个人与组织使用软件的影响也很巨大：

- **永远不要求非自由程序**

  法律和公共部门实践活动必须改革永远不能要求或施压给个人或组织使用非自由程序。他们还必须不能在通信或出版行业引入类似的规定（包括数字限制管理 Digital Restrictions Management[1]）。

- **只散播自由软件**

  国家实体无论在什么时候将软件公开散播，包括网页中包含或附带的程序，必须作为自由软件散播，且必须与只运行自由软件的平台兼容。

- **政府网站**

  政府实体的网站和网络服务必须设计成用户能够只用自由软件而没有障碍（相对专有软件——译者注）地使用。

- **自由的格式和通信协议**

  国家实体必须只使用自由软件广泛支持的文件格式和通信协议，特别是已经公开的规格（我们并不说 “标准”，因为它应该适用于标准和非标准的接口）。例如，他们必须不能散播使用 Flash 或者非自由解码器播放的音频或视频文件，公立图书馆必须不能包含数字限制管理。

- **解除对计算机的限制性许可证**

[1] 参见反 DRM 活动网站 http://defectivebydesign.org/what_is_drm 以及《应避免使用（或慎用）的词语》一文中有关 DRM 的章节 (p. 115) 了解相关问题。

销售计算机必须不能要求购买一份专有软件许可证。必须通过法律来约束销售方提供给消费者购买计算机时不需要专有软件以及不需要为此付费的选项。强制付费是另一个错误，不能让我们从专有软件的不公中分散注意力，应更关注失去自由而导致的损失。虽然如此，强迫用户为其付费等于是给了特定专有软件开发者一个额外的不平等优势——让他们来决定用户的自由。国家应该防止这种情况发生。

## 国家计算主权

很多政策会影响到国家计算主权。国家实体必须维持对它们的计算资源的控制，而不是放弃控制而委之于私人之手。这一点适用于所有计算机，包括智能手机。

- 迁移到自由软件

  国家实体必须迁移到自由软件，并且不能安装和继续使用非自由软件，除非临时例外。只有一个机构可以有权授予这些临时例外，且需要令人信服的理由。这个机构的目标是将这些例外必须减少到零。

- 开发自由的 IT 解决方案

  当政府实体需要为开发计算机解决方案付费时，合同必须要求以自由软件形式交付，并且方案必须设计为可以在百分之百自由软件环境中运行和开发。所有合同都需要这条，这样如果开发商一旦不符合需求，作品就不给报酬。

- 为自由软件选配计算机

  当政府实体购买或租借计算机时，必须在其等级中选择几乎无需专有软件即可运行的型号。政府需要为每个等级的计算机维护一个按此标准授过权的型号的列表。对于公开型号和政府采购型号，应该选择只对政府开放的型号。

- 与厂商协商

国家必须经常与厂商协商市场（对外销售或/和面向政府）上适合的硬件产品，在所有相关产品领域，都不能需要专有软件。

- **与其他国家联合**

  本国需邀请其他国家来和厂商一同协商适合的硬件产品。联合它们获得更多影响力。

## 国家计算主权之二

国家计算主权（和安全）包括控制计算机做国家的事情。这需要 (1) 避免 “替代软件的服务”（Service as a Software Substitute）[1]——除非这类服务是运行在国家机关的同一分支机构，以及 (2) 其他降低国家对计算资源控制的实际行动。因此：

- **国家必须控制其计算机**

  每一台国家使用的计算机必须从属于政府的同一个要使用它们的分支机构或由此机构租借，而这个政府分支机构必须不能把决定谁有权物理访问计算机，谁能维护（软件和硬件），或者能够安装什么软件的权力放弃给外人。如果计算机不可移动，那么计算机所在的物理空间必须是国家所有（无论是所有者或者承租人）。

## 对开发的影响

国家政策会影响自由和非自由软件的开发：

- **鼓励自由**

  国家需要鼓励开发者创立或完善自由软件并将其公开，比如通过减税或其他财政补贴的方式。相反，对非自由软件的开发者则没有此类财政补贴。

[1] 参见《服务器真正是在为谁服务?》 (p. 302) 一文的相关章节，了解 SaaSS。

- **不要鼓励非自由**

  特别的，专有软件开发者不应该“捐献”软件副本给学校，并根据软件的售价申请减税优惠。专有软件在学校是不合法的。

## 电子垃圾

自由不应意味着产生更多电子垃圾：

- **可替代性软件**

  很多现在计算机设计为无法用自由软件替代其预装的软件。因此唯一解放它们的方式就是丢掉，而这对社会不利。

  因此，对于新（即，非二手）计算机或基于计算机的产品，阻碍用户开发、安装或阻碍用户替代任何内部能由制造方升级的预装软件的，硬件接口保密或有意设限的，大量销售、进口或分发的行为都应该是非法的，或者至少课以重税来劝退。这特别适用于任何需要“越狱”方可在其上安装不同操作系统的设备，或包含保密接口外设的设备。

## 技术中立

通过本文中给出的措施，政府可以恢复对其计算过程的控制，并引导国家公民、商业和组织控制自己的计算过程。然而，很多反对意见认为这会违反技术中立的“原则”。

技术中立是指国家不应该在技术选型时强加干涉。无论这是不是有效的原则或是争议，这仅仅只涉及到技术问题。而这里提出来的措施解决的是道义、社会和政治问题，因此超出了技术中立的范畴[1]。除非他们觉得其政府需要在主权和公民自由上保持“中立”。

[1]参见我的文章“Technological Neutrality and Free Software” http://www.gnu.org/philosophy/technological-neutrality.html

# 为什么自由软件需要自由的文档



自由的操作系统最大的不足并不在于软件——而是缺乏可以收录其中的优秀自由手册。我们很多重要的程序还没有完善的手册。文档是任何软件包的重要组成部分；当一个重要的自由软件包发布，却没有附带自由的手册，这会是一个严重的缺陷。而这样的缺陷现在有很多。

曾几何时，多年以前，我想学习一下 Perl。并得到了一份自由手册，但我发现很难读懂。当我寻问 Perl 用户有没有替代品，他们告诉我，有更好的入门手册，但那些不是自由的。

为什么会如此？作者为 O'Reilly 协会编写了优秀的手册，而 O'Reilly 附加了限制条款出版这些手册——不能复制、不能修改、不能获得源文件——这使得它们被排除在自由软件社区之外。

这类事情绝不是第一次发生，而且（对我们社区是巨大的损失）远非最后一次。专有手册出版商诱使许多作者限制他们的手册。很多次我听到 GNU 用户急切地告诉我他正在写一本手册，他希望以此来帮助 GNU 计划，但是随后我的希望破灭了，因为他接着解释说，他已经与出版商签署了一项限制它的合同，所以我们不能使用它。

鉴于编写好的英文手册是程序员中难得的技巧，因此我们就更承担不起因此而失去手册。

和自由软件一样，自由文档是一个关乎自由的问题，并非价格。这些手册相关的问题不是因为 O'Reilly 对印本收取了费用——这本身不是问题（自由软件基金会也销售自由的 GNU 手册印刷版[1]）。但 GNU 手册能以源代码形式获得，而这些（专有）手册只能以纸张的形式获得。GNU 手册允许复制和修改；Perl 的手册则不可以，这些限制就是问题所在。

自由手册的规范与自由软件的规范大致一样：它给所有的用户一定的自由。必须允许重新发布（包括重新商业发布），因此手册可以随着每一份程序，发布到网络上或发行印刷版。允许修改手册也是十分关键的。

[1] 参见 http://shop.fsf.org/category/books/ 和 http://gnu.org/doc/doc.html

作为一般准则，我不认为允许人们修改各种文章和书籍是必要的。写作问题不一定和软件问题相同。例如，我不认为你或我有权利去修改像这样一篇表明我们行为和看法的文章。

不过，自由软件文档的修改对自由很重要，这里有个特殊的原因。当人们行使他们修改软件的权力的时候，添加或是修改软件功能时，若有责任心的人，他们也会同时修改软件的手册——以便为修改后的软件提供准确可用的文档。一本手册若让编程人员不能尽责完成他的工作，或更确切地说要求程序员修改程序之后重新写一份手册，是不能满足我们社区的需求的。

尽管全面禁止修改是不可接受的，但一些受限的修改方法还是可以接受的。例如，要求保留原作者的版权声明，发布条款或是作者名单，上述这些都是可以的。要告知版本已经被修改，甚至不许删除或修改整个章节，只要这些章节处理的是一些非技术性话题，像这些要求都没问题（一些 GNU 手册就是如此）。

这类限制不是问题所在，因为作为一个实际问题，他们没有阻挡有责任心的程序员去修改手册以适应修改过的程序。换句话说，他们没有阻止自由软件社区充分利用该手册。

然而，它必须能修改手册所有的技术内容，且随后通过所有常规渠道，在所有常规媒介中发布结果。否则，这些限制一定会阻碍社区，将手册变得不自由，而因此我们将会需要另一本手册。

很不幸的是，在专有手册存在的情况下往往很难发现有人写了另一本手册。问题在于很多用户认为专有手册已经足够好，因此他们没有意识到撰写自由手册的必要性。他们没有意识到自由操作系统还有缺陷仍需完善。

为什么用户认为专有手册足够好了呢？因为有些人不考虑这个问题。我希望本文将为改变这种状况做点贡献。

其他用户认为，专有手册可以被接受和许多人认为专有软件可以被接受有着同样的原因：他们纯粹是以实际需求来评判，而不将自由作为一项判断依据。这些人有权表达观点，但由于这些观点滋生于不自由的价值观，

所以无法指导我们这些切实重视自由的人。

请将这个问题公之于众。我们仍然认为需要减少专有手册的出版量。如果我们传播专有手册不够好的言论，也许下一个愿意帮助 GNU 写文档的人将会在不是太晚的时候认识到，他必须首先使它自由。

我们也可以鼓励商业出版商销售自由、Copyleft 的手册以代替专有手册[1]。而帮助这项事业的方法是在购买之前检查手册的分发条款，尽量购买 copyleft 的手册，而不是非 copyleft 的手册。

[1]其他出版商的自由图书可见此列表：http://gnu.org/doc/other-free-books.html

# 售卖自由软件



很多人认为 GNU 工程的精神就是不该对软件发行的副本收费，或者应只收取很少的钱——只要是成本价即可。这是一种误解。

事实上，我们鼓励重新发布自由软件[1]的人尽可能多的收取他们想要的费用。如果一个许可不允许用户销售生成的副本，它就是个非自由许可。如果这让你感到惊讶，请继续往下读。

“free” 这个词有两种合理通用的解释，它可以被解释为自由亦或是免费价格。当我们谈到 “free software” 时，我们所指的是自由，而不是价格。（可以理解为 “言论自由”，而不是 “免费啤酒”）。明确地讲，自由软件的意思就是任何用户可以自由的运行、修改和免费或收费地重新发布修改过或未修改的程序。

自由的程序有时是免费发布的，而有时则需要收费。有时同一个程序可以在不同的地方分别以这两种方式发布。无论其价格如何，这种程序都是自由的，因为用户在使用时是自由的。

非自由程序[2]通常以高价出售，但有时销售商会赠送你一份免费副本。尽管如此，那也不能使其成为自由软件。无论收费与否，这个程序都是不自由的，因为用户没有自由。

因为自由软件无关价格，所以低价并不意味着更多自由，或者更加接近自由。因此如果你重新发布自由软件的副本，你也可以收取基本的费用来**获得收益**。重新发布自由软件是一种很好而且合法的行为；如果你那样做，你同样也可能从中获利。

自由软件是一项社区工程，每个依赖社区的人都应该尽其可能为社区建设做出贡献。对一个发布者来说，这样做可以使一部分收益流向自由软件开发项目或者自由软件基金会。只有这样才能推动自由软件在世界的发

[1] 参见《什么是自由软件?》 (p. 1) 一文

[2] 也称为 “专有软件”，参见《自由与非自由软件的分类》的相关章节 (p. 87) 一文

展。

**发布自由软件是筹集资金的一个机会。不要浪费这个机会！**

为了捐献金钱，你需要收取一些额外的费用。如果你收费过低，那就不能为支持自由软件发展而储备资金。

## 过高的发行费用会伤害一些用户吗？

有时人们担心过高的发行费用会使自由软件超出低收入用户的预算。对于专有软件，高昂的费用确实如此——但自由软件与之不同。

不同之处在于自由软件天生就易于传播，可以通过多种方式来获取。

在你没有支付标准价格之前，软件囤积者尽他们该死的最大努力阻止你使用专有软件却不按标价付钱。如果这个定价很高，的确会让一些用户很难使用这个软件。

对于自由软件，用户不必须为了使用软件而支付发行费用。可以从拥有一份副本的朋友那里复制程序，或者请能上网的朋友下载程序。或者很多用户可以一起分摊费用买一张 CD-ROM，然后轮流安装软件。当软件自由时，高价的 CD-ROM 就不再是主要障碍。

## 高昂的发行费用不利于自由软件的使用吗？

另一个经常担心的问题是自由软件的传播。人们认为高昂的发行费用会减少用户数量，或者低廉的价格可能会鼓励用户去使用。

对于专有软件确实是这样——但是自由软件与之不同。可以通过如此多的方式来获得软件副本，因此发行服务的费用对自由软件的普及影响很小。

从长远来看，使用自由软件的用户数量主要取决于自由软件可以做什么，以及自由软件有多好用。很多用户并不把自由放在首位；如果自由软件不能做他们要做的所有工作，许多用户将继续使用专有软件。因此，如果我们想从长远上来增加自由软件的用户数量，我们的首要任务是开发更多的自由软件。

最直接的方式就是自己写一些需要的自由软件[1]或手册[2]。不过如果你是发行方而不是编写软件，你提供帮助的最好方式就是提供资金给其他人来写自由软件和手册。

## “售卖软件”这个说法也很迷惑

严格来说，“售卖”是指通过货物交易来获取金钱。销售一份自由软件的副本是合法的，而且我们鼓励这种行为。

然而，当人们想到“售卖软件”时[3]，他们通常想象成大多数公司的做法：使软件专有化，而不是自由。

因此，除非你打算仔细分析甄别，正如这篇文章所言，我们建议最好避免使用“售卖软件”这个说法，而选用其他说法来代替。例如，你可以说“为发行自由软件收费”——这样就不会有歧义了。

## 收费的高低与 GNU GPL

除非一种特殊情况，GNU 通用公共许可证（GNU GPL）对于你发布自由软件的副本时收取多少费用并没有要求。你可以免费，一美分，一美元，或者十亿美元。这完全取决于你和市场，所以如果没有人愿意花十亿美元买你的软件请不要向我们抱怨。

唯一例外的情况是只发布二进制（的格式）而没有（发布）相对应的完整源代码。这样做的人，GNU GPL 会要求其为后期需求提供源代码。如果对于源代码没有价格上的限制[4]，他们可以确定一个没有人能支付得起的庞大数目——比如十亿美元——并且假装发布源代码，但本质是在隐藏源代码。因此，在这种情况下我们必须限制源码价格，以确保用户的自由。一般情况下，我们没有理由限制发行费用，所以我们不限制他们。

[1]参见 Savannah 任务列表 http://savannah.gnu.org/projects/tasklist

[2]参见 http://gnu.org/doc/doc.html

[3]参考《应避免使用（或慎用）的词语》一文的“贩卖软件（Sell Software）”一节 (p. 124) 一文了解为何“售卖软件”这个说法有歧义

[4]参见《GNU 通用公共许可证》一文的第六段 (p. 234)

有时公司的活动超出了 GNU GPL 所许可的范围，他们辩解说 “没有对 GNU 软件收费” 或者其他类似的理由。无论怎么样，他们都没有因此得逞。自由软件是关于自由的，并且遵循 GPL 就是捍卫自由。当我们维护用户自由时，我们没有因为思考需要收取多少发布费用而心烦意乱。自由才是问题所在，是全部的问题，也是唯一的问题。

# 自由硬件和自由硬件设计

本文分成两部分，分别以标题“为什么我们需要自由的电子硬件设计”http://wired.com/2015/03/need-free-digital-hardware-designs 和“硬件设计必须自由，如何行动见此”http://wired.com/2015/03/richard-stallman-how-to-make-hardware-designs-free，发表于《连线》杂志网站。2015 年本文发布于 http://gnu.org。

自由软件的思想向硬件延伸到什么范围？让我们的硬件设计自由是不是个道德义务，正如使我们的软件自由那样？维持我们的自由是否需要抵制出自非自由设计的硬件？

## 定义

*自由软件*事关自由，而非价格；宽泛地讲，这意味着用户有使用软件的自由和复制并再分发软件的自由，免费或收费。更精确地，其定义可用这四项基本自由[1]公式化的来考察。为强调“free”一词指的是自由而非价格，我们经常把法语和西班牙语中的“libre”一词和“free”一起使用。

将同样的概念直接应用于硬件，*自由硬件*意为用户有使用、复制和免费或收费地再分发硬件的自由。然而，除了钥匙、DNA、和塑料制品的外观之外，没有用于硬件的复制机制。大多数硬件是按照一定的设计装配出来。设计先于硬件存在。

因此，我们实际需要的概念是类似*自由硬件设计*的东西。很简单：其意指一种允许用户使用（即，在其指导下装配硬件）和复制并免费或收费地再分发的设计方案。这种设计方案必须提供和自由软件一致的四项基本自由。

这样我们可以将按照自由的设计装配出的硬件说成“自由硬件”，或“设计自由的硬件”以避免可能的误解。

[1]参见《什么是自由软件?》 (p. 1) 以查看四项基本自由的列表。

首次接触自由软件思想的人们经常认为其含义是你可以免费获得一份副本。很多自由程序可以不要钱地获取，因为下载你自己的副本无需代价，但那不是 “free” 在此处的含义（事实上，有些间谍软件程序——例如 Flash 播放器和《愤怒的小鸟》是免费的，尽管它们不自由）。将 “libre” 一词和 “free” 一起使用有助于澄清观点[1]。

对硬件，混淆趋于走向另一方向；硬件要花钱生产，故商业化生产硬件无法免费（除非是赔本赚吆喝或添头），但这并不妨碍其设计方案变得自由。由你的 3D 打印机生产出的东西可以非常便宜，但并非精确地免费因为你需要支付原料成本。在道德方面，自由的问题完全压倒了价格问题，因为对其用户拒绝自由的设备还不如没有。

由某些人使用的 “开放硬件” 和 “开放源码硬件” 的说法具有和 “自由硬件” 相同的具体含义，但这些说法降低了对自由议题的重视。它们由 “开放源码软件” 的说法派生而来，其多少指的是自由软件，但缺少了对自由的讨论或将自由的议题展示为关乎对错的事[2]。为着重强调自由的重要性，一旦关乎自由我们就要注意谈到自由；因为 “开放” 做不到这一点，让我们不要用它来取代 “自由” 一词。

## 硬件和软件

硬件和软件在根本上不同。一个程序，即使取编译后的可执行形式，也是一组可被计算机解释为指令的数据集合。和任何其他数字作品相同，它可以用计算机来复制和改写。程序的副本没有固有的物理形式或化身。

与之相对，硬件是一个物理构造且其物质性是关键。虽然硬件的设计可以表示为数据，甚至在某些场合下表示为程序，设计仍然不是硬件本身。为 CPU 做的设计不能执行程序。你基本上不可能试着用为键盘做的设计打字或在为显示屏做的设计上显示像素。

更进一步，虽然你可以用计算机来修改或复制一份硬件设计，计算机

[1]参 见 http://gnu.org/philosophy/proprietary/proprietary-surveillance.html 以查看一个在工业中扩散的监视手段的快速增长的列表。

[2]参见《为什么说开源漏掉了自由软件的要点》 (p. 90) 一文

并不能将这份设计转化为其描述的物理结构。那需要装配设备。

## 硬件和软件的边界

在数字设备中，硬件和软件的边界是什么？它遵从其定义。软件是计算机可以操作修改和复制的部分，而硬件则不可以。这是理清其区别的正确方法，因为关乎实际的后果。

在硬件和软件之间有一个包含可被更新或替换，却并未打算在产品售出后更新或替换的灰色地带——固件（Firmware）。从概念上说，这个灰色地带十分狭窄。在实践上，它很重要，因为很多产品落在其中。我们可以把这样的固件看作硬件的小小延伸。

有人会说原装的固件程序和场效应可编程逻辑门阵列（FPGA）“模糊了硬件和软件的边界”，但我认为这是对事实的误读。在使用过程中安装的固件是软件；设备自带并不可修改的固件虽天生是软件，但我们可以把它当成一种电路。如对 FPGA，FPGA 本身是硬件，但加载到 FPGA 中的逻辑门模式是一种固件。

在 FPGA 上运行自由的逻辑门模式可以是一种使得数字设备从电路层变自由的潜在有效方法。然而为了使 FPGA 在自由世界中可用，我们需要为其设计自由的开发工具。其障碍是加载进 FPGA 的逻辑门模式文件的格式是私密的。多年以来，没有一种 FPGA 的型号能够让我们不使用不自由（私有）的工具就能够为其生成逻辑门模式文件。

到了 2015 年，用来给一种通用的 FPGA 型号——Lattice iCE40[1]用硬件描述语言（HDL）写成的输入文件编程的自由软件工具出现了。现在也可以用自由工具编译 C 程序并在 Xilinx Spartan 6 LX9 FPGA 上运行它们[2]，但这些工具不支持 HDL 输入。我们建议您抵制其他型号的 FPGA 直到它们也能被自由的工具支持。

对 HDL 代码本身而言，它可以表现为软件（当运行于模拟器上或载入 FPGA 中），也可以表现为硬件设计（当实现为不可变的硅晶或电路板）。

[1]参见 http://clifford.at/icestorm/.

[2]参见 https://github.com/Wolfgang-Spraul/fpgatools

## 3D 打印机的道德问题

道德上，软件必须自由[1]；不自由的程序是不义的。我们是否应当对硬件设计取同样的观点？

我们肯定应该，在 3D 打印（或者，更普遍地，任何个人装配行为）可处理的领域。用于制造有用的、实用的物品（即，功能性而非装饰性）的打印模式必须是自由的，因为它们是为了实用而产生的作品。用户应得到对这些作品的控制，正如他们应得对他们使用的软件的控制。发布一个不自由的功能性物品的设计和发布非自由程序一样有错。

请仔细甄别，并选择那些只用自由软件就能驱动的 3D 打印机；自由软件基金会为这样的打印机背书[2]。有的 3D 打印机是按照自由的硬件设计制成的，但 MakerBot 的硬件设计是不自由的[3]。

## 我们是否必须抵制不自由的数字硬件？

不自由的数字硬件[4]设计是否不义？我们是否必须为了我们的自由抵制所有按照不自由的设计制造出的数字硬件，如同我们必须抵制非自由软件？

因为硬件设计和软件源码在概念上是平行的，许多硬件黑客像对非自由软件那样迅速声讨非自由硬件设计。我对此并不同意，因为硬件和软件的情况不同。

今天装配芯片和电路板的技术类似印刷业：它适于工厂中的大规模生产。比起今天的软件复制，它更类似 20 世纪 50 年代的书籍复印。

复制和修改软件的自由在道德上是必要的，因为那些活动对使用软件的人们来说是可行的：你能使用软件的设备（计算机）同样适合用来复制

[1] 参见《如今自由软件更加重要》 (p. 32) 一文

[2] 参见 http://fsf.org/resources/hw/endorsement

[3] Rich Brown, “Pulling Back from Open Source Hardware, MakerBot Angers Some Adherents,” 27 September 2012, http://cnet.com/news/pulling-back-from-open-source-hardware-makerbot-angers-some-adherents/.

[4] 如本文中的用法，“数字硬件” 包括在数字部分之外还有模拟部分的硬件。

和改写它。今天的移动计算机太弱小而难以适用于此，但任何人都能找到足够有力的计算机。

而且，计算机足以用来下载并运行那些懂得如何修改软件的人修改的版本，即使你不是程序员。当然，非程序员每天都下载并运行软件。这就是为什么自由软件对非程序员有实在的贡献。

这些在多大程度上适用于硬件？并非使用数字硬件的每个人都知道如何修改电路设计，或芯片设计，但任何有 PC 的人都有这么做的必要设备。到目前为止，硬件平行于软件，但接下来区别就大了。

你不能在你的计算机上运行电路设计或芯片设计。构造大型电路是一系列艰苦的工作，之后你才能得到电路板。装配芯片在今天并不适合个人完成；只有规模化生产才能让它们足够便宜。以今天的硬件技术，用户们无法像运行软件黑客某甲（原文是 John S Hacker——译者注）改造过的软件那样，下载并运行硬件黑客某乙（原文是 John H Hacker——译者注）改造过的硬件设计。因此，四项基本自由无法像给与用户们对程序的集体控制那样给与对硬件设计的集体控制。这就是一切软件都必须自由的原因无法适用于硬件的地方。

1983 年没有自由的操作系统，但显然如果有了，我们可以立刻用上它并拥有软件自由。缺少的只是这样一个操作系统的代码。

而到了 2014 年，就算我们有了适用于 PC 的 CPU 芯片的自由设计，按此设计大规模生产出来的芯片也不能为我们在硬件界给与我们同样的自由。如果我们要购买从工厂大规模生产出来的产品，对工厂的依赖会引发大多数和非自由设计相同的问题。我们需要未来更好的装配技术，这样才能让我们自由的设计得到硬件的自由。

可以展望这样的未来，我们的个人生产者能制造芯片，机器人可以把芯片和变压器、开关、按键、屏幕、风扇等组合焊接在一起。在这样的未来里，我们都可以制造自己的计算机（还有装配机和机器人），而且我们每个人都能从懂硬件的人做出的改造设计中得到好处。那时抵制非自由软件的议题才也能适用于硬件设计。

这个未来距今至少有数年之久。在此期间，原则上没有必要抵制非自

由设计的硬件。

## 我们需要自由的硬件设计

尽管在今天的状况下我们无需抵制按照非自由设计制造出的数字硬件，不过我们仍需要开发自由的设计并在条件允许时使用它们。这些已经在今天提供了优势，而且在未来它们可能是使用自由软件的唯一方法。

自由的硬件设计提供了实用上的优势。很多公司可以装配，这减少了对单一卖家的依赖。集团可以安排大量装配。拥有电路图或 HDL 代码使得学习设计以寻找错误或恶意功能成为可能（NSA 在一些计算机中促成恶意弱点之事已被人所知）。并且，自由设计可作为设计计算机和其他复杂设备的构造砖块，其规格会被公开，对我们不利的部件也会更少。

在我们可以这样制造整台计算机之前，自由硬件设计可以用于计算机和网络的某些部件，还有嵌入式系统。

如果它成为阻止非自由软件的唯一方法，自由硬件设计甚至可能会在我们能以个人之力装配硬件以前就变得至关重要。因为一般的商业硬件正越发设计得使用户屈服，由于私密的规格，甚至需要你以外的某人签名代码，它们越发不兼容于自由软件。甚至蜂窝电话的调制解调器芯片和一些图形加速器已经需要制造商为固件签名。任何在你计算机上允许别人改但不许你改的程序，都是强加于你不义强权的体现；强加了那种要求的硬件是恶意硬件。所以蜂窝电话调制解调器芯片，现在可用的所有型号都是恶意的。

某一天，自由设计的数字硬件可能是唯一可以让自由系统运行的平台。让我们专注于在那一天之前拥有必要的自由硬件设计，并希望我们有方法为所有用户将它们足够便宜地装配出来。

如果你设计硬件，请让你的设计变得自由。如果你使用硬件，请催促公司们并向其施压，让他们使硬件设计变得自由。

## 设计的层次

软件有实现的层次；例如，一个软件包可能包括库、命令和脚本。但这些层次对软件的自由没有明显的贡献，因为可以把所有这些层次都变得自由。设计程序组件和设计把组件组合在一起的代码是同类的工作；与其类似，从源代码编译程序组件和编译把这些组件组合在一起的程序是同类的操作。使整个系统自由只需继续工作直到完成整件任务。

因此，我们坚持主张程序要在全部层次上自由。对一个分类为自由软件的程序，组成它的每一行源代码都得是自由的，这样你才能只用自由的代码重建这个程序。

与此相反，物理对象经常是由不同类型的工厂设计制造的组件所构造而成的。例如，计算机由芯片构成，但设计（或装配）芯片和从芯片设计（或装配）计算机是非常不同的。

因此，我们需要区分数字产品（可能还有其他类型的产品）的设计中的*层次*。连接芯片的电路是一个层次；每个芯片的设计是另一层次。在 FPGA 中，基本单元间的互联是一个层次，而基本单元本身是另一层次。理想的未来中我们会希望设计在所有层次上都是自由的。在当前状况下，仅仅使一个层次变得自由已经是一个明显的进步。

然而，如果一个设计在同一层次上组合了自由和非自由的部分——例如，一个结合了私有 “软核” 的 “自由” HDL 电路——我们必须下结论说此设计作为一个整体在那个层次上是不自由的。类似地，对非自由的 “向导” 或 “宏”，如果它们指的是芯片间的互联或芯片内部结构间的可编程互联的一部分。其自由部分可以是通向自由设计未来目标的一个步骤，但到达那一目标需要把不自由的部分替换掉。它们在自由的世界中决不能接受。

## 自由硬件设计的许可和版权

你通过将一个硬件设计按照自由的许可发行来使其成为自由设计。我们建议您使用第三版或更新的 GNU 通用公共许可证。我们带着这种用途的视点设计了 GPL 第三版。

在电路和非装饰性的物体外观上的左版（Copyleft），不会像想象一般获得成功。这些设计的版权仅适用于这些设计是如何被画出或写下的。左版是一种利用版权法的手段，所以其效果仅限于版权法起效的范围。

例如，一个电路，作为一种拓扑关系，无法拥有版权（也因此无法使用左版）。由 HDL 写成的电路定义可以拥有版权（因此可以使用左版），但左版只能保护这 HDL 代码的具体表达方式，而非依其生成的电路拓扑。与此类似，电路的图纸或布局可以拥有版权，因此可以使用左版，但仅仅保护图纸或布局本身，而非电路拓扑。任何人都可以以不同的外观画出同样的电路拓扑，或写出生成相同电路的不同 HDL 定义。

版权不能保护物理电路，所以当人们构建电路的实例时，设计的许可证对他们能构建出的设备做什么没有法律效力。

对物件的图纸而言和 3D 打印模型而言，版权不能阻止以不同的方式绘制同样功能性物体的外观，它也不能保护按照图纸制造功能性物质的实体。只要版权考虑到，每个人都可以自由地制造和使用它们（而这是我们十分需要的自由）。在美国，版权不涉及某个设计描述的功能特性[1]，但却涉及装饰特性。当一个物体既有装饰特性又有功能特性时，你将进入一个棘手的境地[2]。这一切也许在你的国家也是如此，也许不是。在商业性或大规模生产之前，你需要咨询当地的律师。你需要考虑的不只是版权。你可能会踩到专利地雷，其大部分被跟你正在使用的设计的诞生毫无关系的实体持有，而还可能有其他的法律问题。

切记版权法和专利法完全不同。假设他们有任何相同之处都是个错误。这就是“知识产权”的说法是完全的误导而应被彻底抵制的原因[3]。

[1]参见美国版权局定义的“useful article”，于 http://copyright.gov/register/va-useful.html.

[2]一篇由 Public Knowledge 所写的文章“为你的 3D 打印产物赋予许可的三个步骤” https://publicknowledge.org/assets/uploads/documents/3_Steps_for_Licensing_Your_3D_Printed_Stuff.pdf（2015 年 3 月 6 日）。对其在美国的复杂性给出了很有用的信息，尽管该文落入了使用伪概念“知识产权”，和不应该和版权联用的宣传用语“保护”的常见谬误。若问为何，参见《应避免使用（或慎用）的词语，由于它们是不公正的或者引起混淆的》一文的“保护（Protection）”一节 (p. 122) 。

[3]参见《您说过“知识产权”吗？这是一种迷惑的幻景》 (p. 99) 一文。

## 通过文件仓库提倡自由硬件

通过人们发表硬件设计的文件仓库的规则来劝说他们把已发表的设计变得自由是最有效的方式。仓库的操作员应当把将要使用设计的人们的自由置于设计者的偏好之上。这意味着要求有用物品的自由设计是在该仓库中发表它们的一个条件。

对装饰性物品，此论点不适用，所以我们不必须坚持它们必须是自由的。但是，我们必须坚持它们是可分享的。因此，一个既能处理装饰性物品模型也能处理功能性物品模型的仓库必对每一类存档都有合适的许可政策。

对数字设计，我建议仓库坚持使用 “GNU GPL 第三版或更新”、Apache 2.0 或 CC-0。对功能性三维设计，仓库要劝用户选择以下四种许可之一：“GNU GPL 第三版或更新”、Apache 2.0、CC-SA、CC-BY 或 CC-0。对装饰性设计，要选 “GNU GPL 第三版或更新”、Apache 2.0、CC-0 或任何 CC 族许可。

该仓库应当要求所有设计都以源代码形式发表，且只能被私有软件使用的私密格式源代码并不真正足够。对三维模型而言，STL 格式并不是用来修改设计的最佳格式因此不算源代码，所以仓库不该接受此格式，除非它们可能和真正的源代码一同发表。

没有理由为硬件设计的源代码只选择单一的格式，但尚未能被自由软件处理的源码格式充其量只应该勉强接受。

## 自由硬件和担保

一般说来，自由硬件设计的作者没有对按其设计装配硬件者提供担保的道德义务。这和出于销售硬件实体的问题不同，销售硬件实体必须由其卖家和/或制造者提供担保。

## 结论

我们已经有了可以让我们的硬件设计自由的合适许可证。我们需要的是作为一个社区将其认作我们所必做之事而在我们自行装配物品时坚持自由的设计。

# 应用自由软件判断准则



四项基本自由为判断某一特定代码片断是否为自由的（即尊重用户自由）提供了准则[1]。我们应当如何将它们应用于判断一个软件包、一套操作系统、一台计算机、或是一个网页是否适合被推荐使用呢?

一个程序是否是自由的首先影响到的是我们对于自己的私人行为的决定：为了捍卫我们自己的自由，我们需要拒绝那些将会剥夺我们自由的程序。然而，这也会影响到我们应当对别人怎样说和怎样做。

非自由程序是不公的。发布一个非自由程序、向他人推荐非自由程序、或是更为普遍地将它们引入课程而引导人们使用非自由软件，以上这些行为意味着引导人们放弃自己的自由。可以肯定的是，引导人们使用非自由软件并不等同于在他们的计算机上安装非自由软件，但我们不应该将人们引入歧途。

在更深层次上，我们不能提出一个非自由程序作为一个解决方案，因为这将会承认其合法性。非自由软件是一种问题；将其以一种解决方案的方式呈现否认了这一问题的存在[2]。

本文阐述了我们应当如何应用自由软件的基本准则来判断不同种类的事物，并且决定是否应该推荐它们。

## 软件包

一个软件包若要成为自由的，其中所有代码必须都是自由的。但不仅限于代码。由于文档文件包含手册、自述（README）、更新日志等，这些都是软件包的必要的技术组成部分，它们必须也是自由的[3]。一个软件包通常与很多其他软件包一起使用，并且与其中的一些进行交互。与非自由软件进行的何种交互才是伦理上可接受的呢?

[1]参见《什么是自由软件?》 (p. 1) 一文以获知自由软件的完整定义。

[2]我的文章《避免破坏性的妥协》 (p. 311) 详细论述了这一问题。

[3]参见《为什么自由软件需要自由的文档》 (p. 46) 以获知关于这一问题的更多细节。

我们着手开发 GNU 的目的是带来一款自由的操作系统，由于在 1983 年还没有这样的自由操作系统。当我们于 20 世纪 80 年代开发出最早的 GNU 组件时，其中每个组件都依赖于非自由软件是不可避免的。例如，没有任何一个 C 程序可以离开非自由的 C 编译器而运行，直到 GCC 可以正常工作，并且它们也不能离开 Unix libc 而运行，直到 glibc 可以正常工作。每个组件都只能运行在非自由操作系统上，因为当时所有的操作系统都是非自由的。

每当我们发布一款可以运行在某些非自由操作系统上的组件时，用户将其移植到其他非自由操作系统上。从伦理上讲，这些移植并不比我们曾经用于开发这些组件的限定平台代码更坏，因此我们整合了他们的修补程序。

当 Linux 内核于 1992 年变为自由之时，它填补了 GNU 操作系统的最后一块空缺（Linux 最初于 1991 年以一种非自由许可证发布）。GNU 和 Linux 的组合成为了一种完全自由的操作系统——GNU/Linux[1]。

此时，我们可以选择移除对非自由平台的支持，但是我们决定不这样做。非自由操作系统是不公的，但用户运行非自由操作系统并不是我们的过错。支持该非自由操作系统上的自由软件并不会进一步恶化这种不公。并且这将是实用的，不仅对于那些非自由操作系统用户，也对于吸引更多人为开发该自由软件做贡献。

然而，在自由程序上运行非自由程序是一个完全不同的问题，因为这是在诱导用户在自由之路上倒退。在某些情况下我们完全禁止这样做：例如 GCC 禁止任何非自由插件[2]。当一个程序允许非自由扩展的时候，它至少不应该引导用户使用它们。例如，我们更倾向于选择 LibreOffice 而非 OpenOffice，由于后者提示用户使用非自由扩展，而 LibreOffice 则避开了它们。我们开发冰猫（IceCat）[3]起初也是为了避免向用户推广由火狐（FireFox）建议使用的非自由扩展。

[1] 参见《Linux 和 GNU 操作系统》 (p. 75) 一文以获知更多信息。

[2] 如需获知为何 GCC 拒绝任何非自由插件，参见我在 GCC 邮件列表中的回复，它位于 https://gcc.gnu.org/ml/gcc/2014-01/msg00247.html。

[3] 参见 http://directory.fsf.org/wiki/IceCat。

事实上，如果冰猫解释如何在 MacOS 上运行冰猫，这将不会引导用户去运行 MacOS。但如果它介绍了一些非自由扩展，它将会鼓励冰猫用户安装这些非自由扩展。因此，冰猫软件包及其手册和网站不应该介绍这些东西。

有时一个自由软件和一个非自由软件协同工作，但其中任何一方都不是基于另一方的。我们针对这种情况的规则是，如果该非自由软件非常有名，我们应当告知人们如何使用我们的自由软件与之工作；但如果该专有软件鲜为人知，我们不应该暗示其存在。有时我们会在该非自由软件存在的情况下提供互操作支持，但避免告知用户这么做的可能性。

我们拒绝任何仅可用于某一非自由操作系统的“增强组件”。它们会鼓励人们使用该非自由操作系统而非 GNU，如同自摆乌龙。

## GNU/Linux 发行版

随着 Linux 内核于 1992 年自由化，人们开始开发 GNU/Linux 发行版（distros）。但只有少数发行版是完全由自由软件构成的。

适用于软件包的规则也适用于发行版：一个符合伦理的发行版必须仅包含自由软件并且只将用户向自由软件的方向引导。但是，对于一个发行版而言，“包含”一个特定的软件包是什么意思呢？

某些发行版从二进制包安装作为其发行版一部分的软件；而其他发行版从上游源代码构建每个软件，并且从字面意义上讲，它们所包含的只是需要下载并构建的列表。对于自由的问题，发行版怎样安装一个给定的软件包并不重要；如果它将某个软件包作为可选项提供，或者它的网站这样做，我们称该发行版“包含”该软件包。

自由操作系统的用户拥有对它的控制权，于是他们可以安装他们想要安装的任何东西。自由发行版提供了用户可用于安装他们自己的程序以及他们对于自由软件的修改版本的通用工具；他们也可以安装非自由软件。在这些发行版中提供这些通用工具并不是伦理瑕疵，由于该发行版的开发者对于用户基于其自身的主动权获取并安装什么软件并不负有责任。

然而，当开发者引导用户走向非自由软件的时候，他们对于用户安装

非自由软件就应当负有责任了——例如，将非自由软件置于流行发行版的软件包列表中，或者从它们的服务器进行分发，或者将其呈现为一种解决方案而非一种问题。这正是为何大多数 GNU/Linux 发行版具有伦理瑕疵的问题之所在。

自行安装软件包的用户通常具有一定的判断能力：如果我们告诉他们“Baby 包含非自由代码，而 Gbaby 是自由的”，我们可以预见他们能够留心记住哪个是哪个。但发行版是推荐给那些不了解这些细节的普通用户的。他们将会想“我应该使用他们所说的哪个呢？我想它应该是 Baby”。

因此，要想向公众推荐一款发行版，我们坚持要求它们的名字不能与我们所拒绝的某个发行版相似，唯有如此，我们关于仅仅推荐自由发行版的信息才能被可靠地传达。

发行版和软件包之间的另一个区别在于向其中添加非自由代码的可能性。程序开发者会仔细检查他们向其中添加的代码。如果他们决定使该程序成为自由的，他们不太会向其中添加非自由代码。不过也有例外，包括 Linux 内核中添加“二进制 blobs”这样恶劣的案例，但它们与现存的自由软件相比只占一小部分。

与之相反，一个 GNU/Linux 发行版通常包含数千个软件包，并且其开发者可能每年都会向其中添加数百个新的软件包。如果未能尽力避免那些包含某种非自由软件的软件包，几乎肯定会有一些非自由软件混入其中。由于自由发行版的数量很少，作为列出那些发行版的条件，我们要求每位自由发行版的开发者通过移除任何非自由代码或恶意代码来承诺保持该发行版成为自由软件。参见 GNU 自由操作系统发行版指导意见，它位于 http://gnu.org/distros/free-system-distribution-guidelines.html。

我们不要求自由软件包的开发者也做出这样的承诺，这是不现实的，幸运地是，这也不是必需的。得到超过 30000 个自由软件的开发者的承诺也许能够避免少数问题，但其代价是极大增加自由软件基金会（FSF）员工的工作量；此外，这些开发者中的大部分与 GNU 计划并无关系，他们也不愿意向我们做出任何承诺。因此我们只需在发现问题的时候应对这些使自由软件变为非自由软件的少数案例。

## 外设

计算机外设要求计算机中的软件被操作系统加载到其中以便使其工作——这些软件可以是驱动程序或固件。因此，如果一件外设可以在一台未安装任何非自由软件的计算机上使用——如果该外设的驱动程序以及任何需要由操作系统加载到其中的固件都是自由的，那么它是可接受并使用以及推荐的。

验证这一点是简单的：将该外设连接到一台运行完全自由的 GNU/Linux 发行版的计算机上并且观察它是否正常工作。但是大多数用户需要在购买外设之前获知这一点，因此我们在 h-node.org 列出了很多外设的信息，这是一个完全自由的操作系统的硬件数据库。

## 计算机

一台计算机在不同层次上包含不同的软件。我们应当基于什么准则来判断一台计算机是否“尊重您的自由”呢?

显而易见的是：操作系统和其中的任何软件都必须是自由的。在 20 世纪 90 年代，启动加载软件（当时是“基本输入/输出系统”，即 BIOS）成为可替换的，并且由于它运行在中央处理器（CPU）上，它与操作系统所存在的是同一类的问题。因此，诸如固件或驱动程序，不论安装在操作系统中，或是随操作系统一起安装，或是启动加载程序都必须是自由的。

如果一台计算机拥有某些要求在操作系统中安装的非自由驱动程序或固件的硬件功能，我们可能仍然能够推荐它。如果它在没有那些功能的情况下仍然可用，并且我们认为大部分人不会为了使该功能可用而被引导安装非自由软件，那么我们仍然能够推荐它们。否则我们就不能。这将是一种主观判断。

一台计算机可能在较低的层次上带有预装的可修改固件和微码。它也可能在真正只读的存储器中拥有代码。我们决定在现今我们所使用的认证准则中忽略这些程序，这是由于如若不然就没有任何计算机可以满足，并且因为通常不会被更改的固件在伦理上与电路相同。因此我们的认证准则

仅仅覆盖那些运行在计算机的主处理器上且储存在非真正只读存储器中的代码。当在其他层次上运行自由软件成为可能，我们也会要求这些层次上的软件是自由的。

由于认证一款产品是对它的积极推广，我们要求它们的贩卖者以支持我们作为回报，这可以通过谈论自由软件而非开源软件[1]以及将 GNU 和 Linux 的结合体称为 GNU/Linux [2]来做到。我们没有义务积极支持那些不认可我们的工作或是不支持我们运动的项目。

参见 http://www.fsf.org/resources/hw/endorsement/criteria 以获知我们的认证准则。

## 网页

现在的很多网页都包含复杂的 JavaScript 程序并且需要它们才能工作。这是一种有害的实践，因为它阻碍用户对他们自己计算的控制。更坏的是，这些程序中的大部分是非自由的，这是一种不公。JavaScript 代码常常窥探用户[3]。JavaScript 已经变成了一种对用户自由的威胁。

为了解决这一问题，我们开发了 LibreJS，这是一种用于阻止非普通非自由的 JavaScript 代码的火狐浏览器扩展（没有必要阻止简单的脚本，如果它们只是实现一些次要的用户界面特性）。我们请求网站将它们的 JavaScript 程序自由化并且标记其许可证以便 LibreJS 识别。

与此同时，链接至一个包含非自由 JavaScript 程序的网页是否符合伦理呢？如果我们坚决不做任何妥协，我们将只能链接至自由的 JavaScript 代码。然而，很多网页即使不运行它们的 JavaScript 代码也能工作。此外，除了我们的链接，您会经常在其它网站遇到非自由的 JavaScript 代码；为了避免这些情况，您必须使用 LibreJS 或禁用 JavaScript。因此，我们决定做出让步并且链接那些不运行非自由 JavaScript 程序也能工作的网页，同时鼓励用户在普遍意义上保护自己不受来自非自由 JavaScript 程序的威胁。

[1] 参见《如今自由软件更加重要》(p. 32) 和《为什么说开源漏掉了自由软件的要点》(p. 90) 。
[2] 参见《名字的含义?》 (p. 71) 一文
[3] 参见《JavaScript 陷阱》 (p. 286) 一文

然而，如果某个网页不运行非自由 JavaScript 程序就不能实现其功能，链接到它将会不可避免地要求人们运行该非自由 JavaScript 代码。出于原则，我们不会链接这些网页。

## 结论

将“软件应当是自由的”这一基本理念应用到不同场合要求不同的实践策略。随着新情况的出现，GNU 计划和 FSF 将会适配我们的自由准则，不论在实践中还是理论上，都将计算机用户引向自由。通过仅仅推荐尊重用户自由的程序、发行版和硬件产品，并且宣示您的立场，您可以为自由软件运动提供它所急需的支持。

# 第 2 部分

# 名字的含义

## 名字的含义?



关于此主题了解更多，你也可以阅读我们的 GNU/Linux FAQ http://gnu.org/gnu/gnu-linux-faq.html 和本书的《Linux 和 GNU 操作系统》 (p. 75) 一文，它讲了 GNU/Linux 系统的历史，和这命名的问题有关。以及另一篇文章《从未听过 GNU 的 GNU 用户》http://gnu.org/gnu/gnu-users-never-heard-of-gnu.html。

顾名则思义；我们所选的名字代表了我们所要传达的思想。一个不恰当的名字会向人们传递错误的思想。玫瑰不管叫什么闻起来总是甜的——但是如果你把它叫作笔，当人们用它写字时将会很失望。同样地，如果你把笔叫作 “玫瑰”，人们可能就不知道它能做什么了。如果你把我们的操作系统称为 “Linux”，则会传达一些关于此系统起源、历史和目的的错误概念。如果你称之为 GNU/Linux，这将传达（虽然不详细）一个准确的概念。

这对我们的社区重要吗？人们是否了解系统的起源、历史和目的重要吗？是的——因为忘记历史的人往往注定要重蹈覆辙。目前围绕 GNU/Linux 建立起来的自由世界仍然无法保证生存；那些迫使我们开发 GNU 的问题并没有被彻底根除，而且他们总是想复辟。

当我解释为什么把这个操作系统称为 “GNU/Linux” 比 “Linux” 要好时，人们有时这样回应：

> 就算 GNU 工程因其工作而应该得到赞誉，但如果人们不给予它这种赞誉，就真的值得如此大惊小怪的吗？不管是谁做的，反正工作被完成了，这不是重要的事吗？你应该心态平和，以干好工作为荣，而不是对荣誉的问题耿耿于怀。

这似乎是个明智的建议，如果所有的情况都像那样，大家干完工作后都不计得失，那这建议还真是对的！但现实充满挑战，现在还不是那样对未来想当然的时候。我们社区的力量靠的是保证自由与合作。使用 GNU/Linux 这个名字是一种让人们提醒自己和他人了解这些目标的方式。

虽然不考虑 GNU 也能写出优秀的自由软件；许多优秀的工作是在 Linux 的名义下完成的。但是 “Linux”（这个名字）从它被创造的那一刻起，就与一种不对自由协作做承诺的哲学联系在一起。随着这个名字的广泛商业使用，当我们把它和社区精神相联系时，将会遇到更多的麻烦。

对自由软件来说，未来的一大挑战是来自于那些 “Linux” 发行版公司，为提高易用性和增强功能，将非自由的软件添加到 GNU/Linux 中。所有主要的商业发行版都这样做；没有一个让自己限于自由软件的。他们中的大多数都在其发行版中没有区分非自由软件包。很多甚至开发非自由软件并加入到其系统发行中。还有些甚至蛮横的地宣布那些 “Linux” 系统是 “席位许可” 的，也就是可以给用户和 Microsoft Windows 一样的 “自由”。

人们用 “普及 Linux ” 的名义替这种加入非自由软件的行为辩护——事实是，更重视普及度而不是自由。有时这是被公开承认的。例如，《连线》杂志（Wired）对 Linux Magazine 期刊的编辑 Robert McMillan 说，“感觉推动开源软件应该用技术，而不是政治决策”。并且 Caldera 的 CEO 公开劝

说用户放弃自由的目标，并转而为 “普及 Linux” 而工作[1]。

如果普及率指的是使用混合非自由软件的 GNU/Linux 的用户数量，添加非自由软件到 GNU/Linux 系统或许会增加普及率。但同时，它含蓄地鼓励社区将非自由软件当好东西接受，而忘记自由的目标。如果你的方向不对，那么车开得再快也没用。

当非自由的 “附加组件” 是一种库或者编程工具时，就会成为自由软件开发者的陷阱。当他们编写基于非自由软件包的自由软件时，他们的软件不能成为完全自由系统的一部分。Motif 和 Qt 曾以这种方式诱骗了大量的自由软件，导致的问题花费数年才能解决。Motif 的一些问题直到它被废弃并不再被使用都没有完全解决。Sun 公司实现的非自由 Java 直到现在也有类似的效果：Java 陷阱[2]，庆幸的是大多已经修复。如果我们的社区持续向这个方向发展，将会把 GNU/Linux 的未来引到自由和非自由拼接在一起的地步。从现在起的五年里，我们仍然确信将会有大量的自由软件产生；但如果我们不谨慎一些，用户所期望的自由软件就很难在脱离非自由软件的情况下使用。如果这事发生了，我们为自由做出的努力就前功尽弃了。

如果发布自由的替代品只是简单的编程问题，随着我们社区开发资源的增加，未来解决问题可能就变得愈加简单。但是我们面临的障碍使得这更加困难：法律禁止自由软件。当软件专利出现，并且像数字千年版权法案（DMCA）这样的法律被用来阻止开发一些用于重要工作中的自由软件时（例如看 DVD 或者听 RealAudio 音频流），我们会发现，除了**拒绝使用调用它们的非自由程序**，我们自己并没有明确的方法来抵御专利和私密的数据形式。

面对这些挑战需要许多不同的努力。但是应对任何挑战，我们首先需要牢记自由协作的目标。我们不能仅仅依靠渴望强大、可靠的软件来激励人们作更多的努力。我们需要大家有一种决心，一种为他们的自由和他们的社区而战的决心——一种锲而不舍的决心。

在我们的社区里，这个目标和决心主要源自 GNU 工程。我们就是这样

[1]参见 Dietmar Muller 于 2001 年 7 月 10 日发表的文章《“Stallman: Love Is Not Free”》http://zdnet.com/article/stallman-love-is-not-free/

[2]参见《“自由的桎梏——Java 陷阱”》http://gnu.org/philosophy/java-trap.html 一文

的人——把自由和社区当作不可让步的事情；而那些称之为 “Linux” 的组织通常不说这些。关于 “Linux” 的杂志更是打满了非自由软件的广告；打包 “Linux” 的公司把非自由软件添加进系统；其它公司通过开发在 GNU/Linux 上运行的非自由的应用来 “支持 Linux”；连 “Linux” 的用户群也邀请推销员展示这些应用。人们在社区中最有可能接触到自由的理念和决心的地方，就是在 GNU 工程中。

但是当人们接触到它时，能体会到这与他们息息相关吗?

那些知道自己正在用的系统是源自 GNU 工程的人，能看到他们自身与 GNU 之间有一个直接的关系。他们不会自动地同意我们的哲学，但至少他们会看到一个严肃思考它的理由。相反，那些认为自己是 “Linux 用户” 的人，相信 GNU 工程只是 “开发了被证实在 Linux 中非常有用的工具”，他们通常只了解 GNU 与其间接的关系。当他们接触到 GNU 哲学时，就会忽视它。

GNU 工程是理想主义的，但是那些鼓励理想主义的人面临很大的障碍：普遍的意识形态鼓励人们把理想主义当作 “不现实的” 而摒弃。我们的理想主义其实已经极其实用：这是我们拥有一个自由的 GNU/Linux 操作系统的原因。喜欢这个系统的人应该知道，正是我们的理想主义实现了它。

如果 “工作” 真的做了，如果除了信誉没有其他利害攸关，也许放弃可能会更明智。但是我们没有处在那个境地。为了鼓励人们做那些需要做的工作，我们需要让我们已经做了的工作被认可。要帮助我们就请将这个系统称为 GNU/Linux 吧。

# Linux 和 GNU 操作系统



每天，大量的用户在毫不知情的情况下在他们的计算机上运行着修改版的 GNU 操作系统[1]。而这个现在广为使用的修改版的 GNU 操作系统通常被称为 “Linux”，但是很多它的用户并不知道，它其实就是 GNU 计划开发的 GNU 操作系统[2]。

Linux 也确实存在，并且那些用户也确实是在使用它，不过它仅仅是这些用户所使用的操作系统的一部分。Linux 仅仅是一个内核：它就是分配你硬件设备上的资源给其他你所使用的程序的一个特殊程序。内核是操作系统中不可或缺的一部分，但是只有内核是远远不够的。内核仅能在完整操作系统的环境下才能正常工作。一般来讲，Linux 会与 GNU 操作系统合起来使用：整个系统就是添加了 Linux 内核的 GNU 操作系统，或者简而言之，GNU/Linux。所有所谓的 “Linux” 发行版其实就是 GNU/Linux 发行版。

很多用户并不能区分内核（即 Linux）和整个操作系统（他们也称为 “Linux”）。这种混淆视听的称呼并不能帮助人们正确理解这一区别。所以人们通常会认为 1991 年 Linus Torvalds 在没有借助太多其他帮助的情况下就开发出了整个操作系统。

程序员们基本上都知道 Linux 只是个内核。但是他们通常会听到别人把整个操作系统都称为 Linux，然后他们就会跟从接受以内核名称命名操作系统的传统。比如，很多人都相信当 Linus Torvalds 编写完 Linux 内核之后，它的用户们去寻找其他与之搭配使用的自由软件时，（毫无道理地）发现用于构建类 Unix 系统的一切其实都已然存在。

真实的情况是，当时的他们发现了并不完全完整的 GNU 系统，而

[1] 参见《自由与非自由软件的分类》查看更多有关 GNU 操作系统的信息 (p. 85) 。

[2] 详情请见 “GNU Users Who Have Never Heard of GNU” http://gnu.org/gnu/gnu-users-never-heard-of-gnu.html 和 “Overview of the GNU System”，http://gnu.org/gnu/gnu-history.html。

这一发现决非偶然。当时已有的自由软件[1]加在一起一同组成了一个完整的操作系统。这是因为 GNU 项目从 1984 年就开始致力于实现这一目标。在 GNU 宣言（GNU Manifesto）[2]当中，我们就确立了开发一个称为 GNU 的类 Unix 的自由操作系统的目标。GNU 计划的初始声明（The Initial Announcement）[3]还包括了 GNU 操作系统开发计划的部分大纲。在 Linux 开始开发前，GNU 几乎就要完成了。

绝大多数的自由软件项目都是以“为某项具体的工作开发一款具体的软件”为目的。比如，Linus Torvalds 开发了类似于 Unix 的内核（Linux）；Donald Knuth 编写了一个格式化文本工具（TEX）；而 Bob Scheifler 则开发了一种窗口管理系统（X 窗口管理系统）。对于这类软件而言，衡量特定的软件对该项目的贡献是简单而自然的。

但是如果我们以这种方法去衡量 GNU 项目的贡献的话，我们会得出怎样的结果呢？某个 CD-ROM 的供应商发现在他们的“Linux 发行版”中，GNU 软件[4]是其中最大的必备依赖软件，约占全部源代码数量的 28%，并且还包括了构成完整操作系统不可或缺的一部分组件。而 Linux 自己只占 3%（截至 2008 年，此比例仍然准确：在 gNewSense 的“主源”当中，Linux 占 1.5%，而 GNU 软件包占了 15%）。所以如果你想以写操作系统的人命名这个操作系统的话，最为确切的单词名字恐怕是“GNU”。

但这并不是考虑这个问题最深层次的方法。GNU 项目过去不是，现在也不是仅仅只创造某些特定软件的一个项目。它不是只开发一个 C 语言编译器的项目[5]，虽然我们确实开发了一个 C 语言编译器。它也不是只开发一个文本编辑器的项目，虽然我们也做过。GNU 项目是为了开发一个完全自由的类 Unix 操作系统——GNU。

很多人都为自由软件做过贡献，他们理应被提名。但是问题在于这是一个完整的操作系统——并不是一个有用软件的简单集合——因为 GNU

[1]参见《什么是自由软件?》 (p. 1) 来获得自由软件的完整定义。
[2]查看“GNU 宣言（GNU Manifesto）”，位于 http://gnu.org/gnu/manifesto.html。
[3]参见《GNU 操作系统的初始公告》 (p. 29) 。
[4]参见《自由与非自由软件的分类》了解GNU 软件的更多信息 (p. 86) 。
[5]GCC 主页：http://gnu.org/software/gcc/。

项目就是要创造这样一个操作系统。我们已经列出了一个完整操作系统所需软件的清单，并且我们系统性地发掘、编写，或者找人编写清单上的所有条目。我们编写了一些不太吸引人却又不可或缺的组件[1]，因为操作系统的正常运行不能没有这些软件。在我们系统组件中，有一部分是编程工具，并且在程序员中流行了起来。但是我们也编写了很多不是工具的软件[2]。我们甚至还编写了一个棋类游戏，GNU Chess，因为完整的操作系统还需要一些游戏。

在 90 年代初期，我们已经搞定了一个除了内核之外的系统。我们其实也开始开发一个内核，GNU Hurd（http://gnu.org/software/hurd/hurd.html），一个运行于 Mach 的内核。开发这个内核的难度远超我们的想象；GNU Hurd 在 2001 年终于能稳定运行了，但是距离被人们日常使用的目标仍然相差甚远[3]。

幸运的是，因为有 Linux 的存在，我们并不需要等待 Hurd。在 1992 年，Torvalds 自由化了 Linux 之后，它填补了 GNU 操作系统中重要的最后一道坎。人们从此可以将 GNU 操作系统[4]和 Linux 合并使用来创造一个完全自由的操作系统——一个包含 Linux 的 GNU 操作系统，即 GNU/Linux 操作系统。去让它们能很好地协同工作并不是一件容易的事。一些 GNU 组件 [5] 需要作出一些必要的改动才能和 Linux 一起使用。将一个完整系统整合成一个能 “开箱即用” 的发行版也是一个不小的挑战。这需要解决一个我们之前没有遇到过的问题——如何安装和启动系统，因为我们还没有研究到那一步。因此，制作了各种各样发行版的人们进行了大量的必要工

[1]这些不太引人注意而不可或缺的组件包括 GNU 汇编器（GAS）和链接器（GNU ld），它们现在都是 GNU Binutils 软件包（http://gnu.org/software/binutils/）的一部分，还有 GNU tar（http://gnu.org/software/tar/）等软件。

[2]比如，Bourne Again Shell（BASH），还有 PostScript 解析器 Ghostscript（http://gnu.org/software/ghostscript/ghostscript.html），以及 GNU C 运行库都不是编程工具。并且 GNUCash，GNOME，和 GNU Chess 都不是。

[3]参见 http://gnu.org/software/hurd/hurd-and-linux.html 就会明白为什么 FSF 开发了 GNU Hurd 内核。

[4]参见 “Linux 0.01 发行注记”（“Notes for Linux Release 0.01,”）：http://ftp.funet.fi/pub/linux/historical/kernel/old-versions/RELNOTES-0.01。

[5]比如 GNU C 运行库（GNU C Library）http://gnu.org/software/libc/libc.html。

作。但是这就像万物的规律一样，最终会有人去解决这个问题的。

GNU 项目支持 GNU/Linux 系统和 *GNU 操作系统*。FSF 资助了重写 GNU C 运行库中与 Linux 有关的扩展的工作，这样，最新的 GNU/Linux 操作系统就能毫不修改地使用最新版本的库了。FSF 还资助了 Debian GNU/Linux 的早期开发。

现在，GNU/Linux 操作系统有非常多的发行版（英文中常称为 “distros”）。它们当中的绝大部分遵从 Linux 的哲学而不是 GNU 的哲学。不过彻底自由的 GNU/Linux 发行版也是存在的[1]。FSF 赞助了 gNewSense 计算机等设备（http://gnewsense.org）。

制作一个完全自由的 GNU/Linux 发行版并不只是删去非自由软件程序。现今，原版 Linux 也包含非自由程序。这些程序是在系统启动时加载进 I/O 设备的，大量的这些程序被包含在 Linux 的 “源代码” 中。因此，维护自由版本的 GNU/Linux 发行版现在还指维护一个自由版本的 Linux（http://directory.fsf.org/project/linux）。

不管你是否使用 GNU/Linux，请不要使用歧义词 “Linux” 迷惑大众。Linux 是内核，系统中主要的不可或缺的组件之一。而整个系统其实是加入了 Linux 的 GNU 操作系统。当你提及这个组合时，请说 “GNU/Linux”。

这篇文章和 “GNU 计划” 都是宣扬 “GNU/Linux” 的好选择。如果你提到 Linux 内核，并且你还想引用更有深度的参考资料，FOLDOC（计算机自由在线词典）http://foldoc.org/linux 是一个你可以信赖的网站。

## 附言

除了 GNU，另外一个项目也独立完成了一个自由的类 Unix 操作系统。这个操作系统叫 BSD，研发于加州大学伯克利分校（UC Berkeley）。在八十年代期间，它并不是自由的，但是在九十年代早期，它变成了自由的。

[1] 参见 http://gnu.org/distros/ 以获得我们知道的所有完全自由的发行版的列表。

现存的自由操作系统几乎是非 GNU 即 BSD[1]。

人们有时候会问 BSD 是不是也是某种版本的 GNU，就像 GNU/Linux 那样。其实 BSD 开发者们也是得到了 GNU 项目的启发才使他们的代码自由了。而且很显然这种行为是由 GNU 的积极分子努力游说的结果，但是他们的代码与 GNU 并无太多交集。BSD 操作系统现在也使用一些 GNU 软件，就像是 GNU 操作系统及其变种也使用一些 BSD 软件一样。然而，总体而言，它们是两个完全不同、分开发展的操作系统。BSD 开发者没有采用编写内核然后放进 GNU 操作系统的做法，所以将 BSD 称为 GNU/BSD 很显然是不合适的[2]。

[1]撰写此文时，一个类 Windows 的接近全部自由的操作系统已经开发了出来，但是从技术角度来讲，它不像 GNU 或者 Unix，所以并没有受此问题的影响。Solaris 的大部分内核都已自由开放，但是如果你想基于 Solaris 实现一个完全自由的操作系统，除了需要替换内核中缺失的实现外，还需要放进 GNU 或者 BSD 当中。

[2]从另一个方面来讲，在此文撰写的这几年间，GNU C 运行库已经被移植到了很多版本的 BSD 内核上，这为集成 GNU 操作系统到该内核上带来了方便。就像 GNU/Linux 那样，GNU 还有好多个变种，比如 GNU/kFreeBSD 和 GNU/kNetBSD。普通桌面用户可能不好区分 GNU/Linux 和 GNU/*BSD。

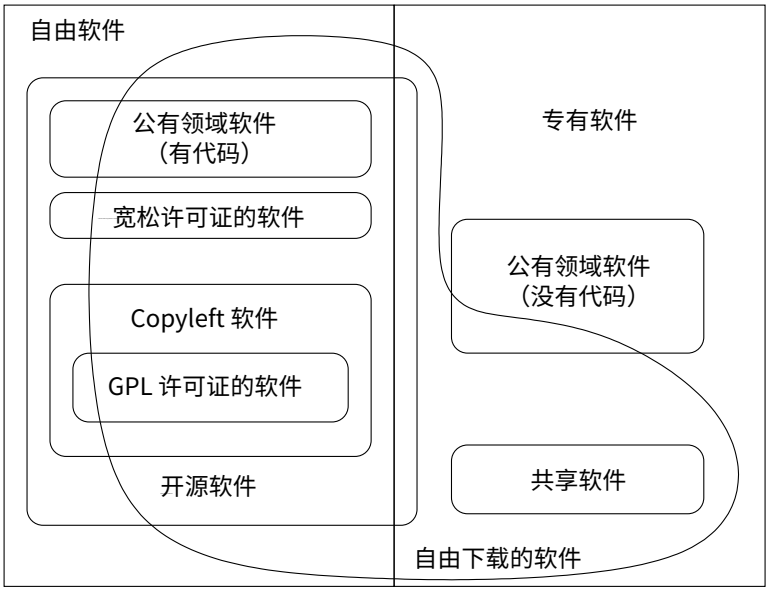


图 **2.1:** 自由与非自由软件的分类

# 自由与非自由软件的分类



参见《应避免使用（或慎用）的词语，由于它们是不公正的或者引起混淆的》 (p. 107) 一文。

本图最初由 Chao-Kuei 创作并且经过其他人的更新，解释了不同类别的软件之间的区别。现有它的矢量图版，位于 http://gnu.org/philosophy/category.svg（以及中文矢量图版，位于 https://www.gnu.org/philosophy/category.

