For Want of a Printer 从一台打印机说起

> I fear the Greeks. Even when they bring gifts.\ – Virgil, *The Aeneid*
>
> 我畏惧希腊人，哪怕他们带着礼物来。\ ——维吉尔，《*埃涅阿斯纪*》

The new printer was jammed, again.

这可是台全新的打印机！怎么又卡纸了？

Richard M. Stallman, a staff software programmer at the Massachusetts Institute of Technology's Artificial Intelligence Laboratory (AI Lab), discovered the malfunction the hard way. An hour after sending off a 50-page file to the office laser printer, Stallman, 27, broke off a productive work session to retrieve his documents. Upon arrival, he found only four pages in the printer's tray. To make matters even more frustrating, the four pages belonged to another user, meaning that Stallman's print job and the unfinished portion of somebody else's print job were still trapped somewhere within the electrical plumbing of the lab's computer network.

理查德·M·斯托曼（Richard M. Stallman）刚刚发现了这令他头痛不已的卡纸问题，那年他27岁，是麻省理工学院人工智能实验室一名资深的程序员。一个小时前，他给那台激光打印机提交了一个50页的打印任务，之后，他就忙于其它事务去了。回来时，本想拿到一叠散发着油香的文件，可这会儿，他只能站在那台毫无生机打印机前，直楞楞地看着它，上面，总共才四页并非他想要的文档。换句话说，斯托曼那份50页的文件，外加别人的半份文档，全被卡住了，成了实验室网络上的两个孤魂野鬼。

Waiting for machines is an occupational hazard when you're a software programmer, so Stallman took his frustration with a grain of salt. Still, the difference between waiting for a machine and waiting on a machine is a sizable one. It wasn't the first time he'd been forced to stand over the printer, watching pages print out one by one. As a person who spent the bulk of his days and nights improving the efficiency of machines and the software programs that controlled them, Stallman felt a natural urge to open up the

machine, look at the guts, and seek out the root of the problem.

干程序员这行也是有风险的：你总得等着机器干活。不过斯托曼可是个能苦中做乐的人。再怎么说，等着机器干活与盯着机器干活，毕竟是两码事。就说现在吧，斯托曼正盯着打印机，看它慢吞吞地一页一页往外吐纸。这情景对他来说可一点也不陌生。作为一个程序员，他成天都在改进机器和控制这些机器的程序，好让它们更有效率的工作。当下，他恨不得掀开这台打印机，仔细查查究竟是哪出的毛病。

Unfortunately, Stallman's skills as a computer programmer did not extend to the mechanical-engineering realm. As freshly printed documents poured out of the machine, Stallman had a chance to reflect on other ways to circumvent the printing jam problem.

可惜，凭编程的能耐，没法解决机械工程的问题。斯托曼只能在重新打印文件的这会功夫，想点别的：能不能用别的办法，绕道解决这个卡纸的问题呢?

How long ago had it been that the staff members at the AI Lab had welcomed the new printer with open arms? Stallman wondered. The machine had been a donation from the Xerox Corporation. A cutting edge prototype, it was a modified version of a fast Xerox photocopier. Only instead of making copies, it relied on software data piped in over a computer network to turn that data into professional-looking documents. Created by engineers at the world-famous Xerox Palo Alto Research Facility, it was, quite simply, an early taste of the desktop-printing revolution that would seize the rest of the computing industry by the end of the decade.

这台打印机来人工智能实验室有多久了？斯托曼努力的回忆着。这机器是施乐公司（Xerox Corporation）捐赠的。它可是新一代产品的原型，是从上一代的复印机改进而来。和上一代比，它不再是简单地复印，而是从网络上接收数据，打印成专业品质的文档。它诞生于著名的施乐帕罗奥多研究中心（Xerox Palo Alto Research Facility）。再过大约十年，将会发生一场打印机革命，那时，许多计算机厂商都会投身其中。而这款打印机则是这场革命的先行者。

Driven by an instinctual urge to play with the best new equipment, programmers at the AI Lab promptly integrated the new machine into the lab's sophisticated computing infrastructure. The results had been immediately pleasing. Unlike the lab's old printer, the new Xerox machine was fast. Pages came flying out at a rate of one per second, turning a 20-minute print job into a

2-minute print job. The new machine was also more precise. Circles came out looking like circles, not ovals. Straight lines came out looking like straight lines, not low-amplitude sine waves.

人工智能实验室的程序员们，本能般地喜欢折腾这些新酷产品。当初他们麻利地把这打印机请回来，把它和实验室中的各种复杂的计算设备连在了一起,结果甚是喜人。比起以前的打印机，这台新打印机打印速度飞快。平均大约一秒钟打印一页纸。以前要花二十来分钟打印的东西，如今两分钟就解决了。新机器的打印精度也更高了。让它给画个圆，它决不会给你打出个鸡蛋来；让它划条直线，它也绝不会给你拐着弯瞎划拉。

It was, for all intents and purposes, a gift too good to refuse.

怎么看，这打印机都是份难以拒绝的珍贵礼物。

Once the machine was in use, its flaws began to surface. Chief among the drawbacks was the machine's susceptibility to paper jams. Engineering-minded programmers quickly understood the reason behind the flaw. As a photocopier, the machine generally required the direct oversight of a human operator. Figuring that these human operators would always be on hand to fix a paper jam, if it occurred, Xerox engineers had devoted their time and energies to eliminating other pesky problems. In engineering terms, user diligence was built into the system.

可没过几周，这机器的缺点就逐渐显现了。其中最要命的，就是卡纸问题。凭着工程师的直觉，程序员们很快就洞察出了这问题背后的原因。倘若是一台复印机，它旁边总会站个人直接操作它。遇到卡纸问题，这个人总能立即发现，着手解决，不至于有什么大影响。而施乐公司的那些工程师们则可以把时间和精力，用在其他更棘手的问题上了。用工程师的行话讲：这系统需要用户参与。

In modifying the machine for printer use, Xerox engineers had changed the user-machine relationship in a subtle but profound way. Instead of making the machine subservient to an individual human operator, they made it subservient to an entire networked population of human operators. Instead of standing directly over the machine, a human user on one end of the network sent his print command through an extended bucket brigade of machines, expecting the desired content to arrive at the targeted destination and in proper form. It wasn't until he finally went to check up on the final output that

he realized how little of it had really been printed.

如今，施乐公司在复印机的基础之上，设计了这个打印机。这个小小的变化，却使得人机关系发生了微妙而深刻的变化。现在，在同一时刻，这机器不再只服务于一个用户。它要同时给整个网络的用户提供打印服务。这些用户也不再老老实实地站在机器前面，而是坐在网络的另一端，可能还远离打印机，向这台机器发布打印命令，并指望它能按要求完成任务。等这些用户忙过其它事情回过神来，想起打印的文件时，他们才发现，大部分的内容已被一张纸卡住了。

Stallman was hardly the only AI Lab denizen to notice the problem, but he also thought of a remedy. Years before, for the lab's previous printer, Stallman had solved a similar problem by modifying the software program that regulated the printer, on a small PDP-11 machine, as well as the Incompatible Timesharing System that ran on the main PDP-10 computer. Stallman couldn't eliminate paper jams, but he could insert software code that made the PDP-11 check the printer periodically, and report jams back to the PDP-10. Stallman also inserted code on the PDP-10 to notify every user with a waiting print job that the printer was jammed. The notice was simple, something along the lines of "The printer is jammed, please fix it," and because it went out to the people with the most pressing need to fix the problem, chances were that one of them would fix it forthwith.

斯托曼是人工智能实验室中第一个发现这个问题的人，也是他第一个提出了解决方案。几年前，实验室还用着一台旧打印机，也有类似的问题。这台打印机的控制程序，运行在实验室里的一台PDP-11计算机上。那会儿，斯托曼修改了这个控制程序，解决了这个问题。当然，他没法直接修理打印机来解决卡纸的问题。他只是修改控制程序，让程序定期检查打印机是否工作正常，再把检查结果传给实验室的核心计算机——一台PDP-10。同时，他也修改了运行在PDP-10主机上的"不兼容分时系统（ITS，Incompatible Timesharing System）"，处理这些检查结果。生怕哪个粗心用户忘了去查看打印机，斯托曼还让控制程序在卡纸的时候，向所有等待打印任务的用户发送一条提醒消息。消息简短明确："打印机被卡住了，请前去维修。"收到这消息的人都是等着要用打印机的，所以问题就迎"人"而解。

As fixes go, Stallman's was oblique but elegant. It didn't fix the mechanical side of the problem, but it did the next best thing by closing the information loop between user and machine. Thanks to a few additional lines of software

code, AI Lab employees could eliminate the 10 or 15 minutes wasted each week in running back and forth to check on the printer. In programming terms, Stallman's fix took advantage of the amplified intelligence of the overall network.

斯托曼这解决方案虽然并不能说很讨巧，但却也漂亮。它没解决打印机机械部件的问题，却提出了一个次优方案：完善人机交互，及早报告问题。多亏了斯托曼加的几行代码，人工智能实验室的员工，不用再跑来跑去地查看打印机状态了。这样，每周起码节省了10到15分钟时间。用编程的术语讲，斯托曼的方案提高了网络的整体智能。

"If you got that message, you couldn't assume somebody else would fix it," says Stallman, recalling the logic. "You had to go to the printer. A minute or two after the printer got in trouble, the two or three people who got messages arrive to fix the machine. Of those two or three people, one of them, at least, would usually know how to fix the problem."

"你要是收到这条消息，你可就坐不住了。你多半等不及别人去解决卡纸问题。"斯托曼回忆着其中的逻辑，"你得亲力亲为，跑到打印机跟前。一两分钟之后，要是问题还没解决，就会聚来两三个人。这其中，总会有个人知道怎么修理。"

Such clever fixes were a trademark of the AI Lab and its indigenous population of programmers. Indeed, the best programmers at the AI Lab disdained the term programmer, preferring the more slangy occupational title of hacker instead. The job title covered a host of activities – everything from creative mirth making to the improvement of existing software and computer systems. Implicit within the title, however, was the old-fashioned notion of Yankee ingenuity. For a hacker, writing a software program that worked was only the beginning. A hacker would try to display his cleverness (and impress other hackers) by tackling an additional challenge: to make the program particularly fast, small, powerful, elegant, or somehow impressive in a clever way.

类似的巧法子是人工智能实验室的一大特色。尤其在其中的资深程序员之间，屡见不鲜。实际上，那些天才的程序员，不屑于用"程序员"这词。他们更喜欢用圈子里的行话，用"黑客"这个词。这词儿涵盖了不少内容：从抖机灵、甩包袱，到改进现有软件，优化计算机系统。而更深之处，则蕴含着旧时美国移民的智慧。对于黑客来说，有这么一条铁律：从头开发个软件只是小儿科；要想在别

人面前展现自己的聪明才智（并让别人对自己留下印象），就得做点更有意思的事：改进一个程序让它体积更小、跑得更快、功能更强，代码更优雅，这才是真本事。

Companies like Xerox made it a policy to donate their products (and software) to places where hackers typically congregated. If hackers used these products, they might go to work for the company later on. In the 60s and early 70s, they also sometimes developed programs that were useful for the manufacturer to distribute to other customers.

这条铁律，恰恰影响着施乐这样的公司，让它们乐意把自己的产品（和配套的软件）捐赠到黑客聚集的地方。要是这些黑客们对这些产品产生了兴趣，他们以后就可能有兴趣去这个公司工作。在20世纪60年代到70年代初，黑客们也常常会开发一些实用的软件，提供给公司发布给其他顾客使用。

When Stallman noticed the jamming tendency in the Xerox laser printer, he thought of applying the old fix or “hack” to this printer. In the course of looking up the Xerox laser-printer software, however, Stallman made a troubling discovery. The printer didn’t have any software, at least nothing Stallman or a fellow programmer could read. Until then, most companies had made it a form of courtesy to publish source-code files–readable text files that documented the individual software commands that told a machine what to do. Xerox, in this instance, had provided software files only in compiled, or binary, form. If programmers looked at the files, all they would see was an endless stream of ones and zeroes – gibberish.

凭着这份礼尚往来的精神，斯托曼在这台新打印机面前并不慌张。他只要找个法子，把以前那套修改方案拿过来，修理修理这机器，或者用圈子里的话说，“黑”它一下，就万事大吉了。可是，他刚找了找这台施乐激光打印机的软件，就碰了一鼻子灰。这打印机根本不提供软件，至少没提供任何给人读的程序代码。要知道，那个时代，大多数公司都会发布软件的源代码（一系列可供人阅读的文本文件，用于详细定义程序的行为）。可当下，施乐公司提供的程序，却只有编译好的可执行文件。程序员倒是可以打开这些文件，可打开之后看见的，全是零和一。除非他们乐意去翻译这些二进制信息，否则这些东西看起来只是一堆胡言乱语。

There are programs, called “disassemblers,” to convert the ones and zeroes into low-level machine instructions, but figuring out what those instructions actually “do” is a long and hard task, known as “reverse engineering.” To

reverse engineer this program could have taken more time than five years' worth of jammed printouts. Stallman wasn't desperate enough for that, so he put the problem aside.

有一种叫作"反汇编器"的程序，可以把这些看不懂的零和一转换成机器底层指令，但是要进一步了解这些指令实际在"做"的事情则是一件费时费力的工作，这样的工作被称作"反向工程"。要想把打印机的控制程序进行反向工程，也许会花去超过五年的时间。卡纸的问题还没有严重到让斯托曼愿意花这么多时间去解决，所以他把这件事放到了一边。

Xerox's unfriendly policy contrasted blatantly with the usual practices of the hacker community. For instance, to develop the program for the PDP-11 that ran the old printer, and the program for another PDP-11 that handled display terminals, the AI Lab needed a cross-assembler program to build PDP-11 programs on the PDP-10 main computer. The lab's hackers could have written one, but Stallman, a Harvard student, found such a program at Harvard's computer lab. That program was written to run on the same kind of computer, the PDP-10, albeit with a different operating system. Stallman never knew who had written the program, since the source code did not say. But he brought a copy back to the AI Lab. He then altered the source code to make it run on the AI Lab's Incompatible Timesharing System (ITS). With no muss and little fuss, the AI Lab got the program it needed for its software infrastructure. Stallman even added a few features not found in the original version, making the program more powerful. "We wound up using it for several years," Stallman says.

施乐公司看上去不太友好的软件条款与黑客社区所习惯的氛围格格不入。打个比方，如果想为一台老式的打印机开发一个在PDP-11上运行的控制程序，或为另一台PDP-11开始一个处理显示终端的程序，人工智能实验室的黑客们需要在PDP-10主机上运行一个交叉汇编的程序，才能创建出他们需要的程序。他们可以自己从头开发一个这样的交叉汇编程序，但是上哈佛大学上学的斯托曼，在哈佛的计算机实验室中找到了这样一个程序。这个程序都可以在PDP-10这种型号的计算机上运行，即使两台机器上运行的操作系统不同也没有关系。斯托曼并不需要知道这份代码是谁写的，因为源代码中并没有标明。但是他可以把这份代码带到人工智能实验室，通过修改其中的一部分代码，这个程序就可以在人工智能实验室的"不兼容分时系统（ITS）"上运行了。没花多大劲，人工智能实验室就得到了一个可以用于充实他们的软件架构的程序。斯托曼甚至还在程序中加入了一些原来没有的新功能，使程序变得更加强大了。"这个程序我们一

直用了很多年。”斯托曼说。

From the perspective of a 1970s-era programmer, the transaction was the software equivalent of a neighbor stopping by to borrow a power tool or a cup of sugar from a neighbor. The only difference was that in borrowing a copy of the software for the AI Lab, Stallman had done nothing to deprive anyone else of the use of the program. If anything, other hackers gained in the process, because Stallman had introduced additional features that other hackers were welcome to borrow back. For instance, Stallman recalls a programmer at the private engineering firm, Bolt, Beranek & Newman, borrowing the program. He made it run on Twenex and added a few additional features, which Stallman eventually reintegrated into the AI Lab’s own source-code archive. The two programmers decided to maintain a common version together, which had the code to run either on ITS or on Twenex at the user’s choice.

对于七十年代的程序员来说，把软件拷贝来拷贝去，就好比从邻居家借碗酱油那么稀松平常。所不同的是，你从邻居那拿了多少酱油，邻居就少了多少。而从别人那里拷贝一个程序出来，并不影响别人继续使用那个程序。要说有影响，也是正面的。因为斯托曼对这程序还加了些功能，别的黑客可以自由地拷来斯托曼改进的程序，这也算是计算机世界的礼尚往来了。后来，有个从Bolt, Beranek & Newman这家公司来的程序员，拷走了斯托曼的程序，他把这个程序移植到了Twenex系统上，并加上了一些新的功能，斯托曼则又把改进后的程序拷了回来，用回到人工智能实验室里。他们两人打算共同来维护一个通用的版本，这个版本既可以在ITS上运行，也可以在Twenex上运行。

“A program would develop the way a city develops,” says Stallman, recalling the software infrastructure of the AI Lab. ``Parts would get replaced and rebuilt. New things would get added on. But you could always look at a certain part and say, ‘Hmm, by the style, I see this part was written back in the early 60s and this part was written in the mid-1970s.’

”

“开发一个程序就好比开发一座城市。”斯托曼回忆着人工智能实验室的软件设施：“有些部分会被换掉，有些得重建翻修。新的东西会逐步加进来。可如果你仔细看某个部分，也许会感慨道：‘这块从风格看应该是六十年代早期的；之后的部分估计是七十年代中期完成的。’”

Through this simple system of intellectual accretion, hackers at the AI Lab and

other places built up robust creations. Not every programmer participating in this culture described himself as a hacker, but most shared the sentiments of Richard M. Stallman. If a program or software fix was good enough to solve your problems, it was good enough to solve somebody else's problems. Why not share it out of a simple desire for good karma?

这套简洁的系统得以让知识逐渐积累。这种积累为发明和创造提供了稳固的基石。人工智能实验室以及世界各地的黑客们，都得益于这一系统。参与到这个文化小圈子的程序员，不一定都把自己称为黑客。不过绝大多数人都有着理查德·M·斯托曼那般的情结：要是一个程序或者一个补丁可以解决你自己的问题，它没准也能帮到别人。独乐乐不如众乐乐，何不分享给大家，起码也算积德行善了。

This system of cooperation was being undermined by commercial secrecy and greed, leading to peculiar combinations of secrecy and cooperation. For instance, computer scientists at UC Berkeley had built up a powerful operating system called BSD, based on the Unix system they had obtained from AT&T. Berkeley made BSD available for the cost of copying a tape, but would only give these tapes to schools that could present a \$50,000 source license obtained from AT&T. The Berkeley hackers continued to share as much as AT&T let them, but they had not perceived a conflict between the two practices.

这种合作的精神不断的被商业秘密限制和贪婪之心所侵蚀，变成了一种商业秘密和合作精神的古怪搭配。加州大学伯克利分校的计算机科学家们基于他们从AT&T得到的Unix系统开发了一个名为BSD的操作系统。伯克利向那些已经向AT&T支付了50000美元软件许可证费用的学院提供BSD系统，仅仅只收取复制磁带的成本费。伯克利的黑客们还是尽可能的发扬分享合作的精神，但只能在AT&T所允许的范围内，并且他们还没有意识这两种不同的模式之间所存在的巨大冲突。

Likewise, Stallman was annoyed that Xerox had not provided the source-code files, but not yet angry. He never thought of asking Xerox for a copy. "They had already given us the laser printer," Stallman says. "I could not say they owed us something more. Besides, I took for granted that the absence of source code reflected an intentional decision, and that asking them to change it would be futile."

施乐公司不分享源代码的行为虽然让斯托曼感到烦恼，不过倒也没激怒到他。

斯托曼在找源代码的过程中，甚至都没想过去联系施乐公司：“人家都送咱打印机了，何必还非得兴师动众地直接管他们要代码呢？再说了，我敢肯定他们不发布源代码是有一种有意的行为，非要找他们要代码也是徒劳的。”

Good news eventually arrived: word had it that a scientist at the computer-science department at Carnegie Mellon University had a copy of the laser printer source code.

好消息最终还是来了：一名卡耐基梅隆大学计算机科学系的教授手头有一份这台激光打印机的源代码。

The association with Carnegie Mellon did not augur well. In 1979, Brian Reid, a doctoral student there, had shocked the community by refusing to share his text-formatting program, dubbed Scribe. This text formatter was the first to have mark-up commands oriented towards the desired semantics (such as “emphasize this word” or “this paragraph is a quotation”) rather than low-level formatting details (“put this word in italics” or “narrow the margins for this paragraph”). Instead Reid sold Scribe to a Pittsburgh-area software company called Unilogic. His graduate-student career ending, Reid says he simply was looking for a way to unload the program on a set of developers that would take pains to keep it from slipping into the public domain. (Why one would consider such an outcome particularly undesirable is not clear.) To sweeten the deal, Reid also agreed to insert a set of time-dependent functions – “time bombs” in software-programmer parlance – that deactivated freely copied versions of the program after a 90-day expiration date. To avoid deactivation, users paid the software company, which then issued a code that defused the internal time-bomb anti-feature.

斯托曼对卡耐基梅隆大学并没有什么好印象。1979年，一位在卡耐基梅隆大学的计算机博士研究生，名叫布莱恩·瑞德（Brian Reid）。他开发了一个很不错的文本排版软件，名叫Scribe。但是他却拒绝公开这个软件的代码，震惊了整个开发社区。这个文本排版软件是世界上第一个可以通过标记来指定特定语意（比如“强调这个单词”或“这是一段引文”）的排版软件，而在它之前，一般的排版软件都是通过低级的排版命令（比如“把这个词改为斜体“或”缩小这一段的页边距“）来表达类似的需求。瑞德决定把Scribe出售给一家匹兹堡名叫Unilogic的公司。他当时刚好博士毕业，正打算把开发Scribe的任务转手给别人，免得它流落到公共领域（Public Domain）里。（为什么他会做出这样的决定，迄今为止还令人费解。）为了让这笔交易更具吸引力，瑞德在Scribe中动了点小手

脚：他在里面放了个“定时炸弹”，让用户有90天的免费试用期，90天一过，如果用户不交费，则不能再使用这个软件。

For Stallman, this was a betrayal of the programmer ethos, pure and simple. Instead of honoring the notion of share-and-share alike, Reid had inserted a way for companies to compel programmers to pay for information access. But he didn’t think deeply about the question, since he didn’t use Scribe much.

在斯托曼看来，这样的做法背叛了程序员应有单纯、简单的精神气质。瑞德的做法不但没有弘扬分享再传播的精神，而且竟然还助纣为虐，开辟了一条让软件公司强迫程序员为信息付费的新途径。不过斯托曼也没有太在意这个问题，因为他本人并不太使用Scribe这个软件。

Unilogic gave the AI Lab a gratis copy to use, but did not remove or mention the time bomb. It worked, for a while; then one day a user reported that Scribe had stopped working. System hacker Howard Cannon spent hours debugging the binary until he found the time-bomb and patched it out. Cannon was incensed, and wasn’t shy about telling the other hackers how mad he was that Unilogic had wasted his time with an intentional bug.

Unilogic提供了一个免费的版本给人工智能实验室使用，但是这个版本中仍然保留了那个“定时炸弹”。这个软件在实验室中正常使用了一段时间，但是有一天，有用户发现它不能正常工作了。一位名叫霍华德·坎农（Howard Cannon）的系统工程师花了几个小时调试了这个软件，最终发现了隐藏在软件内部的“定时炸弹”，他改动了一些代码，把它绕了过去。坎农对此非常愤慨，遇到人就抱怨说Unilogic浪费了他好几个小时的时间去解决一个故意隐藏在软件中的问题。

Stallman had a Lab-related reason, a few months later, to visit the Carnegie Mellon campus. During that visit, he made a point of looking for the person reported to have the printer software source code. By good fortune, the man was in his office.

几个月后，斯托曼因为工作原因，需要去卡耐基梅隆大学出差访问。访问过程中，他在计算机系找到了那位据说拥有激光打印机软件代码的教授。运气不错，那天他还刚好就在办公室。

In true engineer-to-engineer fashion, the conversation was cordial but blunt. After briefly introducing himself as a visitor from MIT, Stallman requested a copy of the laser-printer source code that he wanted to modify. To his chagrin,

the researcher refused.

工程师之间的讨论总是开门见山的。几句寒暄之后，斯托曼就直入主题，说明自己是为了激光打印机的控制程序代码而来，有了这份代码，他就可以对此进行一些改进。可令斯托曼感到吃惊的是，这位教授竟然拒绝了他的要求。

“He told me that he had promised not to give me a copy,” Stallman says.

“他跟我说他答应公司不能泄露代码。”斯托曼说。

Memory is a funny thing. Twenty years after the fact, Stallman’s mental history tape is blank in places. Not only does he not remember the motivating reason for the trip or even the time of year during which he took it, he also has no recollection of who was on the other end of the conversation. According to Reid, the person most likely to have fielded Stallman’s request is Robert Sproull, a former Xerox PARC researcher and current director of Sun Laboratories, a research division of the computer-technology conglomerate Sun Microsystems. During the 1970s, Sproull had been the primary developer of the laser-printer software in question while at Xerox PARC. Around 1980, Sproull took a faculty research position at Carnegie Mellon where he continued his laser-printer work amid other projects.

记忆总是这么有趣。二十年过去了，斯托曼关于这段经历的回忆大部分都是空白。他都不记得当初为了什么事去的卡耐基梅隆大学，连具体哪年都给忘了。他也不记得那位教授是谁。根据Scribe的作者瑞德的回忆，惹到斯托曼的那位很可能是罗伯特·斯布鲁（Robert Sproull）。他曾是施乐公司PARC研究所的研究员，如今在Sun研究所任部门主管。二十世纪七十年代，斯布鲁在施乐公司PARC研究所负责激光打印机程序的主要开发者。八十年代他拿到了卡耐基梅隆大学的教职，并在那里继续他的激光打印机相关的工作。

When asked directly about the request, however, Sproull draws a blank. “I can’t make a factual comment,” writes Sproull via email. “I have absolutely no recollection of the incident.”

可是，直接给斯布鲁发邮件，询问当初的事情，他却不记得这事了：“实在是无可奉告。”斯布鲁在邮件中写道：“我确实没记得有这么一件事。”

“The code that Stallman was asking for was leading-edge, state-of-the-art code that Sproull had written in the year or so before going to Carnegie Mellon,” recalls Reid. If so, that might indicate a misunderstanding that

occurred, since Stallman wanted the source for the program that MIT had used for quite some time, not some newer version. But the question of which version never arose in the brief conversation.

“斯托曼想要的代码包含的可都是前沿技术，那可是斯布鲁去卡耐基梅隆之前，花了几年的时间研发的成果。”瑞德回忆道。如果确实是这样，那可是一个重大的误解，因为斯托曼想要的代码是麻省理工学院已经用了很长一段时间的打印机的程序代码，并不是要的最新版本。不过在他们那次简短的交谈中，也许根本都还没来得及谈及是要的哪个版本。

In talking to audiences, Stallman has made repeated reference to the incident, noting that the man’s unwillingness to hand over the source code stemmed from a nondisclosure agreement, a contractual agreement between him and the Xerox Corporation giving the signatory access to the software source code in exchange for a promise of secrecy. Now a standard item of business in the software industry, the nondisclosure agreement, or NDA, was a novel development at the time, a reflection of both the commercial value of the laser printer to Xerox and the information needed to run it. “Xerox was at the time trying to make a commercial product out of the laser printer,” recalls Reid. “They would have been insane to give away the source code.”

在后来的各种演讲中，斯托曼一再提及这件事。强调斯布鲁由于签署了保密协议，从而不能泄露源代码。如今在各个IT公司，签署保密协议（NDA）已经非常普遍，而在当时为软件代码签署保密协议还算先例。这也间接地反映出施乐公司对激光打印机项目的重视程度。“施乐公司打算把激光打印机做成商业产品。”瑞德说，“他们除非疯了，才会把代码泄露出去。”

For Stallman, however, the NDA was something else entirely. It was a refusal on the part of some CMU researcher to participate in a society that, until then, had encouraged software programmers to regard programs as communal resources. Like a peasant whose centuries-old irrigation ditch had grown suddenly dry, Stallman had followed the ditch to its source only to find a brand-spanking-new hydroelectric dam bearing the Xerox logo.

但在斯托曼看来，签署这种保密协议完全是一种自私的行径。那会儿大家都把软件认为是一种共有的资源，卡耐基梅隆大学一些研究员的做法，就意味着否定这个公理。这对斯托曼来说，就好比是一个农夫眼看着用来灌溉庄稼的河流干枯了。循着河道往上巡查，竟然看到一道大坝从天而降，拦住河水。大坝上赫然印着几个大字：施乐公司。

For Stallman, the realization that Xerox had compelled a fellow programmer to participate in this newfangled system of compelled secrecy took a while to sink in. In the first moment, he could only see the refusal in a personal context. “I was so angry I couldn’t think of a way to express it. So I just turned away and walked out without another word,” Stallman recalls. “I might have slammed the door. Who knows? All I remember is wanting to get out of there. I went to his office expecting him to cooperate, so I had not thought about how I would respond if he refused. When he did, I was stunned speechless as well as disappointed and angry.”

现在来看，那时施乐公司用各种花言巧语，把一位程序员同胞带入了这片新天地，与世隔绝。不过一开始，斯托曼也没把矛头指向施乐公司，而是怪罪到了个人性格上。“我当时非常生气，都不知道怎么表达。我二话没说，扭头就走了。”斯托曼说，“没准我还使劲摔了门，谁知道呢。我能记得的，就是当时想赶快离开。我特地拜访他的原因就是希望能与他合作，所以我完全没想到过会被拒绝。当他拒绝我的时候，我觉得无话可说，非常的失望与生气。”

Twenty years after the fact, the anger still lingers, and Stallman presents the event as one that made him confront an ethical issue, though not the only such event on his path. Within the next few months, a series of events would befall both Stallman and the AI Lab hacker community that would make 30 seconds worth of tension in a remote Carnegie Mellon office seem trivial by comparison. Nevertheless, when it comes time to sort out the events that would transform Stallman from a lone hacker, instinctively suspicious of centralized authority, to a crusading activist applying traditional notions of liberty, equality, and fraternity to the world of software development, Stallman singles out the Carnegie Mellon encounter for special attention.

二十多年过去了，可斯托曼当初的怨气还在。他甚至把这个事件描述为人生转折点。然而，那之后几个月中，在人工智能实验室以及斯托曼身上发生的各种事，却比这次打印机事件还令人难以接受。斯托曼，本是一名孤独的黑客，本能地对绝对权威有戒心。在经历了这一系列事件后，变成了一位斗士，把传统的自由、平等、博爱的精神引入软件开发领域。而这次的打印机事件，在其一生的无数事件中则最值得一书。

“It was my first encounter with a nondisclosure agreement, and it immediately taught me that nondisclosure agreements have victims,” says Stallman, firmly. “In this case I was the victim. [My lab and I] were victims.”

“这是我第一次遭遇所谓的保密协议，但这马上让我明白，在保密协议面前，总会有一些无辜受害者。”斯托曼坚定的说，“在这次事件中，我就是那个受害者。人工智能实验室和我本人都扮演了受害者的角色。”

Stallman later explained, “If he had refused me his cooperation for personal reasons, it would not have raised any larger issue. I might have considered him a jerk, but no more. The fact that his refusal was impersonal, that he had promised in advance to be uncooperative, not just to me but to anyone whatsoever, made this a larger issue.”

斯托曼后来解释说：“如果他只是因为个人原因拒绝与我合作，也许事情也就这样不了了之了。也许我会觉得他是一个蠢蛋，但也就仅此而已。但是事实上，他并不是因为个人原因而拒绝我，是因为他先前已经向公司保证了不会泄露源代码，不但不能把源代码给我，也不能给其它任何人，那这个事情性质就变得严重了。”

Although previous events had raised Stallman’s ire, he says it wasn’t until his Carnegie Mellon encounter that he realized the events were beginning to intrude on a culture he had long considered sacrosanct. He said, “I already had an idea that software should be shared, but I wasn’t sure how to think about that. My thoughts weren’t clear and organized to the point where I could express them in a concise fashion to the rest of the world. After this experience, I started to recognize what the issue was, and how big it was.”

尽管之前也有过类似的不快经历，可都不能与这次的打印机事件相提并论。这次事件让斯托曼彻底意识到，这些一系列的事件，正悄悄地侵蚀自己所珍视的文化——那个神圣不可侵犯的“黑客”小圈子的核心文化。他说：“我一直有个信念，那就是软件应该可以可以被分享的，但是我不太确定应该怎么去把它变成现实。我的想法还不太成熟，也不太系统，所以没有办法把它精确的表达出来，让其它人理解。经历了这些事件，我开始渐渐认识到问题的本质和它的严重性。”

As an elite programmer at one of the world’s elite institutions, Stallman had been perfectly willing to ignore the compromises and bargains of his fellow programmers just so long as they didn’t interfere with his own work. Until the arrival of the Xerox laser printer, Stallman had been content to look down on the machines and programs other computer users grimly tolerated.

作为一个世界顶级研究机构的顶级程序员，斯托曼之前一直都无视那些程序员

同行所作的各种妥协和让步，因为他们还不至于影响到斯托曼的工作。而如今施乐激光打印机的到来，让斯托曼开始注意到其他计算机用户一直在忍受的程序和机器。

Now that the laser printer had insinuated itself within the AI Lab's network, however, something had changed. The machine worked fine, barring the paper jams, but the ability to modify software according to personal taste or community need had been taken away. From the viewpoint of the software industry, the printer software represented a change in business tactics. Software had become such a valuable asset that companies no longer accepted the need to publicize source code, especially when publication meant giving potential competitors a chance to duplicate something cheaply. From Stallman's viewpoint, the printer was a Trojan Horse. After a decade of failure, software that users could not change and redistribute – future hackers would use the term "proprietary" software – had gained a foothold inside the AI Lab through the sneakiest of methods. It had come disguised as a gift.

如今，这台激光打印机已经强势入驻到人工智能实验室了，而外面的世界也悄悄发生着变化。除了偶尔卡纸以外，打印机工作还算正常。可是按照个人喜好修改软件则不可能了。从软件业的角度看，这打印机的出现是个信号，预示着软件是公司的重要财产。谁也不再愿意发布软件源代码，因为这可能会给潜在竞争对手机会，让他们可以轻松山寨自己的产品。而在斯托曼看来，这打印机简直就是个卧底。十来年的尝试，私有软件总算在人工智能实验室中占得一角。私有软件，也就是如今所说的专有软件，这次被打扮成礼物，潜入得悄无声息。

That Xerox had offered some programmers access to additional gifts in exchange for secrecy was also galling, but Stallman takes pains to note that, if presented with such a quid pro quo bargain at a younger age, he just might have taken the Xerox Corporation up on its offer. The anger of the Carnegie Mellon encounter, however, had a firming effect on Stallman's own moral lassitude. Not only did it give him the necessary anger to view such future offers with suspicion, it also forced him to turn the situation around: what if a fellow hacker dropped into Stallman's office someday and it suddenly became Stallman's job to refuse the hacker's request for source code?

施乐公司后来还发出邀请，让一些程序员再使用它们的礼品。斯托曼说，要是再早几年，他没准也无法拒绝这种免费午餐。是那次打印机事件，让斯托曼建

立起了道德防线。它不仅燃起了斯托曼久抑的怒火，还让他对以后的各种礼品心存戒备；它更让斯托曼开始思考一个让他自己也坐立不安的问题：要是以后哪个黑客同行进到自己的办公室，向他索要代码，他究竟会不会拒绝呢?

“When somebody invited me to betray all my colleagues in that way, I remembered how angry I was when somebody else had done that to me and my whole lab,” Stallman says. ``So I said, ‘Thank you very much for offering me this nice software package, but I can’t accept it on the conditions that you’re asking for, so I’m going to do without it.’

”

“要是谁跟我建议，让我以这种方式背叛我的同事，我就会回忆起在打印机事件中，我和我的同事，被别人背叛的感觉，那份怒火终生难忘。”斯托曼说，“我会回敬给分发专有软件的人：‘谢谢你的软件，非常棒！可是我不能接受你的那些保密协议，很遗憾，我不会用它。’”

It was a lesson Stallman would carry with him through the tumultuous years of the 1980s, a decade during which many of his MIT colleagues would depart the AI Lab and sign nondisclosure agreements of their own. They may have told themselves that this was a necessary evil so they could work on the best projects. For Stallman, however, the NDA called the the moral legitimacy of the project into question. What good is a technically exciting project if it is meant to be withheld from the community?

斯托曼带着这种态度，经历了动荡的八十年代。在这期间，麻省理工学院的同事们纷纷离开人工智能实验室，走进公司，签署了保密协议。大多数保密协议都有解密时间，而这则成了很多黑客们的借口。他们会辩解说：软件迟早会成为公共资源。只要保证软件在早期的开发阶段不被泄露，就可以保证让各位黑客朋友们可以进入到顶尖项目中工作。这些借口，在斯托曼看来，是迈向深渊的第一步。把一个出色的项目放到社区以外去开去，能有什么好处可言呢?

As Stallman would quickly learn, refusing such offers involved more than personal sacrifice. It involved segregating himself from fellow hackers who, though sharing a similar distaste for secrecy, tended to express that distaste in a more morally flexible fashion. Refusing another’s request for source code, Stallman decided, was not only a betrayal of the scientific mission that had nurtured software development since the end of World War II, it was a violation of the Golden Rule, the baseline moral dictate to do unto others as

you would have them do unto you.

斯托曼很快就会明白，要拒绝这些要求和邀请，不仅需要一些个人牺牲，更会被其他一些黑客们疏远。这些黑客虽然也对各种保密协议嗤之以鼻，但却会更圆滑地对它加以评判。拒绝提供源代码，在斯托曼看来，不仅违背了二战以来深植入软件开发中的科学精神，更违背了“己所不欲，勿施于人”的道德准则。

Hence the importance of the laser printer and the encounter that resulted from it. Without it, Stallman says, his life might have followed a more ordinary path, one balancing the material comforts of a commercial programmer with the ultimate frustration of a life spent writing invisible software code. There would have been no sense of clarity, no urgency to address a problem others weren't addressing. Most importantly, there would have been no righteous anger, an emotion that, as we soon shall see, has propelled Stallman's career as surely as any political ideology or ethical belief.

打印机事件的重要意义恰在于此。正如斯托曼所言，倘若没有这次事件，他的人生也许就会落入平常，纠结着，一边开发专有软件，一边痛苦地编写没人会看到的代码。当然也不会有着如今清晰的思路，更不会去解决别人沿未想过的各种问题。最重要的，他心中也不会再有那份不平，推动着他去追求他的政治理想和道德信仰。

“From that day forward, I decided this was something I could never participate in,” says Stallman, alluding to the practice of trading personal liberty for the sake of convenience – Stallman's description of the NDA bargain – as well as the overall culture that encouraged such ethically suspect deal-making in the first place. “I decided never to make other people victims as I had been a victim.”

“从那日起，我决定绝不参与其中。”谈起软件保密协议和类似的事情，斯托曼如是说。在他看来，这是一场以个人自由换取便利的交易。“我决定绝不让第二个人为此成为像我一样的受害者。”

2001: A Hacker's Odyssey 黑客路漫漫

The New York University computer-science department sits inside Warren

Weaver Hall, a fortress-like building located two blocks east of Washington Square Park. Industrial-strength air-conditioning vents create a surrounding moat of hot air, discouraging loiterers and solicitors alike. Visitors who breach the moat encounter another formidable barrier, a security check-in counter immediately inside the building's single entryway.

纽约大学计算机学院坐落在沃伦·韦弗大楼（Warren Weaver Hall）之中，和著名的华盛顿广场公园相隔两个街区。大楼周围充斥着从空调压缩机里传出来的股股热浪。这阵热浪好似屏障一般，驱散着各路闲杂人等。倘若冲破这道屏障，面前还会出现另一道防线：整座大楼的唯一一道入口，你被一个安检岗拦下。

Beyond the security checkpoint, the atmosphere relaxes somewhat. Still, numerous signs scattered throughout the first floor preach the dangers of unsecured doors and propped-open fire exits. Taken as a whole, the signs offer a reminder: even in the relatively tranquil confines of pre-September 11, 2001, New York, one can never be too careful or too suspicious.

走过安检口，气氛似乎缓和了些。可依旧还能看到各处林立的警示牌，指向消防出口。一路下来，你会有这样的感受：哪怕是在2001年9月初的纽约，这样一座现代化的都市之中，也要记得，小心驶得万年船。

The signs offer an interesting thematic counterpoint to the growing number of visitors gathering in the hall's interior atrium. A few look like NYU students. Most look like shaggy-haired concert-goers milling outside a music hall in anticipation of the main act. For one brief morning, the masses have taken over Warren Weaver Hall, leaving the nearby security attendant with nothing better to do but watch Ricki Lake on TV and shrug her shoulders toward the nearby auditorium whenever visitors ask about "the speech."

可当你继续向前，走到中央大厅的时候，这份谨慎的感受却被一群人冲破了。他们之中，有些看起来像是纽约大学的学生。大部分则蓬着头，好似等待着一场盛大的音乐会。在这短暂的上午，这群人占据了沃伦·韦弗大楼。这下子，安检人员反倒闲下来了。他们不规矩的斜躺坐在椅子上，看着电视剧。要是哪位访客，问起"演讲"的事，这位保安就会连话也懒得说一句，干脆冲着旁边的大礼堂一耸肩，然后继续看他的电视剧。

Once inside the auditorium, a visitor finds the person who has forced this temporary shutdown of building security procedures. The person is Richard M.

Stallman, founder of the GNU Project, original president of the Free Software Foundation, winner of the 1990 MacArthur Fellowship, winner of the Association of Computing Machinery's Grace Murray Hopper Award (also in 1990), corecipient of the Takeda Foundation's 2001 Takeda Award for Social/Economic Betterment, and former AI Lab hacker. As announced over a host of hacker-related web sites, including the GNU Project's own http://www.gnu.org site, Stallman is in Manhattan, his former hometown, to deliver a much anticipated speech in rebuttal to the Microsoft Corporation's recent campaign against the GNU General Public License.

有如此神通，能让保安放半天假的这位演讲者，此时正坐在大礼堂中。他就是理查德·M·斯托曼：GNU工程的创始人，自由软件基金会的第一任主席，1990年麦克阿瑟奖（MacArthur Fellowship）获得者，同年美国计算机协会（Association of Computing Machinery）格雷斯·莫瑞·霍普奖（Grace Murray Hopper Award）获得者，2001年日本武田基金会（Takeda Foundation）的武田奖（Takeda Award）共同获得者。当然，他也是我们熟悉的那位麻省理工学院人工智能实验室的黑客。前阵子黑客相关的网站上，包括GNU工程的官方网站上（http://www.gnu.org），都刊载了一条新闻：理查德·M·斯托曼将在他的家乡，纽约曼哈顿，发表演讲，回应微软在前些时候关于GNU通用公共许可证（GNU General Public License）的抨击。

The subject of Stallman's speech is the history and future of the free software movement. The location is significant. Less than a month before, Microsoft senior vice president Craig Mundie appeared at the nearby NYU Stern School of Business, delivering a speech blasting the GNU General Public License, or GNU GPL, a legal device originally conceived by Stallman 16 years before. Built to counteract the growing wave of software secrecy overtaking the computer industry – a wave first noticed by Stallman during his 1980 troubles with the Xerox laser printer – the GPL has evolved into a central tool of the free software community. In simplest terms, the GPL establishes a form of communal ownership – what today's legal scholars now call the "digital commons" – through the legal weight of copyright. The GPL makes this irrevocable; once an author gives code to the community in this way, that code can't be made proprietary by anyone else. Derivative versions must carry the same copyright license, if they use a substantial amount of the original source code. For this reason, critics of the GPL have taken to calling it a "viral" license, suggesting inaccurately that it spreads itself to every software program it touches.

斯托曼演讲的主题，是关于自由软件运动的历史和未来。这次演讲的地点尤为重要。不到一个月前，微软的副总裁克雷格·蒙迪就是出现在纽约大学斯特恩商学院（Stern School of Business），抨击GNU通用公共许可证（GNU General Public License）——简称GPL。GPL是斯托曼16年前想出来的法律武器，用来对抗工业界中越来越盛行的专有软件。1980年，斯托曼经历了那次施乐打印机事件之后，预感到了软件专有化的潮流逐渐到来。为了抗衡这潮流，斯托曼提出了自由软件的概念：即用户可以自由地使用，学习，修改和再发布的软件。如今，GPL已经俨然成为了自由软件社区的核心工具。简单说来，GPL是一个软件使用协议，它利用版权法，将自由软件锁定在公众可以自由使用和修改的领域。一旦锁定，这个软件就不会再被专有化。不仅仅是这个软件本身被锁定为自由软件，这个软件的任何衍生品也会成为自由软件。也就是说，倘若某个软件以GPL形式授权发布，这个软件以及任何它的衍生品，都可以被用户自由使用和修改。所谓一个软件的衍生品，也就是任何使用了该软件的代码的作品。哪怕一个软件仅仅使用了某个GPL授权软件的一小部分代码，这个软件也将被要求以自由软件形式发布。恰恰是这个原因，软件业的很多人把GPL称为病毒式许可证，因为它像病毒一般，“感染”所有它触及到的程序。

In an information economy increasingly dependent on software and increasingly beholden to software standards, the GPL has become the proverbial “big stick.” Even companies that once derided it as “software socialism” have come around to recognize the benefits. Linux, the kernel developed by Finnish college student Linus Torvalds in 1991, is licensed under the GPL, as are most parts of the GNU system: GNU Emacs, the GNU Debugger, the GNU C Compiler, etc. Together, these tools form the components of the free software GNU/Linux operating system, developed, nurtured, and owned by the worldwide hacker community. Instead of viewing this community as a threat, high-tech companies like IBM, Hewlett Packard, and Sun Microsystems have come to rely upon it, selling software applications and services built to ride atop the ever-growing free software infrastructure.

随着信息产业的发展，全球越来越依赖软件和软件标准。在这样的环境下，谁都无法忽视GPL。哪怕曾经嘲笑过GPL的公司，也不能再把它视为空中楼阁。因为越来越多的软件都是以GPL形式授权：Linux，一个最初由芬兰大学生林纳斯·托瓦兹（Linus Torvalds）于1991年开发出的类UNIX操作系统内核；GNU Emacs；GNU调试器；GNU编译器等等都是以GPL授权的。这些工具在一起，形成了一个完整的自由操作系统。世界各地的黑客，为这套操作系统贡献着代码。每个黑客也可以自由地拥有这样一套操作系统。如今，很多计算机公司都不再把这样一套自由操作系统视为威胁。相反，IBM，惠普，Sun等公司都依赖

这个操作系统，并在这套系统之上开发和出售自己的软件产品。

They've also come to rely upon it as a strategic weapon in the hacker community's perennial war against Microsoft, the Redmond, Washington-based company that has dominated the PC-software marketplace since the late 1980s. As owner of the popular Windows operating system, Microsoft stands to lose the most in an industry-wide shift to the GPL license. Each program in the Windows colossus is covered by copyrights and contracts (End User License Agreements, or EULAs) asserting the proprietary status of the executable, as well as the underlying source code that users can't get anyway. Incorporating code protected by the "viral" GPL into one of these programs is forbidden; to comply with the GPL's requirements, Microsoft would be legally required to make that whole program free software. Rival companies could then copy, modify, and sell improved versions of it, taking away the basis of Microsoft's lock over the users.

在黑客这个圈子里，这个操作系统也被当作战略武器，对抗微软公司。这个总部坐落于华盛顿州雷德蒙德市的软件公司，从二十世纪八十年代开始，已经俨然成了软件业的垄断巨头。作为Windows操作系统的拥有者，微软公司面对业界同行转向GPL许可证的事实，已经再也坐不住了。Windows中，几乎每行代码都是私有的。至少按照法律条款看，它是归微软所有。倘若有谁不慎在Windows中放入一点GPL授权的代码，整个Windows产品都会被"感染"为自由软件。在微软看来，这就好比把哪吒请进了龙宫，整个公司都得翻江倒海，它必须把整个操作系统的代码都以自由软件的形式发布。竞争对手就可以自由的复制、修改这个系统，并销售修改后的版本，这可能瞬间颠覆微软公司龙头老大的地位。

Hence the company's growing concern over the GPL's rate of adoption. Hence the recent Mundie speech blasting the GPL and the "open source" approach to software development and sales. (Microsoft does not even acknowledge the term "free software," preferring to use its attacks to direct attention towards the apolitical "open source" camp described in , and away from the free software movement.) And hence Stallman's decision to deliver a public rebuttal to that speech on the same campus here today.

因此，微软极度关注GPL的普及情况。也正因如此，蒙迪才来纽约大学，抨击GPL以及"开源"软件的开发销售模式。（微软并不承认"自由软件"这个概念，而是习惯把矛头指向章节中即将谈到的"开源"阵营，回避自由软件运动的存在。）

这才引得斯托曼也决定来纽约大学，在同一所校园内，回应蒙迪的抨击。

20 years is a long time in the software industry. Consider this: in 1980, when Richard Stallman was cursing the AI Lab's Xerox laser printer, Microsoft, which dominates the worldwide software industry, was still a privately held startup. IBM, the company then regarded as the most powerful force in the computer hardware industry, had yet to introduce its first personal computer, thereby igniting the current low-cost PC market. Many of the technologies we now take for granted – the World Wide Web, satellite television, 32-bit video-game consoles – didn't even exist. The same goes for many of the companies that now fill the upper echelons of the corporate establishment, companies like AOL, Sun Microsystems, Amazon.com, Compaq, and Dell. The list goes on and on.

二十年的时间，对软件业来说，可是个不短的年头。遥想二十多年前的1980年，斯托曼还在咒骂着人工智能实验室的施乐打印机；微软，这个被当今黑客视为全球软件业巨头的公司，还是个私人的创业小公司；而IBM，这个被当年的黑客视为全球计算机业巨头的公司，还没推出个人计算机；至于IBM的个人计算机推动了整个计算机业的发展，则也是之后的事情了。而当今我们习以为常的很多技术，在那时都还没出现。包括WWW网络，卫星电视，32位终端游戏机等等。如今的很多计算机巨头，则也一样在当时还没建立。包括AOL，Sun，亚马逊，康柏，戴尔等等。

Among those who value progress above freedom, the fact that the high-technology marketplace has come so far in such little time is cited both for and against the GNU GPL. Some argue in favor of the GPL, pointing to the short lifespan of most computer hardware platforms. Facing the risk of buying an obsolete product, consumers tend to flock to companies with the best long-term survival. As a result, the software marketplace has become a winner-take-all arena. The proprietary software environment, they say, leads to monopoly abuse and stagnation. Strong companies suck all the oxygen out of the marketplace for rival competitors and innovative startups.

高科技产业在这短暂时间内迅速成长的过程中，则充斥着关于GPL的争论。GPL的支持者声称：由于计算机硬件平台的短暂寿命，为了避免买到过时产品，用户会倾向于购买大品牌的产品。由此带来的结果，造成了软件市场赢者通吃的局面。而如今，由于软件被认为是专有的，具有垄断地位的公司就会滥用它的权利，使得整个产业停滞不前。垄断企业会堵住所有去路，让竞争者无

法生存，也让后起的创业公司无处立足。

Others argue just the opposite. Selling software is just as risky, if not more risky, than buying software, they say. Without the legal guarantees provided by proprietary software licenses, not to mention the economic prospects of a privately owned “killer app” (i.e., a breakthrough technology that launches an entirely new market), companies lose the incentive to participate. Once again, the market stagnates and innovation declines. As Mundie himself noted in his May 3rd address on the same campus, the GPL’s “viral” nature “poses a threat” to any company that relies on the uniqueness of its software as a competitive asset. Added Mundie:

> It also fundamentally undermines the independent commercial software sector because it effectively makes it impossible to distribute software on a basis where recipients pay for the product rather than just the cost of distribution.

GPL的反对者，则恰恰相反。他们声称销售软件和购买软件一样，具有风险。倘若没有法律保证软件可以被私有化，人们将再也看不到杀手级应用的繁荣（所谓杀手级应用，即某种技术足以打开一片全新市场），公司也将失去继续创新的动力。正如蒙迪于五月三日在纽约大学所说的，GPL的“病毒”特性为各大公司带来了威胁，威胁着它们赖以生存的软件产品。蒙迪道：

> 它严重地威胁着独立的商业软件。出售软件不仅仅是要收回发行成本，还有更多的人力附加价值。而GPL则让这种买卖变成天方夜谭。

The mutual success of GNU/Linux and Windows over the last 10 years suggests that both sides on this question are sometimes right. However, free software activists such as Stallman think this is a side issue. The real question, they say, isn’t whether free or proprietary software will succeed more, it’s which one is more ethical.

在过去的10年中，无论是GNU/Linux还是Windows，都有了长足的进步，赢得了各自的成功。两者双赢的局面，似乎使得无论支持或反对GPL都有了足够的理由。但是，作为像斯托曼这样的自由软件活动家来说，成功并不是事情的关键。真正的问题并不是自由软件和私有软件哪个更为成功，而是哪个更道德。

Nevertheless, the battle for momentum is an important one in the software industry. Even powerful vendors such as Microsoft rely on the support of third-

party software developers whose tools, programs, and computer games make an underlying software platform such as Windows more attractive to the mainstream consumer. Citing the rapid evolution of the technology marketplace over the last 20 years, not to mention his own company's impressive track record during that period, Mundie advised listeners to not get too carried away by the free software movement's recent momentum:

不管怎么说，这场自由软件与私有软件的战役在软件工业发展史上是一次重要的事件。哪怕像微软这么强大的企业，也需要以来第三方软件开发者。恰恰是他们开发的工具，程序，游戏才使得Windows系统更加吸引用户。回顾近二十年软件业的发展，哪怕不提自家公司的成长，蒙迪依旧提醒他的听众，不要被自由软件运动的风潮吹昏头脑：

> Two decades of experience have shown that an economic model that protects intellectual property and a business model that recoups research and development costs can create impressive economic benefits and distribute them very broadly.
>
> 二十年的经验表明，保护知识产权的经济社会，配合降低研发成本的商业模式，可以极大地促进经济发展和财富分配。

Such admonitions serve as the backdrop for Stallman's speech today. Less than a month after their utterance, Stallman stands with his back to one of the chalk boards at the front of the room, edgy to begin.

这段来自蒙迪的警告，成了斯托曼今日演讲的背景。现在，距离蒙迪演讲后不到一个月，斯托曼就站在演讲礼堂前方，背对着黑板，摩拳擦掌，迫不及待地准备发表他的演说了。

If the last two decades have brought dramatic changes to the software marketplace, they have brought even more dramatic changes to Stallman himself. Gone is the skinny, clean-shaven hacker who once spent his entire days communing with his beloved PDP-10. In his place stands a heavy-set middle-aged man with long hair and rabbinical beard, a man who now spends the bulk of his time writing and answering email, haranguing fellow programmers, and giving speeches like the one today. Dressed in an aqua-colored T-shirt and brown polyester pants, Stallman looks like a desert hermit who just stepped out of a Salvation Army dressing room.

这二十年来，相比软件业翻天覆地的变化，斯托曼的改变则更加明显。当年他曾是那位精瘦，不留胡须的黑客，曾在那台PDP-10前，没日没夜地和它畅谈。而如今，在讲坛上的那位，已是人近中年，发福，留发，蓄须。他会花大把时间回复邮件，在公众或程序员面前，发表演说，组织演讲——正如今天这次一样。他今天穿着海蓝色的T恤，一条棕色的涤纶裤子，看起来好似刚刚踏出沙漠的隐士。

The crowd is filled with visitors who share Stallman's fashion and grooming tastes. Many come bearing laptop computers and cellular modems, all the better to record and transmit Stallman's words to a waiting Internet audience. The gender ratio is roughly 15 males to 1 female, and 1 of the 7 or 8 females in the room comes in bearing a stuffed penguin, the official Linux mascot, while another carries a stuffed teddy bear.

人群之中，则充斥着和斯托曼有着类似打扮的听众。很多人带着笔记本电脑和蜂窝调制解调器，以及各种录音录像设备，准备把斯托曼的演讲传上互联网，那里还有更多的听众等着他的言论。听众中，男女比例大约十五比一。大约每七八个女听众里，就有一位拿着企鹅的毛绒玩具。这是Linux的吉祥物。还有些女听众则抱着泰迪熊。

Agitated, Stallman leaves his post at the front of the room and takes a seat in a front-row chair, tapping commands into an already-opened laptop. For the next 10 minutes Stallman is oblivious to the growing number of students, professors, and fans circulating in front of him at the foot of the auditorium stage.

人潮涌动，斯托曼则离开了演讲台，坐在第一排的听众席上，把一直开着机的笔记本电脑拿出来，开始敲键盘。接下来的十分钟里，进来的学生，教授和粉丝逐渐走到他旁边，围观他工作。而斯托曼对此则完全不在意。

Before the speech can begin, the baroque rituals of academic formality must be observed. Stallman's appearance merits not one but two introductions. Mike Uretsky, codirector of the Stern School's Center for Advanced Technology, provides the first.

既然在大学演讲，学院派的规矩是少不了的。演讲嘉宾介绍则是重要一环。对斯托曼的介绍可谓阵容强大。纽约大学的两位教授分别为他做两段开场白。第一位，是来自纽约大学斯特恩商学院高新技术研究中心的主任，麦克·乌列茨基。

“The role of a university is to foster debate and to have interesting discussions,” Uretsky says. “This particular presentation, this seminar falls right into that mold. I find the discussion of open source particularly interesting.”

“大学之地，争辩之所；争辩之处，兴趣所在。”乌列茨基道，“本次讲座，恰是秉承此道。个人愚见，所谓开源，甚是有趣。”

Before Uretsky can get another sentence out, Stallman is on his feet waving him down like a stranded motorist.

还没等乌列茨基把话说完，斯托曼就站起来挥着手喊道：

“I do free software,” Stallman says to rising laughter. “Open source is a different movement.”

“我是搞自由软件的，开源是另外一码事！”

The laughter gives way to applause. The room is stocked with Stallman partisans, people who know of his reputation for verbal exactitude, not to mention his much publicized 1998 falling out with the open source software proponents. Most have come to anticipate such outbursts the same way radio fans once waited for Jack Benny’s trademark, “Now cut that out!” phrase during each radio program.

这一嗓子引来了一阵哄堂大笑。笑声褪去，掌声渐起。当下，听众中绝大多数都对斯托曼的这份咬文嚼字有所耳闻，更知道在1998年，他和开源阵营支持者的那次争论。斯托曼的这一行为，就如同整点新闻一般，早在大家意料之中。

Uretsky hastily finishes his introduction and cedes the stage to Edmond Schonberg, a professor in the NYU computer-science department. As a computer programmer and GNU Project contributor, Schonberg knows which linguistic land mines to avoid. He deftly summarizes Stallman’s career from the perspective of a modern-day programmer.

乌列茨基草草结束介绍，走下讲堂。接下来对斯托曼做介绍的，是纽约大学计算机系的教授埃德蒙·舍恩伯格。作为一个计算机程序员，又是GNU工程的贡献者，他很清楚该如何用词。他站在当代程序员的角度，扼要回顾了斯托曼的事业。

“Richard is the perfect example of somebody who, by acting locally, started thinking globally [about] problems concerning the unavailability of source code,” says Schonberg. “He has developed a coherent philosophy that has forced all of us to reexamine our ideas of how software is produced, of what intellectual property means, and of what the software community actually represents.”

“放眼全球，脚踏实地，斯托曼是典范。他严肃地审视了软件源代码不对公众公开的事实，发展出了一套严密的逻辑体系。这套体系敦促着我们，让我们不得不重新思考如何开发软件，何谓知识产权，以及软件社区究竟是何物。”

Schonberg welcomes Stallman to more applause. Stallman takes a moment to shut off his laptop, rises out of his chair, and takes the stage.

舍恩伯格的致辞迎来了更多的掌声。斯托曼在掌声中，合上笔记本电脑，挪出身子，走上讲台。

At first, Stallman’s address seems more Catskills comedy routine than political speech. “I’d like to thank Microsoft for providing me the opportunity to be on this platform,” Stallman wisecracks. “For the past few weeks, I have felt like an author whose book was fortuitously banned somewhere.”

演讲开始，斯托曼表现得更像是个老练的喜剧演员，让人没法察觉这是一场严肃的政论讲座。“我首先要感谢微软公司给了我这样一次机会，来到贵校，畅抒己见。”斯托曼玩笑道，“过去几周里，我都觉得自己像个没落作家，自己的作品都无人问津。”

For the uninitiated, Stallman dives into a quick free software warm-up analogy. He likens a software program to a cooking recipe. Both provide useful step-by-step instructions on how to complete a desired task and can be easily modified if a user has special desires or circumstances. “You don’t have to follow a recipe exactly,” Stallman notes. “You can leave out some ingredients. Add some mushrooms, ’cause you like mushrooms. Put in less salt because your doctor said you should cut down on salt – whatever.”

为了照顾新人，斯托曼用一套类比，简要介绍了自由软件。他把软件代码比作烹饪菜谱。二者都提供清晰的步骤，说明如何一步一步完成某个特定任务。同时，人们都可以很容易地按照自己的需求，对它们进行修改。“你不用按照菜谱的每一步严格执行操作。”斯托曼说，“你可以少放点调料。喜欢蘑菇，放些蘑

菇；口味淡就少放盐；加点胡椒粉什么的。”

Most importantly, Stallman says, software programs and recipes are both easy to share. In giving a recipe to a dinner guest, a cook loses little more than time and the cost of the paper the recipe was written on. Software programs require even less, usually a few mouse-clicks and a modicum of electricity. In both instances, however, the person giving the information gains two things: increased friendship and the ability to borrow interesting recipes in return.

最重要的，斯托曼强调，软件和菜谱都很容易被分享。倘若有位客人来家里吃晚餐，那么把菜谱给他无非是花些时间，费点笔墨。拷贝软件则要求更少，只要轻点鼠标，费点电。而分享之后，你则起码有两份收获：增进了友谊；同时，下次你需要帮忙的时候，对方也会有所回报。

“Imagine what it would be like if recipes were packaged inside black boxes,” Stallman says, shifting gears. “You couldn’t see what ingredients they’re using, let alone change them, and imagine if you made a copy for a friend. They would call you a pirate and try to put you in prison for years. That world would create tremendous outrage from all the people who are used to sharing recipes. But that is exactly what the world of proprietary software is like. A world in which common decency towards other people is prohibited or prevented.”

“想象一下，要是菜谱全被封锁在一个黑匣子里，那会如何？”斯托曼话峰一转，“你不知道他们用了什么调料，只有他们才能更改配方。你如果把菜谱抄出一份，送给朋友，他们就把你叫贼，把你关进牢房，长达数年。如果你早就习惯了把菜谱传来传去，这样的世界一定会让你觉得不可理喻。可这一切恰恰发生在专有软件的世界之中。在这个世界里，很平常的社交行为被严格禁止，或者被想方设法避免。”

With this introductory analogy out of the way, Stallman launches into a retelling of the Xerox laser-printer episode. Like the recipe analogy, the laser-printer story is a useful rhetorical device. With its parable-like structure, it dramatizes just how quickly things can change in the software world. Drawing listeners back to an era before Amazon.com one-click shopping, Microsoft Windows, and Oracle databases, it asks the listener to examine the notion of software ownership free of its current corporate logos.

类比过后，斯托曼又提起了施乐打印机事件。和烹饪菜谱的类比一样，打印机

事件也是一个称手的工具。两者介绍完，听众就可以了解到如今的软件业究竟发生了多大改变。斯托曼的介绍，把听众拉回了曾经那个年代，那时还没有亚马逊和它的一键支付；没有微软和它的Windows系统；没有甲骨文数据库。这样的背景之下，听众有了很大的想象空间，可以不受当下这些所谓的大公司影响，重新审视所谓的软件所有权。

Stallman delivers the story with all the polish and practice of a local district attorney conducting a closing argument. When he gets to the part about the Carnegie Mellon professor refusing to lend him a copy of the printer source code, Stallman pauses.

讲起施乐打印机事件，斯托曼是轻车孰路。他好似律师做法庭最后陈词一般，字斟句酌。当说到卡耐基梅隆大学的那位计算机教授拒绝给他源代码的时候，斯托曼道：

“He had betrayed us,” Stallman says. “But he didn’t just do it to us. Chances are he did it to you.”

“他背叛了我们。”斯托曼稍有停顿，接着说，“但他不止背叛了我们，更有可能背叛你！”

On the word “you,” Stallman points his index finger accusingly at an unsuspecting member of the audience. The targeted audience member’s eyebrows flinch slightly, but Stallman’s own eyes have moved on. Slowly and deliberately, Stallman picks out a second listener to nervous titters from the crowd. “And I think, mostly likely, he did it to you, too,” he says, pointing at an audience member three rows behind the first.

“你”字一出，斯托曼就伸出食指，指向在座听众。听众之中，有人稍有皱眉。而斯托曼的目光，则移到了前排，一位听众正在低头偷笑。“而且我觉得，他更有可能背叛你。”斯托曼指着刚才偷笑的那位听众。

By the time Stallman has a third audience member picked out, the titters have given away to general laughter. The gesture seems a bit staged, because it is. Still, when it comes time to wrap up the Xerox laser-printer story, Stallman does so with a showman’s flourish. “He probably did it to most of the people here in this room – except a few, maybe, who weren’t born yet in 1980,” Stallman says, drawing more laughs. “[That’s] because he had promised to refuse to cooperate with just about the entire population of the planet Earth.”

这个临场的包袱，把一个人的窃笑变成了全场大笑。各种行为，好似舞台剧一般。笑声中，他总结道：“要想不被他背叛，你只能盼着晚点投胎。”笑声又起，“因为这位教授承诺，拒绝和地球上大多数人合作。”

Stallman lets the comment sink in for a half-beat. “He had signed a nondisclosure agreement,” Stallman adds.

斯托曼一字一顿：“他签署了保密协议。”

Richard Matthew Stallman’s rise from frustrated academic to political leader over the last 20 years speaks to many things. It speaks to Stallman’s stubborn nature and prodigious will. It speaks to the clearly articulated vision and values of the free software movement Stallman helped build. It speaks to the high-quality software programs Stallman has built, programs that have cemented Stallman’s reputation as a programming legend. It speaks to the growing momentum of the GPL, a legal innovation that many Stallman observers see as his most momentous accomplishment.

理查德·马修·斯托曼，从久经历练的学术精英，到一呼百应的运动领袖。这一蜕变本身就饱含寓意。它诉说着斯托曼的顽强与固执，诉说着他的决心和毅力。更清晰地诠释了自由软件运动的价值和远见。这之中，也当然包含了斯托曼编写的高质量代码，字里行间，凝结了斯托曼的心血，将他的经历，铸成了传奇。传奇之上，更能看到GPL不可遏制的蓬勃生命力。而GPL本身，作为法律界的一大创新，已经被公认为斯托曼的重要成就。

Most importantly, it speaks to the changing nature of political power in a world increasingly beholden to computer technology and the software programs that power that technology.

当下，计算机和相关软件技术已成为整个世界的一大支柱。理查德的经历，也更彰显了在如此背景之下，政权民意的风云变化。

Maybe that’s why, even at a time when most high-technology stars are on the wane, Stallman’s star has grown. Since launching the GNU Project in 1984, Stallman has been at turns ignored, satirized, vilified, and attacked–both from within and without the free software movement. Through it all, the GNU Project has managed to meet its milestones, albeit with a few notorious delays, and stay relevant in a software marketplace several orders of magnitude more complex than the one it entered 18 years ago. So too has the

free software ideology, an ideology meticulously groomed by Stallman himself.

这一切的一切，也许恰恰能解释，为什么斯托曼发起的运动能如此长久不衰，而很多当年的名牌大厂，却早已风光不再的原因。遥想当年，1984年，斯托曼刚刚发起了GNU工程。面对这一工程，自由软件运动内外，都充斥着对这个工程的无视，嘲讽，甚至攻击。一路走来，GNU工程虽然有过几次跳票延期，却还能在大多数时候按时交付，完成一个又一个的发布计划。踏过了十八个寒暑，GNU工程也日渐成熟，在软件市场中，得以赢得一席之地。而这近二十年来，自由软件的理想被理查德精心呵护，渐渐传遍大江南北。

To understand the reasons behind this currency, it helps to examine Richard Stallman both in his own words and in the words of the people who have collaborated and battled with him along the way. The Richard Stallman character sketch is not a complicated one. If any person exemplifies the old adage “what you see is what you get,” it’s Stallman.

要想了解这潮流背后的缘由，须得兼听八方言论。这不仅包括斯托曼自己的评价，更要倾听和他在同一战壕里作战的战友们的叙述。其实，斯托曼的个性并不复杂，倘若你坚信“所见即所得”，那么斯托曼这个人就不难被理解。

“I think if you want to understand Richard Stallman the human being, you really need to see all of the parts as a consistent whole,” advises Eben Moglen, legal counsel to the Free Software Foundation and professor of law at Columbia University Law School. “All those personal eccentricities that lots of people see as obstacles to getting to know Stallman really ‘are’ Stallman: Richard’s strong sense of personal frustration, his enormous sense of principled ethical commitment, his inability to compromise, especially on issues he considers fundamental. These are all the very reasons Richard did what he did when he did.”

“想要了解斯托曼这个人，你必须要把各处细节联系起来，看成一个有机的整体。”伊本·莫格林（Eben Moglen）说道。他是自由软件基金会法律顾问，同时也是哥伦比亚大学法学院教授。“在斯托曼身上有着各种古怪脾气，这也许会把人拒之千里。而这份不同寻常，恰恰就构成了斯托曼这个活生生的人。他对挫败异常敏感，他对道德准则恪守不渝。他不肯妥协的个性，在关键问题上不肯让步的固执，这一切的总和，最终让我们看到了当今的斯托曼。”

Explaining how a journey that started with a laser printer would eventually lead to a sparring match with the world’s richest corporation is no easy task. It

requires a thoughtful examination of the forces that have made software ownership so important in today's society. It also requires a thoughtful examination of a man who, like many political leaders before him, understands the malleability of human memory. It requires an ability to interpret the myths and politically laden code words that have built up around Stallman over time. Finally, it requires an understanding of Stallman's genius as a programmer and his failures and successes in translating that genius to other pursuits.

一个简单的打印机事件，变成了燎原之火，燃遍全球，足以和全球最富有的软件公司抗衡。要想回顾和解释这一切，并非易事。首先要了解软件所有权，以及它是如何走到当今的重要位置的；更要和人类健忘的本性做斗争；还要能从关于斯托曼的各种神话和攻击之中，看出本质。最后，还要能理解斯托曼在软件领域的过人天赋；以及他如何把这份天赋用到其他领域；还有在这一过程中的各种成败得失。

When it comes to offering his own summary of the journey, Stallman acknowledges the fusion of personality and principle observed by Moglen. "Stubbornness is my strong suit," he says. "Most people who attempt to do anything of any great difficulty eventually get discouraged and give up. I never gave up."

当我请斯托曼做一番自我总结的时候，他也强调了莫格林提到的个性和原则："坚强固执是我的本性。很多人在尝试做一些事，遇到了困难，就退缩放弃了。可我从不言弃。"

He also credits blind chance. Had it not been for that run-in over the Xerox laser printer, had it not been for the personal and political conflicts that closed out his career as an MIT employee, had it not been for a half dozen other timely factors, Stallman finds it very easy to picture his life following a different career path. That being said, Stallman gives thanks to the forces and circumstances that put him in the position to make a difference.

他也同样感激自己的运气。倘若当初没有那次打印机事件，没有其中的各种人事冲突，没有当年的各种机缘巧合，他也许不会放弃麻省理工学院的研究职位，不会重新抉择，选择一条与众不同的道路，并为之奋斗终身。是身边的各种因素，最终让斯托曼得以做出不同凡响的成绩。

"I had just the right skills," says Stallman, summing up his decision for launching the GNU Project to the audience. ``Nobody was there but me, so I

felt like, ‘I’m elected. I have to work on this. If not me, who?’

”

“我正好有合适的技能。”斯托曼回顾着当初发起GNU工程的决定，总结道，“除了我以外，没人在做这事。我就觉得：‘责任在身，我若不做，舍我其谁。’”

A Portrait of the Hacker as a Young Man 黑客正年少

Richard Stallman’s mother, Alice Lippman, still remembers the moment she realized her son had a special gift.

理查德·斯托曼的母亲，爱丽丝·李普曼（Alice Lippman）始终还记得她发现理查德过人天赋的那一刻。

“I think it was when he was eight,” Lippman recalls.

“我记得他那会才八岁。”李普曼回忆着。

The year was 1961, and Lippman, a recently divorced single mother, was whiling away a weekend afternoon within the family’s tiny one-bedroom apartment on Manhattan’s Upper West Side. Leafing through a copy of Scientific American, Lippman came upon her favorite section, the Martin Gardner-authored column titled “Mathematical Games.” A substitute art teacher at the time, Lippman enjoyed Gardner’s column for the brain-teasers it provided. With her son already ensconced in a book on the nearby sofa, Lippman decided to take a crack at solving the week’s feature puzzle.

那是1961年，李普曼刚刚离异，做了单亲妈妈。她家位于曼哈顿上西城，是一个一居室的小公寓。一个周末下午，她正在自己家里翻着一期《科学美国人》杂志，打发时光。她翻开自己最喜欢的一版，马丁·加德纳（Martin Gardner）主编的专栏：《数学游戏》。作为一个教艺术的代课老师，李普曼常常被这一专栏的各种头脑游戏吸引。她看了一眼在旁边沙发上老老实实呆着的理查德，决定开始挑战本周的思考题。

“I wasn’t the best person when it came to solving the puzzles,” she admits. “But as an artist, I found they really helped me work through conceptual

barriers.”

“其实我做这些题目做得不是很好。”李普曼坦言，“可作为艺术家，我发现它们可以帮我缓解压力，激发灵感。”

Lippman says her attempt to solve the puzzle met an immediate brick wall. About to throw the magazine down in disgust, Lippman was surprised by a gentle tug on her shirt sleeve.

可她刚一开始做，就遇到了障碍。眼看她就打算扔开杂志，放弃解答了，可这时候，她发现有人正在轻拽她的衣袖。

“It was Richard,” she recalls, “He wanted to know if I needed any help.”

“那是理查德，”她回忆道，“他想问我要不要帮忙。”

Looking back and forth, between the puzzle and her son, Lippman says she initially regarded the offer with skepticism. “I asked Richard if he’d read the magazine,” she says. “He told me that, yes, he had and what’s more he’d already solved the puzzle. The next thing I know, he starts explaining to me how to solve it.”

她看看自己儿子，又回过头看看杂志上的题目，她还是有点半信半疑。“我问理查德，你读过这杂志了？”李普曼说，“理查德说读过了。而且他竟然已经解决这个题目了。之后，我就记得他开始给我讲解答题思路。”

Hearing the logic of her son’s approach, Lippman’s skepticism quickly gave way to incredulity. “I mean, I always knew he was a bright boy,” she says, “but this was the first time I’d seen anything that suggested how advanced he really was.”

听着自己儿子清晰的逻辑，李普曼开始的半信半疑瞬间转变，一切让她觉得难以置信。“我知道理查德很聪明，”她说，“可这次却是实实在在让我见识了他究竟有多聪明。”

Thirty years after the fact, Lippman punctuates the memory with a laugh. “To tell you the truth, I don’t think I ever figured out how to solve that puzzle,” she says. “All I remember is being amazed he knew the answer.”

三十几年过去了，李普曼回忆起这事，最后笑道，“老实说，我到现在都没明白那道题是怎么做的。我能记得的，只是当时非常震惊，感叹理查德居然能做出

那个题目。”

Seated at the dining-room table of her second Manhattan apartment – the same spacious three-bedroom complex she and her son moved to following her 1967 marriage to Maurice Lippman, now deceased – Alice Lippman exudes a Jewish mother’s mixture of pride and bemusement when recalling her son’s early years. The nearby dining-room credenza offers an eight-by-ten photo of Stallman glowering in full beard and doctoral robes. The image dwarfs accompanying photos of Lippman’s nieces and nephews, but before a visitor can make too much of it, Lippman makes sure to balance its prominent placement with an ironic wisecrack.

李普曼坐在餐桌前，这里是她在曼哈顿住过的第二个公寓，在一座公寓楼里，是一个三居室的宽敞房子。1967年，她再婚嫁给了莫里斯·李普曼（Maurice Lippman），就带着斯托曼搬到了这所公寓里。时过境迁，莫里斯已经去世，斯托曼也长大成人。谈起当年抚养斯托曼，这位犹太妈妈流露出的尽是自豪和欣慰。在餐厅里的橱柜里，摆放着一张八乘十英寸的照片。照片里是满脸络腮胡须的斯托曼，身穿博士服。这张照片摆在那里，让其他几张李普曼侄子侄女的照片显得不再引人注意。不过在客人问起这张照片之前，李普曼总会先调侃几句，解释一下为什么把它放到这么显眼的地方。

“Richard insisted I have it after he received his honorary doctorate at the University of Glasgow,” says Lippman. ``He said to me, ‘Guess what, mom? It’s the first graduation I ever attended.’

”

“理查德坚持要把这张相片给我。这是他当年获得英国格拉斯哥大学（University of Glasgow）荣誉博士学位的时候照的。他跟我说：‘你看，这是我有生以来参加的第一个毕业典礼。’”

Such comments reflect the sense of humor that comes with raising a child prodigy. Make no mistake, for every story Lippman hears and reads about her son’s stubbornness and unusual behavior, she can deliver at least a dozen in return.

抚养理查德这么一个孩子，李普曼总是有很多话可说。每当别人说起理查德最新的各种或古怪或固执的行为时，她总会翻出更多类似陈年旧事作为回应。

“He used to be so conservative,” she says, throwing up her hands in mock

exasperation. ``We used to have the worst arguments right here at this table. I was part of the first group of public city school teachers that struck to form a union, and Richard was very angry with me. He saw unions as corrupt. He was also very opposed to social security. He thought people could make much more money investing it on their own. Who knew that within 10 years he would become so idealistic? All I remember is his stepsister coming to me and saying, 'What is he going to be when he grows up? A fascist?'

"

"他以前是十足的保守派。"她说着，把双臂铺开，以示强调，"曾经就在这张桌子前，我们两人展开过一次很激烈的争论。我曾经是城市公立学校教师，其中，我是第一批坚持要成立工会的人。可理查德却觉得工会是腐败之源。他甚至非常反对社会保险。他认为，人们用买保险的钱，完全可以投资其他东西，赚更多的钱。当初谁会想到他十年之后会成为一个如此的理想主义者。我就记得当初我女儿曾问过我：'你说以后理查德长大了会是什么样？法西斯极右翼分子？'"

As a single parent for nearly a decade – she and Richard's father, Daniel Stallman, were married in 1948, divorced in 1958, and split custody of their son afterwards – Lippman can attest to her son's aversion to authority. She can also attest to her son's lust for knowledge. It was during the times when the two forces intertwined, Lippman says, that she and her son experienced their biggest battles.

李普曼1948年和理查德的父亲丹尼尔·斯托曼结婚，于1958年离异。李普曼成了单亲妈妈，抚养和监护理查德的重担几乎都压在了她的身上。一路走来，她见证了理查德蔑视权威的个性，更见证了他对知识的如饥似渴。这两股力量在理查德身上几经纠缠，也让李普曼和理查德有过不少摩擦冲突。

"It was like he never wanted to eat," says Lippman, recalling the behavior pattern that set in around age eight and didn't let up until her son's high-school graduation in 1970. "I'd call him for dinner, and he'd never hear me. I'd have to call him 9 or 10 times just to get his attention. He was totally immersed."

"他好像一直都不想吃饭一样。"李普曼回忆着理查德八岁到1970年中学毕业的那段时间，"他就跟个聋子似的，我每次都得叫他九到十次，他才肯过来吃饭。"

Stallman, for his part, remembers things in a similar fashion, albeit with a

political twist.

同样是这件事，到了斯托曼嘴里，就掺进了政治的意味。

“I enjoyed reading,” he says. “If I wanted to read, and my mother told me to go to the kitchen and eat or go to sleep, I wasn’t going to listen. I saw no reason why I couldn’t read. No reason why she should be able to tell me what to do, period. Essentially, what I had read about, ideas such as democracy and individual freedom, I applied to myself. I didn’t see any reason to exclude children from these principles.”

“我特别喜欢读书。”斯托曼说，“赶在我想读书的时候，我妈妈恰巧叫我去吃饭，我肯定不会去。我就觉得，凭什么我不能读书？凭什么我妈妈可以指点我该做什么，该读什么书。我坚持民主和个人自由。这些思想不应该仅仅局限在成人身上，未成年人也同样适用。”

The belief in individual freedom over arbitrary authority extended to school as well. Two years ahead of his classmates by age 11, Stallman endured all the usual frustrations of a gifted public-school student. It wasn’t long after the puzzle incident that his mother attended the first in what would become a long string of parent-teacher conferences.

斯托曼坚信个人自由高于任何权威。这样的思想也被他带到了校园之中。斯托曼比其他的孩子早上了两年学。和很多天才儿童一样，斯托曼也有着自己的烦恼。那次他帮妈妈解决完杂志上的数学题之后没多久，老师就妈妈约去学校，进行了一次长谈。

“He absolutely refused to write papers,” says Lippman, recalling an early controversy. “I think the last paper he wrote before his senior year in high school was an essay on the history of the number system in the west for a fourth-grade teacher.” To be required to choose a specific topic when there was nothing he actually wanted to write about was almost impossible for Stallman, and painful enough to make him go to great lengths to avoid such situations.

“他拒绝写作文写文章。”李普曼回忆着当时的情景，“除了高中最后一年，他最后一次写作文是四年级。那是一篇关于西方计数系统的文章。”

Gifted in anything that required analytical thinking, Stallman gravitated toward math and science at the expense of his other studies. What some teachers

saw as single-mindedness, however, Lippman saw as impatience. Math and science offered simply too much opportunity to learn, especially in comparison to subjects and pursuits for which her son seemed less naturally inclined. Around age 10 or 11, when the boys in Stallman's class began playing a regular game of touch football, she remembers her son coming home in a rage. "He wanted to play so badly, but he just didn't have the coordination skills," Lippman recalls. "It made him so angry."

凭借着分析性思维的天赋，斯托曼沉浸在数理化的世界，而代价则是其他方面严重欠缺。老师们看来是头脑简单的行为，在李普曼看来，倒觉得是斯托曼好强所致。比起其他斯托曼天生就不太在行的领域来说，数学和自然科学给斯托曼提供了太多的学习机会。大概在10到11岁那会，斯托曼的同学们都在玩触式橄榄球。她记得有一次斯托曼回家特别生气。"他当时非常想和他们一起玩，可他身体的协调性实在太差了。"李普曼回忆道，"这实在让他非常恼火。"

The anger eventually drove her son to focus on math and science all the more. Even in the realm of science, however, her son's impatience could be problematic. Poring through calculus textbooks by age seven, Stallman saw little need to dumb down his discourse for adults. Sometime, during his middle-school years, Lippman hired a student from nearby Columbia University to play big brother to her son. The student left the family's apartment after the first session and never came back. "I think what Richard was talking about went over his head," Lippman speculates.

这份感觉让斯托曼更埋头在理科世界里。可哪怕在数理化的世界，他的好强有时也会成问题。在八岁的时候，斯托曼就已经把微积分教材翻了个透。他觉得自己完全没必要在大人面前装成傻傻的天真宝宝。他中学的时候，妈妈曾从附近的哥伦比亚大学里雇来了一位大学生，以期在斯托曼面前扮演大哥哥的角色。结果这位学生来了一次就没再来了。"我觉得斯托曼当时讨论的问题可能超出他的极限了。"李普曼推测。

Another favorite maternal memory dates back to the early 1960s, shortly after the puzzle incident. Around age seven, two years after the divorce and relocation from Queens, Richard took up the hobby of launching model rockets in nearby Riverside Drive Park. What started as aimless fun soon took on an earnest edge as her son began recording the data from each launch. Like the interest in mathematical games, the pursuit drew little attention until one day, just before a major NASA launch, Lippman checked in on her son to

see if he wanted to watch.

另一个李普曼喜欢提起的小事，发生在六十年代早期。那时李普曼已经离异两年，也从皇后区搬了出来。斯托曼那会七岁，迷上了火箭模型。他常常去河滨公园发射火箭模型。一开始他只是随便瞎玩，可后来他开始记录每次的发射数据。起初，这些行为并没引起李普曼注意，她觉得这兴趣就和玩数学游戏一样。可有次NASA（美国国家航空航天局）发射火箭前，李普曼问斯托曼要不要看电视上的现场直播。

“He was fuming,” Lippman says. “All he could say to me was, ‘But I’m not published yet.’ Apparently he had something that he really wanted to show NASA.” Stallman doesn’t remember the incident, but thinks it more likely that he was anguished because he didn’t have anything to show.

“他可逗了，上来就跟我说：‘可我的研究成果还没发表呢！’这话说的就好像真有点资料要给NASA看似的。”斯托曼自己不记得这件事了，但他觉得他当时应该是有点苦闷，因为他根本没有任何东西可以给NASA看。

Such anecdotes offer early evidence of the intensity that would become Stallman’s chief trademark throughout life. When other kids came to the table, Stallman stayed in his room and read. When other kids played Johnny Unitas, Stallman played spaceman. “I was weird,” Stallman says, summing up his early years succinctly in a 1999 interview. “After a certain age, the only friends I had were teachers.” Stallman was not ashamed of his weird characteristics, distinguishing them from the social ineptness that he did regard as a failing. However, both contributed together to his social exclusion.

这个小事倒是很好地表现了斯托曼性格中独具一格的特点——紧迫感。这种紧迫感一直伴随在他左右。其他孩子还在桌前玩耍的时候，他在房间里读书；别的孩子在痴迷橄榄球运动员约翰尼·尤尼塔斯的时候，斯托曼则崇拜着火箭专家。“我当时就是这么古怪。”斯托曼在1999年接受一次采访的时候，如此总结着自己的童年，“没过几年，我的朋友就只剩下各位老师了。” 对于他自己古怪的性格，斯托曼并不觉得丢人，但是他对于自己不擅长社交这一点总是深感遗憾。不过，正是他特有性格和社交能力的缺乏造成了他与现实社会之间的隔阂。

Although it meant courting more run-ins at school, Lippman decided to indulge her son’s passion. By age 12, Richard was attending science camps during the summer and private school during the school year. When a teacher

recommended her son enroll in the Columbia Science Honors Program, a post-Sputnik program designed for gifted middle- and high-school students in New York City, Stallman added to his extracurriculars and was soon commuting uptown to the Columbia University campus on Saturdays.

虽然斯托曼总比同龄人显得古怪很多，李普曼还是决定让他追求自己的兴趣和热情。12岁那年，斯托曼参加了科学夏令营，之后上了私立中学。学校的老师建议让斯托曼参加“哥伦比亚科学之星计划”。该计划旨在为具有天赋的纽约中学生提供更多的学习机会。斯托曼参加了这个计划，于是每周六，他都会赶去哥伦比亚大学。

Dan Chess, a fellow classmate in the Columbia Science Honors Program, recalls Richard Stallman seeming a bit weird even among the students who shared a similar lust for math and science. “We were all geeks and nerds, but he was unusually poorly adjusted,” recalls Chess, now a mathematics professor at Hunter College. “He was also smart as shit. I’ve known a lot of smart people, but I think he was the smartest person I’ve ever known.”

丹·柴斯（Dan Chess）也参加了科学之星计划，他回忆中的斯托曼，哪怕在众多同龄的科学爱好者之中，也算是古怪的。“我们都是书呆子，技术宅，可斯托曼在这圈子里仍是特立独行的。”如今的柴斯已经是亨特学院数学系的教授，他回忆道：“他更是聪明绝顶。我见过很多聪明人，可斯托曼是我见过最棒的。”

Seth Breidbart, a fellow Columbia Science Honors Program alumnus, offers bolstering testimony. A computer programmer who has kept in touch with Stallman thanks to a shared passion for science fiction and science-fiction conventions, he recalls the 15-year-old, buzz-cut-wearing Stallman as “scary,” especially to a fellow 15-year-old.

另外一个科学之星计划的成员，赛思·布莱德巴特（Seth Breidbart）也提供了类似的证据。布莱德巴特如今也是个程序员，依然和斯托曼保持联系。他回忆当年只有15岁、剃了短发的斯托曼甚至有些“令人生畏”。

“It’s hard to describe,” Breidbart says. “It wasn’t like he was unapproachable. He was just very intense. [He was] very knowledgeable but also very hardheaded in some ways.”

“这可难解释了。不是说斯托曼会拒人千里，只是说在他身上，总能看到一种紧迫感。他知识丰富，可有些地方又非常爱钻牛角尖，非常固执。”布莱德巴特回

忆说。

Such descriptions give rise to speculation: are judgment-laden adjectives like “intense” and “hardheaded” simply a way to describe traits that today might be categorized under juvenile behavioral disorder? A December, 2001, *Wired* magazine article titled “The Geek Syndrome” paints the portrait of several scientifically gifted children diagnosed with high-functioning autism or Asperger Syndrome. In many ways, the parental recollections recorded in the *Wired* article are eerily similar to the ones offered by Lippman. Stallman also speculates about this. In the interview for a 2000 profile for the *Toronto Star*, Stallman said he wondered if he were “borderline autistic.” The article inaccurately cited the speculation as a certainty.

“紧迫感”，“爱钻牛角尖”，“固执”，这样几个形容词放在一起，放在今天哪个中学生身上，多半会让人觉得是青春期综合症的症状。其实，在2001年12月的《连线》杂志中，就有一篇名为《天才综合症》的报道，采访了几个自闭症儿童患者，他们都在数学和科学方面具有天赋。报道中，患者家长对患者的描述，在很多方面都类似李普曼对斯托曼的形容。哪怕是斯托曼，提到自己精神方面，也曾犹豫再三。在一份2000年《多伦多之星》报纸上的资料中，斯托曼曾描述自己为“濒临自闭”，以此解释自己一直以来，在社交方面的孤僻，并介绍了自己如何不断克服这种心理。

Such speculation benefits from the fast and loose nature of most so-called “behavioral disorders” nowadays, of course. As Steve Silberman, author of “The Geek Syndrome,” notes, American psychiatrists have only recently come to accept Asperger Syndrome as a valid umbrella term covering a wide set of behavioral traits. The traits range from poor motor skills and poor socialization to high intelligence and an almost obsessive affinity for numbers, computers, and ordered systems.

这种判断和当下流行的“异常行为”研究是分不开的。《天才综合症》的作者史蒂夫·西尔贝曼（Steve Silberman）介绍，美国的心理学家在近年来才把自闭症作为一系列症状行为特征的统称。这些行为包括：不擅运动，不擅社交，但又对数学，计算机和类似的具有规律的系统有着极度的依恋，并擅长于此。

“It’s possible I could have had something like that,” Stallman says. “On the other hand, one of the aspects of that syndrome is difficulty following rhythms. I can dance. In fact, I love following the most complicated rhythms. It’s not clear cut enough to know.” Another possibility is that Stallman had a “shadow

syndrome" which goes some way in the direction of Asperger's syndrome but without going beyond the limits of normality.

"我当初也许还真有这种毛病吧。"斯托曼说，"可自闭症患者的另一个症状是对音乐和节奏迟钝。我还能跳舞呢，而且我对节奏特别敏感，喜欢复杂的节奏。这么看倒也未必是自闭症了。"

Chess, for one, rejects such attempts at back-diagnosis. "I never thought of him [as] having that sort of thing," he says. "He was just very unsocialized, but then, we all were."

柴斯则非常反对这种推断："我从不觉得斯托曼有自闭症。他当初就是社交方面有点障碍，可我们这群人都这样。"

Lippman, on the other hand, entertains the possibility. She recalls a few stories from her son's infancy, however, that provide fodder for speculation. A prominent symptom of autism is an oversensitivity to noises and colors, and Lippman recalls two anecdotes that stand out in this regard. "When Richard was an infant, we'd take him to the beach," she says. "He would start screaming two or three blocks before we reached the surf. It wasn't until the third time that we figured out what was going on: the sound of the surf was hurting his ears." She also recalls a similar screaming reaction in relation to color: "My mother had bright red hair, and every time she'd stoop down to pick him up, he'd let out a wail."

而李普曼则认为确有此事。她说了斯托曼婴儿时期的一些事情，佐证这个结论。自闭症患者的一个显著症状，就是对噪声和颜色异常敏感。李普曼记起了两个事情："当时斯托曼还是个婴儿，我们带他去海边，每次去，在离海有两三个街区的时候，他就开始大哭。去了三次，我们才发现，原来是他觉得海浪的声音太吵了。"接着，她又说了斯托曼当时对颜色的反映："斯托曼姥姥的头发是亮红色的，每次他姥姥抱起他的时候，他也会大哭大叫。"

In recent years, Lippman says she has taken to reading books about autism and believes that such episodes were more than coincidental. "I do feel that Richard had some of the qualities of an autistic child," she says. "I regret that so little was known about autism back then."

这几天，李普曼读了一些关于自闭症的书。她觉得斯托曼的这些行为绝非巧合："我确实觉得理查德当时有很多自闭症儿童的症状。我真后悔当初对这种病

没什么了解。”

Over time, however, Lippman says her son learned to adjust. By age seven, she says, her son had become fond of standing at the front window of subway trains, mapping out and memorizing the labyrinthian system of railroad tracks underneath the city. It was a hobby that relied on an ability to accommodate the loud noises that accompanied each train ride. “Only the initial noise seemed to bother him,” says Lippman. “It was as if he got shocked by the sound but his nerves learned how to make the adjustment.”