zh-cn.svg），以及 XFig 格式版本，位于 http://gnu.org/philosophy/category.fig。本图采用 GNU 通用公共许可证（GNU GPL）2.0 或更高版本、GNU 自由文档许可证（GNU FDL）1.2 或更高版本、或者创作共用-署名-相同方式共享（CC BY-SA）许可证 2.0 或更高版本。

## 自由软件（Free Software）

自由软件是指附带这样许可的软件：它允许任何人使用、复制和/或再分发，不论是逐字再分发还是再分发带有更改的版本，也不论免费还是有偿。特别地，这意味着源代码必须可获得。“如果它不是源代码，它就不是程序。”这是一种简化的描述；您可以在《什么是自由软件?》 (p. 1) 找到自由软件的完整定义。

如果一个程序是自由的，它就有可能被包含在一种自由的操作系统中，例如 GNU，或者 GNU/Linux 操作系统的自由版本[1]。

还有很多种方式可以使程序成为自由的——很多细节的问题，对这些问题可以有多种方式的决定，而仍然能够使程序成为自由的。一些可能的变体将会在下文进行描述。如果需要获得关于具体的自由软件许可证的信息，参见许可证列表页面 http://gnu.org/licenses/license-list.html。

自由软件关乎的是自由而非价格。但是私有软件公司一般使用 “free software” 这一短语来指代价格。有时它们的意思是您可以免费获得一份二进制副本；有时它们的意思是在您所购买的计算机上捆绑了一份副本，而整台计算机的价格包含了二者各自的价格。不管是哪种方式，这都与我们在 GNU 计划中所指的自由软件无关。

由于这种潜在的混淆，当一家软件公司宣称它的产品是 “free software” 的时候，务必查看实际的发布条款以确认用户是否真正拥有自由软件所蕴含的所有基本自由。有时它真的属于自由软件，有时它不是。

很多语言都拥有两个独立的单词用来表示 “free” 中的自由和零价格的免费。例如，法语中有 “libre” 和 “gratuit”。而英语则不是；在英语中确实

[1] 参见《Linux 和 GNU 操作系统》 (p. 75) 一文以获得更多信息。

有一个单词 “gratis” 用于无歧义地表示价格，但没有普通的形容词用于无歧义地表示自由。因此，如果您说另一种语言，我们建议您在将 “free” 翻译为您的母语的时候澄清其涵义。参见我们整理的将 “free software” 这一短语翻译为多种其他语言的详细列表（见附录 B (p. 341) ）。

自由软件常常比私有软件更可靠[1]。

## 开源软件（Open Source Software）

“开源软件” 这一短语被某些人用于指代与自由软件或多或少地相同的一类软件。它们并不是与自由软件严格一致的一类软件：它们接受某些在我们看来约束性过强的许可证（条款），也有一些自由软件许可证（条款）是它们所拒绝接受的。然而，两者外延的差别并不大：几乎所有的自由软件也符合开源软件定义，并且几乎所有的开源软件也是自由的。

我们倾向于使用自由软件这一概念，由于它代表自由——这是 “开源” 这一短语所不能体现的[2]。

## 公有领域软件（Public Domain Software）

公有领域软件是指不受版权保护的软件。如果源代码也在公有领域中，这将成为一种非左版自由软件的特例。这意味着某些副本或者修改版本可能完全不是自由的。

在某些情况下，可执行程序可能位于公有领域中，但源代码不可获得。这不是自由软件，由于自由软件要求源代码的可获得性。与此同时，大部分自由软件不在公有领域中，它们是受版权保护的，并且版权持有人通过使用自由软件许可证合法地赋予任何人使用它们的自由许可。

有人以一种不严格的方式来使用 “公有领域” 这一概念来指代 “自由” 或 “免费得到”。然而，“公有领域” 是一个法律概念，准确表示 “不受版权保护” 之意。为了澄清，我们建议将 “公有领域” 仅仅用于它所严格指代的

[1] 参见 “自由软件更值得信赖！” 一文，位于 http://gnu.org/software/reliability.html。

[2] 参见《为什么说开源漏掉了自由软件的要点》 (p. 90) 一文。

涵义，而使用其他短语来指代其他涵义。

根据伯尔尼保护文学和艺术作品公约（大多数国家已经签署）[1]，任何写下的东西自动获得版权，这也包括计算机程序。因此，如果您想要使您的程序进入公有领域，您必须采取某些法律步骤以放弃其版权；否则该程序是受版权保护的。

## 左版软件（Copylefted Software）

左版软件是一类自由软件，其发布条款可以保证它所有版本的所有副本都或多或少带有相同的发布条款。例如，这意味着左版许可证普遍禁止他人为该软件添加额外的限制条款（尽管有限的一些安全的附加条款可能被允许），并且要求源代码可获得。这可以保护该程序及其修改版本，使得常见的某些方式无法将其变为私有软件。

一些左版许可证例如 GNU 通用公共许可证（GNU GPL）第三版阻止了其他一些私有化的方式，例如 Tivo 化（tivoization）[2]。

在 GNU 计划中，我们为我们所写的*几乎所有*软件使用了左版许可证，由于我们的终极目标是赋予每位用户自由软件所蕴含的所有基本自由。参见我们关于左版的文章《什么是 Copyleft?》 (p. 222) 以获知关于左版许可证如何发挥作用以及我们为何使用它们的更多解释。

左版是一个通用的概念，为了以左版许可证发布一个实际存在的程序，您需要使用特定的一系列发布条款。有多种可能的方式编写左版发布条款，因此原则上可以有很多种左版自由软件许可证。然而实际上几乎所有左版软件都使用 GNU GPL。两种不同的左版许可证通常不兼容，这意味着直接将使用一种许可证的代码与使用另一种许可证的代码合并将会是非法的；因此，人们使用同一种左版许可证将会有利于社区。

[1] 《伯尔尼保护文学和艺术作品公约》（法语：Convention de Berne pour la protection des œuvres littéraires et artistiques），简称《伯尔尼公约》，是关于著作权保护的国际条约。《公约》生效至今（2013 年）进行过 8 次补充和修订，共有 167 个签约国。——译者注，摘自维基百科

[2] 参见《为何升级到 GPLv3》 (p. 244) 以获得更多信息。

## 非左版的自由软件（Noncopylefted Free Software）

非左版的自由软件带有来自作者的再分发和修改的许可，也带有添加额外限制条款的许可。

如果一个程序是自由的但并未采用左版许可证，那么某些副本或者修改版本可能完全不是自由的。软件公司可以编译该程序，不论是否对其进行修改，都可以将可执行文件作为私有软件产品发布。

X Window 系统阐释了这一点。X 联盟将 X11 以某种非左版发布条款发布，使其成为非左版的自由软件，并且后续开发者几乎沿用了相同的方式。一份带有此类发布条款的副本是自由软件。然而，也有私有版本的存在，并且确实有（或者至少确实曾经有）流行的用于工作站或个人计算机（PC）的图形显示卡仅支持私有版本。如果您使用的是这类硬件，那么 X11 对您来说不是自由软件。X11 的开发者甚至曾经在一段时期将 X11 作为私有软件发布[1]；他们之所以能够如此，是由于其他人使用相同的非左版许可证贡献了他们的代码。

## 使用包容型许可证的软件（Lax Permissive Licensed Software）

包容型许可证包括 X11 许可证和两种 BSD 许可证[2]。这些许可证几乎允许对其代码进行任何方式的使用，包括发布私有的二进制版本，不论是否更改了源代码。

## 使用 GPL 许可的软件（GPL-Covered Software）

GNU GPL 是一类特定的发布条款集合，用于使该程序成为左版软件。GNU 计划将其用作大部分 GNU 软件的发布条款。

因此，将自由软件等同于使用 GPL 许可的软件是一种错误。

[1] 参见《X Window 系统的陷阱》 (p. 216) 。

[2] 参见 “The BSD License Problem,” 一文，位于 http://gnu.org/philosophy/bsd.html。

## GNU 操作系统（The GNU Operating System）

GNU 操作系统是一种 “类 Unix 操作系统”，它完全由自由软件构成，我们在 GNU 计划中于 1984 年开始 GNU 操作系统开发[1]。

一款类 Unix 操作系统由众多程序构成。GNU 操作系统包括所有 GNU 程序包。它还包括很多其他程序包，诸如 X Window 系统以及 TEX，它们不是 GNU 软件。

完整的 GNU 操作系统的首个测试版于 1996 年发布，它包括 GNU Hurd，我们的内核，后者于 1990 年开始开发。直到 2001 年，GNU 操作系统（包括 GNU Hurd）开始能够相对可靠地运行，但是 Hurd 仍然缺少一些重要功能，因此它不能被广泛使用。与此同时，GNU/Linux 操作系统，作为使用 Linux 内核而非 Hurd 内核的 GNU 操作系统的衍生版本，自 20 世纪 90 年代起已经获得了巨大成功[2]。这表明了 GNU 操作系统不是一个静态的程序集合；用户和再分发者可以根据他们的需求和偏好来选择不同的软件包。其结果仍是一种 GNU 操作系统的变体。

由于 GNU 操作系统的目标是自由，GNU 操作系统中的任何一个组件都是自由软件。但是，它们不必须都是左版的，任何类别的自由软件都可以被合法并且合适地包含进来，如果它有助于实现技术上的目标。

## GNU 程序（GNU Programs）

“GNU 程序” 是 GNU 软件的同义词。若程序 foo 是一个 GNU 程序，那么它也是一个 GNU 软件。我们有时也称之为 GNU 软件包。

[1]参见 “GNU 系统概览” 一文，位于 http://gnu.org/gnu/gnu-history.html 以获知更多历史背景。

[2]参见《Linux 和 GNU 操作系统》 (p. 75) 一文以获得更多信息。

## GNU 软件（GNU Software）

“GNU 软件” 是指在 GNU 计划支持下发布的软件[1]。如果一个程序是 GNU 软件，我们也可称之为 GNU 程序或 GNU 软件包。GNU 软件包的自述文件（README）或手册应当声明它是一个 GNU 软件包；同时，自由软件目录[2]标识了所有 GNU 软件包。

大部分 GNU 软件是左版的，但不是全部；然而，所有 GNU 软件都必须是自由软件。

有些 GNU 软件是由自由软件基金会（FSF）员工所编写的，但是大部分 GNU 软件来自众多志愿者[3]（某些志愿者由商业公司或者大学支付薪酬，但他们确实是我们的志愿者）。某些贡献的软件由 FSF 拥有版权，有些由编写它们的贡献者拥有版权。

## FSF 拥有版权的 GNU 软件（FSF-Copyrighted GNU Software）

GNU 软件包的开发者可以选择将版权转让给 FSF 或者自己保留。这是他们的选择权利。

如果他们将版权转让给 FSF，该程序成为 FSF 拥有版权的 GNU 软件，此时 FSF 可以强制实行它的许可证。如果他们选择自己保留版权，强制实行其许可证是他们自己的责任。

FSF 不会接受非官方 GNU 软件包的软件的版权转让，这是一条规定。

## 非自由软件（Nonfree Software）

非自由软件是指任何不是自由软件的软件。它的使用、再分发或修改被禁止，或者要求您请求授权许可，或者被严格限制以致于您事实上不可

[1]参见 “GNU 系统概览” 一文，位于 http://gnu.org/gnu/gnu-history.html 以获知更多历史背景。

[2]参见 http://directory.fsf.org。

[3]参见 http://gnu.org/people/people.html。

能自由地进行以上行为。

## 私有软件（或译为“专有软件”，Proprietary Software）

私有软件是非自由软件的另一种叫法。在过去，我们曾将非自由软件进一步细分为“半自由软件”（semifree software），它们允许非商业性的修改和再分发；以及私有软件，它们禁止任何修改或再分发。但我们现在已经放弃了这种区分，并且现在将“私有软件”用作非自由软件的同义词。

FSF 遵循这样的规则：我们不能在自己的计算机上安装任何私有软件，除非暂时性地用于一种特定用途，即编写一个自由软件来取代它。除此之外，我们感觉没有任何可能的借口来安装一款私有软件。

例如，在 20 世纪 80 年代，我们认为在我们的计算机上安装 Unix 是合理的，由于我们需要用它编写一个可以取代 Unix 的自由操作系统。而现在，由于自由的操作系统已经有了，因此这一借口不再适用；我们不会使用任何私有操作系统，并且我们所组装的任何一台新计算机都必须运行一款完全自由的操作系统。

我们并不坚持要求 GNU 的用户或者贡献者也必须严格遵守这条规则。它只是我们对自己制定的规则。但我们希望您也愿意遵循它，为了您自己的自由。

## 免费软件（Freeware）

“Freeware”这一短语没有明确并且公认的定义，但它通常被用于指代那些允许再分发但禁止修改（并且其源代码不可获得）的软件包。这些软件包不是自由软件，因此请您不要使用“freeware”来指代自由软件。

## 共享软件（Shareware）

共享软件是指那些附带有允许人们再分发副本的许可协议的软件，但它宣称任何人如果想要继续使用其一份副本就必须支付一笔授权许可费用。

共享软件不是自由软件，甚至不是半自由软件。有两大原因：

- 对于大多数共享软件，源代码不可获得；因此，您完全不能修改它们；
- 共享软件并不带有允许人们在不支付授权许可费用的条件下复制副本并且安装它们的许可条款，甚至对于那些从事非盈利性活动的个人用户也不提供这样的许可（实际上，人们经常不遵守这种发布条款并且仍然这样做，但许可条款并不允许）。

## 私人软件（Private Software）

私人软件或者定制软件是指专为某一特定用户（通常是一家组织机构或者商业公司）开发的软件。该用户保存和使用它，并且不会以源代码或者二进制格式将其对公众发布。

一款私人软件可以是自由软件（尽管这对他人帮助不大），如果它的终极用户拥有四项基本自由。特别地，如果其用户对私人软件拥有完整的权利，该软件就是自由的。然而，如果用户向其他人分发副本但不随之提供四项基本自由，那些副本就不是自由软件。

自由软件关乎的是自由，而非可获得性。总的来说，我们不认为开发一款软件但不发布它是一件坏事。确实有这样的情况，若一款软件举足轻重，以至于人们要争论，独占这款软件而拒绝对公众发布是在对全人类犯错。然而，这样的情况毕竟罕见。大部分软件并非如此生死攸关，拒绝将它对公众发布并不是特别地坏。因此，开发私人软件或者定制软件的实践与自由软件运动的原则之间并无冲突。

几乎所有受雇佣的程序员都在开发某种定制软件；因此大部分编程工作就是或者可以是一种与自由软件运动相容的方式完成的。

## 商业软件（Commercial Software）

“商业” 和 “私有” 并不等同！商业软件是由企业作为其业务的一部分所开发的软件。大部分商业软件是私有软件，但确实也有商业化的自由软

件，并且也有非商业化的私有软件。

例如，GNU Ada 编译器是由一家商业公司开发的。它一直以 GNU GPL 的条款发布，每一份副本都是自由软件；但它的开发者贩卖其支持合同。当它们的销售员同潜在的客户交谈时，有时客户会说 “我们更加信赖商业化的编译器”。销售员回答说 “GNU Ada 确实是一款商业化的编译器；它恰好也是一款自由软件”。对于 GNU 计划，上述优先级应该是另一种顺序：重要的是 GNU Ada 是一款自由软件；它同时也是一款商业软件只是细节。然而，由于其作为一款商业软件而带来的额外发展是大有裨益的。请您帮助我们宣传这种认识：自由的商业软件是可能的。您可以努力通过在您想要表达 “私有” 的时候避免使用 “商业” 来达到这样的效果。

# 为什么说开源漏掉了自由软件的要点



当我们说软件“自由”的时候，我们意指它尊重用户最根本的自由：即运行、学习和修改软件，或者重新发布软件副本（无论是否修改过）的自由[1]。这是一个关于自由的问题，而非价格。因此我们应该理解为“言论自由”，而不是“免费啤酒”。

这些自由极其重要。它们是必要的，不仅是因为它们满足用户的个体利益，更是因为它们促进社会的团结——也就是分享与协作。由于我们的文化和生活变得越来越数字化，因此这些自由就变得越来越重要了。在一个由数字化的声音、图像和文字组成的世界里，自由软件正在逐渐地趋近于通常意义上的自由。

现在全世界有数以千万计的人使用自由软件；印度和西班牙的一些学校正在教授所有的学生使用自由的 GNU/Linux 操作系统[2]。但是，大多数用户都没有听说过我们开发这个系统以及建立自由软件社区的伦理原因，因为现在这个系统和社区更多地被描述为“开放源代码”（简称“开源”），并将其归属为另一种不同的、几乎不提及自由的哲学。

自 1983 年以来，自由软件运动一直为计算机用户的自由而战。1984 年，我们发起开发自由的操作系统 GNU，避免那些否定用户自由的非自由操作系统。在（二十世纪）八十年代，我们开发出了这个系统的重要组件，还有 GNU 通用公共许可证（GNU GPL）——一个专门用于保护所有用户自由的许可证。

并不是所有的自由软件用户和开发人员都认同自由软件运动的目标。1998 年，一部分人从自由软件社区中分裂出去，并且开始了以“开源”为旗号的运动。提出“开源”这个说法原本是为了避免“free software”可能产生的一些误解，但却很快就与自由软件运动的哲学观点分道扬镳了。

[1] 有关自由软件的完整定义，参见《什么是自由软件?》 (p. 1) 一文。

[2] 关于操作系统可参见《Linux 和 GNU 操作系统》 (p. 75) 一文

一些“开源”的支持者认为它是“自由软件的商业市场运动”，因为突出现实利益能吸引商业执行部门，并且避免纠缠于他们不想听的是非观点。其他支持者甚至断然否定自由软件运动的伦理和社会价值观。不管他们的观点是哪一种，“开源”运动中都没有谈起或提倡这种价值观。“开源”这个说法很快便与一些现实想法和论点勾连在一起了，比如构建强大而稳定的软件。从那时起许多“开源”的支持者也因此而加入其中，并持同样观点。

自由软件和“开源软件”这两个说法其实说的几乎都是同一类软件，但是它们所秉持的价值观是根本不同的。开源是一种开发方法论，自由软件是一场社会运动。对自由软件运动而言，自由软件是一种道义责任，尊重用户最根本的自由。而另一方面，开源的哲学更关注如何让软件“更好”——只关心实用价值。开源认为，非自由软件只是手边问题的欠妥解答，关于“开源”的大多数讨论并不关注对与错，只关注知名度和成功[1]。

然而自由软件运动中，非自由软件是一个社会问题，解决方案就是停止使用并转向自由软件。

“自由软件”，“开放源代码”，如果指代的是同样（或几乎相同[2]）的软件，你用什么名字很重要吗？是的，因为不同的文字传达了不同的理念。虽然现在一个以其他方式命名的自由程序能给你同样的自由，但是可持续地构筑自由依赖于教育人们珍惜自由。如果你愿意提供帮助，最基本的就是提到“自由软件”。

自由软件运动中我们并不把开源阵营看作敌人；敌人是专有（非自由）软件。但是我们希望人们知道我们代表自由，所以我们不能接受把我们和开源支持者混为一谈。

[1]一个典型的例子，比如 Jay Lyman 的文章“Open Source Is Woven Into the Latest, Hottest Trends”（2013 年 9 月 12 日）http://www.linuxinsider.com/story/Open-Source-Is-Woven-Into-the-Latest-Hottest-Trends-78937.html

[2]参见“How Free Software and Open Source Relate as Categories of Programs”一文，位于 http://gnu.org/philosophy/free-open-overlap.html

## 自由软件和开放源代码的实际差别

实际上，开源的标准比自由软件要弱一些。据我们所知，目前所有的自由软件都算是开源软件。几乎所有开源软件也都是自由软件，但也有例外。首先是一些开源许可证过于严苛，因此并不能认定为是自由许可证。比如 “Open Watcom” 的许可证就不是自由的，因为它不允许将修改后的软件私人使用。幸运的是，只有很少一些程序使用这样的许可证。

其次，实践中更重要的一点是，很多产品包含了计算机可以检查可执行程序签名的方法，以便阻止用户安装其它版本的可执行程序。只有这个有特权的公司可以构建能在设备上运行，或允许访问其全部功能的可执行程序。这种设备我们称之为 “暴君”（Tyrants），而这种行为称之为 “tivo 化”（tivoization），以我们第一次看见这样做的产品（Tivo）而得名。即使可执行文件是从自由的源代码构建出来，用户却无法运行修改版，所以这个可执行文件也是非自由的。

开源标准并没有认识到这个问题。它只关注源代码的许可证问题。因此，从比如像 Linux 这种开源且自由的源代码构建，却不可修改的可执行文件只是开源软件但不是自由的。很多 Android 产品包含了这种基于 Linux 的非自由 tivo 化的可执行文件。

## 对 “自由软件” 和 “开放源代码” 的常见误解

术语 “free software” 容易被误解：一个下意识的理解，“你可以无偿得到的软件”，而其真正含义 “给予用户真正自由的软件” 同样与其名称相符。我们通过广为散播自由软件的定义和强调 “理解 free 为 ‘自由的言论’，而并非 ‘免费啤酒”’，来解决这个问题。这不是一个完美的解决方法；它不能完全排除这个问题。如果它没有其它问题的话，也许一个没有歧义且正确的说法会更好一些。

然而很不幸，英语里所有可选的说法都存在各种问题。我们曾看到过很多这种情况，人们收到很多建议，但没有一个是明显 “正确” 的选择。（例如，在某些环境中法语和西班牙语单词 “自由” 是有效的，但印度

人却完全不认可。）每个提交的用来替代“自由软件”的词语都存在一些语义问题——这也包括“开源软件”。

“开放源代码软件”的官方定义（由开放源代码促进会公布，这里引用太长了[1]）是间接引述自我们的“自由软件”的标准。与自由软件不同；在一些方面稍微有点宽松。尽管如此，他们的定义大多数情况还是非常接近我们的定义的。

然而，“开源软件”这个概念表面的意思是“你可以看源代码”，因此多数人似乎都把这当作它的真正意义。这个标准比自由软件更脆弱，甚至比开源的官方定义都脆弱。这样就包括了很多既不自由也不开源的软件。

由于“开放源代码”的表面含义并非拥护者们所设想的，结果大多数人曲解了这个说法。以下是撰稿人 Neal Stephenson 所定义的“开源”：“Linux 是‘开源’软件的意思，简单地说，就是任何人都可以得到源代码的副本[2]。”我并不认为他有意抵触或者争论官方的定义。我想他只是用了英语的习俗来提出开源的含义。堪萨斯洲曾经发表过类似的定义：“使用开源软件（OSS）。开源软件是源代码可自由、公开使用的软件，但特定的许可证规定了人们可以用代码做些什么[3]”。

《纽约时报》的一篇文章引申了开源的意思，认为开源就是让用户做测试[4]——让一小部分用户测试产品的早期版本并给出反馈——而这已经是专有软件数十年来一直在做的事情。

[1]全部定义可参见 http://opensource.org/docs/osd

[2]Neal Stephenson, In the Beginning...Was the Command Line（纽约，HarperCollins 出版，1999 年），p 49。

[3]堪萨斯州全州技术架构，“信息架构”，version 8.0, 20.3.8, 2011 年 10 月 11 日。https://web.archive.org/web/20001011193422/http://da.state.ks.us/ITEC/TechArchPt6ver80.pdf

[4]Mary Jane Irwin，“The Brave New World of Open-Source Game Design”（开源游戏设计的勇敢新世界），纽约时报，在线版，2009 年 2 月 7 日。http://www.nytimes.com/external/gigaom/2009/02/07/07gigaom-the-brave-new-world-of-open-source-game-design-37415.html

这个说法还被引申为设计和发布没有专利的产品[1]。无专利的设备对社会而言确实值得称赞，但是 “源代码” 并不属于这些。

开源的支持者们试图通过指出他们的官方定义来应对这个问题，但收效甚微，甚至还不如直接用 “自由软件”。“free software” 这个说法天生就只有两种含义，其中一个是我们设想的含义，所以领会了 “言论自由，而并非免费的啤酒” 的人就不会曲解它。但是 “开源” 只有一个天然的含义，但是这个含义却和它的支持者们预想的不同。所以不存在一个一劳永逸的方法来解释和证明其官方定义。这就产生了更大的歧义。

另外一个关于 “开源” 的误解认为它的意思是 “不使用 GNU GPL”。这还会产对一个对 “自由软件” 的误解，认为自由软件就是 “GPL 许可证保护的软件”。这样都不对了，因为 GNU GPL 被当作是一个开源许可证，而大多数开源许可证被当作是自由软件许可证。而且还有很多不是 GNU GPL 的自由软件许可证[2]。

“开放源代码” 甚至已经被引申和应用到了其他活动中，比如政府、教育和科学，那些没有源代码的，与软件的标准并不相关的领域。这些活动唯一的共同点是他们只是以某种方式邀请人们参加。而目前为止这个概念只是被引申成了 “参与” 或 “透明” 而已，甚至更浅。在最坏的情况下，它已经成为一个空洞的流行语[3]。

## 不同的价值观会得出类似的结论……但并非总是如此

在二十世纪六十年代，激进组织因为派别纷争而获得一些声誉：一些组织因为对策略上细节的分歧而导致分裂。尽管依然有着相似的基本目标

[1] Karl Mathiesen 和 Tess Riley，“Texas Teenager Creates $20 Water Purifier to Tackle Toxic E-Waste Pollution”（德州青少年创建 $20 水净化器，以解决有毒电子废物污染环境），2015 年 8 月 27 日。http://theguardian.com/sustainable-business/2015/aug/27/texas-teenager-water-purifier-toxic-e-waste-pollution

[2] 参见 “Various Licenses and Comments about Them” http://gnu.org/licenses/license-list.html

[3] Evgeny Morozov，“Open and Closed”（开放和封闭），2013 年 3 月 16 日，http://www.nytimes.com/2013/03/17/opinion/sunday/morozov-open-and-closed.html

和价值观，但是分裂出来的两个组织往往把对方当作敌人。右翼利用了这一点，来抨击整个左派。

有些人试图把我们和开源之间的分歧比作这些激进组织之间的分歧，以此来诋毁自由软件运动。但是他们完全错了。我们虽然和开源阵营在根本目标和价值观上无法达成共识，但是他们和我们的观点在许多情况下是一样的，都引领了实际行动——比如开发自由软件。

结果是，来自自由软件运动和开源阵营的人们经常在一些诸如软件开发的实际项目中协同工作。不同的哲学观点能够如此频繁地激励不同人参与到相同的项目中，这是非同寻常的。不过因为这些观点相差很大，所以也存在导致行为相差很大的情况。

开放源代码的理念是允许用户修改和重新发布软件以使它更强大、更可靠。但是这些并不能得到保证。专有软件的开发者们并不一定就不称职。有时他们开发的程序也是强大而且可靠的，尽管它并没有尊重用户的自由。自由软件积极分子和开源爱好者对此的反应是非常不同的。

一个从未受到自由软件思想影响的纯开源爱好者，会说："我非常惊讶你们不使用我们的开发模式，居然能让程序工作得这么好，但是你们确实做到了。我怎样才能得到一份副本呢？"。这种态度会鼓励那些夺走我们自由，或有损自由的项目。

自由软件积极拥护者会说："你的程序很吸引人，但我更珍视我的自由。所以我抵制你的程序。我会以其他方式努力，支持一个开发自由替代品的项目。"如果我们珍视我们的自由，我们就能行动起来保卫它。

## 强大而可靠的软件可能是件坏事

我们想让软件变得强大而可靠的想法，来自"软件是被设计用于服务其用户"的假设。如果软件强大而可靠，这意味着它将更好地服务于用户。

然而只有在软件尊重了用户的自由时，才能说它服务于用户。如果软件被设计的目的是使用户枷锁缠身会怎样呢？那样的话，强大仅仅意味着枷锁套得更紧，而可靠则使枷锁更难去除。恶意的功能，比如针对用户的间谍行为、限制用户、后门和在专有软件里很常见的强制性升级，甚至有

些开源的支持者也想实现这些。

在电影和唱片公司的压力下，个人使用的软件逐渐被有意地设计成能够限制用户的软件。这种恶意的特征被称为 DRM 或者数字限制管理（Digital Restrictions Management，参见 DefectiveByDesign.org），而这正是与自由软件所要提倡的自由精神相对立的。并且不仅仅在精神上：因为 DRM 的目标是践踏你们的自由，DRM 的开发者们试图使你修改实现了 DRM 的软件变得困难、或者不可能，甚至非法。

然而，一些开源支持者提出了所谓“开源 DRM”软件。他们的想法是，通过公开并允许人们修改那些被设计成用来限制你访问加密媒体的程序的源代码，生产更加强大、可靠的软件来限制像你这样的用户。然后，再将这些程序放到那些不允许你修改的设备中发布。

这种软件或许是开放源代码的，而且使用了开源的开发模式，但它并不是自由软件，因为并没有尊重实际运行它的用户的自由。如果开源开发模式通过开发更强大和更可靠的软件来限制用户而获得了成功，这将使它变得更糟糕。

## 对自由的敬畏

起初，开源阵营从自由软件运动中分裂出来，是因为“自由软件”的伦理观念使一些人不安。这的确是事实：谈论关于自由、道义问题、可靠性和方便性，提醒人们去考虑那些可能被他们忽略的问题，比如他们的行为是否是道德的。这会引起不舒服，而且一些人甚至会回避它。但这并不意味着我们应该停止讨论这些事情。

然而，这正是开源的引领者决定要做的。他们希望停止对伦理规范和自由问题的讨论，而仅仅关注某个自由软件直接的实际利润，这样，他们或许就能更有效地将软件“卖”给某些用户，特别是商业用户。

用他们自己的话说，这种方法已经被证实是有效的。开源的修辞方式已经吸引了许多商业和个人用户来使用、甚至开发这些壮大我们社区的自由软件——但仅仅是在表面的、实用上的。开源哲学，和它纯粹的实用主义价值观，阻碍了人们对自由软件更深层观点的理解；它将许多人带进了

我们的社区，但却没有教他们去捍卫社区。就目前的现状来看还是不错的，但是它不足以捍卫自由。要吸引自由软件用户，需要让他们成为个体自由的捍卫者。

迟早这些用户会因为一些实际利益而转向专有软件。无数的公司试图提供这种诱惑，有些甚至提供免费副本。为什么用户会减少？只有他们领悟了自由软件所赋予用户的自由，去珍惜自由的价值，而不只关心特定自由软件在技术上和实用性上的方便性价值，他们就会明白了。为了传播这种思想，我们必须谈论自由。对商业采取一定的沉默方式对于社区来说是有益的，但是，如果大家普遍地把对自由的热爱看作是一种怪癖，那将是很危险的。

这种危险确实已经发生。大多数投入自由软件的人，特别是发行者，很少谈论“自由”——通常是因为他们追求“更容易被商业接受”。几乎所有的 GNU/Linux 操作系统发行版都在基本的自由系统上加入了专有软件包，而且他们让用户认为这是优点，而不是与“自由”思想背道而驰的。

加入专有版权的软件和部分非自由的 GNU/linux 发行版之所以找到了孕育的温床，是因为我们的许多社区没有在他们的软件上贯彻自由。这并非巧合。大多数 GNU/Linux 的用户是通过讨论“开源”而引入到这个系统的，而这些讨论却并没有将“自由”作为目标。那些不支持自由的行为和不探讨自由的言论并肩而行，相互促进。为了避免这种趋势，我们需要更多地，而不是更少地，来谈论“自由”。

## “FLOSS”和“FOSS”

为了在自由软件和开源之间保持中立，常用“FLOSS”和“FOSS”[1]这种说法。如果你的目标是中立性，那么“FLOSS”更好，因为这真的很中立。如果你更想表达自由，那么用中立的说法并不好。站在自由的一边，让人们看到你对自由的支持。

[1]可参见《应避免使用（或慎用）的词语》一文的“自由和开源软件（FLOSS）”一节 (p. 116) 和《FLOSS 和 FOSS》http://www.gnu.org/philosophy/floss-and-foss.html

## 对立的传播

“自由” 和 “开放” 在传播上是一组对立的概念。“自由软件” 和 “开源” 虽然是不同的思想，但大多数人认为这两个在概念上是相同的。当人们变得习惯于说和思考 “开源”，这对他们获取自由软件运动的理念和思考是一个障碍。如果他们已经将 “开放” 和我们联系起来，在他们认识到我们代表其他事情之前，也许需要冲击他们的理性。任何推广 “开放” 的活动实际上都会更深的让自由软件运动隐藏在幕后。

因此，自由软件的活跃者经过深思熟虑以后决定拒绝参加那些标榜自己是 “开放” 的活动。即使活动本身并不错，但每次为 “开放” 多做的贡献，都会对另一方作出伤害。也有很多很好的活动是 “自由” 或 “解放” 的。而参与这些活动，会对这些项目有小小的额外好处。既然有这么多有益的项目可选，为什么不选能更有好处的呢?

## 结论

开源的拥护者把新用户引入我们的社区，我们这些自由软件支持者必须做更多地工作来让这些新用户关注自由的问题。我们必须大声且更大声的高呼：“是自由软件给予了你们自由！” 每当你说 “自由软件”，而不是说 “开源” 的时候，你就是在为我们助阵。

### 备注

Karim R. Lakhani 和 Robert G. Wolf 的论文 “Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects”（剑桥：MIT 出版社，2005）http://ocw.mit.edu/courses/sloan-school-of-management/15-352-managing-innovation-emerging-trends-spring-2005/readings/lakhaniwolf.pdf，在探讨自由软件开发者的动机时，指出相当一部分人被 “软件必须是自由的” 的观点所鼓舞。这里忽略了他们调查的对象是 SourceFourge 的开发人员这一事实，而该站点并不支持这是个伦理问题的观点。

# 您说过“知识产权”吗？这是一种迷惑的幻景



有这样一种时兴的趋势：将版权、专利、商标——三种相互独立而又不同，并且涉及三种独立而又不同的法律概念——再加上其他几十种法律一股脑地装进一个罐中，并称之为“知识产权”。这种涵义扭曲而又使人混淆的概念变得普遍流行并非偶然。从这种混淆中得利的商业公司推动了这种混淆。要想避免这种混淆，最明了的方式就是完全拒绝这一概念。

根据现在供职于斯坦福法学院的 Mark Lemley 教授的观点，对“知识产权”这一概念的广泛使用变得流行是始于 1967 年世界“知识产权”组织（WIPO）的成立，而这一概念只是在最近数年才变得真正普遍（WIPO 是联合国的正式机构之一，但它实际上代表的是版权、专利及商标持有人的利益）。这一概念的广泛使用始于 1990 年前后。

这一概念本身具有一种不难觉察的偏见：它暗示人们在思考版权、专利和商标的时候同实物的产权进行类比（这种类比与版权法、专利法、商标法存在矛盾，但只有专业人士了解这种矛盾）。这些法律实际上并不像物权法，但这一概念的滥用使得立法者将它们修改得越来越像后者。由于这种转变正是那些行使版权、专利和商标权力的商业公司所求之不得的，于是这种由“知识产权”这一概念所带来的偏见迎合了它们。

这种偏见已经足以成为拒绝这一概念的理由，并且人们经常建议我为这一整体类别起个别的名字——也有人提出了他们自己起的名字（通常是幽默的）。这些名字包括“小恶魔”（IMPs，意为 Imposed Monopoly Privileges，即“强制垄断特权”）以及“魔像”（GOLEMs，意为 Government-Originated Legally Enforced Monopolies，即“政府主导的合法强制垄断”），还有人说“专属权利王国”（exclusive rights regime，ERR），但将“限制”（restriction）指代为“权利”也是一种双关。

有些别名可能确实是更好的称谓，但是，使用它们中的任何一个来指

代 “知识产权” 则是一种错误。由于其他名字都未能揭露这一概念的深层次问题：过度广义化。并不存在诸如 “知识产权” 这样高度统一的事物——它是一种幻景。人们之所以会误认为 “知识产权” 是一种合乎逻辑的分类，原因仅仅在于这一概念的广泛使用对于人们理解相关法律产生了误导。

“知识产权” 这一概念充其量只是把不同类的法律混装在一起的杂物容器。非法律专业的人们在听到这一适用于不同法律的概念时，倾向于想象它们相似地基于某种共同的原则和功能。

然而，事实远非如此。这些法律是独立起草的，以不同方式演进的，覆盖不同的行为，拥有不同的规则，并且由此带来了不同的公共政策问题。

例如，版权法被设计为提升作者身份和艺术，并且覆盖了作品表达的细节。专利法的本意是促进发表有用的想法，其代价是赋予想法的发表者关于此想法的暂时垄断权——这种代价对于某些领域是值得付出的，而对于其他领域则不然。

与之不同的是，商标法的本意并非提升任何行为方式，而只是让消费者知道他们所购买的东西是什么。受 “知识产权” 概念影响的立法者却将它变成了一种刺激广告行为的模式。而这些只是他们所谈论的众多法律里的三种。

由于这些法律是独立发展的，它们在任何细节上都是不同的。它们的基本目的和方法也是不同的。因此，如果您知道某些关于版权法的细节，并且您明智地想象专利法在这些方面与之不同，您将几乎不会犯任何错误!

事实上，您所遇到的那些基于 “知识产权” 所阐述的广义化的陈述几乎全部是错误的。例如，您可能会看到这样的宣传：“知识产权” 的目的是 “促进创新”，但这仅仅适用于专利法，也许还有植物品种垄断权。版权法与创新没有关系，一支流行歌曲或是一部小说即使没有任何创新也可以获得版权。商标法和创新也没有关系；如果我自营一家茶店并将其命名为 “RMS 茶”，它将成为合法的商标，即使我以和任何其他人相同的方式贩卖茶饮。商业机密法和创新也没有关系，即使有，也只是稍微沾边；我的茶店客户清单可以成为商业机密，但这与创新没有关系。

您也会见到这样的论断，“知识产权” 与创造力有关，但事实上这只适

用于版权法。除了创造力之外，还需要其他东西才能打造出一件可获专利的发明。商标法和商业机密法与创造力没有关系；“RMS 茶” 的名字并不体现任何创造力，我的茶店客户机密清单也和创造力无关。

当人们谈论 “知识产权” 时，他们通常真正想要表达的是或多或少一系列法律。例如，富国经常向穷国强行施加不平等法律以榨取其钱财。这些不平等法律中的一些也位于它们所称的 “知识产权法” 之中，而其他一些则不是；然而，对这一行为的批评通常也会使用 “知识产权” 这一标签，由于他们对此很熟悉。通过使用这一概念，他们对问题的本质进行了错误解读。最好能够使用一个准确的术语，诸如 “立法殖民”，这样的术语可以切中要害。

非专业人士并不是被 “知识产权” 这一概念所迷惑的唯一人群。即使是讲授这些法律的法学教授也会被这一概念的迷惑性所引诱和误导，并且做出与其所知的事实相违背的广义化陈述。例如，一位教授在 2006 年写道：

> 与现在效力于 WIPO 的继承者不同，美国宪法的制定者对知识产权持有原则坚定、促进竞争的态度。他们知道权利可能是必需的，但是……他们束缚了国会的双手，并且以多种不同的方式限制它的权力。

这段陈述所指的是美国宪法第 I 章，第 8 条，第 8 款，它许可了版权法和专利法。但这一条款与商标法、商业机密法或是任何其他不同的法律无关。而 “知识产权” 这一概念诱使那位教授做出了错误的广义化。

“知识产权” 这一概念也会导致过度简单化的思考。它促使人们仅仅关注这些不同类法律所拥有的，微小的，形式上的共同点——于是他们为某些群体创造了人为的特权——而无视那些构成它们基础的细节：每种法律为公众施加的特别的限制及其带来的后果。这种对形式的过度简单化关注助长了一种应对所有这些问题的 “经济主义” 方式。

如同它经常发挥作用的方式，经济在这里成为了未经检验的设想的载体。这包括了关于价值的设想，诸如产量与之有关，而自由和生活方式与之无关；以及那些经事实证明大部分是错误的设想，诸如音乐版权是在支

持音乐家，或是药品专利是在支持救生药物的研究。

还有一个问题是，由于“知识产权”这一概念在很大程度上的含混不清，由不同法律引起的特定问题变得几乎不可见。这些特别的问题源自每一条特定的法律——而这恰恰是“知识产权”这一概念试图诱导人们所忽略的。例如，版权法所带来的问题之一是音乐分享是否应该被允许；而专利法与此无关。专利法引起的问题诸如穷国是否应该被允许生产救生药物并且以较低的价格出售以挽救生命，而版权法却与此无关。

这些问题究其本质都不是纯粹的经济问题，它们在非经济层面有着显著差别；基于经济方面的肤浅的过度广义化概念来试图思考这些问题，意味着忽略它们之间的区别。将两种法律一同装进“知识产权”的罐中将会阻碍对其任意一方清晰的思考。

因此，任何关于“知识产权问题”的观点以及任何关于这种假想类别的广义化几乎一定是愚蠢的。如果您认为所有这些法律都是一个问题，您就会倾向于从过于笼统的过度广义化概念中选择一种作为您的观点，而这些都是有百害无一利的。

如果您想要对专利法、版权法、商标法或是其他不同法律所引起的问题进行清晰的思考，第一步就是忘记将它们不分青红皂白地放在一起的想法，并且将它们作为独立的问题区别对待。第二步是拒绝“知识产权”这一概念所提示的狭隘视角和过度简单化的情景。只有对每个问题进行独立且完整的思考，您才能有机会清晰地思考它们。

## 注记

- 参见“The Curious History of Komongistan (Busting the Term ‘Intellectual Property’)”一文，位于 http://gnu.org/philosophy/komongistan.html。

- 可以用非洲各国之间的关系类比这些法律之间的关系，并且“非洲”是一个合乎逻辑的地理概念；然而，谈论“非洲”而非某个特定的国

家将会引发大量混乱[1]。

- Rickard Falkvinge 也支持拒绝使用 “知识产权” 这一概念[2]。

[1]Nicolas Kayser-Bril, “Africa Is Not a Country”，于 2014 年一月 24 日发表于 http://theguardian.com/world/2014/jan/24/africa-clinton。

[2]“Language Matters: Framing the Copyright Monopoly So We Can Keep Our Liberties”，于 2013 年七月 14 日发表于 http://torrentfreak.com/language-matters-framing-the-copyright-monopoly-so-we-can-keep-our-liberties-130714。

# 为何称之为诈骗（Swindle）？



我有意地用具有批判性的名字来称呼龌龊的东西。我将苹果迫使用户屈从于它们的计算机称为 “iThings”，而将亚马逊虐待读者的电子书阅读器称为 “诈骗”（Swindle）。有时我将微软的操作系统称为 “Losedows”，而将微软的第一款操作系统称为 “MS-Dog”[1]。当然，我如此做是为了表达自己的不满以及作为消遣。然而这种消遣并不是个人层面的，它服务于一项重要目的。通过嘲讽我们的敌人，我们可以将幽默的力量注入我们的事业。

扭曲名字是一种不尊重对方的行为。如果我们尊重这些产品的制造商，我们将会使用它们所选定的名字……并且那将是恰如其分的。这些恶意的产品只配得到我们的鄙视而非我们的尊重。每一款私有软件有迫使其用户屈从于某些实体的权力，但如今最常用的私有软件已经不限于窥探用户和限制用户，甚至是任意摆布用户：趋势是这些产品正在变得越来越龌龊。它们理应被消灭干净，其中带有数字限制管理（DRM）的产品应该被判定为非法。

当我们谈论它们的时候，我们应当表达出我们对它们的谴责，那么，还有什么方式比扭曲它们的名字更容易呢？如果我们不如此做，就很可能在称呼它们的同时未能表达出我们的谴责。例如，当它们出现在某些其他话题之中时，下笔千言来解释它们为什么是坏的，就会显得离题万里。

点名谈论这些产品但未能表达出我们的谴责，将会产生使其合法化的反效果，这是与它们所应得的谴责正好相反的。

商业公司将为其产品选择名字作为其市场计划的一部分。它们通常选择那些它们认为人们更倾向于重复的名字，然后在市场营销活动中投入数以百万计的资金以迫使人们重复并且思考这些名字。通常，这些营销活动的意图是促使人们基于其表面的诱人之处来崇拜这些产品，而忽视它们所

[1] 采取行动反对 iThings，位于 u.fsf.org/ithings；反对亚马逊诈骗（Swindle）位于 u.fsf.org/swindle 和 u.fsf.org/ebookslist；以及反对 Windows，位于 upgradefromwindows.org。

造成的危害。

每当我们用制造商所使用的名字来称呼这些产品的时候，我们是在为它们的市场营销活动做贡献。重复这些名字是对其产品的积极支持；而扭曲它们的名字则是在拒绝给予这些产品支持。

除了产品名称以外，其他术语也会带来类似的问题。例如，数字限制管理（DRM）指的是为了某些其他人的利益而制造技术产品以限制用户。这种不可原谅的行径只配得到我们的怒火，直到我们将其彻底消灭。很自然地，那些应该为此受到谴责的人们基于他们看待此问题的立场为它起了这样一个名字：数字版权管理。这个名字是一场致力于赢得从各级政府到万维网联盟（W3C）的各种实体的支持的公共关系运动的基础。[1]

使用它们的称谓就是支持他们的立场。但如果那不是您的立场，为什么要给予它们默许的支持呢？

我们站在用户的立场上，并且从用户的观点出发，那些恶意功能所管理的不是权利而是限制，因此我们称之为“数字限制管理”。

这两种称谓都不是中立的：在从中选择一种称谓的同时，您选择了一种立场。请您选择用户的立场并且将其发扬光大。

有一次，我的演讲听众中的某人宣称“数字版权管理”才是 DRM 的官方名称并且是唯一可能的正确名称，由于这是它的第一个名字。此人争论以此出发，我们称之为“数字限制管理”是错误的。

制造产品或者从事商业行为的实体通常会在我们尚未获知其存在的时候就为其选定了一个名字。如果它们的暂时优势地位迫使我们使用它们所起的名字，它们就将在无形之中获得额外的优势，这基于它们的资金、媒体影响力以及技术地位。我们将会不得不言不由衷地对抗它们。

有人不喜欢扭曲名字的方式并且认为这是“幼稚”或者“不专业”的。他们的意思是，这听起来并不显得古板而缺乏幽默——后者是一件好事，由于在我们的立场上，如果我们试图表现得“专业”，就不能拥有欢乐。反抗压迫远比从事专业工作更加严肃，因此我们必须要有一些诙谐的放松。这需要真正的成熟，这里包括一些童稚，而非仅仅是“表现得像个成年人”。

[1]参见 https://u.fsf.org/drm 以获得关于数字限制管理（DRM）的更多信息。

如果您不赞同我们所选定的带有讽刺意味的谐音或形近单词，您可以创造您自己所喜好的名字，越多越欢乐。当然，还有其他方式以表达谴责。如果您想要表现得“专业”，您可以采用其他方式将其呈现。它们能够表达您的用意，但需要更多时间和努力，特别是当您不想使用嘲讽的手法时。注意到这一点，您的谴责才不会有所欠缺；不要让关于反对这种“题外话”的压力迫使您未能对您所提到的那些龌龊的东西进行足够的批判，因为这将会起到使其合法化的反效果。

# 应避免使用（或慎用）的词语，由于它们是不公正的或者引起混淆的



有些单词或短语是我们建议避免使用或者避免在特定上下文或应用场景使用的。其中的一些词语具有歧义或误导性；而其他一些词语则预设了某种我们所不赞同的观点，并且我们希望您也不会同意那些观点。（参见《自由与非自由软件的分类》 (p. 80) 和《为何称之为诈骗（Swindle）?》 (p. 104) 两篇文章。）

## 可获得性（Access）

有一种普遍的误解认为，自由软件意味着公众必须可以获得某一程序。这并非自由软件的本意。

自由软件的准则[1]并不是关于谁能够获得某一程序的；四项基本自由所关注的是拥有它的一份副本的用户可以用它做什么。例如，自由之二称该用户拥有为其复制一份副本并且送给或卖给您的自由。但是任何用户都没有为您复制副本的义务；并且您也没有权利强行要求任何用户为您提供副本。

特别地，如果您为您自己编写了一个程序并且完全不向任何人提供副本，该程序也是自由软件（尽管是在一种平凡的意义上），由于您（作为拥有它的唯一用户）拥有四项基本自由。

事实上，当很多用户拥有某一程序的副本时，某些人确实会将其发布到互联网上，并且供其他人访问。我们认为人们应当如此，如果该程序确实有用。但这并非自由软件的强制要求。

确实有一种特殊情况，此时是否拥有可获得性与自由软件直接相关：GNU 通用公共许可证（GNU GPL）允许赋予一位特定用户下载程序源代

[1] 参见《什么是自由软件?》 (p. 1) 以获得自由软件的完整定义。

码的访问权，作为以实物载体的形式为该用户提供源代码副本的替代方案。这条规则也适用于用户已经拥有一份非源代码形式的副本这一特例。

## 选择（Alternative）

我们从不将自由软件描述为私有软件之外的选择，由于这一单词假设所有 “选择” 都是合法的，并且每增加一种 “选择” 都对用户有利。事实上，这一单词假设自由软件应该和那些不尊重用户自由的软件共存。

我们坚信以自由软件形式发布是发布软件以供他人使用的唯一符合伦理的方式。而其他方式，无论是私有软件还是 “作为软件替代品的服务”（SaaSS）都会迫使用户屈从[1]。因此我们认为不向用户提供这些自由软件之外的 “选择” 才是好的。

## BSD 风格（BSD-Style）

“BSD 风格许可证” 这一表述将会引起混淆，由于它将具有显著不同的许可证混为一谈[2]。例如，最初的 BSD 许可证由于带有广告条款从而与 GNU GPL 不兼容，但是改进过的 BSD 许可证与 GPL 兼容。

为了避免混淆，最好在谈论中明确给出许可证的名字[3]并且避免使用含混不清的短语 “BSD 风格”。

## 闭源（Closed）

将私有软件描述为 “闭源” 很明显是与 “开源” 概念相对。在自由软件运动中，我们不希望被别人同开源阵营混为一谈，因此我们谨慎地避免使用那种促使人们将我们与他们混在一起的表达方式[4]。例如，我们避免将非

[1] 参见《如今自由软件更加重要》 (p. 32) 和《服务器真正是在为谁服务?》 (p. 302) 以获得更多信息。

[2] 参见 “The BSD License Problem” 一文，位于 http://gnu.org/philosophy/bsd.html。

[3] 参见 “Various Licenses and Comments about Them”，位于 http://gnu.org/licenses/license-list.html。

[4] 参见《为什么说开源漏掉了自由软件的要点》 (p. 90) 。

自由软件描述为 “闭源”，我们称之为 “非自由” 或 “私有”[1]。

## 云计算（Cloud Computing）

“云计算” 这一短语（或者在表示计算的上下文中进一步简称为 “云”）是一种含混不清的市场化流行语，它没有任何合乎逻辑的涵义。它被用于指代一系列不同的活动，其唯一的共同点是它们都使用互联网进行文件传输以外的事情。因此，这一短语是在散布混淆。如果您基于它进行思考，您的思考将会是混乱的。

当您思考或是回应一个由他人提出的并且使用这一概念的论述时，第一步是要澄清话题。该论述是关于什么场景的？适用于此场景的恰当并且清晰的概念是什么？只有当话题被清晰阐述之时，合乎逻辑的讨论才是可能的。

“云计算” 的众多可能的涵义之一是将您的数据存储于在线服务上。在大多数场景中，这是愚蠢的做法，由于这使您暴露在监控之下[2]。

另一种可能的涵义（与上一条有所重叠但又不完全相同）是 “作为软件替代品的服务”（SaaSS），它拒绝了您对于您自己的计算的控制权。您应该从不使用 SaaSS[3]。

另一种可能的涵义包括租赁远程的实体或虚拟服务器。这些实践在某些特定环境下是可以接受的。

另一种可能的涵义是通过您自己的移动设备访问您自己的服务器。这并不会产生特别的伦理问题。

美国国家标准技术研究所（NIST）为 “云计算” 的定义[4]提出了三种场

[1] 参见《自由与非自由软件的分类》以获得关于私有软件的更多信息 (p. 87) 。

[2] John Harris，“Why Hackers and Spooks Want Our Heads in the Cloud”，2011 年四月 25 日，http://guardian.co.uk/commentisfree/2011/apr/25/hackers-spooks-cloud-antiauthoritarian-dream。

[3] 参见《服务器真正是在为谁服务?》 (p. 302) 以获得关于此问题的更多信息。

[4] Peter Mell 和 Anthony Grance，“The NIST Definition of Cloud Computing: Recommendations of the National Institute of Standards and Technology”，NIST Special Publication 800-145 (2011 年九月)，http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf。

景，它们会带来不同的伦理问题：软件即服务（SaaS）、平台即服务（PaaS）、基础设施即服务（IaaS）。然而，这种定义与“云计算”通常的用法并不匹配，由于它并未包括向在线服务中存储数据。由 NIST 定义的 SaaS 在很大程度上与 SaaSS 重叠，后者虐待它的用户，但这两个概念并不等同。

这些不同的计算实践甚至不属于同一个讨论范围。避免使用“云计算”这一带来混淆的概念，最佳方式是不使用术语“云”与计算相关联。只谈论您想要表达的话题，并且以一种具体的概念称呼它。

有趣的是，Larry Ellison，一位私有软件开发者也注意到了“云计算”这一概念的空泛性[1]。但他决定仍然使用这一概念，因为他是一位私有软件开发者，毕竟此人与我们不是由相同理念所驱使的。

## 商业（Commercial）

请不要将“商业”用作“非自由”的同义词。这将两种完全不同的问题混为一谈。

一个程序称为商业软件，如果它是作为一种商业行为而被开发的。一个商业软件可以是自由或非自由的，取决于它的发布方式。类似地，由学校或个人开发的程序也可以是自由或非自由的，取决于它的发布方式。这两个问题——程序由何种实体开发，以及它的用户应当拥有什么自由——是不相关的。

在自由软件运动的最初十年，自由软件包几乎都是非商业的；GNU/Linux 操作系统的组件由个人或者诸如自由软件基金会（FSF）和大学的非盈利性组织所开发。其后，在 20 世纪 90 年代，自由的商业软件开始出现。

自由的商业软件是对我们的社区的贡献，因此我们应当鼓励。但是那些认为“商业”等同于“非自由”的人们倾向于认为“自由商业”的组合是自相矛盾的并且否定其可能性。让我们小心对待，不要以这种方式使用“商业”一词。

[1] Dan Farber，“Oracle’s Ellison Nails Cloud Computing”，2008 年九月 26 日，http://news.cnet.com/8301-13953_3-10052188-80.html。

## 补偿（Compensation）

当与版权联系在一起的时候谈论“补偿作者”将会带来以下两种假设：（1）版权是以作者之名而存在的；以及（2）每当我们阅读任何东西的时候，我们已经欠下了作者一笔债务并且必须补偿。第一个假设是错误的[1]，而第二个假设是不可容忍的。

而“补偿版权持有人”的鬼话在此基础上又附加了一层诈骗：您很可能认为这是在补偿作者，尽管在少数情况下确实是这样，但是绝大多数情况下这是在补偿出版商，正是那些出版商对我们强行施加了不平等的法律。

## 消费（Consume）

消费是指我们对食物所做的事情：当我们消化了它以后，它就不再作为食物继续存在。作为类比，我们可以将相同的词语用于那些我们可以将其消耗殆尽的其他产品。将其用于耐用品诸如衣物或电器是一种涵义的延伸。然而，如果将其应用于发表的作品（计算机程序、保存于盘片或文件中的唱片、保存于纸上或文件中的书籍），这些作品的本质意义是永续保存并且可以被无限次地运行、播放或阅读，这显然就是一种错误。播放唱片或者运行程序这些行为并没有消费它们。

“消费”一词是与不可复制的实物产品的经济层面相关联的，而引导人们在毫无察觉的情况下将这一结论套用到可复制的数字作品上——这种错误正是私有软件开发者（以及其他出版商）所热切希望并且积极推动的。他们的这种扭曲的观点在一篇 Business Insider 文章中暴露无遗[2]。这篇文章还将出版的作品称为“内容”（content）。

与我们“消费内容”相关的狭隘想法为诸如数字千年版权法案（DMCA）这样的法律奠定了基础，它们禁止用户破解数字设备中的数字限制管理

[1] 参见《对版权的错误解读——一系列错误》 (p. 137) 一文以获得关于此问题的更多信息。

[2] Lara O’Reilly，“A Former Googler Has Declared War on Ad Blockers with a New Startup That Tackles Them in an Unorthodox Way”，2015 年六月 18 日，http://uk.businessinsider.com/former-google-exec-launches-sourcepoint-with-10-million-series-a-funding-2015-6?r=US&IR=T。

（DRM）。如果用户认为他们对这些设备所做的事情是“消费”，他们就可以将这种限制视为自然的要求。

它还鼓励人们利用 DRM 对数字唱片的“流媒体”服务进行限制，这种形式也可以认为适用于“消费”这一概念。

为何这种不合理的用法广泛流传？有人也许认为这一概念过于高深；如果它吸引了您，用强有力的理由来拒绝它就会显得更为高深。其他人也许是出于（他们自己的或是其雇主的）商业利益。他们在各种有声望的地方对这一概念的使用将会给人以它是“正确”的错觉。

说“消费”音乐、小说或是任何其他艺术作品就是将它们视为产品而非艺术。如果您不想散播这种态度，您应当努力拒绝对它们使用“消费”这一概念。我们建议您说某人“体验”一件艺术作品或者一件表达观点的作品，以及某人“使用”一件实用作品。

## 消费者（Consumer）

当“消费者”一词被用于指代计算机用户的时候，它被附加了一种我们应当拒绝的假设。其中一些假设来自于使用程序就是“消费”程序这一理念（参见上一条目），这将会引导人们将那些从不可复制的实物产品中得出的结论也强行套用到可复制的数字作品上来。

此外，将软件用户描述为“消费者”指的是这样一种境地，人们被限制为只能在“市场”中可以买到的“产品”中做出选择。这里没有这种理念的容身之地，即用户可以对程序所做的事情直接行使控制权[1]。

为了描述那些并不被限制为被动使用作品的人们，我们建议使用“个人”或者“公民”而非“消费者”。

“消费者”一词所带来的问题已经在此前有所注释[2]。

[1] 参见《如今自由软件更加重要》 (p. 32) 一文以获得关于此问题的更多信息。

[2] Owen Hatherley，“Be a User, Not a Consumer: How Capitalism Has Changed Our Language”，2013 年八月 11 日，http://theguardian.com/commentisfree/2013/aug/11/capitalism-language-raymond-williams。

## 内容（Content）

如果您想要描述一种舒适和满足的感觉，您一定会说 “content”，但如果将这个词语用作名词以指代具有作者权的出版物和作品，这将服务于一种您可能很想避免的态度，这种态度将它们视为商品，其目的是装满包装盒并且用于赚钱。事实上，这贬低了作品本身。如果您不同意这种态度，您可以称之为 “作品” 或者 “出版物”。

那些使用 “内容” 一词的人们通常是那些以作者（它们称之为 “创造者”，creator）之名攫取更多版权权力的出版商。“内容” 一词暴露了它们对于这些作品及其作者的真实态度（参见 Courtney Love 致 Steve Case 的公开信[1]并且在页面中搜索 content provider（内容提供商）。哎呀，Love 女士没有注意到 “知识产权” 一词也是带有偏见和欺骗性的[2]）。

然而，只要其他人仍然使用 “内容提供商” 这一概念，持不同政见者也可以称他们自己为 “恶意内容提供商”。

“内容管理”（content management）这一短语的涵义空泛性无出其右。“内容” 指代 “某种信息”，而 “管理” 在此上下文中指代 “对它们做一些事情”。因此某种 “内容管理系统” 是一种用于对某种信息做出某种事情的系统。几乎所有计算机程序都符合这个定义。

在大多数情况下，这一概念指代用于更新网站页面的系统。对于这种定义，我们推荐 “网站修改系统”（WRS，web site revision system）这一概念。

## 创作共用许可的（Creative Commons Licensed）

对于一篇作品的许可证，最重要的方面是它是否自由。创作共用（CC）发布了七种许可证；其中三种（CC BY、CC BY-SA 和 CC0）是自由的，而其他几种则是非自由的。因此，将一篇作品描述为 “创作共用许可的” 未能

[1]美国摇滚音乐家 Courtney Love 于 2000 年五月 16 日在数字好莱坞在线娱乐会议上的演讲的未编辑抄本可以在此找到：http://www.salon.com/2000/06/14/love_7/。

[2]参见《您说过 “知识产权” 吗？这是一种迷惑的幻景》 (p. 99) 以及本文的 “知识产权（Intellectual Property）” 一节 (p. 118) 获知这一问题的原因。

说明它是否是自由的，并且暗示这个问题并不重要。这一论述也许是准确的，但它对于自由的省略是有害的。

为了鼓励人们关注最重要的区别，务必具体指定哪种创作共用许可证被使用，例如“使用 CC BY-SA 许可证”。如果您不知道一篇特定的作品应该使用哪种许可证，您需要设法获知这一点才能下定结论。

## 创造者（Creator）

当“创造者”一词应用于作者时，这是在暗含着将他们比作神（造物主，deity）。这一概念被出版商所使用，以便将作者的道德高度提升到普通人之上，进而以此为理由赋予他们更多的版权权力，然后出版商就能够以作者之名来行使这种权力。我们建议您仍然说“作者”。然而，在很多情况下，“版权持有人”才是您所真正想要表达的。这两个概念不是等效的，“版权持有人”通常不是作者。

## 数字商品（Digital Goods）

当“数字商品”这一短语被应用于具有作者权的作品副本时，这将它们与实物商品等同起来了——这些实物商品是不可复制，并且因此必须被量产并且销售。这种比喻鼓励人们将他们用于评估实物商品的观点和直觉也套用到评估软件或者其他数字作品上来。这种比喻同样将问题限定在经济方面，其肤浅和有限的价值并不包括自由和社区。

## 数字锁（Digital Locks）

“数字锁”这一短语被某些批评者用于指代数字限制管理（DRM）。这一短语的问题在于它未能批判 DRM 的危害。而那些接受这一概念的人们未能对其进行透彻的思考。

锁并不一定是压迫性的或者坏的。您可能拥有很多把锁，以及它们的钥匙或代码；您可能觉得它们有用或是会带来麻烦，但它们并未压迫您，

由于您可以自己打开或锁上它们。类似地，我们发现加密[1]对于保护我们的数字文件是无价之宝。这也是一种数字锁，并且您拥有其控制权。

而 DRM 就像是由其他人为您安置的锁，他们拒绝给您钥匙——换言之，就像手铐。因此，对于 DRM 的恰当的比喻是 “数字手铐” 而非 “数字锁”。

一系列反对 DRM 的运动为其选择了不明智的术语 “数字锁”；为了让事情重回正轨，我们必须坚持改正这一错误。FSF 可以为一场反对 “数字锁” 的运动提供支持，如果我们同意其基本立场；然而，当我们表示我们的支持的时候，我们将会引人注目地将该短语更改为 “数字手铐”，并且解释为何如此称呼。

## 数字版权管理（Digital Rights Management）

“数字版权管理”（简写为 DRM）是指那些被设计用于对计算机用户强行施加限制条件的技术机制。其中 “权利”（rights）一词的使用是一种鼓吹，其用意是诱使您在不经意间用那些施加这些限制的少数人的观点来看待这一问题，并且忽略被强行施加了这些限制的公众。

较好的替代短语包括 “数字限制管理”（Digital Restrictions Management）和 “数字手铐”（digital handcuffs）。

请您签名支持我们致力于废除 DRM 的运动，它位于 DefectiveByDesign.org。

## 生态系统（Ecosystem）

我们不建议将自由软件社区或是任何人类社区描述为 “生态系统”，由于这一短语暗示伦理评价的缺失。

“生态系统” 这一短语暗含着建议这样一种不加批判地服从态度：不要

[1] Cory Doctorow, “Encryption Won’t Work If It Has a Back Door Only the ‘Good Guys’ Have Keys To”, 2015 年五月 1 日, http://theguardian.com/technology/2015/may/01/encryption-wont-work-if-it-has-a-back-door-only-the-good-guys-have-keys-to-。

问什么应该发生，只要学习并理解发生了什么。在生态系统中，一些机体吃掉其他一些机体。在生态学中，我们从不追问一只猫头鹰吃掉一只老鼠或是一只老鼠吃掉一颗种子是好是坏，我们只需观察到它们这样做。物种种群的增长或衰退基于它的生存条件；这无所谓对错，这只是一种生态现象，即使到了某一物种灭绝的程度。

与之相反，对其生存环境持有伦理立场的人们可以决定保护那些如果没有人为干预就会消亡的东西——诸如公民社会、民主、人权、和平、公共卫生、稳定的气候、洁净的空气和水、濒危物种、传统艺术……以及计算机用户的自由。

## 自由和开源软件（FLOSS）

FLOSS 意指“自由和开源软件”（Free/Libre and Open Source Software），这一概念试图在自由和开源之间寻求中立[1]。如果保持中立就是您的最终目标，使用 FLOSS 是保持中立的最佳方式。但如果您想要表达您对自由的支持，就不要使用这种中立性的短语。

## 免费（For Free）

如果您想要表述一个程序是自由软件，不要说它是可以“免费获得”的。这一短语明确表示“零价格”。自由软件关乎的是自由而非价格。

自由软件副本通常可以免费获得——例如，从文件传输协议（FTP）服务器下载。但自由软件副本也可以存储在只读光盘（CD-ROM）上贩卖；与此同时，私有软件副本有时也可以作为促销手段而免费获得，并且一些私有软件也通常免费提供给某些特定用户。

为了避免混淆，您可以称该程序可以“作为自由软件（而获得）”。

[1]参见 http://www.gnu.org/philosophy/floss-and-foss.html 以获得关于此问题的更多信息。

## 自由和开源软件（FOSS）

FOSS 意指“自由和开源软件”（Free and Open Source Software），这一概念试图在自由和开源之间寻求中立[1]。如果保持中立就是您的最终目标，使用 FLOSS 更好。但如果您想要表达您对自由的支持，就不要使用这种中立性的短语。

## 可自由获得（Freely Available）

不要将“可自由获得的软件”用作自由软件的同义词。这两个短语并不等同。如果任何人可以轻松获得一份副本，就可称其为“可自由获得”。而自由软件的定义是基于那些拥有其一份副本的用户所拥有的自由。这是对于不同问题的回答。

## 免费软件（Freeware）

不要将“免费软件”用作自由软件的同义词。“免费软件”这一短语在 20 世纪 80 年代常用于指代那些仅以可执行文件形式发布的程序，其源代码不可获得。现在这一短语没有任何特别的公认定义。

当使用英语以外的语言时，请避免直接借用英语短语诸如“free software”或“freeware”。最好能将 free software 翻译为您所说的语言。（参见本书附录 B (p. 341) 以获得将 free software 无歧义地翻译为多种语言的方式。）

通过使用您自己语言中的词汇，您表明了真正想要表达的自由之意，而非只是对某些神秘的外国市场概念鹦鹉学舌。首次引用自由的概念可能会使您的同胞感到奇怪和不安，但是一旦他们发现它所表达的正是它所宣称的涵义，他们将会真正地理解问题的实质。

[1]参见上一条脚注。

## 赠送软件（Give Away Software）

用 “赠送” 指代 “将程序作为自由软件发布” 是误导性的。这种表达方式与 “免费” 一词相似：它暗示问题的关键是价格而非自由。避免这种混淆的方式要说成 “作为自由软件发布”。

## 谷歌（Google）

请不要将 “谷歌” 用作动词以指代在互联网上进行搜索。它只是众多搜索引擎中的一种。我们建议使用 “网络搜索” 这一短语。尽可能使用一种尊重您隐私的搜索引擎；DuckDuckGo 声称它不会跟踪其用户[1]，尽管我们不能确认这一点。

## 黑客（Hacker）

黑客是指享受智慧乐趣的人们[2]——并不一定与计算机有关。在 20 世纪 60 至 70 年代，麻省理工学院（MIT）的自由软件社区的程序员称他们自己为黑客。大约在 1980 年，那些发现了黑客社区的记者们错误地将这一词语用于指代 “安全破坏者”。

请不要散播这种错误。那些破坏安全的人称为骇客（cracker）。

## 知识产权（Intellectual Property）

出版商和律师喜好将版权描述为 “知识产权”——这一短语也被应用于专利、商标以及其他更为晦涩的法律领域。这些法律之间的共同特性如此之少，而它们之间的区别如此之大，以至于对它们进行广义化是无谋的。最好还是具体地讨论版权，或是专利，或是商标。

“知识产权” 这一短语带有一种隐藏的假设——即对于那些不同类的问题的思考方式应当基于同实物对象的类比，以及我们关于它们的概念应当

[1] “DuckDuckGo Privacy Policy”，最后更新于 2012 年四月 12 日，https://duckduckgo.com/privacy。

[2] 参见我的文章 “On Hacking”，位于 http://stallman.org/articles/on-hacking.html。

基于同实物产权的类比。

当涉及复制的时候，这种类比抹杀了实物对象和信息之间的本质区别：信息可以被毫不费力地复制，而实物对象则不能。

为了避免不必要的偏见和混淆，最好能够采取这样一种严格的策略，不允许将 “知识产权” 作为讨论甚至是思考的基础。

虚伪地将这些权力称为 “权利” 已经开始使得世界 “知识产权” 组织（WIPO）感到尴尬[1]。

## LAMP 系统（LAMP System）

LAMP 是指 “Linux，Apache，MySQL 和 PHP”——一种常用于网络服务器的软件组合，除了此上下文环境中的 “Linux” 实际上是指 GNU/Linux 操作系统。因此，LAMP 应该改为 GLAMP，即 “GNU，Linux，Apache，MySQL 和 PHP”。

## Linux 系统（Linux System）

Linux 是 Linus Torvalds 于 1991 年开始开发的内核的名字，使用 Linux 内核的操作系统基本上是由 GNU 再加上 Linux 内核组成的。将整个操作系统称为 “Linux” 是不公平并且引起混淆的。请将此完整的操作系统称为 GNU/Linux，既尊重 GNU 计划，也将整个操作系统和它的内核区分开来[2]。

## 市场（Market）

将自由软件用户，或者更普遍地将软件用户称为 “市场” 是误导性的。

这并不意味着自由软件社区之中没有市场的立锥之地。如果您拥有一项自由软件的支持业务，您就可以拥有客户，并且您可以在市场中同他们

[1] Richard Stallman，“Public Awareness of Copyright, WIPO, June 2002”，最后更新于 2014 年，http://gnu.org/philosophy/wipo-PublicAwarenessOfCopyright-2002.html。

[2] 参见《Linux 和 GNU 操作系统》 (p. 75) 一文以获知 GNU/Linux 操作系统的更多历史背景，由于它与本文所提到的命名问题相关。

进行交易。只要您仍然尊重他们的自由，我们就会祝愿您在自己的市场中取得成功。

但是，自由软件运动是一场社会运动而非一项业务，其致力于的成功并非某种市场上的成功。我们试图通过赋予公众自由来服务于公众——而非通过竞争从对手手中夺取市场份额。将这场为自由而战的运动同某种仅仅是为了商业上的成功而进行的行为等量齐观，就是否认了自由的重要性，并且使得私有软件合法化了。

## 货币化（Monetize）

“monetize” 一词的恰当定义是 “将某物用作货币”。例如，人类社会曾经将金、银、铜、印刷的纸、某些种类的贝壳、大块的岩石等货币化。然而，我们现在见到一种趋势正在将这个词语用作另一种用途，即 “将某物用作盈利的基础”。

这种用法将利润置于首要位置，而将用于盈利的东西置于次要位置。将这种态度用于软件工程是令人反感的，因为这将引导开发者将软件变为私有的，如果他们得出的结论是使其成为自由软件不能为其带来足够的利润。

一家富有生产力并且合乎伦理的企业也可以盈利，但只要它将所有其他东西置于利润之下，它就不再是合乎伦理的。

## MP3 播放器（MP3 Player）

在 20 世纪 90 年代后期，制造便携式、固态存储的数字音频播放器成为可能。其中的大部分支持受专利限制的 MP3 编码解码器（codec），但这并不是全部。某些播放器支持不受专利限制的音频编码解码器诸如 Ogg Vorbis 或者自由无损音频编码解码器（FLAC），还有些播放器甚至完全不支持 MP3 编码的文件，这显然是为了回避这些专利。将这些设备称为 “MP3 播放器” 不仅仅是引起混淆的，它也赋予了 MP3 额外的好处，而这正是我们应当拒绝的。我们建议使用 “数字音频播放器” 或者更简单的 “音

频播放器”，如果上下文允许如此简略。

## 开放（Open）

请不要将 “开放” 或者 “开源” 用作自由软件的替代词语。这些短语基于不同的价值观代表不同的立场[1]。自由软件是一场政治运动；开源只是一种开发模式。

当指代开源的立场时，使用它的名字是恰当的；但不要为我们或我们的作品贴上 “开源” 的标签——那将会使人们误认为我们也持有那样的立场。

## 个人计算机（PC）

使用 PC 这一缩略词指代某一特定类型的计算机硬件是可以的，但是请您不要用它来暗示该计算机必须运行微软 Windows 操作系统。如果您在同一台计算机上安装 GNU/Linux，它仍然是一台 PC。

建议使用 WC 来称呼一台运行 Windows 的计算机。

## Photoshop（PS）

请不要将 photoshop（PS）用于动词以在普遍意义上指代任何照片处理或图像编辑。它只是一款特定的图像编辑软件的名字，我们应当尽量避免使用它，由于它是私有软件。有很多自由软件可用于图像编辑，例如 GIMP[2]。

## 盗版（Piracy）

出版商通常将不被它们批准的复制行为称为 “盗版”。通过这种形式，它们暗示这在伦理上与在外海攻击船只以及绑架谋杀船上的人一样坏。基

[1] 参见《为什么说开源漏掉了自由软件的要点》 (p. 90) 以获得完整解释。

[2] 参见 http://directory.fsf.org/wiki/GIMP。

于这样一种鼓吹，它们在世界上的大部分地区骗取到了法律支持以便在绝大多数（有时是全部）环境下禁止复制（它们仍在继续施压以使得这些禁令更加完整）。

如果您不认可未经批准的复制行为就像绑架谋杀的理念，您可能也不愿意使用“盗版”一词来描述此行为。中立性的短语诸如“非授权复制”（或者“被禁止的复制”，用于此种行为不合法的情况）可作为替代。我们中的一些人甚至倾向于使用某种褒义词，诸如“与他人分享信息”。

一位主持了版权侵犯审判的美国法官认可“盗版”和“盗窃”属于诽谤性的词语[1]。

## PowerPoint（PPT）

请不要使用 PowerPoint 指代任何类型的幻灯片演示文稿。它只是一款特定的用于制作演示文稿的私有软件的名字。为了您的自由，您应该只用自由软件来制作演示文稿。推荐的可选工具包括 TEX 的 beamer 类以及 OpenOffice.org（或 LibreOffice——译者注）的 Impress。

## 保护（Protection）

出版商的律师喜爱使用“保护”一词来描述版权。这个单词暗示防止破坏或者避免痛苦；因此，它鼓励人们去同情那些通过版权得利的版权持有人和出版商，而非那些受到版权限制的用户。

避免说“保护”而改用中立性的词语是容易做到的。例如，与其说“版权保护持续很长时间”，不如说“版权持续很长时间”。

类似地，与其说“受版权保护”，不如说“被版权覆盖”或者更简单的“具有版权”。

如果您想要批判版权制度而非只是想保持中立，您还可以使用“版权

[1] Ernesto Van der Sar，“MPAA Banned from Using Piracy and Theft Terms in Hotfile Trial”，2013 年十一月 29 日，http://torrentfreak.com/mpaa-banned-from-using-piracy-and-theft-terms-in-hotfile-trial-131129。

限制”这一短语。因此，您可以说“版权限制持续很长时间”。

“保护”一词还被用于描述恶意功能。例如，“复制保护”是一种干扰复制行为的功能。从用户的观点来看，这是一种阻碍。因此我们将此恶意功能称为“复制阻碍”。它更多地被称为数字限制管理（DRM）——参见 Defective by Design 运动，它位于 DefectiveByDesign.org。

## 合理与无差别（RAND，Reasonable and Non-Discriminatory）

一些标准化组织推广那些禁止自由软件实现的，受专利限制的标准，这些组织通常拥有这样的专利许可获得政策，它要求符合此标准的软件的每份副本支付一笔固定的专利费。它们通常将这样的专利许可条款称为 RAND，即“合理与无差别”。

这一短语洗白了一系列通常既不合理也并非无歧视的专利许可条款。确实，这些许可条款并不歧视任何特定的个人，但它们确实是在歧视自由软件社区，这使得它们成为不合理的。因此，RAND 的一半是欺骗性的，而另一半是带有偏见的。

标准化组织应该认识到这些许可条款是具有歧视性的，并且放弃使用“合理与无差别”或者 RAND 来描述它们。在它们这么做之后，那些不希望加入这种“洗白”说法的政策制定者才能拒绝这一短语。仅仅因为那些持有大量专利的商业公司使其变得普遍流行就接受并使用这一短语，将会允许那些商业公司支配您所表达的观点。

我们建议使用“仅限统一付费”（uniform fee only）或者简写为 UFO 来作为替代，这种描述是精准的，由于这些许可条款的唯一条件就是统一的权利金。

## 软件即服务（SaaS，Software as a Service）

我们曾经说“软件即服务”是一种不公，但我们随后发现人们对于哪些活动属于 SaaS 的理解还存在诸多分歧。于是我们转为使用一个新的短语，“作为软件替代品的服务”，即 SaaSS。这个短语有两个优点：由于它

从未被使用过，因此我们的定义是其唯一定义；另外它解释了这种不公包括哪些方面。

参见《服务器真正是在为谁服务?》 (p. 302) 一文以获得关于此问题的更多讨论。

在西班牙语中，我们继续使用 “software como servicio” 这一短语，这是由于 “software como ser vicio”[1]这个笑话太精妙绝伦了，以至于我们不舍得放弃它。

## 贩卖软件（Sell Software）

“贩卖软件” 这一短语是含混不清的。严格地说，用一份自由软件副本换取一笔资金是在贩卖该软件，这样做也无可厚非。然而，人们通常将 “贩卖软件” 与对该软件的后续使用的私有限制相关联。您可以保持清醒并且拒绝混淆，通过说 “有偿发布软件副本” 或者 “为软件的使用施加私有限制”。

参见《售卖自由软件》 (p. 49) 一文以获得关于此问题的更多讨论。

## 分享经济（Sharing Economy）

“分享经济” 这一短语不是一种用于指代诸如 Uber 或者 Airbnb 这样的在人与人之间安排商业交易服务的好方式。我们使用 “分享” 这一词语来指代非商业合作，包括对已发布的作品的原始版本进行非商业性的再分发。将 “分享” 一词的涵义延伸至包含这些商业交易的程度破坏了它的本意，因此我们不会在这种上下文中使用它。

用于描述诸如 Uber 的业务的更好的短语是 “计件工作服务经济”（piecework service economy）。

[1] “软件，作为恶意的” 之意。

## Skype

请不要将 Skype 作为动词使用，用于从一般意义上指代任何类型的视频通讯或者网络电话。Skype 只是一种特定的私有软件的名字，它会窥探用户。[1]如果您想要以一种同时尊重您的自由和隐私的方式进行网络音视频通话，您可以试用众多自由的 Skype 替代品，详情 https://libreplanet.org/wiki/Group:Skype_Replacement。

## 软件工业（Software Industry）

“软件工业”一词鼓励人们想象软件总是由某种工厂开发并且送达“消费者”手中的。自由软件社区已经证实了事情并非如此。软件企业确实存在，并且不同的企业开发自由和/或私有软件，但那些开发自由软件的企业并不是像工厂那样运营的。

“工业”一词正在被软件专利的倡导者用作一种鼓吹。他们将软件开发称为“工业”，然后试图争论这意味着它也应该受到专利垄断权的支配。欧洲议会于 2003 年拒绝软件专利，并且通过表决，将“工业”一词定义为“实物商品的自动化生产”[2]。

## 源代码模型（Source Model）

维基百科（Wikipedia）以一种使人混淆的方式使用“源代码模型”这一概念。从表面看，它指的是程序的源代码如何分发，但其文本将这一概念与开发方法论混淆起来。它将区分“开放源代码”和“共享源代码”作为答案，但它们是重叠的——微软将后者作为一种市场概念以覆盖一系列实践，其中的一些也是“开放源代码”的。因此，这一概念实际上未能传达任

[1]参见这份说明以获得更多信息 http://gnu.org/proprietary/proprietary-surveillance.html/#SpywareInSkype。

[2]欧洲议会，“Directive on the Patentability of Computer-Implemented Inventions”，2003 年九月 24 日，http://web.archive.org/web/20071222001014/http://www.swpat.ffii.org/papers/europarl0309。

何合乎逻辑的信息，但它却为连篇累牍地把自由软件描述为 “开源” 提供了机会。

## 终端（Terminal）

移动电话和平板设备是计算机，并且人们应当被允许在它们上面使用自由软件以进行他们的计算。将它们称为 “终端” 暗示它们仅仅适合于连接到服务器，这是您操控自己计算的一种坏方式。

## 盗窃（Theft）

那些支持过于严格和专制版权制度的支持者，通常使用诸如 “盗版” 或者 “盗窃” 这样的词语来指代侵犯版权的行为。这是一种偏见，但是它们希望您将其作为客观事实而接受。

在美国法律体系中，版权侵犯并不属于盗窃。适用于盗窃的法律并不适用于版权侵犯。专制性的版权制度的支持者正在请求权力机构——并且错误地解读权力机构所说的话[1]，阐明了什么才能被恰当地描述为 “版权盗窃”。

非授权复制在很多情况下（不是全部!）被版权法所禁止，但是被禁止并不意味着它是错误的。从普遍意义上说，法律并不定义正确与错误。法律充其量只能试图实施公正。如果法律（实现）不能适应我们的是非观（具体），那么这样的法律就应该被改变。

一位主持了一场版权侵犯审判的美国法官认可 “盗版” 和 “盗窃” 属于诽谤性的词语[2]。

[1]为了驳斥他们，您可以援引这一真实案例：Harper Lee 起诉她的代理人涉嫌诱骗她将 To Kill a Mockingbird 一书的版权指认给他。

[2]Ernesto Van der Sar，“MPAA Banned from Using Piracy and Theft Terms in Hotfile Trial”，2013 年十一月 29 日，http://torrentfreak.com/mpaa-banned-from-using-piracy-and-theft-terms-in-hotfile-trial-131129。

## 信任计算（Trusted Computing）

“信任计算” 是其倡导者所起的名字，用于指代这样一种重新设计计算机的阴谋，这使得应用程序开发者可以信任您的计算机将会服从他们而非您[1]。从他们的角度看，这是 “信任”；从您的角度看，这是 “背叛”。

## 供应商（Vendor）

请不要使用 “供应商” 一词一般性地指代任何开发或者打包软件的人。很多程序被开发的目的是贩卖其副本并且它们的开发者自然也是其供应商；这甚至也包括部分自由软件包。然而，很多程序是由志愿者或者并非致力于贩卖副本的组织开发的。这些开发者不是供应商。类似地，只有部分 GNU/Linux 发行版的打包者是供应商，我们建议使用 “提供者”（supplier）这一通用性的概念。

[1] 参见《您能够信任您的计算机吗?》 (p. 279) 以获得关于此问题的更多信息。

# 第 3 部分

# 版权和不公

## 阅读的权利


本文撰写于 1996 年并以标题 “The Right to Read: A Dystopian Short Story”（阅读的权利：一段反乌托邦的短故事）于 1997 年二月发表于 Communications of the ACM, vol. 40, n. 2。

本文摘自《第谷之路》（*The Road to Tycho*），这是一系列关于“月球人革命”的前奏的文章，于 *2096* 年发表于月球城。

对于 Dan Halbert 而言，他的“第谷之路”始于大学时代——当 Lissa Lenz 向他借用计算机的时候。她的计算机发生了故障，并且除非她能够借到另一台计算机，她将肯定不能完成中期项目。她不敢向其他任何人求助，除了 Dan。

这将 Dan 置于两难境地，他有责任帮助她——但如果将自己的计算机借给她，她将有机会阅读他的书籍。抛开如果您允许其他人阅读您的书籍，您将被处以很多年监禁这一事实，这种想法首先使他感到震惊。与其他任何人一样，他从小学开始就被这样教育：分享书本是龌龊并且非法

的——只有盗版者才会去做这些坏事。

并且，他不被 SPA——软件保护权力机构——抓住的机会并不大，在他的计算机软件课上，Dan 知道了每本书都带有一个版权监视器，它将会将此书于何时何地被何人阅读的信息上报给中央授权许可机构（该机构对这些信息的利用不仅仅是为了抓住阅读盗版者，也会将个人阅读兴趣偏好出卖给图书零售商）。当他的计算机下一次联网时，中央授权许可机构将会发现这一切。他作为计算机拥有者将会受到最严厉的处罚——由于他未能尽力阻止这种犯罪。

当然，也许 Lissa 本意上并不一定是要阅读他的书籍，她有可能只是想借用计算机来编写她的中期汇报。但是，Dan 知道她来自于中产阶级家庭，并且很难负担得起学费，遑论她的各种阅读费用。阅读他的书籍可能是使她能够顺利毕业的唯一出路。他理解这种情形；他自己也必须借钱以支持其所需的科研论文阅读费用（这些费用的 10% 将会被付给撰写了这些论文的科研人员；由于 Dan 致力于他的学术事业，他可以指望自己的科研论文被频繁引用，这样他就可以赚到足够的钱来支付这笔阅读费用）。

其后，Dan 将会了解到，以前曾经有一段时间，任何人都可以去图书馆阅读期刊文章甚至是图书而无需支付任何费用。那时也有独立的学者，他们可以阅读数千页的文献而无需得到政府图书馆的批准。但是，在 20 世纪 90 年代，商业性和非盈利性的期刊出版者都开始征收访问费用。直到 2047 年，提供对学术文献的免费公开访问的图书馆已经成为一种灰暗的记忆。

当然，也有一些方式可用于绕过软件保护权力机构（SPA）和中央授权许可机构的审查。它们本身是非法的。Dan 曾经有一位计算机软件专业的同学，Frank Martucci，此人得到了一种非法的调试工具，并且将其应用于在阅读时绕过版权监视器的代码。但是，他将此事告知了太多的朋友，其中之一将他告发至 SPA 以获取一份奖赏（债台高筑的学生很容易被利诱叛变）。2047 年，Frank 被捕入狱，并非由于盗版阅读，而是由于非法拥有一份调试工具。

其后，Dan 将会知道曾经有一段时间，任何人都被允许拥有调试工具，

那时甚至有可以从光盘（CD）获得或者从网络上下载得到的自由调试工具。但是，随着普通用户将其用于绕过版权监视器，最终一位法官裁决非法目的成为了调试工具在实际中的主要用途。这意味着它们是非法的，调试工具的开发者都被捕入狱。

当然，程序员们仍然需要调试工具，但在 2047 年，调试工具厂商限量发布它们，并且仅对得到政府官方授权许可与担保的程序员分发。Dan 用于计算机软件课上的调试工具位于一道特殊的防火墙后面，使得它只能被用于课堂练习。

从前，通过安装某种修改过的操作系统内核来绕过版权监视器也是可能的。Dan 最终将会了解到这些自由内核，甚至整个自由操作系统。这些东西直到世纪之交前后都还存在。但是，它们不仅像调试工具那样是非法的——甚至您即使拥有它们也不能安装，因为您不知道您计算机的 root 口令。而美国联邦调查局（FBI）或者微软客服都不会告知您那个口令。Dan 得出结论，他不能简单地将计算机借给 Lissa，但他又不能拒绝帮助她，因为他爱着她。每次和她说话的机会都会使他充满愉悦。而她选择向他求助，这意味着她也爱着他。

Dan 做出了一些更加不可想象的举动来解决这种两难境地——他将计算机借给了她，并且将他的口令告知了她。通过这种方式，如果 Lissa 阅读了他的书籍，中央授权许可机构将会认为实际上是他在阅读这些书。这仍然是一种犯罪，但是 SPA 不会自动发现这一点，除非 Lissa 将他举报。

当然，如果学校最终发现他将自己的口令给了 Lissa，这将使得他们二人的学生身份终结，不论她使用这个口令做了什么。学校的政策是这样的，任何对其监控学生如何使用计算机的行为的干涉是惩戒行为的底线，不论您是否做了任何有害的事——这种所谓的攻击行为使得学校管理人员难于对您进行监控。他们认为这意味着您正在做一些其他的被禁止的事情，并且他们不需要知道这些事情到底是什么。

学生们通常不会因此被开除——不是直接被开除。而是他们将会被禁止使用学校的计算机系统，这将使他们不可避免地不能通过他们的任何课程。

其后，Dan 将会知道，这种大学政策只是始于 20 世纪 80 年代，那时大学生们开始大规模使用计算机。在此之前，大学一直保持着另一种学生惩戒制度；它们将会惩罚那些有害的行为，而非那些仅仅是引起怀疑的行为。

Lissa 没有向 SPA 举报 Dan，而他关于帮助她的决定使得他们终成眷属。这也促使他们质问他们在儿时曾被教给的关于盗版的东西。这对夫妇开始阅读关于版权的历史的东西，关于苏联以及它对复制分享的限制，甚至是关于最初的美国宪法。他们移居月球，在那里他们遇到了其他同样选择逃离 SPA 魔掌的人们。当第谷于 2062 年爆发起义的时候，普遍的阅读权利立即成为这场起义的核心诉求。

## 作者注记

- 本文据说应该是由其他人在未来撰写的一篇历史文章，描述了 Dan Halbert 在一种专制社会下的青年生活，这种专制社会是由那些将“盗版”作为宣传口号的敌人创立的。因此本文使用那种社会下的用语。我尝试从今天的视角呈现它，使其看起来更具压迫性。有关“盗版”可参考《应避免使用（或慎用）的词语》一文的“盗版（Piracy）”一节 (p. 121) 。

- 以下注记自本文初次发表之后经过数次更新。

  阅读的权利是一场于今天打响的斗争，尽管可能需要经过 50 年的时间才能使我们现在的生活方式成为灰暗的记忆，上文所述的大部分特定法律和实践已经被提议，它们中的许多已经在美国和其他地区被通过并成为正式的法律。在美国，1998 年的数字千年版权法案（DMCA）为限制阅读与出借计算机化的图书（及其他形式的作品）建立了法律基础。欧盟在 2001 年通过的一项版权指导意见中实施了类似的限制。在法国，随着“信息社会的作者权利与相关权利法案”（DADVSI）于 2006 年通过，仅仅是拥有 DeCSS，一款用于解密数码多功能影音光盘（DVD）的自由软件，的一份副本即被视为犯罪。

在 2001 年，由迪士尼赞助的参议员 Hollings 提出了一项称为 “安全系统与标准认证法案”（SSSCA）的法律，其要求每台新计算机带有强制性的限制复制设备并且禁止用户绕过它。继 Clipper 芯片及类似的美国政府关于密钥保管的提案之后，这显示了一种长期趋势：计算机系统越来越倾向于赋予不在场者对该计算机系统的实际使用者进行有效控制的权力。SSSCA 随后更名为 “消费者宽带和数字电视促进法案”（CBDTPA）这一不可拼读的名称，它被评注为 “消费但不要尝试编程法案”（Consume But Don’t Try Programming Act）。

随后不久，共和党人控制了美国参议院。他们与好莱坞的联系不如民主党人那样密切，因此他们没有强行推动这些提案。现在民主党人重新获得控制权，我们所面临的威胁进一步升高。

在 2001 年，美国开始试图利用它所倡议的美洲 “自由贸易” 区（FTAA）条约来向整个西半球强制施加相同的规则。FTAA 是一种所谓的 “自由贸易” 条约，它们实际上是被设计用于赋予商业企业更多凌驾于民主政府之上的权力；强制推行诸如 DMCA 的法律正是这种精神的典型体现。而 FTAA 事实上由时任巴西总统卢拉否决，此人拒绝接受 DMCA 及其他类似法规的条件。

此后，美国通过一系列 “自由贸易” 协定将类似的条件强行施加给诸如澳大利亚和墨西哥等国，并且通过另一纸条约，中美洲自由贸易协定（CAFTA），将其强行施加给哥斯达黎加等国。时任厄瓜多尔总统科雷亚拒绝与美国签署一项 “自由贸易” 协定，但我听说厄瓜多尔已经于 2003 年采纳了类似于 DMCA 的法案。

这个故事中的某一种理念直到 2002 年还未在现实中被提议，这种理念是由 FBI 和微软掌控您的个人计算机的 root 口令并且不会让您得到它们。

这一阴谋的倡导者将其命名为诸如 “信任计算” 或者 “Palladium”。我们称之为 “背叛计算”[1]，由于其效果是使您的计算机听命于商业公

[1]参见《您能够信任您的计算机吗?》 (p. 279) 一文以获得更多关于 “信任计算” 的信息。

司，并且到了不再听命于您甚至是违抗您的地步。这在 2007 年作为 Windows Vista 的一部分被实施[1]；我们可以预见苹果也会做出类似的事情。在这种阴谋中，由计算机制造商掌控其秘密代码，但 FBI 对它同样是唾手可得。

确切地说，微软所掌控的并不是传统意义上的口令；由于无人能够在终端上输入它。与之相反，它是一种数字签名和加密密钥，它对应着存储在您的计算机上的另一个密钥。这将允许微软以及任何潜在与微软合作的网站对于用户能够在其计算机上做的事情进行完全控制。

Vista 同样赋予了微软额外的权力；例如，微软可以强制安装更新，它可以命令所有运行 Vista 的计算机拒绝运行某一特定的设备驱动程序。Vista 所施加的诸多限制的主要目的是强制实施用户所不能克服的数字限制管理（DRM）。DRM 的威胁正是我们发起 Defective by Design 运动的原因 DefectiveByDesign.org。

当这段故事初次编写之时，SPA 威胁小型互联网服务供应商（ISP），要求它们允许 SPA 监控所有用户。大多数 ISP 在受到威胁的时候都妥协了，由于它们打不起官司。一家位于加利福尼亚州奥克兰的 ISP，Community ConneXion 拒绝了这一要求，并且确实被起诉了。SPA 随后撤诉，但它得到了 DMCA 的支持，这赋予了它们一直渴求的权力。SPA，实际上是软件发行商协会之意，由商业软件联盟（BSA）取代了其类似于警察的职责。在今天，BSA 不是一支官方警察力量，但它却在以非官方的方式行使这种权力。通过使用让我们想起前苏联的手段，它邀请人们举报其同事和朋友。于 2001 年在阿根廷发生的一起 BSA 恐怖行动几乎毫无遮掩地威胁人们：分享软件将被处以酷刑。

上文描述的大学安全政策并非毫无实据。例如，芝加哥地区的某大学的一台计算机在登录时显示了以下信息：

> 本系统仅供授权用户使用。对于任何未经授权或者超出其

[1]参见 http://badvista.fsf.org/ 以了解我们反对 Windows Vista 的运动。

> 授权范围使用本系统的个人，其在本系统上进行的所有活动将被监控并且记录。在对任何人不当使用本系统进行监控或者系统维护期间，授权用户的行为也可能被监控。任何使用本系统的个人必须明确同意接受这种监控，并且此监控一旦发现可能指证非法行为或者侵犯大学规定的行为的证据，本系统可能将通过此监控发现的证据移交至大学权力机构和/或官方执法机构。

这是一种类似于美国宪法第四修正案的有趣方式：首先施压以迫使几乎所有人放弃他们的权利。

## 坏消息

关于阅读的权利的斗争已经在进行中了，敌人已经有备而来严阵以待，而我们还没有。因此这场斗争正在向着对我们不利的方向进行。自本文初次发表之日起发生的一些代表性的坏消息包括：

- 今天的商业电子书剥夺了读者的传统自由。参见《电子书的威胁》(p. 297) 一文以获得更多信息；
- 某一“生物课本”网站[1]允许您访问其出版物，仅当您签订合同以保证不会将其借给任何人阅读[2]。而出版商可以随意废除此合同；
- 电子出版物限制其用户的自由[3]；

[1] Nature America Inc., “Announcing Principles of Biology, an Interactive Textbook by Nature Education,” http://nature.com/nature_education/biology.html.

[2] Nature America Inc., “Principles of Science Privacy Notice,” accessed August 2015, http://nature.com/principles/viewTermsOfUse.

[3] 参见 Don Clark 的文章 “Seybold Opens Chapter on Digital Books”（31 August 1999, http://www.zdnet.com/article/seybold-opens-chapter-on-digital-books/）以获知有关以电子形式发布图书以及影响到阅读其副本的权利的版权问题的信息。

- 计算机中的图书[1]：用于控制谁可以在一台计算机上阅读图书和文档的软件。

如果我们想要阻止这样的坏消息并且带来一些好消息，我们需要组织起来并且进行斗争。自由软件基金会（FSF）的 Defective by Design 运动已经为这场斗争拉开序幕；订阅该运动的邮件列表以提供支持。并且加入 FSF 以资助我们的工作。

## 参考文献


- 美国专利及商标局（USPTO），Intellectual Property and the National Information Infrastructure: The Report of the Working Group on Intellectual Property Rights, Washington, DC: GPO, 1995.（参见《您说过 “知识产权” 吗？这是一种迷惑的幻景》 (p. 99) 一文以获知为何 “知识产权” 这一短语是含混不清的并且永远不应被使用）。

- Samuelson, Pamela, “The Copyright Grab,” *Wired*, January 1996, 4.01, http://wired.com/wired/archive/4.01/white.paper_pr.html.

- Boyle, James, “Sold Out,” *New York Times*, 31 March 1996, sec. 4, p. 15; 也可以在这里找到：https://law.duke.edu/boylesite/sold_out.htm.

- 社论，*Washington Post*, “Public Data or Private Data,” 3 November 1996, sec. C, p. 6, http://web.archive.org/web/20130508120533/http://www.interesting-people.org/archives/interesting-people/199611/msg00012.html.

- 公有领域联盟（Union for the Public Domain）——一个致力于抵抗并且逆转版权和专利的权力过度扩张的组织。


[1] “Microsoft Announces New Software for Reading on Screen,” 30 August 1999, http://microsoft.com/en-us/news/press/1999/Aug99/SeyboldPR.aspx.

# 对版权的错误解读——一系列错误



版权法中发生了一些奇怪而又危险的事情。在美国宪法环境下，版权存在的意义是使用户获益——用户是指那些读书、听音乐、看电影或者运行软件的人们——而非为了出版商或作者。然而，即使人们越来越倾向于拒绝并且反抗那些“为了他们自己的好处”而强行施加给他们的各种版权限制，美国政府仍然正在施加更多的限制，并且试图以新的严厉惩罚措施来恐吓公众以迫使其服从。

那么，版权政策是如何一步一步地走到与其宣称的初衷背道而驰的地步的呢？而我们又如何才能使其重新回到与其初衷相符的正轨呢？为了理解这些，我们应当首先审视美国版权法的根基：美国宪法。

## 美国宪法中的版权

当美国宪法被起草之时，作者们应当被赋予某种版权垄断权这一想法被提出——然后又被拒绝了。我们国家的创始人采纳了另一种假设，即版权并非作者们的一种自然的权利，而是以进步之名，人为地为他们做出的一种认可。宪法通过这一条款（第 I 章，第 8 条，第 8 款）许可了这样一种版权体系：

> （国会应当拥有权力）通过在一段限定的时间内保证作者和发明者对其作品或发明拥有专属权利，以促进自然科学和实用技艺的进步。

美国最高法院已经一再强调，促进进步意味着保护那些版权作品的使用者的利益。例如，在 *Fox Film* 起诉 *Doyal* 的案例[1]中，法庭说道：

[1] Fox Film Corp. v. Doyal, 286 US 123, 1932.

> 美国的根本利益以及赋予（版权）垄断权的主要目的是基于那些由公众得来、通过作者的劳动获得的普遍利益。

这一根本性的决定解释了为何版权不是被宪法强制要求的，而只是作为一种选项而被许可——以及为何只应该在一段“限定的时间”内有效。如果版权是一种自然的权利，即某种作者们由于应当拥有它而拥有的东西，那么没有任何理由可以解释为何可以在一定时期之后终止这种权利，就如同不能解释为何每个人的房子在建成一定时期之后应当成为公共财产。

## 版权交易

版权体系通过为出版商和作者提供特权以保证其利益进而维持其自身运转；然而，这么做并非为了他们的利益。与之相反，它如此运作是为了改变他们的行为：即刺激作者们创作并且发表更多作品。事实上，政府是在以公众的名义消费公众的自然权利，作为交易的一部分为公众带来更多的发表作品。法学家们将这一概念称为“版权交易”。就如同政府花费纳税人的钱财以购买一条高速公路或者一架飞机，区别只是在于此处政府花费的是我们的自由而非我们的钱财。

但是，这种版权交易对于公众而言真的是一种公平的交易吗？众多其他交易方式都是可能的；哪一种方式最好呢？版权政策的每一个问题都是这个问题的一部分。如果我们对这个问题的本质产生误解，我们将会倾向于对这些问题做出不恰当的抉择。

宪法批准了将版权权力赋予作者。而实际上，作者们通常将其转让给出版商；因此实际上通常是出版商而非作者在行使这些权力从而得到大部分利益，尽管作者有时也能得到一小部分利益。因此通常只是那些出版商在游说政府以进一步增加版权权力。为了更好地反映关于版权的事实而非那些鬼话，本文将会把出版商而非作者称为版权权力的持有者，并且将有版权作品的使用者称为“读者”，即使对这些作品的使用并非只能是阅读，这是由于“用户”一词过于遥远和抽象。

## 第一个错误：寻求平衡

版权交易将公众置于首要地位：公众读者的利益是其一端；出版商的利益（如果有）只是达成这一目的的一种方式。因此，读者的利益和出版商的利益首先在质上是不对等的。对版权的目的的第一步错误解读是将出版商的利益的重要性提升到与读者的利益相同的级别上来。

通常有这样的说法，美国版权法旨在在出版商和读者的利益之间“寻求平衡”。那些援引这种解读方式的人们将其呈现为一种对宪法中所陈述的基本立场的复述，换言之，它应当与所谓的版权交易等价。

但是，这两种解读其实有着天壤之别；它们不仅在概念上截然不同，在涵义上也是完全不同的。这种“平衡”的概念假设读者和出版商的利益在重要性上的差别只是量上的，即我们应当分别给予它们多少权重，以及它们适用于哪些场合。“利益相关方”的概念通常被用于以这种方式描述这一问题。这种观点抹杀了读者和出版商的利益在质上的区别，而这种区别正是政府参与版权交易的基本点。

这种偷换概念的影响是深远的，由于在版权交易中对公众利益的强有力的保护——版权特权只能以读者之名被合理化，而绝非以出版商之名被合理化这一理念——被这种“平衡”的解读方式所抛弃了。由于出版商的利益也被视为其中一端，它也可以使版权特权合理化；换言之，“平衡”概念宣称版权特权可以以公众以外的某人之名被合理化。

从实践角度讲，这种“平衡”概念的结果是逆转了在版权法中做出更改所需正当理由的负担。版权交易将这种负担置于出版商一方，他们需要说服读者转让部分自由。而平衡概念逆转了这种负担，实事求是地说，由于对于出版商将会通过额外的特权得到好处这一点没有异议。除非这对读者造成的伤害能够被证实，并且这种伤害大到了已经超出读者所能得到的好处的程度，我们将会得出结论：出版商被赋予了它们要求的几乎所有特权。

由于在出版商和读者之间“寻求平衡”的理念否定了读者本应拥有的首要位置，我们必须坚决反对。

## 针对什么的平衡?

当政府为公众购买任何东西之时，它是在以公众的名义进行交易；它的责任是试图得到最佳的可能交易方式——对公众而言的最佳，而非对于交易中的另一方。

例如，当政府与建设公司签署合同以修建高速公路时，政府应当致力于花费尽可能少的公众资金。相关政府机构将会通过竞争性投标的方式来降低总价。

从实践角度讲，这一价格不可能为零，因为承包商不可能叫出那么低的价格。尽管没有获得特别报酬的资格，它们在自由社会中仍然享有通常的公民权利，包括拒绝对其不利的合同的权利；即使最低竞价也足以使某些承包商有利可图。因此，这里确实存在着某种平衡。但这并非由于利益双方各自要求得到特殊报酬而达成的一种蓄意的平衡。这是一种在公众目标和市场力量之间达成的平衡。政府试图为驾车的纳税人获得他们在自由社会和自由市场的大背景下可能得到的最佳交易。

在版权交易中，政府花费的是我们的自由而非我们的钱财。自由远比金钱更加珍贵，因此政府对于理性而节约地花费我们的自由的责任更重于花钱的责任。政府绝不应该将出版商的利益与公众的自由相提并论。

## 不是平衡，而是折中

将读者利益与出版商利益进行平衡是对版权政策的错误判断，但是，此处确实有两种利益需要被权衡：关于读者的两种利益。读者自身有其关于使用出版作品的自由的利益；取决于所处环境，他们还可能拥有通过某种激励系统鼓励发表作品的权利。

由于在讨论版权问题时，“平衡”一词被用于指代在读者和出版商之间“寻求平衡”这一理念。因此，对于读者自身的两种利益使用“平衡”一词

将会产生歧义[1]。我们需要另一个术语。

通常意义上，当某一团体拥有两种部分冲突的目标而不能两全之时，我们称之为“折中”。因此，与其说在两个团体之间“寻求正确的平衡”，不如说“在花费与保留我们的自由之间寻求最佳的折中”。

## 第二个错误：最大化一项输出

版权政策中的第二个错误包括将最大化——而非仅仅是增加——出版作品数量作为最终目标。第一个错误概念“寻求平衡”将出版商的地位提升到与读者对等的位置，而这第二个错误则将它们的地位进一步提升到远在读者之上的位置。

当我们购买物品时，我们通常不会买空所有库存或者只买最贵的型号。与之相反，我们将会为其他购买需求预留资金，对于任意特定的物品，只买我们所需要的，并且选择一种能够满足需求的型号而非最高端的型号。根据报酬递减原理，将我们的所有可用资金花在某一特定物品上很可能是一种低效的资源分配方式；我们一般会选择将部分资金留作他用。

报酬递减原理也适用于版权交易，如同它适用于任何其他购买行为。我们应当最先卖出的自由是那些使我们失去得最少的那部分自由，并且这部分自由的牺牲能够换取对作品出版的最大激励。随着我们继续卖出的自由越来越关乎自身核心利益，我们将会发现每一笔新的交易相对于上一笔都是更大的牺牲，同时它们所能带来的出版活力增量越来越小。而当这种增量远未减到零之前，我们就会说不值得为之付出不断增加的代价；而后我们将会达成某种交易，其最终结果是增加了出版量，但并未达到可能的极限程度。

将最大化出版量作为最终目标将会预先拒绝所有这些更加明智、更加有利的交易方式——它强制规定公众必须出卖几乎全部自由来使用出版作

[1]参见 Julian Sanchez 的文章“The Trouble with ‘Balance’ Metaphors”（4 February 2011, http://juliansanchez.com/2011/02/04/the-trouble-with-balance-metaphors/）以获得关于“合理判断与平衡权重之间的类比能够怎样以有害的方式限制我们的思考”这一问题的深入调查。

品，而仅仅是为了换取一点点出版量的增加。

## 具有欺骗性的最大化

在实践中，将最大化出版量作为最终目标而全然不顾自由的代价这一理念是由一系列具有欺骗性的说法所支撑的。这种说法宣称公众对出版物进行复制的行为是非法、不公平、并且本质上错误的。例如，出版商将复制出版物的人们称为“盗版者”，这种诽谤性的称谓将与他人分享信息的行为与攻击船只的行为等同起来（这一诽谤性的称谓最初被作者用于描述那些找到了以某种合法方式出版未经授权许可的版本的出版商；而它被出版商所使用的现代用法几乎与其原意完全相反）。这种欺骗性的说法直接抹杀了版权的宪法基础，而是如同表现美国法律体系中那些公认的传统那样呈现其自身。

“盗版”这一具有欺骗性的说法之所以被普遍接受，是由于它充斥于各种媒体，以至于极少有人意识到它到底有多么重要。它是如此地强有力，因为如果由公众进行复制是根本非法的，我们将完全不能对抗出版商强制要求我们牺牲自己的自由的行为。换言之，当公众被威胁给出理由以证明为何出版商不应获得某些额外权力之时，所有理由当中最重要的一条——“我们想要复制”——已经被预先否决了。

这封死了通过争论反对与日俱增的版权权力的所有出路，除了一些细枝末节的问题。因此，今天对于日益强大的版权权力的反对几乎只能援引一些旁枝问题，并且从来不敢援引本应作为一种合法公众价值的再分发副本的自由。

从实践角度讲，出版量最大化这一最终目标使得出版商可以如此论述：“某种行为正在削减我们的销量——或者我们认为它将会这样——因此，我们假定它使得出版量下降了一定的未知数量，因此它应当被禁止。”我们将会得到这样一种不可容忍的结论：公众的利益应当以出版商的销量来衡量，但凡对大众媒体公司有利的东西，对美国也都是好的。

## 第三个错误：最大化出版商的权力

一旦出版商得到了这样的许可，即为了达到出版输出最大化的政策目标可以不计任何代价，那么下一步就是推断出这需要赋予它们最大可能的权力——使得版权覆盖于一篇作品任何想象到的使用方式，或者应用某些其他法律工具诸如“拆封包装”许可证来实现同样的效果。这一目标涉及废除“合理使用”和“首次销售权”，它已经被各级政府强制执行，从美国各州到各个国际机构。

这一步骤是错误的，由于严酷的版权规则阻碍了新的有益作品的创作。例如，莎士比亚从其他人于数十年前发布的作品中借用情节用于自己的剧本，因此，如果今天的版权法在当时有效，他的剧本将成为非法。

即使我们真的想要最大可能的出版量而无视公众为之付出的代价，最大化出版商的权力也是达成这一目标的错误方式。作为一种本意在于促进进步的方式，这种做法是自相矛盾的。

## 三大错误的后果

当前的版权立法趋势是赋予出版商更宽泛的、持续时间更长的权力。而版权的概念基础自从被这一系列错误所扭曲之后，几乎没有给读者留下任何说“不”的权利。立法者口头上大肆鼓吹版权服务于公众这一理念，而实际上却一直在赋予出版商们想要的任何东西。

例如，参议员 Hatch 在介绍 S.483 法案[1]时如是说，这条于 1995 年通过的法案将版权的期限延长了 20 年。

> 我相信我们现在需要搞清楚这个问题：当前的版权期限是否足以保护作者的利益，以及与之相关的另一个问题：当前的版权保护期限是否足以继续为作者创作新作品提供足够的激励。[2]

[1] Congressional Record, S. 483, “The Copyright Term Extension Act of 1995,” 2 March 1995, pp. S3390–4.

[2] Congressional Record, “Statement on Introduced Bills and Joint Resolutions,” 2 March 1995, p. S3390, http://gpo.gov/fdsys/pkg/CREC-1995-03-02/pdf/CREC-1995-03-02-pt1-PgS3390-2.pdf.

这一法案延长了那些自 20 世纪 20 年代以后创作的已发表的作品的版权期限。这一改变是对出版商的慷慨赠予，而不可能为公众带来任何好处，由于现在不可能以任何方式逆动地增加当时的出版量。然而，这剥夺了公众的一项在今天看来非常有意义的自由——再分发当时出版的作品的自由。注意，他们所鼓吹的 “保护”[1]一词的用法中包含了三大错误中的第二项。

这条法案同时延长了将要被创作的作品的版权期限，对于出资聘用作者创作的作品，版权期限将会持续 95 年而非当前的 75 年。理论上这将会增加对创作新作品的激励；但是，任何宣称需要这项额外刺激的出版商应当被要求以未来 75 年预期的财务状况表来证明这一要求的合理性。

不言而喻的是，国会并不会质疑出版商的论证：一项版权延长法律于 1998 年实施。它的官方名称是 Sonny Bono 版权期限延长法案，它是以一位当年早些时候去世的支持者命名的。我们通常称之为米老鼠版权法案，由于我们认为它的真实动机是防止米老鼠的形象的版权过期。Bono 的遗孀以他的名义完成了剩下的工作，并作了如下声明：

> 事实上，Sonny 希望版权保护期限持续到永远。我被一位工作人员告知这样一种改变将会违反宪法。我现在邀请各位与我共同努力，以所有可能的方式强化我们的版权法。你们应当知道，还有 Jack Valenti [2]的提案建议版权的保护期限为 “比永远少一天”。也许委员会可以期待下一届国会。[3]

美国最高法院随后听说一个案例，这一案例试图在根本上推翻该版权法律，由于反动地延长版权期限未能符合宪法关于促进进步的终极目标。法庭以放弃其判决该问题的责任作为回应；在版权问题上，对宪法只需要空口应酬。

[1]参见《应避免使用（或慎用）的词语》一文的 “保护（Protection）” 一节 (p. 122) 以获知为何在与版权相关联时应当避免使用 “保护” 一词。

[2]Jack Valenti 是美国电影协会（MPAA）常任主席。

[3]Congressional Record, remarks of Rep. Bono, 7 October 1998, p. H9952, http://gpo.gov/fdsys/pkg/CREC-1998-10-07/pdf/CREC-1998-10-07-pt1-PgH9946.pdf.

另一条于 1997 年通过的法律使得为任何已发表作品复制达到一定数量的副本成为一项重罪，即使您将它们送给遵纪守法的朋友。而在此之前，这在美国甚至不是一项罪行。

一条更坏的法律，数字千年版权法案（Digital Millennium Copyright Act，DMCA），被设计为重新带回一种当时被称为“复制保护”的东西——现在称之为数字限制管理（DRM）[1]——这是一种已经被用户所唾弃的东西，它使得破解这些限制规则，甚至只是发表关于如何破解它们的信息成为犯罪行为。这条法律应当被称作“媒体公司支配法案”（Domination by Media Corporations Act），由于它事实上赋予了出版商编写自家版权法的权力。它宣称它们可以对作品的使用强行施加任何限制，并且这些限制将具有法律效力，如果该作品包含某种加密或者许可证管理器之类的强制实行措施。

用于支持该法案的一种论证是它可以实现一种现代条约以增加版权权力。该条约由世界“知识产权”组织（WIPO）[2]所大肆宣扬，这一组织被版权与专利持有人的利益所支配，并且得到了克林顿政府以施压方式提供的支持；由于该条约只是强化了版权权力，至于它在任何国家是否有利于公众利益尚存争议。无论如何，这条法案的效力大大超出了这项条约所要求的。

图书馆曾经是反对此法案的重要来源，尤其针对该法案中禁止某种可被视为合理使用的复制行为这一方面。那么出版商一方是如何回应的呢？前代理人 Pat Schroeder，现在作为美国出版商协会（AAP）的说客如是说：“出版商与图书馆的诉求不共戴天”。由于图书馆的诉求只是维持部分现状，人们可以以此作为回应：出版商到底是怎样活到今天的。

国会议员 Barney Frank 在一次与我和其他反对该法案的人士举行会谈时，展示了美国宪法中关于版权的观点在多大程度上被无视了。他宣称由犯罪刑罚所支撑的新的版权权力是急迫需要的，由于“电影行业感到担忧”，同样还有“音乐行业”和其他“行业”。我问他：“但是，这是为了公众

[1]参见 http://gnu.org/proprietary/proprietary-drm.html 以获得关于这一问题的更多信息。

[2]参见《您说过“知识产权”吗？这是一种迷惑的幻景》(p. 99) 一文以获知为何这一短语是有问题的。

的利益吗？”他如此回应：“你有什么资格谈公众利益？那些富有创造性的人们不需要为了公众利益而牺牲他们的权利！”在这里“行业”被视同它所雇佣的“富有创造性的人们”，版权被视为它应有的权利，而宪法的本意则被彻底颠倒了。

DMCA 于 1998 年被实施。随之，它宣称合理使用在名义上尚属合法，但它允许出版商禁止所有那些您可能对其进行合理使用的软硬件。事实上，合理使用已被禁止。

基于这一法律，电影业对用于读取和播放数码多功能影音光盘（DVD）的自由软件，乃至关于如何读取它们的信息强行实施了审查。2001 年四月，普林斯顿大学 Edward Felten 教授受到来自美国唱片业协会（RIAA）的法律诉讼的威胁以撤销一篇关于他从一种被提议用于限制对录制音乐访问的加密系统中所学到的东西的科研论文。

我们同样开始见到那些剥夺了众多读者的传统阅读自由的电子书——例如，将书借给您的朋友，或是卖给二手书店，或是从图书馆借阅，或是无需将您的名字告知公司数据库便可购书，甚至是再次阅读某本书等等这些自由。加密的电子书通常限制了以上这些活动——您只能利用某种被设计用于限制您自由的特殊私密软件来读取它们。

我坚决不会购买任何一部这种加密、受限的电子书，并且我也希望您也能拒绝它们。如果一部电子书不能赋予您与纸版书相同的自由，就不要接受它！

任何人擅自发布可用于阅读受限的电子书的软件将面临被起诉的风险。一位俄国程序员 Dmitry Sklyarov 于 2001 年访问美国期间参加一次会议演讲时被逮捕，由于此人在俄国编写了一款这样的程序，而这当时在俄国尚属合法。现在俄国正准备通过一条法律以禁止这种行为，而欧盟最近已经采纳了这样的一条法律。

面向大量最终用户销售的电子书目前看来在商业上并不很成功，但这并非由于读者选择捍卫他们的自由；而是由于其他原因不受欢迎，例如计算机屏幕并不是一种适合于轻松阅读的平面。我们不可能长期依赖这种好的缺陷来保护我们自己；用于促销电子书的下一步尝试是使用电子

纸——一种像书本一样的东西，可以向其中下载加密、受限的电子书。如果这种像纸一样的平面能够提供比现在的显示屏更有吸引力，我们将不得不捍卫我们的自由以便继续拥有它。与此同时，电子书正在利基市场（niche market）中觅得商机：纽约大学（NYU）和其他牙科学校要求其学生以受限的电子书格式购买它们的教材。

媒体公司对此仍不满足。2001 年，由迪士尼赞助的参议员 Hollings 提议了一条“安全系统标准与认证法案”（Security Systems Standardsand Certification Act，SSSCA）[1]，它要求所有计算机（及其他数字录制与回放设备）必须带有政府强制要求的限制复制系统。这是它们的最终目的，但它们日程上的第一件事是禁止任何可以调谐数字高清电视（HDTV）的设备，除非它被设计为使得公众不可能对其进行“恶意篡改”（即以公众自己的目的进行修改）。由于自由软件是用户可以修改的，我们在此首次遭遇了这样一种被提议的法律，它将明确禁止自由软件被用于某种特定任务。可以预见对于其他任务的自由软件禁令也一定会随之而来。如果美国联邦通信委员会（FCC）采纳这一规则，现存的自由软件诸如 GNU Radio 将会被审查。

阻止这些法案和规则的通过和实施需要政治行动[2]。

## 寻找正确的版权交易

那么，决定版权政策的恰当方式是什么呢？如果版权是一笔以公众为名进行的交易，它应当把服务于公众利益放在至高无上的地位。政府在消费公众自由的时候应尽的职责是只去花费那部分必须花费的，并且要换取尽可能多的回报。最起码的要求是，我们应当尽可能减少版权权力的程度，同时又能维持与之相适应的出版量级别。

[1] 由于它被更名为不可拼读的 CBDTPA，一种有助于理解和记忆的解读是“消费但不要尝试编程法案”（Consume, But Don’t Try Programming Anything），而其真正名称是“消费者宽带和数字电视促进法案”（Consumer Broadband and Digital Television Promotion Act）

[2] 如果您真的想要提供帮助，我推荐以下网站：http://defectivebydesign.org, http://publicknowledge.org, and http://eff.org.

既然我们不能通过竞标来找到自由的最小代价，如同我们在对待建设工程时那样。我们怎样才能找到正确的版权交易方式呢?

一种可能的方式是分阶段减少版权特权并且考察其效果，通过观察出版量是否发生了显著减少以及何时发生，我们可以获知为了达到公众预期的目标所真正必需的版权权力是多少。我们必须基于真实的观察来评估这一点，而非根据出版商所说将会发生什么来进行判断。由于它们会尽其所能做出夸张的悲观预期，如果它们的权力以任何形式受到限制。

版权政策包含多种独立的方面，它们可以被单独评估。当我们找到了它某一政策方面所必需的最低限度以后，仍然可能继续减少其他方面的版权限制，以维持理想的出版量水平。

版权的一个重要方面是它的持续时期，而现在的版权期限通常是以一个世纪作为数量级的。将这种对复制的垄断权缩短到 10 年，从该作品发表日期开始计算可能是一种合理的第一步。而版权的另一方面，即覆盖了创作其衍生作品的方面，可以持续稍长一段时间。

为何从作品发表之日开始计算? 由于未发表的作品的版权并未直接限制读者的自由；我们是否拥有复制一篇作品的自由并不重要，如果我们还没有得到它的副本。因此，给予作者一段较长时间来发表作品并不会造成伤害。作者们（通常在作品发表之前确实拥有其版权）很少会仅仅为了推迟其版权终止的日期而选择延迟发布作品。

为何是 10 年? 由于这是一种安全的提案；我们能够以基于实践的理由确信这一期限的减少并不会为当今的总体出版活力带来什么冲击。在大部分媒体和流派当中，成功的作品只在短短的几年之内能够带来丰厚利润，并且即使是成功的作品通常也会在远未到达 10 年期限之时绝版。即使是对于参考文献类的作品，它们的有用生命周期可能长达几十年，10 年的版权期限也是足够的：由于更新的版本还将正常出版，并且很多读者将会愿意购买仍然受到版权保护的当前版本而非复制一份已经处于公有领域中的，十多年前的旧版副本。

10 年也许仍然比真正必需的时间要长；一旦这个问题得到解决，我们可以尝试进一步缩短这一期限来对版权体系进行调整。在一次有关文学作

品版权惯例的研讨会上，我提出了 10 年期限的提议，坐在我身旁的一位著名科幻作家表示强烈反对，他认为超过 5 年期限的任何限制条件都是不可容忍的。

但是，我们不必须对所有类别的作品都采用相同的时间期限。维持版权政策的绝对平均对于公众利益并非至关重要，并且版权法对于某些特殊应用场景和媒体类型已经有了很多例外条款。为每项高速公路工程都按照国内成本最高地区的最困难工程所必需的价格标准付费是愚蠢的，而为每种类型的艺术作品都按照我们所发现的对于任何一类作品所必需的自由代价中的最高标准来“花费”自由同样愚蠢。

因此也许小说、字典、计算机程序、歌曲、交响乐、电影等等应当具有不同的版权期限，于是我们可以将任何一类作品的版权期限缩短到众多待发表的同类作品所必需的程度。也许一小时以上的电影可以拥有 20 年的版权，考虑到创作它的开销。在我本人的领域，计算机编程，3 年应当足够，由于产品的生命周期远比这一时限更短。

版权政策的另一方面是合理使用的限度：对已发表的作品的全部或部分进行复制的某些方式为法律所许可，即使该作品受版权保护。减少这方面的版权权力的第一步自然是允许私人进行偶尔的小批量非商业性复制并且对个人用户分发。这将阻止版权警察对人们私人生活的入侵，但对于已发布的作品的销售很可能并不会造成什么影响（可能有必要采取其他法律步骤以保证拆封包装许可不能被用于取代版权并且藉此限制这种复制的权利）。Napster 的经验显示我们还应允许面向公众的非商业性逐字再分发——由于公众之中有那么多的人想要复制和分享，并且他们发现这是那么有用，只有德拉古式的暴政才会禁止这些，而公众理应得到他们想要的。

对于小说及其他用于消遣的一般作品形式，非商业性的逐字再分发可能足以保证读者的自由。计算机程序被用于功能性的目的（用于完成某项任务）需要在此基础上的更多自由，包括发布改进版本的自由。参见本书《什么是自由软件?》 (p. 1) 一文中“自由软件定义”章节以获知关于计算机软件用户应当享有的自由的解释。而在程序发布之后两三年才能使得这些自由全部可获得也不失为一种可接受的妥协。

以上这些改变将使得版权回归公众希望的，使用数字技术进行复制的意愿轨道上来。出版商将会毫无疑问地发现这些提案是“不公平”的；它们也许会威胁退出出版业，但却不会真正如此，由于这一行业仍然有利可图，并且仍将是最好的行业。

在我们思考如何减少版权权力的同时，我们必须保证媒体公司不会简单地将其改为最终用户许可协议（EULA）。有必要禁止使用合同来对版权范围以外的复制施加限制。对于面向大量最终客户的不可谈判合同所能要求的东西进行限制是美国法律体系中的一个标准组成部分。

## 个人注记

我是软件设计者而非法学学者。我之所以对版权问题感到担忧是由于这在计算机网络世界，例如互联网中，是无法回避的。作为一个用了计算机和网络 30 多年的用户，我衡量了我们已经失去的以及将要失去的自由的价值。作为一位作者，我有权拒绝将作者比作半神的创造者这一富有幻想性的神话，而出版商经常援引这一点作为依据以支持增加作者的版权权力——然而作者随后就会把这些权力出让给出版商。

本文的大部分内容包含了您可以查证的事实和推理，还有您可以用于构建您自己想法的提议。但是我再次请求您接受我个人的看法：诸如我这样的作者并不应该拥有凌驾于您之上的权力。如果您想要对我所编写的软件或书籍表示感谢，我将会心存感激地接受一张支票——但请您不要以我的名义放弃您的自由。

# 科学必须摆脱版权束缚



很多观点都指向这一结论：软件自由必须是普遍性的，这一结论通常同样适用于其他形式的表达性的工作，尽管是以多种不同的方式。本文专注于讨论将这些软件自由领域相关的原理应用于文献领域的情况。通常地，这些议题与软件自由相正交，但我们仍然决定将与本文类似的几篇文章添加进来，由于诸多对自由软件感兴趣的人们也想获知关于如何才能将这些原理应用于除了软件之外的其他领域的细节。

科学文献存在的理由应当是传播科学知识，并且科学期刊存在的理由应当是促进这一进程，这应该是一种不言自明的真理。由此可以推论，使用这些科学文献的规则应当被设计为有助于实现这一目标。

我们现在所拥有的规则，称之为版权，最初建立于印刷机时代，这是一种从本质上来说相对中心化的印刷量产方式。在这样一种印刷环境下，期刊文章的版权仅仅约束期刊发行者——要求它们获得出版文章的许可——以及潜在的抄袭者。它有助于期刊运营以及传播知识，而并未影响科学家们或者学生们的有用成果，不论对于文章的作者还是读者。这种规则可以很好地适应这种体系。

但是，用于科学文献发表的现代技术是万维网（World Wide Web）。那么，什么样的规则才能确保在网络上传播科学文献和知识的效果最大化呢？文献应当以非私有格式发布，并且对所有人开放访问权限。而且每个人应当有权利为文献提供“镜像”，即在满足适当署名的条件下逐字发表原文。

这些规则应当同样适用于过去与未来的文献，只要它们是以电子形式发布的。但是，现在并不急迫需要改变当前的版权体系，由于它们仍然适合于纸版期刊，此处的问题不属于那个领域。

不幸的是，看起来并非所有人都认可本文开头提出的那种不言自明的理念。很多期刊发行者看起来所坚信的是，科学文献存在的目的是使它们能够出版期刊以便收取科学家们和学生们的订阅费用。这种想法可称为“方法与目的的混淆”。

它们采取的方式是仅仅允许那些能够并且愿意为之付费的人们访问，甚至只是允许他们阅读科学文献。它们以版权法为理由禁止科学家们选择新的规则。尽管版权法对于计算机网络时代有着诸多不妥，它们仍然具有法律效力。

以科研协作与人类未来之名，我们必须从根本上拒绝这种方式——不仅仅是在其上建立起来的阻碍进步的体系，还有那些促成了这些体系形成的错误前提。

期刊出版商有时宣称在线访问需要昂贵并且耗能的服务器，于是它们必须收取订阅访问费用以维持其服务器开销。这个 “问题” 是其自身的 “解决方案” 所造成的。如果赋予每个人镜像的自由，世界各地的图书馆将会建立其自己的镜像站点以满足此要求。这种去中心化的解决方案将会减少网络带宽开销并且提供更快速的访问，还能防止学术记录意外丢失。

出版商还会争论它们需要收取订阅访问费用以支付编辑的薪酬。我们姑且接受这一假设，即编辑必须得到薪酬；然而这是本末倒置的。编辑一篇普通论文的成本大约是产出它们的科研项目经费支出的 1% 到 3%。这样小的支出比例很难使得以此为理由阻碍对其结果的使用的行为合理化。

与之相反，编辑的成本可以通过诸如按版面向作者收费的方式得到补偿，作者则可以从科研项目赞助者那里得到补偿。赞助者应当不会介意，即使它们当前不得不以一种繁杂的方式支付论文发表费用，例如通过大学图书馆订阅期刊产生的间接费用。通过改为向科研赞助者征收编辑费用，我们可以消除以此支持限制访问的表面理由。对于作者不属于某一科研机构或公司，以及作者没有科研赞助的少数情形，可免除其版面费用，这部分成本改为向附属于科研机构的作者征收。

另一种支持征收在线出版物访问费用的理由是需要资金支持以便将期刊的印刷归档转换为在线形式。这项工作需要完成，但我们需要寻找其他

的资金支持方式，使其不会造成阻碍访问的结果。这项工作本身将不再是困难并且耗资巨大的。将印刷归档数字化但又通过限制访问使其成果被浪费，这种做法是自相矛盾的。

美国宪法宣称版权存在的理由是“促进科学进步”。而当版权阻碍了科学进步的时候，科学必须摆脱版权的束缚。

## 后期的发展

某些大学——例如麻省理工学院（MIT）[1]——已经采纳了某些政策以限制出版者的权力。然而，更加强有力的政策是必需的，因为 MIT 允许个人作者“退出”（屈服）。

美国政府对某些受资金支持的科研项目提出了称为“公共访问”的要求。这要求在一定时期内的出版物发表在某个允许任何人查看论文的网站上。这种要求是积极的一步，但并不足够，由于它并不包含再分发该论文的自由。

有趣的是，2002 年布达佩斯开放存取倡议（BOAI）所提出的“开放访问”确实包含了再分发的自由。我还是签署了该声明，尽管我对“开放”一词极其厌恶，但其立场是正确的。

然而，关于“开放”一词还有一个笑话。具有影响力的“开放访问”运动的支持者后来在其目标中放弃了再分发的自由。我站在 BOAI[2] 的立场上，但现在“开放访问”已经是其他意思了。我将其称为“可再分发出版物”或“自由镜像出版物”。

[1]“MIT Faculty Open Access Policy,” 于 2009 年三月 18 日经教学科研人员无记名投票通过。http://libraries.mit.edu/scholarly/mit-open-access/open-access-at-mit/mit-open-access-policy/（译者注：目前（2019 年一月）该链接已失效，可于 https://libraries.mit.edu/scholarly/mit-open-access/open-access-policy/ 访问到该内容）.