李普曼说，时间一久，斯托曼也逐渐开始适应了环境。七岁的时候，他开始喜欢坐在地铁的第一节车厢里，透过司机窗口看着地铁隧道，记下地铁的路线。这必须得习惯了地铁的噪音才行。“只有启动时候的噪音会让他不安。”李普曼说，“一启动，他就跟被谁打了一下似的。不过他后来也逐渐习惯了。”

For the most part, Lippman recalls her son exhibiting the excitement, energy, and social skills of any normal boy. It wasn’t until after a series of traumatic events battered the Stallman household, she says, that her son became introverted and emotionally distant.

李普曼回忆，斯托曼以前也和其他孩子一样，活泼快乐有生气。可家中一系列的变故，让他越来越内向，逐渐与人疏远。

The first traumatic event was the divorce of Alice and Daniel Stallman, Richard’s father. Although Lippman says both she and her ex-husband tried to prepare their son for the blow, she says the blow was devastating nonetheless. “He sort of didn’t pay attention when we first told him what was happening,” Lippman recalls. ``But the reality smacked him in the face when he and I moved into a new apartment. The first thing he said was, ‘Where’s Dad’s furniture?’

”

第一件事就是斯托曼的父母离异。李普曼和斯托曼的父亲，丹尼尔·斯托曼都有所准备，打算慢慢把离异的事实告诉斯托曼，希望不对他产生太大影响。可结果依旧让斯托曼改变很多。李普曼回忆：“他一开始好像还没把这件事放在心上，可当我和他一起搬到新公寓的时候，他彻底意识到发生了什么。他的第一反应是：‘爸爸的那些家具哪去了？’”

For the next decade, Stallman would spend his weekdays at his mother’s

apartment in Manhattan and his weekends at his father's home in Queens. The shuttling back and forth gave him a chance to study a pair of contrasting parenting styles that, to this day, leaves Stallman firmly opposed to the idea of raising children himself. Speaking about his father, a World War II vet who died in early 2001, Stallman balances respect with anger. On one hand, there is the man whose moral commitment led him to learn French just so he could be more helpful to Allies when they'd finally fight the Nazis in France. On the other hand, there was the parent who always knew how to craft a put-down for cruel effect.

在接下来的十年，每周工作日，斯托曼都住在母亲的公寓；休息日则要跑去皇后区，在爸爸家里住。辗转之间，他也见识了两种完全不同的抚养方式。这一切，让他从此决定不会自己抚养小孩。理查德的父亲曾是一名退伍的二战老兵，于2001年去世。每当说起他，理查德总是对他又尊敬又恼火。一方面，这是一位有担当的士兵，他曾学习法语，只为了更好地帮助盟友；另一方面，他又是个总会把事情变得残酷不堪的父亲。

"My father had a horrible temper," Stallman says. "He never screamed, but he always found a way to criticize you in a cold, designed-to-crush way."

"我爸爸脾气很坏。"斯托曼说，"他虽然从不大吼大叫，但总能用一种冷嘲热讽的方式，去批评指责你。"

As for life in his mother's apartment, Stallman is less equivocal. "That was war," he says. "I used to say in my misery, 'I want to go home,' meaning to the nonexistent place that I'll never have."

至于说起和母亲住在一起的日子，斯托曼的评价就不再那么模棱两可了："那就跟天天在打仗似的。我最大的痛苦就是，别人好歹可以说句'我想家了'，可对我来说，所谓的'家'，根本就是个并不存在的空中楼阁。"

For the first few years after the divorce, Stallman found the tranquility that eluded him in the home of his paternal grandparents. One died when he was 8, and the other when he was 10. For Stallman, the loss was devastating. "I used to go and visit and feel I was in a loving, gentle environment," Stallman recalls. "It was the only place I ever found one, until I went away to college."

父母离异后的最初几年，斯托曼还可以把祖父母和外祖父母家当作一个避风港。可他10岁左右的时候，几位老人相继去世，斯托曼因此受到了巨大的打

击。“以前还能去爷爷奶奶，或者姥姥老爷家，能觉得有人疼我，有人爱我。上大学之前，那是唯一一个让我觉得如此温暖的地方。”

Lippman lists the death of Richard’s paternal grandparents as the second traumatic event. “It really upset him,” she says. He was very close to both his grandparents. Before they died, he was very outgoing, almost a leader-of-the-pack type with the other kids. After they died, he became much more emotionally withdrawn.

李普曼把祖父母和外祖父母去世列为第二个沉重打击理查德的家庭变故。“他当时悲痛欲绝“。他和几位老人关系特别好。他们去世之前，斯托曼特别开朗外向，都属于远近一带的‘孩子王’的角色。他们一走，斯托曼一下子变得极度消沉。”

From Stallman’s perspective, the emotional withdrawal was merely an attempt to deal with the agony of adolescence. Labeling his teenage years a “pure horror,” Stallman says he often felt like a deaf person amid a crowd of chattering music listeners.

对于斯托曼来讲，这种消沉则是一种躲避悲痛的行为。他把自己的青少年时期形容为“不寒而栗“。他说，就好比一个聋子，被扔到了一群音乐爱好者中间，眼睁睁看着他们聚在一起，谈音论乐。

“I often had the feeling that I couldn’t understand what other people were saying,” says Stallman, recalling his sense of exclusion. “I could understand the words, but something was going on underneath the conversations that I didn’t understand. I couldn’t understand why people were interested in the things other people said.”

“我总有一种感觉，觉得无法理解周围人在说什么。”斯托曼回忆当初，说着自己如何被悲痛感包围以至无法与外界沟通，“我明白他们说的每个词，可总觉得在谈话背后，隐藏着我无法理解的东西。我不能理解别人为什么对旁人的话题如此感兴趣。“

For all the agony it produced, adolescence would have an encouraging effect on Stallman’s sense of individuality. At a time when most of his classmates were growing their hair out, Stallman preferred to keep his short. At a time when the whole teenage world was listening to rock and roll, Stallman preferred classical music. A devoted fan of science fiction, *Mad* magazine, and

late-night TV, Stallman came to have a distinctly off-the-wall personality that met with the incomprehension of parents and peers alike.

悲伤往往会唤起青少年极度的自我意识，会强烈展示自己的个性和与众不同。那会儿，别的孩子都喜欢把头发留长，斯托曼则一头短发；全部青少年都开始迷恋摇滚乐，斯托曼则钟情与古典乐。他热衷科幻小说，喜欢读《*Mad*》杂志，爱看深夜电视节目。他发展出了这种特立独行的个性，以此回避与这个世界的格格不入。

“Oh, the puns,” says Lippman, still exasperated by the memory of her son’s teenage personality. “There wasn’t a thing you could say at the dinner table that he couldn’t throw back at you as a pun.”

李普曼回忆起斯托曼少年时的个性：“开玩笑，俏皮话，双关语。只要你在桌上抛出一句话，他肯定能借此抖个包袱耍个宝。”

Outside the home, Stallman saved the jokes for the adults who tended to indulge his gifted nature. One of the first was a summer-camp counselor who lent Stallman a manual for the IBM 7094 computer during his 8th or 9th year. To a preteenager fascinated with numbers and science, the gift was a godsend. Soon, Stallman was writing out programs on paper in the instructions of the 7094. There was no computer around to run them on, and he had no real applications to use one for, but he yearned to write a program – any program whatsoever. He asked the counselor for arbitrary suggestions of something to code.

在外面，斯托曼则省下了这些笑话，只留着和那些鼓励他追求兴趣的成年人沟通。其中一位，是12岁那年科学夏令营的顾问。他曾送给了斯托曼一份打印版的IBM 7094计算机的手册。作为一个迷恋数字和科学的少年，这份手册仿佛是来自上帝的礼物。那年暑假快结束的时候，斯托曼已经根据这份7094的手册，自己在纸上写了一份程序。他迫不及待地期待着有一天能在真正的机器上跑跑自己的程序。

With the first personal computer still a decade away, Stallman would be forced to wait a few years before getting access to his first computer. His chance finally came during his senior year of high school. The IBM New York Scientific Center, a now-defunct research facility in downtown Manhattan, offered Stallman the chance to try to write his first real program. His fancy was to write a pre-processor for the programming language PL/I, designed to add the

tensor algebra summation convention as a feature to the language. “I first wrote it in PL/I, then started over in assembler language when the compiled PL/I program was too big to fit in the computer,” he recalls.

可惜那会还是二十世纪六十年代，十几年之后才会出现第一台个人计算机。至于斯托曼第一次用上计算机，则是在这次夏令营之后几年。11年级结束的那个暑假，在那个暑假，斯托曼加入了IBM纽约科学研究中心，写下了他有生以来第一个跑在计算机上的程序——用PL/I语言编写的预处理器。“我开始是用PL/I写的，后来程序越写越大，大到计算机里放不下了，于是就用汇编语言重写了一遍。”斯托曼回忆说。

For the summer after high-school graduation, the New York Scientific Center hired him. Tasked with writing a numerical analysis program in Fortran, he finished that in a few weeks, acquiring such a distaste for the Fortran language that he vowed never to write anything in it again. Then he spent the rest of the summer writing a text-editor in APL.

高中毕业后的暑假，IBM纽约科学研究中心给他提供了一个工作机会，任务是用Fortran语言编写一个数值分析程序。他只用了短短几周时间就完成了这个程序，但是他发现Fortran语言很不合他的胃口，他发誓再也不用这种语言开发任何东西了。于是，他在暑假剩余的时间中用APL开发一个文本编辑器。

Simultaneously, Stallman had held a laboratory-assistant position in the biology department at Rockefeller University. Although he was already moving toward a career in math or physics, Stallman’s analytical mind impressed the lab director enough that a few years after Stallman departed for college, Lippman received an unexpected phone call. “It was the professor at Rockefeller,” Lippman says. “He wanted to know how Richard was doing. He was surprised to learn that he was working in computers. He’d always thought Richard had a great future ahead of him as a biologist.”

与此同时，斯托曼又从洛克菲勒大学生物系拿到了一份实验员的工作。尽管斯托曼已经确定未来将向数学或物理方向发展，可他严密的思维还是打动了这个生物实验室的主任。斯托曼进入大学没几年，李普曼还收到了一个电话。“那是洛克菲勒大学的教授，他想知道斯托曼近况如何。当知道如今斯托曼在搞计算机的时候，他很吃惊。这位教授始终觉得斯托曼非常适合成为一个优秀的生物学家。”

Stallman’s analytical skills impressed faculty members at Columbia as well,

even when Stallman himself became a target of their ire. “Typically once or twice an hour [Stallman] would catch some mistake in the lecture,” says Breidbart. “And he was not shy about letting the professors know it immediately. It got him a lot of respect but not much popularity.”

斯托曼的理性分析能力，也惹上了哥伦比亚大学的教授。参加哥伦比亚大学科学之星计划的时候，布莱德巴特回忆说：“一般情况下，每隔一两个小时，斯托曼都能挑出讲课老师的一个小疏忽。而且他总会当面把这个问题指出来。就因为这个，我们当时都挺崇拜他的；可也因此让我们都疏远了他。”

Hearing Breidbart’s anecdote retold elicits a wry smile from Stallman. “I may have been a bit of a jerk sometimes,” he admits. “But I found kindred spirits among teachers, because they, too, liked to learn. Kids, for the most part, didn’t. At least not in the same way.”

听了布莱德巴特的回忆，斯托曼笑道：“我当时可能确实有点不太会讨好别人。我总觉得在这些老师教授身上，能看到一种好学求进的精神。可在周围的学生身上，很难看到这一点。至少在学生身上看到的不太一样。”

Hanging out with the advanced kids on Saturday nevertheless encouraged Stallman to think more about the merits of increased socialization. With college fast approaching, Stallman, like many in his Columbia Science Honors Program, had narrowed his list of desired schools down to two choices: Harvard and MIT. Hearing of her son’s desire to move on to the Ivy League, Lippman became concerned. As a 15-year-old high-school junior, Stallman was still having run-ins with teachers and administrators. Only the year before, he had pulled straight A’s in American History, Chemistry, French, and Algebra, but a glaring F in English reflected the ongoing boycott of writing assignments. Such miscues might draw a knowing chuckle at MIT, but at Harvard, they were a red flag.

每周六都和这群优秀学生处在一起，斯托曼也难免会跟他们一样，考虑以后的计划。哥伦比亚科学之星计划种的学生，大多数都会把未来的大学瞄准在两所学校上：哈佛大学和麻省理工学院。李普曼知道自己的孩子希望能去所常春藤学校，她也必须要严肃对待这件事情。可看看当下的斯托曼，正值15岁，还天天和学校老师对着干。虽然上一年，斯托曼的美国历史、化学、法语、代数全都是A，可他还在继续拒绝写作文。在全A的成绩中，只有英语一门拿了个F。这种特立独行如果放在麻省理工学院，也许最多是笑笑了事，可倘若要去哈佛，则是个很严重的问题。

During her son's junior year, Lippman says she scheduled an appointment with a therapist. The therapist expressed instant concern over Stallman's unwillingness to write papers and his run-ins with teachers. Her son certainly had the intellectual wherewithal to succeed at Harvard, but did he have the patience to sit through college classes that required a term paper? The therapist suggested a trial run. If Stallman could make it through a full year in New York City public schools, including an English class that required term papers, he could probably make it at Harvard. Following the completion of his junior year, Stallman promptly enrolled in public summer school downtown and began making up the mandatory humanities classes he had shunned earlier in his high-school career.

在斯托曼11年级的时候，李普曼曾咨询过心理医生。医生一下子注意到斯托曼拒绝写作文的行为，以及他和老师之间的矛盾。斯托曼显然是聪明过人，智商上完全胜任哈佛的任何课程。可他到底能不能静下心来，踏踏实实坐在课堂里，完成每学期要求的期末论文呢？心理医生建议先让他在纽约市的公立学校读一年12年级，那里的英语课程也会要求期末论文。看看他究竟能否坚持下来。于是，11年级结束后，斯托曼进入了路易斯·D·布兰德斯高中的暑期班。这是一所位于88街的公立学校。在这所学校里，很多文科和艺术类课程都是必修课。斯托曼再也没法像以前那样躲过这些课程了。

By fall, Stallman was back within the mainstream population of New York City high-school students, at Louis D. Brandeis High School on on West 84th Street. It wasn't easy sitting through classes that seemed remedial in comparison with his Saturday studies at Columbia, but Lippman recalls proudly her son's ability to toe the line.

一个夏天过去，斯托曼终于融入了纽约市的主流公立学校。与每周六的科学之星计划的学习相比，让他踏踏实实坐在公立学校的课堂里确实不容易。可他最终还是做到了，李普曼很是骄傲。

"He was forced to kowtow to a certain degree, but he did it," Lippman says. ``I only got called in once, which was a bit of a miracle. It was the calculus teacher complaining that Richard was interrupting his lesson. I asked how he was interrupting. He said Richard was always accusing the teacher of using a false proof. I said, 'Well, is he right?' The teacher said, 'Yeah, but I can't tell that to the class. They wouldn't understand.'

"

“他被迫学会了妥协，但他最终成功了。”李普曼说，“我居然只被老师们请去了一次，这简直算是奇迹了。当时微积分老师把我请过去了，因为理查德经常在上课期间打断他讲课。我问老师他因为什么打断讲课。他说理查德总是会说老师的证明不够严谨。我就问老师：‘那他说对了吗？’老师说：‘对是对了，可我不可能给全班讲这些证明，那样的话其他学生都会听不懂的。’”

By the end of his first semester at Brandeis High, things were falling into place. A 96 in English wiped away much of the stigma of the 60 earned 2 years before. For good measure, Stallman backed it up with top marks in American History, Advanced Placement Calculus, and Microbiology. The crowning touch was a perfect 100 in Physics. Though still a social outcast, Stallman finished his 10 months at Brandeis as the fourth-ranked student in a class of 789.

第一个学期过去了，一切皆见了分晓。英语课96分，一雪前耻。其他课程，包括美国历史，大学先修课程的微积分，微生物学，斯托曼依旧遥遥领先。物理课斯托曼则满分过关。尽管斯托曼依旧不合群，但他终究是在布兰德斯高中完成了11个月的学习，并且在789班取得了全班第四的成绩。

Outside the classroom, Stallman pursued his studies with even more diligence, rushing off to fulfill his laboratory-assistant duties at Rockefeller University during the week and dodging the Vietnam protesters on his way to Saturday school at Columbia. It was there, while the rest of the Science Honors Program students sat around discussing their college choices, that Stallman finally took a moment to participate in the preclass bull session.

学校以外的活动里，斯托曼表现得更加勤奋。每周他都会去洛克菲勒大学，完成实验员的工作；周六则一路避开反越战抗议者的队伍，去哥伦比亚大学参加科学之星计划的学习班。终于又有机会，科学之星计划的学生凑在一起，聊起未来将去哪所大学。

Recalls Breidbart, “Most of the students were going to Harvard and MIT, of course, but you had a few going to other Ivy League schools. As the conversation circled the room, it became apparent that Richard hadn’t said anything yet. I don’t know who it was, but somebody got up the courage to ask him what he planned to do.”

布莱德巴特回忆：“绝大部分学生都会去哈佛或者麻省理工学院。当然也有人去其他几所常春藤学校。大家聊着说着，斯托曼始终没说话。这时候有个人跳出

来，问斯托曼究竟去哪所大学。”

Thirty years later, Breidbart remembers the moment clearly. As soon as Stallman broke the news that he, too, would be attending Harvard University in the fall, an awkward silence filled the room. Almost as if on cue, the corners of Stallman’s mouth slowly turned upward into a self-satisfied smile.

三十多年过去了，布莱德巴特依旧清晰地记得那一刻。“哈佛大学”几个字从斯托曼口中一出，人群一时没了声音。一切都好像事先排练好的一样，坐在角落中的斯托曼露出了一丝微笑。

Says Breidbart, ``It was his silent way of saying, ‘That’s right. You haven’t got rid of me yet.’

”

布莱德巴特说：“那微笑好像在说：‘没错，你们还没把我甩开呢。’”

Impeach God 逆天行道，选举上帝

Although their relationship was fraught with tension, Richard Stallman would inherit one noteworthy trait from his mother: a passion for progressive politics.

尽管理查德·斯托曼和母亲的关系比较紧张，但他还是继承了母亲的一个特质：对激进政治极度热情。

It was an inherited trait that would take several decades to emerge, however. For the first few years of his life, Stallman lived in what he now admits was a “political vacuum.” Like most Americans during the Eisenhower age, the Stallman family spent the Fifties trying to recapture the normalcy lost during the wartime years of the 1940s.

可这个特质要在十几年之后才在斯托曼身上显现出来。他早年的生活，用他现在的话说，是与“政治绝缘”的。正如艾森豪威尔时期的大多数美国家庭一样，五十年代，斯托曼一家试图重拾二战以前的那份宁静、安逸的正常生活。

“Richard’s father and I were Democrats but happy enough to leave it at that,” says Lippman, recalling the family’s years in Queens. “We didn’t get involved much in local or national politics.”

“理查德的父亲和我虽然都是民主党人，可我们觉得那段时间的生活还算过得去。”李普曼回忆着当年和理查德父亲住在纽约市皇后区的日子，“我们当时并没有参与太多的当地或者全国的政治活动。”

That all began to change, however, in the late 1950s when Alice divorced Daniel Stallman. The move back to Manhattan represented more than a change of address; it represented a new, independent identity and a jarring loss of tranquility.

然而，五十年代末期，李普曼的离异打破了这一切。她要重新搬回纽约市曼哈顿区，这不仅仅是改个住址，它更意味着一份全新的独立生活的开始，更意味着重回喧嚣的大都市生活。

“I think my first taste of political activism came when I went to the Queens public library and discovered there was only a single book on divorce in the whole library,” recalls Lippman. “It was very controlled by the Catholic church, at least in Elmhurst, where we lived. I think that was the first inkling I had of the forces that quietly control our lives.”

“关于政治活动，我最初的体验是在离婚期间。当时我走进皇后区公共图书馆，却发现里面只有一本关于离婚的书。那里当时被天主教严格控制着，至少在我们住的阿母赫斯特区是这样的。这是我第一次意识到，我们的日常生活，被周围的各种权威力量悄悄地控制着。”李普曼说。

Returning to her childhood neighborhood, Manhattan’s Upper West Side, Lippman was shocked by the changes that had taken place since her departure to Hunter College a decade and a half before. The skyrocketing demand for post-war housing had turned the neighborhood into a political battleground. On one side stood the pro-development city-hall politicians and businessmen hoping to rebuild many of the neighborhood’s blocks to accommodate the growing number of white-collar workers moving into the city. On the other side stood the poor Irish and Puerto Rican tenants who had found an affordable haven in the neighborhood.

重回童年居住过的曼哈顿上西城，李普曼被周围的变化震惊了。十五年前，她离家去亨特学院读书。如今，由于战后住房资源紧张，当地居民对高楼有大量需求，把几个街区变成了一个政治决斗场，把人们分成了对立的两组。一组人，主要是政客和商人，他们希望把这一片拆迁扩建，以应付越来越多的白领移居到这里；另一组人，主要是来自爱尔兰或波多黎各的房客，他们都不富

裕，所幸已经在周围找到了便宜舒适的住房。

At first, Lippman didn't know which side to choose. As a new resident, she felt the need for new housing. As a single mother with minimal income, however, she shared the poorer tenants' concern over the growing number of development projects catering mainly to wealthy residents. Indignant, Lippman began looking for ways to combat the political machine that was attempting to turn her neighborhood into a clone of the Upper East Side.

一开始，李普曼不知道站到哪一边。作为一个新来的住户，她觉得的确有必要扩建。可作为一个拿着微薄收入的单亲母亲，她和那些房客有着一样的忧虑，觉得周围越来越多的项目，都是为有钱人开发的。心中忿忿不平，李普曼开始寻找机会，来和庞大的政治机器做斗争，避免让政客把上西城变成上东城那样，只供有钱人吃喝玩乐。

Lippman says her first visit to the local Democratic party headquarters came in 1958. Looking for a day-care center to take care of her son while she worked, she had been appalled by the conditions encountered at one of the city-owned centers that catered to low-income residents. "All I remember is the stench of rotten milk, the dark hallways, the paucity of supplies. I had been a teacher in private nursery schools. The contrast was so great. We took one look at that room and left. That stirred me up."

李普曼第一次去民主党总部是在1958年。当时，她希望为理查德找个日托所，以便在自己工作的时候，有人照顾理查德。在本市的低收入人员帮助中心里，她一下子被那里的环境吓到了。"我就记得当时那股酸臭的坏牛奶味儿，黑洞洞的走廊，还有那么一丁点的救助物资。我以前是护士学校的老师，那里的环境和这里反差太大了。我们看了一眼那个日托所的房间，就立马走人了。那环境太让我恶心。"

The visit to the party headquarters proved disappointing, however. Describing it as "the proverbial smoke-filled room," Lippman says she became aware for the first time that corruption within the party might actually be the reason behind the city's thinly disguised hostility toward poor residents. Instead of going back to the headquarters, Lippman decided to join up with one of the many clubs aimed at reforming the Democratic party and ousting the last vestiges of the Tammany Hall machine. Dubbed the Woodrow Wilson/FDR Reform Democratic Club, Lippman and her club began showing up at planning and city-council meetings, demanding a greater say.

这次造访民主党总部，让李普曼很失望。“大家说的一点也没错，乌烟瘴气，不是个干净地方。”李普曼说，这是她第一次意识到，党内腐败也许才是一切症结所在，大家也许才因此鄙视甚至仇视穷人。李普曼没有再踏入民主党总部一步。她看到当地众多俱乐部，都志在促进民主党的党内改革。李普曼加入了其中的一个俱乐部：伍德罗·威尔逊/FDR民主党改革俱乐部，从此与坦慕尼协会余党作斗争。李普曼开始出现在俱乐部内部的例行会议中，和市政委员会会议会上，以此她取得了更多话语权。

“Our primary goal was to fight Tammany Hall, Carmine DeSapio and his henchman,” says Lippman. “I was the representative to the city council and was very much involved in creating a viable urban-renewal plan that went beyond simply adding more luxury housing to the neighborhood.”

李普曼说：“我们的主要目标，是对抗以卡米思·德·萨皮奥(Carmine De Sapio)为首的坦慕尼协会和它的追随者。我是市政委员会的一名代表委员。主要提议修建更多廉价住房，反对单纯修建舒适豪宅。”

Such involvement would blossom into greater political activity during the 1960s. By 1965, Lippman had become an “outspoken” supporter for political candidates like William Fitts Ryan, a Democrat elected to Congress with the help of reform clubs and one of the first U.S. representatives to speak out against the Vietnam War.

这种参与在六十年代变成了更大的政治活动。1965年，李普曼已经开始公开支持一些民主党的议会候选人，比如威廉·菲茨·赖安，他在各个民主党改革俱乐部的帮助下，进入议会，并成为当时第一批反对越战的议员。

It wasn’t long before Lippman, too, was an outspoken opponent of U.S. involvement in Indochina. “I was against the Vietnam War from the time Kennedy sent troops,” she says. “I had read the stories by reporters and journalists sent to cover the early stages of the conflict. I really believed their forecast that it would become a quagmire.”

不久之后，李普曼也开始公开反对美国涉足印度支那问题。“肯尼迪政府把军队送去越南开始，我就一直反对越战。从一开始，我就读各种来自前线的报道文章。很多文章都预测美国政府会因此深陷泥潭，我也非常赞同这个观点。”

Such opposition permeated the Stallman-Lippman household. In 1967, Lippman remarried. Her new husband, Maurice Lippman, a major in the Air

National Guard, resigned his commission to demonstrate his opposition to the war. Lippman's stepson, Andrew Lippman, was at MIT and temporarily eligible for a student deferment. Still, the threat of induction should that deferment disappear, as it eventually did, made the risk of U.S. escalation all the more immediate. Finally, there was Richard who, though younger, faced the prospect of being drafted as the war lasted into the 1970s.

这种反战之声充斥在理查德的家中。1967年，李普曼再婚。她的新任丈夫莫里斯·李普曼是一名空军少校，越战开始他辞职以示反战。莫里斯的儿子，李普曼的继子，安德鲁·李普曼当时正在麻省理工学院读书。他暂时还可以申请延期入伍。可随着战事升级，战争时间一久，入伍时间到期，他还是要去服役。当战事延续到七十年代，理查德虽然年纪还小，可也不得不考虑入伍的问题。他要么选择去越南参战，要么选择到加拿大避开兵役。

"Vietnam was a major issue in our household," says Lippman. "We talked about it constantly: what would we do if the war continued, what steps Richard or his stepbrother would take if they got drafted. We were all opposed to the war and the draft. We really thought it was immoral."

李普曼说："越南问题成了家里的重点，我们总是不停地讨论它：如果战争持续下去我们怎么办？要是理查德或者安德鲁收到征兵令怎么办。我们全家都反对这征兵令，更反对这场战争。我们从心里觉得这场战争是不道德的。"

For Stallman, the Vietnam War elicited a complex mixture of emotions: confusion, horror, and, ultimately, a profound sense of political impotence. As a kid who could barely cope in the mild authoritarian universe of private school, Stallman experienced a shiver whenever the thought of Army boot camp presented itself. He did not think he could get through it and emerge sane.

对于理查德来说，越战给他带来的情感是复杂的：恐惧，不解，以及最终给他带来的一种政治上无能为力的感受。理查德如此地厌恶权威，他甚至无法忍受私立学校的威权制度。别说参军，哪怕就只让他想想军队里的训练，都会令他不寒而栗。

"I was devastated by the fear, but I couldn't imagine what to do and didn't have the guts to go demonstrate," recalls Stallman, whose March 16th birthday earned him a low number in the dreaded draft lottery. This did not affect him immediately, since he had a college deferment, one of the last before the U.S.

stopped granting them; but it would affect him in a few years. “I couldn’t envision moving to Canada or Sweden. The idea of getting up by myself and moving somewhere. How could I do that? I didn’t know how to live by myself. I wasn’t the kind of person who felt confident in approaching things like that.”

“我当时被吓坏了。可我也不知道能做什么，该怎么做。更没胆量去上街参加游行。”理查德如是说。1971年，政府最终取消了学生延期入伍的政策。抽签征兵的结果让理查德很是担心：他3月18日的生日，征兵顺序中比较靠前。“我很难想象要移民到加大拿或者瑞典。要一个人离开过日子，我当时可不行。我根本不知道怎么照顾自己，我对自己在生活自理方面的能力非常不自信。”

Stallman says he was impressed by the family members who did speak out. Recalling a sticker, printed and distributed by his father, likening the My Lai massacre to similar Nazi atrocities in World War II, he says he was “excited” by his father’s gesture of outrage. “I admired him for doing it,” Stallman says. “But I didn’t imagine that I could do anything. I was afraid that the juggernaut of the draft was going to destroy me.”

理查德说，他至今依然感慨家人在公开场合的反战行为。他记得父亲车上有一个车贴，把美莱村屠杀比作纳粹的大屠杀。这个车贴是父亲亲自制作的，他还做了很多这样的车贴，分发给大家。理查德说，他很受父亲情绪的鼓舞。“我很崇拜他这么做。”理查德说，“可我没想过自己能做什么。当时我很害怕自己的一辈子就这样被征兵令毁了。”

However, Stallman says he was turned off by the tone and direction of much of that movement. Like other members of the Science Honors Program, he saw the weekend demonstrations at Columbia as little more than a distracting spectacle. Ultimately, Stallman says, the irrational forces driving the anti-war movement became indistinguishable from the irrational forces driving the rest of youth culture. Instead of worshiping the Beatles, girls in Stallman’s age group were suddenly worshiping firebrands like Abbie Hoffman and Jerry Rubin. To a kid already struggling to comprehend his teenage peers, slogans like “make love not war” had a taunting quality. Stallman did not want to make war, at least not in Southeast Asia, but nobody was inviting him to make love either.

可理查德也逐渐开始不喜欢反战运动的方向和调调。正如其他科学之星计划的学生一样，理查德每周末都会看到聚集在哥伦比亚大学的抗议群众。最终，理查德形容说，各种非理智力量把反战运动变得与各种其他青年人的非理智活动

一样可怕。一瞬间，理查德那个年级的姑娘们都不再崇拜披头士，改去崇拜各路反战领袖，如阿比·霍夫曼（Abbie Hoffman）和杰里·鲁宾（Jerry Rubin）。理查德，这个中学生里的异类，面对各种流行趋势已然有些应接不暇了，如今又来了“要做爱不作战”这种花哨口号，实在让他觉得甚是沮丧。他显然不喜欢战争，至少不喜欢这次在东南亚的战争。当然这也并不意味着就有会有个姑娘愿意与他一起共度春宵。

“I didn’t like the counter culture much,” Stallman recalls. “I didn’t like the music. I didn’t like the drugs. I was scared of the drugs. I especially didn’t like the anti-intellectualism, and I didn’t like the prejudice against technology. After all, I loved a computer. And I didn’t like the mindless anti-Americanism that I often encountered. There were people whose thinking was so simplistic that if they disapproved of the conduct of the U.S. in the Vietnam War, they had to support the North Vietnamese. They couldn’t imagine a more complicated position, I guess.”

“我并不喜欢这种对抗主流的文化。”理查德坦言，“我不喜欢他们的音乐，不喜欢毒品。我可害怕毒品。我更厌恶他们那套反理性反知识的论调，厌恶他们对各种技术的偏见。因为无论怎么说，我还是喜欢计算机的。我还经常遇到各种没头脑的反美意见，这也让我很反感。有些人的头脑真的太过简单了，他们觉得如果要反对美国参加越战，就意味着支持北越政权。他们就天真到想不出另外一种可能。”

Such comments underline a trait that would become the key to Stallman’s own political maturation. For Stallman, political confidence was directly proportionate to personal confidence. By 1970, Stallman had become confident in few things outside the realm of math and science. Nevertheless, confidence in math gave him enough of a foundation to examine the extremes of the anti-war movement in purely logical terms. Doing so, Stallman found the logic wanting. Although opposed to the war in Vietnam, Stallman saw no reason to disavow war as a means for defending liberty or correcting injustice.

这样的评论标志着理查德自己政治思想的逐步成熟。对于理查德来说，政治上的信心和自信心成正比。到了1970，理查德已经在数学和科学以外的一些领域里树立起了自信。当然，在数学方面的信心和能力，作为基础，让理查德可以从纯逻辑的角度分析越战。这种分析最终带来的结论，让理查德觉得，尽管反对越战，但不可否认，战争在维护自由和捍卫正义方面起到的积极作用。

In the 1980s, a more confident Stallman decided to make up for his past

inactivity by participating in mass rallies for abortion rights in Washington DC. “I became dissatisfied with my earlier self for failing in my duty to protest the Vietnam War,” he explains.

八十年代，理查德已经不再沉默。他带着这份自信，在华盛顿市参加了维护堕胎权利的聚会。“我开始不满从前的自己，觉得不该像当初那样逃避游行抗议，这是我的权利，更是我的责任。”

In 1970, Stallman left behind the nightly dinnertime conversations about politics and the Vietnam War as he departed for Harvard. Looking back, Stallman describes the transition from his mother’s Manhattan apartment to life in a Cambridge dorm as an “escape.” At Harvard, he could go to his room and have peace whenever he wanted it. Peers who watched Stallman make the transition, however, saw little to suggest a liberating experience.

1970年，理查德离开了家，离开了每晚家中餐桌上的越战和政治讨论，去了哈佛大学。理查德回忆，从曼哈顿母亲的公寓，转到马萨诸塞州剑桥市的哈佛大学宿舍，对自己来说是一种“逃离”。可同龄人来看，他的这次大逃离并没有给他带来多少解放。

“He seemed pretty miserable for the first while at Harvard,” recalls Dan Chess, a classmate in the Science Honors Program who also matriculated at Harvard. “You could tell that human interaction was really difficult for him, and there was no way of avoiding it at Harvard. Harvard was an intensely social kind of place.”

“他在哈佛的第一年很痛苦。”丹·柴斯，另外一位去了哈佛大学的科学之星的学生，回忆道，“简单的人际交往对于理查德来说都是非常困难的。可这种交往在哈佛又是无法避免的。哈佛校园就是个大社交场。”

To ease the transition, Stallman fell back on his strengths: math and science. Like most members of the Science Honors Program, Stallman breezed through the qualifying exam for Math 55, the legendary “boot camp” class for freshman mathematics “concentrators” at Harvard. Within the class, members of the Science Honors Program formed a durable unit. “We were the math mafia,” says Chess with a laugh. “Harvard was nothing, at least compared with the SHP.”

为了让这种过渡更顺畅，理查德重新埋头在自己的老伙伴中间：数学和自然科

学。与很多科学之星计划的成员一道，理查德轻松通过数学55的课前考试，获得了选修数学55的资格。所谓数学55，是哈佛大学一门数学类课程的官方编号。它专门为未来数学家设计，以难度大而闻名全校。在这门课上，科学之星的成员凑成了一个小团体。“我们数学党的。”柴斯笑道：“跟科学之星比，哈佛不在话下。”

To earn the right to boast, however, Stallman, Chess, and the other SHP alumni had to get through Math 55. Promising four years worth of math in two semesters, the course favored only the truly devout. “It was an amazing class,” says David Harbater, a former “math mafia” member and now a professor of mathematics at the University of Pennsylvania. “It’s probably safe to say there has never been a class for beginning college students that was that intense and that advanced. The phrase I say to people just to get it across is that, among other things, by the second semester we were discussing the differential geometry of Banach manifolds. That’s usually when their eyes bug out, because most people don’t start talking about Banach manifolds until their second year of graduate school.”

可要修成正果，理查德，柴斯和其他科学之星成员，必须要通过数学55。这个课程把本该四年学完的数学知识，全部放在两个学期里。只有真正肯下功夫的人，才能啃下这块硬骨头。“那真是门让人神往的课。”戴维·哈伯特（David Harbater）回忆道。他曾是这门课上“数学党”的成员，如今已是宾夕法尼亚大学数学系教授。“很保守地说，几乎没有几个给大学新生设计的课程，能这么难，这么深，课业量也很少有这么大的。我经常给人这么形容这门课的难度：第二个学期，我们已经开始讲巴拿赫空间的微分几何了。一般人听了肯定瞪大眼睛，因为绝大多数人到了研究生第二年才开始说点巴拿赫空间的东西。”

Starting with 75 students, the class quickly melted down to 20 by the end of the second semester. Of that 20, says Harbater, “only 10 really knew what they were doing.” Of that 10, 8 would go on to become future mathematics professors, 1 would go on to teach physics.

那年这门课刚一开始有75个学生，到第二学期结束，就只有20个学生还在坚持。戴维·哈伯特说，这20个学生当中，“只有10个人真正听懂了课上讲的是什么。这10个人里，8个人后来做了数学教授，另外有一个做了物理教授。”

“The other one,” emphasizes Harbater, “was Richard Stallman.”

“最后那一个，”戴维·哈伯特强调，“就是理查德·斯托曼。”

Seth Breidbart, a fellow Math 55 classmate, remembers Stallman distinguishing himself from his peers even then.

赛思·布莱德巴特也是数学55课上的学生，他记得当时理查德就显出了与众不同的特质。

“He was a stickler in some very strange ways,” says Breidbart. There is a standard technique in math which everybody does wrong. It’s an abuse of notation where you have to define a function for something and what you do is you define a function and then you prove that it’s well defined. Except the first time he did and presented it, he defined a relation and proved that it’s a function. It’s the exact same proof, but he used the correct terminology, which no one else did. That’s just the way he was.”

“遇到一些事，他总是特别能坚持。当时，我们经常用一种错误的方法来解题。每次我们要定义一个函数的时候，我们都会先定义一个函数，然后证明这个函数是良构的。这么做本身是用错了术语。理查德第一次也犯了这个错。可后来，他都是先定义一个关系，然后证明这个关系是一个函数。其实证明的过程都一样，但他用了正确的符号和术语。而其他没有一个人这么做。他就是这么个人。”

It was in Math 55 that Richard Stallman began to cultivate a reputation for brilliance. Breidbart agrees, but Chess, whose competitive streak refused to yield, says the realization that Stallman might be the best mathematician in the class didn’t set in until the next year. “It was during a class on Real Analysis,” says Chess, now a math professor at Hunter College. “I actually remember in a proof about complex valued measures that Richard came up with an idea that was basically a metaphor from the calculus of variations. It was the first time I ever saw somebody solve a problem in a brilliantly original way.”

数学55的课堂上，理查德又一次展现了他的聪明才智。布莱德巴特早早就承认了这一点。而柴斯好胜的个性则很难让他服输，但到了第二年，他还是不得不承认，理查德也许是整个课堂中最优秀的数学家。如今，柴斯已是亨特学院数学系的教授，他回忆：“在一堂实验分析的课上，理查德想出了一个关于复测度的证明，那个证明实际上是借鉴了变分法里的一些技巧。我是第一次看到有人能自己想出来，用这么漂亮的方法解决这个问题的。”

For Chess, it was a troubling moment. Like a bird flying into a clear glass window, it would take a while to realize that some levels of insight were simply

off limits.

对于柴斯来说，这的确是个恼人的时刻。他就好像一只小鸟飞到了透明玻璃前，他要花些时间，才会不甘心地意识到，有些地方，虽然看得到，却未必是自己能力所及的。

“That’s the thing about mathematics,” says Chess. “You don’t have to be a first-rank mathematician to recognize first-rate mathematical talent. I could tell I was up there, but I could also tell I wasn’t at the first rank. If Richard had chosen to be a mathematician, he would have been a first-rank mathematician.”

“这恐怕就是数学之美吧。”柴斯说：“你不必需要成为一流的数学家，就可以欣赏数学天才的作品。我就处在这么一个尴尬的地方。我算不上一流数学家。如果理查德当初选择做数学家的话，他肯定会是一流的。”

For Stallman, success in the classroom was balanced by the same lack of success in the social arena. Even as other members of the math mafia gathered to take on the Math 55 problem sets, Stallman preferred to work alone. The same went for living arrangements. On the housing application for Harvard, Stallman clearly spelled out his preferences. “I said I preferred an invisible, inaudible, intangible roommate,” he says. In a rare stroke of bureaucratic foresight, Harvard’s housing office accepted the request, giving Stallman a one-room single for his freshman year.

有得就必然有失，理查德在课堂上的得意，换不来社交上的成功。其他数学党的成员一般都会凑在一起讨论数学55的作业，而理查德则自己一个人应付作业题。住宿问题上理查德也坚持如此。在哈佛大学住宿申请的表格上，理查德清晰地说出了自己对室友的要求：“我说我希望能有个几乎看不见，听不见，摸不着的室友。”宿舍管理部门这次倒是难得如此识相，竟然接受了理查德的要求，在他入学的第一年，给他安排了一个单人间。

Breidbart, the only math-mafia member to share a dorm with Stallman that freshman year, says Stallman slowly but surely learned how to interact with other students. He recalls how other dorm mates, impressed by Stallman’s logical acumen, began welcoming his input whenever an intellectual debate broke out in the dining club or dorm commons.

布莱德巴特是数学党成员中，唯一一个在大一期间和理查德同住一个宿舍楼

的。他回忆，理查德当时的确慢慢地学着如何和别人沟通交往。他记得，当时其他几个宿舍的学生，都被理查德严密的逻辑分析能力感染，都喜欢跟他凑在宿舍楼的大厅或者餐厅里，谈天说地。

“We had the usual bull sessions about solving the world’s problems or what would be the result of something,” recalls Breidbart. “Say somebody discovers an immortality serum. What do you do? What are the political results? If you give it to everybody, the world gets overcrowded and everybody dies. If you limit it, if you say everyone who’s alive now can have it but their children can’t, then you end up with an underclass of people without it. Richard was just better able than most to see the unforeseen circumstances of any decision.”

布莱德巴特说：“我们凑一起经常会扯上一些异想天开的事情，比如如何解决各种世界问题，或者某种东西如果成真了会怎么样。好比说，有人发明了一种长生不老药会如何。你该怎么办？这东西的政治影响会是什么？如果说每人都吃了这个药，最后谁都不会老，最后人越来越多，到头来还是会因为资源困乏，大家都得死掉。老是不会老了，但终究也会死。可如果限制发布这个药，好比你可以说，只有现在活着的人能拿到这个药，新生儿不能吃长生不老药，那最后世界上就会被分出一批下等人。理查德总能比大家更早看出各种决策的优劣。”

Stallman remembers the discussions vividly. “I was always in favor of immortality,” he says. “How else would we be able to see what the world is like 200 years from now?” Curious, he began asking various acquaintances whether they would want immortality if offered it. “I was shocked that most people regarded immortality as a bad thing.” Many said that death was good because there was no use living a decrepit life, and that aging was good because it got people prepared for death, without recognizing the circularity of the combination.

理查德至今还清晰地记得当时的讨论：“我总是对长生不老的话题感兴趣。我们要是能看到200年之后，世界是什么样子，那会如何？”出于好奇，理查德找了几个朋友，问他们愿意不愿意长生不老：“我很惊讶，不少人觉得长生不老并非什么好事情。”很多人说，死亡从某种意义上讲是个好事，因为一直活下去的话生命会显得越来越没有意义。而苍老的好处则是让人们能逐渐意识到死亡的到来。

Although perceived as a first-rank mathematician and first-rate informal debater, Stallman shied away from clear-cut competitive events that might

have sealed his brilliant reputation. Near the end of freshman year at Harvard, Breidbart recalls how Stallman conspicuously ducked the Putnam exam, a prestigious test open to math students throughout the U.S. and Canada. In addition to giving students a chance to measure their knowledge in relation to their peers, the Putnam served as a chief recruiting tool for academic math departments. According to campus legend, the top scorer automatically qualified for a graduate fellowship at any school of his choice, including Harvard.

尽管众人已经公认，理查德会是个一流数学家，也是个非正式的辩论能手，可他自己却一直避免参加各种有明确排名或分界线的竞赛。布莱德巴特清晰地记得，理查德大一的时候，大家眼看着他回避参加帕特南竞赛（Putnam Competition）。帕特南竞赛是专门针对美国和加拿大数学系的本科生的竞赛。参赛者借此可以知道自己的数学水平，更重要的，比赛的结果经常被各大院校当作选拔研究生和博士生的依据。按照学生中间的流言来看，如果拿到好成绩，就可以去任何一所大学读研究生或博士，而且是免学费，全额奖学金。当然，这也包括哈佛大学。

Like Math 55, the Putnam was a brutal test of merit. A six-hour exam in two parts, it seemed explicitly designed to separate the wheat from the chaff. Breidbart, a veteran of both the Science Honors Program and Math 55, describes it as easily the most difficult test he ever took. “Just to give you an idea of how difficult it was,” says Breidbart, “the top score was a 120, and my score the first year was in the 30s. That score was still good enough to place me 101st in the country.”

和数学55一样，帕特南竞赛也是各残酷的竞技场。两场考试，一共六个小时。这比赛是铁了心要把学生分出三六九等。布莱德巴特，这位参加了科学之星计划，也上了数学55的学生，依然觉得这个竞赛是他参加过的最难的考试。“就跟你这么说吧，满分120，我第一年的时候分数是30几分。就这分数，我还能排上第101名。”

Surprised that Stallman, the best student in the class, had skipped the test, Breidbart says he and a fellow classmate cornered him in the dining common and demanded an explanation. “He said he was afraid of not doing well,” Breidbart recalls.

可让人吃惊的是，理查德，这位全班数学最好的学生，竟然不参加这个比赛。布莱德巴特记得当时他们听到这个消息之后，几个人把理查德围在墙角，让他

解释为什么不参加。“他说他怕做得不好。”布莱德巴特回忆。

Breidbart and the friend quickly wrote down a few problems from memory and gave them to Stallman. “He solved all of them,” Breidbart says, “leading me to conclude that by not doing well, he either meant coming in second or getting something wrong.”

一听这话，布莱德巴特和其他几个学生迅速地写下了几个帕特南竞赛得题目，扔给理查德，让他做。“他全都做对了。我当时就觉得，所谓‘做得不好’，对他来说也许就是拿个亚军，或者做错一道题。”

Stallman remembers the episode a bit differently. “I remember that they did bring me the questions and it’s possible that I solved one of them, but I’m pretty sure I didn’t solve them all,” he says. Nevertheless, Stallman agrees with Breidbart’s recollection that fear was the primary reason for not taking the test. Despite a demonstrated willingness to point out the intellectual weaknesses of his peers and professors in the classroom, Stallman hated and feared the notion of head-to-head competition – so why not just avoid it?

理查德对这个事情的记忆则略有出入。“我确实记得他们给我题目了，可我好像记得我就做出来一道，反正肯定没全做对。”无论如何，理查德承认，正如布莱德巴特说的，他当时的确是出于害怕才不参加这竞赛的。理查德只是在班上指出同学或者老师的错误。但他却不喜欢，甚至害怕参加任何正面竞争。结果，他就总是避开类似的比赛。

“It’s the same reason I never liked chess,” says Stallman. “Whenever I’d play, I would become so consumed by the fear of making a single mistake and losing that I would start making stupid mistakes very early in the game. The fear became a self-fulfilling prophecy.” He avoided the problem by not playing chess.