[2]参见 http://www.budapestopenaccessinitiative.org/ 以获知 BOAI 指导思想。

# 计算机网络时代的版权与社区之争

本文是 Richard Stallman 所做主题演讲之抄本，该主题演讲于 2009 年十月在新西兰基督城（克赖斯特彻奇）会展中心所举行的新西兰奥特亚罗瓦图书馆与信息协会（LIANZA）会议上进行。



**Brenda Chawner**：今天，本人荣幸介绍 Richard Stallman，他的主题演讲由惠灵顿维多利亚大学信息管理学院所赞助。

Richard 致力于推进软件自由已逾 25 年。他曾于 1983 年创始 GNU 计划以开发一种自由的操作系统（GNU 操作系统），并且于 1985 年创立自由软件基金会（FSF）。每当您读取或发送一条 nz-libs 信息时，您就在使用 Mailman 软件，它是 GNU 计划的一部分。因此无论您是否意识到这一点，Richard 的创作已经深入我们每个人的生活。

我想将他描述为最有影响力却不为大多数人所熟知的人物，尽管他曾对我说，这不可能是真的，由于它们的正确性不可能被证实。

**RMS**：不能这样说。

**BC**：我是想说——我仍然喜欢这样说。他关于软件自由以及信息应当可自由获取的理念被 Tim Berners-Lee 用于创建世界上第一台网络服务器，并且他在 1999 年对于一部自由的在线百科全书的深入思考启发了 Jimmy Wales，后者创立了现在的 Wikipedia。

今天，Richard 将为我们带来关于计算机网络时代的版权与社区之争，以及它们对于图书馆的启示的演讲。有请 Richard。

> **RMS**：我来到新西兰已有两周时间，在北岛的大部分时间都在下雨。现在我可以理解为何他们将长筒橡胶靴子称为“惠灵顿靴”。而后我看到了一位使用银叶蕨的木材制造桌椅的工匠，他将其称为蕨类家具（fern-iture）。然后我们乘坐渡船来到此地，当我们下船时，人们立即开始嘲笑我们；但他们没有任何恶意，只是想让我们感受皮克顿当地的风情。

人们邀请我演讲的原因通常是基于我为自由软件所做的工作。而今天这场演讲并非关于自由软件；而是要回答这样一个问题，即自由软件的理念是否可以延伸至其他类型的作品。但为了使这个问题有意义，我最好还是简要介绍一下自由软件意味着什么。

自由软件关乎的是自由而非价格，因此请思考“自由言论”而非“免费啤酒”。自由软件是尊重用户自由的软件，而这里有四项特定的自由是用户总是应当拥有的：

- 自由之零：以您所希望的任何方式运行该程序的自由；
- 自由之一：研究程序的源代码并对其进行修改使程序能够满足您的需求的自由；
- 自由之二：帮助他人——即再分发该程序原始副本的自由；
- 自由之三：贡献您的社区——即再分发您对该程序的改进版本的自由。

如果程序赋予您这四项基本自由，那么它就是自由软件，也就是意味着它的发布和使用所构成的社会体系属于一种尊重用户自由以及用户社区的社会协作伦理体系。但如果以上自由中的任意一条缺失或是不充分，则它称为私有软件、非自由软件或者迫使用户屈从的软件。这是不符合伦理的，也不能称其为对社会的贡献，而是一种权力的攫取。这种不符合伦理的实践不应当存在；自由软件运动的最终目标是终结这种行径。所有软件应当是自由的，于是所有用户也就因此获得自由。

私有软件使得用户陷入孤立无援的困境：所谓孤立，是由于用户被禁止分享；所谓无援，是由于用户不能拥有源代码，因此不能对其进行修改，他们甚至不能研究它以便确定它真正是在对他们做什么，而且众多私有软件拥有恶意功能用于窥探用户、限制用户、甚至为对用户的攻击提供后门。

例如，微软 Windows 操作系统拥有后门使得微软可以强制安装软件更改而无需得到理应作为计算机的拥有者——用户的许可。您可能认为它仍是属于您的计算机，但如果您犯了用它运行 Windows 的错误，实际上是微软拥有了您的计算机。这样计算机应当被扔出窗外，意即要么将 Windows 从计算机中扔出去，要么将这台计算机从房间的窗户中扔出去。

但是，任何私有软件都赋予了其开发者凌驾于用户之上的不公权力。开发者们或多或少地滥用这种权力，但他们都不应该拥有这种权力。您理应拥有对您所进行的计算的控制权，并且不必屈从于任何一家特定的公司。因此，您应当使用自由软件。

在关于自由软件的演说结束时，人们有时会问，这些相同的自由和理念是否也适用于其他事物。如果您在自己的计算机上拥有一份已发表作品的副本，提问您是否应当拥有同样的四项基本自由是有意义的——即您是否拥有这些自由在伦理上是否重要。这就是我今天将要着重论述的问题。

如果您拥有一份除了软件以外的某种东西的副本，对于大多数情况，唯一可能拒绝您的任何一种自由的东西就是版权法。而对于软件则不是这样。使软件成为私有的主要途径是利用合同以及拒绝对用户公开源代码，而版权只是某种次要的备选方案。而对于其他东西，并没有源代码和可执行代码这样的区分。

例如，我们谈论一篇文本，如果您能够看到该文本以便阅读它，那么该文本就没有您所不能看到的东西。因此这并不是存在于软件中的同类问题。大多数情况下，只有版权可以拒绝您的这些自由。

于是这个问题可以被转述为：“版权法应当允许您对已发布的作品做些什么？版权法应当说些什么？”

版权随着复制技术一起发展，因此有必要回顾一下复制技术的发展史。在古代世界，只要您能在书写表面上使用书写工具，那里就有复制技

术的发展。您可以阅读一份副本并且抄写另一份。

这种复制技术的效率相当低下，但它的一个特性就是没有经济规模效应。为了抄写 10 份副本，您需要花费 10 倍于抄写一份副本的时间，它除了书写工具以外并不需要其他特殊设备，同时除了识字以外您也不需要其他的特殊技能。其结果是任何一本书的副本都是以一种去中心化的方式创作的。只要有一份副本，只要某人想要复制它，他就可以这样去做。

古代世界没有像版权这样的东西的存在。如果您有一份副本并且想要复制它，无人会告知您您不被允许如此做——除非当地的首领不喜欢书中的内容，此时他可能会因为您复制了这本书而对您进行处罚。但这不是版权，而是与之紧密相关的其他东西，称为审查。而如今版权常常被用于对人们进行审查的企图。

这种情况持续了数千年，然而，此后复制技术发生了巨大进步，这称为印刷机。印刷机使得复制过程更有效率，但这并不具有一致性。由于批量复制的效率得到巨大提升，但每次仅仅印制一份副本并不会由于印刷机的存在而变得高效。事实上，您最好还是自己进行手抄，这将会比用印刷机印制一份副本更快。

印刷机具有经济规模效应：需要花费大量工作来进行排版，但您随后就可以快速复制很多份副本。同时，印刷机和字模都属于相对昂贵的设备，大多数人并不拥有它们；而且大部分识字的人们也不知道如何使用它们。使用印刷机是一种不同于书写的技术。其结果是一种中心化的复制方式：任何给定的书本的副本只能在少数地方被印制，它们随后可以被运输到有人想要购买副本的任何地方。

版权始于印刷机的时代。从 16 世纪开始，英格兰的版权制度开始成为一种审查体系。我相信其最初的本意只是审查新教徒，但其随后改为审查天主教徒，并且很可能也被用于审查许多其他人。根据这一法律，为了发行一本书，您必须得到皇家许可，而这种许可是以永久垄断权的形式被授予的。这种制度我相信一直被允许存在直到 17 世纪 80 年代（根据 Wikipedia 相关词条，它于 1695 年被废止）。出版商想要重新得到这项权利，但它们实际得到的是与之不同的东西。安娜法令赋予了作者一份版权，

并且仅持续 14 年，尽管作者可以续期一次。

这是一种完全不同的理念——出版商的永久垄断权变成了作者的临时垄断权。这一理念使版权成为了一种促进创作的方式。

起草美国宪法时，一些人希望作者被授予版权权利，但这被否决了。与之相反，美国宪法声明国会可以有选择地采纳一种版权法，并且如果真的要有一部版权法，其目标必须是促进进步。换言之，其目标不是版权持有人或者与他们进行交易的其他人的利益，而是为了公众的利益。版权必须仅仅持续一段有限的时间；而出版商一直都在盼望我们忘记这一点。

这里，我们有了关于版权的这样一种理念，它是一种约束出版商的行业规范，由作者所控制，并且被设计为最终能够为公众带来利益。它能够以这种方式发挥作用，由于它并不限制读者。

在印刷时代的最初几个世纪，我相信直到 18 世纪 90 年代还是这样，大部分读者仍然采用手抄方式进行复制，由于他们买不起印刷的副本。从未有人期望版权法变成行业规范以外的东西，它的本意是约束出版商。正因为此，它容易被强制执行、不会引起争议、并且可以认为是对社会有益的。

它是容易强制执行的，由于它只需针对出版商强制执行。并且想要找出未经许可的出版商也是容易的——您可以前往一家书店并且说：“这些副本来自哪里？”您无需入侵某人的住宅或者某人的计算机以实现这一目的。

它不会引起争议，由于读者并未被限制，他们无需担心什么。从理论上说，他们被禁止从事出版，但由于他们不是出版商，并且没有印刷机可用，他们无论如何不能从事出版。而在他们实际上可以做的事情当中，他们并未受到什么限制。

它可以被认为是有益的，由于根据版权法的概念，公众所出让的是一种他们所不能行使的、仅存在于理论上的权利。作为回报，他们从更多的作品中得到了好处。

现在，如果您出让的是一些您没有任何办法使用的东西，而得到的回报是一些您可以使用的东西，这是一笔有利的交易。不论您是否能够通过

其他方式得到一笔更加划算的交易，那是另外一个问题，但至少这是一笔不亏的交易。

因此，如果现在仍然处在印刷机时代，我想我不会抱怨版权法。但是，印刷机时代正在逐步让位于计算机网络时代——复制技术的另一次革命，它使得复制更加高效，同时也更加不那么具有一致性。

这是我们曾经在印刷机时代拥有的东西：批量复制非常高效，而一次复制一本时仍然和古代一样慢。数字技术为我们带来了这些：它们都得到了提升，然而一次一本的复制方式所得到的效率提升最大。

我们由此来到了一种更像古代世界的境地，彼时一次复制一本的行为相对于批量复制并不显得非常低效（困难），它只是略微低效，但它的成本足够低，以至于数以亿计的人们都能够进行这样的复制操作。想想看有多少人可以不时地烧录光盘（CD），即使是在欠发达的国家。您可能没有一台 CD 刻录机，不过您可以前往一家可提供此服务的商店。

这意味着版权不再像过去那样适应技术的发展。即使版权法中的词句保持不变，它也不会再产生与之前相同的效果。此时的版权不再是一种由作者控制的针对出版商的行业规范，并且由此使公众受益。它已经成为了主要由出版商控制，以作者之名对公众的限制。

换言之，这是一种暴政，这是不可容忍的，我们决不能允许它以这种方式发展下去。

由于这种性质上的转变，版权不再容易被强制执行、不再没有争议、也不再有益。

它不再容易强制执行，由于现在是那些出版商想要针对每个人强制执行。而要达到这一目的，需要严酷的措施、德拉古式的严厉刑罚、对隐私的侵犯以及废除我们关于公平的基本理念。目前看来，他们将会在发动“消灭分享之战”的道路上走多远，一时还看不到界限。

它不再没有争议。在一些国家已经出现了这样一些政党，其基本立场是“分享自由”。

它也不再有益，由于我们曾经在概念上出让（由于那时我们不能行使它们）的自由现在是我们可以行使的了。它们是那么地重要，以至于我们

现在想要行使它们。

那么，一个民主政府在这种情况下应当做些什么呢?

它应当减少版权权力。它应当说：“我们在以公民之名进行的交易中出让的那部分自由是现在他们所需要的，这是不可容忍的。我们必须改变这一切；我们不能轻易出让这种至关重要的自由。”我们可以如此衡量民主的欠缺，通过考察世界各国政府做出与民主背道而驰的事情的倾向性，即它们在本应减少版权权力的时候反而不断延伸版权权力。

一个例子是版权的时间尺度。放眼全世界，我们看到各种压力使得版权持续时间更长、更长、更长。

一波此类事件于 1998 年发生在美国。过去和将来的作品的版权期限都延长了 20 年。我不明白他们怎样才能指望使得那些活跃于 20 世纪 20 至 30 年代的已经去世或者年迈的作家相信，通过在今天延长他们的作品的版权能够促使他们在当时创作更多的作品。如果他们拥有一台可用于在当时通知他们的时光机器，他们也从未使用过它。我们的历史教材也从未说过，在 20 世纪 20 年代曾经发生过文科创作活力的爆发式增长，由于当时所有的艺术家预知他们的作品的版权将会在 1998 年被延长 20 年。

为未来的作品延长 20 年的版权期限将会说服人们为创作这些作品付出更多努力。这在理论上是可信的，但这不能说服那些有理性的人们，由于在未来 75 年——如果这篇作品是出资聘人完成的——如果该作品拥有一位独立版权持有人，可能还会更长一些，在此基础上额外增加的 20 年版权期限的存在价值将会大打折扣，其存在价值小到不能说服任何有理性的人们为此去做一些不同的事情。任何想要宣称事情并非如此的商业公司应当被要求出示它在未来 75 年中的财务状况表，然而它们当然做不到这一点，由于它们当中没有一家会真正看问题看得那么远。

这条法律的真正原因，以及驱使不同的商业公司向美国国会购买这条法律（国会在很大程度上有权决定法律）的原因在于它们已经拥有使其大大得利的垄断权，并且想让这种垄断权持续下去。

例如，迪士尼意识到米老鼠首次出现的那部电影将会在几年内进入公有领域，然后任何人都可以自由地绘制相同的角色以用于其他作品。迪士

尼不希望这件事发生。此前，迪士尼曾经从公有领域借鉴了很多东西供自己使用，但它决定永远也不向公有领域做出哪怕是最微薄的回馈。于是迪士尼买来了这条法律，我们称之为米老鼠版权法案。

电影公司说它们想要永久版权，但是美国宪法不会让它们以官方的方式得到它。于是它们想出了一种方式以便以非官方形式达到同样的结果：“永久版权无限续期方案”。每隔 20 年，它们会将版权期限再次延长 20 年。因此，在任意给定的时间，对于任意给定的作品，都存在这样一个日期，它们应当在此日期进入公有领域。但是，这个日期就如同明日复明日，永远不会到来。当您等到那一天的时候，它们又将其延后了，除非我们能够在下一次阻止它们这样做。

以上这些是一个方面，即版权期限的方面。但是更重要的一个方面是宽度：即版权会覆盖作品在哪些方面的使用。

在印刷机时代，版权不应覆盖一篇作品的所有应用场景。由于版权所管制的某些应用同时也是一系列更宽泛的未受管制的应用中的一些例外。您很自然地被许可利用您的某本书的副本去做某些特定的事情。

现在，出版商有了这样的理念，它们可以使得我们的计算机背叛我们，并且利用它们攫取对于已发布作品的所有使用可能的绝对控制权力。它们想要建立一种按次付费点播的通用规则。它们正在通过数字限制管理（DRM）来实现这一点——软件的功能被故意设计为限制用户。而且计算机本身也通常被设计为限制用户。

公众首次见识到它的方式是通过数码多功能影音光盘（DVD）。保存于 DVD 上的电影通常是加密的，并且其格式是私密的。DVD 阴谋集团保守这一秘密，因为它们宣称任何人想要制造 DVD 播放器必须加入这一阴谋集团，并且承诺保守这一秘密，同时还要承诺设计出的 DVD 播放器必须根据规则限制用户，也就是说，它必须阻止用户做这件事、还有那件事、还有那件事——一系列精准的要求，对我们都是恶意的。

虽然花费了一些时间，但随后某些人还是破解了这种私密的格式，并且发布了可用于读取并播放 DVD 电影的自由软件。然后出版商说：“既然我们不能在事实上阻止他们，我们必须让这件事成为罪行。”它们于 1998

年在美国利用数字千年版权法案（DMCA）开始这么做，这一法案对能实现上述功能的软件强制实施了审查。

因此，这一类特定的自由软件被诉诸法庭案例。它们在美国的分发被禁止；美国对其实施了审查。

电影公司意识到它们不能真正使得那个程序彻底消失——它仍然很容易被搜索到。于是它们设计出了另一套加密系统，并且希望它更难被破解，它被称为高级访问信息系统（AACS），或者“战斧”。

AACS 阴谋集团为所有播放器制定了细致的规则。例如在 2011 年，它将会禁止模拟视频输出。因此所有视频输出将必须是数字式的，它们将会把信号以加密形式传输到一台特别设计为使这些内容对用户保持私密的显示器上。那是一种恶意硬件。它们宣称如此做的目的是“修补模拟漏洞”（*Stallman* 摘下眼镜）。这里有一个，那里还有一个，而它们想让这些漏洞永久消失[1]。

我是如何知道这些阴谋的呢？其原因是它们并不是秘密——它们有其官方网站。AACS 网站骄傲地展示了生产商必须签署的霸王条款，这就是我之所以会知道这些限制条件。它还骄傲地展示了共同建立了这一阴谋集团的商业公司的名字，包括微软、苹果、英特尔、索尼、迪士尼、IBM 等。

这些公司所设计的阴谋旨在限制公众对技术的访问，这应当作为一项重罪被起诉，就像操纵价格的阴谋那样，只是这比操纵价格更坏而已。因此，这种罪行的刑期应当更长。但是，那些公司信心满满，由于政府站在它们那边共同压制我们。它们丝毫不担心可能会因为这些阴谋而被起诉，这就是它们为何甚至不屑于掩饰这一点的原因。

通常，DRM 是由一些公司组成的阴谋集团所实施的。有时一家单独的公司也能这么做，但通常这需要在技术公司和出版商之间达成某种阴谋，因此它几乎总是阴谋。

它们认为无人能够破解 AACS，但是大约在 3 年半之前，某人发布了

[1]在 2010 年，关于数字视频输出的加密系统被正式破解。参见 Mark Hachman 的文章“HDCP Master Key Confirmed; Blu-Ray Content Vulnerable”（2010 年 9 月 16 日），位于 http://pcmag.com/article2/0,2817,2369280,00.asp 以获得更多信息。

一款可以解密那种格式的自由软件。然而，这是完全无用的，因为如果您想要运行它，您需要知道它的密钥。

然后在 6 个月后，我看到了一幅照片，里面有两只可爱的小狗，每只小狗身上有 32 位的十六进制数，我当时觉得奇怪：“为何将这两样东西放在一起？我怀疑这些十六进制数是否可能是某个重要的密钥，而某人可能是将这些十六进制数和小狗放在一起，希望人们复制这张小狗的照片，由于它们是那么可爱。这也许能够保护该密钥免遭被抹除的厄运。”

事实上，它就是——破解“战斧”的密钥。人们发布它，然后网站编辑删除它，由于现在很多国家的法律动员他们对这类信息实施审查。它再次被发布，他们又将其删除；最后他们放弃了，这组密钥在两周之内被发布到了 70 多万个网站上。

这是公众对 DRM 的不满的一次大规模爆发。但这并未赢得这场战争，由于出版商更换了密钥。不仅如此，对于高清 DVD（HD DVD），这种方式足以破解其 DRM，但对于蓝光（Blu-ray）光盘则不行。蓝光光盘拥有一个额外层级的 DRM，至今没有自由软件可以破解它，这意味着您必须将蓝光光盘看做某种与您自己的自由完全不兼容的东西。它们是您所不可能与之共存的敌人，至少对于我们当前的知识水平是如此。

永远不要接受任何被设计为用于攻击您自由的产品。如果您没有可用于播放 DVD 的自由软件，您必须不要购买或者租用任何 DVD，或者以礼物的形式接受它们，除非是稀有的未加密的 DVD，并且确实存在少数这样的东西。我确实拥有几片——但我没有任何加密的 DVD，我不会设法获得它们。

以上这些就是视频方面的情况，但我们也已经遇到了音乐中的 DRM。

例如，大约 10 年前，我们开始见到一种形似音乐光盘（CD）的东西，但是它们的烧录方式和 CD 并不相似。它们不遵守标准。我们称之为“损坏的光盘”，它们所遵循的理念是它们可以用某种音频播放器进行播放，但不可能在计算机上读取。这些不同的方法具有不同的问题。

最终，索尼想出了一个奸诈的主意。它将一个程序存储在盘片上，于是当您将盘片放入计算机中时，该盘片将会安装该程序。这个程序被设计

为像病毒一样取得系统的控制权。它被称为 “root kit”，这意味着它拥有破解系统安全措施的能力，使得它可以将软件植入系统深处，并且修改系统。

例如，它修改了您可用于查看系统状态以获知某个软件是否存在的命令，因此它隐藏了自己的存在。它修改了您可用于删除它的某些文件的命令，因此这些命令并不能真正删除它们。现在，所有这些都是严重的罪行，但这还不是索尼所犯下的唯一罪行，由于该软件同时包含自由软件代码——这些代码是以 GNU 通用公共许可证（GNU GPL）发布的。

现在，GNU GPL 是一种左版（copyleft）许可证，这如同说：“是的，您拥有将此代码整合到其他程序中的自由，但如果您决定这么做，您将此代码整合到其中的整个程序也必须以相同的许可证作为自由软件发布。并且您必须保证用户可获得源代码，以及为了保证他们对于自己的权利拥有知情权，您必须在他们得到软件的同时为他们提供一份此许可证的副本。”

索尼并未完全遵守这些。这属于商业版权侵犯行为，是一种重罪。这两种行为都是重罪，但索尼并未因此被起诉，由于政府明白，政府和法律的目的是维护这些公司凌驾于用户之上的权力，而非以任何方式帮助用户捍卫他们的自由。

不堪忍受的人们起诉了索尼。然而他们犯了一个错误。他们并未将谴责集中于这一阴谋的邪恶目的之上，而是仅仅局限于索尼将其次要罪恶付诸实现的不同方法上。于是索尼化解了这些法律诉讼，并承诺在未来继续侵犯我们的自由的时候，它将不会再次采用这些方法。

事实上，这种 “损坏的光盘” 的阴谋并不是非常地坏，由于只要您不用 Windows，它不会对您产生任何影响。甚至即使您正在使用 Windows，在您的键盘上有这样一个键——如果您每次都记着按住它，该盘片就不会安装那个软件。但是，每次都记着按住那个键显然不是一件容易做到的事情；您总有一天会疏忽。这展示了我们必须去设法应对的某种事情。

幸运的是，音乐所受的 DRM 威胁正在减少。即使是大型唱片公司也会销售不带 DRM 的下载版。但是，我们又看到了一波试图为电子书强行施加 DRM 的攻势。

您应该看到了，出版商想要剥夺读者的传统阅读自由——诸如从公共

图书馆借阅；或者将书借给朋友；或者将书卖给旧书商店；或者使用现金匿名购书等自由（这是我购书的唯一方式——我们必须抵御各种诱惑，不让当权者知道我们所做的每一件事情）。

甚至是您想要保有该书任意长的时间，或者阅读该书任意多次的自由，它们也计划剥夺。

它们通过 DRM 的方式做到这一点。它们知道有那么多的人读书，如果直接剥夺他们的这些自由，他们将会强烈反对，因此它们不认为自己能够简单地通过购买一条法律来特定地废除这些自由——这将会招致太多反对。民主是有缺陷的，但有时人们也能要求得到某些东西。于是它们想出了一种两步走的方案。

首先，剥夺电子书的这些阅读自由，然后，迫使用户从纸版书转向电子书。它们已经成功做到了第一步。

在美国，它们通过 DMCA 做到了这一点，而在新西兰，这是版权法案（2008 年）的一部分，即对于可用于破解 DRM 的软件实施审查。这是一种不公平的条款，它应当被废除。

第二步是说服人们从纸版书转向电子书；这项进展并不十分顺利。

2001 年，一家出版商想出了一种方法，如果它能够以我的传记开头，它的系列电子书将会变得非常流行。于是它们找到一位作者，后者询问我是否愿意合作，我说：“除非该电子书以未加密并且无 DRM 的形式发布。”出版商不愿意接受这一条，而我则强烈要求这样——我最终拒绝了。最后我们找到了另一家出版商，它愿意如此做——事实上是希望使用自由许可证发布该书以赋予您四项基本自由——于是该书最终出版，并且售出了很多份纸版副本[1]。

但不管怎么说，电子书在这十年之初并未取得预期成果。人们并不十分愿意阅读它们。我曾说过：“它们将会卷土重来。”我们见到了数量可观的关于类似电子墨水的新闻文章（或者称为电子纸，我记不清是哪个了），我认为之所以会有那么多文章，很可能是由于出版商想让我们考虑这种载

[1]此书即《Free as in Freedom》。中文译名《若为自由故：自由软件之父理查德 · 斯托曼传》，邓楠、李凡希翻译，人民邮电出版社 2015 年出版。——译者注。

体。它们希望我们迫切地想要成为下一代电子书的读者。

现在，它们终于来了。诸如索尼碎纸机（Shreader，它的官方名字是 reader，但是如果您加上 sh 两个字母，这就能很好地解释它是被设计用于对您的书做什么的了）和亚马逊诈骗（Swindle）这样的设备是被设计为用于在您毫无戒备的情况下骗走您的传统阅读自由的。当然，亚马逊将其称为 Kindle（焚烧）恰如其分地解释了它将会对您的书做些什么。

亚马逊焚书机（Kindle）是一款极端恶意的设备，几乎和微软 Windows 操作系统一样坏。它们都拥有间谍功能，都拥有 DRM，都拥有后门。

对于 Kindle，您唯一可能的购书方式就是从亚马逊购买，它要求您提供所有个人信息，于是它们知道关于您所购买的东西的所有信息。

由于存在 DRM，您不能将书借出或是卖给旧书商店，图书馆也不能将书借出。

另外是存在后门，这是我们在大约 3 个月前因亚马逊使用了它而得知的。亚马逊向所有 Kindle 发送了一条指令以删除一本特定的书，即乔治·奥威尔（George Orwell）的《一九八四》。是的，它们找不出另一本更具讽刺意味的书来删除了。这就是我们如何知道亚马逊拥有某种后门并且通过它远程删除电子书的。

它还能做出什么事情来呢？谁知道。也许它就像微软 Windows。也许亚马逊可以远程升级它的软件，这意味着所有那些今天还没有的恶意功能，它们都可能在明天被带来。

这是不可容忍的——这其中的任何一条限制都是不可容忍的。它们想要创造这样一个世界，再也无人能够将书籍借给他人。

假设您前去拜访一位朋友，他的书架上并没有一本书，这并不是由于您的朋友从来不读书，而是由于他的所有书都保存在某一台设备中，他当然不能将那些书借给您。他所能够借给您任意一本书的唯一方式是借给您他的整个图书馆。请求他人如此做显然是荒唐的。因此，热爱读书的人们之间的友谊将不复存在。

请您确保告诉人们这种设备意味着什么。它意味着其他读者不再是您的朋友，因为您在他们面前表现得像个傻瓜。一定要先发制人地传播这句

话。这款设备是您的敌人，它是任何读者的敌人。那些尚未意识到这一点的人们是那些思想短视，未能看穿这一点的人们。我们的任务是帮助他们透过眼前的易用性看穿这款设备的本质。

我绝不反对以数字形式发布书籍，如果它们并非被设计用于剥夺我们的自由。严格地说，要想拥有这样一款电子书阅读器是可能的：

- 不是被设计用于攻击您；
- 运行自由软件而非私有软件；
- 不带 DRM；
- 不要求用户购书时提供身份信息；
- 不带后门；
- 不限制您可以用它对您的计算机上的文件做什么事情。

这是可能的，但是，真正推广电子书的大型出版公司如此做的目的都是侵犯我们的自由，这是我们无论如何不能容忍和支持的。这正是政府和大型出版企业正在合谋所做的事情，通过使得版权变得越来越严酷和龌龊，越来越具有约束性来侵犯我们的自由。

但是，它们应该做些什么呢？政府应该减少版权权力。以下是我的一些特别建议。

首先，是关于版权的时间尺度。我提议版权应该仅仅持续 10 年，从作品发布的日期计算。

为何从作品发布的日期开始计算？由于在此之前，我们并没有任何副本。我们是否被允许复制那些我们尚未拥有的作品副本并不重要，因此我认为我们同样可以允许作者拥有足够的时间来安排作品的发行，然后再开始计时。

那么为何是 10 年？我不知道这个国家（新西兰）情况如何，但在美国，出版周期正在变得越来越短。现在几乎所有的书都只在 2 年内被廉价

出售，并且会在 3 年内绝版。因此，10 年的期限是通常出版周期的 3 倍有余——这应当足以令人满意。

但是并非所有人都同意这一点。我曾在一次同科幻作家的研讨会上提出这一观点，而坐在我身边的一位获奖科幻作家说道：“10 年？不可能。超过 5 年的任何东西都是不可容忍的。”您可以看到，他与他的出版商之间进行了与法律有关的争论。他的书似乎将要绝版，但是出版商并不同意。该出版商正在使用此人自己所著的书的版权来禁止此人自行分发其副本，而他则希望如此做以使得人们可以阅读此书。

这正是每位作者正在开始要求的——他们想要分发自己的作品使其被人们阅读和感激。极少有人能赚得盆满钵满。这很小的一部分大发横财的人正在面临道德败坏的风险，比如 JK Rowling。

在加拿大，JK Rowling 取得了一项针对那些从某家书店购买了她所著的书的人们的禁令，强行要求这些读者不得阅读该书。作为回应，我号召发动了一场抵制《哈利波特》丛书的运动。但我并未说您不应该阅读它们，我将这种抵制留给作者和出版商。我只是说，您不应该购买它们。

只有少数作者能够赚到足以使得他们腐化堕落的钱。他们中的大多数远未达到这种境地，并且继续要求那些他们从一开始就在诉求的东西：他们想要自己的作品受到感激。

他想要分发自己的书，但版权禁止他如此。他意识到超过 5 年的版权期限不可能为他带来任何好处。

如果人们更想让版权仅仅持续 5 年，我不会反对。我只是提议将 10 年作为解决问题的第一步。首先让我们将其缩短到 10 年，然后观望一段时间，接下来我们就可以继续调整它。我并没有说我认为 10 年就刚好是恰当的数字——我也不知道到底应该是多少。

那么，关于版权的宽度又如何呢？版权应当覆盖哪些活动？我在此区分三大类作品。

首先，有一些功能性的作品是您可以在您的生活中将其用于执行一项实践任务的。这包括软件、菜谱、教材、参考文献、字体、以及您所能想到的其他东西。这些作品应当是自由的。

如果您在生活中使用这样一件作品来完成一项任务，那么如果您不能修改该作品以使其适合您的需求，您就不能控制您的生活。一旦您更改了该作品以适应您的需求，那么您应该拥有发布它的自由——发布您的版本——因为将会有其他人想要得到您所做的修改。

这很快就能推出结论，用户必须拥有同样的四项基本自由（对于所有功能性的作品）而非仅限于软件。并且您将会注意到，对于菜谱，或者从实践角度讲，烹饪的行为总是在分享并且改进菜谱，如同菜谱是自由的一般。设想一下，人们将会作何反应，如果政府试图禁止所谓的“菜谱盗版”。

“盗版”一词完全是一种鼓吹的概念。当人们问我关于音乐盗版的看法时，我会说：“就我所知，当海盗们发动攻击时，他们并非采取拙劣地演奏乐器的方式，而是使用武力。因此这不是音乐‘盗版’，由于海盗行为是攻击船只，而分享和攻击船只的行为在道德上相去甚远。”攻击船只是坏事，而与他人分享则是好事，因此我们应当在我们听到“盗版”这一鼓吹的概念的时候给予坚决的揭露。

早在 20 多年前，人们可能会反对这一点：“如果我们不放弃自己的自由，如果我们不让这些作品的出版商支配我们，这些作品就没有机会被创作出来，这是一种恐怖的灾难。”现在，请看自由软件社区，还有那些广为流传的菜谱，还有诸如 Wikipedia 这样的参考文献——我们甚至还开始见到自由的教科书的发布——我们就可以知道这种恐惧是被误导的结果。

完全没有必要因为想到若非如此作品就不会被创作出来而感到绝望并且因此放弃我们的自由。还有很多方式可以鼓励作者来创作这些作品，如果我们想要更多的作品——很多方式都可以维持并且尊重我们的自由。对于这一大类作品，它们都应当是自由的。

那么，对于第二大类作品又如何呢？即对于那些包含人们的思想的作品，诸如回忆录、议论文、科研论文[1]以及其他不同形式的作品。发布某些其他人对其自身思想的论述的修改版本将会曲解此人的本意，这绝不是对

[1]在 2015 年，我加入了科研论文这一条，由于我认为发布他人的论文的修改版本是有害的；然而，将物理和数学论文以创作共用：署名许可证发布在 arxiv.org 或是众多其他自由期刊上似乎并不会带来什么问题。于是，我随后得出结论，科研论文应当是自由的。

社会的贡献。

因此，创立这样一种在一定程度上弱化的版权体系是可行并且可接受的，在此体系中，所有商业应用受版权限制，所有对原作的修改受版权限制，但是任何人都拥有对原始版本的副本进行非商业性再分发的自由。

这些自由是我们必须建立起来的针对所有已发布作品的最低限度自由，因为正是对这些自由的否定发动了“消灭分享之战”——它创造恶毒的鼓吹的概念：分享就是盗窃，分享（盗版）行为就像海盗攻击船只的行为。这是荒谬的，但是这种由金钱支撑的荒谬理念腐化了我们的政府。我们需要终结“消灭分享之战”；我们需要使得分享任何已发布作品的原始版本合法化。

对于这第二大类作品，以上这些就是我们所需要的；我们不必须使其成为自由的。因此我认为拥有一种覆盖了商业应用以及任何修改的弱化的版权体系是可以接受的。这可以为作者提供一种收入来源（通常是不足以体现其应得收入的），以某种与现有体系或多或少地相同的方式。您应当留意在现有体系中，除了少数明星大腕，绝大多数作者完全没有得到应得的收入。

那么对于艺术娱乐作品又如何呢？在此，我花费了一些时间来决定如何看待对作品的修改。

您可以看到，一方面，一幅艺术作品可能具有某种艺术上的完整性，对其进行修改可能会破坏这种完整性。当然，版权并不会必然地防止作品以这种方式被糟蹋。好莱坞就一直在这样干。另一方面，修改一幅作品也可能成为对艺术的一种贡献，它使得民间传承的素材也能变得富有美感和意蕴。

即使我们仅仅着眼于著名的作家：比如莎士比亚，他从几十年前发布的作品中借鉴故事情节并且以其他方式进行呈现，最终创作出了文学史上的巨著。如果今天的版权法在当时也存在，这种创作方式将被禁止，那些剧本也不可能被创作出来。

但是，我最终意识到修改一幅艺术作品可能成为对艺术的贡献，然而这在大多数情况下并非急迫需求。如果您必须等上 10 年使得版权过期，您

是可以等上那么长时间的。不像现在的版权法迫使您等上 75 年或是 95 年。在墨西哥，有时您可能不得不等上将近 200 年，由于墨西哥的版权将会在作者去世 100 年后过期。这是极其不理智的，但是如我所提议的 10 年，作为版权应该持续的时间，是人们可以在有生之年等到的。

因此我提议相同的弱化版权覆盖商业应用和作品修改，但每个人都应当拥有对其原始版本进行非商业性再分发的自由。10 年之后，它将进入公有领域，人们可以通过发布他们的修改版本来继续为艺术发展做出贡献。

还有一件事情，如果您想要从一系列作品中挑选一些片断并且将其整合为某种全新的作品，这应当是合法的，由于版权的初衷是促进艺术发展，而非阻碍艺术发展。使用版权来禁止这种使用作品片断的创作方式是愚蠢的——它是自相矛盾的。这是一种扭曲，只有当政府被那些现存成功作品的出版商所支配的时候才会发生这种事情，并且这已经完全迷失了其初衷。

这就是我所提议的，特别要指出的是，这意味着在互联网上分享副本必须是合法的。分享是美好的。分享构建了社会的契约。反对分享就是反对社会。

因此，每当政府提议新的措施以制裁分享作品的人们并且强行禁止他们分享时，我们必须认识到这是一种罪恶。不仅仅是因为这些被提议的制裁措施无一例外地侵害了公平这一基本理念。但这并非巧合；其原因是这种提议的目的是罪恶的。分享是美好的，政府应当鼓励分享。

但不管怎么说，版权确实有其有用的目的。只是版权作为实现该目的的手段在当下遇到了问题，由于它不能适应当今我们所使用的技术。它干涉了读者、听众、观众及其他人的所有基本自由，但它促进艺术发展的目标仍然是美好的。因此，版权系统将继续作为版权系统而存在，除了在一定限度上减少它的权力。我提议两种其他方式。

其一是（通过）征税（来实现）——直接将税金分发给作者。这可以是一种特殊税收，可以是对于互联网连接服务的税收，也可以来自一般性的收入，由于它的总量不会非常大，如果它是通过某种有效的方式进行分发的。以促进艺术发展为目的对其进行有效分发意味着不是按照作品流行

度的线性比例进行分发。它应当基于流行度，由于我们不希望由官僚主义者独断决定哪些作者应该得到支持而哪些作者将会被无视。但是，基于流行度并不必然意味着按照线性比例进行分发。

我所提议的是统计不同作品的流行度，您可以通过投票（样本统计）做到这一点，并且不是每个人都被强制参与，然后取其立方根。立方根就像这样：它基本意味着（收入）增加值将会逐渐变小。

如果某位明星大腕 A 的流行度是某位成功艺术家 B 的 1000 倍，按照这种体系，A 将得到 10 倍于 B 的收入，而非 1000 倍。

如果按照线性比例，A 将得到 1000 倍于 B 的收入，这意味着如果我们想让 B 得到足以维持其生计的钱，我们必须使 A 赚得盆满钵满。这是一种对税金的浪费——不应该这样做。

但如果我们使得收入增量逐渐变小，那么是的，每位明星仍将得到比普通的成功艺术家多得多的收入，但所有这些明星的收入总和将只占税金（总额）的一小部分。大部分税金将会被用于支持一大批相对成功、相对受欢迎、相对流行的艺术家。因此，相对于现有的体系，这一体系将能够更有效地利用这笔税金。

现有的体系是倒退的。它实际上，例如对每张唱片，给予明星大腕的钱远远超过给予其他任何人的。这种花钱的方式非常不合理。其结果是我们按照这种方式所实际花费的钱要少得多。我希望这足以安慰那些对税收有着如同膝跳反应般的敌意的人们——我并不反对税收，因为我信任一个福利国家。

我还有一个建议是关于志愿者捐助的。假设每台播放器都有这样一个按钮，您可以通过按下它来向您正在播放的作品或者已经播放的上一部作品的作者发送 1 美元。这笔钱将会被匿名地发送给作者。我相信会有很多人比较经常地按下那个按钮。

例如，我们当中的所有人应当能够负担得起每天按一次按钮的费用，这不会花费我们那么多钱。这笔钱对于我们是可接受的，我非常确信。当然，也会有贫困的人们不能负担按一下按钮的费用，然而即使他们不去按下按钮也没关系。我们不需要从穷人身上榨取钱财来支持作者。有足够多

的经济状况尚可的人们，他们可以很好地完成这件事。我相信您已经注意到，有很多人确实喜爱某种艺术，并且非常乐意支持作者。

于是我又有了一个想法。播放器还可以给您一份证书，表示您曾经支持过某位无名作者，它甚至还可以统计您曾经送出多少次支持，并且给您这样一份证书："我为这些艺术家付出了这么多。"总之，我们可以通过很多方式鼓励那些想要这么做的人们。

例如，我们可以开展一场友善的 PR 活动："您今天向某位艺术家捐助 1 美元了吗？为什么不呢？这只是 1 美元——您将永远不会失去它，您难道不喜爱他们所创作的作品吗？现在就按下按钮！"这将使人们产生好感，他们将会想："是的，我喜爱我所观看的作品，我将要捐助 1 美元。"

这在一定程度上已经开始产生效果。有一位曾经叫做 Jane Siberry 的加拿大歌手。她将自己的音乐放在自己的网站上，邀请人们下载并且捐助他们所愿意支付的任意金额。她曾经报道收到了平均每份副本多于 1 美元的收入，这是有趣的事情，由于大型唱片公司的收费还不到每份副本 1 美元。通过让人们自行决定是否捐助与捐助多少，她实际上得到了更多——如果按照实际下载了某些东西的访问者来统计，她通过每位访问者得到的甚至更多。但这可能还没有考虑是否存在这样一种效应，即吸引更多人访问，从而增大了这一平均值所对应的分母。

因此，这是可行的，但在当前环境下是一种令人头痛的事。首先您必须拥有一张信用卡，这意味着您不能进行匿名捐助。并且您必须找到去哪里进行支付，而对于小额支付，现有的支付系统并不非常高效，因此艺术家们只是成功了一半。如果我们能够建立一种理想的支付系统，它将能够更好地实现我们的目的。

在 mecenat-global.org 网站[1]，您可以找到结合了这两条建议思想的另一种方案，它由 Francis Muguet 发明，并且被设计为旨在更好地适应现存法律体系以使其更容易实施。

当心这种关于"补偿版权持有人"的鬼话，由于当他们在说"补偿"的

[1]此网页不再可访问，请移步至 https://stallman.org/mecenat/global-patronage.html

时候，他们是在试图假设如果您感激一篇作品，您已经欠下了某人一笔特殊的债务，并且您必须“补偿”此人。当他们说“版权持有人”的时候，它应该是在迫使您认为这是在支持艺术家，而实际上这是在支持出版商——正是那些从根本上剥削了所有艺术家的出版商（除了少数各位都听说过的艺术家，由于他们那么出名，以至于他们拥有了某种势力）。

我们并不欠任何人任何东西；我们没有“补偿”任何人的理由和义务。但是，支持艺术发展仍然是有益的事情。这曾经是版权的动机，彼时的版权仍然是适应那个时代的技术发展水平的。而今天，版权对于支持艺术发展成了一种坏的实施方式。但以尊重我们的自由的方式来实施它，这还是好的。

我们想要利用它们改变新西兰版权法案中的两大邪恶部分，它们不应取代“三振出局法”[1]，由于分享是美好的，它们应当摆脱针对用于破解 DRM 的软件的审查。当心反假冒贸易协定（ACTA），它们试图在不同国家之间商定一种条约，使这些国家可以攻击它们的国民，我们不知道如何攻击，由于它们不会告诉我们。

[1]新西兰实施了一种未经审判的刑罚系统，对于被指控进行复制的互联网用户；然后，迫于大众的抗议，政府并未立即实施，而是宣布了一项计划以实施一种修订过的但仍然不公平的刑罚系统。本文此处的观点是，它们不应实施一种替代方案——与之相反，它们不应拥有这类系统。然而，我的用词未能清楚地表达这一点。新西兰政府随后实施了一种与原本计划的或多或少相同的刑罚方案。

# 第 4 部分

# 软件专利：对程序员的威胁

## 软件专利和文学专利

 本文最初于 2005 年 6 月 23 日以标题 “Patent Absurdity” 发表于英国卫报，并且针对当时被提出的 “欧洲软件专利指导意见”。

当政治家们考虑软件专利的问题的时候，他们通常会盲目地进行表决；由于他们不是程序员，他们并不了解软件专利真正在做什么。他们通常认为专利法与版权法类似（“除了少数细节以外”）——然而事实并非如此。例如，当我公开询问法国工业部长 Patrick Devedjian 关于法国将如何针对软件专利问题进行表决的时候，Devedjian 用一段对版权法的充满激情的辩护作为回应，并且盛赞维克多 • 雨果为了使版权这一概念为人类所接纳所做出的贡献。(“知识产权” 这一带有误导性的短语加深了这种混淆——这是它永远不应该被使用的原因之一)。

那些认为软件专利将会产生与版权法类似效果的人们未能抓住并认清软件专利所可能带来的灾难性后果。我们可以以维克多 • 雨果为例，说明这二者的区别。

一部小说和一个现代化复杂程序具有某些共同点：其一是大型化，以及在实现它的过程中需要将诸多设计思想融会贯通。于是，让我们沿着这个类比，假设专利法已经于 19 世纪被应用于小说；并且假设诸如法国这样的国家允许对文学创意进行专利保护。这将会对维克多 • 雨果的写作带来怎样的影响？文学专利带来的影响与文学版权带来的影响相比又会如何呢？

考虑维克多 • 雨果的名著《悲惨世界》。由于他是作者，该书的版权属于他本人。他完全无需担心某些陌生人对他进行侵犯版权诉讼并且胜诉。那是不可能的，由于版权仅仅覆盖一篇作品的作者身份的细节，而非蕴含在作品中的思想创意，并且仅仅限制对作品的抄袭。维克多 • 雨果并未抄袭《悲惨世界》，所以他不会受到版权的威胁。

专利发挥作用的方式则与之不同。专利将会覆盖想法和灵感；每一项专利都是对实施某种想法的垄断，这是描述在专利本身的内容之中的。以下是一项假想的文学专利案例：

- 要求 1：一种在读者脑海中表达了这样一种概念的交流过程：某一角色在监狱中被监禁了很长时间，由此对社会和人性产生怨恨；
- 要求 2：一种基于要求 1 的交流过程：该角色随后在他人的关怀下得到了道德的救赎；
- 要求 3：一种基于要求 1 和要求 2 的交流过程：该角色在故事发展过程中改名换姓。

如果这样一项专利存在于 1862 年《悲惨世界》发表之时，该著作将会与上述全部三项专利要求相冲突，由于所有这些事情都发生在小说中的冉阿让（Jean Valjean）[1]一个角色身上。维克多 • 雨果可能会因此被起诉，而

[1]这里选用了人民文学出版社出版的《悲惨世界》（李丹、方于译）对小说主人公的翻译——冉阿让。《悲惨世界》讲述了主人公冉阿让因偷窃面包而获刑长达十九年，出狱后被米里哀主教救赎，之后他多次改名换姓重获新生，并试图赎罪的故事。小说人物丰富，时间跨度大，涵盖拿破仑战争和 1832 年巴黎共和党人起义等政治现象。——译者注

且一旦被起诉，他将会败诉。这部小说可能会被禁止——事实上是被审查了——由专利拥有者。

现在考虑这样一项假想的文学专利：

- 要求 1：一种在读者脑海中表达了这样一种概念的交流过程：一位角色在监狱中被监禁了很长时间，并且随后改名换姓。

《悲惨世界》同样会被这项文学专利禁止，由于这一描述同样适用于冉阿让的人生故事。现在还有这样一项假想的文学专利：

- 要求 1：一种在读者脑海中表达了这样一种概念的交流过程：一位角色得到了道德的救赎并且随后改名换姓。

冉阿让同样会被这项专利所禁止。

所有这三项专利都将覆盖并且禁止这一角色的人生故事。它们相互重叠，但它们又都不是对其他专利的精确复制，因此它们可以共存；三项专利的持有人都可以起诉维克多•雨果，而他们中的任何一位都可以禁止《悲惨世界》。

这项专利同样会被违反：

- 要求 1：一种交流过程表现了这样一位角色，其名字与其姓氏的最后一个音节相同。

这是通过角色的姓名 “冉阿让” 而产生的，然而至少这项专利是容易回避的。

您可能认为这些想法过于简单，以至于没有任何一家专利机构会去批准它们。我们作为程序员通常也会对于真实的软件专利所覆盖的想法是如此地简单而感到惊讶——例如欧洲版权厅（EPO）批准了一项关于进度条的专利，以及一项接受信用卡支付的专利。这些专利看起来将会显得非常可笑，如果它们并非如此阴险。

《悲惨世界》的其他方面也可能与专利产生冲突。例如，可能有关于对滑铁卢战役进行小说化描写的专利，或是关于在小说中使用巴黎俚语的专

利。这将带来两场新的法律诉讼。事实上，并没有关于同时适用于对某一作品诸如《悲惨世界》的作者进行起诉的专利数量的限制，所有那些专利持有人都会宣称他们理应得到奖励，由于他们那些受专利保护的思想和灵感为文学发展所带来的贡献；然而，这些障碍并不能促进文学发展，反而只会阻碍文学的发展。

然而，一项非常宽泛的专利将会使得以上这些问题失去意义。考虑一项带有宽泛要求的专利：

- 一种交流过程，带有连续数页的旁白结构；
- 一种旁白结构，有时类似于赋格曲或者即兴作品；
- 一种围绕特定角色之间的冲突所展开，随后又依次为其他人布下机关的阴谋；
- 一种表现了诸多社会阶层的旁白；
- 一种揭示了隐藏的阴谋的旁白。

谁会是这些专利的持有人？可能是其他小说家，也许是大仲马或者巴尔扎克，由于他们写过这样的小说——但并不一定。要想为软件创意申请专利，其本人不需要进行编程，因此如果我们假想的文学专利符合真实的专利体系，其持有人不需要自己写小说，或者故事，或者任何东西——除了专利申请。那些除了威胁和法律诉讼，没有创造任何东西的专利寄生虫公司、企业正在日渐繁荣。

如果有了这些宽泛的专利，维克多·雨果也许不会走到询问他所使用冉阿让这一角色将会受到哪些专利起诉这一步，由于他根本不会考虑去写这样一部小说。

这种类比可以让非程序员看清软件专利到底在做什么。软件专利覆盖了功能，例如在文字处理器中定义缩写，或是在电子表格中自然顺序的重新计算。专利也覆盖了程序设计所需要使用的算法。专利还覆盖了文件格式的方方面面，例如微软 Office 开放可扩展标记语言（OOXML），而 MPEG 2 视频格式被多达 39 项美国专利所覆盖。

正如一部小说可以同时与许多不同的文学专利相冲突，一个程序也可以同时被许多项软件专利所禁止。想要鉴定所有可能与某个大型程序相关的专利的工作量如此之大，以至于只有一项这样的研究被实施。在 2004 年关于 Linux，即 GNU/Linux 操作系统的内核，的研究发现 283 项不同的美国软件专利可能对其进行覆盖。也就是说，这 283 项专利中的任何一项都可能禁止可以在 Linux 的数千页源代码中的某处找到的计算过程。与此同时，Linux 内核只占整个 GNU/Linux 操作系统的 1%。那么，这样一个系统的发布者又将会受到多少项专利的起诉呢?

要想阻止软件专利对软件发展造成的危害的方法很简单：不要批准它们。这应该是容易做到的，由于大部分专利法都拥有反对软件专利的条款。它们通常会说 “软件本身” 不能被专利保护。但是，全世界的版权机构正在试图曲解其涵义并且为程序设计中实现的想法和创意提供专利保护。除非我们能够阻止这一趋势，这种趋势将会将所有软件开发者置于险境。

# 软件专利的威胁



本文是 Richard Stallman 于 2009 年 11 月 8 日在位于新西兰惠灵顿的惠灵顿维多利亚大学所做的演讲的未编辑抄本。

我所做的最出名的事情是发起自由软件运动以及领导开发 GNU 操作系统——尽管大部分使用它的人们错误地认为它是 Linux，并且认为它是由其他人在十年之后发起的。但是我今天并不打算谈论与之相关的任何话题。我在此是想讲述一种对所有的软件开发者、发布者和用户的合法威胁：专利的威胁——不论是对于计算思想、计算技术还是对于您所能在计算机上进行的任何事情的想法。

现在，为了认识这个问题，您需要认识到的最重要的事情是，专利法与版权法没有任何关系——它们是完全不同的。对于您所了解的关于它们之一的任何东西，您都可以确认它们并不适用于另一方。

于是，举个例子，只要某人做出关于“知识产权”的论述，这就是在散播一种混淆。由于它不仅仅将专利法与版权法混为一谈，并且还混入了至少数十种其他东西。它们都是不同的，其结果是，任何大意是关于“知识产权”的论述都是一种混淆——要么做出这种论述的人本身处于混乱不清的状态，要么此人正在试图使其他人混乱不清。但不论何种情况，不论是无意的还是恶意的，这都是一种混淆。

您必须拒绝接受任何使用那个短语的论述，才能使自己不被混淆。要想对于这些法律中的任何一个做出深刻的评价，并且进行清晰的思考，首先只有将其同其他东西区分开来，并且专注于讨论或思考某一特别的法律。这样，我们才能了解它真正在做什么，并且对其得出结论。因此，我在此将会讨论专利法，以及在那些允许专利法对软件进行限制的国家发生了什么。

那么，专利到底在做什么？一项专利是一种具体的、由政府批准的对

于某一种想法的使用权的垄断。在专利中有一部分称为权利要求，它精确地描述了您所不被允许做的事情（尽管它们被书写为一种您很可能不能理解的形式）。要想获知那些禁令的真实涵义是一种费力的抗争，它们可能拥有很多页小字印刷的繁琐细则。

专利的有效期一般会持续 20 年，这对于我们的领域是一段相当长的时间。20 年前还没有万维网——海量的计算机应用进入了一个在 20 年前甚至不可能被提出的领域。当然，人们在其上做的每一件事对于 20 年前都是新奇的——至少在某些方面是新奇的。因此，如果专利被申请了，我们会被禁止做所有这些事情，并且在那些蠢到拥有这样的政策的国家，我们会被禁止做所有这些事情。

在大多数时间，当人们描述专利体系的功能时，他们会表现出对该体系的既定的兴趣。他们可能是专利律师，或者他们在专利局工作，或者他们在某家行业大鳄的专利部门工作。总之，他们想让您喜欢这种专利体系。

《经济学人》杂志曾将专利系统称为“一种耗时的撞大运的事”。如果您曾经见过关于彩票的宣传，您将理解它如何运作：它们反复强调微不足道的获胜几率，并且从不提及压倒性的失败几率。通过如此做，它们有意地、系统性地展示了关于什么事情将会发生在您身上的带有偏向性的场景，而它们事实上却又并没有针对任何特定的事实撒谎。

对于专利系统的宣传与之类似：他们描述的是您走在街上，口袋里装着一纸专利证书的时候如何如何——或者先说获得一项专利将会如何如何，然后您在口袋里装有一纸专利证书的时候如何如何，通常您可以从口袋里拿出那张专利证书，用它指着别人说：“把你的钱给我。”

为了补偿他们的偏见，我将会从另一方面进行描述——即受害者的角度——那些想要开发、发布或者运行软件的人们将会如何如何。您不得不担心有朝一日，某人可能会向您走来，拿出一张专利证书指着您，说道：“把你的钱给我。”

如果您想要在一个允许软件专利的国家开发软件，并且您想要配合专利法，您将必须去做哪些事呢?