理查德说：“同样的原因，我也不喜欢下棋。每次我下棋的时候，我都害怕那种一步走错，满盘皆输的情况。而这种恐惧，最后总是成真。”于是，理查德也就只能靠避免下棋，来解决这个问题。

Whether such fears ultimately prompted Stallman to shy away from a mathematical career is a moot issue. By the end of his freshman year at Harvard, Stallman had other interests pulling him away from the field. Computer programming, a latent fascination throughout Stallman’s high-school

years, was becoming a full-fledged passion. Where other math students sought occasional refuge in art and history classes, Stallman sought it in the computer-science laboratory.

究竟是不是因为这种恐惧，才没让理查德成为数学家，我们不得而知。但无论如何，大一结束的时候，理查德已经有了新的兴趣：编程。这兴趣从理查德中学的时候就逐渐形成，但潜伏了很久，最终在大一后期成为了他的一个主要兴趣，让他投入了自己的全部热情。其他数学系的学生都靠上艺术和历史课来放松，理查德则跑去机房缓解压力。

For Stallman, the first taste of real computer programming at the IBM New York Scientific Center had triggered a desire to learn more. “Toward the end of my first year at Harvard school, I started to have enough courage to go visit computer labs and see what they had. I’d ask them if they had extra copies of any manuals that I could read.” Taking the manuals home, Stallman would examine the machine specifications to learn about the range of different computer designs.

当年第一次在IBM纽约科学中心编程的经历，诱使着理查德去了解更多。他说：“在哈佛学习快一年的时候，我开始跑去哈佛的几个计算机实验室，看看他们那里有什么新东西。刚到那里，我就问他们能不能给我一份使用手册。”把这些使用手册拿回家，理查德开始自习阅读，比较各个机器这间的差别和联系。

One day, near the end of his freshman year, Stallman heard about a special laboratory near MIT. The laboratory was located on the ninth floor of a building in Tech Square, the mostly-commercial office park MIT had built across the street from the campus. According to the rumors, the lab itself was dedicated to the cutting-edge science of artificial intelligence and boasted the cutting-edge machines and software to match.

第一年快结束的时候，理查德有天听说在麻省理工学院，有个特殊的实验室。那个实验室就在校区旁边的技术广场大厦九层。传说，这个实验室是专门为顶尖的人工智能研究而设立，里面各种高级计算机和软件。

Intrigued, Stallman decided to pay a visit.

理查德被这一切吸引住了，决定亲自前往一探究竟。

The trip was short, about 2 miles on foot, 10 minutes by train, but as Stallman would soon find out, MIT and Harvard can feel like opposite poles of the same

planet. With its maze-like tangle of interconnected office buildings, the Institute's campus offered an aesthetic yin to Harvard's spacious colonial-village yang. Of the two, the maze of MIT was much more Stallman's style. The same could be said for the student body, a geeky collection of ex-high school misfits known more for its predilection for pranks than its politically powerful alumni.

从哈佛大学到那里并不远，走路大概三公里，坐地铁只要十分钟。可等到了那里，理查德才发现，麻省理工学院和哈佛大学，简直就像两个世界。麻省理工学院里真可说是“五步一楼，十步一阁”。各个建筑纠缠在一起；相比之下，哈佛大学则显得明快宽敞。两所大学，一张一弛，一阴一阳。这不仅体现在建筑风格上，在校风学风上也是如此。麻省理工学院的学生，多是中学里的怪才，他们好开玩笑，喜欢恶作剧；而哈佛大学的学生，则更多是家有深厚背景的孩子，或是从小就有政治抱负的青年。

The yin-yang relationship extended to the AI Lab as well. Unlike Harvard computer labs, there was no grad-student gatekeeper, no clipboard waiting list for terminal access, no atmosphere of “look but don't touch.” Instead, Stallman found only a collection of open terminals and robotic arms, presumably the artifacts of some AI experiment. When he encountered a lab employee, he asked if the lab had any spare manuals it could loan to an inquisitive student. “They had some, but a lot of things weren't documented,” Stallman recalls. “They were hackers, after all,” he adds wryly, referring to hackers' tendency to move on to a new project without documenting the last one.

这种学校风格同样延伸到计算机实验室中。与哈佛的各个计算机实验室不同，麻省理工大学的这个人工智能实验室，没有门卫；没有拿着小本子，记下等候使用终端的人，给他们排队的人；也没有贴着“严禁触摸”标识的那种氛围。到了那里，理查德就看到了几个空着没人用的终端，和几个机械臂，提醒着人们这里正在做着人工智能方面的试验。理查德找到一位在这里上班的雇员，问他能不能送自己一份使用手册。理查德回忆：“那人说他手头确实有几份闲置的手册，可好多东西都没写在手册里。怎么说，他们也是黑客。”理查德笑道，暗指黑客们往往会直奔下一个项目，而不去把之前的项目记录在文档里。

Stallman left with something even better than a manual: A job. His first project was to write a PDP-11 simulator that would run on a PDP-10. He came back to the AI Lab the next week, grabbing an available terminal, and began writing the code.

在对方的帮助下，理查德最终得到了一份比手册更好的东西：一个在实验室的工作职位。他的第一个工作任务，是在PDP-10上写一个PDP-11的模拟器。于是，他第二周又来到人工智能实验室，找了个终端，开始写代码。

Looking back, Stallman sees nothing unusual in the AI Lab's willingness to accept an unproven outsider at first glance. "That's the way it was back then," he says. "That's the way it still is now. I'll hire somebody when I meet him if I see he's good. Why wait? Stuffy people who insist on putting bureaucracy into everything really miss the point. If a person is good, he shouldn't have to go through a long, detailed hiring process; he should be sitting at a computer writing code."

回想起来，理查德觉得，人工智能实验室把一份工作"随便"扔给一个素不相识的外校学生这件事，并没什么不妥。"当年都是这样，如今也是如此。我见一个人，他要是很优秀，我就雇他。等什么呢？很多刻板的人喜欢在每件事上都安插各种官僚流程。这实在太愚蠢。要是一个人真的很优秀，他就不会花时间准备各种冗长拖沓的面试。他应该踏踏实实坐电脑前头写代码。"

To get a taste of "bureaucratic and stuffy," Stallman need only visit the computer labs at Harvard. There, access to the terminals was doled out according to academic rank. As an undergrad, Stallman sometimes had to wait for hours. The waiting wasn't difficult, but it was frustrating. Waiting for a public terminal, knowing all the while that a half dozen equally usable machines were sitting idle inside professors' locked offices, seemed the height of irrational waste. Although Stallman continued to pay the occasional visit to the Harvard computer labs, he preferred the more egalitarian policies of the AI Lab. "It was a breath of fresh air," he says. "At the AI Lab, people seemed more concerned about work than status."

要想了解什么是"刻板"和"官僚流程"，理查德说只要去哈佛的计算机实验室看看就行了。在那里，使用终端的先后顺序，是按照年级排序的。作为一个本科生，理查德经常要等好几个小时才能用上计算机。等待本身并不是件难事，但却是个很无聊的事情。尤其是你明明知道，在各个教授的办公室里，都有一个能用的计算机终端，可每个办公室的房门都是紧锁的，你就只能在那里按部就班地等着用那几台公共终端。在理查德看来，这实在是暴殄天物。尽管他后来也会偶尔去哈佛大学的计算机实验室，但他显然更喜欢人工智能实验室的平等环境。他说："在那里总算能呼吸到新鲜空气了。人工智能实验室里，人们更关心手头的工作，而不是各自的地位状态。"

Stallman quickly learned that the AI Lab's first-come, first-served policy owed much to the efforts of a vigilant few. Many were holdovers from the days of Project MAC, the Department of Defense-funded research program that had given birth to the first time-share operating systems. A few were already legends in the computing world. There was Richard Greenblatt, the lab's in-house Lisp expert and author of MacHack, the computer chess program that had once humbled AI critic Hubert Dreyfus. There was Gerald Sussman, original author of the robotic block-stacking program HACKER. And there was Bill Gosper, the in-house math whiz already in the midst of an 18-month hacking bender triggered by the philosophical implications of the computer game LIFE.

理查德很快了解到，人工智能实验室这种先到先用的原则，很大一部分要归功于当年一批警觉的小团体。其中很多人，参与了当年国防部资助的MAC项目（Porject MAC）。在这个项目里，他们开发了第一个分时操作系统。而这个小团体中的有些人，如今已经在世界计算机史册上留了名。包括实验室中的Lisp专家理查德·格林布拉特（Richard Greenblatt)，他设计了国际象棋程序MacHack，让休伯特·德莱弗斯（Hubert Dreyfus）关于人工智能的观点受到重创；还有著名的杰拉尔德·萨斯曼（Gerald Sussman），他曾设计HACKER这个程序，利用机器人解决堆垛问题；也有著名的数学怪才比尔·高斯伯（Bill Gosper），他当年发现了生命游戏（LIFE game）中的模式并因此获奖。

Members of the tight-knit group called themselves "hackers." Over time, they extended the "hacker" description to Stallman as well. In the process of doing so, they inculcated Stallman in the ethical traditions of the "hacker ethic." In their fascination with exploring the limits of what they could make a computer do, hackers might sit at a terminal for 36 hours straight if fascinated with a challenge. Most importantly, they demanded access to the computer (when no one else was using it) and the most useful information about it. Hackers spoke openly about changing the world through software, and Stallman learned the instinctual hacker disdain for any obstacle that prevented a hacker from fulfilling this noble cause. Chief among these obstacles were poor software, academic bureaucracy, and selfish behavior.

这批小团体里的人把自己称作"黑客"。当然将来，他们也会把理查德归类为"黑客"。在成为"黑客"的过程中，理查德被逐渐地引进黑客文化的小圈子里。这群人，会痴迷于不断挖掘计算机的极限。他们遇到挑战，可以一连36个小时坐在终端显示器前面工作。最重要的，他们对计算机和各种相关信息都有无尽的需

求。他们会公开大谈如何用计算机和软件改变世界。理查德也逐渐学到，黑客们会本能地蔑视任何阻止他们实现这一伟大梦想的障碍和困难。这些障碍之中，最为重要的有以下几个方面：一是低劣的软件，二是各种学术官僚主义，三是自私自利的垄断的行为。

Stallman also learned the lore, stories of how hackers, when presented with an obstacle, had circumvented it in creative ways. This included various ways that hackers had opened professors' offices to "liberate" sequestered terminals. Unlike their pampered Harvard counterparts, MIT faculty members knew better than to treat the AI Lab's limited stock of terminals as private property. If a faculty member made the mistake of locking away a terminal for the night, hackers were quick to make the terminal accessible again – and to remonstrate with the professor for having mistreated the community. Some hackers did this by picking locks ("lock hacking"), some by removing ceiling tiles and climbing over the wall. On the 9th floor, with its false floor for the computers' cables, some spelunked under it. "I was actually shown a cart with a heavy cylinder of metal on it that had been used to break down the door of one professor's office," Stallman says.

理查德在这里学到的东西里，还包括黑客们解决问题的各种新颖技巧和相关的轶事。这其中，就包括黑客们如何打开教授们紧锁的办公室门，"解放"被囚禁的计算机终端。哈佛大学的计算机终端每天都被娇生惯养；相比之下，麻省理工学院的教授们更了解如何应付有限的终端资源。要是哪位教授，把某个终端锁在自己的办公室里过夜，黑客们就会想法把终端重新搞到手。之后，还会大摇大摆走到教授面前，劝诫他不要如此对待实验室的同僚。黑客们的做法多种多样，有些人会直接撬锁，他们自称这是"黑"锁术；有些人则会把天花板打开，爬到天花板上面，从天花板和房顶中间的通风空隙中爬入教授的办公室；在大厦的第九层，地板和楼层地面中间有空隙，本来是用于铺设机房的电线，而黑客们则会撬开地板，从下面溜进办公室。理查德说："我当初还给他们展示过一架手推车，上面放着一个实心的圆柱形金属，这东西后来就曾用来撞开办公室的门。"

The hackers' insistence served a useful purpose by preventing the professors from egotistically obstructing the lab's work. The hackers did not disregard people's particular needs, but insisted that these be met in ways that didn't obstruct everyone else. For instance, professors occasionally said they had something in their offices which had to be protected from theft. The hackers responded, "No one will object if you lock your office, although that's not very

friendly, as long as you don't lock away the lab's terminal in it."

黑客们如此坚持，使得教授们很难起私心而阻碍整个实验室的工作。当然黑客们也没有无视教授们的个人需求，只是教授的需求不能影响到别人。比如，有些教授有时候会辩解称，自己办公室里有重要的私人物品，所以必须要锁门。黑客们则会回应："你锁上办公室的门没问题，但是别把实验室的终端也锁在里面。"

Although the academic people greatly outnumbered the hackers in the AI Lab, the hacker ethic prevailed. The hackers were the lab staff and students who had designed and built parts of the computers, and written nearly all the software that users used. They kept everything working, too. Their work was essential, and they refused to be downtrodden. They worked on personal pet projects as well as features users had asked for, but sometimes the pet projects revolved around improving the machines and software even further. Like teenage hot-rodders, most hackers viewed tinkering with machines as its own form of entertainment.

虽然在人数上，黑客们处于劣势，但黑客的文化在不断追求、不断进步的计算机领域还是占了上风。这些黑客，很多是实验室的成员，或者是学生。实验室的计算机上，有些硬件就是他们设计的；而几乎所有软件，都是这些黑客写的。他们不仅创造了整个系统，还维护着这个系统的运行。他们的工作至关重要，因此他们拒绝来自任何方面的压抑。他们会根据其他用户的需求写程序，也会同时维护着自己的个人项目。有时候，这些个人项目会越来越庞大，甚至影响深远。黑客们每天为计算机系统增砖添瓦，并以此为乐。

Nowhere was this tinkering impulse better reflected than in the operating system that powered the lab's central PDP-10 computer. Dubbed ITS, short for the Incompatible Time Sharing system, the operating system incorporated the hacking ethic into its very design. Hackers had built it as a protest to Project MAC's original operating system, the Compatible Time Sharing System, CTSS, and named it accordingly. At the time, hackers felt the CTSS design too restrictive, limiting programmers' power to modify and improve the program's own internal architecture if needed. According to one legend passed down by hackers, the decision to build ITS had political overtones as well. Unlike CTSS, which had been designed for the IBM 7094, ITS was built specifically for the PDP-6. In letting hackers write the system themselves, AI Lab administrators guaranteed that only hackers would feel comfortable using the

PDP-6. In the feudal world of academic research, the gambit worked. Although the PDP-6 was co-owned in conjunction with other departments, AI researchers soon had it to themselves. Using ITS and the PDP-6 as a foundation, the Lab had been able to declare independence from Project MAC shortly before Stallman's arrival.

这种工作的结果之一，就是运行在实验室核心的PDP-10计算机上的操作系统。它被称做ITS，全称"非兼容型分时系统"（Incompatible Time Sharing system）。这个系统从头到脚都体现着黑客的文化。它的诞生，本来是用来抗议MAC项目的操作系统：CTSS，即"兼容型分时系统"（Compatible Time Sharing System）。这种抗议从名字中的"非兼容型"几个字上就能看出来。当时，黑客们觉得CTSS的很多设计太过严苛，因为它限制了程序员改进现有程序的行为。沿袭着黑客们一贯以来的作风，设计ITS的决定本身就是政治上正确的。CTSS本身是为IBM 7094设计的，ITS则是为PDP-6设计。为了让黑客们更好地写出自己的操作系统，人工智能实验室的管理员保证可以让黑客们使用PDP-6计算机。在分封制度盛行的学术圈，这策略确实奏效。虽然PDP-6以前是和其他几个系共用的，可后来最终还是归了人工智能的研究人员。理查德来之前的那段时间，黑客们终于利用ITS和这台PDP-6做基础，正式宣告自己独立于MAC项目。

By 1971, ITS had moved to the newer but compatible PDP-10, leaving the PDP-6 for special stand-alone uses. The AI PDP-10 had a very large memory for 1971, equivalent to a little over a megabyte; in the late 70s it was doubled. Project MAC had bought two other PDP-10s; all were located on the 9th floor, and they all ran ITS. The hardware-inclined hackers designed and built a major hardware addition for these PDP-10s, implementing paged virtual memory, a feature lacking in the standard PDP-10.

1971年，ITS被安装在更新一代的PDP-10上。那台老的PDP-6则被单独拿出来以做他用。人工智能实验室这台PDP-10有大约1MB的内存，在当时算是非常大的了。到七十年代末，又给它的内存加了一倍。MAC项目也为实验室带来了两台PDP-10，它们都被放在大厦九楼，运行着ITS系统。PDP-10本身没有内存分页机制。为此，对硬件感兴趣的黑客们为这三台PDP-10设计了内存分页模块。

As an apprentice hacker, Stallman quickly became enamored with ITS. Although forbidding to some non-hackers, ITS boasted features most commercial operating systems wouldn't offer for years (or even to this day), features such as multitasking, applying the debugger immediately to any

running program, and full-screen editing capability.

作为一名黑客学徒，理查德一下子被ITS迷住了。尽管ITS会吓跑一些门外汉，可它却比其他商业操作系统提早几年实现了很多特性，甚至有些特性是今天很多商业操作系统都没有提供的。这包括多任务系统，在线调试任何运行中的程序，全屏编辑等等。

“ITS had a very elegant internal mechanism for one program to examine another,” says Stallman, recalling the program. “You could examine all sorts of status about another program in a very clean, well-specified way.” This was convenient not only for debugging, but also for programs to start, stop or control other programs.

理查德回忆：“ITS内部有一套非常优雅的机制，可以让一个程序实时监测另一个运行中的程序。你可以获得另外某个程序的各种信息，内容清晰，接口明确。”这个功能不仅对调试程序有很大帮助，而且可以轻松地开关和控制别的程序。

Another favorite feature would allow the one program to freeze another program’s job cleanly, between instructions. In other operating systems, comparable operations might stop the program in the middle of a system call, with internal status that the user could not see and that had no well-defined meaning. In ITS, this feature made sure that monitoring the step-by-step operation of a program was reliable and consistent.

另外一个功能，就是允许一个程序暂停另一个程序，让那个程序停在指定的两个指令之间。在别的操作系统里，类似的功能只能让程序停留在某个系统调用中，把程序停在这个状态，会有很多内部隐藏状态用户都看不到。而在ITS里，这一功能则可以保证一条指令一条指令地监视另一个程序运行，并且能做到停顿之后，一样能继续运行。

“If you said, ‘Stop the job,’ it would always be stopped in user mode. It would be stopped between two user-mode instructions, and everything about the job would be consistent for that point,” Stallman says. “If you said, ‘Resume the job,’ it would continue properly. Not only that, but if you were to change the (explicitly visible) status of the job and continue it, and later change it back, everything would be consistent. There was no hidden status anywhere.”

理查德说：“如果你说：‘把这个任务暂停’，那它肯定会停在用户态。而且会停在

两条用户态的指令中间。任何运行过程中的变量状态，都会保留。你再说：‘继续跑它’，它就真的会继续跑下去，不会因为暂停而搞乱程序，导致前后不一致。还不止如此，你还能把某个变量或者哪个状态给修改了。让它继续跑，然后再把它改回去，这程序就能和当初一样。所有状态都是用户可见的，没有什么隐藏状态。”

Starting in September 1971, hacking at the AI Lab had become a regular part of Stallman’s weekly school schedule. From Sunday through Friday, Stallman was at Harvard. As soon as Friday afternoon arrived, however, he was on the subway, heading down to MIT for the weekend. Stallman usually made sure to arrive well before the ritual food run. Joining five or six other hackers in their nightly quest for Chinese food, he would jump inside a beat-up car and head across the Harvard Bridge into nearby Boston. For the next hour or so, he and his hacker colleagues would discuss everything from ITS to the internal logic of the Chinese language and pictograph system. Following dinner, the group would return to MIT and hack code until dawn, or perhaps go to Chinatown again at 3 a.m.

从1971年开始，理查德每周都会定期跑去人工智能实验室。周日到周五，理查德会呆在哈佛大学。一到了周五下午，他就坐上地铁，直奔麻省理工学院，在那里度过整个周末。他一般会在晚饭时间赶到，然后约上五六个黑客，一起去吃中餐。他们跳进一辆旧车上，穿过哈佛大桥，直奔波士顿。接下来的四个小时里，他们凑在一起就开始谈天说地。话题从ITS系统，一直到汉语和象形文字的内在逻辑。晚饭结束，这群人再回到麻省理工，写代码一直到天亮。期间也许会在半夜三点多的时候，再跑去唐人街一趟。

Stallman might stay up all morning hacking, or might sleep Saturday morning on a couch. On waking he would hack some more, have another Chinese dinner, then go back to Harvard. Sometimes he would stay through Sunday as well. These Chinese dinners were not only delicious; they also provided sustenance lacking in the Harvard dining halls, where on the average only one meal a day included anything he could stomach. (Breakfast did not enter the count, since he didn’t like most breakfast foods and was normally asleep at that hour.)

第二天上午，理查德也许会继续折腾代码，也有可能找个沙发睡上一觉。醒来之后，他会继续写代码，找点中餐吃吃，然后回到哈佛。有时候，他会再多呆一天，等到周日再回去。这些中餐不仅美味，而且比哈佛大学食堂的菜内容丰

富，营养全面。吃上一顿能管一天。所以理查德他们一般一天也就吃一顿饭。（对于他来说，早餐一般不在考虑范围之内。一方面，早餐中的食物他不喜欢；另一方面，早餐时间他一般都会睡大觉。）

For the geeky outcast who rarely associated with his high-school peers, it was a heady experience to be hanging out with people who shared the same predilection for computers, science fiction, and Chinese food. “I remember many sunrises seen from a car coming back from Chinatown,” Stallman would recall nostalgically, 15 years after the fact in a speech at the Swedish Royal Technical Institute. “It was actually a very beautiful thing to see a sunrise, ’cause that’s such a calm time of day. It’s a wonderful time of day to get ready to go to bed. It’s so nice to walk home with the light just brightening and the birds starting to chirp; you can get a real feeling of gentle satisfaction, of tranquility about the work that you have done that night.”

作为一个中学时期很少有社交活动的宅男，如今和这么多跟自己类似的人混在一起，确实让理查德甚是陶醉。他们这些人，都对计算机感兴趣，都喜欢科幻小说，喜欢吃中餐。“我记得很多次，我们从唐人街回来，在车上看日出。”十五年后，在瑞典皇家技术研究所的一次演讲上，理查德回想当年，一切依旧历历在目：“欣赏日出真的是个很享受的事情。那是一天之中最平静的时刻。也是一天之中上床睡觉的好时候。倘若这时候徒步走回家，伴着几缕阳光，耳畔回响着鸟鸣，再想想你整整一夜完成的工作，你会有种由内而外地舒适和满足感。”

The more Stallman hung out with the hackers, the more he adopted the hacker world view. Already committed to the notion of personal liberty, Stallman began to infuse his actions with a sense of communal duty. When others violated the communal code, Stallman was quick to speak out. Within a year of his first visit, Stallman was the one opening locked offices to recover the sequestered terminals that belonged to the lab community as a whole. In true hacker fashion, Stallman also sought to make his own personal contribution to the art. One of the most artful door-opening tricks, commonly attributed to Greenblatt, involved bending a stiff wire into several right angles and attaching a strip of tape to one end. Sliding the wire under the door, a hacker could twist and rotate the wire so that the tape touched the inside doorknob. Provided the tape stuck, a hacker could turn the doorknob by pulling the handle formed from the outside end of the wire.

理查德和这群黑客呆得越来越久，他也逐渐开始了解和接纳黑客们的世界观。

曾经他坚持个人自由，如今，在这基础之上，又注入了公共责任的概念。当其他人违反了这种公共准则，理查德会毫不留情地当面指出。在麻省理工学院的第一年，理查德也曾想法打开教授办公室的门，拿回属于实验室的计算机终端。作为一个黑客，理查德也贡献了自己的开门技巧。以前大家一直用着一个开门的方法，据说是格林布拉特（Greenblatt）最先使用的。这个方法要求用到一条硬的电线，把这条电线折几个弯，从一盒磁带里抽出磁条，拴在弯过的电线一头，再把磁条做成一个绳套。开门的时候，先从门下的缝隙里把带磁条一头的电线躺着穿过门缝，等电线和磁条都进了门，再把电线立起来。来回晃几下电线，直到磁条做的绳套能套在门把手上。一旦套紧，黑客就可以把电线往后拉，这样，门就从里面被打开了。

When Stallman tried the trick, he found it hard to execute. Getting the tape to stick wasn't always easy, and twisting the wire in a way that turned the doorknob was similarly difficult. Stallman thought about another method: sliding away ceiling tiles to climb over the wall. This always worked, if there was a desk to jump down onto, but it generally covered the hacker in itchy fiberglass. Was there a way to correct that flaw? Stallman considered an alternative approach. What if, instead of slipping a wire under the door, a hacker slid away two ceiling panels and reached over the wall with a wire?

当理查德自己亲自尝试这个方法的时候，他发现这法子操作起来难度不小。要让磁条套上门把手，可是个体力活。而之前要怎么把电线折出几个弯，也是有学问的。为此，理查德开始另寻他法：他开始想法把天花板上的板子移开，然后爬进天花板和房顶中间的通风空隙中。爬到办公室上方，跳到桌子上，一切就大功告成了。这法子一直都行得通，可这一路，总会沾上一身让人痒痒的玻璃纤维。理查德开始考虑怎么解决这个问题。他想到，如果不把电线从门下缝隙滑过去，而是从天花板上悬下去开门，会不会可以呢?

Stallman took it upon himself to try it out. Instead of using a wire, Stallman draped out a long U-shaped loop of magnetic tape with a short U of adhesive tape attached sticky-side-up at the base. Reaching across over the door jamb, he dangled the tape until it looped under the inside doorknob. Lifting the tape until the adhesive stuck, he then pulled on one end of the tape, thus turning the doorknob. Sure enough, the door opened. Stallman had added a new twist to the art of getting into a locked room.

理查德自己开始尝试。他并没有用电线，而是把一盒磁带里的磁条拿出来，拿着两端，向下悬成一个U形，把磁条有粘性的部分冲上。移开天花板上的两块板

子，站到高处，把U形的磁条从通风道甩到房间里。磁条会从天花板上垂下去，晃悠几下，把U形磁条套在门把手上。接着把磁条往上一拉，门就顺势打开了。就这样，理查德又为黑客开门技巧做了贡献。

“Sometimes you had to kick the door after you turned the door knob,” says Stallman, recalling a slight imperfection of the new method. “It took a little bit of balance to pull it off while standing on a chair on a desk.”

“有时候，门锁打开后你还得踹两脚才能把门弄开。”理查德回忆起这个方法的一点瑕疵：“拉磁条的时候你得小心点，一般是站在一个椅子上，下面还要再垫个桌子。这时候保持平衡就很重要。”

Such activities reflected a growing willingness on Stallman’s part to speak and act out in defense of political beliefs. The AI Lab’s spirit of direct action had proved inspirational enough for Stallman to break out of the timid impotence of his teenage years. Opening up an office to free a terminal wasn’t the same as taking part in a protest march, but it was effective in a way that most protests weren’t: it solved the problem at hand.

这些活动从一个方面反映出理查德期望表达自己的政治诉求。人工智能实验室的实干精神，让理查德打破了中学以来的沉默。打开办公室的门和上街游行虽然不可同日而语，但从效率的角度分析，与游行比起来，它快速有效，能直接解决问题。

By the time of his last years at Harvard, Stallman was beginning to apply the whimsical and irreverent lessons of the AI Lab back at school.

在哈佛大学的最后一年，理查德把人工智能实验室的那套古灵精怪的行事理念用到了哈佛的校园里。

“Did he tell you about the snake?” his mother asks at one point during an interview. “He and his dorm mates put a snake up for student election. Apparently it got a considerable number of votes.”

“他跟你说那条蛇了吗？”他母亲在一次采访中突然问起，“他还有他宿舍那边的几个人一起，把一条蛇拿去参加了学生会选举。而且最后那条蛇还拿到不少选票。”

The snake was a candidate for election within Currier House, Stallman’s dorm, not the campus-wide student council. Stallman does remember the snake

attracting a fair number of votes, thanks in large part to the fact that both the snake and its owner both shared the same last name. “People may have voted for it because they thought they were voting for the owner,” Stallman says. “Campaign posters said that the snake was ‘slithering for’ the office. We also said it was an ‘at large’ candidate, since it had climbed into the wall through the ventilating unit a few weeks before and nobody knew where it was.”

根据理查德的回忆，那条蛇实际是拿去参加了卡瑞尔楼的学生干部竞选。那是理查德住的宿舍楼，并非全校范围内的学生竞选。理查德的确记得当时那条蛇拿到了不少选票，但很大一个原因，是因为这条蛇的姓和它主人一样。“人们也许以为是给这条蛇的主人在投票。”理查德说，“我们的选举海报上写那条蛇是个很有‘潜力’的候选人。因为竞选开始前几周，它刚刚‘潜’到通风口里，我们好几个人极尽全“力”也没能找到它。”

Stallman and friends also “nominated” the house master’s 3-year-old son. “His platform was mandatory retirement at age seven,” Stallman recalls. Such pranks paled in comparison to the fake-candidate pranks on the MIT campus, however. One of the most successful fake-candidate pranks was a cat named Woodstock, which actually managed to outdraw most of the human candidates in a campus-wide election. “They never announced how many votes Woodstock got, and they treated those votes as spoiled ballots,” Stallman recalls. “But the large number of spoiled ballots in that election suggested that Woodstock had actually won. A couple of years later, Woodstock was suspiciously run over by a car. Nobody knows if the driver was working for the MIT administration.” Stallman says he had nothing to do with Woodstock’s candidacy, “but I admired it.”

理查德和几个朋友还提名了楼管的三岁儿子。“给他选举时候提出的政治纲领，就是七岁强制退休。”不过，理查德的这些恶作剧与麻省理工学院里的选举恶作剧比起来，实在是小巫见大巫了。在麻省理工学院最成功的一次选举恶作剧，是提名一只名叫伍德斯托克的猫参加全校学生委员会的选举。最后这只猫得到的选票比许多候选人的选票都多。“他们最后也没有宣布伍德斯托克到底得了多少票。他们把这些选票当作了废票。可选举结果一出，大家看到了一大堆的废票，从废票的数量上看，伍德斯托克应该是可以赢得这次选举了。几年之后，伍德斯托克据说是在马路上被一辆车压死了。”理查德如此回忆道。他说他当年和伍德斯托克的选举没有一点关系，“可我很崇拜这事。”

At the AI Lab, Stallman's political activities had a sharper-edged tone. During the 1970s, hackers faced the constant challenge of faculty members and administrators pulling an end-run around ITS and its hacker-friendly design. ITS allowed anyone to sit down at a console and do anything at all, even order the system to shut down in five minutes. If someone ordered a shutdown with no good reason, some other user canceled it. In the mid-1970s some faculty members (usually those who had formed their attitudes elsewhere) began calling for a file security system to limit access to their data. Other operating systems had such features, so those faculty members had become accustomed to living under security, and to the feeling that it was protecting them from something dangerous. But the AI Lab, through the insistence of Stallman and other hackers, remained a security-free zone.

在人工智能实验室，理查德的政治活动更加积极，产生的矛盾也更尖锐。在七十年代，学校的一些教授和管理员一度发出通知，说即将停止在实验室的计算机上运行ITS，也将停止支持很多黑客们喜欢的设计。ITS允许任何人坐在终端前，做任何事情。包括把整个计算机关机。这在当年，可是个大事。因为很多人都会通过终端使用同一台计算机，关机意味着其他人也必须暂停手上的工作。不过，如果别人关机，你会收到即将关机的通知。要是你手头的工作还没完成，你也可以取消这次关机操作。到了七十年代中期，一些教授开始对文件的安全保护提出要求，要求他们能够控制哪些用户才能访问自己的文件。可人工智能实验室的这些黑客们，还是坚持要维护一个没有控制的系统。

Stallman presented both ethical and practical arguments against adding security. On the ethical side, Stallman appealed to the AI Lab community's traditions of intellectual openness and trust. On the practical side, he pointed to the internal structure of ITS, which was built to foster hacking and cooperation rather than to keep every user under control. Any attempt to reverse that design would require a major overhaul. To make it even more difficult, he used up the last empty field in each file's descriptor for a feature to record which user had most recently changed the file. This feature left no place to store file security information, but it was so useful that nobody could seriously propose to remove it.

于是，理查德对这些安全控制特性从道德和实践上给予了否定。从道德上说，人工智能实验室的这个圈子里，大家都有着开放和信任的传统。从实践上看，ITS系统本身也是为了开发和协作而设计的，并非要控制哪些用户。任何与这初衷相违背的特性都需要额外的精力来开发维护，完全没有必要。为了让安全特

性更难实现，理查德把文件描述符的最后一点富裕空间，用来存储用户的最后修改时间。最后修改时间是个很重要的信息，谁也不想把它去掉。可这么一来，整个文件描述符也就没有任何地方来存储安全信息了。

“The hackers who wrote the Incompatible Timesharing System decided that file protection was usually used by a self-styled system manager to get power over everyone else,” Stallman would later explain. “They didn’t want anyone to be able to get power over them that way, so they didn’t implement that kind of a feature. The result was, that whenever something in the system was broken, you could always fix it” (since access control did not stand in your way).

“设计ITS的黑客们认为，文件访问控制这个特性，是为自私的系统管理员设计的，这样他们这些管理员就可以凌驾在任何用户之上。”理查德事后解释：“黑客们可不希望有谁被任何人控制约束，所以他们不会实现这样的功能。这么做的结果是，一旦系统出了毛病，任何人都能赶在第一时间修复它。”（因为没有访问控制功能去阻止你去访问任何文件。）

Through such vigilance, hackers managed to keep the AI Lab’s machines security-free. In one group at the nearby MIT Laboratory for Computer Sciences, however, security-minded faculty members won the day. The DM group installed its first password system in 1977. Once again, Stallman took it upon himself to correct what he saw as ethical laxity. Gaining access to the software code that controlled the password system, Stallman wrote a program to decrypt the encrypted passwords that the system recorded. Then he started an email campaign, asking users to choose the null string as their passwords. If the user had chosen “starfish,” for example, the email message looked something like this:

这种警觉，让黑客们得以在人工智能实验室营造出一个没有管制的世界。可在旁边的麻省理工学院计算机科学实验室里，对安全敏感的教授们则取得了胜利。动态建模组最先在1977年在系统中使用了密码登陆。又一次，斯托曼挺身而出，纠正了这种自己看来“不道德”的行为。他先获得了密码登录系统代码的访问权限，然后写了一个程序，解密已经加密存储的密码。然后他开始给系统上的每个用户发邮件，劝说他们放弃使用密码。这封邮件大概是这样的：

> I see you chose the password “starfish”. I suggest that you switch to the password “carriage return”, which is what I use. It’s easier to type, and also opposes the idea of passwords and security.

> 我看到你用“starfish”做密码了，我建议你把密码直接改成“回车”，我用的就是这个密码。这密码简单好记，而且可以让你忘了密码和安全这些繁杂琐事。

The users who chose “carriage return” – that is, users who simply pressed the Enter or Return button, entering a blank string instead of a unique password – left their accounts accessible to the world at large, just as all accounts had been, not long before. That was the point: by refusing to lock the shiny new locks on their accounts, they ridiculed the idea of having locks. They knew that the weak security implemented on that machine would not exclude any real intruders, and that this did not matter, because there was no reason to be concerned about intruders, and that no one wanted to intrude anyway, only to visit.

所谓用“回车”做密码，就等同于没有密码。这个密码让其他用户也可以访问自己的帐号。这和以前没有密码的日子没什么分别。这就是理查德和黑客们要表达的：他们拒绝任何光鲜亮美的枷锁，他们会嘲弄任何企图使用枷锁的想法。他们知道，这种安全保护能力很弱的系统，不可能防范任何入侵者。可这无所谓，因为完全没必要考虑什么入侵者，从来就没有入侵者，只有访客。

Stallman, speaking in an interview for the 1984 book *Hackers*, proudly noted that one-fifth of the LCS staff accepted this argument and employed the null-string password.

理查德在一次为1984年出版的《黑客》一书的采访中，很骄傲地提到，当年计算机科学实验室里，有五分之一的人换了“回车”做密码。

Stallman’s null-string campaign, and his resistance to security in general, would ultimately be defeated. By the early 1980s, even the AI Lab’s machines were sporting password security systems. Even so, it represented a major milestone in terms of Stallman’s personal and political maturation. Seen in the context of Stallman’s later career, it represents a significant step in the development of the timid teenager, afraid to speak out even on issues of life-threatening importance, into the adult activist who would soon turn needling and cajoling into a full-time occupation.

理查德的这种反密码抗议，以及各种拒绝安全保护的行为，最终还是要失败的。八十年代起，哪怕是人工智能实验室，也开始支持密码登陆系统。可即便如此，理查德的行为标志着他个人政治思维成熟的里程碑。理查德迈出了一大

步。当年他还是怯懦的中学生时，哪怕遇到决定生死的大事，他也不敢上街公开反对。如今，他则成了一个活跃人士，把各种嘻笑怒骂当作家常便饭。

In voicing his opposition to computer security, Stallman drew on many of the key ideas that had shaped his early life: hunger for knowledge, distaste for authority, and frustration over prejudice and secret rules that rendered some people outcasts. He would also draw on the ethical concepts that would shape his adult life: responsibility to the community, trust, and the hacker spirit of direct action. Expressed in software-computing terms, the null string represents the 1.0 version of the Richard Stallman political worldview – incomplete in a few places but, for the most part, fully mature.

反对计算机安全这事，体现了理查德早年性格中的几个关键特质：对知识如饥似渴，对权威厌恶嘲弄；可又烦恼于别人对自己的各种偏见不解，被一些人看作异类。这些也正反映出了未来主导他行动的道德基础：对社区的责任感和信任；还有遇到问题，直接行动的黑客精神。用软件界的行话说，反密码抗议行动体现的正是理查德1.0版。虽然有些地方还很不完善，但大部分已经成熟。

Looking back, Stallman hesitates to impart too much significance to an event so early in his hacking career. “In that early stage there were a lot of people who shared my feelings,” he says. “The large number of people who adopted the null string as their password was a sign that many people agreed that it was the proper thing to do. I was simply inclined to be an activist about it.”

理查德自己，则不太赞成把他这样的早年的黑客经历强调得太多。他说：“早年间，很多人都和我有同样的想法。当时那么多人最后选择了不用密码，这事实本身就说明他们很多人认为这么做是对的。我只是在这方面比较活跃。”

Stallman does credit the AI Lab for awakening that activist spirit, however. As a teenager, Stallman had observed political events with little idea as to how he could do or say anything of importance. As a young adult, Stallman was speaking out on matters in which he felt supremely confident, matters such as software design, responsibility to the community, and individual freedom. “I joined this community which had a way of life which involved respecting each other’s freedom,” he says. “It didn’t take me long to figure out that that was a good thing. It took me longer to come to the conclusion that this was a moral issue.”

不过理查德依旧很感激人工智能实验室氛围唤醒了他作为政治活动家潜质。曾

经还是少年的时候，理查德目睹了很多政治事件，他很难想象自己能做出什么有影响的事情。如今虽只过弱冠之年，他却已经可以在自己的领域里指手划脚，自信地为整个社区负责，也为个人自由出头。他说：“我加入了这个圈子，这个圈子的文化非常强调尊重每个人的自由。我很快就意识到这是个好事，可花了很长的时间后，我才觉得这是一种道德责任。”

Hacking at the AI Lab wasn’t the only activity helping to boost Stallman’s esteem. At the start of his junior year at Harvard, Stallman began participating in a recreational international folk dance group which had just been started in Currier House. He was not going to try it, considering himself incapable of dancing, but a friend pointed out, “You don’t know you can’t if you haven’t tried.” To his amazement, he was good at it and enjoyed it. What started as an experiment became another passion alongside hacking and studying; also, occasionally, a way to meet women, though it didn’t lead to a date during his college career. While dancing, Stallman no longer felt like the awkward, uncoordinated 10-year-old whose attempts to play football had ended in frustration. He felt confident, agile, and alive. In the early 80s, Stallman went further and joined the MIT Folk Dance Performing Group. Dancing for audiences, dressed in an imitation of the traditional garb of a Balkan peasant, he found being in front of an audience fun, and discovered an aptitude for being on stage which later helped him in public speaking.

理查德不仅在人工智能实验室获得了黑客们的尊重。在哈佛的第一年，理查德参加了一个世界民间舞蹈小组。这个小组刚刚在卡瑞尔宿舍楼成立。理查德一开始也没打算加入，因为他觉得自己根本不适合跳舞。可他朋友却说：“你不试试怎么知道呢？”后来他真的去试了，出乎意料的是，他不仅很擅长跳舞，而且很享受其中的乐趣。这个无心之举最后竟然发展成了他的另一大爱好。这也给了他不少机会和女孩子搭讪，可惜在大学期间，每次搭讪都没能换来单独的约会。跳舞让理查德明白自己不再是那个手脚不协调，不会玩橄榄球的10岁少年了。他由此变得更加自信，灵活，有活力。在八十年代早期，理查德走得更远了：他加入了麻省理工学院民间舞蹈表演队。他穿着巴尔干半岛的传统服饰，在观众面前表演舞蹈。这让他甚是陶醉。舞台上面对观众的经历，也为他日后在各种公共场合中演讲打下了基础。

Although the dancing and hacking did little to improve Stallman’s social standing, they helped him overcome the sense of exclusion that had clouded his pre-Harvard life. In 1977, attending a science-fiction convention for the first time, he came across Nancy the Buttonmaker, who makes calligraphic buttons

saying whatever you wish. Excited, Stallman ordered a button with the words “Impeach God” emblazoned on it.

尽管跳舞和写代码没能提高多少他的社交能力，可刚到哈佛时那种疏远不自然的感觉却从此烟消云散。1977年，在一次科幻小说聚会上，他看到了南希徽章制作机，你可以随便写点格言座右铭什么的，把它制作成徽章。理查德很是兴奋，他当场制作了一个徽章，上面写着“选举上帝”。

For Stallman, the “Impeach God” message worked on many levels. An atheist since early childhood, Stallman first saw it as an attempt to start a “second front” in the ongoing debate on religion. “Back then everybody was arguing about whether a god existed,” Stallman recalls. “‘Impeach God’ approached the subject from a completely different viewpoint. If a god was so powerful as to create the world and yet did nothing to correct the problems in it, why would we ever want to worship such a god? Wouldn’t it be more just to put it on trial?”

在理查德看来，所谓“选举上帝”可以从很多层面上来解释。作为一个从小就坚持无神论的人，理查德把“选举上帝”看作是信仰争论的另一种角度。“那时候，很多人都在讨论上帝究竟是不是存在。‘选举上帝’四个字给了人们一个全新的角度。如果说，咱们这位上帝能耐这么大，大到可以创造整个世界，可却把这个世界中的问题扔在那里，不闻不问，那我们何必要膜拜这么一个上帝呢？这样的上帝难道不该放到法庭上审讯一番吗？”

At the same time, “Impeach God” was a reference to the the Watergate scandal of the 1970s, in effect comparing a tyrannical deity to Nixon. Watergate affected Stallman deeply. As a child, Stallman had grown up resenting authority. Now, as an adult, his mistrust had been solidified by the culture of the AI Lab hacker community. To the hackers, Watergate was merely a Shakespearean rendition of the daily power struggles that made life such a hassle for those without privilege. It was an outsized parable for what happened when people traded liberty and openness for security and convenience.

另一方面，“选举上帝”也影射了七十年代的水门事件。它把尼克松比作暴虐的天神。水门事件对理查德的影响很深。理查德从小就厌恶权威。如今长大成人，这份厌恶在人工智能实验室的黑客精神影响下，更加牢固。黑客们看来，水门事件好似莎士比亚剧一般，诠释着上层权力斗争如何戏虐无权的人。它恰似一幅缩影，展现着人们如何出于安全和方便的考虑，放弃自由和开放。

Buoyed by growing confidence, Stallman wore the button proudly. People curious enough to ask him about it received a well-prepared spiel. “My name is Jehovah,” Stallman would say. “I have a secret plan to end injustice and suffering, but due to heavenly security reasons I can’t tell you the workings of my plan. I see the big picture and you don’t, and you know I’m good because I told you so. So put your faith in me and obey me without question. If you don’t obey, that means you’re evil, so I’ll put you on my enemies list and throw you in a pit where the Infernal Revenue Service will audit your taxes every year for all eternity.”

戴上徽章，理查德自信满满。好奇的路人倘若问上一句，必然换来一篇准备已久的长篇大论：“我乃耶和华，从天而来，替天行道。除忧治病，铲逆为公。我眼中所见之事，并非肉眼凡胎所能看；我心中所思之物，也非莽夫俗子所能解。神谕无价，安全第一，行道之法，乃是天机，万万不可泄露。你等世人，尽管加信于我，毋疑毋问，顺天听命。旦有不从，即为妖孽，从此名姓入簿，打入非常之所，不得入三界，列五行。妖孽之地，行严法，加重税，世代如此，不得翻案。”

Those who interpreted the spiel as a parody of the Watergate hearings only got half the message. For Stallman, the other half of the message was something only his fellow hackers seemed to be hearing. One hundred years after Lord Acton warned about absolute power corrupting absolutely, Americans seemed to have forgotten the first part of Acton’s truism: power, itself, corrupts. Rather than point out the numerous examples of petty corruption, Stallman felt content voicing his outrage toward an entire system that trusted power in the first place.

如果只是觉得这段话是对水门事件的恶搞，那只听懂了一半。另外一半，恐怕只有那群黑客的同类能听懂。百年之前，阿克顿男爵（Lord Acton）曾经发出警告：绝对权力导致绝对的腐败。而很多美国人，恐怕忘了这个警告的前半部分：权利本身导致腐败。理查德并没有直接列举各种腐败案例，他直指过分信任权利，从而酿成腐败的整个社会。

“I figured, why stop with the small fry,” says Stallman, recalling the button and its message. “If we went after Nixon, why not go after Mr. Big? The way I see it, any being that has power and abuses it deserves to have that power taken away.”

理查德回忆起徽章的事情，说：“我觉得，为什么把讨论就停留在这里了？如果

仅仅看到水门这一桩事件，你也许只是不再信任尼克松和他的幕僚。可明天你也许就又开始信任尼克柏，尼克树，你依旧信任权力。我对待这些掌权者都是一个态度：倘若他滥用权力，就活该在哪天被剥夺权力。”

Puddle of Freedom 自由一隅

Ask anyone who’s spent more than a minute in Richard Stallman’s presence, and you’ll get the same recollection: forget the long hair. Forget the quirky demeanor. The first thing you notice is the gaze. One look into Stallman’s green eyes, and you know you’re in the presence of a true believer.