您可以尝试列出一项关于您想要编写的程序中可能找到的各种想法和

创意的清单。除此之外，您还需要面对这样的现实：在您开始开始编写这个程序的时候，您还不知道那个清单。但是，即使您已经完成编写该程序，您也不可能列出这样一份清单。

造成这种情况的原因是……在您以某种特定的方式想出一种计划的过程中，您已经有了一种应用于您的设计的思想上的架构。正因为如此，它将阻止您去认识那些其他人可能用于理解同一程序的其他架构——因为您并不是首次想到它的；您已经在脑海中使用某种架构设计出了它。其他人在首次见到它的时候可能看到的是另一种架构，它涉及不同的想法和创意，并且对你来说看到那些其他的想法和创意是什么是困难的。但是尽管如此，它们在您的程序中实现了，因此相关的专利将会禁止您的程序，如果那些想法是受专利保护的。

例如，假设有一些关于图形创意的专利，并且您想要绘制一个正方形。当然，您可能会意识到如果有一项关于“底边”的专利，它将阻止您绘制正方形。您可以将“底边”置于您的绘画中所可能实现的所有想法的清单之中。但是，您也许不会预料到，如果某人持有一项“底角”专利，他仍然可以轻松地起诉您，由于他可以拿到您的绘画并且将其旋转 45 度。此时您的正方形看起来就有了底角。

因此，您不可能列出一项想法的清单，其中这些想法一旦受到专利保护就将禁止您的程序。

您可能会尝试去做的是试图找出所有可能在您的程序中使用的受专利保护的思想。然而，事实上您不可能做到这一点，由于专利申请在至少 18 个月内保密；其结果是专利局可以在这段时间内考虑是否批准一项专利，并且它们不会告知您。这不仅仅是学术上、理论上的可能性。

例如，一款名为 Compress 的程序于 1984 年被编写出来，这是一款使用数据压缩算法来压缩文件的软件，而当时并没有关于那种文件压缩算法的专利。该软件的作者通过某家期刊上的一篇文章学到了该算法。当时，我们认为计算机科学期刊的目的应该是发布算法以供人们使用。

此人编写并且发布了该软件，并且该算法于 1985 年获得专利。但是该专利拥有者非常精明，并不急于通告人们立即停止使用它。其专利拥有

者看到了这一点：让每个人将自己的坟墓挖得更深一些。几年之后，该专利拥有者开始威胁人们，很显然，我们不能再使用 Compress，于是我请求人们推荐我们可用于压缩文件的其他算法。

而后，有人在来信中说到："我开发出了另一种效果更佳的数据压缩算法，并且写出了一段程序，我愿意把它送给您。"于是，我们做好了发布它的准备，就在它将要被发布的一周之前，我特意阅读了《纽约时报》周刊的专利专栏，我很少去阅读它——可能每年只看几次——但幸运的是，我看到了某人已经得到了一项关于"发明一种用于数据压缩的新方法"的专利。于是我说我们最好看看这项专利。确实，它覆盖了我们马上就要发布的软件。但是事情还有可能变得更坏：这项专利可以再晚一年被批准，也许是两年、三年，或者五年。

尽管如此，还是有其他人带来了又一种更好的压缩算法，这种压缩算法被用于 gzip 软件中，而几乎每一个想要压缩文件的用户都转向了 gzip，这看起来像是一个完美的结局。但是，您随后将会知道，并非整个过程都是这样美好。

因此，您不能获知有关正在被评估的专利的任何信息，即使当它们粉墨登场的时候可能会禁止您的工作，但您可以获知已被批准的专利的信息。它们会由专利局公布，问题是您不可能全部阅读它们，由于它们实在是太多了。

在美国，我相信总共有数十万的软件专利；对它们保持跟踪是一项庞杂的工作。因此您将不得不随时查询相关专利。您将会找到大量相关专利，但您必然不可能找到它们的全部。

例如，在 20 世纪 80 到 90 年代，有一项关于在电子表格"重新计算自然顺序"的专利。曾经有人向我索取一份它的副本，于是我在自己的列出了专利号的计算机文件中进行查询，并且复印了一份发送给对方。对方收到后说："我想您发给我的不是我想要的专利。这项专利是关于编译器的。"于是我想可能是我们的文件中记录了错误的专利号。我再次查询它，并且确信它所描述的是"一种将公式编译到对象代码中的方法"。然后我开始仔细阅读它，以便确认它是否真的并非我们想找的专利。我读到了它的权

利要求，发现它确实是关于重新计算自然顺序的专利，但它并未使用这些短语。它没有使用“电子表格”这一短语。事实上，该专利禁止了数十种用于实现拓扑排序的方法——他们所能想到的所有方法。但是我并不认为它用到了“拓扑排序”这一短语。

因此，如果您正在编写一份电子表格并且试图通过搜索查询相关专利，您可能会找到大量相关专利。但是，您并不会发现这样一条专利，直到某一天您对某人说：“哦，我正在设计一份电子表格。”而对方说：“哦，你知道其他那些正在使电子表格软件受到起诉的公司吗？”然后您才会知道这一点。

您不能通过搜索找出全部相关专利，但可以找到它们中的许多。然后您必须获知它们是什么意思，这是很困难的，由于专利是用冗长并且含混不清的法律语言书写的，很难理解其真正涵义。因此您将不得不花费大量时间和钱财向律师解释您想要做什么，以便从律师那里获知您是否被允许如此做。

即使是那些专利持有人，通常也不能完全认出他们的专利到底是什么意思。例如，Paul Heckel 曾经发布了一个用于在小屏幕上显示大量数据的程序，基于此程序中的一些思想，此人获得了两项专利。

我曾经试图找出一种简单的方式以描述其中一项专利的要求 1 所覆盖的内容。然而，我发现我未能找到一种比其专利原文更简单的解释方式；而对于那条陈述，我不能试图一次性将其全部装入我的脑中，不论我多么努力地进行尝试。

然而，Heckel 也未能抓住其本质，由于当他看到 HyperCard 这一概念时，他所能发现的是这与他的程序全然不似。他并没有想到他的专利书写方式将会禁止 HyperCard；但是他的律师却有这种想法，于是他威胁苹果。然后他威胁了苹果的用户，最终，苹果与此人达成了一项不为外人所知的和解方案，因此我们不知谁是真正赢家。因此，这一例子足以说明任何人想要获知某项专利是否禁止某一事情到底有多么困难。

事实上，我之前曾进行过这篇演讲，并且 Heckel 也是听众之一。当演讲进行到这个节点之时，他跳了起来，说道：“这不是真的，我只是不知道

我自己的专利所保护的范围。”我于是说道：“这正是我所讲的。”此时他坐下了，而这正是我被 Heckel 所诘问的经历的结束。如果我当时说“不”，他很可能找到某种方式与我辩论。

尽管如此，在与律师进行过一段冗长并且昂贵的交谈之后，律师很可能会给您这样的解答：

> 如果您做了这个领域中的一些事情，您几乎肯定会输掉一场诉讼；如果您做了这个领域中的一些事情，您有相当大的几率输掉一场诉讼；如果您真的想要确保安全，您必须避免涉足这个领域。但是，任何诉讼的结果都存在相当大的变数。

现在，您已经了解从事商业活动的清晰、可预测的规则。接下来，您实际上将会如何做？为了应对任何专利可能带来的问题，您有三件事可以做。其一是回避专利；其二是获得它的使用许可；其三是使其作废。接下来，我将会逐条讨论。

其一，回避该项专利是有可能的，也就是说，不要实施那些被它禁止了的东西。当然，如果很难区分什么才是它所禁止的，可能也很难说出哪些方式足以回避它。

两年以前，柯达起诉 Sun，由于对方使用了一项与面向对象程序设计有关的专利，而 Sun 认为这并未侵犯该项专利。但最终法庭判决这构成专利侵犯；然而，其他人在看到那项专利之后，都难以得出有关该判决公正与否的哪怕是最微弱的想法。没有人能够区分什么才是那项专利所覆盖或者没有覆盖的，但是 Sun 不得不支付数亿美元的赔款，由于它侵犯了一条完全不可理解的法律。

有时，您能够区分什么才是您需要回避的；有时您需要回避的是一种算法。

例如，我曾经见过某项专利是关于快速傅里叶变换（FFT）相关的东西，但它的运行速度快两倍。当然，如果普通的 FFT 对于您的应用足够快，那么这将成为回避这项专利的简单方式。并且这将适用于大多数情况。而您一旦试图做一些一直需要运行 FFT 的事情，并且那种快速算法勉强足够

快，那么您将不能回避它，尽管您可以等上几年以得到一台更快的计算机。不过这毕竟是较为罕见的情形。对于大多数情形，那项专利还是容易回避的。

另一方面，一项关于某种算法的专利也许是不可能回避的。考虑诸如 LZW 数据压缩算法。如我之前所述，我们发现了一种更好的数据压缩算法，并且每个需要压缩文件的用户都转向了用到更好算法的 gzip 程序。其原因是，如果您只是需要压缩某些文件并且在以后进行解压，您可以告知人们使用该程序进行解压；然后您可以使用基于任何算法的任何程序，并且您只需关心该程序工作得如何。

但是 LZW 也被应用于其他事情；例如 PostScript 语言指定了用于 LZW 压缩和 LZW 解压的操作符。拥有其他更好的算法并没有意义，由于这只会产生一种新的数据格式，而它们将是不可互操作的。如果您使用 gzip 算法对其进行压缩，您将不能使用 LZW 对其解压。因此不论您的其他算法多么好，不论它是什么，它都不能允许您根据该算法的规范实施 PostScript。

但是，我注意到用户极少会要求他们的打印机去压缩什么东西。一般情况下，他们需要打印机去做的只是解压；而我也注意到，关于 LZW 算法的两项专利都是以这种方式书写，如果您的系统只能进行解压，这并未被禁止。这些专利被这样书写使得它们覆盖了数据压缩，并且它们拥有同时覆盖了压缩和解压的其他权利要求；但是并没有仅仅覆盖了解压的权利要求。于是我意识到，如果我们仅仅实施 LZW 解压，我们将会是安全的。尽管这样做并不足以满足标准规范，这已经足以取悦用户；它将会只做那些它们实际上所需要去做的事情。于是这就是我们如何勉强侥幸地回避了那两项专利的经历。

现在，有一种用于图像的 GIF 格式，它也使用了 LZW 算法用于压缩。人们并没有花费太长时间就定义出了另一种图像格式，它称为 PNG，其涵义为 PNG 不是 GIF（PNG's Not GIF）。我认为它使用了 gzip 压缩算法。并且我们开始对人们说："不要使用 GIF，它是危险的。转向使用 PNG。"而用户则会说："好吧，也许某天我们会这样做，但是浏览器还没有实施它。"并且浏览器开发者也会说："我们也许会在某一天实施它，但是现在还没

有太多的用户需求。”

当然，事情的发展状况是非常明显的——GIF 是一种事实上的标准。实际上，要求人们从他们所习惯使用的事实标准格式转向另一种不同的格式，就如同试图要求每位新西兰人改说匈牙利语。人们会说：“好吧，也许某一天我会去学着说匈牙利语，只要所有其他人都这样做。”于是我们对于要求人们停止使用 GIF 的努力从未成功过。即使其专利持有者之一已经开始将矛头对准网站操作者，并且威胁起诉他们，除非他们能证明其网站的所有 GIF 图片都是使用经过认证和授权许可的软件制作的。

于是，GIF 对于我们社区的大部分人来说都是危险的陷阱。我们认为我们有一种 GIF 格式的替代品，称为 JPEG，但是其后就有人说：“我刚刚翻阅了我的专利公文包。”——我认为这是某些刚刚买了某项专利并且立即将其用于威胁众人的人——并且他接着说：“然后我就发现其中一项专利覆盖了 JPEG 格式。”

当然，JPEG 格式并非事实上的标准，它只是一种官方标准，由一个标准委员会颁布；该委员会也有一位律师。他们的律师声称他并不认为那项专利真正覆盖了 JPEG 格式。

那么，究竟谁是正确的？这位专利持有人起诉了多家公司，并且如果有这样的一种结论，它将会说出谁是正确的。但是，我未曾听说过有这样一种定论；我不确定是否真的有定论，我认为他们达成了某种和解，而最终的解决方案几乎肯定是绝密的，这意味着它不能向我们传达关于谁对谁错的任何信息。

这些是相当轻量级的案例：关于 JPEG 的一项专利，关于 GIF 所使用的 LZW 算法的两项专利。现在，您可能想知道为何关于同一种算法会有两项专利。这本不应该发生，但却就是发生了。其原因是专利审查者事实上不可能拿出时间来研究他们也许需要仔细研究和比对的每一对概念，这是由于他们不可能拿出那么长的时间。并且由于算法实际上就是数学，您不可能定义清楚您需要比较哪些应用和专利。

您将会看到，在物理工程领域，他们可以使用将要发生的事物的物理本性来缩窄其适用范围。例如，在化学工程中，他们可以说：“输入的物质

是什么，输出的物质又是什么。”如果两项专利申请是以这种形式相互区别的，那么它们并不是指的同一过程，因此您不必为此担心。但是，同一种数学原理可以被表现为看起来非常不同的形式，并且即使您将它们放在一起研究，您仍然不能意识到它们描述的是同样的东西。并且由于这个原因，在软件领域发现同一事物被多次赋予专利是非常普通的事情。

您还记得那个在我们准备发布之前被一项专利扼杀的程序吗？那种算法也被赋予了两次专利。在一个小的领域中，我们已经看到这种情况发生在我们偶然遇到的两起案例中——同一种算法被赋予两次专利。好吧，我想我的解释已经让您明白这为什么会发生。

但是，仅仅需要面对一两项专利只是一种轻量级的案例。那么，MPEG2 这种视频格式的情况又如何呢？我曾见到过一份列出了超过 70 项覆盖它的专利的清单，想要通过某种方式的谈判获得所有相关专利的使用授权许可所花费的时间超过了该标准本身的开发过程。JPEG 委员会想要开发一种后继的标准，但他们还是放弃了。他们说这涉及的专利实在是太多了，以至于没有可能实现、

有时，受专利保护的是某种功能特性，回避该专利的唯一方式是不去实现那项特性。例如，文字处理器 Xywrite 的用户曾经收到有关功能降级的邮件，这项降级移除了一项特性。该特性允许您定义一系列缩略语。例如，如果您将 exp 定义为 experiment 的缩略语，那么当您输入 exp 空格或者 exp 逗号时，exp 将会自动变为 experiment。

然后，此功能的专利持有者威胁他们，他们得出结论，所能做的唯一事情就是移除该功能，他们于是向所有用户发送了降级。

但是，他们也联系了我，由于我编写的 Emacs 文本编辑器自 20 世纪 70 年代后期就拥有一项类似功能。这是被写在 Emacs 用户手册中的，于是他们认为我也许能够帮助他们使那项专利作废。我对于自己的一生之中能有至少一项可获专利的想法而感到满意，但我对其他人对它申请了专利并不高兴。

事实上，幸运的是，那项专利最终被判无效，这在一定程度上得益于我先于该专利将其公开这一事实。然而在当时他们确实不得不移除了那项

功能。

现在看来，移除一两项功能可能并不至于带来一场灾难。但是，如果您不得不移除 50 项功能呢？您自己可以这样做，但人们很可能会说：“这个程序不怎么样，缺少我所需要的任何功能。”因此，这可能不能成为一种解决方案。并且，有时一项专利的覆盖范围是如此宽泛，以至于它横扫了某一整个领域，例如关于公钥加密的专利，它事实上使得公钥加密在将近 10 年时间内几乎被完全禁止。

以上说的是回避专利的选项——这通常是可能的，但有时是不可能的，并且您最多能回避多少专利是有限度的。

那么，下一种可能性又当如何？即试图获得专利的使用授权许可。

首先，专利持有者也许根本不会考虑给您一份授权许可，这完全取决于他的意志。他可以说：“我就是要逼你停业。”我曾经收到某人来信求助，此人的家族企业当时正在经营博彩游戏，当然是计算机上的，他被专利持有人威胁关闭他的整个企业。他将该项专利发送给我。其权利要求 1 是关于这样的东西：“一种拥有多台计算机的网络，其中每台计算机支持多种游戏，并且允许多个游戏会话在同时进行。”

现在，我能够确信在 20 世纪 80 年代，有一所大学建立起了这样一间计算机房，内有一系列联网的工作站，每台工作站都拥有某种窗口显示功能。他们所需要做的全部事情就是去安装几种游戏，那些计算机将会可能同时显示多个游戏会话。这是多么地平淡无奇，以至于没有人会闲得发布一篇文章讲述如何实现它。没有人会有兴趣来专门为此发表一篇文章，但是，这件事就是值得为其申请专利。如果这件事发生在您身上，您就将获得对于这样平淡无奇之事的垄断权，然后您就可以用这项专利来迫使您的竞争对手停业。

但是，为何专利局会批准如此之多的在我们看来荒唐可笑而又穷极无聊的专利呢？

这并非由于专利审查者的愚蠢，而是由于他们必须遵守一种体系，这种体系有其规则，而这些规则必然引起这种结果。

您已经看到了，如果某人研究了一台机器用于做一次某事，而其他人

研制了一台能做同样一件事的机器，但它会连续做 N 次。对我们来说这就是 `for` 循环，但对于专利局而言这就是一种创新。如果有某些机器可以做事情 A，而有另一些机器可以做事情 B，然后某人发明了一种机器可以做 A 或 B，对于我们来说，这就是 `if-then-else` 语句，但对于专利局而言这就是一种创新。也就是说，他们的标准如此之低，并且他们遵循这些低标准；其结果就是那些在我们看来荒唐可笑而又穷极无聊的专利。它们是否真正合法有效我不敢确定，但是每个程序员见了它们都会呵呵大笑。

在任何情况下，我都不能给他任何所能采取措施进行自救的建议，他最终被迫关闭了他的企业。但是，大部分专利持有者将会考虑提供给您一份使用授权许可。当然，这通常极其昂贵。

但是，也有一些软件开发者觉得在大多数情况下都很容易获得专利使用授权许可。它们就是行业大鳄。在任何领域，这些行业大鳄通常都会手握该领域的半数专利。它们之间可以进行交叉授权许可，它们也可以迫使任何人同它们进行交叉授权许可，如果此人真的打算生产任何东西。其结果是它们不费吹灰之力就可以获得几乎所有专利的使用授权许可。

IBM 曾在其自家杂志，Think 杂志上刊登一篇文章——我认为那是在 1990 年的第 5 期上——文中提到当时 IBM 通过其所拥有的约 9000 项美国专利（现在已有 45000 项甚至更多）所获得的好处。他们说到其中一项好处就是他们藉此赚得盆满钵满；然而，他们所认为的比上述好处还要大出一个数量级的最主要的好处是通过自身拥有的专利获得其他公司所拥有的专利的使用权，即所谓的交叉授权许可。

这种情况意味着，由于 IBM 本身拥有如此之多的专利，它几乎可以迫使任何人给它交叉授权许可。由此 IBM 得以回避专利体系可能为任何其他人施加的几乎所有灾难。这就可以解释为何 IBM 欢迎软件专利。这也可以解释为何行业大鳄普遍欢迎软件专利。由于它们知道自己可以通过交叉授权许可独占整个行业的制高点，犹如在山顶上成立某种专属俱乐部。而我们当中的所有其他人则位于山脚下，并且没有办法企及它们所在的高度。您应该明白，如果您是一位天才，您也许会创立一家小公司并且可能获得一些专利，但您始终不可能进入 IBM 所在的行列，无论您做什么。

现在，许多公司会对它的员工说：“请你为我们获得专利，这样我们就可以用这些专利保护自己。”它们的实际意思是“试图用这些专利换取交叉授权许可”。但这并不能很好地解决问题。这也不是一种有效的策略，如果您仅仅拥有少数专利。

假设您拥有 3 项专利，其中一项指向那里，另一项指向那里，又一项指向那里。而位于此 3 处以外的某处的某人将一项专利矛头指向了您。此时您的 3 项专利不能为您带来丝毫帮助，由于它们当中没有任何一项指向此人。从另一个角度讲，该公司里的某些人迟早会注意到，这项专利实际上是针对某些人的，并且公司可以以此威胁这些人并且榨取其钱财——却从未注意到这些人并未对公司构成威胁。

于是，如果您的雇主对您说：“我们需要专利来保护自己，请你帮助我们获得专利。”我建议您如此回复：

> 领导，我信任您并且确信您只会在公司受到威胁的时候使用这些专利来保护公司。但是，我不能确定 5 年以后谁将会是这家公司的首席执行官（CEO）。据我所知，这家公司可能会被微软收购。因此我实在不能信任这家公司关于只会使用这些专利保护自己的口头承诺，除非我能够得到书面承诺。请您白纸黑字地保证我所为这家公司提供的任何专利都将只能被用于保护自己以及共同的安全，而非用于压制他人，然后我才可能带着良知去为公司获得专利。

事情将会变得非常有趣，如果您不仅仅在同您的上司私下交谈时提出这个问题；而是同时在公司的讨论列表中提出。

另一种可能发生的事情是这家公司将会破产并且其资产将被拍卖，包括专利；而这些专利的买家将会蓄意使用这些专利去做一些龌龊的事情。

理解这种交叉授权许可的实践是非常重要的，由于这种理解揭穿了软件专利倡导者的论证，他们宣称软件专利是有必要的，这可以用于保护那些穷困潦倒的天才程序员。他们将您带到了一种虚幻的场景，这里有一些不太可能发生的事情。

现在让我们着眼于这件事情。根据这种场景，有一位天才的设计者，他擅长设计任何东西，以他能够以更好的方式去实现任何事情的天赋起家，独立工作了若干年，现在时机成熟，他想要创办一家企业以量产他的产品；并且由于他的创意是如此之高明，以至于他的公司不可避免地获得成功——除了一件事情以外：行业大鳄将会与他竞争并且夺走他的全部市场份额。正由于此，他的企业几乎肯定会破产，然后他将会穷困潦倒。

现在，让我们来看看这里的所有那些不太可能发生的假设。

首先，是关于此人以独立工作的方式起家。这是一种不太可能发生的事情。在高科技领域，大部分进展是由在同一领域共同工作的人们共同取得的。但我并不想说这是不可能的，即任何一件事都不是由他独立完成的。

但尽管如此，下一个假设是此人将会创立一家企业并且该企业将会获得成功，这是不太可能的。这是因为此人是一位天才的工程师并不意味着此人在经营企业方面有任何优势。绝大多数初创公司都失败了；超过 95% 的初创公司，我想，将会在短短几年内破产。因此这很可能就是将会发生在他身上的事情，不论他做什么。

好吧，让我们做一些附加的假设：一位天才的工程师以他自己的卓越的设计起家，并且此人也精通企业运营。如果他拥有关于经营企业的独到本领，也许他的企业一时不会破产。不管怎么说，并非所有的初创企业都破产了，确实有少数取得了成功。如果此人了解商业规则，那么此人与其试图与行业大鳄正面交锋，不如试图做一些小公司更加擅长的事情，这样成功的机会更大。他也许能够成功，但我们假设不管怎样他还是失败了。如果他真的如此有天赋并且有一套经营企业的本领，我相信他不会穷困潦倒，由于有些人可能愿意给他一个职位。

这就是一系列不太可能的事情——这已经是一种不太现实的场景。但我们继续着眼于它。

由于不管走到哪里都会有这样一种说法，专利体系将会“保护”那些穷困潦倒的天才，由于他可以获得一项关于他的技术的专利。于是，当 IBM 想要同他竞争的时候，他会说：“IBM，你不能同我竞争，因为我拥有这项专利。”然后 IBM 会说：“哦，不，再也不会了！”

然而，这才是真正将会发生的事情：

IBM 将会说：“哦，真不错，你拥有一项专利。可是我们拥有这项专利，还有这项专利，还有这项专利，还有这项专利，还有这项专利，所有这些专利覆盖了你的产品中实施的其他思想。如果你认为你能够对抗我们以上这些专利，我们将会拿出更多专利。这样吧，咱们签订一份交叉授权许可协议，这样谁都不会受到伤害。”现在，由于我们已经假设我们的天才懂得商业规则，他一定会意识到自己别无选择。他将会签订交叉授权许可的城下之盟，如同 IBM 要求这个的时候每个人所做的。这意味着 IBM 将获得他的专利的使用权，也就是 IBM 可以同他展开自由竞争，如同他没有任何专利一般。这也就意味着他们所宣称的他通过拥有这项专利所理应获得的好处是不现实的，他不会得到这样的好处。

专利也许确实能够“保护”他不受来自您或者我的竞争威胁，但不能阻止来自 IBM 的竞争威胁——来自那些被此场景证实是对他的威胁的行业大鳄的竞争威胁。您已经事先知道，当那些效忠于行业大鳄的说客建议这样一套政策，据说是由于它有利于保护小公司免于来自行业大鳄的竞争威胁时，其理由当中必然存在某种瑕疵。如果它真的将会产生其所宣称的结果，他们不可能支持它。但这也解释了为什么软件专利不可能达到这样的目的。

即使是 IBM 也不能总是成功采取这样的方式，由于还存在着一些我们称之为专利流氓或者专利寄生虫的公司，它们所从事的唯一业务就是在人们真正想要做一些事情的时候，跳出来使用其所掌控的专利来榨取他们的钱财。

专利律师对我们说：在你们的领域里有专利的存在真的是一件美好的事情。但是，在他们的领域里并没有专利。那里没有关于如何书写并寄出恐吓信的专利，没有关于如何发起一桩法律诉讼的专利，没有关于如何说服法官或陪审团的专利。因此，即使是 IBM 也不能迫使专利流氓签订交叉授权许可协议。但是 IBM 算清了：“我们的竞争对手也必须向它们付钱；这只是进行商业活动的成本的一部分，我们能够承受。”IBM 和其他行业大鳄得出这样的结论：通过其所拥有的专利能够获得对于所有商业活动的

普遍统治地位，扣除付给专利流氓的保护费后仍然能够承受。这就是它们欢迎软件专利的原因。

确实也有软件开发者发现他们很难获得一项专利的使用授权许可，这些人是自由软件的开发者。其原因是通常的专利授权许可协议包含我们完全不可能接受的条款：由于通常的专利授权许可协议要求按照再分发的副本数量付费。但是由于自由软件赋予了用户再分发和复制的自由，我们没有办法统计总共存在多少副本。

如果某人向我提供了一项专利使用授权许可，要求为每份副本支付一百万分之一美元的费用，我需要支付的总金额现在也许能够装在我的口袋中，也许是 50 美元，但我不能确定到底是 50 美元，还是 49 美元，还是其他金额，由于我不可能确定人们所复制的副本数量。

还有些专利持有人不愿意按照再分发的副本数量收费；他可以向您开出提供专利使用授权许可的一次性总价，不过这样的总价通常是很高的，例如 10 万美元。

我们之所以能够开发出这么多的尊重用户自由的软件，其原因是我们可以在没有钱的条件下开发软件，但我们不能在没有钱的情况下支付一大笔专利授权许可费用。如果我们被迫花钱以换取为公众编写软件的权利，我们不可能在这方面有所建树。

以上就是关于获得专利使用授权许可的可能性。还有一种可能性是试图使专利作废。如果国家将软件专利视为大体有效的并且允许批准它们，那么唯一的问题是某项特定的专利是否符合评估标准。只有当您拥有压倒性的证据的时候，您前去法庭才有意义。

那么，到底需要的是什么样的证据呢？您必须找到证据以证明早在数年之前，即那项专利被申请之前，人们已经了解相同的思想。您必须找到今天仍然存在的东西以证明当时人们已经普遍知道这种想法。因此，骰子已经于数年前被掷出，如果掷骰子的结果在今天看来仍然对您有利，并且您能够在今天证实当时的事情，那么您将拥有可能用于尝试推翻该专利的证据，这也许能够成功。

打完这场官司也许会花费您很多钱。其结果是，一项很可能是无效的

专利仍然是可用于威胁您的可怕武器，如果您没有那么多的钱。有人就付不起钱来捍卫他们的权利——非常多的人。当然，那些能够付得起这笔钱的人除外。

以上三件事就是当任何专利禁止您的程序中的某些东西的时候，您所能采取的措施。问题的关键是，其中的任何一种方法是否可能，取决于不同的环境细节。因此在有些时候，它们都是不可能的；当这种情况发生时，您的项目已经死了。

但是，大多数国家的律师会这样对我们说："不要想着事先找到相关专利。"其理由是如果您已知某项专利，侵犯它的罚金会更高。于是他们所传达给您的无外乎是："闭上眼睛，不要试图查找专利，只要盲目地实施你的设计决定，然后去撞大运。"

当然，对于每次单一的设计决定，您可能不会触碰专利，也许您不会遇到任何麻烦。但是，您需要迈出那么多步才能走出雷区，以至于您想要毫发无伤全身而退是非常不现实的。另外显而易见的是，那些专利持有人不会一下子全都现身，于是您不可能知道到底将会遇到多少专利持有人。

电子表格中的自然顺序重新计算方法专利的持有人要求按照每份电子表格销售总额的 5% 支付费用。您可以想象为少数几项类似的专利使用授权许可付费，但是，当第 20 位专利持有人前来拜访，要求您将最后剩下的 5% 的钱用于支付专利使用授权许可费用的时候又当如何呢？而第 21 位专利持有人前来拜访的时候又当如何呢?

商务人士可能会说这种场景虽然有趣但却是荒唐的，由于在您走到那种境地之前，您的企业早就破产了。他们告诉我，只要两三项类似的专利使用授权许可费用就足以让您的企业破产，于是您不会等到第 20 位专利持有人。由于他们一个接一个地现身，您不可能知道还会来多少位。

软件专利是一团混乱，它对于软件开发者是一团糟，但除此之外，它们是对每一位计算机用户的限制，因为软件专利限制了您可以用您的计算机去做什么事情。

这与其他领域的专利是非常不同的，例如关于汽车引擎的专利，它们只限制汽车制造商，而不会限制您和我。但是，软件专利确实在限制您和

我，以及所有使用计算机的人们。因此我们不能将其仅仅作为经济概念考虑；我们不能仅仅从单纯的经济角度评估这个问题。这里有更重要的，生死攸关的事情。

但是，即使只是在经济方面，这种体系也是自相矛盾的。由于它的初衷是促进发展。据说它想要通过创造这种人为的激励机制以鼓励人们发表想法，它将会促进这个领域的进步。但是，它所产生的实际效果恰恰与之相反，因为软件开发过程中的复杂工作不是随着想法信手拈来的，它需要在一个程序中实施数千种想法。而软件专利阻碍了这一过程，因此它们从经济角度上讲也是自相矛盾的。

甚至还有经济学研究证实事实确实如此——这些研究结果显示，在一个拥有大量增量创新的领域，专利体系确实会减少研发投入。当然，它也会通过其他方式阻碍发展。因此，即使我们无视软件专利所带来的不公，甚至如果我们仅仅狭隘地从它们通常被提议的经济方面考察软件专利，它仍然是有害的。

人们有时会以这种观点作为回应：“其他领域的人们已经与专利共存了几十年，他们已经习惯了专利的存在，为什么你就应该成为例外？”

在这里，这个问题当中包含了一种荒唐的假设，这就如同说：“其他人都患了癌症，为什么你就应该幸免？”我认为无论如何，人们不患癌症才是好的，不管其他人如何。这种问题之所以荒唐，是由于它预设了这样的观点：无论如何我们都必须有义务去忍受专利对我们造成的伤害。

但是，这其中又蕴含着一个合理的问题，这个合理的问题是：“在不同的领域之间，究竟存在着什么方面的不同，以致于这种不同将会影响到专利政策在这些领域中是好是坏？”

在不同领域之间，确实存在着某种基本的重要差别，即任何一款产品的组成部分当中有可能被多少项专利禁止或覆盖。

现在，我们的脑海中可能有这样一种天真的想法，这是我正在尽力克服的，由于它不是真实的。这种想法是在任何一款产品的背后都只有一项专利，而这项专利覆盖了这款产品的全部设计理念。因此如果您设计一款新产品，它不能是已获专利的，并且您将有机会获得关于该产品的“那项

专利”。

事情并不是这样的。也许早在 19 世纪确实是这样，但现在则不是。事实上，领域之间可以像光谱那样按照每个产品对应多少项专利来划分。这种光谱的起始点是 1，但如今已经没有那样的领域；当今的领域分布在光谱上的不同位置。

最接近这种情况的领域是制药。几十年前，确实每种药物只有一项专利，在任何时候都如此，由于该专利覆盖了一种特定物质的全部化学分子式。在当时，如果您开发了一种新药，您可以确认它没有被任何其他人申请专利，并且您可以获得该药物的唯一专利。

但是现在的情况与之不同。现在有了更宽泛的专利，于是现在您可能开发一种新药，但是您不被允许生产它，由于某人已经拥有一项覆盖了它的宽泛的专利。

也许甚至会有几项专利同时覆盖了您的新药，但不会是多达数百项。其原因是，我们进行生物工程研究的能力还相对有限，没有人知道如何将这么多的思想组合在一起以生产出在医学方面有用的物质。如果您能够组合其中的两种，您的成就对于我们的知识水平已经非常了不起。但是，其他领域将会涉及将众多想法组合起来以做成一件事。

这幅光谱的另一端是软件领域。在这里，我们比任何其他人都能将更多的想法融入一项有用的设计，由于我们的领域从根本上比任何其他领域都更简单。我假设我们的领域中的人们的智力与物理工程领域的人们的智力相当。这不是说我们从根本上比他们更有能力；这只是在说我们的领域从根本上说更加简单，因为我们是用数学来工作。

程序是由众多数学的成分构成的，这些数学成分拥有某种定义，而物理对象是没有定义的。物质会按其规律发生作用，这是由于物质的本性，而您的设计不一定会按照它们 “应当” 采取的作用方式发生作用。这只是一种困难。您不能说物质有错误，而物理宇宙应当受到修复。而我们程序员可以在一条数学中的没有粗细的线上建起一座城堡，它能够屹立不倒，因为它里面的任何东西都没有重量。

在物理工程中，您必须解决众多复杂性，而我们无需为之担心。

例如，当我将一个 `if` 语句置于一个 `while` 循环中时：

- 我无需担心如果 `while` 循环以错误的频率重复，其中的 `if` 语句可能将会开始振动，它也许将会由于发生共振而断裂；
- 我无需担心如果它们的共振频率过快——您知道，大约每秒数百万次——以至于它将会生成无线电频率信号并由此导致程序中的其他部分产生错误的值；
- 我无需担心环境中的腐蚀性液体可能会渗入 `if` 语句和 `while` 语句之间的缝隙并且开始侵蚀它们直到信号再也不能被传递；
- 我无需担心 `if` 语句产生的热如何才能传导至 `while` 语句以外，以保证这不会使得 `if` 语句过热烧毁；并且
- 我无需担心我应当以何种方式移除受损的 `if` 语句，如果它的确断裂、烧毁或者被腐蚀了，并且将其更换为另一个完好的 `if` 语句以使得程序再次能够运行。

基于此原因，我无需担心在我每次为程序复制一份副本的时候应当以何种方式将 `if` 语句插入到 `while` 语句中。我无需设计一座工厂来复制我的程序，由于几个通用的命令就能用于复制任何东西。

如果我想要在光盘（CD）上制作副本，我只需刻录一片母盘；有一种程序使得我可以将其用于为任何东西制作母盘或者烧录任何我需要写入的数据。我可以制作一片母盘，刻录之后将其送至一座工厂，他们将会复制我所发送的任何东西。我无需为我想要复制的每件不同的东西设计一座不同的工厂。

而对于物理工程，您通常不得不去做这些事情：您必须基于可制造性来设计产品。设计工厂也许甚至比设计产品的任务更为艰巨，并且而后您可能还必须花费数百万美元建厂。由于以上这些困难，您将不能将如此之多的想法融入一款产品并且使之可用。

一项拥有一百万项不重复的设计元素的物理设计是浩大的，而一个拥有一百万项设计元素的程序则再普通不过。它只是数十万行代码，几个人

可以在几年之内完成，因此它不是什么大事。其结果是，专利体系对我们造成的压力相比之下更重，相对于那些在其他领域工作，被物质的本性所阻挡的人们。

一位律师曾经研究过一个特定的大型程序，也就是 Linux 内核。它与我所发起的 GNU 操作系统配合使用。这是在 5 年之前的事情；他发现共有 283 项不同的美国专利，其中每一项看起来都会禁止在 Linux 代码中的某处进行某种计算。与此同时，我看到的一篇文章称 Linux 约占整个 GNU 操作系统的 0.25%。因此，将 300 乘以 400，我们便可预计出可能禁止了整个系统中的某些东西的专利数量约为 10 万。这是一个非常粗略的估计，没有更加精确的信息了，由于试图弄清这个问题将会是一项过于庞杂的任务。

现在，这位律师并未公布相关专利的清单，由于这将威胁 Linux 内核开发者，将其置于一种一旦被起诉将面临更高罚金的境地。他并不想伤害他们；他只是想展示，关于专利困局，问题究竟有多么严重。

程序员可以立即理解这些，但是政治家通常对编程知之甚少；他们通常想象专利大体上就像版权，只是略强一些。他们想象既然软件开发者没有受到关于他们的工作的版权的威胁，于是他们也不会受到关于他们的工作的专利的威胁。他们想象既然当您编写一个程序的时候您可以获得它的版权，与之相似地，当您编写一个程序的时候也将获得它的专利。这是不正确的——那么我们怎样才能给他们一条线索，使他们明白专利真正将会造成什么后果呢？它们在像美国这样允许软件专利的国家里究竟造成了什么后果呢？

我发现将软件和交响乐进行类比是有用的。以下是为什么这是一种好的类比的理由：

一段程序或者一段交响乐都将集合诸多灵感创意。所不同的是，交响乐所汇集的是众多音乐灵感。但是，您不能只是简单地拎起一串灵感并且说：“这是我的灵感集合，你喜欢吗？”由于为了使它们有意义，您必须去实现这些灵感。您不能只是挑选若干灵感，列出清单并且说：“嗨，你到底有多么喜欢这种灵感组合？”您不能将那份灵感清单当成音乐来听。您必

须写出音符来将这些灵感一起实现。

而这项我们当中的大部分人可能完全不擅长的艰巨任务，是写出全部所需的音符以使得所有这一切悦耳动听。当然，我们中的很多人都能从音乐灵感清单中挑选一些，但我们并不知道如何创作出动听的交响乐来实现这些灵感。我们当中只有一些人拥有这样的天赋。正是这件事限制了您。我也许可能发明少数音乐创意，但我不知道怎样使用它们以产生任何效果。

假设现在是 18 世纪，欧洲各国政府决定它们想要通过创立一种音乐灵感专利体系来促进交响乐的发展，于是任何以文字形式描述的音乐灵感都可以被专利保护。

例如，将某一特定序列的音符用于主题可以被专利保护；或者某种和弦进程可以被专利保护；或者某种旋律结构可以被专利保护；或者由其本人使用某些特定乐器可以被专利保护；或者音乐行进过程中的某种重复格式可以被专利保护。总之，任何类型的音乐灵感只要能被文字描述皆可获得专利。

假设现在是在 19 世纪，您是贝多芬，并且您想要创作一首交响乐。您将会发现想要创作出一首使您不会受到专利起诉的交响乐远比创作出一首动听的交响乐更加困难。由于您必须绕过所有已存在的专利。如果您对此不满，专利持有者将会说：“哦，贝多芬，您看起来只是对我们先于您拥有这些灵感而感到嫉妒，您为何不走开，并且回去想出一些属于您自己的灵感呢？”

现在，贝多芬有了自己的灵感。他被称为一位伟大的作曲家是由于他所拥有并且实际使用的所有灵感。并且他还知道如何使用这些灵感，使得它们能够发挥作用，即将他们与众多为人们所熟知的灵感相结合。他可以将少数创新的灵感与众多古老的并且不会引起争论的灵感一同融入创作。其创作结果是一段可能引起争议的乐章，但还没有达到足以使得人们难以适应的程度。

对于我们来说，贝多芬的音乐听起来并不那么容易引起争议；我听说他的作品曾经是引起争议的，在它仍然新颖的时候。但由于他将其创新的

灵感与众多已知的灵感相结合，他才给人们以延伸其境界的机会，他们也能够这样做，这正是为什么对我们来说那些灵感其实很好的原因。但是，没有人能够，即使是贝多芬也不能，成为这样的天才，其天赋足以使他能够从零开始重新发明音乐，而完全不去使用任何已经为人们所熟知的灵感，并且创造出人们想要欣赏的东西。而且也没有人能够成为这样的天才，其天赋足以使他能够从零开始重新发明计算机，而完全不用任何已经为人们所熟知的灵感，并且创造出人们想要使用的东西。

在这个技术环境变化如此频繁的时代，您终将达到这样一种境地，20 年之前所实现的东西现在完全不足以胜任需求。20 年前根本没有万维网。当然，人们在那之前也曾用计算机做过很多事情，但他们今天想要做的事情是那些能够与万维网协同工作的事情。并且您不可能只用 20 年前为人们所知的想法去做这些事情。并且我假设技术环境将会持续发生变化，从而创造出崭新的机会使得某些人可以得到那些欺骗了整个领域的专利。

行业大鳄甚至可以自己来做这件事情。例如几年之前，微软决定设计一种用于文档的伪开放标准并且通过贿赂腐化国际标准化组织（ISO）使其被批准为一项标准，他们的确做到了。但是微软在设计它的过程中使用了一些受微软自家的专利所保护的东西。微软足够强大，使得它可以从一项专利起手，使用这种受专利保护的想法来设计一种格式或协议（不论其是否有益），这样一种设计方式使得没有任何方法可以与其兼容，除非您使用完全相同的设计思想。然后微软可以使其成为事实上的标准，不论能否得到已经腐化堕落的标准化组织的助纣为虐。仅凭其自身的地位，微软就可以迫使用户使用那种格式，这就基本上宣告微软已经扼住了全世界的咽喉。因此，我们需要向政治家说明这将会真正造成什么后果。我们需要向他们证明为什么这绝不是一件好事。

现在我听说新西兰正在考虑软件专利的原因是一家行业大鳄想要藉此被赋予某种垄断地位。以限制国家里的每一个人的自由的方式来让一家商业公司大发横财，这是完全违背治国理念的。

# 保护软件领域免受专利困扰



另请参阅我的文章 “Patent Reform Is Not Enough” http://gnu.org/philosophy/patent-reform-is-not-enough.html。

专利会对每位软件开发者构成威胁，我们长久以来一直担忧的专利之争终于打响了。软件开发者和软件用户——我们社会中的大多数人——需要保证软件不受专利困扰。

对我们构成威胁的那一类专利通常称为 “软件专利”，但这个短语是带有误导性的。这些专利并非针对任何具体的程序。与之相反，每一项专利描述了某种实践上的想法和创意，并且宣称任何人试图实现这样的思想将会被起诉。因此，将它们称为 “计算思想专利” 更清楚。

美国专利体系并不会为每一项专利贴标签，称这项专利属于软件专利而那项不属于。事实上，是由软件开发者来为我们区分那些将会对我们构成威胁的——那些覆盖了可能在软件开发过程中实现想法的专利——以及其他的专利。例如，如果一项专利所保护的想法是关于一种物理结构或者一种化学反应的，没有程序可以实现那种想法；这样的专利并不对软件领域构成威胁。但如果一项专利所保护的想法是一种计算过程，那么该专利的炮筒是对准软件开发者和用户的。

这并不意味着那些计算思想专利仅仅会禁止软件。这些想法也会被实施在硬件中——并且它们中的很多已经被实施在硬件中。每项专利通常同时覆盖关于某种想法的硬件和软件实现。

## 关于软件的特别问题

时至今日，软件领域仍然是计算思想专利会造成特殊问题的地方。在软件领域，在一个程序中实施数千种想法是很平常的。如果它们中的 10% 受到专利保护，这意味着该程序受到数百项专利的威胁。

当供职于公共专利基金会（PUBPAT）的 Dan Ravicher 于 2004 年对一款大型软件（Linux，即 GNU/Linux 操作系统的内核）进行研究时，发现了 283 项美国专利可能覆盖了该程序中实施的计算思想。同年，一家杂志预计 Linux 约占整个 GNU/Linux 操作系统的 0.25%。将 300 乘以 400，我们可以得到这样的数量级估计，即该系统作为一个整体正在受到*约 10 万项软件专利的威胁*。

如果这些专利中的半数由于 “品质不良”——即由于专利系统本身的问题——而被判无效，这并不能真正带来改观。不论是 10 万项专利还是 5 万项专利，都是同样的灾难。这是为什么不应该将我们对于软件专利的批评局限于 “专利流氓” 或 “劣质专利” 的层面的原因。当今最坏的专利侵略者是苹果，它不仅仅是通常意义上的 “专利流氓”；我不清楚苹果的专利是否属于 “优质”，但是那些专利的 “质量” 越好，其所带来的威胁就越大。

我们需要解决全部的问题，而非部分。

关于解决这一问题的常见建议包括在立法层面修改专利授予的评价尺度——例如，禁止对计算实践以及实现它们的系统进行专利保护。这种方式有两个明显的缺陷。

其一，专利律师精于将专利换一种方式表达，使其能够适合任何规则；他们将任何旨在限制专利实质的尝试化解为仅仅是形式上的要求。例如，美国的很多计算思想专利描述了这样一种系统，它包括一个算术单元、一个指令排序器、一块内存、加上用于执行特定计算过程的控制器。这是对于一台计算机运行一个程序以进行某种计算的一种古怪描述；它是被设计用于使专利申请满足那些美国专利体系在一段时间内要求满足的评估标准。

其二，美国已经有了数千项计算思想专利，更改评估标准以阻止批准更多的专利并不能帮助我们摆脱已有的那些。我们不得不等上将近 20 年，

才能等到这一问题随着那些已有专利过期而彻底解决。我们可以展望立法废除那些现存的专利，但这很可能是不符合宪法的（美国最高法院不顾一切地坚持要求国会有权以牺牲公共权利为代价扩展私人特权，而不是相反）。

## 另一种方式：限制专利的效力，而非专利的可获得性

我的建议是改变专利的效力。我们应当立法规定，在通用计算硬件上开发、发布、运行程序并不构成专利侵犯。这样有几个好处：

- 不再需要区分专利或者专利申请到底属于“软件专利”还是“非软件专利”；
- 可以对开发者和用户提供保护，不论对于现存的还是将来潜在的计算思想专利；
- 专利律师不能通过以不同的方式编写专利申请来抵抗这种法律

这种方式并不会使现存的计算思想专利完全失效，因为它们仍然可以被应用于使用特殊用途硬件的实现。这是一种好处，由于它打消了有关这一计划的合法性的争论。几年之前，美国曾经通过一项法律，保护外科医生不受专利起诉，因此即使一些外科手术流程受专利保护，外科医生仍然是安全的。这样的实践为这种解决方案提供了先例。

软件开发者和软件用户需要得到对于专利威胁的保护。这是唯一能够提供对所有人全面保护的法律解决方案。然后我们可以重新回到竞争或合作……并且无需担心某些陌生人会抹除我们的工作成果。

# 第 5 部分

# 自由软件许可证

## 许可证简介



这部分包括了最新版的 GNU 主要许可证：GNU 通用公共许可证（GNU GPL），GNU 宽通用公共许可证（LGPL）以及 GNU 自由文档许可证（FDL）。虽然都是法律文书，依然放到本书里是因为这些是自由软件理念的具体呈现形式。

GNU 操作系统的软件开发始于 1984 年。当时理查德 · 斯托曼的一部分 GNU 系统组件已经可以发布，它需要一份许可证来保护。当时很多自由软件许可证已经存在，这些给了用户修改和再分发的权限，但同时也允许代码可以用于专有版本和程序。使用这些许可证，GNU 可能无法再向所有用户带去自由，因为中间人可以将 GNU 代码放到专有软件里。

因此斯托曼设计了一款许可证来保证每个用户都有修改和再分发软件的自由。它通过一个关键条件来授予这些权限：无论谁发布软件都必须将修改和再分发的授权传递下去，同时包含源代码以完成这个目标。斯托曼

创造了 “Copyleft”（参见《什么是 Copyleft?》 (p. 222) 一文）来表述这个关键转折，即使用法律权利，来保护所有用户的自由。

GNU 的 Copyleft 许可证最初的发行是针对软件，之后拓展到了相关领域比如软件文档。这些将自由软件运动的原则放到了实践领域，也都在本书中有解释。这些许可证的每一次修订都体现了自由软件法律与实际障碍之间的斗争，并体现了自由软件理念是如何变成法律条款的。

## 原始版本的 GPL

第一版 GNU 通用公共许可证发布于 1989 年——然而早在 1985 年，斯托曼就已经在用 copyleft 许可证发布 GNU 工程中的一部分软件了。到 1989 年时，每一个已经发布的 GNU 程序都有其自己的许可证。比如 GNU CC 通用公共许可证，GDB 通用公共许可证等等，而不是统一的 GNU 通用公共许可证。这些许可证除了细微差别几乎一样：比如给用户显示许可证时提示的信息只是不同的程序不同而已，除非该程序只包含一个源文件，否则许可证都会包含该程序的名字。

到 1989 年，斯托曼已经在不同 GNU 软件包的许可证上有足够多的经验，因此一个关键问题是将这些许可证集合在一起，以便能覆盖到所有软件包。他与在帕金斯史密斯和科恩律师事务所供职的律师杰里 · 科恩（Jerry Cohen）一起，将之前所有这些许可证的理念汇集起来，并将之合并成一个许可证。这样 1989 年 2 月 1 日，GNU 通用公共许可证诞生了。

第一版的许可证力求达到两个成果：第一，所有派生软件都可以使用相同许可证，第二，获得该软件的任何人都有机会获得源代码。这就要求实现强而有力的 Copyleft 以避免这三种使程序专有化的方式：版权，最终用户许可协议以及不发布源代码。

与之前特定程序的许可证相比，GPL 第一版有很多实质性变化——GPL 是一种进化而不是革命——然而确实有实质性不同。之前，开发者若想发布一个 copyleft 的程序需要修改既有的许可证，许多人都没有这么做。GPL 发布以后，这些开发者有了直接就可以用的许可证，给了所有用户分享和修改软件的自由。这是个强而有力的工具。

## 第二版

1981 年美国最高法院对 Diamond 诉讼 Diehr 一案做出判决之后，美国专利和商标局开始为软件发放专利。软件专利既侵害了自由软件也侵害了专有软件（参见本书第四部分），斯托曼意识到需要将 GNU GPL 里的 copyleft 做大幅修改。

通过选择性地发放专利许可，专利持有人可以随心所欲地控制如何根据它们的软件分发或修改。专利持有人可以给一部分人转售程序的权限，给另一部分人在其公司开放和使用修改版的权限，而给另一部分人符合 GPL 允许的活动。为换取这些权限他们会狮子大开口。他们将权力凌驾于任何想实现这个软件专利的人，无论他们自己是否开发或分发软件。而这个权力会损害自由软件，因为专利持有的第三方可以给自由软件的用户和开发者强制增加限制。

如果一个专利持有者没有分发或修改软件，那么一个使用 GPL 这种 copyleft 许可证的软件是不能控制其行为的：他们没有索要任何许可证上的权限。但是专利持有人可以通过在软件许可证中增加限制性条款的方式停止每一个程序发行商的行为。而 GPL 第二版中增加了一个新的许可证章节（第七节），明确说如果要增加新的法律许可——比如专利许可证——与 GPL 条款冲突的话，那么这个许可证必须在软件发行时完全撤销。结果就是，任何想发布和修改软件的人都会获得一份专利许可，而条款必须保证与 GPL 条款完全兼容：这样软件的使用者必须收到相同的条款，没有额外限制，同时也意味着获得源代码。

这一章节力图保护 GPL 软件的完整性。这个许可证的基本原则是希望覆盖从卑微个体到大型企业都有相同的分享和修改软件的权利。那些不分发软件的专利持有者有选择地提出专利许可，会潜在干扰这一目标的实现，虽然将许可条款分开，但他们却认为是合适的。GPL 第二版的第七部分在力图避免这种情况。

## LGPL

在 GNU 工程早期，GPL 许可证在编程工具、实用程序和游戏方面都可以很好地被使用。然而，斯托曼发现以同样的方式发布新开发的 GNU C 库的时候会事与愿违。除了一些扩展，GNU C 库可以看成是 Unix C 库的兼容性替代，因此任何 C 程序都可以与这两个任何一个链接。而如果一个专有的 C 程序不能允许使用 GNU C 库的话，他们就只能去使用 Unix 库。如此严格却没有什么好处。

斯托曼决定通过修改一个 copyleft 许可证让步：既能保护库本身的自由，又不能限制程序使用。这个想法最开始被称为 GNU 库通用公共许可证（GNU Library General Public License），于 1991 年六月首发 2.0 版。最初的 LGPL 和 GPL 相似，只有一处例外：如果其他人只是以库的形式使用这个该库，那么程序代码的许可证可以按照作者的选择。然而，可执行程序与库结合以后需要同时发布 LGPL 许可证文本和库的源代码，并需要为用户提供一些机制，让用户修改以后的库可以更新可执行程序。

如何让开发者在使用库的时候享受 LGPLv2 的好处呢？若把计算机程序想成执行特定工作的一系列指令：程序与库一起编译或链接就相当于说，“当程序执行到这一点，其他指令可以去库里找，执行完后再回到这里”。在软件开发中，使用库是非常普遍的，这样可以减少的大量重复开发，同时也减少了错误的发生：程序员不需要重复造轮子——也可以减少在此过程中产生的问题。正是因为库的广泛创建和使用，开发者可以很容易地利用 LGPL 的额外权限。

2.0 版许可证的初衷是：在一些情况下，专有软件开发者使用 LGPL 保护库而不是专有库，这样用户就可以自由地去分享和修改库。而这并没有产生 “理想” 的结果——用户并没有完全控制程序——当然 GPL 也不能达到理想的结果。LGPL 需要保障用户在其他地方没有的自由。

许可证名字里的 “库” 导致一些自由软件开发者假设所有的库原则上都应该用此许可证，但这不是设立此许可证的初衷——当一个自由的库没有专有竞争者的时候，将其以 GPL 许可证发布会对自由软件有益。为了避免造成的误解，斯托曼将许可证改名为 “宽通用公共许可证”，并增加版本

到 2.1 表示这只是文本上的小幅修改：许可证增加了一些前言，一小段声明，允许程序通过特定的系统设备以共享库的方式调用库。宽通用公共许可证 2.1 版于 1999 年 2 月发布。

### FDL

进入新世纪以后，自由软件的发展比最初还要快；然而文档却没有受到重视。于时斯托曼认为“自由软件需要自由的文档”（参见本书《为什么自由软件需要自由的文档》 (p. 46) 一文）。

诚然软件和文档有一些相似——都是指导实践的作品——但在使用方式上却有很多重要的不同点。GPL 和 LGPL 对手册并不适用。

一段时间以来，GNU 软件包对每个手册一直在使用一个未命名的，简单的，临时性的 copyleft 许可证。因为每一个手册的许可证都不同，这样无法在不同手册之间复制文本。因此斯托曼写了 GNU 自由文档许可证，一个为软件手册和其它实践作品设计的 copyleft 许可证。

FDL 首发于 2000 年 3 月。而 copyleft 则保持一致：每个收到作品副本的人都可以修改和再发布。FDL 与软件许可证不同的地方在于实现细节：关于如何指明作品的原作者以及提供“源代码”——可编辑的文档——是不同的。

## 第三版

20 世纪 90 年代，因为自由软件的迅速普及，GPL 成为社区首选的清晰的 copyleft 许可证，并应用在了主流的自由软件项目里；然而同时，专有软件开发者想出了在不违反 GPL 的情况下，有效地否定 GPL 所保护的用户的自由的方法。此外还有其他做法，且 GPL 处理起来并不方便。为了处理这种情况，一个新版本的许可证就呼之欲出了。

2002 年左右，斯托曼和其他自由软件基金会（Free Software Foundation，以下简称 FSF）成员开始讨论如何更新 GPL，还有 LGPL。FSF 建立了公众复核流程，并与自由软件法律中心（Software Freedom Law Center）的律

师合作，尽可能在发布之前解决潜在的问题。社区的顾问委员参考了由公众提交给斯托曼的观点和争论，并决定什么需要采纳。然后他基于律师的建议和意见写了许可证的文本，关于其中的修改解释，可参见本书《为何升级到 GPLv3》 (p. 244) 一文。

第三版使用了新的术语来推动使用统一的司法解释，并修改了一些需求以适应自由软件社区新的实践需要。除了这些，这一版还引入很多新条件以加强 copyleft，从而使自由软件社区成为一个整体。比如：

- 禁用限制用户修改硬件并拒绝用户修改版的发行商（所谓 “tivoization”）；
- 允许代码加入有限的额外需求，以兼容一些流行的自由软件许可证；
- 并通过提供清晰的术语加强了对专利的要求来处理跨许可证的专利问题，这在大的专利持有公司是很常见的合约形式。

GPLv3 和 LGPLv3 都包括了解决这些问题的所有方法，并最终在 2007 年 6 月 29 日发布。这些许可证都是 copyleft 艺术的一部分，以后会有更多其他的软件许可证保护用户的自由，并最终达到本书理想中的世界大同。

# 如何为你的作品选择一份许可证



## 前言

人们经常问我们，为他们的项目向他们推荐什么许可证。我们过去已经公开地写了关于这个的文章，但信息在不同的文章、FAQ 条目和许可证注释之间分散开了。这篇文章收集所有的信息至一个单一的来源中，让人们更简单地跟着做并可以参考。

这些推荐是用于那些被设计来做实用工作的作品的。这包括了软件、文档和一些其他的东西。艺术作品、以及表达观点的作品，则不一样；GNU 项目对它们该怎么发布没有通用的立场，除了它们应该在没有非自由软件的情况下可以使用（特别是没有 DRM[1]）。然而，对于和某个程序在一起的艺术作品，你可能需要听从这些推荐。

这些推荐可以用于许可一个你创建的作品——不论是已有作品的修改版还是原创作品。它们不能解决结合已有不同许可证的材料的问题。如果你在寻求那方面的帮助，请查看 GPL FAQ[2].

在你看了我们的推荐之后，如果你想要建议，你可以写邮件到 licensing@gnu.org。注意许可证小组的回信可能需要几个星期；如果你在一个月内没有得到回复，请再写一次。

## 贡献到一个已有的项目

当你贡献到一个已有的项目时，你通常应该在和原作品相同的许可证之下发布你修改过的版本。和项目的维护者合作是件好事，而为你的修改使用不同的许可证经常会让合作变得困难。你只应在有强大理由支持的时

[1]参见我们对数字限制管理 Digital Restrictions Management 的斗争活动 DefectiveByDesign.org。

[2]参见 http://gnu.org/licenses/gpl-faq.html.

候才那样做。

一种可以合理使用不同许可证的情形是，你在非 copyleft 许可证下的作品上做出了主要修改。如果你创建的这个版本比原来的版本有用得多，那么就值得让你的作品 copyleft 化，就和我们通常推荐 copyleft 的原因一样。如果你是在这种情况下，请遵循以下用于许可一个新项目的建议。

无论是什么理由，如果你选择以不同许可证发布你的贡献，你必须确保原有的许可证允许素材可以在你选择的许可证之下使用。至少从良心上讲，请明确地表示哪部分的作品是在哪种许可证之下。

## 软件

对不同的项目我们推荐不同的许可证，这主要依据软件的目的。通常来说，我们推荐使用最 copyleft 而不影响目的的许可证。我们的文章 “什么是 copyleft?” 更详细地解释了 copyleft 的概念，以及为什么它通常是最好的许可方案。

对大多数程序，我们推荐你为你的项目使用最新版的 GPL。它强大的 copyleft 适合所有类型的软件，并对用户的自由有很多保护。

现在看看例外情形。

### 小程序

对大多数小程序，使用 copyleft 是不值得的。我们用 300 行作为基准：当一个软件包的源码比这个短，copyleft 带来的好处通常太小，不足以对抗确保许可证的复本总是伴随软件的不便。

对这些程序，我们推荐 Apache 2.0 许可证[1]。这是一个 pushover（非 copyleft）软件许可证，它有用于避免贡献者和分发者起诉专利侵权的条目。这并不会让软件避免来自专利的威胁（一个软件许可证是做不到的），但它避免了专利持有者打着自由的幌子发布软件，这种情况下专利持有者会相当于做了一次 “诱导转向”，然后要求接受者同意专利证书中的非自由

[1]此许可证全文参见 http://apache.org/licenses/LICENSE-2.0

条目。

在不严格的 pushover 许可证中，Apache 2.0 是最好的；所以如果你要用一个不严格的 pushover 许可证，不论什么原因，我们推荐用这一个。

### 库

对于库，我们分三种情形。

一些实现了自由标准的库与那些限制性标准竞争，例如 Ogg Vorbis（和 MP3 音频竞争）和 WebM（和 MPEG-4 视频竞争）。对于这些项目，代码的广泛使用对于推进自由软件事业非常重要，会比在项目代码上的 copyleft 有更多的好处。

在这些特殊的情况，我们推荐 Apache 2.0 许可证。

对所有其他的库，我们推荐某种 copyleft 许可证。如果开发者已经使用现有的以非自由或不严格的 pushover 许可证发布的库，那么我们建议使用 GNU 宽通用公共许可证（LGPL）。

在第一种情形中，库实现了一个伦理上更优秀的标准，但在这里不像第一种情形，库的广泛使用并不会达成实际好处，所以完全没有理由避免 copyleft。然而，如果你要求用库的开发者在 copyleft 下发布整个程序，他们会简单地用另一个可用的库，那样也不会推进我们的事业。较为宽松的 GPL（LGPL）是设计于填补这些情形中间地带的，允许私有软件开发者使用其保护起来的库，但考虑到库代码本身，LGPL 提供了给用户自由的弱 copyleft。

对于提供了特别设计，并且不会与现有非 copyleft 或非自由库竞争的，我们推荐使用原始的 GNU GPL。要知道原因，请阅读 “为什么你不应该为你下一个库用宽松的 GPL” http://gnu.org/licenses/why-not-lgpl.html。

### 服务器软件

如果其他人很有可能会给你在服务器上跑的软件制作改进版并且不向其他人分发他们的版本，而且你担心这将把你的版本置于一个不利的地位，

我们推荐 GNU Affero 通用公共许可证（AGPL[1]）。AGPL 的条目和 GPL 几乎相同；唯一实质的区别是它有一个额外的条件确保通过网络用这个软件的人们可以获得它的源代码。

对于用户们而言，AGPL 的要求没有解决当他们委托其计算或者数据给别人的服务器时，会产生的问题。例如，它不会制止作为软件替代品的服务（SaaSS）拒绝给予用户的自由[2]——但大多数服务器不做 SaaSS。要想知道更多关于这些的问题，请在 http://gnu.org/licenses/why-affero-gpl.html 阅读 “为什么用 Affero GPL”。

## 文档

对教程、参考手册和其他大型文档工作，我们推荐 GNU 自由文档许可证。对教育型工作，这是个很强的 copyleft 许可证，最早为软件文档而编写，并特别说明了当这些作品被分发或修改时的常见问题。

对于短的、次要的文档工作，例如参考卡片，最好使用 GNU 全面容许性许可证[3]，因为一份 GFDL 的复本难以放进一张参考卡片里面。不要用 CC-BY，因为它和 GFDL 不兼容。

对于 man 手册页面，如果页面很长，我们推荐 GFDL，而如果它很短，则推荐 GNU 全面容许性许可证。

一些文档包括了软件源代码。例如，一个编程语言的的手册可能包括让用户遵循的例子。你应当既在 FDL 之下在手册中包含它们，又在另一个适合软件的许可证下发布它们。这样做使得在其他项目中用这些代码变得简单。我们推荐你用 CC0[4] 贡献小段的代码到公有领域，并在有关软件项目使用的相同的许可证之下分发大段的代码。

[1] 此许可证全文参见 http://gnu.org/licenses/agpl.html
[2] 有关 SaaSS 可参见《服务器真正是在为谁服务?》 (p. 302) 一文
[3] 参见 http://gnu.org/licenses/license-list.html/#GNUAllPermissive
[4] 关于此许可证可参见 http://creativecommons.org/about/cc0

## 其他用于程序的数据

这段讨论你会包括在软件中的所有其他实用作品。例如图表和其他功能性或有用的图像、字体和地理数据等。你也可以在艺术中遵循这些，尽管你不这样做我们也不会批评你。

如果你正特意为一个软件项目创作这些作品，我们通常推荐你在和软件使用相同的许可证下发布你的作品。用我们推荐的许可证这样做不会有问题：GPLv3、LGPLv3、AGPLv3 和 GPLv2 都可以用于任何类型——不只是软件——只需受版权保护并对修改版有清晰首选的形式。使用与软件相同的许可证会让分发者更容易遵守规定，并可以避免任何对潜在的兼容性问题的顾虑。如果提供了一些特别的实用的好处，比如和其他自由项目更好地合作，那么使用一个不同的许可证可能是合适的。

如果你的作品不是为某个特定的软件项目而创作的，或者使用和项目相同的许可证不合适，那么我们只推荐你选择一个适合你作品的 copyleft 许可证。有一些在我们的许可证列表中列出[1]。如果没有许可证看起来合适的，创作共用署名-相同方式共享许可证（CC-BY-SA[2]）是一个可以用于很多不同种类作品的 copyleft 许可证。

[1]参见 http://gnu.org/licenses/license-list.html/#OtherLicenses
[2]关于使用此许可证可参见 http://gnu.org/licenses/license-list.html/#ccbysa

# X Window 系统的陷阱



左版（Copyleft），还是不要左版？这是自由软件社区中的主要分歧之一。左版的理念之一是我们可以以火灭火——即我们应当利用版权以保证我们的代码持续自由。GNU 通用公共许可证（GNU GPL）就是一个左版许可证的例子。

一些自由软件开发者更倾向于使用非左版进行分发。诸如 XFree86 和 BSD 这样的非左版许可证基于这样的理念，即永远不对任何人说不——即使是对那些企图用您的工作作为限制他人基础的人。非左版许可证并没有做错事，但它错失了对我们更改和再分发软件的自由提供主动保护的机会。基于此原因，我们需要左版。

多年以来，X 联盟一直是反对左版的主要力量。它通过施加道德劝说和压力来阻止自由软件开发者对其程序采用左版许可证。它通过暗示说“不”是不友善的来施加道德劝说。它利用它的“左版软件不能出现在 X 发行版中”的规则来施加压力。

X 联盟为何采取这种政策？这必然和他们关于成功的概念有关。X 联盟将成功定义为流行——这特指的是使得计算机公司采用 X Window 系统。这一定义将计算机公司置于主导地位：不论它们想要什么，X 联盟都必须设法帮助它们获得。

计算机公司通常发布私有软件。它们要求自由软件开发者捐献他们的工作用于此用途。如果他们直接如此要求，人们会呵呵大笑。但是 X 联盟在面对它们的时候可以将这种要求看做是无私的。“加入我们，将我们的工作捐赠给私有软件开发者。”他们如是说，暗示这是一种高贵的自我牺牲。“加入我们，以获取流行度。”他们如是说，暗示这甚至不是一种牺牲。

但是，自我牺牲还不是问题之所在：扔掉左版对于整个社区自由的保护，所牺牲掉的远不止您自己。那些顺应了 X 联盟要求的人们将整个社区的未来托付给了 X 联盟的一厢良愿。

这种信任是错位的。在它的最后一年，X 联盟计划限制即将发布的 X11R6.4 版本使其不再是自由软件。他们终于决定开始说不，但不仅仅是对私有软件开发者，更是对我们的社区。

这里有个讽刺。如果当 X 联盟要求您不要使用左版许可证的时候，您说“是”，您便将 X 联盟置于这样一种境地，它可以将您的程序许可并限制为它自己的版本，对于 X 的核心部分的代码也是如此。

X 联盟最终没有采取这种计划。与之相反，它解散了，并且将 X 的开发权移交给了开放小组（Open Group），后者的员工正在实施一种类似的计划。值得称道的是，当我要求他们以 GNU GPL 发布 X11R6.4 并且与他们所计划的限制性许可证共存的时候，他们表示愿意考虑这种想法（他们坚决反对墨守旧的 X11 发布条款）。但是他们还未来得及肯定或是否定这一提议，它已经由于其他原因失败了：XFree86 小组继承了 X 联盟的旧政策，并且拒绝接受任何左版软件。

就在 1998 年九月，即 X11R6.4 以非自由分发条款发布数月之后，开放小组逆转了他们的决定，并且将其改为和 X11R6.3 相同的非左版自由软件许可证重新发布。至此，开放小组终于做了一件正确的事，但远未解决其基本问题。

但是，即使 X 联盟或是开放小组本意上从未打算限制 X，其他人可能已经这么做了。非左版软件在任何方向都是脆弱的；它使得任何制造非自由软件的人处于统治地位，如果此人投入足够多的资源，利用私有代码向其中添加重要功能。任何基于技术特性而非自由选择软件的人们很容易出于短期的易用性考虑而被引诱使用非自由版本。

X 联盟和开放小组再也不能通过宣称“说‘不’是不友善的”来施加道德劝说，这将会使得它更容易决定以左版发布与 X 相关的软件。

如果您贡献于 X 核心或是诸如 X server、Xlib、Xt 等程序，确实有一种不使用左版的实践上的原因。由于 X.org 小组在为整个社区维护这些程序的过程中已经做了重大贡献，使用左版发布我们的更改所带来的好处少于分叉（fork）整体开发所带来的负面影响。于是最好与他们协同工作并且不用左版发布我们对这些程序的修改。与之相似的还有诸如 xset 和

xrdb 等工具，它们与 X 核心很接近并且不需要大规模改进。至少我们知道 X.org 小组对于将这些程序作为自由软件发布有着坚定的承诺。

然而，X 核心外围程序的问题与此不同：诸如应用程序、窗口管理器、附加的库和部件工具箱。没有理由不为其使用左版许可证，我们应当为其使用左版许可证。

如果任何人感受到了由 X 发行版的软件包含准则所施加的压力，GNU 计划将会承担宣传能够与 X 协同工作的左版软件包的责任。如果您想要以左版许可证发布什么东西，但又担心从 X 发行版中将其排除将会影响它的流行度，请向我们寻求帮助。

与此同时，如果我们不再那么看重流行度，事情会变得更好。当一位商务人士以 “更高的流行度” 来引诱您的时候，此人可能是在试图使您相信他对您的程序的使用对于它的成功至关重要。不要相信这种鬼话！如果您的程序确实出类拔萃，它总能得到众多用户；您无需为失去任何特别的用户感到绝望，并且如果您不为所动，您将会更加强大。如果您能够回复 “爱用不用——我才不会在乎所谓的流行度”。通常，那位商务人士将会改变主意并且接受该程序作为左版程序，只要您动了真格。

朋友们，自由软件开发者们，不要再重复过去的错误！如果我们不以左版发布我们的软件，我们就会将它的未来置于任凭那些拥有的资源多于良知的人们摆布的境地。只要有左版，我们就能捍卫自由，不仅为我们自己，更是为了我们的整个社区。

# 程序不得限制它们的自由运行



自由软件意味着软件被其用户所控制，而不是相反。具体的说，这意味着软件在到达用户手中时，就具有用户应该得到的四大基本自由。列表[1]中的第一条自由是自由之零：按照你的意愿运行程序来做你自己想做的事情。

一些开发者提议在软件许可证中对某些使用加以限制，来禁止程序的某些用途，但这种做法是灾难性的。这篇文章解释了为何不得限制自由之零。限制程序用途的许可条件几乎不能实现它们的目的，而且还可能破坏自由软件社区。

首先，我们要明确自由之零的含义，即软件的分发并不限制你的使用方式。但这并不能使你在法律上免责。例如，诈骗在美国法律中是犯罪行为——我认为这法律是正确且适当的。无论自由软件许可证里写了些什么，你如果使用自由程序行骗并不能保护你不被逮捕。

在诈骗是犯罪的国家，用许可证条款来打击诈骗显然是多余的累赘。但在很多国家，“安全部队”实施的酷刑可是常常被纵容的，为什么不加入一项禁止酷刑的条款呢?