如果谁和理查德在一起，呆上超过一分钟，他的感受一般都是这样：不管他那一头的长发，不管他怪异的举止，给人印象最深刻的，是理查德凝视你的眼神。透过他绿色的眼睛，你能看到一种真实的信仰。

To call the Stallman gaze intense is an understatement. Stallman’s eyes don’t just look at you; they look through you. Even when your own eyes momentarily shift away out of simple primate politeness, Stallman’s eyes remain locked-in, sizzling away at the side of your head like twin photon beams.

要说理查德眼神犀利，都要算是轻描淡写了。他不止是看着你，他简直就是要看透看穿你的心。哪怕你出于礼貌，暂时把自己的视线从他身上移开，你依旧能感受到他的眼神像灯塔一样，锁定在你的身上。

Maybe that’s why most writers, when describing Stallman, tend to go for the religious angle. In a 1998 *Salon.com* article titled “The Saint of Free Software,” Andrew Leonard describes Stallman’s green eyes as “radiating the power of an Old Testament prophet.” A 1999 *Wired* magazine article describes the Stallman beard as “Rasputin-like,” while a *London Guardian* profile describes the Stallman smile as the smile of “a disciple seeing Jesus.”

也许这就是为什么，很多描写理查德的文章，都会从信仰的角度开始写起。在1998年salon.com上的一篇文章，标题定为《自由软件圣徒》。作者安德鲁·伦纳德描述理查德那双绿色眼睛的时候，写道：“[它们] 散发出一种旧约中先知般的力量。”1999年《*连线*》杂志的一篇文章中，形容理查德的络腮长胡须好似拉斯普京的胡须一般。《伦敦卫报》则把理查德的微笑说成好似“门徒看到耶稣”一般。 .....

Such analogies serve a purpose, but they ultimately fall short. That's because they fail to take into account the vulnerable side of the Stallman persona. Watch the Stallman gaze for an extended period of time, and you will begin to notice a subtle change. What appears at first to be an attempt to intimidate or hypnotize reveals itself upon second and third viewing as a frustrated attempt to build and maintain contact. If his personality has a touch or "shadow" of autism or Asperger's Syndrome, a possibility that Stallman has entertained from time to time, his eyes certainly confirm the diagnosis. Even at their most high-beam level of intensity, they have a tendency to grow cloudy and distant, like the eyes of a wounded animal preparing to give up the ghost.

这些比喻有一定道理，但似乎也有不足。它们都忽视了理查德人格中脆弱的一面。盯着理查德的眼睛再久些，你会发现其中潜在的改变。一开始，是一种强势带着几分催眠般的眼神；可再看几次，就会发现那眼神其实是在寻求沟通，建立联系。如果他个性中真的有着一些自闭症的成分，那么他的这种眼神倒确实符合这种个性。哪怕当他目光如炬地盯着你的时候，你依旧能够感受到他眼神中透出的几分迷茫和疏远。恰似受伤的猛兽一般，眼中传出的有几分绝望。

My own first encounter with the legendary Stallman gaze dates back to the March, 1999, LinuxWorld Convention and Expo in San Jose, California. Billed as a "coming out party" for the "Linux" software community, the convention also stands out as the event that reintroduced Stallman to the technology media. Determined to push for his proper share of credit, Stallman used the event to instruct spectators and reporters alike on the history of the GNU Project and the project's overt political objectives.

我第一次见识传说中理查德的目光，是在1999年3月的第一届LinuxWorld大会上。大会在加利福尼亚州的圣何塞市举行，是一场"Linux"社区的大聚会。其中一个环节，就是在媒体面前介绍理查德。理查德决心在这次见面上简要回顾GNU工程的历史以及这个工程政治上的目标，以此说明自己在社区中的工作和价值。

As a reporter sent to cover the event, I received my own Stallman tutorial during a press conference announcing the release of GNOME 1.0, a free software graphic user interface. Unwittingly, I push an entire bank of hot buttons when I throw out my very first question to Stallman himself: "Do you think GNOME's maturity will affect the commercial popularity of the Linux operating system?"

作为一个报道此次大会的记者，我在GNOME 1.0的媒体发布会上，被理查德亲自上了一课。当时，我的第一个问题在无意之中触碰到了敏感话题：“请问您觉得随着GNOME逐渐成熟，是否会推进Linux的商业应用？”

“I ask that you please stop calling the operating system Linux,” Stallman responds, eyes immediately zeroing in on mine. “The Linux kernel is just a small part of the operating system. Many of the software programs that make up the operating system you call Linux were not developed by Linus Torvalds at all. They were created by GNU Project volunteers, putting in their own personal time so that users might have a free operating system like the one we have today. To not acknowledge the contribution of those programmers is both impolite and a misrepresentation of history. That’s why I ask that when you refer to the operating system, please call it by its proper name, GNU/Linux.”

“请你不要再把整个操作系统都叫做Linux。”理查德回应，眼光迅速聚焦到我的身上：“Linux是一个操作系统的内核，它仅仅是操作系统的一小部分。你所说的Linux，实际还包含了很多软件。这些软件并不是莱纳斯·托瓦尔兹（Linus Torvalds）开发的。它们是由GNU工程的志愿者开发的，这些志愿者用自己的业余时间创造了这些软件，让用户最终可以自由使用的一个操作系统。如果不提及这些开发人员的工作，那是既不礼貌，也不尊重历史的。所以我坚持把这个操作系统称作GNU/Linux，也希望请你这么称呼它。”

Taking the words down in my reporter’s notebook, I notice an eerie silence in the crowded room. When I finally look up, I find Stallman’s unblinking eyes waiting for me. Timidly, a second reporter throws out a question, making sure to use the term “GNU/Linux” instead of Linux. Miguel de Icaza, leader of the GNOME project, fields the question. It isn’t until halfway through de Icaza’s answer, however, that Stallman’s eyes finally unlock from mine. As soon as they do, a mild shiver rolls down my back. When Stallman starts lecturing another reporter over a perceived error in diction, I feel a guilty tinge of relief. At least he isn’t looking at me, I tell myself.

把这段话记在采访本上时，我感到整个会场异常地安静。等到我抬起头来，才看到理查德的眼睛眨都不眨地盯着我，让我心生一丝胆怯。这时候，另外一名记者提问，正确地使用了GNU/Linux的名字，才打破僵局。GNOME项目的组长，米格尔·德·伊卡萨回答了这名记者的问题。米格尔回答到一半，理查德才把目光从我身上移开。这时我才觉得些许放松。等到理查德继续纠正了另外一个

记者的措辞问题后，我才有种罪孽被洗清的感觉。他起码不再盯着我了——我对自己说。

For Stallman, such face-to-face moments would serve their purpose. By the end of the first LinuxWorld show, most reporters know better than to use the term “Linux” in his presence, and Wired.com is running a story comparing Stallman to a pre-Stalinist revolutionary erased from the history books by hackers and entrepreneurs eager to downplay the GNU Project’s overly political objectives. Other articles follow, and while few reporters call the operating system GNU/Linux in print, most are quick to credit Stallman for launching the drive to build a free software operating system 15 years before.

从理查德的角度看，这种面对面的时刻是有意义的。这次大会结束的时候，各路记者在理查德面前已经改用GNU/Linux这名字。而《连线》的记者则开始忙着写一篇通讯，其中把理查德比作斯大林之前的革命者，被黑客和各大公司从历史上抹去了姓名，他们还闭口不谈GNU工程，因为不喜欢GNU工程中的政治目标。其他的报道也相继出现，尽管没有在报道中使用GNU/Linux的名字，但它们大多都提及了理查德的贡献，说到他十五年前发起的开发自由操作系统的项目。

I won’t meet Stallman again for another 17 months. During the interim, Stallman will revisit Silicon Valley once more for the August, 1999 LinuxWorld show. Although not invited to speak, Stallman does manage to deliver the event’s best line. Accepting the show’s Linus Torvalds Award for Community Service – an award named after Linux creator Linus Torvalds – on behalf of the Free Software Foundation, Stallman wisecracks, “Giving the Linus Torvalds Award to the Free Software Foundation is a bit like giving the Han Solo Award to the Rebel Alliance.”

再次见到理查德，是17个月后。在这17个月中间，理查德又一次前往位于硅谷的圣何塞，参加1999年8月的第二届LinuxWorld大会。尽管没有演讲，但理查德此次还是有目的的。他是代表自由软件基金会，领取莱纳斯·托瓦尔兹社区贡献奖。理查德接受奖杯的时候，打趣道：“这次代表自由软件基金会领取莱纳斯·托瓦尔兹社区贡献奖，感觉就像唐僧代表师徒四人去领取悟空奖。”

This time around, however, the comments fail to make much of a media dent. Midway through the week, Red Hat, Inc., a prominent GNU/Linux vendor, goes public. The news merely confirms what many reporters such as myself already suspect: “Linux” has become a Wall Street buzzword, much like “e-commerce”

and “dot-com” before it. With the stock market approaching the Y2K rollover like a hyperbola approaching its vertical asymptote, all talk of free software or open source as a political phenomenon falls by the wayside.

可惜，这次的玩笑并没有吸引多少媒体的关注。倒是这周的红帽公司（Red Hat, Inc.）上市，引起了各路记者的关注。这是一家以发布GNU/Linux发行版为主的软件公司。这次上市，印证了我们大多数记者的猜测：“Linux”一词，会和“电子商务”，“点com”等一样，成为华尔街上的流行语。股票市场的热闹会把大家搞得像中了千年虫病毒一般，忘掉过去，丢掉历史，把自由软件或开源的政治方面抛诸脑后。

Maybe that’s why, when LinuxWorld follows up its first two shows with a third LinuxWorld show in August, 2000, Stallman is conspicuously absent.

也许这就是为什么，在2000年第三届LinuxWorld大会上，大家再也没看到理查德的身影。

My second encounter with Stallman and his trademark gaze comes shortly after that third LinuxWorld show. Hearing that Stallman is going to be in Silicon Valley, I set up a lunch interview in Palo Alto, California. The meeting place seems ironic, not only because of his absence from the show but also because of the overall backdrop. Outside of Redmond, Washington, few cities offer a more direct testament to the economic value of proprietary software. Curious to see how Stallman, a man who has spent the better part of his life railing against our culture’s predilection toward greed and selfishness, is coping in a city where even garage-sized bungalows run in the half-million-dollar price range, I make the drive down from Oakland.

我第二次再和理查德见面，是在第三届LinuxWorld大会之后不久。当时我听说理查德又要来硅谷，我就和他联系好，定在加利福尼亚州帕罗奥多市（Palo Alto）进行采访。这个地点选的很有意思。不仅仅是因为他上次缺席LinuxWorld大会，还因为这里作为硅谷重镇，是除了微软所在的华盛顿州雷蒙德市以外，最支持专有软件经济模型的几个城市之一。而理查德则花费了自己整个青春和大半生的时间，去与我们文化之中的自私与贪婪作斗争。我很好奇，他来到这座城市，看着车库大小的平房里，都在做着几十万美元的买卖的现实，理查德会作何反应。带着好奇心，我离开奥克兰市，驱车前往帕罗奥多。

I follow the directions Stallman has given me, until I reach the headquarters of Art.net, a nonprofit “virtual artists collective.” Located in a hedge-shrouded

house in the northern corner of the city, the Art.net headquarters are refreshingly run-down. Suddenly, the idea of Stallman lurking in the heart of Silicon Valley doesn't seem so strange after all.

按照理查德给我的地址，我来到了Art.net的总部。这是一家汇集各路“网上艺术家”的非盈利组织。位于城市北部，坐落在一所被篱笆围起来的房子里。房子有些旧，但又透着些小清新。这氛围顿时又让我觉得，理查德混在硅谷，似乎也不是个很怪的想法。

I find Stallman sitting in a darkened room, tapping away on his gray laptop computer. He looks up as soon as I enter the room, giving me a full blast of his 200-watt gaze. When he offers a soothing “Hello,” I offer a return greeting. Before the words come out, however, his eyes have already shifted back to the laptop screen.

我看到理查德坐在一个背阴的房间里，正在他那台灰色笔记本电脑上敲着键盘。我一进门，他就看着我，投来了那著名的目光。互相寒暄，他又把目光聚焦回笔记本屏幕上。

“I'm just finishing an article on the spirit of hacking,” Stallman says, fingers still tapping. “Take a look.”

“我刚刚写了一篇关于黑客精神的文章。”理查德一边打字，一边说：“过来看看？”

I take a look. The room is dimly lit, and the text appears as greenish-white letters on a black background, a reversal of the color scheme used by most desktop word-processing programs, so it takes my eyes a moment to adjust. When they do, I find myself reading Stallman's account of a recent meal at a Korean restaurant. Before the meal, Stallman makes an interesting discovery: the person setting the table has left six chopsticks instead of the usual two in front of Stallman's place setting. Where most restaurant goers would have ignored the redundant pairs, Stallman takes it as challenge: find a way to use all six chopsticks at once. Like many software hacks, the successful solution is both clever and silly at the same time. Hence Stallman's decision to use it as an illustration.

我走过去开始读文章。房间阴暗，屏幕上的编辑软件主题，又被理查德设置成了黑底绿字。我读了几个字，眼睛才逐渐适应。文章开始回忆了理查德有次去

一家韩国餐馆的经历。当时服务员摆桌的时候，在理查德面前放了三双筷子。一般人恐怕都会把多余的两双筷子拿开。可理查德却又借此开始玩起把戏，他企图找个法子，用上所有的筷子。和很多黑客技巧一样，最终的解决方案既聪明灵巧，可又傻里傻气。理查德以这个事情来作为整个文章的引子。

As I read the story, I feel Stallman watching me intently. I look over to notice a proud but child-like half smile on his face. When I praise the essay, my comment barely merits a raised eyebrow.

我读的时候，感觉理查德一直在盯着我。起身看了他一眼，见他一脸骄傲的表情，好似孩子般笑着站在那。我夸奖了一下他的文章，勉强让他眉毛抬了抬。

“I’ll be ready to go in a moment,” he says.

“稍等一下，我们马上就出发。”他说。

Stallman goes back to tapping away at his laptop. The laptop is gray and boxy, not like the sleek, modern laptops that seemed to be a programmer favorite at the recent LinuxWorld show. Above the keyboard rides a smaller, lighter keyboard, a testament to Stallman’s aging hands. During the mid 1990s, the pain in Stallman’s hands became so unbearable that he had to hire a typist. Today, Stallman relies on a keyboard whose keys require less pressure than a typical computer keyboard.

理查德坐回座位，开始继续敲键盘。他用的笔记本是个灰色的四方盒子，和LinuxWorld上看到的大多笔记本不同，理查德的这台没那么光鲜亮丽，也不是什么新款式。笔记本的键盘之上，还放着一个更小，更轻的键盘。理查德的一双大手，则在这个小键盘上飞舞。九十年代中期，理查德双手的剧痛让他无法容忍，以至于他曾一度雇用了一位打字员。今天，他用的键盘，是一种特殊的，比常规键盘按键力度更小的键盘。

Stallman has a tendency to block out all external stimuli while working. Watching his eyes lock onto the screen and his fingers dance, one quickly gets the sense of two old friends locked in deep conversation.

理查德在工作时，会全然不顾外界刺激。看着他眼睛盯着屏幕，手指飞舞，让人觉得理查德和电脑，恰似老友重逢，当下正促膝长谈。

The session ends with a few loud keystrokes and the slow disassembly of the laptop.

理查德用力敲了几下键盘，合上笔记本电脑，拔掉电源，结束了写作。

“Ready for lunch?” Stallman asks.

“吃午饭去吧？”理查德问。

We walk to my car. Pleading a sore ankle, Stallman limps along slowly. Stallman blames the injury on a tendon in his left foot. The injury is three years old and has gotten so bad that Stallman, a huge fan of folk dancing, has been forced to give up all dancing activities. “I love folk dancing intensely,” Stallman laments. “Not being able to dance has been a tragedy for me.”

理查德抱怨着自己脚踝的伤，和我一起徐步走到我的车前。他左脚跟腱三年前受了伤，让他这么一个民间舞爱好者从此告别任何舞蹈活动。理查德叹道：“我可喜欢跳舞了，不能跳舞对我来说简直是个悲剧。”

Stallman’s body bears witness to the tragedy. Lack of exercise has left Stallman with swollen cheeks and a pot belly that was much less visible the year before. You can tell the weight gain has been dramatic, because when Stallman walks, he arches his back like a pregnant woman trying to accommodate an unfamiliar load.

这悲剧从理查德的身材上就体现出来了。由于缺乏锻炼，理查德开始发福。脸颊越来越圆，啤酒肚也凸出来了。你能明显看出，他发胖简直就是一瞬间完成的。他自己都还来不及适应，以至于在走路的时候，都要撑着腰，好似一个孕妇一般。

The walk is further slowed by Stallman’s willingness to stop and smell the roses, literally. Spotting a particularly beautiful blossom, he strokes the innermost petals against his nose, takes a deep sniff, and steps back with a contented sigh.

走到中途，理查德停下脚步，在一簇玫瑰花前伏下身子。剥开花瓣，理查德把鼻子凑到花蕊前，深吸一口气，然后起身长叹，甚是满足。

“Mmm, rhinophytophilia,” he says, rubbing his back.

“嗯，rhinophytophilia。”他揉揉腰说。

The drive to the restaurant takes less than three minutes. Upon recommendation from Tim Ney, former executive director of the Free Software

Foundation, I have let Stallman choose the restaurant. While some reporters zero in on Stallman's monk-like lifestyle, the truth is, Stallman is a committed epicure when it comes to food. One of the fringe benefits of being a traveling missionary for the free software cause is the ability to sample delicious food from around the world. "Visit almost any major city in the world, and chances are Richard knows the best restaurant in town," says Ney. "Richard also takes great pride in knowing what's on the menu and ordering for the entire the table." (If they are willing, that is.)

开车到餐馆不到三分钟。根据前自由软件基金会执行总监蒂姆·内伊（Tim Ney）的建议，我让理查德选择去哪家餐馆。尽管很多报道，把理查德描述成苦行僧一般，可事实上，理查德在饮食方面，可谓是个美食家。作为一个布道全球的自由软件使者，一个额外收获就是可以尝遍各地美食。内伊介绍："他去了几乎全世界所有的主要城市，这让他总能知道全城最好的餐馆在哪。理查德知道菜单上每个菜都是什么，点上一桌美食，他经常以此为荣。"

For today's meal, Stallman has chosen a Cantonese-style dim sum restaurant two blocks off University Avenue, Palo Alto's main drag. The choice is partially inspired by Stallman's recent visit to China, including a stop in Hong Kong, in addition to Stallman's personal aversion to spicier Hunanese and Szechuan cuisine. "I'm not a big fan of spicy," Stallman admits.

今天，理查德选择了一家广式点心店。这家店和帕罗奥多市的主干道学院街（University Ave.）相隔两个街区。选这里，一部分是因为理查德刚刚去过中国，期间在香港做了短暂停留。理查德本人并不喜欢太辣的东西，所以他并不喜欢川菜和湘菜。"我对辣的东西不感冒。"理查德坦言。

We arrive a few minutes after 11 a.m. and find ourselves already subject to a 20-minute wait. Given the hacker aversion to lost time, I hold my breath momentarily, fearing an outburst. Stallman, contrary to expectations, takes the news in stride.

快十一点的时候，我们赶到餐馆。结果发现居然在餐馆的门前已经排了长队，我们要等二十分钟才能入席。我深知黑客们不喜欢浪费时间，于是屏住呼吸，生怕理查德大动肝火。可出乎意料，理查德竟然坦然接受了这个事实。

"It's too bad we couldn't have found somebody else to join us," he tells me. "It's always more fun to eat with a group of people."

理查德对我说：“就只有我们俩来了，真可惜。人多了吃起来才热闹。”

During the wait, Stallman practices a few dance steps. His moves are tentative but skilled. We discuss current events. Stallman says his only regret about not attending LinuxWorld was missing out on a press conference announcing the launch of the GNOME Foundation. Backed by Sun Microsystems and IBM, the foundation is in many ways a vindication for Stallman, who has long championed that free software and free-market economics need not be mutually exclusive. Nevertheless, Stallman remains dissatisfied by the message that came out.

等待期间，理查德开始走起舞步。他步步小心，可还是能看出有功底。我们开始聊起时事。理查德说，这次缺席LinuxWorld大会，最遗憾的部分是没能出席GNOME基金会的成立发布会。这个基金会是Sun公司和IBM一手操办，它很好地诠释了理查德的观点：所谓自由软件，并不与自由市场和自由经济相冲突。然而，理查德依旧不满意大会的论调。

“The way it was presented, the companies were talking about Linux with no mention of the GNU Project at all,” Stallman says.

“发布的时候，各个公司都在说着Linux，闭口不提GNU工程。”理查德说。

Such disappointments merely contrast the warm response coming from overseas, especially Asia, Stallman notes. A quick glance at the Stallman 2000 travel itinerary bespeaks the growing popularity of the free software message. Between recent visits to India, China, and Brazil, Stallman has spent 12 of the last 115 days on United States soil. His travels have given him an opportunity to see how the free software concept translates into different languages of cultures.

这种冷漠恰恰和地球另一边的热络形成对比。尤其是亚洲各国，甚是热情。看看理查德2000年的行程安排，就能看出自由软件的迅猛增长。在115天之中，理查德只在美国停留了12天，其他时间，则在印度，中国或巴西境内。他的旅行，让他见识了自由软件如何被翻译成不同语言，进入各国文化。

“In India many people are interested in free software, because they see it as a way to build their computing infrastructure without spending a lot of money,” Stallman says. “In China, the concept has been much slower to catch on. Comparing free software to free speech is harder to do when you don’t have

any free speech. Still, the level of interest in free software during my last visit was profound."

"在印度，很多人对自由软件感兴趣。因为他们觉得这样可以用很低的成本，建设起自己的计算机基础设施。"理查德说："在中国，自由软件的概念就传播得相对慢了一些。我们经常把软件自由和言论自由并提，表示这是软件用户的一种基本自由，可在中国，这种类比显然不太奏效。不过无论如何，我这次的中国之行，还是在一定程度上提高了自由软件的关注度。"

The conversation shifts to Napster, the San Mateo, California software company, which has become something of a media cause célèbre in recent months. The company markets a controversial software tool that lets music fans browse and copy the music files of other music fans. Thanks to the magnifying powers of the Internet, this so-called "peer-to-peer" program has evolved into a de facto online jukebox, giving ordinary music fans a way to listen to MP3 music files over the computer without paying a royalty or fee, much to record companies' chagrin.

话题很快转到Napster，这是一家加利福尼亚州圣马特奥市的软件公司。这几个月，这家公司成了各大媒体的宠儿。这家公司开发了一款倍受争议的软件，使得音乐爱好者之间可以互相拷贝音乐文件。借着互联网的东风，这类称作p2p（即端到端）的软件逐渐流行。如今，Napster俨然成了大型音乐盒，让各路音乐爱好者可以免费欣赏音乐。这一下子就惹恼了各大唱片公司。

Although based on proprietary software, the Napster system draws inspiration from the long-held Stallman contention that once a work enters the digital realm – in other words, once making a copy is less a matter of duplicating sounds or duplicating atoms and more a matter of duplicating information – the natural human impulse to share a work becomes harder to restrict. Rather than impose additional restrictions, Napster execs have decided to take advantage of the impulse. Giving music listeners a central place to trade music files, the company has gambled on its ability to steer the resulting user traffic toward other commercial opportunities.

尽管Napster是个专有软件，但它的流行却印证了理查德的想法：一旦某个作品进入数字世界，或者说，一旦复制这个作品变得便宜简单，那么人类本能地分享这个作品的行为将会难以阻止。Napster给了用户一片分享音乐的空间，这家公司接着就可以借助用户量来寻求更多商业机会。

The sudden success of the Napster model has put the fear in traditional record companies, with good reason. Just days before my Palo Alto meeting with Stallman, U.S. District Court Judge Marilyn Patel granted a request filed by the Recording Industry Association of America for an injunction against the file-sharing service. The injunction was subsequently suspended by the U.S. Ninth District Court of Appeals, but by early 2001, the Court of Appeals, too, would find the San Mateo-based company in breach of copyright law, a decision RIAA spokesperson Hillary Rosen would later proclaim a “clear victory for the creative content community and the legitimate online marketplace.”

Napster的迅猛成功让各大唱片公司开始担心。我来帕罗奥多之前的几天，玛莉莲·帕特尔法官刚刚在美国地方法院通过了一项由美国唱片协会提交的禁令。该禁令禁止了文件共享服务。而Napster之后上诉到上诉法院，使得这个禁令被美国第九巡回上诉法院暂停。但在2001年年初，上诉法院判定Napster的确违反了版权法。美国唱片协会发言人希拉里·罗森对这个判决的评价是：“保护了创作者，完善了在线市场方面的法律。”

For hackers such as Stallman, the Napster business model is troublesome in different ways. The company’s eagerness to appropriate time-worn hacker principles such as file sharing and communal information ownership, while at the same time selling a service based on proprietary software, sends a distressing mixed message. As a person who already has a hard enough time getting his own carefully articulated message into the media stream, Stallman is understandably reticent when it comes to speaking out about the company. Still, Stallman does admit to learning a thing or two from the social side of the Napster phenomenon.

对于理查德这样的黑客来说，Napster的商业模型内包涵着复杂的信息。一方面，这家公司大力倡导黑客社区所欣赏的信息共享精神；而另一方面，又出售着基于专有软件的服务。这种复杂性，让理查德对这一事件少言寡语，因为他知道，自己已经花了很大功夫去传播自己的思想，而对这种部分和自己思想相同；但很大程度又有矛盾的事件发表评论，很容易被媒体断章取义，平添混乱。可理查德依旧承认，从Napster事件上，他也认识到了一些有价值的东西。

“Before Napster, I thought it might be [sufficient] for people to privately redistribute works of entertainment,” Stallman says. “The number of people who find Napster useful, however, tells me that the right to redistribute copies not only on a neighbor-to-neighbor basis, but to the public at large, is essential

and therefore may not be taken away."

“以前，我觉得人们私下分享娱乐信息就[足够]了。”理查德说：“可如此庞大数量的用户，在Napster上分享音乐，这让我觉得，不仅仅是私下朋友和朋友之间的共享很重要。在公共场合，公开给大家分享信息的自由，也同样重要，不可剥夺。”

No sooner does Stallman say this than the door to the restaurant swings open and we are invited back inside by the host. Within a few seconds, we are seated in a side corner of the restaurant next to a large mirrored wall.

话到此，餐厅服务员赶来，告诉我们有了空位。于是我们被领班带到桌前，坐在餐厅一角，身旁的墙壁，挂着一整面镜子。

The restaurant's menu doubles as an order form, and Stallman is quickly checking off boxes before the host has even brought water to the table. “Deep-fried shrimp roll wrapped in bean-curd skin,” Stallman reads. “Bean-curd skin. It offers such an interesting texture. I think we should get it.”

服务员拿来菜单，和笔，让我们在菜单上勾选要点的菜。还没等服务员把茶水拿上桌，理查德就已经开始在自己的菜单上点菜了。“香煎鲜蝦腐皮卷”理查德读着菜单：“腐皮，我喜欢这口感。点一个。”

This comment leads to an impromptu discussion of Chinese food and Stallman's recent visit to China. “The food in China is utterly exquisite,” Stallman says, his voice gaining an edge of emotion for the first time this morning. “So many different things that I've never seen in the U.S., local things made from local mushrooms and local vegetables. It got to the point where I started keeping a journal just to keep track of every wonderful meal.”

这一下子又引来了理查德关于中餐和这次访问中国的即兴评论：“中国的饮食真是博大精深。”理查德大声说，这要算是他整个上午嗓门最大的一次了：“各种我在美国都没听说过的食物。还有很多当地特产，当地美食。当时突然我又想起来，我要办份杂志，专门记录每顿大餐的菜谱。”

The conversation segues into a discussion of Korean cuisine. During the same June, 2000, Asian tour, Stallman paid a visit to South Korea. His arrival ignited a mini-firestorm in the local media thanks to a Korean software conference attended by Microsoft founder and chairman Bill Gates that same week. Next to getting his photo above Gates's photo on the front page of the top Seoul

newspaper, Stallman says the best thing about the trip was the food. “I had a bowl of naeng myun, which is cold noodles,” says Stallman. “These were a very interesting feeling noodle. Most places don’t use quite the same kind of noodles for your naeng myun, so I can say with complete certainty that this was the most exquisite naeng myun I ever had.”

这讨论接着又转向韩国菜。同样也是在2000年6月，理查德的亚洲之行中，也去了韩国。他的到来，在当地媒体引起了一个不小的轰动，这部分原因，或许还要部分归功于当时也在韩国的微软创始人比尔·盖茨。理查德的照片这次上了首尔的热销报纸的头版，把比尔·盖茨的照片竟然挤到了下面。除了这件事情让理查德非常满意之外，当地的美食也让他欢欣鼓舞。他回忆：“我吃了一碗冷面。那面条口感非常独特，我们这边都不用那种面条做冷面。我敢肯定，那冷面是我有生以来吃到的最正点的冷面。”

The term “exquisite” is high praise coming from Stallman. I know this, because a few moments after listening to Stallman rhapsodize about naeng myun, I feel his laser-beam eyes singeing the top of my right shoulder.

“正点”一词在理查德嘴里应该是很高评价了。因为正当理查德介绍完冷面，我突然觉察到他的眼神飘到了我的右后方。

“There is the most exquisite woman sitting just behind you,” Stallman says.

“你后面坐了个很正点的妹子。”理查德说。

I turn to look, catching a glimpse of a woman’s back. The woman is young, somewhere in her mid-20s, and is wearing a white sequined dress. She and her male lunch companion are in the final stages of paying the check. When both get up from the table to leave the restaurant, I can tell without looking, because Stallman’s eyes suddenly dim in intensity.

我赶紧回头，瞟到了一个女人的背影。这女人很年轻，二十来岁的样子。穿了一件白色裙子，裙子上贴着闪闪发亮的饰品。她和同桌的一位男性已经用完餐，正在结帐。当两人起身离开的时候，我发现了理查德眼中瞬间失落的神情。

“Oh, no,” he says. “They’re gone. And to think, I’ll probably never even get to see her again.”

“啊！别啊！他们要走了。我估计以后也见不到她了。”理查德抱怨道。

After a brief sigh, Stallman recovers. The moment gives me a chance to discuss Stallman's reputation vis-à-vis the fairer sex. The reputation is a bit contradictory at times. A number of hackers report Stallman's predilection for greeting females with a kiss on the back of the hand.

一声叹息，理查德心神回位。这倒给了我一个机会，问问理查德的感情生活和审美偏好。因为坊间传言，他可是个轻佻的公子。有些时候，很多传言还有些自相矛盾。很多黑客都声称，理查德和女性见面，都会以吻手背的方式来打招呼。

A May 26, 2000 *Salon.com* article, meanwhile, portrays Stallman as a bit of a hacker lothario. Documenting the free software-free love connection, reporter Annalee Newitz presents Stallman as rejecting traditional family values, telling her, "I believe in love, but not monogamy."

在Salon.com上，有一份2000年5月26日的采访，把理查德描写成了一位黑客界的花花公子。文中把自由软件和自由恋爱做类比，作者安娜利·奈唯姿把理查德说成是拒绝传统家庭观念的叛逆者，他还说："我相信爱情，但不相信一夫一妻。"

Stallman lets his menu drop a little when I bring this up. "Well, most men seem to want sex and seem to have a rather contemptuous attitude towards women," he says. "Even women they're involved with. I can't understand it at all."

当我问起这方面问题时，理查德把菜单放下后说："这个……很多男人似乎都会不自觉地蔑视女性，看到女性就想到上床一类的事情。就连很多女人也自觉不自觉地融入到男性的这种倾向的思维中，可我无法理解这种态度。"

I mention a passage from the 1999 book *Open Sources* in which Stallman confesses to wanting to name the GNU kernel after a girlfriend at the time. The girlfriend's name was Alix, a name that fit perfectly with the Unix developer convention of putting an "x" at the end names of operating systems and kernels – e.g., "Linux." Alix was a Unix system administrator, and had suggested to her friends, "Someone should name a kernel after me." So Stallman decided to name the GNU kernel "Alix" as a surprise for her. The kernel's main developer renamed the kernel "Hurd," but retained the name "Alix" for part of it. One of Alix's friends noticed this part in a source snapshot and told her, and she was touched. A later redesign of the Hurd eliminated that

part.

我提到，在1999年，一本名为《开源软件之声》的书中，理查德在一篇文章里提到，当初他想以女朋友的名字来命名GNU操作系统的内核。这位女孩子名叫Alix，因为是以x结尾，所以和Unix界对操作系统内核的命名传统非常契合，比如Linux就遵照了这个命名传统。Alix当初是个Unix系统管理员，而且曾经和她朋友开玩笑地提到过：“应该有个内核以我的名字命名。”理查德于是决定把GNU的内核命名为Alix，以此给她一份惊喜。可最后内核的主要开发人员把内核名字改成了Hurd，把Alix作为其中一个模块的名字。Alix的一个朋友看到源码中出现了Alix的名字，于是把这件事情告诉了她。Alix听后非常感动。之后Hurd又经历了几次重新设计，最终还是去除了Alix这个模块。

For the first time all morning, Stallman smiles. I bring up the hand kissing. “Yes, I do do that,” Stallman says. “I’ve found it’s a way of offering some affection that a lot of women will enjoy. It’s a chance to give some affection and to be appreciated for it.”

整个一个上午，理查德脸上都挂着笑容。我问起关于吻手背的事情，理查德说：“哦，是的。我的确这么做。我觉得这种方式可以给很多女性带来亲切感，她们也都会喜欢。这是个很好的机会，你可以借此拉近彼此距离。”

Affection is a thread that runs clear through Richard Stallman’s life, and he is painfully candid about it when questions arise. “There really hasn’t been much affection in my life, except in my mind,” he says. Still, the discussion quickly grows awkward. After a few one-word replies, Stallman finally lifts up his menu, cutting off the inquiry.

亲切感是理查德生命中永恒的主题。问起这个问题，他非常坦诚：“在我的生命里，确实很少体验到亲切感，只是在我头脑之中还存在着这种感受。”可聊着聊着，话题就变得尴尬。理查德几次回应都只是从嘴里蹦出几个单字，再问几个问题，他就举起菜单，插话道：

“Would you like some shu mai?” he asks.

“你想吃点烧麦吗？”

When the food comes out, the conversation slaloms between the arriving courses. We discuss the oft-noted hacker affection for Chinese food, the weekly dinner runs into Boston’s Chinatown district during Stallman’s days as a staff programmer at the AI Lab, and the underlying logic of the Chinese

language and its associated writing system. Each thrust on my part elicits a well-informed parry on Stallman's part.

上菜间隙，谈话越显尴尬。我开始和他闲聊起来，问他关于中餐的问题，问他当初他在人工智能实验室做程序员的时候，每周去唐人街吃饭的问题，还问了他关于中文口语和汉字书写之间的逻辑联系。本想借此绕回主题，可每次尝试，都被理查德一套太极功夫挡了回去。

"I heard some people speaking Shanghainese the last time I was in China," Stallman says. "It was interesting to hear. It sounded quite different [from Mandarin]. I had them tell me some cognate words in Mandarin and Shanghainese. In some cases you can see the resemblance, but one question I was wondering about was whether tones would be similar. They're not. That's interesting to me, because there's a theory that the tones evolved from additional syllables that got lost and replaced. Their effect survives in the tone. If that's true, and I've seen claims that that happened within historic times, the dialects must have diverged before the loss of these final syllables."

"我上次去中国的时候听到人们说上海话。"理查德说，"很有趣，和普通话非常不同。我问他们把同一个词分别用上海话和普通话说一遍。有些时候你能大概听出点联系。我问他们声调是不是也类似，他们说不一样。这让我觉得很有趣。因为曾经有个假说认为，不同声调曾经是不同音素，之后语言不断演化，几个音素被不同声调取代。如果这个假说成立，那么声调不同就意味着这个方言是在音素演变成声调之前就独立发展的。"

The first dish, a plate of pan-fried turnip cakes, has arrived. Both Stallman and I take a moment to carve up the large rectangular cakes, which smell like boiled cabbage but taste like potato latkes fried in bacon.

第一个上来的菜，是萝卜糕。我和理查德都花了些力气，才把这大块的萝卜糕切开。它闻起来像是白灼生菜，吃起来又像煎土豆饼配培根。

I decide to bring up the outcast issue again, wondering if Stallman's teenage years conditioned him to take unpopular stands, most notably his uphill battle since 1994 to get computer users and the media to replace the popular term "Linux" with "GNU/Linux."

我决定继续讨论刚才说到一半的话题，我问他，是否觉得自己的童年经历使自己养成了如此特立独行的性格。其中很出名的一个事，就是他从1994年开始，

就致力于纠正广大计算机用户和各地媒体的误读，让他们使用GNU/Linux一词来替代Linux。

“I believe [being an outcast] did help me [to avoid bowing to popular views],” Stallman says, chewing on a dumpling. “I have never understood what peer pressure does to other people. I think the reason is that I was so hopelessly rejected that for me, there wasn’t anything to gain by trying to follow any of the fads. It wouldn’t have made any difference. I’d still be just as rejected, so I didn’t try.”

“我觉得，[特立独行的性格]的确让我[避免了对流行观点的盲从]。”理查德嚼着一个饺子，继续说道，“很多人都说会感受到旁人观点的压力，而我却从不在意。我觉得我一直在拒绝盲从别人的观点。随波逐流不会有什么好处，不会带来任何改变。因此，我也没太在意别人对我的看法。”

Stallman points to his taste in music as a key example of his contrarian tendencies. As a teenager, when most of his high school classmates were listening to Motown and acid rock, Stallman preferred classical music. The memory leads to a rare humorous episode from Stallman’s middle-school years. Following the Beatles’ 1964 appearance on the Ed Sullivan Show, most of Stallman’s classmates rushed out to purchase the latest Beatles albums and singles. Right then and there, Stallman says, he made a decision to boycott the Fab Four.

k理查德以音乐喜好为例，向我讲述他叛逆的倾向。在他少年时期，大多数同龄的中学生都迷恋魔城音乐和摇滚乐，而理查德则喜欢听古典乐。他回忆1964年，自己中学生时候，披头士在《埃德·沙利文秀场》栏目出现，他的同学们都冲出去买披头士的最新专辑或单曲。而就在那个时候，理查德决心抵制披头士音乐。

“I liked some of the pre-Beatles popular music,” Stallman says. “But I didn’t like the Beatles. I especially disliked the wild way people reacted to them. It was like: who was going to have a Beatles assembly to adulate the Beatles the most?”

理查德说：“我喜欢披头士之前的一些流行乐。可我不喜欢披头士。我更不喜欢人们追求披头士的那份疯狂。当时就好像说：谁能攒齐披头士的所有专辑，谁就是最崇拜披头士的。”

When his Beatles boycott failed to take hold, Stallman looked for other ways to point out the herd-mentality of his peers. Stallman says he briefly considered putting together a rock band himself dedicated to satirizing the Liverpool group.

理查德的抵制行为显然没造成多大影响，他还试图找点其他法子戏弄一下盲目的同龄人。他说，他曾经考虑过要自己组织个乐队，专门调侃披头士。

“I wanted to call it Tokyo Rose and the Japanese Beetles.”

“我想把这乐队叫做‘东京花’，或者‘日本披头士’。”

Given his current love for international folk music, I ask Stallman if he had a similar affinity for Bob Dylan and the other folk musicians of the early 1960s. Stallman shakes his head. “I did like Peter, Paul and Mary,” he says. “That reminds me of a great filk.”

我知道理查德喜欢世界各地的民族乐，我就问他对民谣歌手鲍勃·迪伦什么看法。理查德摇摇头：“我倒是更喜欢彼得，保罗和玛丽。他们让我想起了伟大的诽客（filk）音乐。”

When I ask for a definition of “filk,” Stallman explains that the term is used in science fiction fandom to refer to the writing of new lyrics for songs. (In recent decades, some filkers write melodies too.) Classic filks include “On Top of Spaghetti,” a rewrite of “On Top of Old Smokey,” and “Yoda,” filk-master “Weird” Al Yankovic’s Star Wars-oriented rendition of the Kinks tune, “Lola.”

我问他什么是诽客音乐，理查德解释说是一个在科幻圈子里的术语，专门指代为的现成歌曲重新填词后创作的歌曲。（在最近几年，一些诽客乐作者也会自己谱曲。）传统的诽客乐有《意大利面条之上》（On Top of Spaghetti），这是借用《旧思墨客山之上》（On Top of Old Smokey）的曲调。还有由诽客乐大师，“奇士”埃尔·杨可维克（“Weird” Al Yankovic，“奇士”是此人绰号，也被人称“奇士埃尔”）填词，改变自《罗拉》（Lola）的歌曲《尤达》（Yoda），显然，《尤达》是描写《星球大战》里的绝地武士尤达的。“

Stallman asks me if I would be interested in hearing the filk. As soon as I say yes, Stallman’s voice begins singing in an unexpectedly clear tone, using the tune of “Blowin’ in the Wind”:

理查德问我是不是想听他唱诽客歌。我说想听。他很快开始唱起来，声音出奇

地清晰。用的是《随风飘散》（Blowin' in the Wind）的曲调：

> How much wood could a woodchuck chuck,\ If a woodchuck could chuck wood?\ How many poles could a polak lock,\ If a polak could lock poles?\ How many knees could a negro grow,\ If a negro could grow knees?\ The answer, my dear,\ is stick it in your ear.\ The answer is, stick it in your ear…
>
> 一只土拨鼠要拨多少土，\ 才能称得上是土拨鼠？\ 波兰人要播多少栏目，\ 才配得上波兰这名属？\ 黑奴兄弟啊，你要黑掉多少弓弩，\ 才敢被人称作黑奴。\ 这一切的答案，亲爱的，\ 都掉进了你耳朵，粘到了听小骨，\ 答案啊，都粘到了听小骨……

The singing ends, and Stallman's lips curl into another child-like half smile. I glance around at the nearby tables. The Asian families enjoying their Sunday lunch pay little attention to the bearded alto in their midst. After a few moments of hesitation, I finally smile too.

曲罢音散，理查德唇角又露出了孩子般的笑容。我环视周围，旁边的几个亚洲人正享受着自己周日的饕餮，没把心思放在这位大胡子叔叔身上。犹豫了几秒，我也冲理查德笑了笑。

"Do you want that last cornball?" Stallman asks, eyes twinkling. Before I can screw up the punch line, Stallman grabs the corn-encrusted dumpling with his two chopsticks and lifts it proudly. "Maybe I'm the one who should get the cornball," he says.

"这个粟米球你还吃吗？"理查德眨着眼睛问到。我还没回答，理查德就拿筷子夹起粟米球往嘴里送，"那我就不客气了！"他说。

The food gone, our conversation assumes the dynamics of a normal interview. Stallman reclines in his chair and cradles a cup of tea in his hands. We resume talking about Napster and its relation to the free software movement. Should the principles of free software be extended to similar arenas such as music publishing? I ask.

饭菜吃完，我们准备进入采访的正题。理查德端起茶杯，靠在在椅子上。我们继续开始讨论Naspter和自由软件运动的话题。我问，自由软件的原则是否应该扩展到其他领域，比如音乐领域上？

“It’s a mistake to transfer answers from one thing to another,” says Stallman, contrasting songs with software programs. “The right approach is to look at each type of work and see what conclusion you get.”

“把一个问题的答案无条件地嫁接到另外一个问题上，这可就不对了。”理查德说。说起歌曲音乐和程序，他说：“正确的思路是，根据每类不同的作品做分析，由此再得出相应的结论。”

When it comes to copyrighted works, Stallman says he divides the world into three categories. The first category involves “functional” works – e.g., software programs, dictionaries, and textbooks. The second category involves works that might best be described as “testimonial” – e.g., scientific papers and historical documents. Such works serve a purpose that would be undermined if subsequent readers or authors were free to modify the work at will. It also includes works of personal expression – e.g., diaries, journals, and autobiographies. To modify such documents would be to alter a person’s recollections or point of view, which Stallman considers ethically unjustifiable. The third category includes works of art and entertainment.

当说起版权作品的时候，理查德把整个领域分为三个类型。第一个类型，是所谓的“功能类”。软件，辞典，教科书都属于此类。第二个类型，可以称为“证据类”。科技论文，古籍文献都属于此类。这些作品，如果可以被任意修改，则没有了价值。这一类也同样包括对个人感受的记录，比如日记，游记，自传。修改这些作品，则会影响到对某人的回忆或观点。因此理查德觉得对此类作品，读者则不能有权修改。第三类，则是“艺术类”，包含了艺术和娱乐作品。

Of the three categories, the first should give users the unlimited right to make modified versions, while the second and third should regulate that right according to the will of the original author. Regardless of category, however, the freedom to copy and redistribute noncommercially should remain unabridged at all times, Stallman insists. If that means giving Internet users the right to generate a hundred copies of an article, image, song, or book and then email the copies to a hundred strangers, so be it. “It’s clear that private occasional redistribution must be permitted, because only a police state can stop that,” Stallman says. “It’s antisocial to come between people and their friends. Napster has convinced me that we also need to permit, must permit, even noncommercial redistribution to the public for the fun of it. Because so many people want to do that and find it so useful.”

这三类作品中，应该无条件赋予用户权力去修改的是“功能类”的作品；而对于“证据类”和“艺术类”作品，则要根据作者意愿，才能赋予用户一定的修改权力。理查德坚信，无论哪类作品，用户出于非商业目的的复制与分享行为都应该被允许。如果说，一旦允许如此，就会有互联网用户把各种图片，音乐或者书籍复制成百上千份，然后把拷贝发送给几百个陌生人，那么也要允许这种行为。“很显然，私下偶尔分享这些作品的行为，必须被允许。只有警察国家才会禁止这些行为。”理查德说。“把一个人和他的朋友隔离，这是反社会的行为。而Napster事件则让我觉得，我们不仅要允许私下朋友之间的分享，更应该，也必须要允许用户，出于非商业目的，把这些作品的拷贝分享给公众。因为太多的人希望如此，并且觉得这么做非常有价值。”

When I ask whether the courts would accept such a permissive outlook, Stallman cuts me off.

我问他，法庭是不是会接受这种放任的观点。理查德打断了我说：

“That’s the wrong question,” he says. “I mean now you’ve changed the subject entirely from one of ethics to one of interpreting laws. And those are two totally different questions in the same field. It’s useless to jump from one to the other. How the courts would interpret the existing laws is mainly in a harsh way, because that’s the way these laws have been bought by publishers.”

“这是个错误的问题。你把这个话题从道德伦理的领域，转移成了法律领域里的释法问题。这是一个话题的两个完全不同的方面。从一个反面跳到另一个是没有任何意义的。法庭如何解释现有法律是有严格规定的，而这些法律也接着这些规定被各路出版商搬上台面。”

The comment provides an insight into Stallman’s political philosophy: just because the legal system currently backs up businesses’ ability to treat copyright as the software equivalent of land title doesn’t mean computer users have to play the game according to those rules. Freedom is an ethical issue, not a legal issue. “I’m looking beyond what the existing laws are to what they should be,” Stallman says. “I’m not trying to draft legislation. I’m thinking about what should the law do? I consider the law prohibiting the sharing of copies with your friend the moral equivalent of Jim Crow. It does not deserve respect.”