禁止酷刑的条款是不会有用的，因为任何自由软件许可证的实施都是通过国家政权进行的。一个希望实施酷刑的国家将会忽略许可协议。当美国酷刑的受害者试图起诉美国政府时，法院以涉密、威胁国家安全为由驳回了案件。如果一个软件开发者试图起诉美国政府，说他的程序被用来实施酷刑，违反了其许可证，这纸诉状也会被驳回的。通常，国家对它们所做的任何恶毒的事情，总能“聪明”得找到法律上的借口。游说力量强大的企业也能这么做。

那加入一些禁止某些特定私人行为的条款又如何呢？例如，善待动物组织（PETA）提议了禁止使用软件于脊柱，造成动物痛苦的许可证。或者

[1]参见《什么是自由软件?》 (p. 1) 一文来了解自由软件的完整定义

禁止使用某程序制作或发表穆罕默德画像的条款又如何呢？又或者禁止将程序用于胚胎干细胞实验？亦或禁止利用程序未经授权复制音乐录音？

这些条款能否实施是不明确的。自由软件许可协议是基于版权法的，试图以这样的方式施加使用上的限制，延伸了版权法的允许范围，这种延伸是非常危险的。你愿意让书籍限制你使用其信息的方式吗？

那假如条款是合法可实施的，这又会有好处吗？

事实是，人们对于使用软件应当完成的活动有许多不同的道德见解。顺便一说，我认为上述四类不寻常的社会活动是合法且不应当禁止的。具体说，我支持使用软件进行动物医学实验或者加工肉类。我捍卫动物权利活动人士的人权，但我并不支持他们的观点——我也不希望善待动物组织以它们的方式限制软件的使用。

由于我不是和平主义者，我同样反对“非军用”的条款。我谴责侵略战争，但我不谴责反击。事实上，我支持说服多支军队切换到自由软件的努力，以便它们检查危害国家安全的后门和监控特性。

由于我并不反对一切商业，我反对限制商用的条款。一个只能用于娱乐、爱好和学校的系统很大程度上限制了我们通过计算机能做的事情。

我刚才已经表述了我对许多其他政治问题，和某些政治活动是否正当的看法。你自己的观点可能有所不同，而这恰恰准确表明了我的论点。如果我们接受这些带有使用限制的程序作为一个自由操作系统，例如 GNU 的一部分，人们将最终制定出一大堆不同的使用限制。到时候会存在禁止用于肉类加工的程序；仅禁止加工猪肉的程序；仅禁止加工牛肉的程序；仅允许加工“洁净”食物程序。讨厌菠菜的人可能会编写一个允许加工任何蔬菜，除了菠菜的食物；而《大力水手》的粉丝可能只允许程序用来加工菠菜。而只允许说唱音乐的音乐程序也会出现；而其他音乐程序则只允许古典音乐。

这么做的结果是产生一个你不能指望它用于任何目的的系统。你每做一件事情，你就得检查一大堆的许可证来看看系统的什么部分对你的任务各有什么限制。

用户将会怎么回应呢？我想大部分用户将转而使用私有系统。允许在

自由软件中对使用方式施加各自限制，总的来说将会驱使用户使用非自由软件。试图利用自由软件中的限制条款来阻止用户做什么，就好比使用一根又长又软又直的意大利面条推动一个物体一样，没有效果。

它甚至比“无效”更加糟糕：它同样是错误的，因为软件开发者不得在用户想做的事情上行使这种权力。想象一下出售一支笔，笔上附有你能够用它来写什么的条款吧。对通用的软件而言也是一样的道理。如果你制造了一样非常有用的物品，例如一支笔，人们将用它做各种各样的事情，包括那些最恶毒的事，例如对异见人士实施酷刑的命令。但你不应该拥有通过笔来控制他们行为的权力。这对于一个文本编辑器、编译器或者内核来说也是同理。

你所拥有的是决定软件能够用来做什么的机会——当你决定实现什么功能的时候。你可以编写一个主要被你认为行为正当的人使用的软件，你也没有任何义务去编写任何可能被别人用于你不认同的行为的功能。

结论是明确的：程序不应当限制用户利用它所完成的工作。自由之零必须完整无缺。我们需要禁止酷刑，但我们不能通过软件许可协议来做到这点。软件许可协议的正确职责是建立与保护用户的自由。

# 什么是 Copyleft?



Copyleft 是一种使一个程序或其它工作自由的通用方法，并要求该程序（或其他作品）的所有修改和衍生版本也是自由的。

使一个程序成为自由软件最简单的方法就是将它放到一个不受版权限制的公有领域。如果他们足够明智，就应该允许人们共享该程序和他们的改进成果。但是这也使得那些不愿合作的人有权利将该程序转化为专有软件。他们可以或多或少地做一些修改，并将其作为专有软件发布。而获得修改版程序的人将不能拥有原作者赋予他们的自由，因为中间人已经将其剥夺了。

在 GNU 工程中，我们的目标是赋予所有用户重新发布和修改 GNU 软件的自由。如果中间人剥夺了这种自由，即便我们可能会有很多使用者，但他们将不能享有自由。因此我们使它 “Copyleft”，来代替将 GNU 软件放到公有领域。Copyleft 指明任何人在重新发布软件时，不管有没有修改，都必须将这种自由传递到下一个副本中并改变它，以使人们更多的地复制和修改。Copyleft 保证了每一位用户都拥有自由。

Copyleft 也对其他程序员发布自由软件提供了一种激励，重要的自由程序如 GNU C++ 编译器就是因为这个原因而存在。

Copyleft 也可以帮助那些想给自由软件做改进的程序员获得自由去改进自由软件。这些程序员通常在那些为了赚更多钱，而几乎什么都干的公司或大学里工作。一个程序员可能想向社区贡献他的改进，但是他的老板可能反而想将其应用到专有软件产品中。

当我们向老板解释不以自由软件形式发布改进版本是非法时，老板常常决定与其将其舍弃不如将其作为自由软件发布。

要让一个程序 Copyleft，我们首先应指出它是有版权的；随后我们附加发布条款，这些条款有法律效力并且赋予每个人使用、修改、重新发布该程序，或任何基于该程序而派生出程序的源代码的权利。但前提是这些

发布条件没有改变。因此，代码和自由在法律上是不可分割的。

专有软件开发者利用版权来剥夺用户的自由，我们用版权来保证他们的自由。这正是为什么我们颠倒这个名字，将 “copyright”（版权）改为 “copyleft”。

“Copyleft” 是一种对程序享有版权的方式。它不意味着放弃版权；事实上，那样做就不是 copyleft 了。Copyleft 里的单词 “left” 与动词 “to leave”（离开）没什么联系——只是一种与 “right”（右边）反向的说法。

Copyleft 是一个抽象的概念，而你不能直接使用抽象的概念；你只能使用该概念的一个具体实现。在 GNU 工程中，为大多数软件所使用的具体发布规则都包含在了 GNU 通用公共许可证（GNU General Public License）中。GNU 通用公共许可证经常被简称为 GNU GPL。还有一个关于 GNU GPL 常见问题的页面 http://gnu.org/licenses/gpl-faq.html，你也可以阅读为什么 FSF 从贡献者那得到版权转让 http://gnu.org/copyleft/why-assign.html。

Copyleft 的一种替代形式，是 GNU Affero 通用公共许可证（GNU Affero General Public License，AGPL），主要应用于服务器上的程序。它可以确保公开服务器上的修改版也公开发布源代码。

Copyleft 的另一种替代形式，GNU 宽通用公共许可证（GNU Lesser General Public License，LGPL），应用于一小部分（并非全部）的 GNU 库。想了解更多关于 LGPL 的正确使用，请阅读文章《为什么我们不应该在新的开发库中使用 LGPL》http://gnu.org/philosophy/why-not-lgpl.html。

GNU 自由文档许可证（GNU FDL）是 Copyleft 的一种形式，用于在手册、教材或其它文档上以保证任何人都可以自由地复制和发布它们，不管是否对它们进行了修改，也不管是不是进行商业化使用。

相应的许可证被包含在众多手册和每个 GNU 源代码的发布中。

假如你是版权所有人，所有这些设计好的许可证能让你很容易的接受并应用于自己的作品中。做这件事你不需要修改许可证，仅仅在作品中包含一份许可证的副本，并且在源文件中添加声明，以示使用了适当的许可证。

对许多不同的程序使用相同的发布条款，使得在不同的程序间复制代码变得很容易。既然都有相同的发布条款，这就没有必要去考虑这些条款是否相容。LGPL 中有一项规定，它可以允许你把发布条款改成普通的 GPL，因此你可以将代码拷贝到另一个使用 GPL 的程序当中。第三版的 LGPL 被当作了 GPL 第三版的例外，所以自动具有兼容性。

如果你想用 GNU GPL 或 GNU LGPL 来 copyleft 你的程序，请查看许可证介绍页面的建议 http://gnu.org/copyleft/gpl-howto.html。作为建议，请注意你必须使用我们许可证的全文。每篇许可证都是不可分割的整体，因此部分复制是不允许的。

如果你用 GNU FDL 来 copyleft 你的手册，请查看 FDL 文本结尾的说明和 GFDL 说明页面 http://gnu.org/copyleft/fdl-howto.html。同样，部分复制是不允许的。

使用反转的字母 C 外套一个圆圈作为版权符号是一个法律错误。Copyleft 是基于合法的版权，所以作品应该有版权声明。版权声明要求要么使用版权符号（园圈里的字母 C），或使用单词 “Copyright”（版权所有）。

反转的字母 C 外套一个圆圈没有特别的法律意义，因此它不能成为一个版权声明。也许放在书籍封面、海报上很有趣，但请注意其代表网页时的情况。

# 为什么使用 Copyleft



当涉及到为别人保护自由的时候，躺下什么都不做是一种软弱的表现，而不是谦卑。

在 GNU 工程中，我们通常建议人们使用 Copyleft [1] 许可证，比如 GNU GPL，而不是宽松的非 Copyleft 自由软件许可证。我们不会极力反对非 Copyleft 许可证——事实上在特殊情况下我们还建议使用这些——然而这些许可证的倡导者往往极力反对 GPL。

比如一个人说，使用 BSD 许可证是“一种谦卑的举动”：“我不会对使用我代码的人索要任何权力，只要尊重我就行。”这种描述相当夸张，将一种法律诉求说成是“谦卑”，我们需要思考得更深刻一些。

谦卑是放弃你自己的利益，然而你和使用你代码的人不是唯一受到你选择这种自由软件许可证影响的人。还包括另外一些将你的代码使用在非自由程序的人，会试图拒绝给别人以自由，如果你让他这么做了，你就没能捍卫自己的自由。当涉及到为别人保护自由的时候，躺下什么都不做是一种软弱的表现，而不是谦卑。

以 BSD 或其他宽松的非 Copyleft 许可证发布你的代码，并不是错的；程序依旧是自由软件，对我们社区来讲也是一份贡献。然而这很弱，绝大多数情况下，这也不是推广用户分享和修改自由的最佳方式。

[1] 参看《什么是 Copyleft?》 (p. 222) 一文。

# Copyleft：务实的理想主义



一个人做出的每个决定都取决于这个人的价值观和目标。人们可以有很多种不同的目标和价值观；声誉、利益、爱情、生存、快乐和自由，其中有一些目标只有一些优秀的人才可能拥有。当这个目标是一种原则时，我们称之为理想主义。

一个理想主义的目标激励着我在自由软件上的工作：传播自由与协作的理念。我想鼓励自由软件的传播，取代阻碍合作的专有软件，从而使我们的社会更加美好[1]。

这就是 GNU 通用公共许可证被写成 Copyleft 的根本原因。所有代码被添加到由 GPL 许可证保护的项目必须是自由软件，即使它被放进一个单独的文件里。我使我的代码在自由软件里可用，并且在专有软件里不可用，目的是为了鼓励其他写软件的人也使它自由。我想既然专有软件开发者们利用版权来阻止我们分享，我们合作的人可以利用版权给予其他合作者一些对于他们自身的好处：他们可以使用我们的代码。

不是所有使用 GNU GPL 的人都有这样的目标。许多年以前，我的一个朋友被要求在非 copyleft 的条款下重新发布一个 copyleft 的程序。他大概是这样回应此事的：“有时我从事于自由软件，而有时我从事于专有软件——但当我从事于专有软件时，我希望得到回报。”

他很乐意与一个分享软件的社区来分享自己的成果，但似乎没有理由向那些偏离我们社区限制的商业产品给出一个发布版。他的目标不同于我的，但是他觉得 GNU GPL 对他的目标也有用。

如果你想完成某件事情，光靠理想主义是不够的，你需要选择一种实现你的目标的方法。换句话说，你需要更“务实”。GPL 许可证务实吗？让我们看看它的成果。

就拿 GNU C++ 来说吧。为什么我们会有一个自由的 C++ 编译器呢？仅仅因为 GNU GPL 规定它必须是自由的。GNU C++ 是由起源于 GNU C

[1] 参见《为什么使用 Copyleft?》 (p. 225) 一文

编译器的一个工业协会 MCC 开发的。MCC 通常尽可能使它的作品专有化。但是他们让 C++ 前端自始至终是自由软件，因为 GNU GPL 规定这是发布它的唯一方式。C++ 前端引入了很多新的文件，但是因为它们要链接到 GCC 上，所以 GPL 能约束它们。并且这样做对我们社区的益处是显而易见的。

再说说 GNU Object C。NeXT 公司最初想把这个前端变成专有的；他们打算以 .o 文件的形式发布，并让用户使用 GCC 的其余部分来链接他们，想这样能绕过 GPL 的要求。但是我们的律师说这不能躲过这些要求，那是被禁止的。所以他们使 Objective C 前端成为了自由软件。

这些例子发生在好多年前，但是 GNU GPL 一直持续带给我们更多的自由软件。

许多 GNU 库遵循 GNU 宽通用公共许可证（GNU Lesser General Public License），但不全是，Readline 就是一个使用普通 GNU GPL 许可证保护的 GNU 库，它实现了命令行编辑功能。我曾经发现一个非自由程序被设计为使用 Readline，并告知程序开发者这是不允许的。他本可以从程序中移除命令行编辑功能，但他实际所做的是让这个程序基于 GNU GPL 许可证重新发布。现在这个程序是自由软件了。

这些写代码改进 GCC（或者 Emacs、Bash、Linux 或其它遵守 GPL 的程序）的程序员们经常被公司或者大学雇佣。当程序员想把他们的改进回馈给社区，从而在下一次发布能看到他的代码时，老板可能会说，“到此为止，你的代码属于我们！我们不想分享它；我们已经决定把你的改进版本添加到专有软件产品里了。”

这时候 GNU GPL 就会出来拯救你的成果了。程序员告诉老板这样的专有软件产品是侵犯版权的，从而老板会意识到他只有两种选择：把这些新代码以自由软件的形式发布，或者什么都不做。大部分时候老板会让程序员按照他一直所想的来做，把这些代码加到下一个发布版。

GNU GPL 不是“和事佬”，有时它也会对人们想要做的一些事情说“不”。有一些用户说这是一件糟糕的事情——GPL 许可证会“拒绝”一些“需要被带进自由软件社区的专有软件开发者”。

但是我们并没有把他们拒绝在我们的社区之外；是他们选择的不加入。他们使软件专有化的决定也是在我们社区之外的决定。融入我们的社区就意味着与我们的合作；如果他们不想加入，我们就不能 “把他们带入我们的社区”。

我们能做的是提供给他们一个加入的动机。GNU GPL 许可证被设计用来从我们现有的软件中产生一个动机：“如果你想让你的软件自由化，你就可以使用这个源代码”。当然，虽然不会赢得一切，但会赢得一些时间。

专有软件的开发不会对我们的社区有所贡献，但它的开发者们经常想从我们这里得到资料。自由软件用户可以为自由软件开发者提供一些自我暗示——认可和感谢——但是当一个商人告诉你，“只要让我们把你的软件包放进我们的专有软件中，你的程序会被成千上万的人使用！”，这是非常引诱人的。这个诱惑是很有力的，但从长远来看，如果我们坚持这一点，最好离远点。

当这种诱惑和压迫通过迎合专有软件策略的自由软件组织间接到来时，它们就更加难以辨别。X 联盟（与他的继任者，Open Group）提供了一个例子：由开发专有软件的公司建立，他们努力了十年来说服程序员们不要使用 Copyleft。Open Group 曾经试图使 X11R6.4 变为非自由软件[1]时，我们当中很多人抵抗住了这个压迫，很高兴我们做到了。

在 1998 年 9 月，X11R6.4 以非自由发行条款发布的几个月之后，Open Group 推翻了自己的决定，并且用 X11R6.3 使用过的非 copyleft 的自由软件许可证重新发布。感谢你，Open Group——但是这个后来的逆转并不能改变我们已经得出的结论，他们依旧有可能加入限制。

从实际来讲，考虑更长远的目标会更加坚定你反抗这种压迫的信念。如果你把你的思想专注于自由和可以待在公司建设的社区，你会找到这样做的动力。“坚定信念，否则你会上当受骗”。

如果有愤世嫉俗的人嘲笑自由，嘲笑社区……如果 “硬鼻子的现实主义者” 说利润是唯一的理想……忽略他们就好，继续使用 Copyleft。

[1] 更多相关信息，请参见《X Window 系统的陷阱》 (p. 216) 一文

# GNU 通用公共许可证

中文翻译：bergwolf bergwolf@gmail.com，源出处：https://sites.google.com/site/bergwolf02/gplv3_zh

## 声明

This is an unofficial translation of the GNU General Public License into Chinese. It was not published by the Free Software Foundation, and does not legally state the distribution terms for software that uses the GNU GPL--only the original English text of the GNU GPL http://www.gnu.org/licenses/gpl-3.0.html does that. However, we hope that this translation will help Chinese speakers understand the GNU GPL better.

这是一份 GNU 通用公共授权的非官方中文简体翻译。它并非由自由软件基金会发布，也不是使用 GNU 通用公共授权的软件的法定发布条款——只有 GNU 通用公共授权的英文原版 http://www.gnu.org/licenses/gpl-3.0.html 具有这样的效力。然而，我们希望这份翻译能够帮助中文读者更好的理解 GNU 通用公共授权。

---



## 导言

GNU 通用公共授权是一份针对软件和其他种类作品的自由的 copyleft 授权文件。

大多数软件授权申明被设计为剥夺您共享和修改软件的自由。相反地，

GNU 通用公共授权力图保护您分享和修改自由软件地自由——以确保软件对所有使用者都是自由的。我们，自由软件基金会，对我们的大多数软件使用 GNU 通用公共授权；本授权同样适用于任何其作者以这种方式发布的软件。您也可以让您的软件使用本授权。

当我们谈论自由软件时，我们指的是行为的自由，而非价格免费。GNU 通用公共授权被设计为确保您拥有发布自由软件副本（以及为此收费，如果您希望的话）的自由，确保您能收到源代码或者在您需要时能获取源代码，确保您能修改软件或者将它的一部分用于新的自由软件，并且确保您知道您能做这些事情。

为了保护您的权利，我们需要做出要求，禁止任何人否认您的这些权利或者要求您放弃这些权利。因此，如果您发布此软件的副本或者修改它，您就需要肩负起尊重他人自由的责任。

例如，如果您发布自由软件的副本，无论以免费还是以收费的模式，您都必须把您获得的自由同样的给予副本的接收者。您必须确保他们也能收到或者得到源代码。而且您必须向他们展示这些条款，以使他们知道自己享有这样的权利。

使用 GNU 通用公共授权的开发者通过两项措施来保护您的权利：（1）声明软件的版权；（2）向您提供本授权文件以给您复制、发布并且/或者修改软件的法律许可。

为了保护软件开发者和作者，通用公共授权明确阐释自由软件没有任何担保责任。如用户和软件作者所希望的，通用公共授权要求软件被修改过的版本必须明确标示，从而避免它们的问题被错误地归咎于先前的版本。

某些设备被设计成拒绝用户安装或运行其内部软件的修改版本，尽管制造商可以安装和运行它们。这从根本上违背了通用公共授权保护用户能修改软件的自由的宗旨。此类滥用本授权的系统模式出现在了最让人无法接受的个人用户产品领域。因此，我们设计了这个版本的通用公共授权来禁止那些产品的侵权行为。如果此类问题在其他领域大量出现，我们准备好了在将来的通用公共授权版本里扩展这项规定，以保护用户的自由。

最后，每个程序都经常受到软件专利的威胁。政府不应该允许专利权限制通用计算机软件的发展和使用，但是在政府确实允许这种事情的地区，我们希望避免应用于自由软件的专利权使该软件有效私有化的危险。为了阻止这样的事情的发生，通用公共授权确保没有人能够使用专利权使得自由软件非自由化。

以下是复制，发布和修改软件的详细条款和条件。

## 条款和条件

### 0. 定义

“本授权” 指 GNU 通用公共授权第三版

“版权” 一词同样指适用于其他产品如半导体掩膜等的保护版权的法律。

“本程序” 指任何在本授权下发布的受版权保护的作品。被授权人称为“您”。“被授权人” 和 “版权接受者” 可以是个人或组织。

“修改” 作品是指从软件中拷贝或者做出全部或一丁点儿的修改，这不同于逐字逐句的复制，是需要版权许可的。修改成果被称为先前作品的“修改版本” 或者 “基于” 先前作品的软件。

“覆盖程序” 指未被修改过的本程序或者基于本程序的程序。

“传播” 程序指使用该程序做任何如果没有许可就会在适用的版权法下直接或间接侵权的事情，不包括在电脑上执行程序或者是做出您不与人共享的修改。传播包括复制，分发（无论修改与否），向公众共享，以及在某些国家的其他行为。

“发布” 作品指任何让其他组织制作或者接受副本的传播行为。仅仅通过电脑网络和一个用户交流，且没有发送程序拷贝的行为不是发布。

一个显示 “适当的法律通告” 的交互的用户接口应包括这样一个方便而显著的可视部件，它具有以下功能：（1）显示一个合适的版权通告；（2）告诉用户对本程序没有任何担保责任（除非有担保明确告知），受权人可以在本授权下发布本程序，以及如果阅读本授权协议的副本。如果该接口

显示了一个用户命令或选项列表，比如菜单，该列表中的选项需要符合上述规范。

### 1. 源代码

“源代码” 指修改程序常用的形式。“目标代码” 指程序的任何非源代码形式。

“标准接口” 有两种含义，一是由标准组织分支定义的官方标准；二是针对某种语言专门定义的众多接口中，在该类语言的开发者中广为使用的那种接口。

可执行程序的 “系统库” 不是指整个程序，而是指任何包含于主要部件但不属于该部件的部分，并且只是为了使能该部件而开发，或者为了实现某些已有公开源代码的标准接口。“主要部件” 在这里指的是执行程序的特定操作系统（如果有的话）的主要的关键部件（内核，窗口系统等），或者生成该可执行程序时使用的编译器，或者运行该程序的目标代码解释器。

目标代码中的程序 “对应的源代码” 指所有生成，安装，（对可执行程序而言）运行该目标代码和修改该程序所需要的源代码，包括控制这些行为的脚本。但是，它不包括程序需要的系统库，通用目的的工具，以及程序在完成某些功能时不经修改地使用的那些不包括在程序中的普遍可用的自由软件。例如，对应的源代码包括与程序的源文件相关的接口定义文件，以及共享库中的源代码和该程序设计需要的通过如频繁的数据交互或者这些子程序和该程序其他部分之间的控制流等方式获得的动态链接子程序。

对应的源代码不需要包含任何拥护可以从这些资源的其他部分自动再生的资源。

源代码形式的程序对应的源代码定义同上。

### 2. 基本的许可

所有在本授权协议下授予的权利都是对本程序的版权而言，并且只要所述的条件都满足了，这些授权是不能收回的。本授权明确的确认您可以不受任何限制地运行本程序的未修改版本。运行一个本授权覆盖的程序获

得的结果只有在该结果的内容构成一个覆盖程序的时候才由本授权覆盖。本授权承认您正当使用或版权法规定的其它类似行为的权利。

只要您的授权仍然有效，您可以无条件地制作，运行和传播那些您不发布的覆盖程序。只要您遵守本授权中关于发布您不具有版权的资料的条款，您可以向别人发布覆盖程序，以要求他们为您做出专门的修改或者向您提供运行这些程序的简易设备。那些为您制作或运行覆盖程序的人作为您专门的代表也必须在您的指示和控制下做到这些，请禁止他们在他们和您的关系之外制作任何您拥有版权的程序的副本。

当下述条件满足的时候，在任何其他情况下的发布都是允许的。

转授许可证授权是不允许的，第 10 节让它变的没有必要了。

### 3. 保护用户的合法权利不受反破解法侵犯

在任何实现 1996 年通过的世界知识产权组织版权条约第 11 章中所述任务的法律，或者是禁止或限制这种破解方法的类似法律下，覆盖程序都不会被认定为有效的技术手段的一部分。

当您发布一个覆盖程序时，您将放弃任何禁止技术手段破解的法律力量，甚至在本授权关于覆盖程序的条款下执行权利也能完成破解。同时，您放弃任何限制用户操作或修改该覆盖程序以执行您禁止技术手段破解的合法权利的企图。

### 4. 发布完整副本

你可以通过任何媒介发布本程序源代码的未被修改过的完整副本，只要您显著而适当地在每个副本上发布一个合适的版权通告；保持完整所有叙述本授权和任何按照第 7 节加入的非许可的条款；保持完整所有的免责申明；并随程序给所有的接受者一份本授权。

您可以为您的副本收取任何价格的费用或者免费，你也可以提供技术支持或者责任担保来收取费用。

**5. 发布修改过的源码版本**

您可以在第 4 节的条款下以源码形式发布一个基于本程序的软件，或者从本程序中制作该软件需要进行的修改，只要您同时满足所有以下条件：

- a）制作的软件必须包含明确的通告说明您修改了它，并给出相应的修改日期。
- b）制作的软件必须包含明确的通告，陈述它在本授权下发布并指出任何按照第 7 节加入的条件。这条要求修改了第 4 节的 “保持所有通知完整” 的要求。
- c）您必须把整个软件作为一个整体向任何获取副本的人按照本授权协议授权。本授权因此会和任何按照第 7 节加入的条款一起，对整个软件及其所有部分，无论是以什么形式打包的，起法律效力。本授权不允许以其他任何形式授权该软件，但如果您个别地收到这样的许可，本授权并不否定该许可。
- d）如果您制作的软件包含交互的用户接口，每个用户接口都必须显示适当的法律通告；但是，如果本程序包含没有显示适当的法律通告的交互接口，您的软件没有必要修改他们让他们显示。

如果一个覆盖程序和其他本身不是该程序的扩展的程序的联合体，这样的联合的目的不是为了在某个存储或发布媒体上生成更大的程序，且联合体程序和相应产生的版权没有用来限制程序的使用或限制单个程序赋予的联合程序的用户的合法权利的时候，这样的联合体就被称为 “聚集体”。在聚集体中包含覆盖程序并不会使本授权应用于该聚集体的其他部分。

**6. 发布非源码形式的副本**

您可以在第 4、5 节条款下以目标代码形式发布程序，只要您同时以一下的一种方式在本授权条款下发布机器可读的对应的源代码：

- a）在物理产品（包括一个物理的发布媒介）中或作为其一部分发布目标代码，并在通常用于软件交换的耐用的物理媒介中发布对应的源代码。

- b）在物理产品（包括一个物理的发布媒介）中或作为其一部分发布目标代码，并附上有效期至少 3 年且与您为该产品模型提供配件或客户服务的时间等长的书面承诺，给予每个拥有该目标代码的人（1）要么在通常用于软件交换的耐用物理媒介中，以不高于您执行这种源码的发布行为所花费的合理费用的价格，一份该产品中所有由本授权覆盖的软件的对应的源代码的拷贝；（2）要么通过网络服务器免费提供这些对应源代码的访问。

- c）单独地发布目标代码的副本，并附上一份提供对应源代码的书面承诺。这种行为只允许偶尔发生并不能盈利，且在您收到的目标代码附有第 6 节 b 规定的承诺的时候。

- d）在指定的地点（免费或收费地）提供发布的目标代码的访问并在同样的地点以不增加价格的方式提供对应源代码的同样的访问权。您不需要要求接收者在复制目标代码的时候一道复制对应的源代码。如果复制目标代码的地点是网络服务器，对应的源代码可以在另外一个支持相同复制功能的服务器上（由您或者第三方运作），只要您在目标代码旁边明确指出在哪里可以找到对应的源代码。无论什么样的服务器提供这些对应的源代码，您都有义务保证它在任何有需求的时候都可用，从而满足本条规定。

- e）用点对点传输发布目标代码，您需要告知其他的节点目标代码和对应的源代码在哪里按照第 6 节 d 的条款向大众免费提供。

目标代码中可分离的部分，其源代码作为系统库不包含在对应的源代码中，不需要包含在发布目标代码的行为中。

“用户产品” 指（1）“消费品”，即通常用于个人的、家庭的或日常目的的有形个人财产；或者（2）任何为公司设计或销售却卖给了个人的东西。

在判断一个产品是否消费品时，有疑点的案例将以有利于覆盖面的结果加以判断。对特定用户接收到的特定产品，“正常使用” 指该类产品的典型的或通常的使用，无论该用户的特殊情况，或者该用户实际使用该产品的情况，或者该产品要求的使用方式如何。一个产品是否是消费品与该产品是否具有实质的经济上的、工业的或非消费品的用处无关，除非该用处是此类产品唯一的重要使用模式。

用户产品的 “安装信息” 指从对应源码的修改版本安装和运行该用户产品中包含的覆盖程序的修改版本所需要的任何方法、过程、授权密钥或其他信息。这些信息必须足以保证修改后的目标代码不会仅仅因为被修改过而不能继续运行。

如果您在本节条款下在用户产品中，或随同，或专门为了其中的使用，发布目标代码程序，而在发布过程中用户产品的所有权和使用权都永久地或在一定时期内（无论此项发布的特点如何）传递给了接收者，在本节所述的条款下发布的对应的源代码必须包含安装信息。但是如果您或者任何第三方组织都没有保留在用户产品上安装修改过的目标代码的能力（比如程序被安装在了 ROM 上），那么这项要求不会生效。

提供安装信息的要求并没有要求为接收者修改或安装过的程序，或者修改或安装该程序的用户产品，继续提供支持服务、担保或升级。当修改本身实际上相反地影响了网络的运行，或者违反了网络通信的规则和协议时，网络访问可以被拒绝。

根据本节发布的对应源代码和提供的安装信息必须以公共的文件格式发布（并附加一个该类型文档的实现方法以源码形式向公众共享），解压缩、阅读或复制这些信息不能要求任何密码。

### 7. 附加条款

“附加许可” 是通过允许一些本授权的特例来补充本授权的条款。只要它们在使用法律下合法，对整个程序都生效的附加许可就应当被认为是本授权的内容。如果附加许可只是对本程序的一部分生效，那么该部分可以在那些许可下独立使用，但整个程序是在本授权管理下，无论附加许可如

何。

当您发布覆盖程序的副本时，您可以选择删除该副本或其部分的任何附加许可。（当您修改程序时，附加许可可能要求在某些情况下将自身删除）。您可以把附加许可放在材料上，加入到您拥有或能授予版权许可的覆盖程序中。

尽管本授权在别处有提供，对于您加入到程序中的材料，您可以（如果您由该材料的版权所有者授权的话）用以下条款补充本授权：

a. 拒绝担保责任或以与本授权第 15 和 16 小节条款不同的方式限制责任；或者

b. 要求保留特定的合理法律通告，或者该材料中或包含于适当法律通告中的该程序的作者贡献；或者

c. 禁止误传该材料的来源，或者要求该材料的修改版本以合理的方式标志为与原版本不同的版本；或者

d. 限制以宣传为目的的使用该材料作者或授权人的姓名；或者

e. 降低授权级别以在商标法下使用一些商品名称，商标或服务标记；或者

f. 要求任何发布该材料（或其修改版本）的人用对接收者的责任假设合同对授权人和材料作者进行保护，避免任何这样的假设合同直接造成授权人和作者的责任。

所有其他不许可的附加条款都被认为是第 10 节中的 “进一步的约束”。如果您收到的程序或者其部分，声称自己由本授权管理，并补充了进一步约束，那么您可以删除这些约束。如果一个授权文件包含进一步约束，但是允许再次授权或者在本授权下发布，只要这样的进一步的约束在这样的再次授权或发布中无法保留下来，您就可以在覆盖程序中加入该授权文件条款管理下的材料。

如果您依据本小节向覆盖程序添加条款，您必须在相关的源码文件中加入一个应用于那些文件的附加条款的声明或者指明在哪里可以找到这些条款的通告。

附加的条款，无论是许可的还是非许可的条款，都可以写在一个单独

的书面授权中，或者申明为例外情况；这两种方法都可以实现上述要求。

### 8. 终止授权

您只有在本授权的明确授权下才能传播或修改覆盖程序。任何其它的传播或修改覆盖程序的尝试都是非法的，并将自动终止您在本授权下获取的权利（包括依据第 11 节第三段条款授予的任何专利授权）。

然而，如果您停止违反本授权，那么您从某个特定版权所有者处获取的授权许可能够以以下方式恢复（a）您可以暂时地拥有授权，直到版权所有者明确地终止您的授权；（b）如果在您停止违反本授权后的 60 天内，版权所有者没有以某种合理的方式告知您的违背行为，那么您可以永久地获取该授权。

进一步地，如果某个版权所有者以某种合理的方式告知您违反本授权的行为，而这是您第一次收到来自该版权所有者的违反本授权的通知（对任何软件），并且在收到通知后 30 天内修正了违反行为，那么您从该版权所有者处获取的授权将永久地恢复。

当您的授权在本节条款下被终止时，那些从您那获取授权的组织只要保持不违反本授权协议，其授权就不会被终止。您只有在授权被版权所有者恢复了之后才有资格依据第 10 节的条款获取该材料的新的授权。

### 9. 获取副本不需要接受本授权

您不需要为了接收或运行本程序的副本而接受本授权协议。仅仅是因为点对点传输获取副本引起传播行为，也不要求您接受本授权协议。然而，除了本授权外，任何授权协议都不能授予您传播或修改覆盖程序的许可。因此，如果您修改或者传播了本程序的副本，那么您就默认地接受了本授权。

### 10. 下游接收者的自动授权

每次您发布覆盖程序，接收者都自动获得一份来自原授权人的依照本授权协议运行、修改和传播该程序的授权。依据本授权，您不为执行任何第三方组织的要求负责。

“实体事务” 指转移一个组织的控制权或全部资产，或者拆分组织，或者合并组织的事务。如果覆盖程序的传播是实体事务造成的，该事务中每一个接收本程序副本的组织都将获取一份其前身拥有的或者能够依据前面的条款提供的任何授权，以及从其前身获取程序对应的源代码的权利，如果前身拥有或以合理的努力能够获取这些源代码的话。

您不可以对从本授权协议获取或确认的权利的执行强加任何约束。比如，您不可以要求授权费用，版税要求或对从本授权获取的权利的执行收取任何费用。您不可以发起诉讼（包括联合诉讼和反诉）声称由于制作、使用、销售、批发或者引进本程序或其任何一部分而侵犯了任何专利权。

### 11. 专利权

“贡献者” 是在本授权下授予本程序或者本程序所基于的程序的使用权的版权所有者。这样的程序被成为贡献者的 “贡献者版本”。

一个贡献者的 “实质的专利申明” 是该贡献者所占有和控制的全部专利，无论已经获得的还是在将来获得的，那些可能受到某种方式侵犯的专利权。本授权允许制作、使用和销售其贡献者版本，但不包括那些只会由于对贡献者版本进一步的修改而受到侵犯的专利的申明。为此，“控制” 一词包括以同本授权要求一致的方式给予从属授权的权利。

每个贡献者在该贡献者的实质的专利申明下授予您非独家的，全世界的，不需要版税的专利授权，允许您制作、使用、销售、批发、进口以及运行、修改和传播其贡献者版本内容。

在以下三个自然段中，“专利授权” 指任何形式表达的不执行专利权的协议或承诺（例如使用专利权的口头许可，或者不为侵犯专利而起诉的契约）。向一个组织授予专利授权指做出这样的不向该组织提出强制执行专

利权的承诺。

如果您在自己明确知道的情况下发布基于某个专利授权的覆盖程序，而这个程序的对应的源代码并不能在本授权条款下通过网络服务器或其他有效途径免费地向公众提供访问，您必须做到：（1）使对应的源代码按照上述方法可访问；或者（2）放弃从该程序的专利授权获取任何利益；或者（3）以某种与本授权要求一致的方法使该专利授权延伸到下游的接收者。“在自己明确知道的情况下”指您明确地知道除了获取专利授权外，在某个国家您传播覆盖程序的行为，或者接收者使用覆盖程序的行为，会由于该专利授权而侵犯一个或多个在该国可确认的专利权，而这些专利权您有足够的理由相信它们是有效的。

在依照或者涉及某一次事务或安排时，如果您通过获取发布或传播覆盖程序的传输版本，并给予接收该覆盖程序的某些组织专利授权，允许他们使用，传播，修改或者发布该覆盖程序的特殊版本，那么您赋予这些组织的专利授权将自动延伸到所有该覆盖程序及基于该程序的作品的接收者。

一份专利授权是“有偏见的”，如果它没有在自身所覆盖的范围内包含，禁止行使，或者要求不执行一个或多个本授权下明确认可的权利。以下情况，您不可以发布一个覆盖程序：如果您与软件发布行业的第三方组织有协议，而该协议要求您根据该程序的发布情况向该组织付费，同时该组织在你们的协议中赋予任何从您那里获得覆盖软件的组织一份有偏见的专利授权，要么（a）连同您所发布的副本（或者从这些副本制作的副本）；要么（b）主要为了并连同某个的产品或者包含该覆盖程序的联合体。如果您签署该协议或获得该专利授权的日期早于 2007 年 3 月 28 日，那么您不受本条款约束。

本授权的任何部分不会被解释为拒绝或者限制任何暗含的授权或其他在适用专利权法下保护您的专利不受侵犯的措施。

### 12. 不要放弃别人的自由

如果您遇到了与本授权向矛盾的情况（无论是法庭判决，合同或者其他情况），它们不能使您免去本授权的要求。如果您不能同时按照本授权中的义务和其他相关义务来发布覆盖程序，那么您将不能发布它们。比如，如果您接受了要求您向从您这里或许本程序的人收取版税的条款，您唯一能够同时满足本授权和那些条款的方法是完全不要发布本程序。

### 13. 和 GNU Affero 通用公共授权一起使用

尽管本协议有其他防备条款，您有权把任何覆盖程序和基于第三版 GNU Affero 通用公共授权的程序链接起来，并且发布该联合程序。本授权的条款仍然对您的覆盖程序有效，但是 GNU Affero 通用公共授权第 13 节关于通过网络交互的要求会对整个联合体有效。

### 14. 本授权的修订版

自由软件基金会有时候可能会发布 GNU 通用软件授权的修订版本和/或新版本。这样的新版本将会和现行版本保持精神上的一致性，但是可能会在细节上有所不同，以处理新的问题和情况。

每个版本都有一个单独的版本号。如果本程序指出了应用于本程序的一个特定的 GNU 通用公共授权版本号 “以及后续版本”，您将拥有选择该版本或任何由自由软件基金会发布的后续版本中的条款和条件的权利。如果本程序没有指定特定的 GNU 通用公共授权版本号，那么您可以选择任何自由软件基金会已发布的版本。

如果本程序指出某个代理可以决定将来的 GNU 通用公共授权是否可以应用于本程序，那么该代理的接受任何版本的公开称述都是您选择该版本应用于本程序的永久认可。

后续的授权版本可能会赋予您额外的或者不同的许可。但是，您对后续版本的选择不会对任何作者和版权所有者强加任何义务。

### 15. 免责申明

在适用法律许可下，本授权不对本程序承担任何担保责任。除非是书面申明，否则版权所有者和/或提供本程序的第三方组织，“照旧” 不承担任何形式的担保责任，无论是承诺的还是暗含的，包括但不限于就适售性和为某个特殊目的的适用性的默认担保责任。有关本程序质量与效能的全部风险均由您承担。如本程序被证明有瑕疵，您应承担所有必要的服务、修复或更正的费用。

### 16. 责任范围

除非受适用法律要求或者书面同意，任何版权所有者，或任何依前述方式修改和/或发布本程序者，对于您因为使用或不能使用本程序所造成的一般性、特殊性、意外性或间接性损失，不负任何责任（包括但不限于，资料损失，资料执行不精确，或应由您或第三人承担的损失，或本程序无法与其他程序运作等），即便该版权所有者或其他组织已经被告知程序有此类损失的可能性也是如此。

### 17. 第 15 和 16 节的解释

如果上述免责申明和责任范围不能按照地方法律条款获得法律效力，复审法庭应该采用最接近于完全放弃关于本程序的民事责任的法律，除非随同本程序的责任担保或责任假设合同是收费的。

## 条款和条件结束

## 如何在您的新程序中应用这些条款?

如果您开发了一个新程序，并且希望能够让它尽可能地被大众使用，达成此目的的最好方式就是让它成为自由软件。任何人都能够依据这些条款对该软件再次发布和修改。

为了做到这一点，请将以下声明附加到程序上。最安全的作法，是将

声明放在每份源码文件的起始处，以有效传达无担保责任的讯息；且每份文件至少应有「版权」列以及本份声明全文位置的提示。

> 版权所有 (C) 本程序为自由软件；您可依据自由软件基金会所发表的 GNU 通用 公共授权条款，对本程序再次发布和/或修改；无论您依据的 是本授权的第三版，或（您可选的）任一日后发行的版本。本程序是基于使用目的而加以发布，然而不负任何担保责任；亦无对适售性或特定目的适用性所为的默示性担保。详情请 参照 GNU 通用公共授权。您应已收到附随于本程序的 GNU 通用公共授权的副本；如果没有，请参照 http://www.gnu.org/licenses/.

同时附上如何以电子及书面信件与您联系的资料。

如果程序进行终端交互方式运作，请在交互式模式开始时，输出以下提示：

> 版权所有 (C) 本程序不负任何担保责任，欲知详情请键入'show w'。这是一个自由软件，欢迎您在特定条件下再发布本程序；欲知详情请键入'show c'。

所假设的指令'show w' 与'show c' 应显示通用公共授权的相对应条款。当然，您可以使用'show w' 与'show c' 以外的指令名称；对于图形用户界面，您可以用 "关于" 项代实现此功能。

如有需要，您还应该取得您的雇主（若您的工作为程序设计师）或学校就本程序所签署的 "版权放弃承诺书"。欲知这方面的详情，以及如何应用和遵守 GNU 通用公共授权，请参考 http://www.gnu.org/licenses/。

GNU 通用公共授权并不允许您将本程序合并到私有的程序中。若您的程序是一个子程序库，您可能认为允许私有的应用程序链接该库会更有用。如果这是您所想做的，请使用 GNU 松弛通用公共授权代替本授权。但这样做之前，请阅读 http://www.gnu.org/philosophy/why-not-lgpl.html。

# 为何升级到 GPLv3



GNU 通用公共许可证（GNU GPL）第三版已经发布，使得自由的软件包可以从 GPL 第二版升级到第三版。本文章阐述了升级许可证的必要性。

首先要说明重要的一点，是否升级许可证完全是一个选择上的考虑。GPL 第二版依然会是一份有效的许可证，就算一些程序依然留守 GPLv2 而另一些升级到 GPLv3，也完全不是什么灾难。这两份协议固然是不兼容的，但这并不是根本性的问题。

当我们说 GPLv2 和 GPLv3 不兼容时，我们的意思是指没有合法的方法将 GPLv2 发布的代码与 GPLv3 发布的代码合并到同一个程序中。这是由于两份许可证都是 copyleft 许可证，都各自要求“如果你将以本协议发布的代码合并到一个更大的程序中，那么这个程序也必须遵守该许可。”因此使它们兼容是不可能的。我们固然可以在 GPLv3 许可证中加入一项兼容 GPLv2 的豁免条款，但这是没有用的，因为 GPLv2 也需要一项类似条款。

幸运的是，许可证兼容性问题仅仅在你希望链接、合并或组合两个不同程序的代码到一起时才是问题。在操作系统中 GPLv2 与 GPLv3 的程序共存是没有任何问题的。例如，TEX 许可证与 Apache 许可证都和 GPLv2 不兼容，但是这并不能阻止我们把它们与 Linux、Bash 和 GCC 共同运行在系统中。这是因为它们都是独立的程序，同理，如果 Bash 和 GCC 升级到了 GPLv3 但 Linux 依然使用 GPLv2，也不会有任何冲突。

将程序保留 GPLv2 的许可不会创造新的问题。迁移到 GPLv3 的理由是，GPLv3 解决了目前现有的问题。

GPLv3 能够阻止的一大危险就是“机顶盒化”（Tivoization，译者注：Tivo 是美国一大有线电视机顶盒产品），机顶盒化的意思是，某些“电器”（带有计算机在内）包含了以 GPL 许可证发布的软件，但你实际上却不可能修改它们，因为一旦设备发现了任何软件修改，就会自动关机。通常，

厂商进行“机顶盒化”行为的动机是：厂商知道其软件可以被人们修改，因此力图避免这样的修改。这些设备厂商享受了自由软件带来的自由，却不允许你也这样做。

有些人认为，自由市场中不同电器的自由竞争能够使这种恶心的特性尽可能的少。单靠竞争大概也确实能够避免像“每周二下午 1 点到 5 点强制关机”这样的反特性出现，但即使这样，主人的选择依然不是自由的。自由意味着你自己控制你软件的所作所为，而不仅仅是你能祈求或者威胁那个替你做出决定的人。

在那些被数字限制管理（DRM）——一个用来限制你使用自己电脑中数据的恶心功能——所控制的关键领域，市场竞争毫无作用，因为相关的竞争是被禁止的：《千年数字版权法案》（DMCA）和类似的法律下，在美国和许多国家，发布 DVD 播放器是非法的，除非播放器遵守由 DVD 利益团体制定的规则去限制用户（它们的网站是 http://www.dvdcca.org/，但那些规则却似乎没有刊登其上）。公众们没有办法购买没有 DRM 的播放器将 DRM 拒之门外，因为市场上根本就没有啊。无论有多少产品可供你选择，它们都带有相同的数字手铐。

GPLv3 则确保你具有去除这些手铐的自由。它本身并没有禁止 DRM 或者任何类似的特性。它没有对你可以加入到程序中的实质性功能做任何的限制，同样也没有限制你删除这些功能。也就是说，发行商有权利加入这些恶心的功能，你也有权利去除这些功能。而“机顶盒化”则是厂商拒绝你自由的方式，而为了保护你的自由，GPLv3 禁止了“机顶盒化”。

对“机顶盒化”的取缔包括了消费者所能购买到的任何产品，即使是不常见的产品。GPLv3 仅仅在产品几乎完全被企业和机构独占时才对“机顶盒化”网开一面。

GPLv3 所能抵御的另一个威胁就是像 Novell 和 Microsoft 这样的专利交易。微软希望通过它的数千项专利，被迫用户为获得运行 GNU/Linux 的特权而交费，而签订了这样一份协议来实现这个目标。这笔交易给 Novell 的消费者在微软的专利面前提供了相当有限的保护。

然而微软在与 Novell 的交易中犯了一些错误，而 GPLv3 旨在利用它

们反击微软，排除这种对整个社区相当有限的专利保护。为了得益于这个保护措施，程序需要使用 GPLv3。

微软的律师们当然也不是傻子，他们下次的时候肯定会设法避免错误。因此，GPLv3 中也明确表示，他们没有“下次”了。将程序以 GPL 第三版发布，将保护发行商免于微软向最终用户的程序征收专利费。

GPLv3 同样提供了明确的专利保护，让用户免受程序的贡献者与分发者的侵扰。在 GPLv2 中仅仅规定用户可以从公司获得一份程序拷贝，或者分发程序拷贝而免于起诉，专利保护仅仅是隐含在其中的。

然而，GPLv3 中的专利保护并没有我们希望的那么强力。在理想的状态下，我们当然希望任何分发 GPL 许可软件的人，和不使用 GPL 许可证份发软件的人，都放弃他们全部的软件专利，因为软件专利根本就不应该存在。软件专利的体制是恶毒的、荒谬的，将所有软件开发者都置于被一个闻所未闻的公司起诉的危险之下，更别说那些大型集团了。大型程序往往包含了成百上千的思想，因此这些思想被上百个专利所涵盖就不足为奇了。大型集团公司收集这些成千上万的专利，然后利用它们欺压中小开发者。专利已经对自由软件的发展形成了阻碍。

使软件开发变得安全的唯一方式就是废除软件专利，而我们希望在未来的某一天实现。然而，我们不能够通过一份软件许可证就能做到这一点。任何程序，无论自由与否，都能够被八杆子打不着的一个人手中的专利杀死，而这是程序的许可协议本身所不能避免的。唯一能够使软件开发免于专利威胁的，只有法院的一纸判决，或者专利法律的修订。如果我们试图靠 GPLv3 来做到这点，这必然是会失败的。

因此，GPLv3 力求限制与“疏导”危险，更具体的说，就是我们试图将自由软件从比死亡更加悲惨的命运中拯救，避免自由软件通过专利都变为了事实上的私有软件。GPLv3 中明确的专利协议确保了使用 GPL 的公司能给予用户四大自由，而不能通过专利来改变这点，对用户说“自由不属于你”。这同时也避免了勾结其他专利持有人做这种事情的可能性。

GPLv3 的更多优点包括了更好的国际化、更温和的权利终止、对 BitTorrent 的支持，以及对 Apache 许可证的兼容。所有的这一切都是升级

的众多理由。

变化，在 GPLv3 发布之后显然是不会停止的。如果产生了对用户的危险，我们将研发 GPL 第四版。因此，假如我们写好了 GPLv4，确保程序能顺利升级显然就很重要了。

做到这一切的方式之一就是将程序以 “GPL 第三版或任何更新的版本” 发布；另一个方法是让项目所有的贡献者指定一位可以解决是否升级 GPL 版本的代理人；而第三种方式则是让所有的贡献者将各自的版权统一移交给一位版权持有人，这位版权持有人将负责升级协议版本。无论采用何种方式，程序应当能灵活升级到未来的 GPL 版本。

# GNU 宽通用许可证

中文翻译 Leo leohca@yahoo.com 源网页：http://www.thebigfly.com/gnu/lgpl/lgpl-v3.php 修改：IFRFSX 1079092922@qq.com

---

这是一份 GNU 宽通用许可证非正式的中文翻译。它并非由自由软件基金会所发表，亦非使用 GNU 宽通用许可证的文件的法定发布条款——只有 GNU 宽通用许可证英文原文 http://www.gnu.org/licenses/lgpl.html 的版本始具有此等效力。然而，我们希望这份翻译能帮助中文的使用者更了解 GNU 宽通用许可证。

This is an unofficial translation of the LGPL Free Documentation License into Chinese. It was not published by the Free Software Foundation, and does not legally state the distribution terms for documents that uses the GNU LGPL--only the original English text of the GNU LGPL http://www.gnu.org/licenses/lgpl.html does that. However, we hope that this translation will help Chinese speakers understand the GNU LGPL better

---

第 3 版，2007 年 6 月 29 日



本版本的 GNU 较宽松公共许可证包含了 GNU 通用公共授权第三版里的项目和条款，以下是复制，发布和修改软件的详细条款和条件。

## 0. 附加定义

“本许可证” 指 GNU 较宽松公共许可证第三版，“GNU GPL” 是指通用公共授权第三版。

“库” 指在本许可证管理下的涉及的作品，有别于下列定义的应用软件或组合软件。

“应用程序” 指任何通过 “库” 提供用于接口使用的软件，但非基于 “库”。用 “库” 定义一个集的子集被认为是 “库” 提供的使用接口的一种模式。

“组合软件” 是 “库” 和 “应用软件” 结合或连接而产生的一种软件。这种通过 “组合软件” 产生的特殊版本的 “库” 也被称之为 “连接版本”。

“组合软件” 里 “最小对应源代码” 是指 “组合软件” 基于 “应用程序”，而非 “连接版本” 里的 “对应源代码”，不包括被认为是孤立 “组合软件” 部份里的任何原代码。

“组合软件” 里 “对应应用程序代码” 是指 “应用程序” 的目标编码和/或原代码，包括 “应用程序” 里重新编制 “组合软件” 所需的数据和有效程序，但不包括 “组合软件” 里的系统数据库。

## 1. 通用公共授权第三版之例外

你可以发布本许可证第三和四部份下的涉及的作品，不受 GNU GPL 第三部份的限制。

## 2. 发布修订版本

如果你修改了 “库” 的副本，并且在你的修改版本中，一个组件引用了使用此组件的 “应用程序” 提供的函数或数据（有别于作为组件被调用时传递的参数），那么你可以发布修改版的副本：

a) 在此许可下，假如你尽力确保在 “应用程序” 不能提供这些函数和数据的情况下，该组件仍能工作，并执行其目标中仍有意义的任何部份，否则

b) 在通用公共授权下，任何本授权中的额外许可都不适用于该副本。

## 3. 合并库标头文件里的资料为目标代码

以目标代码形式的程序可以从库里标头文件的资料合并而成是库的一部分。你可以按照你选择的条件上传这样的目标代码，只要组合资料对数字参数，数据结构设计，存取器或小的宏，排列功能和模版不受限定（长度小于 10 排），你可以做下列两项情况：

a) 目标代码的每一个副本要给出明确通告库的使用要在本许可证下发布。

b) 目标代码要与 GNU 公用授权副本和本许可证文件一起使用。

## 4. 组合软件

你可以根据你的选择来传送一个合并后的作品，并且有效地发布但不限制修改库（包含在组合软件里的库）的一部分，还可逆向调试这种修改，只要你满足以下的每一个条件：

a) 目标代码的每一个副本要给出明确通告库的使用要在本许可证下发布。

b) 目标代码要与 GNU 公用授权副本和本许可证文件一起使用。

c) 组合软件在完成期间要显示版权通告，在这些通告中要包括库的版权通告，同时也要指出给使用者 GNU 公用授权的副本和本许可证文件作为参考。

d) 做下列之一：

0) 在本许可证下发布最小对应源代码，对应应用程序代码的形成，在此条件下允许用户用修改版重新组合或联接应用程序产生修改的组合程序，用此方式用以说明通过通用公共授权第六部分发布对应源代码。

1) 使用适当的共享库机构联接库。适当机制是指 (a) 在运行时使用已经存在于用户电脑系统的库的副本。(b) 库的修改版可以正常工作并且与连接版本兼容。

e) 提供安装信息，但只有如果你，否则必须根据通用共公授权第六部份提供这些资料，范围仅限于这些资料是必要的安装和执行修改后的版本合并的工作所产生的重组或再重新联接应用程序与联系版本的修改版本。（如果选择 4d0，安装信息必须与最小通讯资源和通讯应用软件代码一起使用。如果选择 4d1，你必须在指定的方式下提供安装信息，由通用公共授权第六部份传输通讯资源。）

## 5. 组合的库

您可以将基于该库的作品的库组件与其他非应用程序且不在本许可范围内的库组件并排放置在一个单独的库中，并根据您的选择，发布这样组合的库，只要做到以下全部两项：

a) 将组合的库与基于该（按本协议发布的）库的同一作品的未与任何其他库组件组合的副本一并发布。

b) 对组合库给出醒目的通知，说明其部分是基于该库的作品，并解释在哪里找到一并发布的同一作品的未组合形式。

## 6.GNU 宽通用许可证的修改版

自由软件基金会有时候可能会发布 GNU 较宽松公共许可证的修订版本和/或新版本。这样的新版本将会和现行版本保持精神上的一致性，但是可能会在细节上有所不同，以处理新的问题和情况。

每个版本都有一个单独的版本号。如果您收到的库指出了应用于本程序的一个特定的 GNU 较宽松公共许可证版本号 “以及后续版本”，您将拥有选择该版本或任何由自由软件基金会发布的后续版本中的条款和条件的权利。如果您收的库没有指定特定的 GNU 较宽松公共许可证版本号，那么您可以选择任何自由软件基金会已发布的版本。

如果您收到的库指出某个代理可以决定将来的 GNU 较宽松公共许可证是否可以应用于本库，那么该代理的接受任何版本的公开称述都是您选择该版本应用于本库的永久认可。

# GNU 自由文档许可证

此版本由 Leo-Hong (leohca (at) yahoo.com) 翻译整理。源网页：http://www.thebigfly.com/gnu/FDLv1.3/。修改者：IFRFSX 1079092922@qq.com

---

这是一份 GNU 自由文档许可证非正式的中文翻译。它并非由自由软件基金会所发表，亦非使用 GNU 自由文档许可证的文件的法定发布条款——只有 GNU 自由文档许可证英文原文 http://www.gnu.org/licenses/fdl.html的版本始具有此等效力。然而，我们希望这份翻译能帮助中文的使用者更了解 GNU 自由文档许可证。

This is an unofficial translation of the GNU Free Documentation License into Chinese. It was not published by the Free Software Foundation, and does not legally state the distribution terms for documents that uses the GNU FDL--only the original English text of the GNU FDL http://www.gnu.org/licenses/fdl.html does that. However, we hope that this translation will help Chinese speakers understand the GNU FDL better.

---

第 1.3 版，2008 年 11 月



## 0. 导言

本许可证的目的在于作为一种手册、教科书或其它的具有功能性的有用文件获得自由：确保每一个人都具有复制和重新发布它的有效自由，而

不论是否作出修改，也不论其是否具有商业行为。其次，本许可证保存了作者以及出版者由于他们的工作而 得到 名誉的方式，同时也不被认为应该对其他人所作出的修改而担负责任。

本许可证是一种 “copyleft”，这表示文件的衍生作品本身必须具有相同的自由涵义。它补足了 GNU 公共通用许可证——一种为了自由软件而设计的 “copyleft” 许可证。

我们设计了本许可证是为了将它使用到自由软件的使用手册上，因为自由软件需要自由的文档：一种自由的程序应该提供与此软件具有相同的自由的使用手册。但是本许可证并不被限制在软件使用手册的应用上；它可以被用于任何以文字作基础的作品，而不论其主题内容，或者它是否是一个被出版的印刷书籍。我们建议本许可证主要应用在以使用说明或提供参考作为目的的作品上。

## 1. 效力与定义

本许可证的效力在于任何媒体中的任何的使用手册或其它作品，只要其中包含由版权所有人所指定的声明，说明它可以在本许可证的条款下被发布。这样的一份声明提供了全球范围内的，免版税的和没有期限的许可，在此所陈述条件下使用那个作品。以下所称的文件，指的是任何像这样的使用手册或作品。公众中的任何成员都是被许可人，并且称作为你。如果你以一种需要在版权法下取得允许的方式进行复制、修改或发布作品，你就接受了这项许可。

“修改版本” 指的是任何包含文件或是它的其中一部份，不论是逐字的复制或是经过修正，或翻译成其它语言的任何作品。

“次要章节” 是一个具名的附录，或是文件的本文之前内容的章节，专门用来处理文件的出版者或作者，与文件整体主题（或其它相关内容）的关系，并且不包含任何可以直接落入那个整体主题的内容。（因此，如果文件的部分内容是作为数学教科书，那么其次要章节就可以不用来解释任何数学。）它的关系可以是与主题相关的历史连接，或是与其相关的法律、商业、哲学、伦理道德或政治立场。

“不变章节” 是标题已被指定的某些次要章节，在一个声明了是以本许可证加以发行的文件中，依此作为不变章节。如果一个章节并不符合上述有关于次要的定义时，则它并不允许被指定为不变。文件可以不包含不变章节。如果文件并没有指出任何不变章节，那么就是没有。

“封面文字” 是某些被加以列出的简短文字段落，在一个声明了是以本许可证加以发行的文件中，依此作为前封面文字或后封面文字。前封面文字最多可以包含 5 个单词，后封面文字最多可以包含 25 个单词。

文件的 “透明” 副本指的是一份机器可读的副本，它以一种一般公众可以取得其规格说明的格式来表现，适合于直接用一般文字编辑器、一般点阵图像程序用于由图元像素构成的影像或一些可以广泛取得的绘图程序用于由向量绘制的图形直接地进行修订；并且适合于输入到文字格式化程式，或是可以自动地转换到适合于输入到文字格式化程序的各种格式。一份以透明以外的档案格式所构成的副本，其标记或缺少标记，若是被安排成用来挫折或是打消读者进行其后续的修改，则此副本并非透明。一种影像格式，如果仅仅是用来充斥文本的资料量时，就不是透明的。一个不是透明的副本被称为混浊。

透明副本适合格式的例子包括有：没有标记的纯 ASCII、Texinfo 输入格式、LaTeX 输入格式、使用可以公开取得其 DTD 的 SGML 或 XML、合乎标准的简单 HTML、PostScript 或 PDF。透明影像格式的例子有 PNG、XCF 和 JPG。混浊格式包括只能够以私人文书处理器阅读以及编辑的私人格式、DTD 以及或处理工具不能够一般地加以取得的 SGML 或 XML、以及由某些文书处理器只是为了输出的目的而做出的，由机器制作的 HTML、PostScript 或 PDF。

标题页对一本印刷书籍来说，指的是标题页本身，以及所需要用来容纳本许可证必须出现在标题页的易读内容的，如此的接续数页。对于并没有任何如此页面的作品的某些格式，标题页指的是本文主体开始之前作品标题最显着位置的文字。

出版者指的是把那些文本副本发布给公众的任何人或实体。

一个标题为 XYZ 的章节指的是文件的一个具名的次要单元，其标题

精确地为 XYZ 或是将 XYZ 包含在跟着翻译为其它语言的 XYZ 文字后面的括号内——这里 XYZ 代表的是名称于下提及的特定章节，像是感谢、贡献、背书或历史。当你修改文件时，给像这样子的章节保存其标题指的是，它保持为一个根据这个定义的标题为 XYZ 的章节。

文件可以在用来陈述本许可证效力及于文件的声明后，包括担保放弃。这些担保放弃被考虑为以提及的方式，包括在本许可证中，但是只被看作为放弃担保之用：任何这些担保放弃可能会有的其它暗示都是无效的，并且也对本许可证的含义没有影响。

以下是有关复制、发布及修改的明确条款及条件。

## 2. 逐字的复制

你可以复制或发布文件于任何媒介，而不论其是否具有商业买卖行为，其条件为具有本许可证、版权声明和许可声明，说明本许可证适用于文件的所有副本，并且你没有增加任何其它条件到本许可证的条件中。你不可以使用技术手段，来妨碍或控制你所制作或发布的副本阅读或进一步的发布。然而，你可以接受补偿以作为副本的交换。如果你发布了数量足够大的副本，你也必须遵循第三条的条件。

你也可以在上述的相同条件下借出副本，并且你可以公开地陈列副本。

## 3. 大量地复制

如果你出版文件的印刷副本或者通常具有印刷封面的媒体的副本，数量上超过一百个单位，而且文件许可声明要求有封面文字，那么你必须将这些副本附上清楚且易读的文字：前封面文字于前封面上、后封面文字于后封面上。这两种封面必须清楚易读地辨认出，你是这些副本的出版者。前封面文字必须展示完整的标题，而标题的文字应当同等地显著可见。你可以增加额外的内容于封面上。仅在封面作出改变的复制，只要它们保存了文件的标题，并且满足了这些条件，可以在其它方面被看作为逐字的复制。

如果对于任意一个封面所需要的文字，数量过于庞大以至于不能符合易读的原则，你应该在实际封面的最前面列出所能符合易读原则的内容，然后将剩下的接续在相邻的页面。

如果你出版或发布数量超过一百个单位文件的混浊副本，你必须与此混浊副本一起包含一份机器可读的透明副本，或是与一份混浊副本一起或其陈述一个计算机网络位置，使一般的网络使用公众具有存取权，可以使用公开标准的网络协定，下载一份文件的完全透明副本，此副本中并且没有增加额外的内容。如果你使用后面的选项，当你开始大量地发布混浊副本时，你必须采取合理的审慎步骤，以保证这个透明副本将会在发布的一开始就保持可供存取，直到你最后一次发布那个发行版的一份混浊副本给公众后，至少一年为止。以保证这个透明副本，将会在所陈述的位址保持如此的可存取性，直到你最后一次发布那个发行版直接或经由你的代理商或零售商的一份混浊副本给公众后，至少一年为止。

你被要求，但不是必须，在重新发布任何大数量的副本之前与文件的作者联络，给予他们提供你一份文件的更新版本的机会。

## 4. 修改

你可以在上述第二条和第三条的条件下，复制和发布文件的修改版本，其条件为你要精确地在本许可证下发布修改版本，且修改版本补足了文件的角色，从而允许修改版本的发布和修改权利给任何拥有它副本的人。另外，你必须在修改版本中做这些事：

**第一款、** 在标题页或在封面上使用，如果有与先前版本不同的文件，应该被列在文件的历史章节不同的标题。如果版本的原始出版者允许，你可以使用与某一个先前版本相同的标题。

**第二款、** 在修改版本的标题页上列出担负作者权的一个或多个人或实体作为作者，并且列出至少五位文件的主要作者。如果少于五位，则列出全部的主要作者，除非他们免除了你这个要求。

**第三款、** 在标题页陈述修改版本的出版者的名称作为出版者。

**第四款、** 保存文件的所有版权声明。

**第五款、** 为你的修改增加一个与其它版权声明相邻的适当的版权声明。

**第六款、** 在版权声明后面，以许可证附录所显示的形式，包括一个给予公众在本许可证条款下使用修改版本的许可声明。

**第七款、** 在那个许可声明中保存固定章节和文件许可声明中必要封面文字的全部列表。

**第八款、** 包括一个未被改变的本许可证的副本。

**第九款、** 保存标题为历史的章节和其标题，并且增加一项至少陈述如同在标题页中所给的修改版本的标题、年份、新作者和出版者。如果在文件中没有标题为历史的章节，则制作出一个陈述如同在它的标题页中所给的文件的标题、年份、新作者和出版者，然后增加一项描述修改版本如前面句子所陈述的情形。

**第十款、** 如果有的话，保存在文件中为了给公众存取文件的透明副本，而给予的网络位址，以及同样地在文件中为了它所根据的先前版本，而给予的网络位址。这些可以被放置在历史章节。你可以省略一个在文件本身之前，已经至少出版了四年的作品的网络位址，或是如果它所参照的那个版本的原始出版者给予允许的情形下也可以省略它。

**第十一款、** 在任何标题为感谢或贡献的章节，保存章节的标题，并且在那章节保存到那时候为止，每一个贡献者的感谢以及或贡献的所有声色。

**第十二款、** 保存文件的所有固定章节，于其文字以及标题皆不得变更。章节号码或其同等物并不被认为是章节标题的一部份。

**第十三款、** 删除任何标题为背书的章节。这样子的章节不可以被包括在修改版本中。

**第十四款、** 不要重新命名任何现存的章节，而使其标题为背书，或造成与任何固定章节相冲突的标题。

**第十五款、** 保存任何的担保放弃。

如果修改版本包括新的本文之前内容的章节，或合乎作为次要章节的附录，并且没有包含复制自文件的内容，则你具有选择可以指定一些或全

部这些章节为恒常的。要这样做，将它们的标题增加到在修改版本许可声明中的固定章节列表中。这些标题必须可以和任何其它章节标题加以区别。

你可以增加一个标题为背书的章节，其条件为它仅只包含由许多团体所提供的你的修改版本的背书——举例来说，同侪评审的说明，或本文已经被一个机构认可为一个标准的权威定义。

你可以增加一个作为前封面文字的最多五个字的段落，以及一个作为后封面文字的最多二十五个字的段落，到修改版本的封面文字列表的后面。前封面文字和后封面文字都只能有一个段落，可以经由任何一个实体，或经由任何一个实体所作出的安排而被加入。如果文件已经在同样的封面包括了封面文字——先前由你或由你所代表的相同实体所作出的安排而加入，则你不可以增加另外一个；但是你可以在先前出版者的明确允许下替换掉旧的。

文件的作者和出版者并不由此授权，而给予允许使用他们的名字以为了或经由声称或暗示任何修改版本背书为自己所应得的方式而获得名声的权利。

## 5. 组合文件

你可以在上述第四条的条款中对于修改版本的定义之下，将文件与其它在本许可证下发行的文件组合起来，其条件是你要在组合品中，包括所有原始文件的所有固定章节，不做修改，同时在组合作品的许可声明中将它们全部列为固定章节，并且你要保存它们所有的担保放弃。

组合作品只需包含本许可证的一份副本，并且重复的固定章节可以仅以单一个副本来取代。如果名称重复但内容不同的固定章节，则将任此章节的标题，以在它的后面增加的方式加以独特化，如果已知的话，于括号中指出那个章节的原始作者或出版者的名称，或是指定一个独特的号码。在此组合作品许可声明中固定章节的列表中，对其章节标题也作出相同的调整。

在组合品中，你必须组合在不同原始文件中，标题为历史的任何章节，

形成一个标题为历史的章节；同样组合任意标题为感谢或贡献的章节。你必须删除标题为背书的所有章节。

## 6. 文件的收集

你可以制作含有文件以及其它以本许可证发行文件的收集品，并且将本许可证对不同文件中的个别副本，以单一个包括在收集品的副本取代，其条件是你要遵循在其它方面，给予一个文件逐字复制的允许本许可证的规则。

你可以从这样的一个收集品中抽取出一份单一的文件，并且在本许可证下将它单独地发布，其条件是你要在抽取出的文件中插入本许可证的一份副本，并且在关于那份文件的逐字的复制的所有其它方面，遵循本许可证。

## 7. 独立作品的聚集

一个文件的编辑物，其中或附加于储存物或发布媒体的一册的，具有其它分别且独立的文件或作品的衍生品，如果经由编辑而产生的版权，并没有用来限制此编辑物使用者的法律权力，而超过了个别的作品所允许的，则被称为一个聚集品。当文件中包括一个聚集品，本许可证的效力并不仅在于此聚集品中的，于其本身并非文件的衍生作品的其它作品。

如果第三条的封面文字要求效力于这些文件的副本，并且文件的篇幅少于整个聚集品的一半，则文件的封面文字可以被放在只围绕着文件，并于聚集品内部的封面或是电子的封面同等物上，如果文件是以电子的形式出现的话。否则它们必须出现在绕着整个聚集品的印刷封面上。

## 8. 翻译

翻译被认为一种修改，因此你可以在第四条的条款下发布文件的翻译。用翻译更换固定章节需要取得版权所有者的特别允许，但是你可以包括部份或所有固定章节的翻译，使其附加到这些固定章节的原始版本之

中。你可以包括本许可证、文件中的所有许可声明和任何的担保放弃的翻译，其条件为你也必须包括本许可证的原始英文版本，以及这些声明与放弃的原始版本。如果发生翻译与本许可证、声明或放弃的原始版本有任何的不同意时，将以原始版本为准。

如果在文件中的章节被标题为感谢、贡献或历史，则保存标题第一条的必要条件第四条，典型上将会需要去更动实际的标题。

## 9. 终止

除了本许可证明确规定外，你不可以复制、修改、在本许可证下再设定额外条件的次授权或发布文件。任何其它的复制、修改、在本许可证下再设定额外条件的次授权、或发布文件的意图都是无效的，并且将会自动地终止你在本许可证下所被保障的权利。

然而，如果你终止所有违反本许可证的行为，特定版权所有人会暂时恢复你的授权直到此版权所有者明确并最终地终止你的授权。或者特定版权所有人永久地恢复你的授权如果此版权所有人在停止违反许可证后的 60 天内没有通过合理的方式通知你违反许可证。

此外，此版权所有者用一些合理的方式通知你违反了许可证规定，也是你第一次从此版权所有者收到违反许可证的通知，并且你在收到通知后的 30 天内终止了这种行为，那么此版权所有者会永久地恢复你的授权。

你的权利在此章节的终止并不代表终止得到你的副本和权利的当事人的授权。如果你的权利被终止而没有被永久性地恢复，那么你将没有任何权利去使用此章节的全部或部分资料和资料的副本。

## 10. 本许可证的未来改版

自由软件基金会可能会不定期地出版自由文档许可证新修订过的版本。这种新版本在精神上将会与现在的版本相似，但在细节上可能会有不同，以对应新的问题或相关的事。请见 http://www.gnu.org/copyleft/。

本许可证的任何版本都被指定一个可供区别的版本号码。如果文件指

定一个效力于它的特定号码版本的本许可证或任何以后的版本，你就具有选择遵循指定的版本，或任何已经由自由软件基金会出版的后来版本并且不是草稿的条款和条件。如果文件并没有指定一个本许可证的版本号码，你就可以选择任何一个曾由自由软件基金会所出版的不是草稿的版本。如果文件指定一个代理能够决定本许可证未来的那一种版本可以用，而且还指定代理公开声明如果你接受一种版本你将会永久地被授权为文本选择此版本的权利。

## 11. 重新授权

“MMC 网站” 是指任何发布有版权作品的网站服务器，也为任何人提供卓越的设施去编辑一些作品。任何人都可编辑的一种公众维基就是这种服务器的一个例子。包含在这个网站的 “MMC” 是指任何一套在 MMC 网站上发布的具有著作版权的作品。

“CC-BY-SA” 是指 Creative Commons Attribution-Share Alike 3.0 许可证，它是被 “知识共享组织” 颁布的，“知识共享组织” 是一家非赢利性的，在圣弗朗西斯科，加利福尼亚具有重要的商业地位的组织。而且未来的 “copyleft” 版本的许可证也是被同一个组织发布的。

“合并” 是指以整体或作为另一个文本的部分发布或重新发布一个文本。

MMC 有重新授权的资格，如果它是在 MMC 下授权；或者所有的作品首次发布并非在 MMC 下授权，后来以整体或部分合并到 MMC 下，它们没有覆盖性文本或不变的章节并且在 2008 年 11 月 1 日之前合并的。

那么 MMC 网站的操作者会在 2009 年 8 月 1 日之前的任何时间在同一网站重新发布包含在这一网站的 MMC 是经过 CC-BY-SA 授权的，只要那个 MMC 有资格重新授权。

## 许可证附录：如何使用本许可证用于你的文件

为使用本许可证成为你撰写成的一份文件，必须在文件中包括本许可证的一份复本，以及标题页的后面包括许可声明：

```
Copyright (c) YEAR YOUR NAME. Permission is granted to copy,
distribute and/or modify this document under the terms of
the GNU Free Documentation License, Version 1.2 or any later
version published by the Free Software Foundation; with no
Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts. A copy of the license is included in the section
entitled "GNU Free Documentation License".
```

如果你有固定章节、前封面文字和后封面文字，请将 with... Texts 这一行以这些文本取代：

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts
being LIST.
```

如果你有不具封面文字的固定章节，或一些其它这三者的组合，将可选择的二项合并以符合实际情形。

如果你的文件中包含有并非微不足道的程序码范例，我们建议这些范例平行地在你的自由软件授权选择下，比如以 GNU General Public License 的自由软件授权来发布，从而允许它们作为自由软件而使用。

因为本书是基于 **GNU FPL** 许可证发布的，所以下面附加 **FDL** 许可证的英文原版，同时亦可与前面中文翻译版共同帮助理解。特此注明。

---

## GNU Free Documentation License

Version 1.3, 3 November 2008

1. **PREAMBLE**

   The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

   This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

   We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

2. **APPLICABILITY AND DEFINITIONS**

   This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions

stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general pub-

lic, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeXinput format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text

that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

3. **VERBATIM COPYING**

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

4. **COPYING IN QUANTITY**

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers

that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

5. **MODIFICATIONS**

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified

Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

(a) Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
(b) List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
(c) State on the Title page the name of the publisher of the Modified Version, as the publisher.
(d) Preserve all the copyright notices of the Document.
(e) Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
(f) Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
(g) Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
(h) Include an unaltered copy of this License.
(i) Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year,

authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

(j) Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

(k) For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

(l) Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

(m) Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

(n) Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

(o) Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

6. **COMBINING DOCUMENTS**

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if

known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

7. **COLLECTIONS OF DOCUMENTS**

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

8. **AGGREGATION WITH INDEPENDENT WORKS**

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of

the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

9. **TRANSLATION**

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

10. **TERMINATION**

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until

the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

11. **FUTURE REVISIONS OF THIS LICENSE**

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version

for the Document.

12. **RELICENSING**

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

**ADDENDUM: How to use this License for your documents**

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this
document under the terms of the GNU Free Documentation License,
Version 1.3 or any later version published by the Free Software
Foundation; with no Invariant Sections, no Front-Cover Texts,
and no Back-Cover Texts. A copy of the license is included in
the section entitled "GNU Free Documentation License".
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with…Texts.” line with this:

```
with the Invariant Sections being list their titles, with the
Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

# 关于出售例外对 GNU GPL 的影响



当我参与联合签署知识生态国际（Knowledge Ecology International）发起的公开信，警告 Oracle 收购 MySQL（加上 Sun 公司剩下的部分）可能不利于 MySQL 的时候，出售许可证例外便成为了一个热门话题。

就像以下文章解释的那样，我对出售许可证例外的感觉是混杂的。明显的是，不出售例外在 GNU GPL 下开发强大而复杂的软件包是可能的，我们在做这个。MySQL 也可以这样的方式开发。然而，MySQL 开发者已经用到出售例外的做法了。谁应该决定是否继续这样做呢？我认为把关于一个自由软件项目的主要决定交给一个大型私有竞争者是不明智的，它会自然地使项目缩减，而不是更多地开发。

一件不合理的事情是更改 MySQL 的许可证到某种非 copyleft 许可证的想法。那样会消除出售例外的可能，但允许所有种类的私有修改版本。无论 MySQL 应该走到哪里，都不应选择非 copyleft 许可证。

当我共同签署了反对 Oracle 计划好的 MySQL 收购[1]（还有 Sun 公司剩下的部分）的信件的时候，一些自由软件支持者很惊讶我会赞成 MySQL 开发者用过的出售许可证例外的行为。他们期望我完全地谴责这种行为。这篇文章解释了我对这种行为的想法，以及原因。

出售例外的意思是代码的版权所有者以自由软件许可证将代码发布至公众，然后让顾客为在不同条款下使用同样代码的许可付款，例如允许它

[1] James Love and Malini Aisola (Knowledge Ecology International), Richard Stallman (FSF), Jim Killock (Open Rights Group), letter to Neelie Kroes (Commissioner for Competition, European Commission), 19 October 2009, http://keionline.org/sites/default/files/ec_letter_mysql_oct19.pdf.

包括在私有应用里面。

我们必须把出售例外的做法和另一件至关重要的不同事情区分开：一个自由程序的完全私有的扩展或版本。这两种行动，即使是同时被一个公司所实行，都是不同的问题。在出售例外中，例外所用于的相同代码是以自由软件的形式对公众可用的。只在私有许可证下的一个扩展或者修改后的版本是完全的私有软件，并且不会优于任何其他私有软件。这篇文章关心的是严格地并且只涉及例外出售的情形。

我从 20 世纪 90 年代开始就认为出售例外是可以接受的，并且有时我已经对公司建议过。有时这种做法已经让重要的程序成为自由软件成为可能。

KDE 桌面在 90 年代基于 Qt 库开发。Qt 那时是私有软件，并且 TrollTech 公司对把它嵌入在私有应用中的许可收费。TrollTech 允许 Qt 在自由应用中免费使用，但这不会让它成为自由软件。因此完全自由的操作系统不会包括 Qt，所以它们也不能用 KDE.

1998 年，TrollTech 的管理层认识到他们可以让 Qt 成为自由软件并且继续对把它嵌入在私有软件中的许可收费。我不记得这个建议是不是从我来的，但看到了这个改变我当然高兴，它让在自由软件世界中使用 Qt 和 KDE 成为可能。

开始的时候，他们用自己的许可证，Q 公共许可证（QPL）——就自由软件许可证来说是很有限制性的，并且和 GNU GPL 不兼容。之后，他们转到了 GNU GPL；我想我已经给他们解释了这可以为这种目的起作用。

出售例外从根本上依赖于为自由软件的发布使用一个 copyleft 许可证，如 GNU GPL. 一个 copyleft 许可证允许嵌入一个更大的程序仅当整个合并后的程序是以那个许可证发布的；这是它确保扩展版本也自由的做法。因此，想让合并后的程序私有化的用户需要特殊的许可。只有版权所有者可以授予那个许可，而出售例外就是这样做的一种方式。其他人，在 GNU GPL 或者另一个 copyleft 许可证下收到代码的，不能授予例外。

当我第一次听说出售例外的做法时，我问自己这种行为是否符合伦理。如果某人买一个例外来把一个程序嵌入一个更大的私有程序，他正在

做错误的事情（换句话说，制作私有软件）。是否出售例外的开发者也在做错误的事情呢?

如果那个含义是有效的，那么它也能用于发布同样软件于非 copyleft 自由许可证下的行为，如 X11 许可证，那也允许了这样的嵌入。所以我们要么不得不得出结论，以 X11 许可证发布任何东西都是错误的，或者拒绝这种含义。使用一个非 copyleft 许可证是不好的，并且通常是一个次等的选择，但它不是错误的。

换句话说，出售例外允许私有软件中有限地嵌入代码，但 X11 许可证走得更远，允许无限制地在私有软件中使用代码（及其修改后的版本）。如果这不让 X11 许可证不可接受，那么它不能让出售例外不可接受。

FSF 不实行出售例外有三个原因。一个是它不会通往 FSF 的目标：确保我们软件的每个用户是自由的。那是我们写 GNU GPL 的目的，最彻底地实现这个的方法就是在 GPL 第三版或后续版本下发布，并且不允许嵌入于私有软件。出售例外不能实现这个，就像在 X11 许可证下发布不能做到一样。所以正常我们不会做那两种事情。我们只在 GPL 下发布。

我们只在 GPL 下发布的另一个原因是为了不允许在我们自由程序上出现会呈现实用优势的私有扩展。不把自由看作价值的用户可能会选择那些非自由的版本，而不是它们所基于的自由的版本——从而失去他们的自由。我们不想鼓励那种情况。

但会有偶然的情况，为了特定的战略原因，我们判定在一个特定的程序用更宽松的许可证对自由的事业更为有利。在那些情形，我们在那个宽松许可证下给每个人发布这个程序。

这是因为另一个 FSF 跟随的伦理准则：无差别地对待所有用户。为了自由的理想主义运动不应当歧视，所以 FSF 致力于给所有用户同样的许可证。FSF 从来不出售例外；无论我们以什么许可证发布一个程序，它都是所有人能够获得的。

但我们不需要坚持公司们遵循那条原则。对于公司我认为出售例外是可以接受的，并且我会在合适的时候建议这样做，作为让程序自由化的一种方式。

# 第 6 部分

# 陷阱和挑战

## 您能够信任您的计算机吗？



您的计算机应该听从谁的命令？大多数人认为他们自己的计算机应当听从他们自己的命令而非他人的。但是，大型媒体公司（包括电影公司和唱片公司）正在联合大型计算机公司，诸如英特尔和微软，试图通过一项它们称之为“信任计算”的阴谋使得您的计算机听命于它们而非您（这种阴谋的微软版本称为 Palladium）。以前的私有软件本来就包含恶意功能，但现在这项阴谋将会使它们变为无恶不作的恶意软件。

从本质上说，私有软件意味着您不能控制它们的所作所为；您不能研究它们的源代码也不能修改它们。精明的商业人士总会找到办法以便利用他们的控制权将您置于不利地位，这并不令人感到丝毫惊讶。微软曾经多次这样做：某个版本的 Windows 被设计为能够向微软报告您的硬盘驱动器上安装的所有软件；最近的一项用于 Windows Media Player 的“安全”更新强制用户接受新的限制。但是，微软并不是唯一一家正在如此做的：例如

KaZaa 音乐共享软件被设计为使得 KaZaa 的业务合作伙伴可以将您的计算机使用状况出卖给它们的客户。这些恶意的功能通常是隐秘的，但即使您能够知道它们的存在，也难以移除它们，由于您不能访问它们的源代码。

在过去，这些行为只是孤立事件。然而，所谓的“信任计算”将会使得这种行为无处不在。“背叛计算”无疑是一个更为贴切的名字，由于这个阴谋被设计的初衷是确保您的计算机系统性地不再听命于您。事实上，它被设计为使您的计算机不再成为通用目的计算机。任何一项操作都可能要求得到它们的明确授权。

背叛计算的基本技术思想是使计算机包含一个数字加密与签名设备，而它所用的密钥对您是绝密的。私有软件将会利用这一设备来控制您可以运行哪些其他程序，您可以访问哪些文档或数据，以及您可以把它们传给什么程序。这些程序将会不断从互联网上下载更多的认证限制规则，并且自动将这些限制强加到您的工作中。如果您不想让您的计算机定期从互联网上获取新的限制规则，一些功能将会自动停止工作。

当然，好莱坞和唱片公司计划将背叛计算用于数字限制管理（DRM），这样，您下载到的视频和音乐将只能在某一台特定的计算机上播放。分享将会完全成为不可能，至少是对于那些您从这些公司获得的经过认证的文件。您，作为公众的一员，理应拥有分享这些内容的自由和能力（我期望某些人能够找到一种方式来制作未加密的版本并且上传分享它们，这样 DRM 将不能完全得逞，但是，这不能成为我们原谅它的理由）。

禁止分享已经足够恶劣了，但是事情正在向着更坏的方向发展。已经有计划将这种机制用于邮件和文档——使得邮件将会在两周之后消失，或者文档只能被某家公司的计算机读取。

假设您收到上司的邮件，对方要求您去做某些您认为过于激进冒险的事情。一个月之后，当事情产生相反的效果时，您不能利用那封消失了的邮件证明这一决策并非您所做出。此时的白纸黑字并不能证明您的清白，如果它是用会褪色的墨水写的。

假设您收到上司的邮件，邮件中说明了一种非法的或者通常是令人无法容忍的政策，例如销毁您的公司的审计档案，或者放任某种对您的国家

的威胁发展坐大。今天，您还可以将其发送给记者并曝光这种行为。但是，一旦有了背叛计算，记者将不能读取您的文档；由于记者的计算机拒绝听命于本人。背叛计算将会成为腐败和犯罪的渊薮。

诸如微软 Word 等文字处理器可能会使用背叛计算来保存您的文档，以使得竞争对手的文字处理器不能读取它们。如今，我们必须试图通过耗时费力的试验来破解 Word 格式的秘密，以使得自由的文字处理器能够读取 Word 文档。但如果 Word 在保存文档时使用背叛计算加密文档内容，自由软件社区将毫无机会开发出能够读取它们的软件——即使我们能够开发出来，这样的软件也可能甚至会被数字千年版权法案（DMCA）判定为非法。

使用背叛计算的软件将会持续从互联网下载新的认证限制规则，并且将其强加于您的工作。如果微软或者美国政府不喜欢您在文档中所写的内容，它们可以发送新的指令通知所有计算机拒绝任何人读取该文档。每台计算机在下载到新的指令时都会执行它。这样，您所写的内容将会遭受《一九八四》式的反动的抹除，甚至您自己都可能再也不能读取它。

您可能认为您能够通过某种方式发现一款背叛计算的应用程序会做出哪些龌龊的事情，了解那些行为是多么地让人痛苦，然后决定是否接受它们。但即使您能够发现这些，对您来说接受这些协议也是愚蠢的，您甚至不能指望这些条款会维持现状而不是变得更坏。您一旦依赖于使用这些软件，您已经对此成瘾并且它们知道这一点；然后它们可以更改条款使其对您更加不利。一些应用程序会自动下载更新，而这些更新会在暗中做一些其他事情——而它们不会给您是否接受更新的选择权。

如今，您仍然可以通过拒绝使用私有软件来避免受其限制。如果您运行 GNU/Linux 或者其他的自由操作系统，并且您避免在其上安装私有软件，那么您仍然掌控着您的计算机的行为。如果一款自由软件包含恶意功能，社区中的其他开发者将会将其移除，您将能够使用修正后的版本。您也可以在非自由操作系统上运行自由软件和工具；这并不能完全赋予您自由，但很多用户确实如此做。

背叛计算将自由操作系统和自由软件置于绝境，因为您甚至可能完全

不能运行它们。某些版本的背叛计算将会要求操作系统必须是由某家特定公司所具体认证的。这时自由操作系统将不被允许安装。某些版本的背叛计算可能会要求每个应用程序都要经过操作系统开发厂商具体认证。这时，您不能在这样的操作系统上运行自由软件。如果您设法找到破解方法并且告知他人，这将被视为犯罪行为。

事实上，美国法律已经有提案要求所有计算机支持背叛计算，并且禁止老旧计算机接入互联网。美国消费者宽带和数字电视推广法案（CBDTPA，我们称之为 Consume But Don't Try Programming Act，即“消费但不要尝试编程法案”）就是其中之一。但即使它们还没有从法律层面完全强制您转向背叛计算，这也会为您施加巨大压力以迫使您最终接受。如今，人们通常在通讯中使用 Word 文档格式，尽管这会导致诸多问题[1]。如果只有支持背叛计算的机器才能读取最新的 Word 文档，那么很多人将会转向它，如果他们仅仅将这种情况看作个人行为（即“爱用不用”）。为了反对背叛计算[2]，我们必须联合起来，并且作为一种集体的抉择来勇敢面对这一困境。

阻止背叛计算将会需要大量的公民来组织参与。我们需要您的帮助！请支持 DefectiveByDesign.org，自由软件基金会反对数字限制管理的运动。

## 补篇

1. 计算机安全领域以另一种方式使用短语“信任计算”——注意不要混淆二者的涵义。

2. GNU 计划发布了 GNU 隐私卫士（GPG），这是一款能够实施公钥加密和数字签名的软件，您可以使用它发送安全私密的邮件。您需要

[1] 参见我的文章“We Can Put an End to Word Attachments,”位于 http://gnu.org/philosophy/no-word-attachments.html，以获知 Word 文档所能导致的一系列问题的描述以及我们关于如何解决它们的一些建议。

[2] 如需获得更多信息，参见“‘Trusted Computing’ Frequently Asked Questions,”一文，位于 http://www.cl.cam.ac.uk/users/rja14/tcpa-faq.html.

认清 GPG 和背叛计算的本质区别，并且知道为何前者是有益的而后者是极度阴险的。

当某人使用 GPG 向您发送一份加密文档，并且您使用 GPG 对其解密，将会得到一份未加密的文档。您可以读取、复制、回复甚至重新对其加密并且安全地发送给他人。而背叛计算应用程序只是让您可以读取屏幕上的单词，但并不允许您生成一份未加密的文档副本用于其他用途。GPG，作为一款自由软件，使安全特性对用户可用，即 “他们利用它”。而背叛计算被设计为向用户施加限制，即 “它利用他们”。

3. 背叛计算的支持者总是着重论述其好处。他们所说的往往是正确的，但并不重要。

   同大多数硬件一样，背叛计算硬件也可以被用于非恶意用途，但这些功能也可由不带背叛计算的硬件以其他方式实现。基本的区别在于背叛计算对用户所做的是这样龌龊的事情：让您的计算机以对您不利的方式运行。

   他们说的是事实，而我说的也是事实。将二者放在一起考虑将会得出什么结论？背叛计算是一套剥夺我们的自由的阴谋，它提供了一些小恩小惠以吸引我们的注意力，使我们忽视我们将会因此失去的更为重要的东西。

4. 微软将 Palladium 作为一项安全措施推出，并宣称它将提供反病毒保护，但这是确凿的谎言。由微软研究院于 2002 年十月提供的一份演示文稿指出 Palladium 的特性之一是现存的操作系统和应用程序仍然能够运行；因此，病毒自然可以继续做它们现在所能做的一切。

   当微软员工提及与 Palladium 相关联的 “安全” 概念时，它并非指的是我们通常理解的安全：保护您的机器免受您不想要的东西的危害。相反，他们指的是保护您的计算机上的数据，并且阻止您以其他人不喜欢的方式访问它们。演示文稿的一页幻灯片列出了 Palladium 可

用于保守的几类机密，包括“第三方机密”和“用户机密”——但它将“用户机密”置于引号之中，指出这在 Palladium 的语境中是一种荒谬的东西。

该演示文稿频繁使用一些我们经常与安全语境相关联的短语，诸如“攻击”、“恶意代码”、“冒名顶替”，当然还有“信任”。它们在这里的涵义都不是其通常的涵义。例如，“攻击”不是指某人试图伤害您，而是指您试图复制受保护的音乐；“恶意代码”指您自行安装的代码，用于使您的机器去做那些其他人不希望您做的事情；“冒名顶替”不是指其他人愚弄您，而是指您愚弄 Palladium，等等。

5. 先前由 Palladium 开发者发出的一份声明提出了一种基本的前提条件，即信息的任何制造者和收集者对于您如何使用该信息应当拥有完全控制权。这意味着对已有的伦理和法律系统理念的彻底颠覆，并且将会创建一个史无前例的控制体系。这些系统的具体问题绝非偶然，因为它是基于上述基本目标的。这种基本目标是我们必须坚决反对的。

6. 直至 2015 年，背叛计算在个人计算机（PC）中以“信任平台模块”（TPM）的形式实现；然而由于实践上的原因，TPM 被证实完全未能实现其目标：为数字限制管理（DRM）提供远程认证平台。因此，计算机公司以其他方式实现 DRM。现在，TPM 完全没有被用于实现 DRM，并且有理由怀疑将它们用于 DRM 是根本不可行的。讽刺的是，这意味着只有当前的 TPM 应用是无害的非重要应用——例如，证实没有人偷偷地更改了计算机中的系统。

   因此，我们得出结论：PC 中的 TPM 并不危险，没有理由不在 PC 中安装一块 TPM，或者不对其提供任何系统软件支持。

   这并不意味着所有事情都是美好的。其他禁止计算机的所有者改变其中的软件的硬件系统正在某些 ARM PC 以及移动电话、汽车、电视机和其他设备的处理器上使用，这与我们所预期的一样坏。

这也不意味着远程认证是无害的。如果一旦有一款设备成功实施了它，它将对用户的自由造成毁灭性的影响。当前的 TPM 无害，仅仅是因为它暂时未能实现其使远程认证成为可能的企图。我们一定不要假设它在未来的所有企图都将失败。

# JavaScript 陷阱



> 您可能每天都在您的计算机上通过您的浏览器运行非自由软件而没有意识到这一点。

在自由软件社区，非自由软件虐待它们的用户这一观念为人们所熟知。我们中的一些人完全拒绝安装任何私有软件，众多其他人将没有自由视为不采用这种软件的一大理由。很多用户注意到这一准则同样适用于浏览器提供安装的插件，因为它们可以是自由的，也可以是非自由的。

但是，浏览器会运行其他非自由软件而不会询问甚至不会告知您——那些包含在网页中或者链接到网页的程序。这些程序通常由 JavaScript 编写，尽管其他编程语言也会被用到。

JavaScript（官方名称为 ECMAScript，但是这个名字很少被使用）曾经仅仅用于网页中的非主要修饰部分，例如精巧但无关紧要的导航或是显示特性。将这些次要功能仅仅看作超文本标记语言（HTML）的扩展而非真正意义上的软件是可以接受的；它们并不会构成一个主要问题。

很多网站仍然以这种方式使用 JavaScript，但有些网站将其用于主要的程序以执行大型任务。例如谷歌 Docs 将会向您的计算机下传一个大约 0.5 MB 的 JavaScript 程序，这个程序用一种我们称之为混淆脚本（Obfuscript）的紧密格式编写，由于其中没有注释，也极少使用空白字符，而方法的名称只有一个字母。程序的源代码应当采用有利于修改它的形式；因此这种紧密的代码不能称之为源代码，而此程序的真正源代码仍然对其用户不可用。

浏览器通常不会告知您它们何时加载 JavaScript 程序。大多数浏览器提供了一种完全关闭 JavaScript 的方式，但是没有浏览器可以检测出哪些 JavaScript 程序是非平凡、非自由的。即使您注意到了这个问题，要想识别并阻止这些程序可能会给您带来相当可观的麻烦。然而，即使是在自由软件社区内，也有很多用户并未觉察到这一问题；浏览器的沉默倾向掩盖了

这个问题的存在。

可以将一段 JavaScript 程序作为自由软件发布，通过在自由软件许可证下发布其源代码。但是，即使程序的源代码可获得，并没有一种简易的方法来使用您的修改版本替代原始版本。当前的自由浏览器并没有提供这样一种机制来使用您自己修改过的版本替代来自网页的版本。这种影响可以和 “TiVo 化”（tivoization）相比，尽管并不是那么非常难以克服。

JavaScript 并非网站向用户推送的程序所使用的唯一编程语言。Flash 支持通过 JavaScript 的一种扩展变体进行编程。我们需要研究 Flash 的问题来做出合理的建议。Silverlight 看起来将会带来和 Flash 相似的问题，唯一的区别在于它比 Flash 更坏，由于微软将其用作非自由编码解码器的平台。一款 Silverlight 的自由软件替代品并不能为自由软件世界解决问题，除非它也能正常地提供自由的编码解码器替代品。

Java 小程序也是在浏览器中运行的，它们也会带来类似的问题。从一般意义上来说，任何类型的小程序系统都会带来这类问题。然而，只要拥有一个自由的小程序执行平台，就足以使我们能够克服这个问题。

已经有一场声势浩大的运动号召网站仅仅使用自由的（有些人称之为 “开放的”）格式和协议进行通讯；这些格式和协议分别是指用于发布文档的格式和任何人均可自由实现的协议。由于程序在网页中的存在，这样的标准尺度是必需的，但并不足够。JavaScript 作为一种格式，其本身是自由的，并且在网络中使用 JavaScript 程序并非绝对地坏。然而，如同我们在上文所见，这也不一定就没问题。当网站向用户传送某个程序时，如果程序是用一种有文档说明的、不受专利限制的编程语言所编写的，仅仅满足这一点并不足够。该程序本身也必须是自由的。“只有自由软件才能被传送到用户” 必须成为网络良好行为准则的一部分。

静默地加载并运行程序是 “网页应用” 所带来的诸多问题之一。“网页应用” 这一短语是被设计为故意忽视交付给用户的软件与在服务器上运行的软件之间的根本区别的。它可以指代在浏览器中运行的专门的客户端程序；也可以指代专门的服务器软件；还可以指代与专门的服务器软件共同工作的专门的客户端程序。客户端和服务器端的不同将会引起不同的伦理

问题，即使它们整合得如此紧密，以至于它们可以被认为是同一个软件的组成部分。本文仅仅讨论客户端软件带来的问题。我们将会在另一篇文章中讨论服务器端的问题。

在实践层面，我们怎样才能解决由网络中的私有 JavaScript 程序带来的问题呢？第一步当然是避免运行它们。

那么，我们之前提到的“非平凡”是指什么呢？这指的是一种程度，因此与之相关的问题是设计出一种简单的标准尺度使其能够带来好的结果，而非试图寻求唯一正确的答案。

我们试着提出了一种策略，这种策略认为一段 JavaScript 程序如果是非平凡的，那么它需要满足：

- 它会提交异步 JavaScript 和可扩展标记语言（AJAX）请求，或者与提交 AJAX 请求的脚本同时被加载；
- 它会动态加载外部脚本，或者与动态加载外部脚本的脚本同时被加载；
- 它定义了这样的函数或方法：这些函数或方法要么从 HTML 加载外部脚本，要么作为外部脚本被加载；
- 它使用了不对程序进行转译就难以对其进行分析的动态 JavaScript 结构，或者与使用了这样的结构的脚本一同被加载。这些结构包括：
  - 使用 eval 函数；
  - 使用方括号标记调用方法；
  - 使用除了带有某些方法的字符串字面量（例如 Obj.write、Obj.createElement）以外的任何其他结构。

那么，我们又该如何判断一段 JavaScript 代码是否自由呢？我们在本文末尾提出了一种常用约定。根据这种约定，网页中的一段非平凡 JavaScript 程序可以指出其源代码所位于的统一资源定位符（URL），并且使用形象化的注释指出其所使用的许可证类型。

最后，我们还需要改变自由的浏览器来检测并阻止网页中的非平凡非自由 JavaScript 代码。一款名为 LibreJS 的自由软件能够检测并阻止您所访问的网页中的非平凡非自由的 JavaScript 代码[1]。LibreJS 是一款适用于 IceCat 和 IceWeasel（还有 FireFox）浏览器的插件。

浏览器用户还需要一款简便的工具以指定用户想要使用的 JavaScript 代码，而非直接使用某一特定网页提供的 JavaScript 代码。（用户所指定的代码可以是基于网页自带的自由 JavaScript 程序的完全替代品或者修改版本。）Greasemonkey 几乎可以实现这一点，但仍稍有欠缺，由于它不能完全保证自己能够在网页中的 JavaScript 程序开始被执行之前完成对其代码的修改。使用本地代理能够解决这个问题，但其便捷程度还远未达到能够使其成为一种真正的解决方案的要求。我们需要构建一种兼具可靠性与便捷性的解决方案，以及能够用于分享这些更改的网站。GNU 计划只会推荐那些专注于自由更改的网站。

这些特性将会使得网页中的 JavaScript 程序在事实上有可能成为自由的。JavaScript 将不再是实现我们的自由的一种特别的障碍——现在只有 C 语言和 Java 是这样的。我们将能够拒绝甚至替换网页中的非平凡非自由 JavaScript 程序，正如我们能够拒绝或替换通过常规方式提供安装的私有软件那样。我们要求网站将其 JavaScript 代码自由化的运动就可以更好地展开。

与此同时，确实有一种情况是允许运行非自由 JavaScript 程序的：向网站维护者发送投诉消息，告诉他们应该移除其网站中的 JavaScript 代码或使其成为自由的。不要犹豫，马上暂时允许 JavaScript 以便进行投诉——然后记得重新关闭 JavaScript。

## 附件 A：用于发布自由 JavaScript 程序的一种常用约定

如需引用相关程序的源代码，我们建议采用这种格式：

[1] LibreJS 工程 (http://gnu.org/software/librejs/) 需要 JavaScript 程序员（的帮助）。如果您掌握了必备的技能，请帮助我们维护这个价值连城的浏览器扩展。

```
// @source:
```

后面跟着 URL。这种风格满足 GNU 通用公共许可证（GNU GPL）用于发布源代码的要求。如果源代码位于另一站点，您必须妥善处理这种情形。要想使软件成为自由的，源代码是必须提供的。

如需指出嵌入本页面的 JavaScript 代码所使用的许可证类型，我们建议将许可证文本置于如下所示的两段文字之间：

```
The following is the entire license notice for
the JavaScript code in this page.
...
The above is the entire license notice for the
JavaScript code in this page.
```

当然，所有这些内容应当位于多行注释之间。

同许多其他自由软件许可证一样，GNU GPL 要求连同程序的源代码和二进制形式一起发布一份许可证的副本。然而，由于 GNU GPL 过于冗长，将其同 JavaScript 程序一起包含在页面中可能会带来某种不便。对于您所拥有著作权的代码，您可以移除这条要求，代之以类似下面这样的许可证声明：

```
        Copyright (C) YYYY  Developer

The JavaScript code in this page is free software:
you can redistribute it and/or modify it under the
terms of the GNU General Public License (GNU GPL)
as published by the Free Software Foundation, either
version 3 of the License, or (at your option) any
```

```
later version. The code is distributed WITHOUT ANY
WARRANTY; without even the implied warranty of MERC-
HANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See
the GNU GPL for more details.

As additional permission under GNU GPL version 3 sec-
tion 7, you may distribute non-source (e.g.,minimized
or compacted) forms of that code without the copy of
the GNU GPL normally required by section 4, provided
you include this license notice and a URL through which
recipients can acc ess the Corresponding Source.
```

感谢 Jaffar Rumith 提醒我关注这一问题。

## 附件 B：以网络管理员身份发布自由的 JavaScript 程序

如果您是一位网络管理员并且在您的网站上部署自由的 JavaScript 软件，明晰并且一致地发布这些程序文件的许可证信息和源代码有助于您的网络访问者确定他们正在运行自由软件，并且有助于您指导您自己遵守这些许可条款。

发布这些许可证信息的一种方式如上文中的附件 A。另一种方法，称为 JavaScript 许可证网络标签，可能对于精简的 JavaScript 代码更为便捷，特别是对于那些并非您编写的代码。

# 如果您在大学工作，请发布自由软件



在自由软件运动中，我们坚信用户应当拥有对他们所使用的软件进行修改和再分发的自由。自由软件中的 “free” 是指 “freedom” 中的自由：即用户拥有运行、修改和再分发的自由。自由软件能够为人类知识做贡献，而私有软件则不能。因此，大学应当以推进人类知识发展之名鼓励自由软件，如同它们应当鼓励科学家和其他学者发表其工作成果。

呜呼，很多大学的管理人员对软件（以及对科学）持有一种贪婪的态度；他们将软件视为赢利的机会而非为人类知识做贡献的机会。自由软件开发者同这种倾向进行斗争已有将近 20 年了。

在 1984 年，当我开始着手开发 GNU 操作系统时，我的第一步是先辞去我在麻省理工学院（MIT）的工作。我特意选择这么做，这样 MIT 许可证机构就不能再对我将 GNU 作为自由软件发布这一行为进行干涉。我计划了这样一种方式对 GNU 中的程序进行授权许可，这种方式可以保证它的所有修改版本必须仍是自由的——这种授权许可方式后来发展成为了 GNU 通用公共许可证（GNU GPL）——而且我不希望乞求 MIT 管理人员的允许才能使用它。

这些年来，一些大学的分支机构经常前来自由软件基金会（FSF）寻求帮助，关于如何应对那些仅仅将软件视为用于贩卖之物的管理人员。有一种方法甚至对于那些受到特别资助的项目也是适用的，就是使您的工作基于一款已经在 GNU GPL 下发布的现有自由软件。然后您可以告知管理人员：“我们不能发布此修改版本，除非使用 GNU GPL 许可证——任何其他方式都将侵犯版权。” 当他们眼中的美元符号逐渐褪色时，他们通常将会不再反对将其作为自由软件发布。

您也可以求助于资金赞助商。当纽约大学（NYU）的一个开发小组在美国空军资助下开发出 GNU Ada 编译器时，合同中明确提出将其代码捐赠自由软件基金会。首先同赞助商达成某种默契，然后礼貌地对大学管理

人员说明这不允许重新谈判。他们将会想要签订一纸合同来开发自由软件，而不想失去它，于是他们很可能会同意。

不论您采取哪种方式，一定要尽早提出关于自由软件的问题——例如程序开发进程达到一半以前。此时，大学仍然需要您的工作，因此您可以采取强硬措施：告知管理人员您将会完成该程序并且使之可用，仅当他们白纸黑字地同意使其成为自由软件（并且同意由您选择使用何种自由软件许可证）。否则您为这个项目所能继续做的只是写一篇关于它的论文，并且不会做出一个适合于发布的良好版本。当管理人员意识到他们只能在得到一个可以为大学争得荣誉的自由软件与什么也得不到之间二选一，他们通常会选择前者。

FSF 可能能够说服您的大学接受 GNU GPL，或者 GPL 版本 3。如果您难以独立完成这一任务，请给我们帮忙的机会。请发送邮件至 licensing@fsf.org，并且在主题中加上“紧急”（“urgent”）标记。

并非所有大学都实行贪婪政策。德州大学有一项宽松的政策使得在那里开发的软件在 GNU GPL 下作为自由软件发布变得容易。巴西 Univates 大学和印度海德拉巴信息技术国际学院都有有利于以 GPL 发布软件的政策。在首先得到了开发团队的支持后，您也可以在您的大学实行类似的政策。将这个问题作为一种原则性的问题提出：大学是否应该以推进人类知识发展为己任，还是仅仅满足于使自己长期存在？

在试图说服大学的过程中，最好能够从伦理的视角出发并且下定决心，如同我们在自由软件运动中所做的那样。为了能够以符合伦理的方式对待大众，软件应当是自由的——如同 “freedom” 中的自由——对于全体大众。

很多自由软件开发者狭隘地宣称如此做的一些实践上的原因：他们倡导允许他人分享和修改软件只是一种使得软件变得更加强大且可靠的权宜之计。如果是这样的价值观驱使您开发自由软件，也罢，感谢您和您的贡献。但是，这样的价值观并不能给予您坚定的立场，一旦大学管理人员威逼利诱您将您的软件变成私有的。

例如，他们可能会争论 “我们能够让它变得更加强大且可靠，只要我们能够得到钱”。这样的宣言最后也许能够实现，也许不能，但它很难在事

先被证伪。他们可能会提出这样一种分发副本的许可证：“免费，仅限学术应用”，这将会向普通大众暗示他们不应获得自由，他们还会论证这种行为将会得到学术界的支持，而这一切是（他们宣称）您所需要的。

如果您的出发点只是易用性的价值，这将很难说服人们拒绝这些没有出路的提议。而如果您的立场是基于伦理和政治方面的价值，就很容易做到。如果牺牲了用户的自由，一款软件做得再强大再可靠又有何益？难道自由不是应该普惠大众而不仅限于学术界内吗？如果自由和社区是您的目标，这些问题的答案是很明了的。自由软件尊重用户的自由，而私有软件否定用户的自由。

凡是能够使您坚定信心的，莫强如了解什么才是社区的自由所依赖的。举个例子，比如您。

# GNU/Linux 上带有数字限制管理（DRM）的非自由游戏：是好是坏？



Valve，一家知名的游戏公司，主要发布各种带有数字限制管理（DRM）的非自由计算机游戏，最近它宣称将会为 GNU/Linux 平台发布游戏。这会带来哪些好的或者坏的影响？

我假设流行的非自由软件在 GNU/Linux 上的可获得性，将会推动 GNU/Linux 操作系统的普及。然而，GNU 的终极目标并非仅限于获得“成功”；它的目的是赋予用户自由[1]。因此，更大的问题是，这种发展趋势将会怎样影响用户的自由。

这些游戏的问题并不在于它们是商业的[2]（我们不认为商业化有什么问题）。也不在于其开发者贩卖其副本[3]；这也没有问题。它们的真正问题在于这些游戏包含了非自由软件（当然，指的是“freedom”中的自由）[4]。

（同其他非自由软件一样）非自由游戏软件是不符合伦理的，由于它们拒绝用户的自由（游戏艺术是另一个问题，由于它们不是软件）。如果您渴求自由，一个起码条件是不要拥有或者在自己的计算机上运行非自由软件。这一点是很清楚的。

然而，如果您一定要玩这些游戏，您最好还是在 GNU/Linux 上进行而非在微软 Windows 上进行。至少您可以藉此避开 Windows 对您自由的践踏[5]。

因此，从直接的实践角度看，这种发展趋势既有好的影响又有坏的影响。这可能会鼓励 GNU/Linux 用户去安装这些游戏，也可能会鼓励这些游

[1] 参见《如今自由软件更加重要》 (p. 32) 一文以获得更多信息。

[2] 参见《应避免使用（或慎用）的词语》一文的“商业化”一节 (p. 110) 以获得关于“商业化”一词所带来的困惑的解释。

[3] 参见《售卖自由软件》 (p. 49) 一文以获得更多信息。

[4] 参见《什么是自由软件?》 (p. 1) 以获知自由软件的完整定义。

[5] 参见我们的运动位于 http://upgradefromwindows8.org/ 以获得更多信息。

戏的玩家使用 GNU/Linux 来取代 Windows。我猜想直接的好处大于直接的坏处。但是，这里还有一种间接的影响：这些游戏的玩家将会向我们社区里的人传授什么？

任何自带软件以提供这些游戏的 GNU/Linux 发行版将会暗示用户：自由并不是关键。GNU/Linux 发行版中的非自由软件[1]已经在破坏自由这一终极目标。向发行版中添加这类游戏将会加深这种危害。

自由软件的实质是自由而非价格。一款自由的游戏不必须是免费的。商业化开发自由的游戏软件是可行的，同时尊重您修改所使用软件的自由。由于游戏中的艺术不属于软件，它也不必须是免费的。事实上，确实有由商业公司开发自由的游戏软件，与那些由志愿者以非商业化方式开发的自由的游戏软件并存。众筹式的开发只会让开发变得更容易。

但是，如果我们假设在当前情况下开发某类自由游戏软件是不可行的——将会发生什么？将其作为一款非自由游戏软件开发并无任何益处。想要在您的计算中拥有自由，必须拒绝非自由软件，就是这么简单直白。您，作为热爱自由者，不会由于一款非自由游戏的存在而去玩它，因此，您也不会由于它的不存在而失去任何东西。

如果您想要推进计算中的自由事业，请注意不要将这类游戏在 GNU/Linux 上的可获得性说成是对我们事业的支持。与之相反，您可以向人们介绍 LibreGameWiki[2]，那里试图列出所有自由的游戏软件，以及 FreeGameDev 论坛[3]，还有 LibrePlanet Gaming Collective 的自由游戏之夜[4]。

[1]参见 http://gnu.org/distros/common-distros.html 以获知为何我们不推荐某些（通常是流行的）发行版。

[2]参见 https://libregamewiki.org/Main_Page.

[3]参见 http://forum.freegamedev.net/index.php.

[4]参见 http://libreplanet.org/wiki/Group:LibrePlanet_Gaming_Collective.

# 电子书的威胁



在这样一个由企业支配我们的政府并且制定法律的时代，任何技术上的进展都会为企业提供对公众施加新的限制条件的机会。本应赋予我们权利的技术现在被用于束缚我们。

对于纸版书：

- 您可以用现金匿名购买；
- 然后您拥有它；
- 您不需要签订限制您使用它的霸王条款；
- 它的格式是已知的，不需要借助私有技术就能阅读；
- 您可以赠送、借阅或转卖给其他人；
- 您可以物理地扫描或者复制整本书，有时这也是符合版权法的；
- 任何其他人无权销毁您的书。

与之相反，对于亚马逊电子书（比较普遍）：

- 亚马逊要求用户提供身份信息才能购书；
- 在某些国家，包括美国，亚马逊宣称用户不能拥有其发行的电子书；
- 亚马逊要求用户接受限制其对电子书使用的霸王条款；
- 电子书的格式是私密的，必须借助限制用户自由的私有软件才能阅读；
- 在一段限定时间内，代用词“借阅”被允许用于某些电子书，但仅限使用相同系统的实名用户。禁止赠与或转卖。

- 在阅读器中复制电子书是不可能的，由于数字限制管理（DRM[1]），这也被霸王条款所禁止，它比版权法更加苛刻。
- 亚马逊可以利用后门远程删除电子书，例如它曾于 2009 年利用此后门远程删除了数千份乔治 · 奥威尔（George Orwell）所著的《一九八四》。

这些侵害行为只需其一便可使得电子书相对于纸版书成为一种倒退。我们必须拒绝电子书，直到他们尊重我们的自由。

电子书公司宣称它们拒绝我们的传统阅读自由是必需的，唯有如此它们才能继续向作者支付稿酬。现行的版权系统对这些公司给予了慷慨的支持，但却未能给予大多数作者起码应有的支持。我们可以用其他方式对作者提供更好的支持，而无需牺牲自己的自由，甚至使分享合法化。我在此建议两种方式：

- 根据每位作者受欢迎程度的立方根[2]将税金分发给作者；
- 开发出这样的阅读器，使得用户可以向作者匿名捐款。

电子书不应该侵害我们的自由（古腾堡计划的电子书不会这样），但是，一旦商业公司决定如此做，它们就会侵害我们的自由。我们有责任阻止这种行为。

加入这场斗争，在这里签名：http://DefectiveByDesign.org/ebooks.html。

[1] 参见《阅读的权利》 (p. 129) 一文以获得更多信息。

[2] 参见《计算机网络时代的版权与社区之争》 (p. 154) 一文和我在 2012 年致巴西参议院主席 Senator José Sarney 的公开信，位于 https://stallman.org/articles/internet-sharing-license.en.html，以获得更多信息。

# 电子书必须增进我们的自由而非限制我们的自由



我喜欢 The Jehovah Contract 这本书，也希望每个人都喜欢它。这些年来，我已经至少借出它六次了。纸版书允许我们这么做。

我不能对大部分商业电子书如此，因为这是 “不被允许的”。如果我们试图违反，阅读器中的软件具有一种称为数字限制管理（DRM）的恶意功能，它将会限制用户阅读，因此不能分享。由于电子书是加密的，只有具有恶意功能的私有软件才能显示它们。

我们作为读者所习惯的很多其他阅读方式对于电子书是 “不被允许的”。对于亚马逊 Kindle（更合适它的名字是 Swindle [1]），作为一个例子，用户不能使用现金匿名购买电子书。Kindle 的电子书通常仅可以从亚马逊获得，而亚马逊要求用户提供身份信息。由此亚马逊知道每位用户读了哪些书。在某些国家，例如英国，您可能会由于拥有一本禁书而被起诉[2]，这比假想中的奥威尔主义（Orwellian）[3]更加残酷。

> 参阅《电子书的威胁》 (p. 297) 一文，并且考虑加入我们关于电子书威胁的邮件列表，位于 http://defectivebydesign.org/ebooks.html.

更坏的是，您不能在读完一部电子书之后将其转卖（如果亚马逊按其

[1] 参见《为何称之为诈骗（Swindle）?》 (p. 104) 一文以获得更多信息。

[2] Ben Quinn, “Man in London Charged with Terrorism Offences over Al-Qaida Document,” 4 April 2012, http://www.theguardian.com/world/2012/apr/04/al-qaida-terrorism.

[3] 指现代专制政权借由严厉执行政治宣传、监视、故意提供虚假资料、否认事实（双重思想）和操纵过去（包括制造 “非人”，意指把一个人过去的存在从公共记录和记忆中消除）的政策以控制社会，就如《一九八四》（作者乔治 · 奥威尔）中的世界观一样。——译者注，摘自维基百科

意志行事，那些我可以在一个下午光顾好几家的旧书店将成为历史）。您也不能将其赠与他人，由于根据亚马逊的霸王条款，您不能真正拥有它。亚马逊要求用户签订的最终用户许可协议（EULA）如是说。

您甚至不能保证某部电子书明天还存在于您的机器中。人们在使用 Kindle 阅读《一九八四》的时候亲历了奥威尔主义：他们的电子书在眼皮底下消失了，由于亚马逊使用了一种带有后门的恶意软件将其远程删除（虚拟世界的焚书行为，这就是 Kindle 的涵义吗?）。但是请不必担心，亚马逊承诺不再如此，除非有国家命令。

对于软件，要么用户控制程序（称这类软件为自由的[1]），要么程序控制用户（非自由）。亚马逊的电子书政策模仿私有软件的发布政策，但这并非二者之间的唯一关联。上述恶意软件特性[2]是通过私有软件强加给用户的。如果一款自由软件拥有类似的恶意特性，一些具有编程技能的用户可以移除它们，然后将修正后的版本提供给其他用户。用户不能更改私有软件，这使得它们成为其开发者对公众行使不公权力的理想工具[3]。

以上这些对我们自由的侵害任何一种都足以作为我们说 “不” 的理由。如果这些政策仅限于亚马逊，我们可以跳过它们，但是其他电子书发行商的政策大致相似。

最让我们担心的是未来可能失去纸版书的选项。英国卫报宣称 “仅限数字阅读”，换言之，书籍只能靠牺牲自由去换取。我不会以此为代价阅读任何书籍。从现在起的五年以后，对于大部分书籍，未经许可的副本会不会成为仅存的伦理上可接受的版本呢?

事情不必须朝着那个方向发展。通过互联网匿名支付，购买并下载无 DRM、无 EULA 的电子书将会尊重我们的自由。实体书店可以接受现金购买这样的电子书，如同光盘（CD）上的数字音乐仍然可能买到，即使是现

[1] 参见《什么是自由软件?》 (p. 1) 一文以获知自由软件的完整定义。

[2] 参见 http://gnu.org/proprietary/proprietary.html 以获知一系列日益增长的威胁。

[3] 参见我的文章《如今自由软件更加重要》 (p. 32) 和 “The Problem Is Software Controlled by Its Developer,” 位于 http://gnu.org/philosophy/the-root-of-this-problem.html 以获得更多信息。

在这个音乐产业极具侵略性地推广带有 DRM 限制服务（诸如 Spotify）的时代。实体 CD 商店不得不面对昂贵的库存成本负担，而实体电子书商店可以将电子书副本复制到您的 USB 存储器上，这是唯一可以卖给您的载体，如果您需要。

出版商关于它们限制电子书使用自由所给出的解释是为了阻止用户分享电子书副本。它们宣称这是为了作者着想；但即使它们真的服务于作者的利益（对于非常著名的作者，也许是这样），这并不能成为剥夺读者自由的 DRM、EULA 和数字经济法案的辩护理由。事实上，版权系统对于支持作者，除了那些最著名的作者以外，做出了极坏的事情。其他作者的基本利益应当被更好地为人们所了解，因此分享他们的作品对于他们和读者同样有益。为何不转向一种能够更好地解决这个问题而又与分享相容的体系呢?

对存储器和互联网连接征税，与大多数欧盟国家正在实施的一般准则放在一起，可以更好地完成这项任务，如果以下三点得到满足。这笔资金应该由国家收集并依法发放，而非给予私人的集资组织；它应当分给所有作者，我们不能允许商业公司侵占其中的任何一分钱；并且这笔钱的分配应当基于一种变化的尺度而非按照流行度的线性比例。我建议使用作者流行度的立方根作为依据：如果 A 的流行度是 B 的八倍，则 A 得到 B 的两倍的钱（而不是八倍）。这将给予众多普通作家足够的支持，而非仅仅使得少数明星大腕赚得盆满钵满。

另一种体系是在阅读器上设置一个按钮以便向作者匿名支付一小笔钱（例如在英国，25 便士可能是合适的）。

分享是美好的，并且有了数字技术，分享应该变得更容易。（我指对原始版本进行非商业的再分发）因此，分享理应合法，而阻止分享并不能成为将电子书变为读者数字手铐的理由。如果电子书意味着必须在增进与限制读者自由中二选一，我们必须要求它们增进读者的自由。

# 服务器真正是在为谁服务？


本文最初于 2010 年三月 8 日以标题 “What Does That Server Really Serve?” 发表于 Boston Review 在线版。

**在互联网环境中，私有软件并非剥夺您的自由的唯一方式。“作为软件替代品的服务”（Service as a Software Substitute），或者 SaaSS，是另一种赋予其他人支配您的计算的权力的途径。**

SaaSS 指的是使用某种由其他人实现的服务来替代您所运行的软件副本。这是我们创造的短语，文章和广告并不会使用它，它们也不会告知您某种服务是否为 SaaSS。与之相反，它们很可能蔽之以一种含混不清并且旨在转移您的注意力的短语：云。这将 SaaSS 与其他实现，不论是恶意的还是有益的，混为一谈。不过，看了本文的解释和示例之后，您将能够区分某种服务是否为 SaaSS。

## 背景知识：私有软件是如何剥夺您的自由的

数字技术可以赋予您自由，也可以剥夺您的自由。对于我们掌控自己计算的首要威胁来自*私有软件*：私有软件是用户所不能控制的，由于它受其拥有者（例如苹果或微软这样的商业公司）控制。其拥有者通常利用这种不公的权力向其中植入恶意功能，诸如间谍软件、后门、数字限制管理（DRM，它们将其鼓吹为数字版权管理）[1]。

> 如需获得关于此问题的更多信息，参见 “The Bug Nobody Is Allowed to Understand,” 一文，它位于 http://gnu.org/philosophy/bug-nobody-allowed-to-understand.html。

我们针对这个问题的解决方案是开发*自由*软件并且拒绝私有软件。自由软件意味着您作为用户拥有四项基本自由：（0）以任何目的运行该软件

[1] 请加入我们的运动一起反对 DRM，它位于 DefectiveByDesign.org。

的自由；（1）研究和修改源代码使其以您所期望的方式工作的自由；（2）再发布相同副本的自由；（3）再发布您的修改版本的自由（参见《什么是自由软件?》 (p. 1) 一文中有关自由软件定义）。

只要有了自由软件，我们作为用户就能夺回我们计算的控制权。私有软件仍然存在，但我们可以将其排除在生活之外，并且我们中的很多人已经在这么做。然而现在，我们对自己的计算的控制权面临新的威胁：作为软件替代品的服务（SaaSS）。为了捍卫我们的自由，我们必须像拒绝私有软件那样拒绝 SaaSS。

## 作为软件替代品的服务是如何剥夺您的自由的

作为软件替代品的服务（SaaSS）指的是使用某种服务作为运行您的软件副本的替代品。具体地说，它指的是由某人架设一台网络服务器以实现某种计算任务——例如，修改一幅照片，或者将文本翻译为另一种语言等——然后邀请用户通过那台服务器进行计算。服务器的用户会将其本人的数据上传至服务器，后者对用户所上传的数据进行*用户自己的计算*，再将计算结果返回用户，或者直接以用户的名义做什么事情。

假设满足这样的条件，这种计算可以称之为*用户自己的*，即用户原则上能够在其本人的计算机上通过运行某软件可以达到相同的目的（不论当时能够达到此目的的软件对于用户是否可获得）。当这种假设不成立时，这就不是 SaaSS。

这些服务器比私有软件更加残酷地剥夺用户对其计算的控制权。对于私有软件，用户一般会得到一份可执行文件而非源代码。这使得用户难以研究其所运行的代码，因而难以判定此程序真正在做什么，并且难以对其进行修改。

对于 SaaSS，用户连用于计算的可执行文件都得不到：它位于某些其他人的服务器上，用户不能企及。因此用户完全不可能确认它们到底在做什么，也完全不可能对其进行修改。

进一步讲，SaaSS 将会自然而然地引起与某些私有软件中的恶意功能相当的结果。

例如，一些私有软件是“间谍软件”：它们将会发送与用户的计算活动相关的数据[1]。微软 Windows 向微软发送用户活动相关信息。微软 Windows 媒体播放器（Windows Media Player）向微软汇报每位用户播放过的内容。亚马逊 Kindle 汇报用户于何时阅读哪本书的第几页。愤怒的小鸟（Angry Birds）将会汇报用户地理位置的历史记录。

与私有软件不同的是，SaaSS 并不依赖隐秘的代码以获取用户数据。而是用户被强制要求将其数据上传到服务器以便使用 SaaSS。这与间谍软件具有相同的效果：服务器运营商可以不费吹灰之力得到用户数据——正因为 SaaSS 的天性。Amy Webb 曾经刻意避免发布她女儿的任何照片，但她犯了使用 SaaSS（Instagram）的错误，用它来编辑自己的照片。这些照片最终通过这条途径泄露[2]。

一些私有操作系统拥有通用的后门，允许某人远程进行软件更改。例如，微软可以通过 Windows 的通用后门强制更改用户计算机上的软件。此类后门也存在于几乎所有的移动电话中。一些私有应用程序也拥有通用后门，例如 Steam 的 GNU/Linux 客户端允许其开发者远程安装修改版本。

对于 SaaSS，服务器的运营商可以更改服务器上所用的软件。他应该可以这么做，因为那是他自己的计算机；然而，使用这样的 SaaSS 与使用带有通用后门的私有应用程序，在结果上并无区别：某人拥有偷偷改变如何完成用户计算的权利。

因此，使用 SaaSS 相当于使用带有间谍软件和通用后门的私有应用程序，这赋予了其服务器运营商凌驾于用户之上的不公权力，这种权力是我们必须坚决反对的。

[1]如需获知监控正在通过哪些日益增加的方式在业界蔓延，参见 http://gnu.org/philosophy/proprietary/proprietary-surveillance.html。

[2]Amy Webb, “Congratulations, You Found a Photo of My Daughter Online,” 12 September 2013, http://slate.com/articles/technology/data_mine_1/2013/09/privacy_facebook_kids_don_t_post_photos_of_your_kids_on_social_media.html.