这段话反映出了理查德的政治哲学：现有的法律系统保护各大公司，让他们能利用现有版权法保护自己的软件，但这并不意味着计算机用户必须得遵守所谓

的规则。自由本身是个伦理问题，不是法律问题。他说：“我们需要超越现有法律，暂时忘记现在实际是什么样，去思考我们究竟应该要什么样。我不是要立法，而是在考虑法律应该做些什么。我觉得，禁止朋友之间分享拷贝的法律，和种族歧视法律一样，不该得到尊重。”

The invocation of Jim Crow prompts another question. How much influence or inspiration does Stallman draw from past political leaders? Like the civil-rights movement of the 1950s and 1960s, his attempt to drive social change is based on an appeal to timeless values: freedom, justice, and fair play.

说起种族歧视法，我又问了另外一个问题：他是否被曾经的那些政治领袖影响或鼓舞？比如五六十年代的民权运动，理查德的政治思想也是基于类似的普世价值：自由，正义，公平竞争。

Stallman divides his attention between my analogy and a particularly tangled strand of hair. When I stretch the analogy to the point where I’m comparing Stallman with Dr. Martin Luther King, Jr., Stallman, after breaking off a split end and popping it into his mouth, cuts me off.

.....

“I’m not in his league, but I do play the same game,” he says, chewing.

“我们不是一个团体，但我们做着同样的事情。”理查德一边嚼着东西一边说。

I suggest Malcolm X as another point of comparison. Like the former Nation of Islam spokesperson, Stallman has built up a reputation for courting controversy, alienating potential allies, and preaching a message favoring self-sufficiency over cultural integration.

我接着提起另外一位民权活动家，马尔科姆·艾克斯（Malcolm X），把他的观点和理查德的做类比。马尔科姆曾是美国非洲裔人的新宗教组织“伊斯兰国度”的领袖。和他一样，理查德也喜欢谈论有争议的话题，疏远潜在的盟友，并且传播着一套自恰的信息，无视与现有文化体系的整合。

Chewing on another split end, Stallman rejects the comparison. “My message is closer to King’s message,” he says. “It’s a universal message. It’s a message of firm condemnation of certain practices that mistreat others. It’s not a message of hatred for anyone. And it’s not aimed at a narrow group of people. I invite anyone to value freedom and to have freedom.”

理查德嚼完嘴里的东西，表示他拒绝这种类比：“我要传达的内容更类似马丁·路德·金的思想。这是一个普世价值观。它谴责一些伤及他人的行为。我的思想并不是要人去憎恨谁，也不是针对一小撮人。我希望每个人都能珍视自由，也能拥有自由。”

Many criticize Stallman for rejecting handy political alliances; some psychologize this and describe it as a character trait. In the case of his well-publicized distaste for the term “open source,” the unwillingness to participate in recent coalition-building projects seems understandable. As a man who has spent the last two decades stumping on the behalf of free software, Stallman’s political capital is deeply invested in the term. Still, comments such as the “Han Solo” comparison at the 1999 LinuxWorld have only reinforced Stallman’s reputation, amongst those who believe virtue consists of following the crowd, as a disgruntled mossback unwilling to roll with political or marketing trends.

很多人都指责理查德，说他经常拒绝触手可得的政治盟友。有人从心理学的角度，认为这是理查德的一个性格特征。他拒绝使用已经被大众接受的“开源”一词，拒绝与这个圈子的人合作项目。他为自由软件奋斗了二十多年，他的名字已经深深地和自由软件绑在了一起，他的政治资本也是要不断投资在这四个字上。在LinuxWorld上领奖时，他的那翻类比，更加深了人们对他的这一印象。茫茫众生之间，很多人早已把跟风赶潮当作一种美德。而理查德在这些人之中，则更显得头脑守旧，不肯在政治大趋势和市场面前有半点让步。

“I admire and respect Richard for all the work he’s done,” says Red Hat president Robert Young, summing up Stallman’s paradoxical political conduct. “My only critique is that sometimes Richard treats his friends worse than his enemies.”

红帽公司总裁罗伯特·杨（Robert Young）总结理查德看似矛盾的政治行为时，说道：“我崇拜也尊敬理查德和他所做的一切。我对他唯一的批评就是，有些时候，他对待朋友甚至比对待敌人还要无情。”

Stallman’s reluctance to ally the free software movement with other political causes is not due to lack of interest in them. Visit his offices at MIT, and you instantly find a clearinghouse of left-leaning news articles covering civil-rights abuses around the globe. Visit his personal web site, stallman.org, and you’ll find attacks on the Digital Millennium Copyright Act, the War on Drugs, and the World Trade Organization. Stallman explains, “We have to be careful of

entering the free software movement into alliances with other political causes that substantial numbers of free software supporters might not agree with. For instance, we avoid linking the free software movement with any political party because we do not want to drive away the supporters and elected officials of other parties."

理查德这种独行侠的作风，并非是他不关注其他政治人物。在他MIT的办公室里，你能一下子看到很多左派文章，记录着全球各地公民权被侵犯的事件。登陆他的个人网站，stallman.org，你可以看到很多文章，反对千禧年版权法案，抵制反毒品运动，抵制世贸组织等等。理查德解释说："我们必须要谨慎地为自由软件运动挑选政治盟友。因为有些自由软件支持者也许并不支持有些运动。比如，我们会避免把自由软件运动和其他政党建立联系，因为我们不希望这样的行为去驱散那些自由软件的支持者。"

Given his activist tendencies, I ask, why hasn't Stallman sought a larger voice? Why hasn't he used his visibility in the hacker world as a platform to boost his political voice? [RMS: But I do – when I see a good opportunity. That's why I started stallman.org.]

鉴于理查德这种活动家的倾向，我问他为什么不尝试发出更大的声音？为什么不借助他在黑客圈的影响力，去推进他的政治主张？[RMS: 可我其实已经在这么做，但只有在时机成熟的时候才如此。这也是我为什么建立了 stallman.org 这个个人网站]

Stallman lets his tangled hair drop and contemplates the question for a moment. [RMS: My quoted response doesn't fit that question. It does fit a different question, "Why do you focus on free software rather than on the other causes you believe in?" I suspect the question I was asked was more like that one.]

理查德整理了一下头发，思考了一阵。[RMS: 我下面的回答有些所问非所答。对于我下面的话，更合适的问题也许是："你为什么专注于自由软件，而不是其他你关注的问题？"我记得当时被问的问题似乎是这个。]

"I hesitate to exaggerate the importance of this little puddle of freedom," he says. "Because the more well-known and conventional areas of working for freedom and a better society are tremendously important. I wouldn't say that free software is as important as they are. It's the responsibility I undertook, because it dropped in my lap and I saw a way I could do something about it.

But, for example, to end police brutality, to end the war on drugs, to end the kinds of racism we still have, to help everyone have a comfortable life, to protect the rights of people who do abortions, to protect us from theocracy, these are tremendously important issues, far more important than what I do. I just wish I knew how to do something about them."

答道："软件自由，仅仅是自由一隅，冰山一角。我不希望过分强调这一角。因为还有更多的领域需要自由，这一点十分重要。和这些自由相比，自由软件则显得微不足道了。自由软件是我的一份责任，因为这项自由恰好落在我熟悉的领域之中。可除此以外，还有很多极其重要的问题需要关注：警察暴力，刑讯逼供；人们对某些毒品的偏见；种族歧视；如何让人们过上安逸的生活；如何让有堕胎经历的人免遭歧视；如何避免极权政治等等。这些都和我们的生活息息相关，它们都比我的工作重要。我只是希望能了解如何在这些问题上做点什么。"

Once again, Stallman presents his political activity as a function of personal confidence. Given the amount of time it has taken him to develop and hone the free software movement's core tenets, Stallman is hesitant to believe he can advance the other causes he supports.

理查德之后的一席话，让我觉得他利用了各种政治活动增进了个人信心。他在自由软件上花了大量时间和精力，很多活动，他虽然支持，但若真投入其中，则要犹豫再三，要权衡自己的时间和精力。

"I wish I knew how to make a major difference on those bigger issues, because I would be tremendously proud if I could, but they're very hard and lots of people who are probably better than I am have been working on them and have gotten only so far," he says. "But as I see it, while other people were defending against these big visible threats, I saw another threat that was unguarded. And so I went to defend against that threat. It may not be as big a threat, but I was the only one there [to oppose it]."

"我真心希望我能为解决这些问题，贡献出自己的一份力量。倘若我的贡献能让这些问题有所好转，我将引以为豪。可这些问题困难至极，多少比我更优秀的人，前赴后继，才只能做到今天这地步。当初我审视这一切，我发现依然有一处自由无人守护，于是我才投身至此。软件自由也许并非举足轻重，但守卫之人，仅我一个。"

Chewing a final split end, Stallman suggests paying the check. Before the

waiter can take it away, however, Stallman pulls out a white-colored dollar bill and throws it on the pile. The bill looks so clearly counterfeit, I can't help but pick it up and read it. Sure enough, it did not come from the US Mint. Instead of bearing the image of a George Washington or Abe Lincoln, the bill's front side bears the image of a cartoon pig. Instead of the United States of America, the banner above the pig reads, "Untied Status of Avarice." The bill is for zero dollars, and when the waiter picks up the money, Stallman makes sure to tug on his sleeve.

吃完最后一口，理查德要结帐。服务员还没来收钱，他又掏出一张白色纸币。一眼看去，这张纸币显然是张假币，我忍不住拾起来仔细端详了一下。它自然不是美国铸币局制造的。上面画的不是华盛顿，不是林肯，而是一头卡通猪。上面写的也不是"美利坚合众国"（United States of America），而是"放纵贪婪中"（Untied Status of Avarice）。这张纸币一钱不值。服务员来的时候，理查德拽着他的袖子：

"I added an extra zero to your tip," Stallman says, yet another half smile creeping across his lips.

"我又给你加了零美元的小费。"理查德说着，又露出了他标志的微笑。

The waiter, uncomprehending or fooled by the look of the bill, smiles and scurries away.

这位服务员可能没有领会什么意思，或者真的被这假币迷惑住了。他也回敬笑了一下，快步走开了。

"I think that means we're free to go," Stallman says.

"看来我们可以离开了。"理查德说。

The Emacs Commune Emacs公社

The AI Lab of the 1970s was by all accounts a special place. Cutting-edge projects and top-flight researchers gave it an esteemed position in the world of computer science. The internal hacker culture and its anarchic policies lent a rebellious mystique as well. Only later, when many of the lab's scientists and software superstars had departed, would hackers fully realize the unique and ephemeral world they had once inhabited.

二十世纪七十年代，从哪个角度看，麻省理工学院的人工智能实验室都是个特别的地方。里面有尖端的项目，顶级的研究人员，他们为整个实验室赢来了美名。而实验室内部的黑客文化，无政府主义的基调又为实验室平添了一层反抗权威的个性。几年之后，实验室中很多科学家和明星开发人员纷纷离开，黑客们这才意识到自己曾经的环境是多么独特。

“It was a bit like the Garden of Eden,” says Stallman, summing up the lab and its software-sharing ethos in a 1998 *Forbes* article. “It hadn’t occurred to us not to cooperate.”

在一篇1998年《福布斯》杂志上的文章中，斯托曼回忆当时实验室的氛围，说：“那就好像是个伊甸园。在那样的环境中，我们不可能拒绝互相合作。”

Such mythological descriptions, while extreme, underline an important fact. The ninth floor of 545 Tech Square was more than a workplace for many. For hackers such as Stallman, it was home.

这样的描述也许略显夸张，但却反映了一个很重要的事实：技术广场545号楼9层，那里曾经不仅仅是一个工作场所。对于像理查德一样的黑客来说，那里更像是家。

The word “home” is a weighted term in the Stallman lexicon. In a pointed swipe at his parents, Stallman, to this day, refuses to acknowledge any home before Currier House, the dorm he lived in during his days at Harvard. He has also been known to describe leaving that home in tragicomic terms. Once, while describing his years at Harvard, Stallman said his only regret was getting kicked out. It wasn’t until I asked Stallman what precipitated his ouster, that I realized I had walked into a classic Stallman setup line.

“家”这个字，在理查德的心中，有着特别的份量。他少年时，家中的变故和经历让他直到上了大学，才对家这个概念有所感悟，并心存感激。他曾把哈佛的宿舍当作自己第一个真正的家。描述起当年离开宿舍的时候，他甚至依然心存悲伤。他有次提起自己的大学生涯，说道大学期间，最悔恨的一件事情，就是被哈佛踢出校门。我问他究竟是触怒了何方神圣，才被赶出学校的。这才发现理查德早有准备：

“At Harvard they have this policy where if you pass too many classes they ask you to leave,” Stallman says.

“哈佛有个规矩，你修了太多的课程，就必须得毕业了。”理查德道。

With no dorm and no desire to return to New York, Stallman followed a path blazed by Greenblatt, Gosper, Sussman, and the many other hackers before him. Enrolling at MIT as a grad student, Stallman rented a room in an apartment in nearby Cambridge but soon viewed the AI Lab itself as his de facto home. In a 1986 speech, Stallman recalled his memories of the AI Lab during this period:

离开了宿舍，又不想回纽约。理查德跟随着和格林布拉特，高斯伯，萨斯曼等人的足迹，来到了麻省理工学院，继续读博士。他在剑桥市一带租了一所公寓，但没有多久，他就把人工智能实验室当作了自己真正的家。在1986年的一次演讲中，理查德回忆起那时的人工智能实验室：

> I may have done a little bit more living at the lab than most people, because every year or two for some reason or other I'd have no apartment and I would spend a few months living at the lab. And I've always found it very comfortable, as well as nice and cool in the summer. But it was not at all uncommon to find people falling asleep at the lab, again because of their enthusiasm; you stay up as long as you possibly can hacking, because you just don't want to stop. And then when you're completely exhausted, you climb over to the nearest soft horizontal surface. A very informal atmosphere.
>
> "比起其他人，我睡在实验室里的日子可能更久些。因为每隔一两年，我总会因为各种原因，会有那么几个月的时间没地方住。这期间，我就住在人工智能实验室里。我一直觉得那里很舒服，冬暖夏凉。那个时候，睡在实验室一点也不稀奇。他们都是热情满满，一直写代码调程序，因为你实在不想停下来。等到实在太累了，就在旁边找片平坦舒服的地方，倒头睡下。大家都是这么不拘一格。。"

The lab's home-like atmosphere could be a problem at times. What some saw as a dorm, others viewed as an electronic opium den. In the 1976 book *Computer Power and Human Reason*, MIT researcher Joseph Weizenbaum offered a withering critique of the "computer bum," Weizenbaum's term for the hackers who populated computer rooms such as the AI Lab. "Their rumpled clothes, their unwashed hair and unshaved faces, and their uncombed hair all testify that they are oblivious to their bodies and to the world in which they move," Weizenbaum wrote. "[Computer bums] exist, at least when so engaged, only through and for the computers."

实验室里的这种氛围，有时候也会引来麻烦。有些人把它看作是个宿舍，而外人看来就像是个电子大烟馆。1976年，麻省理工学院的一位研究人员约瑟夫·魏岑鲍姆（Joseph Weizenbaum）曾在一本名为《计算机的能力与人类的理性》（Computer Power and Human Reason）中，用“电脑流浪汉”来形容类似人工智能实验室里的这批人，并对此多有指责：“衣服打着褶，头发不洗，胡子不刮，蓬头垢面，这群人不管自己的形象，更不会关注外面的世界……这种电脑流浪汉就存在于我们的现实世界之中，他们沉迷于计算机，一切生活只能依赖电脑，而他们的日子里只有计算机。”

Almost a quarter century after its publication, Stallman still bristles when hearing Weizenbaum’s “computer bum” description, discussing it in the present tense as if Weizenbaum himself was still in the room. “He wants people to be just professionals, doing it for the money and wanting to get away from it and forget about it as soon as possible,” Stallman says. “What he sees as a normal state of affairs, I see as a tragedy.”

如今这篇文章已经发表了二十多年，而理查德听到“电脑流浪汉”这词依然心存不快。一说起这事，理查德就用一般现代式来形容，就好像魏岑鲍姆在他身边一般：“他就希望大家把这当成一份职业。你干这活就该只为挣钱，干完活你就走人，回到家就什么都不记得。他所谓的这种正常生活，在我看来根本就是个悲剧。”

Hacker life, however, was not without tragedy. Stallman characterizes his transition from weekend hacker to full-time AI Lab denizen as a series of painful misfortunes that could only be eased through the euphoria of hacking. As Stallman himself has said, the first misfortune was his graduation from Harvard. Eager to continue his studies in physics, Stallman enrolled as a graduate student at MIT. The choice of schools was a natural one. Not only did it give Stallman the chance to follow the footsteps of great MIT alumni: William Shockley (’36), Richard P. Feynman (’39), and Murray Gell-Mann (’51), it also put him two miles closer to the AI Lab and its new PDP-10 computer. “My attention was going toward programming, but I still thought, well, maybe I can do both,” Stallman says.

然而，黑客的生活中，也依然会有悲剧。理查德回忆，在他从人工智能实验室的兼职黑客，到转成全职黑客的这些年间，经历了一系列不平静的遭遇。他只能靠玩弄计算机来度过这段难熬的日子。正如他之前说的，第一个不幸遭遇就是从哈佛毕业。之后他凭着对物理学的兴趣，在麻省理工学院开始攻读物理系

的博士学位。选择麻省理工学院是个非常顺理成章的事情。一来，有众多著名校友做榜样：有1936年毕业的校友，晶体管发明者之一威廉· 肖克利; 1939年毕业的校友，著名的理论物理学家理查德·费曼；1951年毕业的校友，夸克之父默里·盖尔曼。除了这些名人以外，吸引理查德的，自然还有人工智能实验室和里面那台全新的PDP-10计算机。理查德说：“我那会越来越喜欢编程，但当时觉得，我没准能两者兼顾，编程、物理两不耽误。”

Toiling in the fields of graduate-level science by day and programming in the monastic confines of the AI Lab by night, Stallman tried to achieve a perfect balance. The fulcrum of this geek teeter-totter was his weekly outing with the Folk-Dance Club, his one social outlet that guaranteed at least a modicum of interaction with the opposite sex. Near the end of that first year at MIT, however, disaster struck. A knee injury forced Stallman to stop dancing. At first, Stallman viewed the injury as a temporary problem; he went to dancing and chatted with friends while listening to the music he loved. By the end of the summer, when the knee still ached and classes reconvened, Stallman began to worry. “My knee wasn’t getting any better,” Stallman recalls, “which meant I had to expect to be unable to dance, permanently. I was heartbroken.”

白天读着物理学的博士课程，晚上跑去人工智能实验室鼓捣计算机，理查德试着在这种生活里保持平衡。两头的忙碌，让他只能在每周末偷出几分闲暇，跑去民间舞俱乐部，放松一下身心。而这个活动，也成了他唯一能认识异性朋友的机会。然而，在麻省理工学院的第一年结束时，悲剧再次降临。他膝盖受伤，从此不能再跳舞。一开始，他还以为只是个小伤，养一阵子就会好。他还是照常参加俱乐部活动，聊聊天，听听音乐。可暑假结束，他膝盖依旧疼痛，加上新学期课程开始，理查德开始担心。他回忆：“我膝盖一直也没见好，这就意味着我很可能终生都无法跳舞了。我当时伤心透了。”

With no dorm and no dancing, Stallman’s social universe imploded. Dancing was the only situation in which he had found success in meeting women and occasionally even dating them. No more dancing ever was painful enough, but it also meant, it seemed, no more dates ever.

离开了大学时的宿舍，又没法继续跳舞，理查德丧失了任何社交的机会。跳舞是唯一能让他在女性面前获得成就感的活动，他甚至有时候还能借此和女生单独出来约会。不能跳舞本身已经让他够痛苦了，而这也同样意味着，他从此恐怕就没机会和女生约会了。

“I felt basically that I’d lost all my energy,” Stallman recalls. “I’d lost my energy

to do anything but what was most immediately tempting. The energy to do something else was gone. I was in total despair."

"我当时就觉得自己像个泄了气的气球。我失去了做各种事的动力，我彻底地绝望了。"

Stallman retreated from the world even further, focusing entirely on his work at the AI Lab. By October, 1975, he dropped out of MIT and out of physics, never to return to studies. Software hacking, once a hobby, had become his calling.

这之后，理查德再次远离了这个世界，专心投入人工智能实验室的工作。1975年十月，他放弃了麻省理工学院的课程，从此没再回到课堂。软件开发，曾经只是他的兴趣，如今则成了他的责任。

Looking back on that period, Stallman sees the transition from full-time student to full-time hacker as inevitable. Sooner or later, he believes, the siren's call of computer hacking would have overpowered his interest in other professional pursuits. "With physics and math, I could never figure out a way to contribute," says Stallman, recalling his struggles prior to the knee injury. "I would have been proud to advance either one of those fields, but I could never see a way to do that. I didn't know where to start. With software, I saw right away how to write things that would run and be useful. The pleasure of that knowledge led me to want to do it more."

回头看看这段往事，理查德觉得，从一个学生转变成一个全职黑客几乎是命中注定的。他觉得，自己早晚都会放弃其他任何追求，专心成为一名黑客。他说："物理和数学领域里，我很难做出一些自己原创的贡献。倘若我能在那里有所成就，我自然乐见其成。可我始终没能在这些领域里，找出自己的套路。我甚至都不知道从何开始。而在软件领域，我立即就能知道怎么写软件，怎么让它跑起来，怎么做出有用的东西。软件领域里的知识，让我有了继续深究下去的动力，也让我从中感受到了无尽的乐趣"

Stallman wasn't the first to equate hacking with pleasure. Many of the hackers who staffed the AI Lab boasted similar, incomplete academic résumés. Most had come in pursuing degrees in math or electrical engineering only to surrender their academic careers and professional ambitions to the sheer exhilaration that came with solving problems never before addressed. Like St. Thomas Aquinas, the scholastic known for working so long on his theological summae that he sometimes achieved spiritual visions, hackers reached

transcendent internal states through sheer mental focus and physical exhaustion. Although Stallman shunned drugs, like most hackers, he enjoyed the “high” that came near the end of a 20-hour coding bender.

这份乐趣，并非只有理查德一人能感受到。很多黑客早已乐此不疲。人工智能实验室里，充斥着很多这样的人，他们和理查德一样，都在学业中途，转职成了全职黑客。其中大多数人都是数学或电子工程专业出身，他们当初选专业，一来为拿个文凭，二来可以体验解决旷世难题后的那份欣喜。当年圣·托马斯·阿奎那（St. Thomas Aquinas）编写《神学大全》时，曾号称感受过上帝的异象。而这份体验，黑客们也常有感受。他们通过集中精神，身心齐动，也达到了内心上的超脱，开始了忘我地工作。虽然理查德和其他黑客都拒绝毒品，但在电脑前写上二十来小时的代码，也同样能给他们带来一份飘飘然的感觉。

Perhaps the most enjoyable emotion, however, was the sense of personal fulfillment. When it came to hacking, Stallman was a natural. A childhood’s worth of late-night study sessions gave him the ability to work long hours with little sleep. As a social outcast since age 10, he had little difficulty working alone. And as a mathematician with a built-in gift for logic and foresight, Stallman possessed the ability to circumvent design barriers that left most hackers spinning their wheels.

这其中，最能让他们陶醉的，恐怕就是那份无法言状的自我满足感。理查德简直天生就是个黑客。他少年时熬夜学习的经历，让他可以轻易胜任长时间的工作，并且只睡很少的觉。而他的数学背景，又让他有着过人的严密逻辑和数感。很多让其他黑客望而生畏的设计难题，放到理查德面前却变得轻而易举。

“He was special,” recalls Gerald Sussman, an AI Lab faculty member and (since 1985) board member of the Free Software Foundation. Describing Stallman as a “clear thinker and a clear designer,” Sussman invited Stallman to join him in AI research projects in 1973 and 1975, both aimed at making AI programs that could analyze circuits the way human engineers do it. The project required an expert’s command of Lisp, a programming language built specifically for AI applications, as well as understanding (supplied by Sussman) of how a human might approach the same task. The 1975 project pioneered an AI technique called dependency-directed backtracking or truth maintenance, which consists of positing tentative assumptions, noticing if they lead to contradictions, and reconsidering the pertinent assumptions if that occurs.

杰拉尔德·萨斯曼（Gerald Sussman）是人工智能实验室的教授，1985年之后，他还担任了自由软件基金会董事会成员。回忆起理查德，他赞叹：“他太特别了。思维清晰，设计流畅。”萨斯曼曾邀请了理查德加入人工智能实验室，参与了1973年和1975年的项目。两个人工智能的项目目标，都是通过编写程序，让计算机能够像人类的电子工程师一样，分析电路。这个项目需要一位Lisp语言的专家，同时，还能了解人类是如何解决类似的电路分析的问题。Lisp语言曾是专门设计用来编写人工智能程序。1975年的项目，则开辟了相关性制导回溯技术（dependency-directed backtracking）的先河，这个技术又名真值维护（Truth maintenance），基本思路是首先产生几个假设，然后检测在假设前提下是否存在矛盾，如果存在矛盾，则重新设计假设。

When he wasn’t working on official projects such as these, Stallman devoted his time to pet projects. It was in a hacker’s best interest to improve the lab’s software infrastructure, and one of Stallman’s biggest pet projects during this period was the lab’s editor program TECO.

这些正式项目之外，理查德也会花时间维护着自己的个人项目。黑客们都喜欢改进实验室的各种基础软件。理查德当时手头最大的个人项目，就是实验室的一个编辑器软件，名为TECO。

The story of Stallman’s work on TECO during the 1970s is inextricably linked with Stallman’s later leadership of the free software movement. It is also a significant stage in the history of computer evolution, so much so that a brief recapitulation of that evolution is necessary. During the 1950s and 1960s, when computers were first appearing at universities, computer programming was an incredibly abstract pursuit. To communicate with the machine, programmers created a series of punch cards, with each card representing an individual software command. Programmers would then hand the cards over to a central system administrator who would then insert them, one by one, into the machine, waiting for the machine to spit out a new set of punch cards, which the programmer would then decipher as output. This process, known as “batch processing,” was cumbersome and time consuming. It was also prone to abuses of authority. One of the motivating factors behind hackers’ inbred aversion to centralization was the power held by early system operators in dictating which jobs held top priority.

七十年代，理查德在TECO上的工作和之后的自由软件运动是一脉相承的。而这段历史在计算机史上，也是值得一书的。五六十年代那会，计算机刚刚进入大

学校园，所谓编程还是个很抽象的概念。那会的程序员，如果想要和计算机沟通一下，就必须得拿着一大摞卡片，上面打着孔，记录着软件中的指令。他们得把这一叠卡片交给系统管理员，让管理员把它们一张一张插到计算机里。计算机执行完卡片上的指令，然后把结果用打孔的方式，输出在另外几张卡片上。程序员则拿着这些计算机输出的卡片，回去分析解读。这个流程，通常被成为“批处理”。批处理是个非常费时费力的活，而且也给了计算机管理员太大的权力。黑客们痛恨权威的传统，恐怕很大一部分原因，要归咎于当时的计算机管理员权力过大，他们有权决定哪个程序优先运行。

In 1962, computer scientists and hackers involved in MIT’s Project MAC, an early forerunner of the AI Lab, took steps to alleviate this frustration. Time-sharing, originally known as “time stealing,” made it possible for multiple programs to take advantage of a machine’s operational capabilities. Teletype interfaces also made it possible to communicate with a machine not through a series of punched holes but through actual text. A programmer typed in commands and read the line-by-line output generated by the machine.

1962年，很多计算机科学家和黑客都参与了麻省理工学院的MAC计划（Project MAC）。这一计划是人工智能实验室的前身，它曾试图解决批处理带来的问题。MAC计划引入了“分时”（Time sharing）的概念。这个概念之前曾叫做“偷时”（Time stealing），它利用一个程序运行中间的空隙，来执行另外一个程序，由此多个程序可以有效利用计算机。同时，电子打字机也被引入进计算机系统，作为人际交互的设备。从此人们再也不用靠打孔来和计算机交互，人们可以利用打字机，把命令敲进去，然后等着计算机把结果一行一行打印到纸上。

During the late 1960s, interface design made additional leaps. In a famous 1968 lecture, Doug Engelbart, a scientist then working at the Stanford Research Institute, unveiled a prototype of the modern graphical interface. Rigging up a television set to the computer and adding a pointer device which Engelbart dubbed a “mouse,” the scientist created a system even more interactive than the time-sharing system developed at MIT. Treating the video display like a high-speed printer, Engelbart’s system gave a user the ability to move the cursor around the screen and see the cursor position updated by the computer in real time. The user suddenly had the ability to position text anywhere on the screen.

二十世纪六十年代，交互界面的设计也有了长足进步。在1968年，一次讲座从

此出名。讲座上，斯坦福研究中心的科学家，道格·英格巴特（Doug Engelbart）展示了第一款现代图形用户界面的原型。他们把计算机和电视机连起来，并且还加入了一个定点设备——英格巴特把这个定点设备昵称为“老鼠”，也就是我们今天的鼠标。这套系统的交互性比麻省理工学院的分时系统更好。他们把电视机当作一个高速打印机，来显示各种输出。这套系统还允许用户使用鼠标来移动屏幕上的光标，并且实时地显示出光标的位置。用户可以使用鼠标把光标移动到屏幕上的任何字符上。

Such innovations would take another two decades to make their way into the commercial marketplace. Still, by the 1970s, video screens had started to replace teletypes as display terminals, creating the potential for full-screen – as opposed to line-by-line – editing capabilities.

不过，这些发明要等二十年才能进入市场。到了七十年代，显示器逐渐开始作为显示设备，取代了电子打字机。这一改变，使得计算机可以使用全屏幕来显示。这就不必像以前一样，每次只打印几行内容在纸上。

One of the first programs to take advantage of this full-screen capability was the MIT AI Lab’s TECO. Short for Text Editor and COrrector, the program had been upgraded by hackers from an old teletype line editor for the lab’s PDP-6 machine.

人工智能实验室的TECO程序，是早期的几个使用全屏幕显示的程序之一。TECO是“文本编辑及修正程序”（Text Editor and COrrector）的缩写。它的前身，诞生于PDP-6和电子打字机的年代，是黑客们把它一步一步升级至今。

TECO was a substantial improvement over old editors, but it still had its drawbacks. To create and edit a document, a programmer had to enter a series of commands specifying each edit. It was an abstract process. Unlike modern word processors, which update text with each keystroke, TECO demanded that the user enter an extended series of editing instructions followed by an “end of command string” sequence just to change the text. Over time, a hacker grew proficient enough to make large changes elegantly in one command string, but as Stallman himself would later point out, the process required “a mental skill like that of blindfold chess.”

比起以前的那些编辑器，TECO是一大进步，但仍旧存在不足。如果要创建并编辑某个文件，程序员必须要输入各种命令，才能完成不同的操作。这个过程非常抽象。今天的编辑器，你每敲进一个字母，都会在屏幕上显示出来。而使用

TECO来编辑文件，则需要输入一些命令，然后告诉它“命令结束”，然后才能把文件修改好。经过一段时间的练习，一名黑客可以使用一套漂亮的命令来完成很大的修改。不过这种技能，正如理查德所说，是需要“类似下盲棋一样的脑力消耗”。

To facilitate the process, AI Lab hackers had built a system that displayed both the text and the command string on a split screen. Despite this innovative hack, editing with TECO still required skill and planning.

为了辅助这种编辑流程，人工智能实验室的黑客们开发了一套系统，可以把屏幕分为两部分，分别显示正在编辑的文件内容，和输入的命令。这个小改变的确有用，可是想要使用TECO依旧需要很多技巧，和事先规划。

TECO wasn’t the only full-screen editor floating around the computer world at this time. During a visit to the Stanford Artificial Intelligence Lab in 1976, Stallman encountered an edit program named E. The program contained an internal feature, which allowed a user to update display text after each command keystroke. In the language of 1970s programming, E was one of the first rudimentary WYSIWYG editors. Short for “what you see is what you get,” WYSIWYG meant that a user could manipulate the file by moving through the displayed text, as opposed to working through a back-end editor program.”

在当时，除了TECO，还有几个其他的全屏编辑器。1976年，理查德去了一趟斯坦福的人工智能实验室。在那里，他见到了一个叫做E的编辑器。这个程序有个功能，可以在根据用户的输入，实时更新显示器上的内容，让用户看到最新的修改结果。用七十年代的话讲，E是早期的几个“所见即所得”的编辑器之一。所谓“所见即所得”，常常写作WYSIWYG，是What you see is what you get的缩写。它意味着用户可以直接在显示出来的文本上进行编辑，而不用再另外使用一个后台的编辑器程序。

Impressed by the hack, Stallman looked for ways to expand TECO’s functionality in similar fashion upon his return to MIT. He found a TECO feature called Control-R, written by Carl Mikkelson and named after the two-key combination that triggered it. Mikkelson’s hack switched TECO from its usual abstract command-execution mode to a more intuitive keystroke-by-keystroke mode. The only flaws were that it used just five lines of the screen and was too inefficient for real use. Stallman reimplemented the feature to use the whole screen efficiently, then extended it in a subtle but significant way. He made it possible to attach TECO command strings, or “macros,” to keystrokes.

Advanced TECO users already saved macros in files; Stallman's hack made it possible to call them up fast. The result was a user-programmable WYSIWYG editor. "That was the real breakthrough," says Guy Steele, a fellow AI Lab hacker at the time.

理查德一下子对这技术感兴趣了，他决定回到麻省理工学院之后，把这个功能加到TECO上。他发现TECO上有个功能叫做"Control-R"。这个功能是卡尔·米克尔松（Carl Mikkelson）开发的。使用这个功能的快捷键和它的名字一样：Control-R。利用这个快捷键，用户可以让TECO实时地显示出编辑的内容。不过有个小瑕疵：用户只能使用五行来显示编辑的内容，这显然是不够的。理查德重新实现了这个功能，让程序可以使用整个屏幕来显示编辑内容。他还做了个不起眼，但影响深远的修改：他还允许用户把一连串TECO命令绑定在快捷键组合上。这一连串的命令称为"宏"。TECO的资深用户早就把各种常用的命令组合记下来了。理查德的这个改变让他们可以通过宏更快捷地使用这些组合。最后的结果，是一个允许用户自己扩展的"所见即所得"的编辑器。盖·斯蒂尔（Guy Steele)曾是当年人工智能实验室的一名黑客，他回忆起理查德的这个改动，说："这在当时是个突破。"

By Stallman's own recollection, the macro hack touched off an explosion of further innovation. "Everybody and his brother was writing his own collection of redefined screen-editor commands, a command for everything he typically liked to do," Stallman would later recall. "People would pass them around and improve them, making them more powerful and more general. The collections of redefinitions gradually became system programs in their own right."

理查德回忆，加入"宏"之后，各种创造接踵而来。他说："大家都开始把各种常用命令的组合写成宏，然后把各自的宏互相分享，大家一起不断改进，再分享。这样，这些宏越来越强大，涵盖了很多常用操作。这些宏俨然成为了一套单独的系统软件。"

So many people found the macro innovations useful and had incorporated it into their own TECO programs that the TECO editor had become secondary to the macro mania it inspired. "We started to categorize it mentally as a programming language rather than as an editor," Stallman says. Users were experiencing their own pleasure tweaking the software and trading new ideas.

越来越多的人开始使用宏，把各种宏加入到自己的TECO编辑器上。随着大家对宏的狂热，TECO作为编辑器的功能反而倒在其次了。理查德说："我们开始意识到，TECO已经不仅仅是个编辑器，更是个编程语言。"用户不断改进自己的

想法，不断交流分享，大家身在其中，自得其乐。

Two years after the explosion, the rate of innovation began to exhibit inconvenient side effects. The explosive growth had provided an exciting validation of the collaborative hacker approach, but it had also led to incompatibility. “We had a Tower of Babel effect,” says Guy Steele.

又过了两年，各种创新最后带来了一些麻烦。大家互相交流分享，使用的宏越来越多。每个人手头都有自己的一套宏，由此就引来了各种兼容性问题。盖·斯蒂尔说：“我们遇到了沟通障碍。”

The effect threatened to kill the spirit that had created it, Steele says. Hackers had designed ITS to facilitate programmers’ ability to share knowledge and improve each other’s work. That meant being able to sit down at another programmer’s desk, open up a programmer’s work and make comments and modifications directly within the software. “Sometimes the easiest way to show somebody how to program or debug something was simply to sit down at the terminal and do it for them,” explains Steele.

这种障碍阻碍了大家的交流，盖·斯蒂尔说。黑客们当年设计了ITS系统，其中一个很重要的设计原则，就是帮助程序员们分享知识，互相改进各自的工作。这就意味着，程序员可以坐到另外一个同事的电脑前，打开这个同事的程序代码，直接修改代码，或者加上几行注释。斯蒂尔解释说：“有些时候，要想教人怎么写程序，或者怎么调试某个程序，最好的办法就是坐在电脑前，实际演示给他们看。”

The macro feature, after its second year, began to foil this capability. In their eagerness to embrace the new full-screen capabilities, hackers had customized their versions of TECO to the point where a hacker sitting down at another hacker’s terminal usually had to spend the first hour just figuring out what macro commands did what.

引入宏之后的第二年，这种做法变得越来越难。黑客们开始在自己的TECO编辑器里加入各种宏，不断扩展TECO的功能。这就导致你坐到一台电脑前，第一件事情是需要了解每个宏都是做什么的。

Frustrated, Steele took it upon himself to solve the problem. He gathered together the four different macro packages and began assembling a chart documenting the most useful macro commands. In the course of implementing

the design specified by the chart, Steele says he attracted Stallman's attention.

一番受挫之后，盖·斯蒂尔决定着手解决这个问题。他收集到四套常用的宏，打算根据每套宏实现的功能，列出常用的功能，写成一份文档，并根据这四套宏，实现出一套符合这份文档的宏。在实现的过程中，盖·斯蒂尔的工作吸引了理查德的注意。

"He started looking over my shoulder, asking me what I was doing," recalls Steele.

盖·斯蒂尔回忆说："他站在我身后，看着我的屏幕，问我在做什么。"

For Steele, a soft-spoken hacker who interacted with Stallman infrequently, the memory still sticks out. Looking over another hacker's shoulder while he worked was a common activity at the AI Lab. Stallman, the TECO maintainer at the lab, deemed Steele's work "interesting" and quickly set off to complete it.

盖·斯蒂尔是个说话温和的黑客，并没怎么和理查德打过交道。回忆起当时的情景，他依旧记忆犹新。在当时的人工智能实验室里，站在别人身后看看别人的工作，是个非常平常的举动。理查德，在当年是TECO的维护者。看到盖·斯蒂尔的工作，他觉得非常有趣，于是决定帮助他完善这个工作。

"As I like to say, I did the first 0.001 percent of the implementation, and Stallman did the rest," says Steele with a laugh.

说起这事，盖·斯蒂尔笑道："我常跟人们这么说，我做了最初的百分之零点零零一的工作，理查德把剩下的都做了。"

The project's new name, Emacs, came courtesy of Stallman. Short for "editing macros," it signified the evolutionary transcendence that had taken place during the macros explosion two years before. It also took advantage of a gap in the software programming lexicon. Noting a lack of programs on ITS starting with the letter "E," Stallman chose Emacs, making it natural to reference the program with a single letter. Once again, the hacker lust for efficiency had left its mark.

这个项目的名字被称为Emacs，是理查德建议的名字。Emacs是Editing macros的缩写，意思是"宏编辑器"。它标志这宏出现之后的又一个进步。它也参考了当

年各种软件的名字，理查德注意到，在ITS系统上，还没有哪个软件名是以E开头的。把它叫做Emacs，用户就可以自己在设置中把它叫做E，一个字母就能运行这个程序。这又得归咎到黑客简约的风格上了。

Of course, not everyone switched to Emacs, or not immediately. Users were free to continue maintaining and running their own TECO-based editors as before. But most found it preferable to switch to Emacs, especially since Emacs was designed to make it easy to replace or add some parts while using others unchanged.

当然，一开始，并不是所有人都转头使用Emacs。用户依旧可以继续使用他们以前的TECO编辑器。不过大多数人都觉得转用Emacs更加方便。Emacs有着更强的扩展性，用户可以按照自己的需求，轻松地替换或者增加某些功能。

“On the one hand, we were trying to make a uniform command set again; on the other hand, we wanted to keep it open ended, because the programmability was important,” recalls Steele.

盖·斯蒂尔说：“一方面，我们试图创造一个统一的命令集。另一方面，我们还希望程序能有很强的扩展性，因为这一点至关重要。”

Stallman now faced another conundrum: if users made changes but didn’t communicate those changes back to the rest of the community, the Tower of Babel effect would simply emerge in other places. Falling back on the hacker doctrine of sharing innovation, Stallman embedded a statement within the source code that set the terms of use. Users were free to modify and redistribute the code on the condition that they gave back all the extensions they made. Stallman called this “joining the Emacs Commune.” Just as TECO had become more than a simple editor, Emacs had become more than a simple software program. To Stallman, it was a social contract. In a 1981 memo documenting the project, Stallman spelled out the contract terms. “EMACS,” he wrote, “was distributed on a basis of communal sharing, which means that all improvements must be given back to me to be incorporated and distributed.”

这个时候，理查德又遇到了一个难题：如果用户做出了修改，但是并不把这些修改分享出去，沟通障碍早晚会再次出现。本着黑客的分享精神，理查德在源代码里写上了使用条款。用户可以自由地修改和分发这个软件的代码，但必须把所有的改动都发回来。理查德把这称作“加入Emacs公社”。当年的TECO最后

变得不仅仅是个编辑器，如今这个Emacs则变得不仅仅是个软件。对于理查德来说，这更是个社会契约。在1981年的一份备忘录上，理查德在Emacs这个项目下写道：“Emacs是以分享的形式发布的。对Emacs的各种修改，必须要发给我，以便加入到以后的版本中。”

The original Emacs ran only on the PDP-10 computer, but soon users of other computers wanted an Emacs to edit with. The explosive innovation continued throughout the decade, resulting in a host of Emacs-like programs with varying degrees of cross-compatibility. The Emacs Commune's rules did not apply to them, since their code was separate. A few cited their relation to Stallman's original Emacs with humorously recursive names: Sine (Sine is not Emacs), Eine (Eine is not Emacs), and Zwei (Zwei was Eine initially). A true Emacs had to provide user-programmability like the original; editors with similar keyword commands but without the user-programmability were called "ersatz Emacs." One example was Mince (Mince is Not Complete Emacs).

Emacs原本只能运行在PDP-10上。不过很快，其他平台上的用户也希望能运行Emacs编辑器。之后十年，各种创造依旧继续。带来了各种类似Emacs的编辑器，运行在不同平台上。Emacs公社的规矩则不适用于这些编辑器上，因为它们完全是另外一套代码。这些编辑器中，有些在名字上就提及了理查德的Emacs编辑器。比如：Sine，全称Sine is not Emacs；Eine，全称Eine is not Emacs以及Zwei，全称Zwei was Eine initially。真正的类Emacs软件必须要提供用户可编程的扩展。一些Emacs的克隆软件只是使用了Emacs的快捷键，并没有提供扩展机制，这些一般被成为“伪Emacs”。其中一个例子就是Mine，全称Mine is Not Complete Emacs，意为“Mine不是一个完成的Emacs”。

While Stallman was developing Emacs in the AI Lab, there were other, unsettling developments elsewhere in the hacker community. Brian Reid's 1979 decision to embed "time bombs" in Scribe, making it possible for Unilogic to limit unpaid user access to the software, was a dark omen to Stallman. "He considered it the most Nazi thing he ever saw in his life," recalls Reid. Despite going on to later Internet fame as the co-creator of the Usenet *alt* hierarchy, Reid says he still has yet to live down that 1979 decision, at least in Stallman's eyes. "He said that all software should be free and the prospect of charging money for software was a crime against humanity."

理查德在人工智能实验室开发Emacs的时候，黑客圈子里还有很多其他的开发，也干得热火朝天。正如第一章提及的那样，1979年，布莱恩·瑞德（Brian

Reid）决定在Scribe里放入“定时炸弹”，这样就可以让没有付费的用户无法使用这个软件，以此让Unilogic公司获利。布莱恩的这个决定在理查德看来不是个好兆头。布莱恩回忆起理查德对此的看法，说：“他觉得，这简直就是法西斯行为。”布莱恩之后创立了Usenet新闻组上的alt分支，可每次提起理查德的指责，他依旧愤愤不平：“他说所有的软件都得免费，他觉得所有软件的收费行为都是违背人性的。”

Although Stallman had been powerless to head off Reid’s sale, he did possess the ability to curtail other forms of behavior deemed contrary to the hacker ethos. As central source-code maintainer for the original Emacs, Stallman began to wield his power for political effect. During his final stages of conflict with the administrators at the Laboratory for Computer Science over password systems, Stallman initiated a software “strike,” refusing to send lab members the latest version of Emacs until they rejected the security system on the lab’s computers. This was more gesture than sanction, since nothing could stop them from installing it themselves. But it got the point across: putting passwords on an ITS system would lead to condemnation and reaction.

虽然理查德无法阻止布莱恩给用户装上“定时炸弹”，可他还是有自己的一套法子，来遏制其他类似的行为。作为Emacs代码的主要维护者，理查德开始利用他的力量，来发起一轮政治影响。当时，为了去掉系统的登录密码，他正跟计算机系的机房管理员闹得不可开交。在冲突的关键时刻，他发起了一轮“软件抗议”：如果实验室的成员不反对系统登录密码，理查德就不给他们Emacs用。理查德的这种做法，只是做出一个姿态。因为如果哪个人想用Emacs，他们还是可以自己动手安装的。理查德是想借此传达一个信息：如果你在ITS系统上使用密码，那么将会被人嫌弃，受人指责。

“A lot of people were angry with me, saying I was trying to hold them hostage or blackmail them, which in a sense I was,” Stallman would later tell author Steven Levy. “I was engaging in violence against them because I thought they were engaging in violence to everyone at large.”

在接受《黑客》一书作者史蒂芬·李维（Steven Levy）采访时，理查德回忆说：“很多人觉得我绑架了实验室的各位成员，我在勒索大家。从某种意义上说，我的确如此。因为我觉得那些管理员在用暴力威胁着每一个人，我只能以暴制暴。”

Over time, Emacs became a sales tool for the hacker ethic. The flexibility Stallman had built into the software not only encouraged collaboration, it

demanded it. Users who didn't keep abreast of the latest developments in Emacs evolution or didn't contribute their contributions back to Stallman ran the risk of missing out on the latest breakthroughs. And the breakthroughs were many. Twenty years later, users of GNU Emacs (a second implementation started in 1984) have modified it for so many different uses – using it as a spreadsheet, calculator, database, and web browser – that later Emacs developers adopted an overflowing sink to represent its versatile functionality. "That's the idea that we wanted to convey," says Stallman. "The amount of stuff it has contained within it is both wonderful and awful at the same time."