## SaaSS 和 SaaS

最初，我们将这种有问题的实践称为“SaaS”，意为 Software as a Service（软件即服务）。这个短语通常用于指代在服务器上部署软件而非向其用户提供副本。我们当时认为，在这些问题发生时，这一短语对实际情况做出了精确描述。

然后，我们开始意识到，短语 SaaS 有时用于通讯服务——而这一名称并不适用于这些活动。此外，短语“软件即服务”并不能解释为何这是一种坏的实践，于是我们打造了这一短语“作为软件替代品的服务”，这一称谓对这种坏的实践给出了更为清晰的定义，并且解释了它为什么是坏的。

## 理清 SaaSS 带来的问题和私有软件带来的问题

SaaSS 和私有软件将会带来相似的后果，但二者的作用机制并不相同。对于私有软件，其机制是您拥有一份副本，但对其进行修改是困难或者非法的。而对于 SaaSS，其机制是您不能拥有一份您所用于计算的软件副本。

这两类问题通常被混淆，而混淆不仅仅是一种偶然。网络开发者使用含混不清的短语“网页应用”将服务器软件和您在计算机浏览器中运行的程序混为一谈。一些网页会在您的浏览器中安装非平凡甚至是大型的 JavaScript 程序而不会告知您。如果这些 JavaScript 程序是私有的[1]，它们将会造成与任何其他私有软件类似的不公。然而，我们在此将会主要讨论使用服务本身带来的问题。

很多自由软件支持者设想：SaaSS 带来的问题将可以通过开发自由的服务器软件来解决。对于服务器运营商而言，其服务器上的软件最好是自由的；否则如果它们是私有的，其拥有者便可拥有控制服务器的权力。这对于服务器运营商是不公的，并且根本不会使用户受益。但是，即使服务器上运行的软件是自由的，这也不能保护服务器的用户免受 SaaSS 的影响。这些软件赋予了服务器运营商自由，但却不能赋予服务器用户自由。

公开发布服务器软件的源代码确实会使社区受益：这使得具有适当技

[1]参见《JavaScript 陷阱》 (p. 286) 一文以获得更多信息。

能的用户能够架设类似的服务器，也许还能更改软件本身。我们建议对通常在服务器上运行的软件使用 GNU Affero 通用公共许可证（GNU Affero GPL）[1]。

但是，这些服务器都不能使您能够控制由它进行的计算，除非那是您自己的服务器。也许，对于某些工作，您可以信任您的朋友的服务器，如同您可能会请求朋友维护您计算机上的软件。对于除此之外的任何情况，这些服务器对您来说都是 SaaSS。SaaSS 总是迫使您屈从于服务器运营商的权力。唯一的解决方式是：*坚决不要使用 SaaSS*！不要使用任何其他人的服务器来进行您自己的计算以处理由您自己提供的数据。

这个问题解释了“开放”和“自由”之间有着多大程度的区别。“开源”概念中的源代码几乎都是自由的[2]。但是“开源软件”服务[3]的理念并不能解决 SaaSS 的问题，由于它只意味着这种服务所依赖的服务器软件是开源的，并且/或者是自由的。

服务与软件有着本质区别，因而由服务引起的伦理问题与那些由软件引起的问题有着本质区别。为了避免这种混淆，我们避免将一项服务描述为“自由”或“私有”的[4]。

## 区分 SaaSS 和其他网络服务

哪些在线服务是 SaaSS？最清楚的例子是翻译服务，例如将英文文本翻译为西班牙文。为您翻译一段文本是一种完全属于您自己的计算。您可以在您自己的计算机上运行相应软件进行翻译（为了符合伦理，这款软件应该是自由的）。翻译服务替代了那款程序，因此它是一种作为软件替代

[1] 参见《如何为你的作品选择一份许可证》(p. 211) 一文以获得我们关于许可证选择的建议。

[2] 参见 “How Free Software and Open Source Relate as Categories of Programs,” http://gnu.org/philosophy/free-open-overlap.html 一文以获得更多信息。

[3] 如需获知 “Open Software Service Definition,” 参见 http://opendefinition.org/ossd/index.html 一文。

[4] 如需获得更多信息，参见我的文章 “Network Services Aren’t Free or Nonfree; They Raise Other Issues,” 位于 http://gnu.org/philosophy/network-services-arent-free-or-nonfree.html。

品的服务，即 SaaSS。由于它拒绝您对您的计算的控制权，它对您作了恶。

另一个清楚的例子是使用诸如 Flickr 或者 Instagram 来修改照片。人们在自己的计算机上修改照片已有几十年时间了；在服务器而非您自己的计算机上进行这些工作是一种 SaaSS。

拒绝 SaaSS 并不意味着必须拒绝使用任何由他人而非您所运行的网络服务器。大部分服务器并非 SaaSS，由于它们所进行计算并非用户自己的计算。

设立网络服务器的初衷并非为您做计算，而是发布信息以供您访问。即使在今天，这也是大部分网站正在做的，这并不会带来 SaaSS 的问题，由于访问由某人发布的信息并非是在做您自己的计算。通过博客网站或者诸如 Twitter 或 StatusNet 等微博服务发布您的个人材料也都不是（当然，这些服务可能会有其他问题）。其他并非有意成为私密的通讯，例如群聊，也是如此。

从本质上说，社交网络是一种形式的通讯和信息发布，并非 SaaSS。然而，一项以社交网络为主要功能的服务可能拥有属于 SaaSS 的特性或扩展。

如果一项服务不属于 SaaSS，并不意味着它就是没有问题的。服务可能存在其他伦理问题。例如，Facebook 以 Flash 格式发布视频，这将迫使用户运行私有软件；它要求运行私有 JavaScript 代码；它给用户一种关于隐私的错误印象，同时引诱用户在 Facebook 上暴露他们的生活。那些都是重要的问题，但不属于 SaaSS 的问题。

诸如搜索引擎的服务会从网络上搜集数据以供您检索。浏览它们所收集的数据在通常意义上并不属于您自己的计算——由于您并不提供那些数据——因此使用这样的服务检索互联网并非 SaaSS。然而，使用他人的服务器来实现用于您自己网站的搜索工具是一种 SaaSS。

在线购物不是 SaaSS，由于这种计算不是您自己的；相反，这是由您和卖家为了共同的目的而共同实现的。在线购物的真正问题是您能否将您的钱财和其他个人信息（首当其冲的是您的名字）托付给对方。

仓库网站诸如 Savannah 或 SourceForge 本质上不是 SaaSS，由于它们

的职责是发布那些提供给它的数据。

使用联合项目的服务器不是 SaaSS，由于您以这种方式所做的计算不是您自己的。例如，如果您编辑 Wikipedia 上的页面，您并非在进行您自己的计算；与之相反，您正在参与 Wikipedia 的计算。Wikipedia 控制其自己的服务器，但是如果任何个人或组织使用他人的服务器进行他们自己的计算，他们将会遇到 SaaSS 问题。

有些网站提供多种服务，如果其中之一不是 SaaSS，其他服务有可能是 SaaSS。例如，Facebook 的主要服务是社交网络，那不是 SaaSS；然而，它提供了第三方应用，其中的某些是 SaaSS。Flickr 的主要服务是发布照片，这不是 SaaSS，但它也会提供照片编辑功能，这就是 SaaSS。类似地，使用 Instagram 发布这些照片并非 SaaSS，但使用它转换照片则是。

Google Docs 向我们展示要评估单一服务到底可以有多么复杂。它邀请用户通过运行大量私有 JavaScript 程序[1]以编辑文档，这无疑是坏的。然而，它还提供了用于标准格式上传或下载文档的应用程序接口（API）。自由的文档编辑器可以通过这个 API 工作，这种应用场景不是 SaaSS，由于它仅仅把 Google Docs 用作仓库。将您的所有数据展示给一家商业公司不是好事，但这属于隐私范畴，而非 SaaSS；依赖这样的服务来访问您的数据不是好事，但这属于风险范畴，而非 SaaSS。另一方面，使用这种服务转换文档格式是一种 SaaSS，因为您可以通过在您的计算机上运行适当的软件（最好是自由的）来实现这一目的。

当然，通过自由的文档编辑器使用 Google Docs 的情形并不常见。更为常见的情形是，人们通过私有 JavaScript 程序使用它，这类私有 JavaScript 程序和任何其他私有软件一样坏。而这种应用场景也可能涉及 SaaSS；这取决于文档编辑工作的哪些部分由 JavaScript 程序完成而哪些部分由服务器完成。我们并不清楚这个问题的答案，但既然 SaaSS 和私有软件都以相似的方式对用户作恶，我们并不急迫地想知道这个问题的答案。

通过他人的仓库发布信息并不会引起隐私泄露的问题，但是通过 Google Docs 发布则会遇到一个特别的问题：如果不借助运行私有

[1]参见《JavaScript 陷阱》 (p. 286) 一文以获得更多信息。

JavaScript 代码，甚至不可能在浏览器中显示一段 *Google Docs* 中的文本。因此，您应该避免使用 Google Docs 发布任何信息——然而其原因不属于 SaaSS 范畴。

信息技术（IT）业界试图阻止用户作出这样的区分，这正是他们用行话“云计算”来表达的用意所在。这个短语是那么地空泛，以至于它可以指代有关互联网的几乎所有应用。它包括了 SaaSS 以及许多其他网络应用实践。在任何给定的上下文语境中，一位作者（如果是一位技术工作者）在写出“云”这个词的时候，他的脑海中可能确实有一种确切的涵义，但通常并不会指出这个词在其他文章中可能具有的其他确切的涵义。这个短语将会诱导用户将那些他们本来应该分开单独考虑的实践概念一般化。

如果“云计算”确实有一种涵义，这种涵义并不是指一种进行计算的方式，与之相反，它应该指的是一种对计算的思考方式：一种不顾一切的极端思考方式可能会主张：“什么也不要问；不要关心谁控制您的计算，也不要关心谁掌控您的数据；在像鱼饵一样吞下我们的服务之前也不要去检查里面是否藏着鱼钩……什么都不要犹豫，只要信任我们的商业公司就好。”换言之，就是“做一个没有独立思想的傀儡任人摆布”。思想中的阴云是进行清晰思考的障碍。为了能够理清关于计算的思考，必须避免使用“云”这个概念。

## 应对 SaaSS 所带来的问题

只有少数的网站提供 SaaSS；大部分并不会带来这样的问题。但是，对于那些带来了 SaaSS 问题的网站，我们又该如何应对？

对于简单的情形，如果您使用您的双手进行您的计算，解决方案很简单：使用您自己的自由软件副本。例如编辑文本时，使用您自己的自由文本编辑器，比如 GNU Emacs，或者自由的文字处理器的副本。编辑照片时，使用您自己的自由软件的副本，比如 GIMP。如果没有可用的自由软件又当如何呢？由于私有软件和 SaaSS 都会剥夺您的自由，您不应当使用它们。您可以贡献您的时间或钱财来支持开发自由软件替代品。

与他人一起作为一个小组进行协作的情况又如何呢？直到现在，不借

助服务器可能难以实现这一目的，并且您的用户组可能不知道如何架设自己的服务器。如果您使用他人的服务器，至少不要信任由商业公司运营的服务器。对于消费者而言，仅凭一纸合同并不能带来任何保护，除非您能够抓住其中的漏洞并且真正能够进行诉讼。即使如此，商业公司很可能在其编写的合同条款中允许其在较宽的范围内恣意妄为。假如那些公司不像之前那些为布什实施非法监听其客户的美国通讯公司那样主动就范，国家仍然可以从商业公司那里传唤您和任何其他人的数据，如同奥巴马对通讯公司所做的。因此，如果您不得不使用服务器，选择那些运营商能够给予您信任基础而非仅仅是一层商业关系的服务器。

然而，在长远的时间尺度上，我们可以开发出替代服务器的解决方案。例如，我们可以开发一种端对端的程序使得协作者可以分享加密数据。自由软件社区应该试图为关键的 “网页应用” 开发出端对端的分布式替代品。并且将它们在 GNU Affero GPL 许可证下发布可能是明智的，由于它们可以作为候选者供其他人转换为基于服务器的软件[1]。GNU 计划正在寻找志愿者以开发这些替代品。我们可以邀请其他自由软件项目在它们进行设计时考虑这一问题。

与此同时，如果一家商业公司邀请您使用它的服务器来进行您自己的计算任务，不要屈服；也不要使用 SaaSS。不要购买或安装所谓的 “瘦客户机”，它们只不过是厂商为了使您在其服务器上进行真实工作而提供的低性能计算机，除非您将会在您自己的服务器上使用它们。使用一台真实的计算机来工作并且保存您的数据。使用您自己的自由软件副本来进行您自己的计算，这是为了您的自由。

[1]参见 “Why the Affero GPL” http://gnu.org/licenses/why-affero-gpl.html 以获得完整解释。

# 第 7 部分

# 珍视社区和你的自由

## 避免破坏性的妥协



1983 年 9 月 27 日，我发起了一项计划，致力于创造一个完全自由的操作系统，它称作 GNU——意为 GNU 不是 Unix（GNU's Not Unix）。为了纪念 GNU 操作系统 25 周年，我写出了这篇文章，旨在阐述我们的社区怎样才能避开破坏性的妥协。除了避开这类妥协之外，您还可通过很多其他方式帮助 GNU 和自由软件。其中一条基本方式是以合作成员的身份加入自由软件基金会（FSF）[1]。

在生活中的另一个领域存在这样一种类似的观点，例如文章 "'Nudge' Is Not Enough, It's True. But We Already Knew That" http://guardian.co.uk/commentisfree/2011/jul/19/nudge-

[1]您可以在此通过加入会员的方式支持自由软件基金会：http://my.fsf.org/join.

is-not-enough-behaviour-change（Jonathan Rowson，2011 年 7 月 19 日）所指出的观点：除了温和的劝说，我们还需要态度和视角方面的转变。

自由软件运动致力于带来一场社会变革：让所有软件成为自由的[1]，这样所有软件用户将获得自由，并且将会成为协作化社区的一部分。与之相反，任何一个非自由的程序赋予了其开发者凌驾于用户之上的不公权力。我们的目标是终结这种不公。

自由之路，路漫漫兮[2]。这需要经历很多步骤，花费很多年的时间才能达到这样一种境地：软件用户拥有自由成为一种常态。这些步骤当中的一些可能会非常艰难，甚至需要为之付出一些牺牲。而其中有些步骤可能会变得简单，如果我们与其他拥有不同最终目标的人们达成某种妥协。

因此，自由软件基金会做出了一些妥协——甚至是重大妥协。例如，我们在 GNU 通用公共许可证（GNU GPL）第三版的专利提供条款中做出了妥协，使得主要的商业软件公司可以贡献并发布基于 GPL v3 的软件，并藉此将部分软件专利置于此条款的效力之下。

GNU 宽通用公共许可证（GNU LGPL）的目的在于一种妥协，我们将它用于一些选定的自由程序库，以便允许将它们用于非自由软件中。这是由于我们认为如果在法律层面禁止这样做只会迫使开发者转向使用私有程序库。我们在 GNU 软件中接受并安装某种代码使得它们能够与常见的非自由软件协同工作，并且我们对这种行为以这样的方式进行描述和宣传，鼓励后者的用户安装并使用 GNU 软件而不是相反。我们也会支持我们所认同的一些特定的运动，即使我们并不完全认同那些支持这些运动的组织。

但是，我们会坚决拒绝某些妥协，即使我们的社区中的很多其他人愿意做出这样的妥协。例如，我们仅仅支持并推广有这样的发行政策的

[1]这里的“自由”应理解为 freedom 中的自由，参见《什么是自由软件?》 (p. 1) 一文以获得自由软件的完整定义。

[2]参见自由软件基金会执行董事 John Sullivan 于 2008 年发表的文章：“The Last Mile Is Always the Hardest,”，位于 http://fsf.org/bulletin/2008/spring/the-last-mile-is-always-the-hardest，所提及的他本人对于这一问题的看法。

GNU/Linux 发行版[1]：它们不会在其中包含非自由软件，也不会引导用户安装非自由软件。支持并推广非自由的发行版将会是一种破坏性的妥协。

我们称某种妥协是破坏性的，如果从长期来看，它们将会对实现我们的终极目标起到反作用。这种破坏性的妥协可以发生在想法的层面，也可以发生在行动的层面。

在想法层面，破坏性的妥协将会增强而非削弱那些我们所致力于改变的不利条件。我们的最终目的是实现这样一种境地，即软件用户成为自由的。但是时至今日，大部分计算机用户甚至不认可“自由”是一件重要的事。他们仅仅重视“消费者”价值。也就是说，他们评估任何软件时，仅仅基于实践上的特性，诸如价格和易用性。

戴尔·卡耐基的经典自立书籍《怎样赢得朋友并且影响人们》（How to Win Friends and Influence People）所建议的说服某人做某事最有效的方式是提供那些满足对方价值观的论证。我们可以通过多种方式满足我们社会中典型的消费者价值观。很多自由软件也是兼具易用性与可靠性的。通过援引这些实践层面的好处，我们已经成功地说服诸多用户采纳多种自由软件，其中的一些已经大获成功。

但是，如果长远目标仅仅限于使更多的人使用某些自由软件，您可能会选择对“自由”这一概念保持沉默，并且将目光仅仅集中在那些实践上对消费者价值有意义的好处上。而这正是“开源”这一概念以及与之关联的一些术语所处的状态。

然而，这样一条路径只能将我们带到通往自由的终极目标的半路上。如果人们仅仅因为易用性而去使用某个自由软件，那么这些人将会仅仅在这款自由软件仍然易用的情况下继续坚持使用它。接下来，他们就再也找不到理由不去使用与这款自由软件共存但更加易用的私有软件。

“开源”的哲学满足于消费者价值并且以此为前提，肯定并且强化这些消费者价值，这就是我们不能支持“开源”的原因。

为了完全地、长久地建立一个自由社区，我们要做的绝不仅限于让人

[1]您可以在这里找到自由操作系统发行版准则（GNU FSDG）：http://gnu.org/philosophy/free-system-distribution-guidelines.html。

们使用某些自由软件。我们需要宣传按照公民价值评估软件（及其他事物）的理念，即基于它们是否尊重用户的自由和社区，而非仅限于所谓的“易用性”。这样人们就不会落入以吸引人、易用的特性作为诱饵的私有软件的陷阱之中。

为了推广公民价值，我们必须讨论它们并且说明它们怎样才能成为我们的行为准则。我们必须拒绝戴尔 • 卡耐基式的妥协，因为这将通过强调人们的消费者价值而影响他们的行为。

不过，这并不意味着我们完全不能援引实践上的好处——我们可以适度地这样做，并且我们也确实正在这样做。只有当这种实践上的好处占据了用户的注意力并且让“自由”沦为背景时，这才成为问题。因此，当我们援引自由软件实际的好处时，我们必须不厌其烦地反复强调这些实践上的好处只是用户应该倾向于使用自由软件*附加的、次要的*原因。

仅仅让我们的口号与我们的理念保持一致并不够，我们的行动也必须与理念保持一致。因此，我们必须避免做出这样的妥协，它们会做那些我们所致力于终结的事情，或者使之合法化。

例如，经验已经表明，您可以通过包含某些非自由软件以吸引某些用户使用 GNU/Linux。这可能是一个会吸引一些用户目光的漂亮的非自由的应用程序，或者一种非自由的编程平台，例如 Java[1]（曾经是这样）或者 Flash 运行时的环境（仍然是这样），或者某种用于支持特定硬件型号的非自由设备驱动程序。

这些妥协无疑是诱人的，但它们会在无形之中毁掉我们的终极目标。如果您发布非自由软件或者引导用户使用非自由软件，您将会觉得自己没有底气说出“非自由软件是一种不公，一种社会问题，我们必须终结它们”这样的话。以至于即使您嘴上能够继续这样说，您的行动也会渐渐破坏掉这些信念。

这里的问题不在于用户是否应该*能够*或者*被允许*安装非自由软件；一个通用目的的系统应当支持并且允许用户能够去做他们所想要去做的任何事情。问题在于我们是否应当引导用户投向非自由软件。用户独立自主地

[1]参见 http://gnu.org/philosophy/java-trap.html 获取更多细节。

去做什么事情是他们的责任，而我们为他们做了什么，以及我们引导他们去做什么则是我们的责任。我们无论如何不应该将用户导向私有软件，即使它看起来像是一种解决方案。因为私有软件正是问题的根源。

破坏性的妥协不仅仅是对他人的不良影响，它还会通过认知上的不一致扭曲您自己的价值观。如果您抱有某种价值观，但是您的行动表现出来的是与之相冲突的其他价值观，您很可能将会试图改变您的价值观或者行动，以便解决这种矛盾。因此，那些仅仅争论实践上的好处的，或者引导用户转向某些非自由软件的项目，几乎不可避免地羞于哪怕只是暗示“非自由软件是不符合伦理的”。对于它们的参与者，以及对于公众，它们只能强化消费者价值。我们必须拒绝这些妥协，如果我们想要坚持我们自己的价值观不动摇。

如果您想要转向自由软件并且是为了自由的终极目标而不会做出妥协，您可以查阅自由软件基金会的相关资源，它们位于 http://www.fsf.org/resources。这里列出了能够与自由软件协同工作的硬件和机器配置列表，可供安装的完全自由的 GNU/Linux 发行版的列表，以及能够在百分之百自由软件环境下工作的数千款自由软件包[1]。如果您想要帮助社区走在通往自由的正确道路上，很重要的一点就是公开支持公民价值。当人们正在讨论什么是好或者坏，或者要做什么事情的时候，您可以援引自由和社区的价值观并与他们争论。

一条能够让您更快前行的道路有时并不是一条更好的道路，如果它通往错误的目的地。为了成就一个充满雄心壮志的目标，做出一些妥协是必需的。但是，一定要警惕那些将您引离最终目标的妥协。

[1] 自由软件目录，位于 http://directory.fsf.org，这里列出了我们所知的所有自由软件。

# 克服社会惯性



自从 GNU 和 Linux 的结合使得自由地使用个人计算机（PC）成为可能已过去将近 20 年时间。我们也由此走过了一段漫长的历程。现在，您甚至可以从不止一家硬件供应商那里买到一台预装了 GNU/Linux 的便携计算机——尽管它自带的操作系统发行版可能不是完全由自由软件组成的。那么，究竟是什么阻止了我们获得完全的成功呢?

通往软件自由的胜利之路上的主要障碍是社会惯性。它有多种存在形式，而且您一定曾见识过它们当中的某些形式。例如某些设备只能在 Windows 操作系统下使用，以及某些商业网站只能在 Windows 下访问等。如果您相对于自由更加看重短期的易用性价值，您可能认为这些理由足以说服您使用 Windows。很多公司目前正在使用 Windows，因此那些思考短期价值的学生将会想要学习怎样使用 Windows 并且要求学校讲授 Windows。然后学校就会讲授 Windows，从而培养出一届又一届只习惯于使用 Windows 的毕业生，而这又会鼓励公司使用 Windows。

微软一直在积极培养这种社会惯性：它鼓励学校反复灌输对 Windows 的依赖性，并且约定企业建立起了众多经后来证明只能从 Internet Explorer 浏览器访问的网站。

几年以前，微软的广告试图论证运行 Windows 比运行 GNU/Linux 更加低成本。这些广告里做出的比较已经被揭穿。但是仍然值得注意的是，它们论证中的深层次瑕疵。一条暗含的前提条件援引一种形式的社会惯性，即 “目前更多的技术人员相对于 GNU/Linux 更加熟悉 Windows”。真正珍视自由的人们不会为了节约成本而放弃自由，但是很多企业的执行官在思想意识上仍然坚信他们所拥有的任何东西都应该是可以买卖的，甚至是他们的自由。

社会惯性由那些屈从于社会惯性的人们构成。当您屈从于社会惯性时，您也成为了这种施加于他人的无形压力的一部分。而当您抵抗这种社会惯

性的时候，您已经削弱了它的存在。我们可以通过识别社会惯性来克服它，并且坚定地拒绝成为它的一部分。

这是一个阻碍我们的社区发展的弱点：大部分 GNU/Linux 用户甚至从未听说过曾经推动了 GNU 运动的自由理念，因此他们仍然倾向于根据短期的易用性来评价事物，而非基于他们的自由。这使得他们更易被社会惯性牵住鼻子，从而也成为这种社会惯性的一部分。

为了建立起我们的社区力量以抵抗社会惯性，我们需要谈论自由软件和自由理念——而非仅仅谈论那些开源支持者们所援引的实际好处。随着更多人认识到他们为了克服社会惯性需要做什么，我们将会取得更大进展。

# 自由还是权力?



本文由 Bradley M. Kuhn 和 Richard Stallman 撰写。

对自由的爱是对他人的爱；对权力的爱是对自己的爱。——*William Hazlitt*

在自由软件运动中，我们主张的是软件用户的自由。我们通过观察哪些自由是美好的生活方式所必需的来阐述我们的观点，并且让实用的程序能够帮助一个友善、合力、协作的社区成长。我们对于自由软件的评价尺度[1]指出了软件用户赖以同社区协作所需的自由。

我们主张程序员和其他用户的自由。我们中的大部分人是程序员，在赋予您自由的同时也需要拥有我们自己的自由。但我们中的每个人都需要使用由他人编写的软件，并且我们需要在使用那些软件的时候拥有自由，而不仅仅是使用我们自己编写的代码的自由。我们代表所有用户的自由，不论他们经常、偶尔还是从不进行编程。

然而，有一种所谓的自由是我们绝不提倡的，即“为您所编写的软件随意选择授权许可的自由”。我们拒绝承认这一条的原因在于它实际上是一种权力而非自由。

这种时常被忽视的差别是至关重要的。自由是指能够做出主要是影响自己的决定；而权力是指能够做出影响他人甚于自己的决定。如果我们将权力和自由相混淆，我们将不能支持真正的自由。

使一个程序成为私有软件的行为是一种行使权力的行为。当今的版权法律赋予了软件开发者这样的权力，因此他们，也只有他们，可以选择施加于他人的规则——相对少数人替所有用户做出了关于软件的基本决定，通常是否定了用户的自由。当用户缺少定义了自由软件的那些基本自由的时候，他们不能获知自己所用的软件到底在做什么，不能检查软件的后门，

[1] 关于自由软件的评价标准可参见《什么是自由软件?》 (p. 1) 一文。

不能监视可能存在的病毒或蠕虫，不能找出哪些个人信息正在被泄露（或者即使能够知道也无法阻止泄露）。如果软件不能正常工作，他们不能修复错误，而只能等待开发者行使他们的权力来进行修复。如果软件本身不能根据用户所需要的那样工作，他们将受困于此。他们不能帮助他人改进这些软件。

私有软件的开发者往往也是商业公司。在自由软件运动中，我们并不反对商业行为，但我们也看到了，当一家软件公司拥有对其用户施加任意规则的“自由”的时候将会发生什么。关于拒绝用户的自由是如何导致直接危害的，微软是一个极坏的例子，但还不是唯一。即使是在没有垄断的情况下，私有软件也会对社会造成危害。被迫在众多主宰者中选择一个的权利并不能称之为自由。

关于软件权利和规则的讨论往往专注于程序员的利益。世界上只有少数人经常编写程序，更少的人是私有软件公司的拥有者。但是如今整个发展着的世界都需要并且使用软件，因此现今的软件开发者实际控制了整个世界的生活、经营、通讯和娱乐的方式。伦理和政治上的问题并不是一句口号“选择的自由（仅对开发者适用）”就能解决的。

如果“代码就是法律”[1]，那么我们所面对的真正问题就是：您所使用的代码究竟应该由谁来控制——您，还是少数菁英人士。我们相信您应该有权控制您所使用的软件，并且赋予您这样的控制权正是自由软件的终极目标。

我们坚信您应该有权决定您要使用软件做什么；然而，这不是当今的法律所表达的。当前的版权法律将我们置于这样的境地：权力凌驾于代码用户之上，不论我们是否喜欢这样。对于这种情况的伦理回应是宣示每一位用户的自由，如同权利法案要求以保证每位公民的自由为前提来行使政府的权力。这正是 GNU 通用公共许可证所致力于实现的：它使您能够控制您对软件的使用，同时保护您不被他人剥夺您做出决定的权利。[2]

[1] William J. Mitchell, *City of Bits: Space, Place, and the Infobahn* (Cambridge, Mass.: MIT Press, 1995), p. 111, as quoted by Lawrence Lessig in *Code and Other Laws of Cyberspace, Version 2.0* (New York, NY: Basic Books, 2006), p. 5.

[2] 参见《为什么使用 Copyleft?》 (p. 225) 获得更多信息。

随着越来越多的人意识到代码就是法律，并且开始感受到他们也应当拥有自由，他们将会看到我们所主张的自由是多么重要，如同越来越多的用户开始感激我们所开发的自由软件带来的实际价值。

# 缺陷并不等同于压迫



当一款自由软件缺少用户想要的某些功能的时候，这是一件不幸的事情；我们需要人们添加缺失的功能。某些人可能会更进一步，并且宣称一款软件甚至不能称之为自由软件，如果它缺少某些功能——由于它拒绝了其所不能支持的用户或应用场景的自由之零（以任何方式运行软件的自由）。这一论述是理解错误的，因为它将功能判定为自由，而将缺陷等同于压迫。

任何软件在拥有特定功能的同时，不可避免地缺少某些可能是值得期待的其他功能。它可被用于执行某些任务，而它需要今后的进一步开发完善才能执行其他一些任务，这正是软件的本质。

缺少某些关键的功能可能意味着特定用户将会认为该软件完全不可用。例如，如果您只懂得使用图形界面，一款命令行界面的软件可能对您而言完全不可用；如果您不能看清屏幕，一款不带屏幕读取器的软件可能对您而言完全不可用；如果您只会讲希腊语，一款带有英语菜单和提示信息的软件可能对您完全不可用；如果您的源程序是使用 Ada 语言编写的，一款 C 语言编译器可能对您完全不可用。要求您自己克服这样的障碍是不合理的。自由软件确实应该提供您所需要的功能。

自由软件确实应该提供这些功能，然而缺少这些功能并不会使得这款软件成为非自由的，因为这只是一种缺陷而非一种压迫。

使一款软件成为非自由的是由拒绝用户自由的开发者做出的不公行为。这样的开发者应该因此而受到谴责。谴责这样的开发者是重要的，因为只要开发者一直这么做，其他人就不能够解除这种不公。我们可以，并且确实在尝试通过开发自由的替代品来挽救受害者。但我们不能使那些非自由软件变成自由的。

开发一款自由软件而没有添加某些重要功能并不是对任何人作错事。与之相反，这在某种程度上是在做好事，只是尚未实现人们所需要的全部好处。并没有某些特定的人应该因为未能开发出某些缺失的功能而受到谴

责，因为任何有能力进行弥补的人都可以做。单独指出自由程序的作者们并且指责他们未能完成某些额外的工作是不通情理的，并且这样做只会自讨苦吃。

我们所能做的是指出完成这项任务尚需进行哪些额外的开发工作。这将是建设性的，因为这有助于我们说服某些有能力的人去从事那些开发工作。

如果您认为某个自由软件中的某个特定扩展程序十分重要，请以一种尊重我们贡献的方式提出。不要批判那些已经为我们贡献了有用代码的人。与之相反，您应该试着找到一种方式以完成任务。您可以敦促该程序的开发者在他们有时间进行更多工作的时候关注所缺失的功能；您还可以为他们提供帮助；您也可以通过招募人员或者筹集资金的方式支持这项工作。

# 民主可以承受多少监控?



多亏了爱德华 • 斯诺登的揭露，我们才知道当今社会中的普遍监控级别已经与人权不相容。在美国和世界其他地区持续发生的针对持不同政见者、信息来源和新闻工作者的骚扰和指控确认了这一点。我们需要适当降低普遍监控的级别，但是到底应当在多大程度上？我们所必须保证其不被超越的，可接受的最大监控级别到底是多少？这种监控级别应当这样定义：一旦超过了这样的级别，监控行为将会干涉民主的运行。此时揭露者（例如斯诺登）很可能因此而被逮捕。

面对政府的保密政策，我们作为人民只能依靠揭露者告知国家正在做什么[1]。然而，当今的监控对潜在的揭露者进行了威迫，也就是说监控程度过高。为了重获我们对于国家民主的控制，我们必须设法降低监控级别使得揭露者能够确信他们是安全的。

如同我们已经倡导了 30 多年的，使用自由软件是掌控我们自己的数字生活的第一步。这也包括了避免被监控。我们不可能信任任何私有软件；由于美国国家安全局（NSA）通过利用[2]甚至还有制造[3]私有软件中的安全漏洞来入侵我们自己的计算机和路由器。自由软件赋予了我们控制自己计算机的权利，但仅凭这一点并不足以在我们涉足互联网的时候保护我们的

[1] Maira Sutton, “We’re TPP Activists: Reddit Asked Us Everything,” 21 November 2013, https://www.eff.org/deeplinks/2013/11/reddit-tpp-ama.

[2] Glyn Moody, “How Can Any Company Ever Trust Microsoft Again?” 17 June 2013, http://www.computerworlduk.com/blogs/open-enterprise/how-can-any-company-ever-trust-microsoft-again-3569376/.

[3] James Ball, Julian Borger and Glenn Greenwald, “Revealed: How US and UK Spy Agencies Defeat Internet Privacy and Security,” 6 September 2013, http://theguardian.com/world/2013/sep/05/nsa-gchq-encryption-codes-security.

隐私[1]。

美国国内 “限制国内监控权力”[2]的两党立法已经被提出，但它依赖于限制政府对我们的虚拟档案的使用。这不足以保护揭露者，如果 “逮捕揭露者” 是获取足够信息以确认此人身份的基础。我们需要做得更多。

## 民主社会中的监控程度上限

如果揭露者不敢揭发罪行和谎言，我们将会失去对政府和制度仅存的有效控制。这可以解释为何那种使得国家可以知道是谁同记者进行交谈的监控行为是过分的——超出了民主所能承受的范围。

在 2011 年，一位匿名的美国政府官员不怀好意地告诉新闻工作者：美国政府在传唤记者时，并不会以 “我们知道你同谁交谈” 作为理由[3]。有时，记者的通话记录将会被传唤以获知这个问题的答案[4]。但是，斯诺登已经向我们展示了，它们实际上将会在任何时刻传唤美国境内的任何人的所有通话记录，通过 Verizon [5]或其他通讯公司[6]。

反对者和持不同政见者的活动需要对国家保密，由于国家将会主动对

[1]Bruce Schneier, “Want to Evade NSA Spying? Don’t Connect to the Internet,” 7 October 2013, http://www.wired.com/2013/10/149481/.

[2]Dan Roberts, “Patriot Act Author Prepares Bill to Put NSA Bulk Collection ’Out of Business,”’ 10 October 2013, http://theguardian.com/world/2013/oct/10/nsa-surveillance-patriot-act-author-bill.

[3]Lucy Dalglish, “Lessons from Wye River,” The News Media & the Law (Summer 2011): p. 1, http://www.rcfp.org/browse-media-law-resources/news-media-law/news-media-and-law-summer-2011/lessons-wye-river.

[4]Washington Agencies, “Yemen leak: former FBI man admits passing information to Associated Press,” 24 September 2013, http://www.theguardian.com/media/2013/sep/24/yemen-leak-sachtleben-guilty-associated-press.

[5]参见 “Verizon forced to hand over telephone data—full court ruling” (6 June 2013) 位于 http://www.theguardian.com/world/interactive/2013/jun/06/verizon-telephone-data-court-order，以获得关于美国外国情报监控法庭（FISA）之下美国政府采集 Verizon 旗下数以百万计的美国客户的通话记录的信息。

[6]Siobhan Gorman, Evan Perez, and Janet Hook, “NSA Data-Mining Digs into Networks Beyond Verizon,” 7 June 2013, http://www.marketwatch.com/story/nsa-data-mining-digs-into-networks-beyond-verizon-2013-06-07.

他们施展阴谋诡计。美国公民自由联盟（ACLU）已经指出美国政府对和平的持不同政见者和组织所进行的系统性渗透行为，理由是他们当中可能潜伏着恐怖分子[1]。

## 信息，一旦被采集，将会被滥用

当人们认识到普遍监控的级别过高的时候，他们的第一反应可能会是提议限制对采集到的数据的访问。这听起来很好，但这甚至丝毫不能解决问题，即使假定政府遵守这条规则。（美国国家安全局（NSA）曾经欺骗美国外国情报监控法庭（FISA），后者声称它不能有效地证实 NSA 有义务对其监控行为作出说明[2]。）犯罪的嫌疑将成为访问这些数据的理由，于是，一旦揭露者被指控从事间谍活动，试图找到这个间谍将会为访问所采集到的数据提供理由。

此外，国家监控系统的工作人员会出于个人原因滥用数据。一些 NSA 特工使用美国国家监控系统来跟踪他们的情人——不论是过去的、现在的还是正在追求的——并将这种做法称为 LOVEINT[3]。NSA 宣称它已经发现了几次这样的行为并对其进行了处罚；但是我们不知道还有多少次这样的行为没有抓到。然而，这些事件并不会让我们感到惊讶，由于警方已经长期利用他们可访问的驾照记录以跟踪那些迷倒了他们的对象。这种行为称为“查询车牌获得约会”（running a plate for a date）[4]。

监控数据总会被用于其他目的，即使这是被禁止的。只要数据被采集，国家就有可能访问它们。国家可以用极坏的方式滥用这些数据，发生在欧

[1] ACLU, “Policing Free Speech: Police Surveillance And Obstruction of First Amendment-Protected Activity,” 29 June 2010, https://www.aclu.org/files/assets/Spyfiles_2_0.pdf.

[2] David Kravets, Kim Zetter, Kevin Poulsen, “NSA Illegally Gorged on U.S. Phone Records for Three Years,” 10 September 2013, http://www.wired.com/2013/09/nsa-violations/.

[3] Adam Gabbatt and agencies, “NSA Analysts ‘Wilfully Violated’ Surveillance Systems, Agency Admits,” 24 August 2013, http://theguardian.com/world/2013/aug/24/nsa-analysts-abused-surveillance-systems.

[4] M. L. Elrick, “Cops Tap Database to Harass, Intimidate,” 31 July 2001, http://sweetliberty.org/issues/privacy/lein1.htm/#/.VeQiuxcpDow.

洲[1]和美国[2]的一些例子证实了这一点。

国家所采集的个人信息同样可能由于服务器的安全措施被攻陷而被境外骇客获得，甚至是被那些效力于敌对国家的骇客获得[3]。

政府可以轻松地利用大规模监控能力直接颠覆民主制度[4]。

国家对监控数据的完全访问权力使得国家可能对任何人发动大规模的彻底搜查。为了保护新闻业和民主的安全，我们必须限制对信息的采集，而这些信息是国家可以轻松访问的。

## 对隐私的有效保护必须是技术层面的

电子前哨基金会（EFF）和一些其他组织提出了一系列法律准则以期阻止滥用大规模监控[5]。这些关键性准则包括了对揭露者明确的法律保护。其结果是这些准则将足以保护民主自由——如果它们能够永久地得到完全采纳，并且被没有例外地严格强制执行。

然而，这样的法律保护是不牢靠的：如同最近多年的历史事实所展示的，它们可以被废除（如同在美国外国情报监控法案 FISA 修正案中）、被架空或者被无视[6]。

同时，一些蛊惑民心的政客将会援引常见的理由作为支持完全监控的基础；任何恐怖袭击，即使只是造成了极少的人员伤亡，也可以被夸大作

[1] Rick Falkvinge, “Collected Personal Data Will Always Be Used against the Citizens,” 17 March 2012, http://falkvinge.net/2012/03/17/collected-personal-data-will-always-be-used-against-the-citizens/.

[2] 考虑二战期间被收容的日裔美国人。

[3] Mike Masnick, “Second OPM Hack Revealed: Even Worse Than the First,” 12 June 2015, https://www.techdirt.com/articles/20150612/16334231330/second-opm-hack-revealed-even-worse-than-first.shtml.

[4] Joanna Berendt, “Macedonia Government Is Blamed for Wiretapping Scandal,” 21 June 2015, http://www.nytimes.com/2015/06/22/world/europe/macedonia-government-is-blamed-for-wiretapping-scandal.html?_r=0.

[5] “International Principles on the Application of Human Rights to Communications Surveillance,” last modified May 2014, https://en.necessaryandproportionate.org/text.

[6] Eric Lichtblau and James Risen, “Officials Say U.S. Wiretaps Exceeded Law,” 15 April 2009, http://nytimes.com/2009/04/16/us/16nsa.html.

为证据以支持上述理由。

如果对数据访问的限制被搁置一旁，就如同这些限制从未存在过一般：多年来的有价值的档案将会立即可被国家和它的特工所滥用。如果这些档案被商业公司获得，也会被这些商业公司出于其私有目的而被滥用。然而，如果我们能够阻止对个人档案的采集，这些档案将不复存在，并且国家不能追溯既往地编制这些档案。新的反自由的政体将会不得不从头实施监控，并且它自成立之初只会采集新的数据。至于架空或者立即无视这条法律，这种想法几乎没有任何意义。

## 最重要的是，不要犯傻

如果您想要拥有隐私，您一定不要主动将其放弃：有义务保护您的隐私的最重要的人就是您自己！您必须避免向网站暴露自己的身份，您可以使用 Tor 同它们联络，或者使用那些能够阻止网站用于跟踪访问者的阴谋诡计的浏览器。您可以使用 GNU 隐私卫士（GnuPG）加密您的邮件内容。您可以使用现金支付任何费用。

保护您自己的数据；不要将您的数据储存在某家商业公司的“简便易用”的服务器上。然而，在下面这种情况下，将数据存储在商业服务之上是安全的：只要您将文件置于归档文件中，并且加密整个归档文件，包括文件名。务必使用自由软件在您自己的计算机上进行操作，然后再上传。

出于隐私考虑，您必须避免使用私有软件，这是由于私有软件赋予他人控制您的计算机使用的权力，它们很可能在监控您[1]。您还应该拒绝使用“作为软件替代品的服务”（SaaSS）[2]，同样由于这赋予了他人控制您如何计算的权力，它要求您将全部相关数据提交到它们的服务器。

您也需要保护您的朋友或熟人的隐私。除了联系方式以外，不要泄露

[1]几十年来，自由软件运动一直致力于揭露私有软件公司诸如微软和苹果的专权的监控机制。监控行为已经在各行业之间蔓延，而不再限于软件产业，并且——离开键盘的限制——进入移动计算领域，在办公室、家庭、交通工具、教室内等。如需获知监控行为进入这些领域的不断增加的方式，参见 http://gnu.org/philosophy/proprietary/proprietary-surveillance.html.

[2]参见《服务器真正是在为谁服务?》 (p. 302) 一文以获取更多信息。

他们的任何个人信息[1]。并且不要向任何网站泄露您的邮件列表或者电话联系人。不要将您的朋友的任何信息告诉诸如 Facebook 这样的公司，因为您的朋友也许并不想在报纸上公布他们的名字。如果可能，根本不要被 Facebook 所利用。拒绝使用任何要求用户提供真实姓名的通讯系统，即使您愿意供出您的名字，由于这些通讯系统会向他人施压以迫使他们交出隐私。

自我保护是至关重要的，但即使是最严密的自我保护也不足以保护您的隐私免于被不属于您的系统所泄露。当我们与他人通讯或者在城市内出行的时候，我们的隐私取决于社会的实践。我们可以避开一些但不是全部的可以监控我们的通讯或行踪的系统。显然，更佳的解决方案是让所有这些系统停止监控任何人，除了法律意义上的嫌犯。

## 我们需要基于隐私原因设计每一个系统

如果我们不想要一个全面监控的社会，我们必须将监控视为一种社会污染，并且限制监控对每一个新系统的影响，如同我们要限制实体建设工程对环境的影响。

例如，“智能” 电表宣称它能够向电力公司持续发送每位用户的电力消耗，包括与普通用户相比较的情况。这种统计是基于普遍监控而实现的，但实际上却又不需要任何监控行为。电力公司可以容易地计算出某一居住区的平均电力消耗，通过将总消耗除以购电者的数量，并且将这个平均值发送至电表。每个用户的电表可以将本人的电力使用情况同任何时期的平均电力使用相比较，这样就可以不用监控而实现所有好处！

我们需要将这样的隐私设计融入我们所有的数字系统。

[1]Nicole Perlroth, “In Cybersecurity, Sometimes the Weakest Link Is a Family Member,” 21 May 2014, http://bits.blogs.nytimes.com/2014/05/21/in-cybersecurity-sometimes-the-weakest-link-is-a-family-member/.

## 针对数据采集的补救：让数据分散开来

使监控行为不危害隐私的一种方式是使数据保持分散状态从而难于访问。老式的安保摄像头对隐私很少构成威胁[1]，由于录像数据存储在安装这些设备的营业场所内，并且至多被保存几周。由于访问这些数据相对困难，这种数据采集行动从未被大规模部署；只有当某人报导一起犯罪行为时，这些影响数据才会被访问。每天对数以百万计的数据卡带进行人工采集再进行观看或复制几乎是不现实的。

如今，老式的安保摄像头已经变成了监控摄像机：由于它们被连接到互联网，采集的影像可以传输到一所数据中心并被永久保存。这已经是很危险的了，但事情正在变得更坏。面部识别技术的发展使得这样的事情成为可能：对可疑的新闻记者在街道上进行不间断跟踪以便察看他们同何人交谈。

通常，联网摄像机的自身数据安全措施极差，使得几乎任何人都能察看它所记录的内容[2]。为了重获隐私，我们应当禁止在针对公众的场合使用联网的摄像机，除非它由人来操作。每个人必须被允许偶尔发布照片或视频记录，但对于互联网上相关数据的系统性采集行为必须受到限制。

## 针对互联网商业监控的补救

大部分数据采集行为来自于人们自身的数字活动。数据通常首先由商业公司进行采集。但是，当讨论监控行为对隐私和民主的威胁的时候，监控行为直接由国家进行或者由商业公司代为进行并无本质区别，由商业公司所采集的数据，国家也可以系统性地获取。

[1]我在此假定安保摄像头用于诸如商店内部或大街上。任何由其他人架设的对准某人私人空间的摄像机侵犯了隐私，但这是另一个问题。

[2]Ms. Smith, “CIA Wants to Spy On You through Your Appliances,” 18 March 2012, http://networkworld.com/article/2221934/microsoft-subnet/cia-wants-to-spy-on-you-through-your-appliances.html.

NSA 通过棱镜计划（PRISM）进入了多家大型互联网公司的数据库[1]。AT&T 自 1987 年起保存了所有通话记录并且允许美国缉毒局（DEA）搜索其所有数据[2]。严格地说，美国政府并不直接拥有这些数据，但它实际上确实拥有了这些数据。

因此，为了保证新闻业和民主的安全，我们必须减少由任何组织采集的关于个人的数据，而不仅仅减少由国家采集的数据。我们必须这样重新设计各种数字系统以使其不再采集其用户的数据。如果它们确实需要关于我们的重要数字资料，当超出处理数据所需的基本时间以后，这些机构和个人就不应该被允许继续保留那些数据。

当前，互联网监控级别的动机之一是经济支持，即网站通过跟踪用户的活动和偏好而进行的广告行为。这使得广告，作为一种我们可以学会无视之的行为，从一种仅仅是恼人的行为变成了一种对我们造成伤害的监控系统，不论我们是否了解它的内情。互联网购物同样会跟踪用户，我们都已经意识到，所谓的 “隐私条款” 与其说是维护用户隐私的承诺，不如说是它们用于侵犯用户隐私的借口。

我们可以通过采用一种匿名支付系统——即隐藏付款人的身份——来解决以上两个问题（我们并不想协助收款人避税）。比特币不是匿名的[3]，尽管有人试图开发出允许使用比特币进行匿名支付的方式。然而，数字货币技术的开发始于 20 世纪 80 年代[4]；我们只需要对商业规则进行适当调整，并且使得国家不会阻止它们。

网站对个人数据的采集的更大的威胁在于骇客可以攻陷安全措施，获

[1] Jon Queally, “Latest Docs Show Financial Ties between NSA and Internet Companies,” 23 August 2013, http://www.commondreams.org/news/2013/08/23/latest-docs-show-financial-ties-between-nsa-and-internet-companies.

[2] Scott Shane and Colin Moynihan, “Drug Agents Use Vast Phone Trove, Eclipsing N.S.A.’s,” 1 September 2013, http://www.nytimes.com/2013/09/02/us/drug-agents-use-vast-phone-trove-eclipsing-nsas.html?_r=0.

[3] Dan Kaminsky, “Let’s Cut through the Bitcoin Hype: A Hacker-Entrepreneur’s Take,” 3 May 2013, http://wired.com/2013/05/lets-cut-through-the-bitcoin-hype/.

[4] Steven Levy, “E-Money (That’s What I Want),” Wired, 2.12 (December 1994), http://archive.wired.com/wired/archive/2.12/emoney_pr.html.

取并滥用个人数据。这可能还包括用户的信用卡信息。而匿名支付系统可以终结这种威胁：如果网站不知道您的任何信息，那么网站的安全漏洞就不会危害到您。

## 针对旅行监控的补救

我们必须将数字收费系统改为匿名支付系统（例如使用数字货币）。车辆牌照识别系统将会识别各种牌照，而这些数据可以被无限期保存[1]；应该由法律要求它们仅仅记录那些由法庭命令要求追查的牌照号码。另一种不太安全的措施是在本地记录所有车辆牌照，但仅保存几天时间，并且不允许从网络访问所有数据；对数据的访问应该限于搜索一系列由法庭命令要求追查的牌照号码之中。

美国“禁飞黑名单”必须被废除，由于这是一种未经审判的刑罚[2]。

要求对黑名单上某个乘客的行李进行额外的搜查是可以接受的，国内航班上的匿名乘客可以视为在此黑名单上。禁止非某国公民登上飞往该国的航班也是可以接受的，如果他们根本没被批准入境。这些措施对于任何法律目的都是足够的。

很多公共交通系统使用某种智能卡或者射频识别（RFID）设备进行支付。这些系统将会采集个人数据：只要您错误地使用现金以外的任何方式进行支付，它们将会将此卡片和您的姓名永久关联起来。接下来，它们将会记录与每块卡片相关联的所有出行信息。这些行为加起来已经构成了大规模监控，这样的数据采集必须被限制。

导航服务也会进行监控：用户的计算机将用户的所在地和目的地告知地图服务；而后服务器确定路线，返回用户的计算机并且显示出来。现在，服务器很可能会记录用户的位置信息，由于没有什么措施能够阻止它们这

[1] Richard Bilton, “Camera Grid to Log Number Plates,” last updated on 22 May 2009, http://news.bbc.co.uk/2/hi/programmes/whos_watching_you/8064333.stm.

[2] Nusrat Choudhury, “Victory! Federal Court Recognizes Constitutional Rights of Americans on the No-Fly List,” 29 August 2013, https://www.aclu.org/blog/victory-federal-court-recognizes-constitutional-rights-americans-no-fly-list.

样做。这种监控行为本质上并不必要，并且可以通过重新设计来解决：用户计算机中的自由软件将会下载相关地区的地图数据（如果之前从未下载），计算出最佳路线并且显示出来，而无需告知任何人用户的所在地或目的地。

用于诸如自行车租借等目的的系统可以这样设计：租借者的身份仅在其借出物品的站点内可知。物品借出时，将会通知所有站点某件物品处于借出状态，这样，当用户将物品返还回任何站点（通常是另一处站点）的时候，该站点将会获知该物品被借出的时间和地点，同时将会告知所有其他站点该物品不再处于借出状态。站点还会计算用户的账单并将账单信息（等待随机长度的时间之后）沿着一系列环形拓扑结构的站点之间的线路发送至总部。这样，总部将不会获知账单信息来自哪个站点。当这一操作完成后，归还站点将会忘记所有与这笔已完成的业务有关的信息。如果某一物品长时间处于借出状态，借出该物品的站点将会告知总部。此时，它可以立即发送借用者的身份信息。

## 关于通讯档案的补救

互联网服务供应商（ISP）和电信公司保存着海量的用户联系人信息（浏览、通话记录等）。对于移动电话，还会记录用户的物理位置[1]，例如 AT&T 已保存了超过 30 年。不久以后它们甚至还会记录用户的身体活动[2]。并且 NSA 很可能正在大规模采集移动电话的物理位置数据[3]。

只要通讯系统创建这样的通讯档案，不受监视的通讯就不可能实现。因此创建或记录这些通讯档案应该被判定为非法。ISP 和电信公司必须不被允许长期保存这些信息，或者在没有法庭命令的情况下长期监控某一特

[1]Kai Biermann, “Betrayed by Our Own Data,” 26 March 2011, http://www.zeit.de/digital/datenschutz/2011-03/data-protection-malte-spitz.

[2]Sara M. Watson, “The Latest Smartphones Could Turn Us All into Activity Trackers,” 10 October 2013, http://wired.com/2013/10/the-trojan-horse-of-the-latest-iphone-with-the-m7-coprocessor-we-all-become-qs-activity-trackers/.

[3]Patrick Toomey, “It Sure Sounds Like the NSA Is Tracking Our Locations,” 30 September 2013, https://aclu.org/blog/it-sure-sounds-nsa-tracking-our-locations.

定人群。

这种解决方案并不完全令人满意，由于这实际上并不能阻止政府在通讯信息生成的时候立即对其进行采集——这正是美国政府对部分或全部电信公司所做的[1]。我们可能必须依靠法律禁止这种行为。但是，这种假设比现实的状况好得多，现实中的相关法律（美国爱国者法案，我称之为 PATRIOT Act，即“镇压暴动法案”）并不明确禁止这种行为。此外，如果政府重启这种监控，它不应得到重启监控的时间点之前发生的每位用户的通话记录数据。

为了保护您的电子邮件联系人的隐私，一种简单的方式是您和他人都使用某个不会与您所在国家政府进行合作的国家提供的邮件服务，并且在通讯过程中使用加密。然而，Ladar Levison（Lavabit 的拥有者，美国监控系统试图对其邮件服务实现完全控制）提出了一种更为高级的加密系统设想：您向我的邮件服务的某位用户发送邮件，我的邮件服务所知道的只是我收到了来自您所使用的邮件服务的某位用户的邮件，但难以确认是您向我发送了邮件。

## 但是，适度的监控是必需的

国家为了缉捕罪犯，它需要能够在法庭命令下调查特定的犯罪行为或者疑似犯罪预谋。在互联网时代，监听通话的权力自然延伸到监听互联网连接的权力。这种权力容易出于政治原因而被滥用，但这也是必需的。幸运的是，这并不会使得在案件发生之后找到揭露者变为可能，如果（如我所建议的）能够阻止数字系统在事件发生之前进行大规模档案信息采集。

拥有国家赋予权力的个人，例如警察，将被收回个人的隐私权并且

[1] Glenn Greenwald, “NSA Collecting Phone Records of Millions of Verizon Customers Daily,” 6 June 2013, http://www.theguardian.com/world/2013/jun/06/nsa-phone-records-verizon-court-order.

必须被监视（事实上，警察拥有属于他们自己的伪证罪别名 “testilying”[1]。由于他们经常做出这样的事情，特别是对抗议者和摄影师[2]）。加州的某个城市要求警察随时随身携带摄像机之后，他们的武力使用下降了 60% [3]。ACLU 对此表示欢迎。

商业公司不是自然人，因此不应被赋予自然人的权利[4]。要求商业公司公开其行为的细节是正当合理的，这些行为可能会造成化学、生物、核、财政、计算机相关（例如数字限制管理 DRM[5]）或者政治（例如游说拉票行为）等方面对社会的危害，这些危害必须被控制在公众福祉所要求的范围以内。这些行为所造成的危害（考虑诸如墨西哥湾漏油事件、福岛核电站事故、2008 年财政危机等）更甚于恐怖主义。

然而，新闻业必须被保护免遭监控，即使这种监控行为是作为某项事务的一部分而被执行的。

数字技术的发展极大地提高了我们的出行、活动和通讯所受的监控水平。这种监控水平远远超过了 20 世纪 90 年代我们所经历过的，也远远超过了 20 世纪 80 年代生活在铁幕笼罩之下的人们所经历的[6]。而提议国家对使用采集到的数据进行法律限制并不能改变这种状况。

商业公司正在设计更具侵略性的监控设施。一些充斥着监控行为的项目依附于诸如 Facebook 之类的公司，它们可能对人们的思考方式产生深

[1] 一些例子 “Testilying: Cops Are Liars Who Get Away with Perjury” (Nick Malinowski, 3 February 2013, http://vice.com/read/testilying-cops-are-liars-who-get-away-with-perjury) 以及 “Detective Is Found Guilty of Planting Drugs” (Tim Stelloh, 1 November 2011, http://nytimes.com/2011/11/02/nyregion/brooklyn-detective-convicted-of-planting-drugs-on-innocent-people.html?pagewanted=all&_r=0)

[2] 关于这一点参见 “摄影不是犯罪” 网站 http://photographyisnotacrime.com/

[3] Kevin Drum,“Ubiquitous Surveillance, Police Edition,” 22 August 2013, http://motherjones.com/kevin-drum/2013/08/ubiquitous-surveillance-police-edition.

[4] Public Citizen, “Call Your Representative: Tell Her or Him to Co-Sponsor a Constitutional Amendment to Overturn Citizens United and Restore Democracy to the People,” August 2015, http://action.citizen.org/p/dia/action3/common/public/?action_KEY=12266.

[5] 参见《应避免使用（或慎用）的词语》一文中有关 DRM 的叙述 (p. 115) 。

[6] James Allworth, “Your Smartphone Works for the Surveillance State,” 7 June 2013, https://hbr.org/2013/06/your-iphone-works-for-the-secret-police.

远的影响[1]。这样的可能性是不可预测的；然而它对民主的威胁已经不是推测。这种威胁无处不在，随时可见。

除非我们坚信我们自由的国家之前的监控行为严重欠缺，并且我们理应受到更甚于苏联和民主德国（东德）那样的监控，否则我们必须逆转这种监控升级的趋势。这依赖于阻止对民众的大数据进行大规模采集。

[1]Evan Selinger and Brett Frischmann, “Will the Internet of Things Result in Predictable People?” 10 August 2015, http://theguardian.com/technology/2015/aug/10/internet-of-things-predictable-people.

# 附录 A

# 关于软件的基础知识



由 Richard E. Buckman 和 Joshua Gay 撰写

本节内容是特意为对计算机科学技术了解不多的人准备的。这节内容对于理解书中的文章和演讲并不是必须的，不过，它可以帮助那些不熟悉编程和计算机的人理解一些术语。

计算机程序员来写软件或者计算机程序。而程序可以认为是告诉计算机如何完成特定任务的一系列指令。你应该熟悉许多不同类型的应用程序：比如你的网页浏览器、你的文字处理器和你的邮件客户端等等。

程序最初的形态通常是源代码。这一系列高级指令由编程语言（比如 C 或者 Java）编写而成。之后会被一个名为编译器的工具编译为一种更底层的语言——汇编语言。另一种被称为汇编器的工具会将汇编代码分解为最终的机器语言——计算机可以原生理解的最底层代码。

例如，“Hello World” 这个程序，通常是人们学习 C 语言时的第一个程

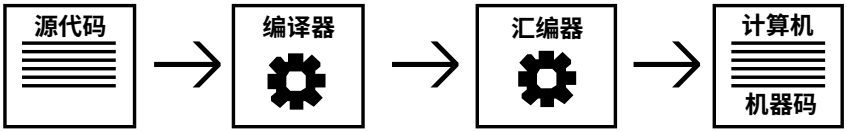


图 **A.1:** 计算机程序是如何从源代码生成二进制可执行文件的

序，编译和执行后会在屏幕上打印出 “Hello World”[1]。

```
int main(){
    printf("Hello World!");
    return 0;
}
```

在 Java 语言中，同样的程序会是这么写：

```
public class hello {
    public static void main(String args[]) {
        System.out.println("Hello World!");
    }
}
```

然而用机器码表示，其中的一小部分可能会类似这样：

```
11000111101110101001010010010010101011 10
01101010100110000011110010110101011111 01
```

[1]对于其他的编程语言，比如 Scheme，通常不从 Hello World 程序开始入门。在 Scheme 中，你通常会从这样的程序开始：

```
(define (factorial n)
    (if (= n 0)
    1
    (* n (factorial (- n 1)))))
```

这段程序是求一个数的阶乘；也就是运行 `(factorial 5)` 将会输出 120，即 5 乘以 4 乘以 3 乘以 2 乘以 1。

```
01001111111111110010110110000000010100100
01001000011001010110110001101100011011111
00100000010101110110111101110011001101100
01100100001000010100001001101111011011111
```

上面这段机器码就是基本的二进制形式。计算机中所有的数据都是由 0 或 1 组成的，但是人们很难理解这些数据。如果想要对二进制的值进行一个简单的更改，就不得不对特定计算机编译机器语言的原理了如指掌。这对于类似于上面这种小程序来说或许是可行的，但是对于任何有趣的程序，做一个简单的修改就需要耗费大量的精力。

比如说，如果我们想让让 C 语言编写的 “Hello World” 程序输出的英语 “Hello World” 变成法语。只需要做出简单的修改；新的程序如下：

```
int main() {
    printf("Bonjour, monde!");
    return 0;
}
```

可以肯定地说，我们可以很容易的推断出如何修改 Java 代码以达到同样的效果。然而更多的程序员不懂如何修改二进制形式。当我们说 “源代码” 的时候，不是指只有机器可以理解的机器语言——我们指的是一些高级语言比如 C 和 Java。还有一些其他的比较流行的编程语言，比如 C++、Perl 和 Python。或许有些在理解或编程时比较难以理解，但都比理解编译和汇编以后的机器语言简单多了。

另一个重要概念是理解什么是*操作系统*。操作系统是控制输入输出，内存分配和任务调度的软件。通常点说，一些更常见且更有用的程序，比如*图形用户界面*（*Graphical User Interface*，GUI），也是操作系统的一部分。GNU/Linux 操作系统包括 GNU 和非 GNU 软件，以及一个被称为 *Linux* 的*内核*。内核负责处理底层任务比如输入输出和任务调度。GNU 软件包含了操作系统其余的大部分，比如 GCC，一款支持多种语言的通用编译器；GNU Emacs，一款可扩展的并且有很多很多特性的文本编辑器；GNOME，

GNU 桌面环境；GNU libc，一个程序库，除内核外所有其他程序与内核通讯时都必须使用的库；以及 Bash，GNU 命令行解释器可以读取你的命令行。这些程序很多都是早期由理查德·斯托曼在 GNU 工程里开发，并成为现代 GNU/Linux 操作系统的组成部分。

重要的是理解即使你没有修改程序源代码或直接使用所有这些工具的能力，找到一个可以做到的人也是相对容易的。因此，有源代码的程序你就有权力去修改、修复、定制和学习编程——而如果得不到源代码就没有这些权力。源代码是让一个软件变自由的必要条件之一，而其他必要条件可从本书的哲学和理想中找到答案。

# 附录 B

# 不同语言对 “自由软件” 和 “免费软件” 的翻译



以下列出的是对 “自由软件” 和 “免费软件” 不同语种的翻译，拆分成不同的列以便比较。个别条目中的拉丁字母表示的是音译（相关位置增加了元音）。

| | 语言名称 | 自由软件 | 免费软件 |
|---|---|---|---|
| en | 英语 | free software | gratis software |
| af | 南非荷兰语 | vrye sagteware | gratis sagteware |
| ar | 阿拉伯语 | برمجيات حرة *(barmajiyat □orrah)* | |

| | 语言名称 | 自由软件 | 免费软件 |
|---|---|---|---|
| be | 白俄罗斯语 | свабоднае праграмнае забесьпячэньне *(svabodnae pragramnae zabes’pjachen’ne)* | |
| bg | 保加利亚语 | свободен софтуер *(svoboden softuer)* | безплатен софтуер *(bezplaten softuer)* |
| bn | 孟加拉语 | স্বাধীন সফটওয়্যার *(swadhin software)* | |
| ca | 加泰罗尼亚语 | programari lliure | programari gratuït |
| cs | 捷克语 | svobodný software | bezplatný software |
| cy | 威尔士语 | meddalwedd | rydd |
| da | 丹麦语 | fri software 或 frit programmel | gratis software |
| de | 德语 | freie Software | Gratis-Software 或 kostenlose Software |
| el | 希腊语 | ελεύθερο λογισμικό *(elefthero logismiko)* | δωρεάν λογισμικό *(dorean logismiko)* |
| eo | 世界语 | libera programaro 或 programo | |
| eu | 巴斯克语 | software librea | doako softwarea |
| es | 西班牙语 | software libre | software gratuito |
| et | 爱沙尼亚语 | vaba tarkvara | tasuta tarkvara |
| fa | 波斯语 | نرم افزار آزاد *(narmafzar azad)* | نرم افزار رایگان *(narmafzar raygan)* |
| fi | 芬兰语 | vapaa ohjelmisto | ilmainen ohjelmisto |
| fr | 法语 | logiciel libre | logiciel gratuit |

| | 语言名称 | 自由软件 | 免费软件 |
|---|---|---|---|
| ga | 爱尔兰语 | saorbhogearraí | bogearraí saora in aisce |
| he | 希伯来语 | חופשית תוכנה *(tochna chofshit)* | חינמית תוכנה *(tochna chinamit)* |
| hi | 印地语 | मुक्त सॉफ्टवेयर *(mukt software)* | मुफ़्त सॉफ्टवेयर *(muft software)* |
| hr | 克罗地亚语 | slobodan softver | besplatan softver |
| hu | 匈牙利语 | szabad szoftver | ingyenes szoftver 或 ingyen szoftver |
| hy | 亚美尼亚语 | ազատ ծրագիր/ծրագրեր *(azat tsragir/tsragrer)* | |
| ia | 国际语 | libere programmage 或 libere programmario | |
| id | 印度尼西亚语 | perangkat lunak bebas | |
| io | 伊多 | libera programaro | |
| is | 冰岛语 | frjáls hugbúnaður | |
| it | 意大利语 | software libero | software gratuito |
| ja | 日语 | 自由ソフトウェア *(jiyū-sofutouea)* | 無料ソフトウェア *(muryō-sofutouea)* |
| ka | 格鲁吉亚语 | თავისუფალი პროგრამები *(tavisupali programebi)* | უფასო პროგრამები *(upaso programebi)* |
| ko | 韩语 | 자유 소프트웨어 *(ja-yu software)* | |
| lt | 立陶宛语 | laisva programinė įranga | nemokama programinė įranga |

| | 语言名称 | 自由软件 | 免费软件 |
|---|---|---|---|
| lv | 拉脱维亚语 | brīva programmatūra | bezmaksas programmatūra |
| mk | 马其顿语 | слободен софтвер *(sloboden softver)* | бесплатен софтвер *(besplaten softver)* |
| ml | 马拉雅拉姆语 | സ്വതന്ത്രസോഫ്റ്റ്വെയർ *(svatantrasoph□□veyar)* | സൗജന്യസോഫ്റ്റ്വെയർ *(soujanyasoph□□ve-yar)* |
| ms | 马来语 | perisian bebas | |
| nl | 荷兰语 | vrije software | gratis software |
| no | 挪威语 | fri programvare | |
| pl | 波兰语 | wolne oprogramowanie | darmowe oprogramowanie |
| pt | 葡萄牙语 | software livre | |
| ro | 罗马尼亚语 | programe libere | programe gratuite |
| ru | 俄语 | свободные программы *(svobodnie programmi)* | бесплатные программы *(besplatnie programmi)* |
| sc | 撒丁语 | software liberu | |
| si | 僧伽罗语 | නිදහස් මෘදුකාංග *(nidahas m□dukā□ga)* | |
| sk | 斯洛伐克语 | slobodný softvér | |
| sl | 斯洛文尼亚语 | prosto programje | |
| sq | 阿尔巴尼亚语 | software i lirë | software falas |
| sr | 塞尔维亚 | слободни софтвер 或 slobodni softver | бесплатни софтвер 或 besplatni softver |
| sv | 瑞典语 | fri programvara 或 fri mjukvara | |

| | 语言名称 | 自由软件 | 免费软件 |
|---|---|---|---|
| sw | 斯瓦希里语 | software huru 或 programu huru za kompyuta | |
| ta | 泰米尔文 | கட்டற்ற மென்பொருள் *(kaṭṭaṟṟa meṉporuḷ)* | இலவச மென்பொருள் *(illavasa menporul)* |
| th | 泰语 | ซอฟต์แวร์เสรี *(sofotwerseri)* | |
| tl | 他加禄语（菲律宾） | malayang software | |
| tr | 土耳其语 | özgür yazilim | |
| uk | 乌克兰语 | вільне програмне забезпечення *(vil'ne prohramne zabezpechennja)* | |
| ur | 乌尔都语 | آزاد سافٹ ویئر *(azad software)* | مفت سافٹ ویئر *(muft software)* |
| vi | 越南语 | phần mềm tự do | |
| zh-cn | 中文（简体） | 自由软件 *(zi-you ruan-jian)* | 免费软件 *(mian-fei ruan-jian)* |
| zh-tw | 中文（繁体） | 自由軟體 *(zih-yo)* | 免費軟體 *(mien-fei)* |
| zu | 祖鲁语 | isoftware ekhululekile | |

# 附录 C

# 自由软件之歌



《自由软件之歌》取材自保加利亚民歌《Sadi moma bela loza》的旋律。可以到这里聆听由保加利亚乐器演奏的传统民歌版自由软件之歌 http://gnu.org/music/FreeSWSong.ogg。

这首歌是 7/8 拍的；这种不寻常的不均匀古怪节奏常常被看成是一种错误。该部分可拆分为三个单拍子，以慢—快—快或 3—2—2 拍子划分。保加利亚音乐的这种节奏，经常可以拉长，有些音乐家分析这首歌曲可以替代为 3—2—3 拍子；然而，最后的 3 拍并不如第一个 3 拍那么长。搜集并教授这种舞蹈的伊夫 · 莫罗，倾向于 7 拍节奏。

译者注：读者可下载 *Musescore* 格式的乐谱文件[1]，通过自由的乐谱软件 *Musescore*[2] 聆听。

[1]乐谱文件下载：https://github.com/beijinglug/fsfs-zh/blob/master/docs/song-book-jutta-scrunch-crop-zh.mscx

[2]可前往其官方网站下载 https://musescore.org/。GNU/Linux 发行版可通过包管理器搜索 `musescore` 来安装。

# 自由软件之歌

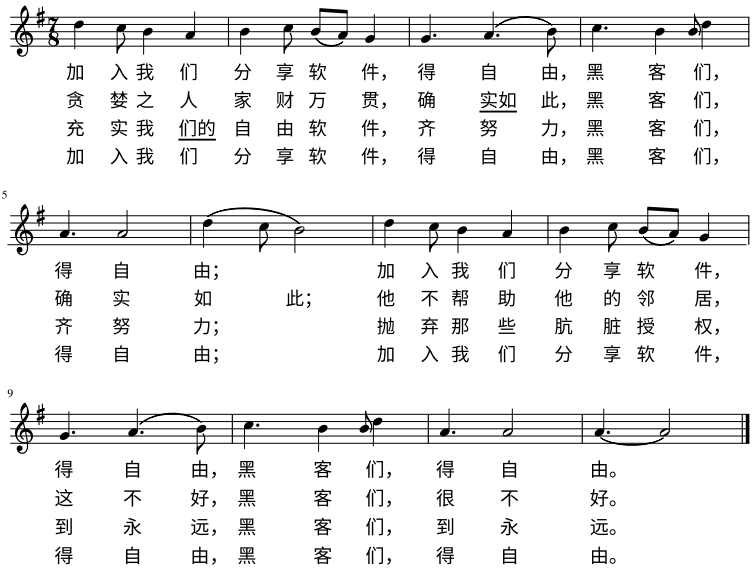


图 **C.1:** 自由软件之歌乐谱