渐渐地，Emacs成了黑客文化的宣传品。Emacs有着极强的扩展性。这种扩展性增进了用户之间的交流合作。不仅如此，Emacs甚至要求这样的合作。如果用户不把自己的修改贡献出来，就很可能用不到最新的版本，用不上最新的功能。而每次更新，各种新功能可是不少。1984年，理查德又重写了Emacs，命名为GNU Emacs。如今，GNU Emacs的用户遍布全球，大家把它已经扩展得异常强大。用户可以把它用作电子表格，当作计算器，用作数据库，用作网络浏览器等等。以至于之后的Emacs开发者，都找不出什么词来概括Emacs的功能了。理查德说："这就是我们想要传达的。Emacs里包含的东西既有用，又有趣。"

Stallman's AI Lab contemporaries are more charitable. Hal Abelson, an MIT grad student who worked with Sussman during the 1970s and would later assist Stallman as a charter board member of the Free Software Foundation, describes Emacs as "an absolutely brilliant creation." In giving programmers a way to add new software libraries and features without messing up the system, Abelson says, Stallman paved the way for future large-scale collaborative software projects. "Its structure was robust enough that you'd have people all over the world who were loosely collaborating [and] contributing to it," Abelson says. "I don't know if that had been done before."

和理查德同时代的人工智能实验室成员更是感谢理查德的贡献。哈尔·埃布尔森（Hal Abelson）曾是麻省理工学院的一名博士。七十年代，他曾在萨斯曼教授手下做过研究。之后，帮助理查德成立自由软件基金会，并成为董事会成员。形容起Emacs，他说："那绝对是个精品佳作！"他描述，Emacs既可以让程序员不断添加新功能，还不会影响整个系统。理查德的经验，为未来大规模合作开发的软件工程铺平了道路。"它结构稳定，可以接受世界各地的人贡献代码。这种松散的大规模协作开发，恐怕要算前无古人了。"

Guy Steele expresses similar admiration. Currently a research scientist for Sun Microsystems, he remembers Stallman primarily as a “brilliant programmer with the ability to generate large quantities of relatively bug-free code.” Although their personalities didn’t exactly mesh, Steele and Stallman collaborated long enough for Steele to get a glimpse of Stallman’s intense coding style. He recalls a notable episode in the late 1970s when the two programmers banded together to write the editor’s “pretty print” feature. Originally conceived by Steele, pretty print was another keystroke-triggered feature that reformatted Emacs’ source code so that it was both more readable and took up less space, further bolstering the program’s WYSIWYG qualities. The feature was strategic enough to attract Stallman’s active interest, and it wasn’t long before Steele wrote that he and Stallman were planning an improved version.

盖·斯蒂尔（Guy Steele)对此也表示了钦佩。如今，盖·斯蒂尔已经是Sun公司的一名科研人员。回忆起理查德，他脑海中浮现的是“一位才华横溢的程序员，他可以瞬间写出大量几乎没有bug的代码。”尽管他和理查德性格不太合得来，可他们俩还是合作了一段时间。这段时间里，盖·斯蒂尔对理查德的那种暴风闪电般的编程风格印象深刻。他记得在七十年代末，他俩曾一起为Emacs添加“重排版”的功能。所谓“重排版”，最初是斯蒂尔的点子。它可以让用户使用快捷键，来为Emacs中编辑的代码重新排版，使得代码更加易读。这个功能更增强了软件“所见即所得”的质量。这个功能一下子吸引了理查德的注意，接着斯蒂尔和理查德决定一起开发这个功能的改进版。

“We sat down one morning,” recalls Steele. “I was at the keyboard, and he was at my elbow,” says Steele. ``He was perfectly willing to let me type, but he was also telling me what to type.

斯蒂尔回忆：“我们早晨开始坐到电脑前。我敲键盘，他就坐我身边，告诉我怎么写。”

The programming session lasted 10 hours. Throughout that entire time, Steele says, neither he nor Stallman took a break or made any small talk. By the end of the session, they had managed to hack the pretty print source code to just under 100 lines. “My fingers were on the keyboard the whole time,” Steele recalls, “but it felt like both of our ideas were flowing onto the screen. He told me what to type, and I typed it.”

俩人就这样，持续开发了10个小时。斯蒂尔说，这10个小时，他们俩人谁也没

休息，甚至都没聊多余的话。到最后，他们把整个功能精简到一百多行代码。斯蒂尔回忆：“我当时手指就没离开键盘。我就觉得我俩的想法直接就流到了屏幕上。他告诉我写什么，我就按他说的写。”

The length of the session revealed itself when Steele finally left the AI Lab. Standing outside the building at 545 Tech Square, he was surprised to find himself surrounded by nighttime darkness. As a programmer, Steele was used to marathon coding sessions. Still, something about this session was different. Working with Stallman had forced Steele to block out all external stimuli and focus his entire mental energies on the task at hand. Looking back, Steele says he found the Stallman mind-meld both exhilarating and scary at the same time. “My first thought afterward was [that] it was a great experience, very intense, and that I never wanted to do it again in my life.”

整整10个小时，光是这时间长度，就足够显示出理查德的编程风格了。斯蒂尔离开实验室，走出技术广场545号大楼。外面已是夜幕降临，他虽然早就习惯了这种马拉松式的编程，但这次却别有一番滋味。跟理查德一起工作，他必须要集中精力，心无旁骛。如今回忆起来，斯蒂尔说，理查德的这份才智和精力，既让人振奋，又令人生畏。“我回想起来，第一感觉是觉得那是个很好的经历，流程紧凑，快速高效。再细想想，妈呀，我可不想再来一次。”

A Stark Moral Choice 道分左右 义无旁支

On September 27, 1983, computer programmers logging on to the Usenet newsgroup net.unix-wizards encountered an unusual message. Posted in the small hours of the morning, 12:30 a.m. to be exact, and signed by rms@mit-oz, the message’s subject line was terse but attention-grabbing. “New UNIX implementation,” it read. Instead of introducing a newly released version of Unix, however, the message’s opening paragraph issued a call to arms:

1983年9月27日，众多计算机用户像往常一样，登录Usenet的net.unix-wizards新闻组。一条不太寻常的消息映入眼帘。这条消息在当天凌晨过后发出来，准确时间是零点三十分。消息来自署名rms@mit-oz的用户。标题甚是简单，却异常乍眼：《重写UNIX系统》（New UNIX implementation）。这个消息并非是介绍新的UNIX发行版，而是更像一篇邀人入伍的檄文，文章开头写道：

> Starting this Thanksgiving I am going to write a complete Unix-compatible software system called GNU (for Gnu's Not Unix), and give it away free to everyone who can use it. Contributions of time, money, programs and equipment are greatly needed.
>
> 这个感恩节假期，我将开始写一个完全的UNIX兼容系统，叫做GNU，意为“GNU's Not Unix（GNU不是UNIX）“。这个系统任何人都可以自由使用。你如果愿意贡献时间，金钱，程序或者设备，我随时欢迎。

To an experienced Unix developer, the message was a mixture of idealism and hubris. Not only did the author pledge to rebuild the already mature Unix operating system from the ground up, he also proposed to improve it in places. The new GNU system, the author predicted, would carry all the usual components – a text editor, a shell program to run Unix-compatible applications, a compiler, “and a few other things.” It would also contain many enticing features that other Unix systems didn't yet offer: a graphic user interface based on the Lisp programming language, a crash-proof file system, and networking protocols built according to MIT's internal networking system.

对于UNIX的资深用户来说，这个消息显得太过理想主义，甚至有些自大傲慢。面对已经非常成熟的UNIX系统，这则消息不仅号称要从头克隆一套类似的操作系统，甚至还要在开发过程中改进现有UNIX的设计。这则消息的作者声称，新的GNU系统，会提供各种常用软件。包括文本编辑器，用来执行各种命令和程序的shell，编译器，还有“一些其他的东西”。除此以外，它还提供很多原本UNIX并不提供的功能，也非常吸引眼球。包括一套基于Lisp语言的图形用户界面；一个防崩溃的文件系统；以及一套基于麻省理工学院内部网络系统的网络协议栈。

“GNU will be able to run Unix programs, but will not be identical to Unix,” the author wrote. “We will make all improvements that are convenient, based on our experience with other operating systems.”

“GNU可以运行现有的UNIX程序，但和现有的UNIX并不完全相同，”作者写道：“我们会在开发过程中，融合我们的经验和对其他操作系统的了解，不断改进现有的UNIX系统设计。”

Anticipating a skeptical response on some readers' part, the author made sure to follow up his operating-system outline with a brief biographical sketch titled, “Who am I?”:

读者的各种质疑显然早在意料之中，整个消息最后，作者加入了一段《我是谁？》的自我介绍：

> I am Richard Stallman, inventor of the original much-imitated EMACS editor, now at the Artificial Intelligence Lab at MIT. I have worked extensively on compilers, editors, debuggers, command interpreters, the Incompatible Timesharing System and the Lisp Machine operating system. I pioneered terminal-independent display support in ITS. In addition I have implemented one crashproof file system and two window systems for Lisp machines.
>
> 我是理查德·斯托曼，我创造了众人效仿的EMACS编辑器。现在在麻省理工学院人工智能实验室。我的工作涉及编译器，编辑器，调试器，解释器，还有非兼容分时系统（ITS）以及Lisp机的操作系统。我创造了ITS上早期的独立于各终端硬件的显示技术。我还在Lisp机上，实现了一个防崩溃的文件系统，和两个窗口系统。“

As fate would have it, Stallman’s fanciful GNU Project missed its Thanksgiving launch date. By January, 1984, however, Stallman made good on his promise and fully immersed himself in the world of Unix software development. For a software architect raised on ITS, it was like designing suburban shopping malls instead of Moorish palaces. Even so, building a Unix-like operating system had its hidden advantages. ITS had been powerful, but it also possessed an Achilles’ heel: MIT hackers had written it specifically to run on the powerful DEC-built PDP-10 computer. When AI Lab administrators elected to phase out the lab’s PDP-10 machine in the early 1980s, the operating system that hackers once likened to a vibrant city became an instant ghost town. Unix, on the other hand, was designed for portability, which made it immune to such dangers. Originally developed by junior scientists at AT&T, the program had slipped out under corporate-management radar, finding a happy home in the cash-strapped world of academic computer systems. With fewer resources than their MIT brethren, Unix developers had customized the software to ride atop a motley assortment of hardware systems, primarily the 16-bit PDP-11 – a machine considered fit for only small tasks by most AI Lab hackers – but later also 32-bit mainframes such as the VAX 11/780. By 1983, a few companies, most notably Sun Microsystems, were developing a more powerful generation of desktop computers, dubbed “workstations,” to take advantage of that increasingly ubiquitous operating system on machines

comparable in power to the much older PDP-10.

似乎是命中注定一般，理查德的GNU系统没能赶在感恩节假期结束前发布。不过理查德依旧赶在1984年1月的时候做出了不小的成果，他自己也融入到了UNIX软件开发的世界之中。对于这位从ITS世界来的架构师来说，在UNIX中设计软件似乎更像是设计城郊购物中心，而非设计紫禁城皇宫。即便如此，设计成类UNIX系统依旧有它的优势。ITS的确强大，但仍有一处致命缺陷：麻省理工学院的黑客们创造的ITS系统，是专门针对当时强大的DEC PDP-10计算机设计。可到了八十年代，人工智能实验室的管理员决定淘汰掉实验室的PDP-10计算机。当年辉煌一时的ITS系统也因此成了一座无人的鬼城。而UNIX的设计则与此不同。它的设计非常强调移植性，它并不依赖于某个特定硬件环境。因此，面对硬件更替，UNIX则毫无压力。UNIX最初是由当时AT&T贝尔实验室的几个年轻科学家创造。之后，这套系统被大家纷纷传阅，不断改进。在资金并不富裕的学术界，UNIX系统甚是流行。和麻省理工学院的同僚们不同，UNIX的作者可用的硬件资源非常有限。他们必须把软件设计得可以在各种良莠不齐的硬件上运行无阻。最初是针对PDP-11系统设计，PDP-11是DEC推出的另一种16位计算机。在麻省理工人工智能实验室的黑客们看来，它顶多只能用来跑些小程序。之后，UNIX又陆续开始支持32位计算机，比如VAX11/780。到了1983年，一些公司开始推出更强大的一代被称作“工作站”的桌面计算机。其中以Sun公司产品最为引人注目。这些计算机运行UNIX，有着更小的体积，却和当年的PDP-10性能相当。

To facilitate portability, the developers of Unix had put an extra layer of abstraction between the software and the machine. Rather than writing it in the instructions of a specific machine type – as the AI Lab hackers had done with ITS and the PDP-10 – Unix developers wrote in a high-level language, called C. Focusing more on the interlocking interfaces and specifications that held the operating system’s many subcomponents together, rather than the actual components themselves, they created a system that could be quickly modified to run on any machine. If a user disliked a certain component, the interface specifications made it possible to pull out an individual subcomponent and either fix it or replace it with something better. Simply put, the Unix approach promoted flexibility and economy, hence its rapid adoption.

为了增强可移植能力，UNIX的开发者在软件和计算机之间，加入了一个抽象层。和人工智能实验室的ITS系统不同，UNIX并非使用特定于某个硬件平台的指令集开发。它的开发者们创造了一种被称为“C语言”的更抽象的编程语言，借此屏蔽了底层的硬件差异。开发者们于是可以放眼全局，集中设计各个组件之

间的协调机制，设计组件的接口，而不必把精力分散到每个组件的开发和移植上。他们由此创造了一套系统，可以轻易地移植到几乎各种计算机上。如果某个用户对哪个组件并不满意，他们可以按照定义好的接口，修改现有组件，甚至重新开发一个同样功能的组件，再把改进的组件放回原位，一切依旧会运行正常。简而言之，UNIX的设计具备极大的移植性，也促进了计算机这个市场。这一切，也带来了UNIX的蓬勃发展。

Stallman's decision to start developing the GNU system was triggered by the end of the ITS system that the AI Lab hackers had nurtured for so long. The demise of ITS, and the AI Lab hacker community which had sustained it, had been a traumatic blow to Stallman. If the Xerox laser printer episode had taught him to recognize the injustice of proprietary software, the community's death forced him to choose between surrendering to proprietary software and opposing it.

理查德之所以决定开发一套名为GNU的类UNIX系统，是因为人工智能实验室已经停止使用ITS了。而整个实验室的黑客文化，也伴随这ITS的消亡，渐渐分崩离析。这个改变对理查德来说，可谓打击重大。施乐公司打印机事件，让他意识到了专有软件的不义。而实验室的黑客社区几乎解散，则让理查德必须面对抉择：要么在专有软件面前投降，要么起身反抗它。

Like the software code that composed it, the roots of ITS' demise stretched way back. By 1980, most of the lab's hackers were working on developing the Lisp Machine and its operating system.

ITS的消亡过程历经了很长时间。1980年的时候，实验室里的大部分黑客都在开发Lisp机（Lisp Machine）和运行其上的操作系统。

Created by artificial-intelligence research pioneer John McCarthy, a MIT artificial-intelligence researcher during the late 1950s, Lisp is an elegant language, well-suited for writing complex programs to operate on data with irregular structure. The language's name is a shortened version of LISt Processing. Following McCarthy's departure to the Stanford Artificial Intelligence Laboratory, MIT hackers refined the language into a local dialect dubbed MACLISP. The "MAC" stood for Project MAC, the DARPA-funded research project that gave birth to the AI Lab and the Laboratory for Computer Science. Led by AI Lab arch-hacker Richard Greenblatt, the AI Lab hackers during the late 1970s designed a computer specialized for running Lisp efficiently and conveniently, the Lisp Machine, then developed an entire Lisp-

based operating system for it.

Lisp是一种非常优雅的编程语言。它最初由人工智能领域的先驱，约翰·麦卡锡（John McCarthy）发明。二十世纪五十年代，他曾是麻省理工学院人工智能领域的科学家。他发明的Lisp语言，非常适合编写复杂程序，来处理不具备很好结构的数据。Lisp这个名字来自LISt Processing，即链表处理。之后，约翰·麦卡锡离开了麻省理工学院，去了斯坦福大学的人工智能实验室。麻省理工学院的黑客们则改进了Lisp语言，并创造了一个Lisp方言，名为MACLISP。其中的“MAC”，指的是“MAC项目”。MAC项目是一个由美国国防部高等研究计划局（DARPA）资助的项目。借助这个项目，诞生了如今的人工智能实验室。整个实验室由黑客理查德·格林布拉特（Richard Greenblatt）领导。在七十年代末期，他们设计出了专门用来高校地执行Lisp程序的计算机，命名为Lisp机。接着，开发了一整套基于Lisp的操作系统。

By 1980, two rival groups of hackers had formed two companies to manufacture and sell copies of the Lisp Machine. Greenblatt started Lisp Machines Incorporated. He planned to avoid outside investment and make a “hacker company.” Most of the hackers joined Symbolics, a conventional startup. In 1982 they entirely ceased to work at MIT.

到了1980年，两组互相竞争的黑客各自成立了两家公司，分别制造和销售Lips机。理查德·格林布拉特成立了“Lisp机公司”（Lisp Machines Incorporated）。他试图避免引入外界的投资，创造一个真正的“黑客公司”。另外一大部分黑客们，则加入了名为Symbolics的传统创业公司。到了1982年，这些黑客则完全放下了在麻省理工学院的工作。

With few hackers left to mind the shop, programs and machines took longer to fix – or were not fixed at all. Even worse, Stallman says, the lab began to undergo a “demographic change.” The hackers who had once formed a vocal minority within the AI Lab were almost gone while “the professors and the students who didn’t really love the [PDP-10] were just as numerous as before.”

随着黑客们的离去，实验室的程序和机器要么没人修，要么也要等上好久才有人打理。理查德·斯托曼说，更糟糕的是，实验室正经历了前所未有的“人事变动”。以前，黑客们虽说人少，却是实验室中很重要的一伙人。如今，黑客几乎绝迹，而“不喜欢PDP-10的教授和学生则依旧那么多。”

In 1982, the AI Lab received the replacement for its main computer, the PDP-10, which was over 12 years old. Digital’s current model, the Decsystem 20,

was compatible for user programs but would have required a drastic rewrite or “port” of ITS if hackers wanted to continue running the same operating system. Fearful that the lab had lost its critical mass of in-house programming talent, AI Lab faculty members pressed for Twenex, a commercial operating system developed by Digital. Outnumbered, the hackers had no choice but to comply.

1982年，人工智能实验室接到通知，要求替换那台已经服役了12年的PDP-11计算机。PDP-11曾是迪吉多公司（Digital Equipment Corporation，简称DEC）出品的16位计算机。如今迪吉多公司的主打产品是Decsystem 20。对于应用程序来说，Decsystem 20和PDP-11兼容。但是如果想要在上面运行ITS这样的操作系统，则需要投入大量劳力，把系统从PDP-11移植到Decsystem 20上。如今的人工智能实验室早已物是人非，实验室中的编程能手几乎都已走光。而实验室中的一些教授则大力鼓吹Twenex系统，它是一款由迪吉多开发的商业操作系统。如今的黑客们在人数上占了劣势，只好将就着用Twenex系统。

.....

``Without hackers to maintain the system, [faculty members] said, ‘We’re going to have a disaster; we must have commercial software,’

” Stallman would recall a few years later. “They said, ‘We can expect the company to maintain it.’ It proved that they were utterly wrong, but that’s what they did.”

几年之后，理查德回忆起那时的情景：“那些教授们说：‘没有足够的黑客们来维护ITS系统，我们将会面临各种灾难。要避免这些发生，我们只能投靠商业软件。我们可以让商业公司来提供维护。’后来的事实证明，他们的这番论调大错特错。可当时他们的确这么做了。“

At first, hackers viewed the Twenex system as yet another authoritarian symbol begging to be subverted. The system’s name itself was a protest. Officially dubbed TOPS-20 by DEC, it was named as a successor to TOPS-10, a proprietary operating system DEC distributed for the PDP-10. But TOPS-20 was not based on TOPS-10. It was derived from the Tenex system which Bolt Beranek Newman had developed for the PDP-10. Stallman, the hacker who coined the Twenex term, says he came up with the name as a way to avoid using the TOPS-20 name. “The system was far from tops, so there was no

way I was going to call it that," Stallman recalls. "So I decided to insert a 'w' in the Tenex name and call it Twenex."

一开始，黑客们觉得Twenex只不过又是一个专制的标志。和之前那些专制标志一样，只要把它推翻，就万事大吉了。这个系统从名字看，就像是对黑客们的挑衅：迪吉多公司对这个系统的官方名字是TOPS-20，意为"顶尖20"。它是TOPS-10的后继产品，而TOPS-10则是迪吉多当年给PDP-10配备的标准系统。不过TOPS-20并非是在TOPS-10基础上做的开发。它的主要代码都继承自BBN公司（Bolt, Beranek and Newman）为PDP-10开发的Tenex系统。理查德则由此把TOPS-20称作Twenex，因为他实在不想用TOPS-20这个名字。他说："这个系统可根本算不上'顶尖'，我说什么也不会这么夸它。所以我在Tenex里添了个字母w，把它叫做Twenex。"

The machine that ran the Twenex/TOPS-20 system had its own derisive nickname: Oz. According to one hacker legend, the machine got its nickname because it required a smaller PDP-11 machine to power its terminal. One hacker, upon viewing the KL-10-PDP-11 setup for the first time, likened it to the wizard's bombastic onscreen introduction in the Wizard of Oz. "I am the great and powerful Oz," the hacker intoned. "Pay no attention to the PDP-11 behind that console."

在黑客界，人们把跑着TOPS系统的Decsystem计算机戏称作Oz，意为"奥兹国"。奥兹国这名字源自《绿野仙踪》这部小说。因为在黑客界，人们都说这样一台计算机需要有个PDP-11作为终端机才能运行。于是，人们看到这台计算机连着PDP-11作为终端，就不禁想到了《绿野仙踪》中那位吹牛皮的奥兹国大法师。黑客们常戏谑道："我是伟大全能的奥兹国大法师！我可没看到身后那台PDP-11。"

If hackers laughed when they first encountered the KL-10, their laughter quickly died when they encountered Twenex. Not only did Twenex boast built-in security, but the system's software engineers had designed the tools and applications with the security system in mind. What once had been a cat-and-mouse game over passwords in the case of the Laboratory for Computer Science's security system, now became an out-and-out battle over system management. System administrators argued that without security, the Oz system was more prone to accidental crashes. Hackers argued that crashes could be better prevented by overhauling the source code. Unfortunately, the number of hackers with the time and inclination to perform this sort of overhaul

had dwindled to the point that the system-administrator argument prevailed.

黑客们当年第一次看到Decsystem的时候也许还会借机嘲笑两句，可很快他们就在Twenex面前笑不起来了。Twenex系统增进了系统的安全强度，而且Twenex上的软件也都考虑了各种安全措施。当年，黑客们试图避免设置任何密码和安全设施，那些行为就好像在和管理员玩猫捉老鼠的游戏。而如今，他们面对的，则是一场彻头彻尾的战斗了。系统管理员辩解说，如果没有这些安全设施，Oz系统就随时都会面临崩溃。而黑客们则辩解说，避免崩溃的最好办法，就是公开程序的源代码。遗憾的是，黑客人数上不占优，而且大多数黑客也逐渐失去了当年的那份果断。最终，系统管理员还是胜利了。

The initial policy was that any lab member could have the “wheel privilege” to bypass security restrictions. But anyone who had the “wheel privilege” could take it away from anyone else, who would then be powerless to restore it. This state of affairs tempted a small group of hackers to try to seize total control by canceling the “wheel privilege” for all but themselves.

一开始的政策是，任何一个人工智能实验室的成员，都拥有一个“特权车服务”（wheel privilege），可以用来绕过系统的安全限制。可是，任何有“特权车”的人，都有权吊销别人的“特权车”。这样一来，就促使一小撮黑客总是尝试吊销其他人的特权，借此掌控全局。

Cadging passwords, and applying the debugger during startup, Stallman successfully foiled these attempts. After the second foiled “*coup d’état*,” Stallman issued an alert to all the AI Lab personnel.

见得此状，理查德破解了密码，利用开机时的调试器，成功破坏了几个这样企图夺权的阴谋。在第二轮“政变”之后，理查德向全体人工智能实验室的人发出了警告信。

“There has been another attempt to seize power,” Stallman wrote. “So far, the aristocratic forces have been defeated.” To protect his identity, Stallman signed the message “Radio Free OZ.”

信中写道：“还有另外一波强大的力量，企图剥夺我们的权利。不过现在，这些官僚们已经暂时被我们打败了。“这是封匿名信，信底的签名是“自由奥兹国电台”。

The disguise was a thin one at best. By 1982, Stallman’s aversion to passwords and secrecy had become so well known that users outside the AI

Laboratory were using his account from around the ARPAnet – the research-funded computer network that would serve as a foundation for today's Internet. One such "tourist" during the early 1980s was Don Hopkins, a California programmer who learned through the hacking grapevine that all an outsider needed to do to gain access to MIT's vaunted ITS system was to log in under the initials RMS and enter the same three-letter monogram when the system requested a password.

不过，所谓的匿名信并没把理查德挡在幕后。到了1982年，理查德关于密码和安全设施的抗议早就内外闻名。以至于有不少人工智能实验室以外的人，通过ARPAnet网络，使用理查德的登录帐号，访问人工智能实验室的计算机。他们当初使用的ARPAnet，是如今互联网的雏形。它曾是一个研究项目，旨在构造一个大型计算机网络。这个网络之后不断发展，最终成了今天大家见到的互联网。唐·霍普金斯（Don Hopkins）在八十年代是加利福尼亚州的一名程序员，他当年就曾利用理查德的帐号访问人工智能实验室的计算机。他当年从黑客圈的小道消息里听说，想要使用麻省理工学院大名鼎鼎的ITS系统，只需要使用一个简单的用户名和密码登录就可以：用户名，RMS；密码，RMS。

"I'm eternally grateful that MIT let me and many other people use their computers for free," says Hopkins. "It meant a lot to many people."

霍普金斯说："麻省理工学院让我和很多其他的人免费使用他们的计算机，这让我受益终生。这在当时，对大家来说可是份厚礼。"

This so-called "tourist" policy, which had been openly tolerated by MIT management during the ITS years, fell by the wayside when Oz became the lab's primary link to the ARPAnet. At first, Stallman continued his policy of repeating his login ID as a password so outside users could have access through his account. Over time, however, Oz's fragility prompted administrators to bar outsiders who, through sheer accident or malicious intent, might bring down the system. When those same administrators eventually demanded that Stallman stop publishing his password, Stallman, citing personal ethics, instead ceased using the Oz system altogether.

这种允许"游客"的政策，在ITS时代还能被麻省理工学院的管理人员容忍。可等到Oz成为实验室连接ARPAnet的主节点后，这行为就被逐渐禁止了。一开始，理查德依旧使用原来简单的用户名和密码，这样外部人员还是可以访问实验室的计算机。但是脆弱的Oz禁不起这么折腾，很快管理员就禁止了外部人员的访问。因为有些访客会有意无意地把整个系统搞垮。再后来，管理员们严重警告

了理查德公开用户名和密码的行为。对此，理查德并没有放弃使用Oz系统，而是在回应中强调了自己的底线。

“[When] passwords first appeared at the MIT AI Lab I [decided] to follow my belief that there should be no passwords,” Stallman would later say. “Because I don’t believe that it’s really desirable to have security on a computer, I shouldn’t be willing to help uphold the security regime.”

理查德之后说：“当年人工智能实验室第一次要求设置密码，我依旧坚持自己的原则：计算机上不该有密码。我也因此坚决不会帮助他们维护一个充满安全设施的计算机系统。”

Stallman’s refusal to bow before the great and powerful Oz symbolized the growing tension between hackers and AI Lab management during the early 1980s. This tension paled in comparison to the conflict that raged within the hacker community itself. By the time the Decsystem 20 arrived, the hacker community was divided into two camps, LMI and Symbolics.

八十年代早期，理查德的这次抗议之后，系统管理员掌控下的伟大全能的Oz系统和实验室的黑客之间，关系更加紧张。不过随着时间的流逝，这种紧张关系反而被黑客圈子内部的矛盾逐渐取代。在Decsystem 20来到实验室的时候，实验室的黑客圈子已经俨然分成了两派：LMI派和Symbolics派。

Symbolics, with its outside investment, recruited various AI Lab hackers and set some of them working on improving parts of the Lisp Machine operating system outside the auspices of the AI Lab. By the end of 1980, the company had hired 14 AI Lab staffers as part-time consultants to develop its version of the Lisp Machine. The remaining few, apart from Stallman, worked for LMI. Stallman, preferring the unpressured life at the AI Lab and not wishing to take a side, chose to join neither company.

LMI和Symbolics分别是两家做Lisp机的计算机公司。Symbolics得到了一些外部的投资，它招募了很多人工智能实验室的黑客，把其中的一部分黑客安排去改进Symbolics的Lisp机的操作系统。八十年代末期，这家公司已经吸纳了人工智能实验室的14位黑客作为兼职顾问，负责开发他们的Lisp机。剩下的黑客们，除了理查德以外，则都在LMI公司就职。理查德则更享受人工智能实验室里没有压力的工作环境，更不想把自己划进哪个阵营。所以他哪个公司都没加入。

At first, the other hackers continued spending some of their time at MIT, and

contributed to MIT's Lisp Machine operating system. Both LMI and Symbolics had licensed this code from MIT. The license required them to return their changes to MIT, but did not require them to let MIT redistribute these changes. However, through 1981 they adhered to a gentleman's agreement to permit that, so all their system improvements were included in the MIT version and thus shared with all Lisp Machine users. This situation allowed those still at MIT to remain neutral.

一开始，这些在公司干活的黑客们还会抽出些时间，在麻省理工学院继续做些工作，也会给麻省理工学院的Lisp机的操作系统贡献些代码。LMI和Symbolics的Lisp机操作系统都是从麻省理工学院的Lisp机操作系统上衍生而来。在使用许可证上，麻省理工学院要求两家公司必须允许麻省理工学院使用它们的操作系统。但是，这两家公司可以禁止麻省理工学院再发布他们的产品。不过，1981年间，两家公司之间倒是有个不成文的君子协议，他们都允许麻省理工学院使用并发布两家公司的代码。这样，麻省理工学院的Lisp机操作系统就包含了来自外部公司的各种改进，而基于许可证，改进后的系统代码有可以继续流入各家公司。这相当于公司之间可以共享代码。如此，那些同时在麻省理工学院和公司工作的黑客们，就可以继续保持中立。

On March 16, 1982, a date Stallman remembers well because it was his birthday, Symbolics executives ended the gentleman's agreement. The motive was to attack LMI. LMI had fewer hackers, and fewer staff in general, so the Symbolics executives thought that LMI was getting the main benefit of sharing the system improvements. By ending the sharing of system code, they hoped to wipe out LMI. So they decided to enforce the letter of the license. Instead of contributing their improvements to the MIT version of the system, which LMI could use, they provided MIT with a copy of the Symbolics version of the system for users at MIT to run. Anyone using it would provide the service of testing only to Symbolics, and if he made improvements, most likely they too would only be useful for Symbolics.

1982年3月16日，理查德清晰地记得这一天，那是他的生日。Symbolics的主管们决定不再遵循当初那君子协议。他们禁止麻省理工学院发布包含Symbolics公司的代码。此举旨在打击LMI。因为在LMI公司就职的黑客比较少，所以Symbolics的主管们觉得LMI从共享的代码中获益。切断代码流通的途径，Symbolics企图借此把LMI赶出市场。他们决定执行许可证上的权利。为了不让LMI获得自己的代码，他们仅允许麻省理工学院的学生运行他们的操作系统。借此，他们可以间接地让学院中的用户为自己做测试。同时，如果哪个用户对系

统做了任何改进，则也会为Symbolics公司所用。

As the person responsible (with help from Greenblatt for the first couple of months) for keeping up the lab's Lisp Machine system, Stallman was incensed. The Symbolics hackers had left the system code with hundreds of half-made changes that caused errors. Viewing this announcement as an "ultimatum," he retaliated by disconnecting Symbolics' microwave communications link to the laboratory. He then vowed never to work on a Symbolics machine, and pledged to continue the development of MIT's system so as to defend LMI from Symbolics. "The way I saw it, the AI Lab was a neutral country, like Belgium in World War II," Stallman says. "If Germany invades Belgium, Belgium declares war on Germany and sides with Britain and France."

理查德在那时候负责维护实验室中的Lisp机，最开始的几个月还多亏了LMI公司的创始人，格林布拉特（Greenblatt）的指点帮助。当下，Symbolics的做法可是惹恼了理查德。那些在Symbolics公司工作的黑客们都曾给实验室的Lisp机系统贡献过代码，如今，还有很多错误和bug遗留在这些代码中，甚至有些特性还都是半成品。如今，Symbolics高层的做法无疑相当于下了"最后通牒"，令身在Symbolics的黑客们无法改进麻省理工学院的Lisp机系统。理查德也发起了反攻，他切断了连接实验室和Symbolics公司的微波通信信道，发誓再也不会在Symbolics生产的机器上工作。并且决心要继续完善麻省理工学院的Lisp机系统上，以此帮助LMI，打击Symbolics。理查德回忆说："当初我觉得，人工智能实验室是个中立国，就好像二战时的比利时一样。如果德国入侵比利时，比利时就要对德宣战，就得和英法一个阵营。"

When Symbolics executives noticed that their latest features were still appearing in the MIT Lisp Machine system and, by extension, the LMI Lisp machine, they were not pleased. Stallman knew what copyright law required, and was rewriting the features from scratch. He took advantage of the opportunity to read the source code Symbolics supplied to MIT, so as to understand the problems and fixes, and then made sure to write his changes in a totally different way. But the Symbolics executives didn't believe this. They installed a "spy" program on Stallman's computer terminal looking for evidence against him. However, when they took their case to MIT administration, around the start of 1983, they had little evidence to present: a dozen places in the sources where both versions had been changed and appeared similar.

Symbolics公司的主管们逐渐意识到，自己公司操作系统的最新特性，总会被理查德克隆到麻省理工学院的Lisp机系统上。这让他们甚为不满。理查德知道版权法这么个东西，所以他为学院的系统添加的特性，都是自己从头开始写的。他身在人工智能实验室，可以读到Symbolics操作系统提供给实验室的源代码。他会先读读Symbolics的代码，理解要解决的问题和解决方案，最后再自己重新写一遍，确保和Symbolics的实现完全不同。可Symbolics公司的主管们却不管这个。他们在理查德使用的计算机终端上安装了间谍软件，企图抓住理查德剽窃的证据。到了1983年年初，他们把这件事情告到麻省理工学院的管理层。可依旧拿不出多少像样的证据，只有一些看似类似的代码片段。

When the AI Lab administrators showed Stallman Symbolics' supposed evidence, he refuted it, showing that the similarities were actually held over from before the fork. Then he turned the logic around: if, after the thousands of lines he had written, Symbolics could produce no better evidence than this, it demonstrated that Stallman's diligent efforts to avoid copying were effective. The AI Lab approved Stallman's work, which he continued till the end of 1983.

人工智能实验室的主管找来理查德，给他看了Symbolics公司指责他的证据。理查德一一否认，他说这些相似的代码都是Symbolics诞生之前，在学院Lisp系统里就有的。之后，理查德话锋一转：他自己已经克隆了Symbolics公司的众多特性，在那几千行的代码中，如今Symbolics公司却只能提供这么点证据来证明理查德剽窃，这恰恰说明理查德的确没有抄袭Symbolics的代码。人工智能实验室最终承认了理查德并没有剽窃，他也就继续开发到1983年。

Stallman did make a change in his practices, though. "Just to be ultra safe, I no longer read their source code [for new features and major changes]. I used only the documentation and wrote the code from that." For the biggest new features, rather than wait for Symbolics to release documentation, he designed them on his own; later, when the Symbolics documentation appeared, he added compatibility with Symbolics' interface for the feature. Then he read Symbolics' source code changes to find minor bugs they had fixed, and fixed each of them differently.

不过这件事倒是确实给理查德提了醒。"以防万一，对于新的特性和重大改变，我不再参考Symbolics的代码，而只是参考他们提供的用户文档来做开发。"对于一些重大的新特性，理查德则往往在文档发布之前，就着手设计开发。等到Symbolics公司发布文档，他再对代码修改，以便兼容Symbolics的接口。之后，如果Symbolics提供了补丁，他再阅读补丁的代码，以便确认自己的实现中

是否存在类似bug。如果存在，则尝试用不同的实现来解决。

The experience solidified Stallman's resolve. As Stallman designed replacements for Symbolics' new features, he also enlisted members of the AI Lab to keep using the MIT system, so as to provide a continuous stream of bug reports. MIT continued giving LMI direct access to the changes. "I was going to punish Symbolics if it was the last thing I did," Stallman says. Such statements are revealing. Not only do they shed light on Stallman's nonpacifist nature, they also reflect the intense level of emotion triggered by the conflict.

整个过程也坚定了理查德的决心。理查德的开发，逐渐把人工智能实验室中的成员拉回到麻省理工学院的Lisp机系统上。这些用户也为理查德持续地提供错误报告。麻省理工学院继续允许LMI直接访问学院Lisp机系统的代码。理查德说："我当初就觉得，我要是这辈子只能再做一件事，那我也要惩罚Symbolics。"这话说得甚是坦白。显然，理查德不是个和平主义者。随着矛盾的升级，给双方带来的情感变化也逐渐加大。

The level of despair owed much to what Stallman viewed as the "destruction" of his "home" – i.e., the demise of the AI Lab's close-knit hacker subculture. In a later email interview with Levy, Stallman would liken himself to the historical figure Ishi, the last surviving member of the Yahi, a Pacific Northwest tribe wiped out during the Indian wars of the 1860s and 1870s. The analogy casts Stallman's survival in epic, almost mythical, terms. The hackers who worked for Symbolics saw it differently. Instead of seeing Symbolics as an exterminating force, many of Stallman's colleagues saw it as a belated bid for relevance. In commercializing the Lisp Machine, the company pushed hacker principles of engineer-driven software design out of the ivory-tower confines of the AI Lab and into the corporate marketplace where manager-driven design principles held sway. Rather than viewing Stallman as a holdout, many hackers saw him as the representative of an obsolete practice.

由此带来的绝望感，让理查德又有了家园被毁的感受。人心散了，人工智能实验室的黑客文化也逐渐褪去。之后在与史蒂芬·李维的采访邮件中，理查德把自己形容为Ishi——加利福尼亚州最后一位Yahi族印第安人。这样的说法倒是为理查德的经历凭添了几分史诗色彩。不过在Symbolics工作的各位黑客们则另有看法。他们并非觉得Symbolics好像要上灭下绝。很多理查德当年的同事，都觉得Symbolics倒是迟到的正义。在把Lisp机商业化的过程中，Symbolics把黑客文化中的"工程师做主"的原则从象牙塔中带入了商业公司里，让公司里不再有"外

行领导内行“的现象。其他黑客们也没觉得理查德是在捍卫什么，而是觉得他只是代表着一种过时的想法而已。

Personal hostilities also affected the situation. Even before Symbolics hired away most of the AI Lab's hacker staff, Stallman says many of the hackers who later joined Symbolics were shunning him. “I was no longer getting invited to go to Chinatown,” Stallman recalls. “The custom started by Greenblatt was that if you went out to dinner, you went around or sent a message asking anybody at the lab if they also wanted to go. Sometime around 1980-1981, I stopped getting asked. They were not only not inviting me, but one person later confessed that he had been pressured to lie to me to keep their going away to dinner without me a secret.”

私下的过结也让事态趋于严重。理查德说，Symbolics公司雇走大量黑客之前，理查德就觉得很多黑客都会有意躲避自己。这些黑客之后大部分都加入了Symbolics公司。理查德回忆：“他们很少问我去不去唐人街吃饭了。当年从格林布拉特（Greenblatt）开始，实验室就有个传统：谁要是去吃饭，谁就发个消息或者亲自走一圈，问问实验室里还有谁想一起去。可大概在1980年到1981年间，逐渐就没什么人问我了。他们不邀请我倒也罢了。之后，他们之中有人才告诉我，当年他被人告之，不许他跟我说大家一起出去吃饭故意不带我这个事。”

Although Stallman felt hurt by this petty form of ostracism, there was nothing to be done about it. The Symbolics ultimatum changed the matter from a personal rejection to a broader injustice. When Symbolics excluded its source changes from redistribution, as a means to defeat its rival, Stallman determined to thwart Symbolics' goal. By holing up in his MIT offices and writing equivalents for each new software feature and fix, he gave users of the MIT system, including LMI customers, access to the same features as Symbolics users.

虽然理查德对这种排斥行为非常反感，但并没对此有什么行动。可Symbolics的最后通牒则改变了一切，这事从此也不再是私人恩怨了。当Symbolics公司不再给用户提供源代码的时候，理查德决心要对此进行反击。他日夜坐在他的办公室里，在麻省理工学院的Lisp机系统上，实现了Symbolics提供的各种新功能和bug修正。他把修改后的版本代码分发给各个用户，包括LMI公司的客户。这样，LMI公司的客户也可以拥有Symbolics系统类似的功能。

It also guaranteed Stallman's legendary status within the hacker community.

Already renowned for his work with Emacs, Stallman's ability to match the output of an entire team of Symbolics programmers – a team that included more than a few legendary hackers itself – still stands as one of the major human accomplishments of the Information Age, or of any age for that matter. Dubbing it a "master hack" and Stallman himself a "virtual John Henry of computer code," author Steven Levy notes that many of his Symbolics-employed rivals had no choice but to pay their idealistic former comrade grudging respect. Levy quotes Bill Gosper, a hacker who eventually went to work for Symbolics in the company's Palo Alto office, expressing amazement over Stallman's output during this period:

这也为理查德在黑客圈子里凭添了几分名气。理查德早就因为Emacs而名声大噪。如今，他一个人单枪匹马，对抗整个Symbolics公司的开发团队，而且这团队之中还尽是各色传奇黑客。理查德的这一行为本身，就足以成为信息时代的一段传奇。史蒂芬·李维在《黑客》一书中称这一行为是"黑客杰作"；并把理查德比作现代的约翰·亨利（John Henry）。史蒂芬·李维在书中说，很多Symbolics公司工作的黑客们都钦佩理查德的能力。他引述了比尔·高斯伯（Bill Gosper），他曾在Symbolics的帕罗奥多（Palo Alto）分部工作，对于理查德那段时期的产出，比尔·高斯伯甚是惊奇：

> I can see something Stallman wrote, and I might decide it was bad (probably not, but somebody could convince me it was bad), and I would still say, "But wait a minute – Stallman doesn't have anybody to argue with all night over there. He's working alone! It's incredible anyone could do this alone!"
>
> 我读过理查德那段时期的一些代码，有些代码写的并不怎么出色（至少在我看来），但我还是要说："且慢，他就只有一个人，他没法和别人整晚探讨，他是单枪匹马啊！一个人能完成如此大量的工作，简直是逆天了！"

For Stallman, the months spent playing catch up with Symbolics evoke a mixture of pride and profound sadness. As a dyed-in-the-wool liberal whose father had served in World War II, Stallman is no pacifist. In many ways, the Symbolics war offered the rite of passage toward which Stallman had been careening ever since joining the AI Lab staff a decade before. At the same time, however, it coincided with the traumatic destruction of the AI Lab hacker culture that had nurtured Stallman since his teenage years. One day, while taking a break from writing code, Stallman experienced a traumatic moment

passing through the lab's equipment room. There, Stallman encountered the hulking, unused frame of the PDP-10 machine. Startled by the dormant lights, lights that once actively blinked out a silent code indicating the status of the internal program, Stallman says the emotional impact was not unlike coming across a beloved family member's well-preserved corpse.

对于理查德来说，这段时期和Symbolics的竞争，既让他既感骄傲，又让他感到了深深的难过。理查德是个彻头彻尾的自由主义者，自己的父亲也曾在二战期间为自由而战。他不会期盼勉强得来的和平。在加入人工智能实验室之前，他本就心向自由。如今与Symbolics之间燃起的硝烟，则又把他推向了一个极端。无巧不成书，这次的矛盾，恰恰发生在人工智能实验室黑客文化消退的时候。理查德曾被这文化滋润，如今眼见它消失殆尽，其中心酸，难以言表。曾有一日，在他编程的间歇，偶尔路过实验室的设备间。眼见那台曾在实验室服役的PDP-10摆放在房间之中，无人问津。当年那几个忙碌闪烁的状态灯，如今也黯淡无光。往事片段，涌上心头。眼看着这台几十年前的计算机，仿佛看着家中亲人，静静的躺放在那里，魂归西天。

"I started crying right there in the machine room," he says. "Seeing the machine there, dead, with nobody left to fix it, it all drove home how completely my community had been destroyed."

"我当时泪如泉涌。"理查德说，"它就在那儿，可却没人关心，无人维护。眼前的一切在告诉我，我们当年的那个黑客大家庭，早就不复存在了。"

Stallman would have little opportunity to mourn. The Lisp Machine, despite all the furor it invoked and all the labor that had gone into making it, was merely a sideshow to the large battles in the technology marketplace. The relentless pace of computer miniaturization was bringing in newer, more powerful microprocessors that would soon incorporate the machine's hardware and software capabilities like a modern metropolis swallowing up an ancient desert village.

然而，留给理查德感伤怀古的时间却并不多了。无论曾投入过多少人力物力，整个Lisp机产业却仅仅是昙花一现。计算机小型化的脚步一步不停。带来了更新，更强大的微处理器。这一波趋势，如同风卷残云般，将其他竞争者一举赶出主流市场。

Riding atop this microprocessor wave were hundreds – thousands – of proprietary software programs, each protected by a patchwork of user licenses

and nondisclosure agreements that made it impossible for hackers to review or share source code. The licenses were crude and ill-fitting, but by 1983 they had become strong enough to satisfy the courts and scare away would-be interlopers. Software, once a form of garnish most hardware companies gave away to make their expensive computer systems more flavorful, was quickly becoming the main dish. In their increasing hunger for new games and features, users were putting aside the traditional demand to review the recipe after every meal.

伴随这波风潮而来的，是成千上万的专有软件。每个专有软件，都带着自己的使用许可证和保密协议扑向用户。令其他黑客无法触碰其中的代码。很多软件使用许可证对用户粗暴无礼。但是在1983年，这些专有软件依旧成了主流，填补了市场。也让潜在的竞争对手望而却步。软件，曾经一度只是各个硬件厂商的随机赠品，如今却成了业界新宠。当下，用户们开始极度索要新软件，新功能；而至于是否可以知道软件内部究竟做了什么，则甚少提及。

Nowhere was this state of affairs more evident than in the realm of personal computer systems. Companies such as Apple Computer and Commodore were minting fresh millionaires selling machines with built-in operating systems. Unaware of the hacker culture and its distaste for binary-only software, many of these users saw little need to protest when these companies failed to attach the accompanying source-code files. A few anarchic adherents of the hacker ethic helped propel that ethic into this new marketplace, but for the most part, the marketplace rewarded the programmers speedy enough to write new programs and savvy enough to write End User License Agreements to lock them up tight.

个人计算机的到来把这股潮流推向了巅峰。苹果，Commodore等公司跨入了百万富翁的行列。它们出售个人计算机，并在上面安装了自己公司的操作系统。这些计算机的用户们可不再像当年的黑客，他们并不关心软件的源代码，要是买来的软件不附带源代码，他们也不会大呼小叫。一些还恪守当年黑客信条的人，曾试图把黑客的这种传统带入个人计算机这个新兴市场。但无论如何，这个唯利是图的新兴市场推动着程序员写出更多的软件，也同时带来了更多的使用许可证。

One of the most notorious of these programmers was Bill Gates, a Harvard dropout two years Stallman’s junior. Although Stallman didn’t know it at the time, seven years before sending out his message to thenet.unix-wizards

newsgroup, Gates, a budding entrepreneur and general partner with the Albuquerque-based software firm Micro-Soft, later spelled as Microsoft, had sent out his own open letter to the software-developer community. Written in response to the PC users copying Micro-Soft's software programs, Gates' "Open Letter to Hobbyists" had excoriated the notion of communal software development.

这其中，最为著名的程序员恐怕要算比尔·盖茨了。这位哈佛的辍学生比理查德小两年入学哈佛，算是理查德的学弟。不过理查德当年并不认识他。1983年9月27日，理查德在thenet.unix-wizards新闻组上发布了GNU工程计划。而七年之前，比尔·盖茨在软件开发者的社区中发表了那封著名的公开信。当年，微软还是坐落在新墨西哥州，阿尔伯克基市的一家小公司。那时，曾有很多PC用户私下拷贝微软的软件。由此，才让比尔·盖茨发表了那封《致爱好者的公开信》。

"Who can afford to do professional work for nothing?" asked Gates. "What hobbyist can put three man-years into programming, finding all bugs, documenting his product, and distributing it for free?"

"有谁会在没有任何报酬的情况下来做这些专业工作？"比尔·盖茨在信中发问："什么样的爱好者可以为他的产品投入三人年的精力，开发完软件还得去发现错误，编写文档，最后还要免费发布他的产品？"

Although few hackers at the AI Lab saw the missive, Gates' 1976 letter nevertheless represented the changing attitude toward software both among commercial software companies and commercial software developers. Why treat software as a zero-cost commodity when the market said otherwise? As the 1970s gave way to the 1980s, selling software became more than a way to recoup costs; it became a political statement. At a time when the Reagan Administration was rushing to dismantle many of the federal regulations and spending programs that had been built up during the half century following the Great Depression, more than a few software programmers saw the hacker ethic as anticompetitive and, by extension, un-American. At best, it was a throwback to the anticorporate attitudes of the late 1960s and early 1970s. Like a Wall Street banker discovering an old tie-dyed shirt hiding between French-cuffed shirts and double-breasted suits, many computer programmers treated the hacker ethic as an embarrassing reminder of an idealistic age.

比尔·盖茨1976年2月发表了这封公开信，那会还没有几个人工智能实验室的黑客读过这信。可显然，这封信代表了商业软件公司和软件开发者对于软件态度

的转变。既然市场都这么说了，那还为什么把软件当作免费赠品呢？从七十年代走到八十年代，出售软件已经不再仅仅为了弥补开发的成本，它变成了一种政治宣言。当年，里根政府正在极力清除大萧条时期政府制定的各种避免竞争的限制，越来越多的程序员也把黑客文化视为一种反对竞争，乃至违背美国精神的东西。最起码，他们觉得所谓黑客精神，最多是股复古风，不过是六七十年代反对大型企业态度的延续，一种企盼回到理想年代的情结。

For a man who had spent the entire 1960s as a throwback to the 1950s, Stallman didn't mind living out of step with his peers. As a programmer used to working with the best machines and the best software, however, Stallman faced what he could only describe as a "stark moral choice": either swallow his ethical objection for "proprietary" software – the term Stallman and his fellow hackers used to describe any program that carried copyright terms or an end-user license that restricted copying and modification – or dedicate his life to building an alternate, nonproprietary system of software programs. After his two-year battle with Symbolics, Stallman felt confident enough to undertake the latter option. "I suppose I could have stopped working on computers altogether," Stallman says. "I had no special skills, but I'm sure I could have become a waiter. Not at a fancy restaurant, probably, but I could've been a waiter somewhere."

理查德曾生活在六十年代，却被当成是活在五十年代的人。所以所谓的复古，不赶潮流对他来说倒也没什么。可作为一个一度使用顶级计算机，顶级软件的程序员，面临着那些带有各种使用许可证，禁止用户随意拷贝或修改的专有软件，理查德却面临了一场艰难的道义选择：摆在眼前的有两条道，要么默许专有软件，然后忘掉自己当初对它的各种反对；要么倾其一生，创造一套独立于各种专有软件的系统。在和Symbolics公司对抗了两年之后，理查德自信有能力去走第二条路。他说："我倒是可以从此不再使用计算机。可这是我看家本事了。别的工作怕也做不来。没准可以做个餐厅服务员，不过也去不了什么大餐馆。"

Being a waiter – i.e., dropping out of programming altogether – would have meant completely giving up an activity, computer programming, that had given him so much pleasure. Looking back on his life since moving to Cambridge, Stallman finds it easy to identify lengthy periods when software programming provided the only pleasure. Rather than drop out, Stallman decided to stick it out.

要让理查德放弃他所热爱的计算机和编程，放弃从剑桥以来这个最大的乐趣来源，转头去做个餐厅服务员或是别的工作，理查德可绝对不能答应。他没有退却，决定主动出击。

An Atheist, Stallman rejects notions such as fate, karma, or a divine calling in life. Nevertheless, he does feel that the decision to shun proprietary software and build an operating system to help others do the same was a natural one. After all, it was Stallman's own personal combination of stubbornness, foresight, and coding virtuosity that led him to consider a fork in the road most others didn't know existed. In his article, "The GNU Project," Stallman affirms agreement with the ideals encapsulated in the words of the Jewish sage Hillel:

作为一个无神论者，理查德不愿把这一系列事件归咎于命运，因果或是缘分。他决定避免使用专有软件，并且创造完全自由的一套操作系统及其外围软件，来帮助其他用户获得自由。作出这个选择，对于他来说是再自然不过了。毕竟，凭借理查德内心的那份反抗精神，和他的智慧能力，他选择了一条少有人走的路。这条路，甚至还没被很多人发现。在他的一篇名为《GNU工程》的文章中，理查德曾引用了犹太先贤希肋耳（Hillel）的话来表明他的决心：

> If I am not for myself, who will be for me? If I am only for myself, what am I? If not now, when?
>
> 我不为我，谁人为我？我只为我，我为何物？此时不为，更待何时？

Speaking to audiences, Stallman avoids the religious route and expresses the decision in pragmatic terms. "I asked myself: what could I, an operating-system developer, do to improve the situation? It wasn't until I examined the question for a while that I realized an operating-system developer was exactly what was needed to solve the problem."

在外演讲时，理查德会避免借用任何宗教语言，而采用一些世俗的说法来描述。他说："我问自己，我，作为一个操作系统开发者，究竟可以做些什么，来改变现状？这个问题稍加思考，很快就可以看出来，能解决这个问题的恰恰需要是一个操作系统开发者。"

Once he recognized that, Stallman says, everything else "fell into place." In 1983, MIT was acquiring second-generation Lisp Machines from Symbolics, on which the MIT Lisp Machine system could not possibly run. Once most of the MIT machines were replaced, he would be unable to continue maintaining

that system effectively for lack of users' bug reports. He would have to stop. But he also wanted to stop. The MIT Lisp Machine system was not free software: even though users could get the source code, they could not redistribute it freely. Meanwhile, the goal of continuing the MIT system had already been achieved: LMI had survived and was developing software on its own.

理查德说，一旦想通了这点，其他的就"顺理成章了"。在1983年，麻省理工学院从Symbolics采购了他们公司的第二代的Lisp机。在这批Lisp机上，根本无法运行麻省理工学院自己的Lisp机操作系统。旧机器被这些新采购的机器替代，几乎没有什么人再使用麻省理工学院的Lisp机系统，也就没人来报告bug。理查德也就无法继续开发学院的Lisp机系统。他必须停下这个工作，不过这也恰恰是他想要做的。因为麻省理工学院的Lisp机系统并不是理查德心目中的自由软件：用户虽然可以获得系统的源代码，但是却不允许用户独立发行它。另一方面，麻省理工学院的系统也依旧继续开发着：LMI没有被Symbolics打倒，他们依旧在开发自己的软件。

Stallman didn't want to spend his whole life punishing those who had destroyed his old community. He wanted to build a new one. He decided to denounce software that would require him to compromise his ethical beliefs, and devote his life to the creation of programs that would make it easier for him and others to escape from it. Pledging to build a free software operating system "or die trying – of old age, of course," Stallman quips, he resigned from the MIT staff in January, 1984, to build GNU.

理查德可不希望花费自己的一生，来惩罚那些摧毁黑客社区的人。他要自己创建一个新的社区。他决定远离那些与他理想背道而驰的软件。他打算要投入自己的全部精力，开发出心中理想的软件，帮助自己和他人远离那些不讲道义的程序。他发誓要创造一个自由的操作系统，"或者为此而奋斗致死"。为此，他在1984年1月辞去了麻省理工学院的工作，专职去开发GNU系统。

The resignation distanced Stallman's work from the legal auspices of MIT. Still, Stallman had enough friends and allies within the AI Lab to continue using the facilities, and later his own office. He also had the ability to secure outside consulting gigs to underwrite the early stages of the GNU Project. In resigning from MIT, however, Stallman negated any debate about conflict of interest or Institute ownership of the software. The man whose early adulthood fear of social isolation had driven him deeper and deeper into the AI Lab's embrace

was now building a legal firewall between himself and that environment.

辞去工作可以让他在法律上与麻省理工学院断绝联系。不过，依然有很多人工智能实验室的朋友支持理查德的工作，让他得以继续使用实验室的设备。之后，甚至给他准备了一个独立的办公室。凭借理查德的能力，他在开发GNU系统之余，还兼职一些咨询师的职位，借此收入来支持GNU系统的继续开发。在从麻省理工学院辞职的过程中，他拒绝了任何机构拥有GNU系统。这位曾经一度畏惧社交活动的人，如今则把这种心理发挥到极致，让自己的社交障碍变成了一堵防火墙，隔离了各种可能的法律纠纷。

For the first few months, Stallman operated in isolation from the Unix community as well. Although his announcement to the net.unix-wizards group had attracted sympathetic responses, few volunteers signed on to join the crusade in its early stages.

在项目开发的最初几个月，理查德甚至也把自己隔离在UNIX社区之外。尽管他在net.unix-wizards新闻组发布的消息获得了很多同道的支持，但在最初阶段，只有很少人加入了这个工程。

“The community reaction was pretty much uniform,” recalls Rich Morin, leader of a Unix user group at the time. ``People said, ‘Oh, that’s a great idea. Show us your code. Show us it can be done.’

”

当年一个UNIX用户组的一位领导李奇·莫林（Rich Morin）说：“整个社区（对于理查德的项目）的反响比较一致。大家都会说，‘嗯，不错，好啊。是个挺棒的想法。期待你的代码，希望能看到它成功。’”

Aware that the job was enormous, Stallman decided to try to reuse existing free software wherever possible. So he began looking for existing free programs and tools that could be converted into GNU programs and tools. One of the first candidates was a compiler named VUCK, which converted programs written in the popular C programming language into machine-runnable code. Translated from the Dutch, the program’s acronym stood for the Free University Compiler Kit. Optimistic, Stallman asked the program’s author if the program was free. When the author informed him that the words “Free University” were a reference to the Vrije Universiteit in Amsterdam, and that the program was not free, Stallman was chagrined.

可说到底，这可确实是个超大的工程。理查德决定尽量采用已有的自由软件。于是他开始调研已有的自由或免费的程序，试着修改它们并把它们纳入GNU工程之中。第一个相中的，是一款名为VUCK的编译器。它可以把C语言编写的程序转换成可执行的机器码。VUCK是荷兰人开发的，荷兰语中，是“自由大学编译器”（Free University Compiler Kit）的缩写。带着几分期待，理查德联系VUCK的作者，询问它们这个软件是不是自由的。作者告诉理查德，所谓“自由大学”，指的是阿姆斯特丹自由大学（Vrije Universiteit Amsterdam）。虽然大学的名字里有“自由”二字，但并不意味着软件也是自由的。这番回复令理查德非常沮丧。

“He responded derisively, stating that the university was free but the compiler was not,” recalls Stallman. He had not only refused to help – he suggested Stallman drop his plan to develop GNU, and instead write some add-ons to boost sales of VUCK, in return for a share of the profits. “I therefore decided that my first program for the GNU Project would be a multi-language, multi-platform compiler.”

理查德回忆：“他的回复带着几分嘲讽。他说，这所大学是自由的，可编译器不是。”VUCK的作者非但没能给理查德提供什么帮助，反而建议理查德放弃开发GNU工程，并希望他转而为VUCK开发些插件，没准可以提高VUCK的销量，而且答应会按利润给理查德分成。理查德说：“由此，我决定，为GNU工程开发的第一个软件，将是一个多语言，跨平台的编译器。”

Instead of VUCK, Stallman found the Pastel compiler (“off-color Pascal”), written by programmers at Lawrence Livermore National Lab. According to what they said when they gave him a copy, the compiler was free to copy and modify. Unfortunately, the program was unsuitable for the job, because its memory requirements were enormous. It parsed the entire input file in core memory, then retained all the internal data until it finished compiling the file. On mainframe systems this design had been forgivable. On Unix systems it was a crippling barrier, since even 32-bit machines that ran Unix were often unable to provide so much memory to a program. Stallman made substantial progress at first, building a C-compatible frontend to the compiler and testing it on the larger Vax, whose system could handle large memory spaces. When he tried porting the system to the 68010, and investigated why it crashed, he discovered the memory size problem, and concluded he would have to build a totally new compiler from scratch. Stallman eventually did this, producing the GNU C Compiler or GCC. But it was not clear in 1984 what to do about the

compiler, so he decided to let those plans gel while turning his attention to other parts of GNU.

放弃了VUCK，理查德转向另一款名为Pastel的Pascal编译器。它由劳伦斯·利弗莫尔国家实验室（Lawrence Livermore National Lab）开发。Pastel的作者告诉理查德，这个编Lawrence Livermore National Lab译器可以自由传播和修改。遗憾的是，这款编译器占用内存太多，以至于无法在很多平台上运行。它会对整个文件做语法分析，然后把所有的状态都存入内存，等到编译结束，才释放所占用的空间。在当年，这种做法只能在大型机上行得通。而运行UNIX的机器，普遍没有那么大的内存。哪怕是运行32位UNIX系统的机器，也很少能提供如此大的内存给一个程序。一开始，理查德还取得了一些小进展。他给这个编译器做了一个C语言的前端，然后跑在内存较大些的Vax机器上。可是，当他把这个编译器移植到68010上的时候，编译器则总是崩溃。仔细查清原因，理查德发现是内存太小造成的。于是，他决定只能从头开发一个编译器。最终，他实现了这个编译器，并把它命名为GNU C编译器，缩写GCC。不过，在1984年，大家也还不知道这个编译器后续开发将会如何。于是，理查德一方面慢慢等着编译器相关的计划逐渐成型；另一方面着手开发GNU的其他组建。

In September of 1984, thus, Stallman began development of a GNU version of Emacs, the replacement for the program he had been supervising for a decade. Within the Unix community, the two native editor programs were vi, written by Sun Microsystems cofounder Bill Joy, and ed, written by Bell Labs scientist (and Unix cocreator) Ken Thompson. Both were useful and popular, but neither offered the endlessly expandable nature of Emacs.

在1984年9月，理查德开始开发GNU版本的Emacs编辑器。这个编辑器试图克隆并改进他主导了十几年的Emacs项目。在当年，UNIX社区里，有两大流行的编辑器：一款是Sun公司的创始人比尔·乔伊开发的vi编辑器；另一款是贝尔实验室的科学家，UNIX创始人肯·汤普森开发的ed编辑器。这两款编辑器都很不错，也很流行。但是它们都没有提供类似Emacs的扩展功能。

Looking back, Stallman says he didn’t view the decision in strategic terms. “I wanted an Emacs, and I had a good opportunity to develop one.”

如今回忆起来，理查德说，他决定为UNIX开发Emacs，并没从什么战略角度考虑。“我就是想在GNU系统里用Emacs。而且由我来开发它再合适不过了。”

Once again, Stallman had found existing code with which he hoped to save time. In writing a Unix version of Emacs, Stallman was soon following the

footsteps of Carnegie Mellon graduate student James Gosling, author of a C-based version dubbed Gosling Emacs or Gosmacs. Gosling's version of Emacs included an interpreter for a simplified offshoot of the Lisp language, called Mocklisp. Although Gosling had put Gosmacs under copyright and had sold the rights to UniPress, a privately held software company, Stallman received the assurances of a fellow developer who had participated in early Gosmacs development. According to the developer, Gosling, while a Ph.D. student at Carnegie Mellon, had given him permission by email to distribute his own version of Gosmacs in exchange for his contribution to the code.

又一次，他找到了一些现成代码，希望能借此节省些时间。在开发UNIX版的Emacs的过程中，理查德很快找到了卡耐基·梅隆大学毕业的一位博士生詹姆斯·高斯林（James Gosling）。他用C语言开发了一款名为Gosling Emacs的编辑器，简称Gosmacs。它克隆了部分Emacs的功能，内建了一个简单的Lisp语言解释器，实现了简单的Lisp语法，并把这种Lisp方言成为Mocklisp。尽管高斯林已经把Gosmacs卖给了一家名为UniPress的私人软件公司，但是高斯林在读博士期间，还曾通过邮件，授权给了一位参与过Gosmacs早期开发的人，让这位开发者可以独立发布Gosmacs。本着互助的精神，这位开发者将代码给了理查德，并担保可以自由使用。

At first Stallman thought he would change only the user-level commands, to implement full compatibility with the original PDP-10 Emacs. However, when he found how weak Mocklisp was in comparison with real Lisp, he felt compelled to replace it with a true Lisp system. This made it natural to rewrite most of the higher-level code of Gosmacs in a completely different way, taking advantage of the greater power and flexible data structures of Lisp. By mid-1985, in GNU Emacs as released on the Internet, only a few files still had code remaining from Gosmacs.

一开始，理查德以为他只要修改一些用户命令，就可以实现当年PDP-10上的Emacs的功能。然而，他发现Gosmacs使用的Mocklisp解释器实在太山寨，很难称得上是个真正的Lisp解释器。于是理查德不得不先去实现一个真正的Lisp系统。理查德要大刀阔斧地修改Gosmacs的代码了。他要让这个编辑器用上Lisp的灵活又强大的数据结构。1985年中期，GNU Emacs正式在因特网上发布。发布的代码中，只有很少的几个文件里，还保留着Gosmacs原来的代码。

Then UniPress caught wind of Stallman's project, and denied that the other developer had received permission to distribute his own version of Gosmacs.

He could not find a copy of the old email to defend his claim. Stallman eliminated this problem by writing replacements for the few modules that remained from Gosmacs.

接下来，UniPress得到了理查德利用Gosmacs开发GNU Emacs的消息。他们拒绝承认高斯林曾把发行权授予了别人。理查德找不到当初的通信邮件，来证明自己的清白。于是，他把剩下的那些从Gosmacs拿来的代码清理干净，解决了这场纠纷。

Nevertheless, the notion of developers selling off software rights – indeed, the very notion of developers having such powers to sell in the first place – rankled Stallman. In a 1986 speech at the Swedish Royal Technical Institute, Stallman cited the UniPress incident as yet another example of the dangers associated with proprietary software.

不过，这次的纠纷又让理查德对专有软件有了更多的思考。在1986年，理查德在瑞典皇家技术研究所的演讲中曾引用了这次与UniPress的纠纷，并把它作为典型案例，来说明专有软件的危害。

“Sometimes I think that perhaps one of the best things I could do with my life is find a gigantic pile of proprietary software that was a trade secret, and start handing out copies on a street corner so it wouldn’t be a trade secret any more,” said Stallman. “Perhaps that would be a much more efficient way for me to give people new free software than actually writing it myself; but everyone is too cowardly to even take it.”

理查德曾说：“有时候，我真觉得我这辈子应该找来一大堆专有软件，给它们一个个破解刻称光盘，然后拿到街角去免费赠送给路人。没准那样倒是可以更快地给用户带来些自由。要比自己亲自操刀写软件方便得多。不过要是哪样，可能很多人都不敢要我送出的光盘。”

Despite the stress it generated, the dispute over Gosling’s code would assist both Stallman and the free software movement in the long term. It would force Stallman to address the weaknesses of the Emacs Commune and the informal trust system that had allowed problematic offshoots to emerge. It would also force Stallman to sharpen the free software movement’s political objectives. Following the release of GNU Emacs in 1985, Stallman issued *The GNU Manifesto*, an expansion of the original announcement posted in September, 1983. Stallman included within the document a lengthy section devoted to the

many arguments used by commercial and academic programmers to justify the proliferation of proprietary software programs. One argument, “Don’t programmers deserve a reward for their creativity,” earned a response encapsulating Stallman’s anger over the recent Gosling Emacs episode:

抛开各种压力不说，抛弃高斯林写的代码倒未必是件坏事。长远来看，对于理查德和整个自由软件运动，这个做法不失为明智之举。它让理查德必须去直面Emacs社区里的各种弱点。让他重新思考这种非正式的信任关系带来的额外麻烦。它也催促着理查德更加明确自由软件运动的政治目标。在1985年，理查德照常又发布了新一版的Emacs。同时，他也发布了《GNU宣言》。这个宣言是他1983年9月那篇文章的扩展。在宣言中，理查德用了很长的一节，引述了各种在商业或学术领域里，为专有软件辩护的观点。其中一个经典观点就是：“难道程序员们的这些创造就不值得什么物质奖励吗？”理查德对此的回应是：

“If anything deserves a reward, it is social contribution,” Stallman wrote. “Creativity can be a social contribution, but only in so far [*sic*] as society is free to use the results. If programmers deserve to be rewarded for creating innovative programs, by the same token they deserve to be punished if they restrict the use of these programs.”

“如果要让大家对什么行为做物质上的奖励，那么这个行为必须是对社会有一定贡献的。创造本身的确是一种有益于社会的活动。但前提是这个社会可以自由地使用那些创造出来的东西。程序员创造了某些有用的软件，如果这值得奖励；那么按照这个逻辑，要是他们借着作者的身份限制软件的使用，那他们就得接受惩罚了。”

With the release of GNU Emacs, the GNU Project finally had code to show. It also had the burdens of any software-based enterprise. As more and more Unix developers began playing with the software, money, gifts, and requests for tapes began to pour in. To address the business side of the GNU Project, Stallman drafted a few of his colleagues and formed the Free Software Foundation (FSF), a nonprofit organization dedicated to speeding the GNU Project towards its goal. With Stallman as president and various friends and hacker allies as board members, the FSF helped provide a corporate face for the GNU Project.

随着GNU Emacs的发布，GNU工程也慢慢地积累起了给大家读的代码。而GNU工程也逐渐要面对越来越多的琐事。越来越多的UNIX开发人员开始使用GNU的软件，各种捐款，礼品，以及索取磁带的请求也接踵而至。为了解决这

些非技术类问题，理查德拉来了一些挚友，组建了自由软件基金会（Free Software Foundation，简称FSF）。这是一个非盈利性组织，旨在促进GNU工程完成目标。理查德担任主席，理查德的很多朋友和黑客同道都成为了董事会成员。自由软件基金会为GNU提供了一个对外的窗口。

Robert Chassell, a programmer then working at Lisp Machines, Inc., became one of five charter board members at the Free Software Foundation following a dinner conversation with Stallman. Chassell also served as the organization's treasurer, a role that started small but quickly grew.

罗伯特·查瑟尔（Robert Chassell）当年曾是LMI公司的一名程序员，之后成为了自由软件基金会的五个注册董事会成员之一。他曾在自由软件基金会担任会计。这个职务一开始并不起眼，后来却越来越重要了。

"I think in '85 our total expenses and revenue were something in the order of \$23,000, give or take," Chassell recalls. "Richard had his office, and we borrowed space. I put all the stuff, especially the tapes, under my desk. It wasn't until sometime later LMI loaned us some space where we could store tapes and things of that sort."

查瑟尔回忆："我们在1985年那会，花费和收入大概在两万三千美元这个级别。理查德有一间自己的办公室，我们还租了点地方。我把所有东西，尤其是很多磁带，放在我桌子底下。后来，LMI借给我们一点地方，我们才把磁带和类似的东西放在了那里。"

In addition to providing a face, the Free Software Foundation provided a center of gravity for other disenchanted programmers. The Unix market that had seemed so collegial even at the time of Stallman's initial GNU announcement was becoming increasingly competitive. In an attempt to tighten their hold on customers, companies were starting to deny users access to Unix source code, a trend that only speeded the number of inquiries into ongoing GNU software projects. The Unix wizards who once regarded Stallman as a noisy kook were now beginning to see him as a software prophet or a software Cassandra, according as they felt hope or despair over escaping the problems he identified.

除了提供一个门面意外，自由软件基金会还吸引了很多抱有类似理想的程序员。UNIX市场融入了越来越多的竞争者，理查德的GNU工程也逐渐占有了一席之地。UNIX厂商们为了能栓住用户，它们开始拒绝为用户提供UNIX的源代码。

不过这个行为最终倒让GNU获得了更多的用户。UNIX界的人士曾一度把理查德视为不切实际的疯子。如今则有不少人把他看作软件界的预言家。因为有越来越多的人，遇到了理查德描述的那些专有软件的问题。

“A lot of people don’t realize, until they’ve had it happen to them, how frustrating it can be to spend a few years working on a software program only to have it taken away,” says Chassell, summarizing the feelings and opinions of the correspondents writing in to the FSF during the early years. ``After that happens a couple of times, you start to say to yourself, ‘Hey, wait a minute.’

”

查瑟尔曾描述过这种感觉：“很多人一直对此没有意识。等到事情真的发生在自己身上，才有所察觉。他们花了几年的时间，去开发完善一个软件，可到头来却被别人占位己有，不许他们再碰。等这种事情发生几次，你就会对自己说，‘等会，这真的不对头。’”

For Chassell, the decision to participate in the Free Software Foundation came down to his own personal feelings of loss. Prior to LMI, Chassell had been working for hire, writing an introductory book on Unix for Cadmus, Inc., a Cambridge-area software company. When Cadmus folded, taking the rights to the book down with it, Chassell says he attempted to buy the rights back with no success.

查瑟尔加入自由软件基金会的一个重要原因，源自曾经的一次个人经历。在加入LMI之前，查瑟尔曾为Cadmus公司写了一本UNIX的入门书。Cadmus是美国剑桥市的一家软件公司，之后关门倒闭了。而查瑟尔写的这本书，也随之灰飞烟灭。查瑟尔曾几次试图买回这本书的版权，但都无果而终。

“As far as I know, that book is still sitting on a shelf somewhere, unusable, uncopyable, just taken out of the system,” Chassell says. “It was quite a good introduction if I may say so myself. It would have taken maybe three or four months to convert [the book] into a perfectly usable introduction to GNU/Linux today. The whole experience, aside from what I have in my memory, was lost.”

“我知道，那本书如今就在某个书架上，别人用不了，读不上，也没法复印。它从此就与世长辞，不见天日。”查瑟尔说，“容我自夸一下，它真的是本不错的入门书。要是能再给我三四个月的时间，可没准可以把它改成一本很好的GNU/Linux的入门书。整个经历，只能留在我的记忆中，别的都没了。”

Forced to watch his work sink into the mire while his erstwhile employer struggled through bankruptcy, Chassell says he felt a hint of the anger that drove Stallman to fits of apoplexy. “The main clarity, for me, was the sense that if you want to have a decent life, you don’t want to have bits of it closed off,” Chassell says. “This whole idea of having the freedom to go in and to fix something and modify it, whatever it may be, it really makes a difference. It makes one think happily that after you’ve lived a few years that what you’ve done is worthwhile. Because otherwise it just gets taken away and thrown out or abandoned or, at the very least, you no longer have any relation to it. It’s like losing a bit of your life.”

眼睁睁地看着自己的作品就此消失，曾经的雇主破产关门，查瑟尔说，他好像感到了当年理查德心中的那份怒火。“对我来说，有件事情越来越清楚：你要想过个像样的日子，你就不想有一丝遗憾。”查瑟尔说：“有深入了解的自由，有修改的自由，无论这是什么，这想法本身确实改变了这个世界。他让人觉得，在自己短短的生命之中，所做之事是值得付出的。因为它还没被谁夺走，也许还没被人弃置遗忘。至少，你还和它有所联系。不会让你觉得自己丢失了一段美妙的人生。”

St. Ignucius 圣·Ignucius

The Maui High Performance Computing Center is located in a single-story building in the dusty red hills just above the town of Kihei. Framed by million-dollar views and the multimillion dollar real estate of the Silversword Golf Course, the center seems like the ultimate scientific boondoggle. Far from the boxy, sterile confines of Tech Square or even the sprawling research metropolises of Argonne, Illinois and Los Alamos, New Mexico, the MHPCC seems like the kind of place where scientists spend more time on their tans than their post-doctoral research projects.

茂宜高性能计算中心（Maui High Performance Computing Center）坐落在夏威夷群岛中的茂宜岛上。这是一座单层建筑，建在满是红色火山灰的山岗上，俯瞰茂宜岛的商业中心纪平镇。这个计算中心身在夏威夷，坐拥迷人的山景海风，更有身价千万的房产。这地方好像和高新技术占不上一点边。它跟麻省理工学院技术广场的见楞见角，死气沉沉的气氛完全不同。哪怕跟阿尔贡国家实验室，洛斯阿拉莫斯或者新墨西哥的那些实验室比，茂宜高性能计算中心依旧显得特别。总会难免让人觉得，茂宜计算中心的科学家们每天可能并不怎么关

心自己的研究课题，没准他们天天都会把时间花在晒太阳，吹海风上。

The image is only half true. Although researchers at the MHPCC do take advantage of the local recreational opportunities, they also take their work seriously. According to Top500.org, a web site that tracks the most powerful supercomputers in the world, the IBM SP Power3 supercomputer housed within the MHPCC clocks in at 837 billion floating-point operations per second, making it one of 25 most powerful computers in the world. Co-owned and operated by the University of Hawaii and the U.S. Air Force, the machine divides its computer cycles between the number crunching tasks associated with military logistics and high-temperature physics research.

不过这个第一感觉可不怎么准确。确实，茂宜计算中心的研究人员的确占尽了地利，也自然会借此休养。不过他们对待工作还是非常认真的。Top500.org 是一个记录全世界超级计算机排名的网站。根据这个网站的资料，茂宜高性能计算中心的IBM SP Power3超级计算机可以每秒完成八千三百七十亿次浮点运算，世界排名前25。这台超级计算机归夏威夷大学和美国空军共同拥有和维护。它主要负责完成军队补给调度计算和高温物理研究。

Simply put, the MHPCC is a unique place, a place where the brainy culture of science and engineering and the laid-back culture of the Hawaiian islands coexist in peaceful equilibrium. A slogan on the lab's 2000 web site sums it up: "Computing in paradise."

简而言之，茂宜计算中心是个非常独特的地方。这地方既有学术研究的文化氛围，又融入了夏威夷上那份慵懒宜人的情调。二者一阴一阳，相得益彰。2000年，他们在自己的官方网站放上了这样一条标语：在天堂之上计算。

It's not exactly the kind of place you'd expect to find Richard Stallman, a man who, when taking in the beautiful view of the nearby Maui Channel through the picture windows of a staffer's office, mutters a terse critique: "Too much sun." Still, as an emissary from one computing paradise to another, Stallman has a message to deliver, even if it means subjecting his hacker eyes to painful solar glare.

这样一个地方，一般你是见不到理查德·斯托曼身影的。他这次来到这里，站在一位员工办公室的窗口，盯着外面如画的风景，口中却蹦出了一句简单的批评："阳光太强。"自然，他从计算机界的圣地麻省理工学院来到这里，是有话要讲的。哪怕再强的阳光，刺入这位黑客的双眼，他倒也在所不辞。

The conference room is already full by the time I arrive to catch Stallman's speech. The gender breakdown is a little better than at the New York speech, 85% male, 15% female, but not by much. About half of the audience members wear khaki pants and logo-encrusted golf shirts. The other half seems to have gone native. Dressed in the gaudy flower-print shirts so popular in this corner of the world, their faces are a deep shade of ochre. The only residual indication of geek status are the gadgets: Nokia cell phones, Palm Pilots, and Sony VAIO laptops.

我赶到现场的时候，会议室里早已人满为患，大家都在等待着理查德的演讲。听众中，男女比例倒是比纽约那次的演讲平衡些，不过也没平衡到哪儿去：大约85%是男性，15%的女性。有一半的听众，都身穿土黄色的裤子，配上印着商标的高尔夫衬衫。另外一半，则显得更加融入了当地的风俗。身穿宽大的大花T恤，皮肤则已被晒成了褐色。这打扮在夏威夷可是正常装束。唯一能显示他们技术本色的东西，也就只剩下了些小玩意儿了：诺基亚手机，掌上电脑，还有索尼VAIO笔记本。

Needless to say, Stallman, who stands in front of the room dressed in plain blue T-shirt, brown polyester slacks, and white socks, sticks out like a sore thumb. The fluorescent lights of the conference room help bring out the unhealthy color of his sun-starved skin. His beard and hair are enough to trigger beads of sweat on even the coolest Hawaiian neck. Short of having the words "mainlander" tattooed on his forehead, Stallman couldn't look more alien if he tried. [RMS: Is there something bad about looking different from others?]

理查德身穿纯蓝色T恤，棕色涤纶休闲裤，白色袜子。不用说，这身行头在当下确实太显眼了。房间中淡淡的花香，倒是让人没太注意理查德缺少阳光滋润的皮肤。不过他一头长发和络腮胡须却依旧能引人注目。外人一看，就知道他不是本地人。他就差在脑门上贴个标签，写着："美国本土人士。"[RMS：把自己打扮得和别人不一样有什么不对吗？]

As Stallman putters around the front of the room, a few audience members wearing T-shirts with the logo of the Maui FreeBSD Users Group (MFUG) race to set up camera and audio equipment. FreeBSD, a free software offshoot of the Berkeley Software Distribution, the venerable 1970s academic version of Unix, is technically a competitor to the GNU/Linux operating system. Still, in the hacking world, Stallman speeches are documented with a fervor

reminiscent of the Grateful Dead and its legendary army of amateur archivists. As the local free software heads, it's up to the MFUG members to make sure fellow programmers in Hamburg, Mumbai, and Novosibirsk don't miss out on the latest pearls of RMS wisdom.

演讲开始前，理查德自己在会议室前面闲逛。旁边几个人鼓捣着录像录音器材。这些人都身穿"茂宜FreeBSD用户组"（Maui FreeBSD User Group，MFUG）的T恤。所谓FreeBSD，是一款从伯克利软件发行版的UNIX系统分支而来的操作系统。而伯克利版UNIX则是七十年代由加州大学伯克利分校开发的一个UNIX分支。说起来，FreeBSD倒还是GNU/Linux的一个竞争对手。在黑客的世界里，理查德的演讲总会被记录拍摄下来，留做档案。作为当地的著名自由软件组织，茂宜FreeBSD用户组可不能让同行失望。他们也要录音录像，让远在德国汉堡，印度孟买，俄罗斯新西伯利亚等世界各地的黑客们，都能听到RMS的箴言。

The analogy to the Grateful Dead is apt. Often, when describing the business opportunities inherent within the free software model, Stallman has held up the Grateful Dead as an example. In refusing to restrict fans' ability to record live concerts, the Grateful Dead became more than a rock group. They became the center of a tribal community dedicated to Grateful Dead music. Over time, that tribal community became so large and so devoted that the band shunned record contracts and supported itself solely through musical tours and live appearances. In 1994, the band's last year as a touring act, the Grateful Dead drew \$52 million in gate receipts alone.

理查德的这些演讲记录，就像是感恩而死乐队（Grateful Dead）的现场演出一样，都是被追捧者现场录下，私下传递。把理查德和感恩而死乐队做类比，可是有说头的。每当理查德描述自由软件的商业盈利模式的时候，他都会提起感恩而死乐队。这个乐队当年允许粉丝们在现场录音录像，这就令它不仅是一个简单的摇滚乐队。它俨然成了感恩而死音乐部落的中心。逐渐地，这个部落越来越大，令感恩而死乐队拒绝了各种唱片公司的合约，完全靠巡回演出，就可以撑起整个乐队的开支。在1994年，这支乐队的最后一次演出仅门票收入就高达五千两百万美元。

While few software companies have been able to match that success, the tribal aspect of the free software community is one reason many in the latter half of the 1990s started to accept the notion that publishing software source code might be a good thing. Hoping to build their own loyal followings,

companies such as IBM, Sun Microsystems, and Hewlett Packard have come to accept the letter, if not the spirit, of the Stallman free software message. Describing the GPL as the information-technology industry's *Magna Carta*, ZDNet software columnist Evan Leibovitch sees the growing affection for all things GNU as more than just a trend. "This societal shift is letting users take back control of their futures," Leibovitch writes. "Just as the *Magna Carta* gave rights to British subjects, the GPL enforces consumer rights and freedoms on behalf of the users of computer software."

当然，还没有哪个软件公司的成就可以媲美感恩而死乐队。不过，自由软件社区的这个部落越来越大，在九十年代后期，大家也逐渐开始认为分享发布软件的源代码没准是个好事。各大公司纷纷出马，企图建立起各自的忠实粉丝群。这其中就包括IBM，Sun，惠普等。它们也许并没有打心眼里接受自由软件的看法，但他们的行为却是在响应理查德的号召。ZDNet的软件专栏作家埃文·列伊博维奇（Evan Leibovitch）把GPL（GNU通用公共许可证，理查德创立的一种自由软件许可证，用于规定软件作者授予软件用户的各种权利）描述为信息时代的大宪章。他认为，人们对于GNU的拥护持续高涨，并非单纯地跟风。他曾写道："这个趋势让用户可以重新掌控自己的命运。就好像当年《大宪章》给了英国人民权利。GPL让消费者作为计算机软件用户有了自己的权利和自由。"

The tribal aspect of the free software community also helps explain why 40-odd programmers, who might otherwise be working on physics projects or surfing the Web for windsurfing buoy reports, have packed into a conference room to hear Stallman speak.

自由软件社区的这种部落文化也解释了为什么今天这四十来位古怪的程序员们，可以暂时不管各自的工作，也不去上网冲浪，而宁愿能挤在这个会议室里，听理查德的讲座。

Unlike the New York speech, Stallman gets no introduction. He also offers no self-introduction. When the FreeBSD people finally get their equipment up and running, Stallman simply steps forward, starts speaking, and steamrolls over every other voice in the room.

和那次在纽约的讲座不一样，这次没有人为他做开场白，理查德自己也没做自我介绍。FreeBSD用户组的人把设备一架好，理查德就自己踏上前来，开始演讲，声音一下盖过了台下的私语。

"Most of the time when people consider the question of what rules society

should have for using software, the people considering it are from software companies, and they consider the question from a self-serving perspective," says Stallman, opening his speech. "What rules can we impose on everybody else so they have to pay us lots of money? I had the good fortune in the 1970s to be part of a community of programmers who shared software. And because of this I always like to look at the same issue from a different direction to ask: what kind of rules make possible a good society that is good for the people who are in it? And therefore I reach completely different answers."

"一个社会该给软件的使用设立何种规则？每当想到这个问题，大多数人总是站在软件公司的立场思考。他们的想法往往像是在自我圆场。"理查德开门见山："他们想：我们如何设立规则，才能让大家多给我们钱呢？我很幸运，在七十年代，曾加入了一个大家庭，在那里面，大家都会互相分享软件。正因如此，我总会从另外一个角度去思考这个问题：我们应该如何设立规则，才能帮助我们构建一个更好的社会，一个可以让每个人都受益的社会？也正因如此，我得到了完全不一样的答案。"

Once again, Stallman quickly segues into the parable of the Xerox laser printer, taking a moment to deliver the same dramatic finger-pointing gestures to the crowd. He also devotes a minute or two to the GNU/Linux name.

理查德又一次提起了施乐打印机事件，形式和以前很像，同样是讲到最后，手指指向听众，口中说着："也许就会背叛你！"之后，他又花了一两分钟，来解释GNU/Linux的名字问题。

"Some people say to me, 'Why make such a fuss about getting credit for this system? After all, the important thing is the job is done, not whether you get recognition for it.' Well, this would be wise advice if it were true. But the job wasn't to build an operating system; the job is to spread freedom to the users of computers. And to do that we have to make it possible to do everything with computers in freedom."

"有些人会和我说：'你为什么要花这么多精力去在乎人们叫它什么？为什么非要在乎这种名利上的事？如今任务已经做完了，你何必非要在乎人们有没有注意到GNU呢？'这个……如果这话描述的是事实，那我确实同意这话的观点。可现实是：我们的任务并没有完成。我们的任务不是要创造一套操作系统；我们的任务是要给计算机用户自由。要完成这个任务，我们必须要尽一切努力，让用户可以自由地使用计算机和软件。"

Adds Stallman, “There’s a lot more work to do.”

理查德最后说：“我们要走的路还很长。”

For some in the audience, this is old material. For others, it’s a little arcane. When a member of the golf-shirt contingent starts dozing off, Stallman stops the speech and asks somebody to wake the person up.

对于一些听众来说，理查德的这些话是老生长谈了。对于其他人来说，这些话则有点令人不可思议了。旁边那个身穿高尔夫T恤的听众已经开始打盹了，理查德停下来，让旁边的人把这位叫醒。

“Somebody once said my voice was so soothing, he asked if I was some kind of healer,” says Stallman, drawing a quick laugh from the crowd. “I guess that probably means I can help you drift gently into a blissful, relaxing sleep. And some of you might need that. I guess I shouldn’t object if you do. If you need to sleep, by all means do.”

“曾经有个人，说我的声音特别像安慰剂，他问我是不是做过催眠师之类的工作。”理查德说着，下面一阵笑声。在笑声中他接着说：“我想这意味着我可以帮你们快速进入梦乡。可能你们之中有些人正想要这样。我觉得我确实也不该打搅你们，你要是真想睡觉，尽管睡吧。”

The speech ends with a brief discussion of software patents, a growing issue of concern both within the software industry and within the free software community. Like Napster, software patents reflect the awkward nature of applying laws and concepts written for the physical world to the frictionless universe of information technology.

演讲最后，斯托曼简要地探讨了软件专利的问题。这个问题逐渐在软件业和自由软件社区中受到关注。专利的概念和想法本是为现实的物理世界准备。可把它用在信息世界中，就越来越显得古怪了。

Copyright law and patent law work differently, and have totally different effects in the software field. The copyright on a program controls the copying and adaptation of that program’s code, and it belongs to the program’s developer. But copyright does not cover ideas. In other words, a developer is free, under copyright, to implement in his own code features and commands he has seen in existing programs. Those aspects are ideas, not expression, and thus outside the scope of copyright law.

版权和专利是两个不同的概念，也有着完全不同的目的。对于软件来说，版权法用来约束拷贝或修改软件代码的行为。一个程序的版权是归开发者所有。但是版权不能保护作者的想法。换句话说，开发者可以克隆一个已经存在的软件，只要他不去复制别人的代码，就不会涉及到版权问题。因为这是在重现别人的想法，所以这在版权法的作用范围之外。

It is likewise lawful – though hard work – to decode how a binary program works, and then implement the same ideas and algorithms in different code. This practice is known as “reverse engineering.”

如果开发者直接阅读软件的机器码，尽管会比较费事费力，但这依旧是合法的。同理，开发者也可以通过阅读软件的机器码，再自己独立实现一个类似的软件。这个做法通常被称作“逆向工程”。

Software patents work differently. According to the U.S. Patent Office, companies and individuals can obtain patents for computing ideas that are innovative (or, at least, unknown to the Patent Office). In theory, this allows the patent-holder to trade off disclosure of the technique for a specific monopoly lasting a minimum of 20 years after the patent filing. In practice, the disclosure is of limited value to the public, since the operation of the program is often self-evident, and could in any case be determined by reverse engineering. Unlike copyright, a patent gives its holder the power to forbid the independent development of software programs which use the patented idea.

可软件专利就不同了。按照美国专利局的说法，个人或公司的计算方面的创新想法，可以获得专利（准确说，所谓创新想法，也就是专利局没听说过的想法）。理论上讲，专利拥有者需要公开自己的创新想法，由此可以垄断这个创造至少二十年，计时从专利提交起开始。但是实际上，公开专利的创新想法并不能怎么让公众受益。因为程序的想法往往是不言自明的。或者可以通过逆向工程获得。而专利法和版权法不同，它禁止任何人实现专利上描述的想法，哪怕开发者是独立开发的。

In the software industry, where 20 years can cover the entire life cycle of a marketplace, patents take on a strategic weight. Where companies such as Microsoft and Apple once battled over copyright and the “look and feel” of various technologies, today’s Internet companies use patents as a way to stake out individual applications and business models, the most notorious example being Amazon.com’s 2000 attempt to patent the company’s “one-click” online shopping process. For most companies, however, software

patents have become a defensive tool, with cross-licensing deals balancing one set of corporate patents against another in a tense form of corporate detente. Still, in a few notable cases of computer encryption and graphic imaging algorithms, software vendors have successfully stifled rival developments. For instance, some font-rendering features are missing from free software because of patent threats from Apple.

在软件世界里，二十年的时间可能覆盖了一个市场的整个生命周期。这样，专利就成了一种策略。当年，微软和苹果这些大公司为版权打得不可开交。而如今那些互联网企业，则使用专利来为自己开路，赶走竞争对手和新人。这里有个著名的案例。2000年，亚马逊购物曾试图为它的“一键购买”功能注册专利。简单说，这个功能就是允许用户事先存储好个人信息——比如信用卡信息，收件人地址等——然后在购买商品的时候，只需要按一个按钮，就可以下单，不必再跳转到其他页面填写地址，付款等信息。对于大多数公司来说，软件专利更像是一种防卫措施。几个公司可能会合作共有一系列专利，用来阻止另外一个联盟的专利。尽管如此，还是有不少领域被软件专利占满，阻碍了各种可能的竞争对手。比如图像处理领域，加密领域等等。比较实际的例子是，一些字体渲染算法，不会出现在自由软件中，因为苹果公司持有这些算法的专利。

For Stallman, the software-patent issue dramatizes the need for eternal hacker vigilance. It also underlines the importance of stressing the political benefits of free software programs over the competitive benefits. Stallman says competitive performance and price, two areas where free software operating systems such as GNU/Linux and FreeBSD already hold a distinct advantage over their proprietary counterparts, are side issues compared to the large issues of user and developer freedom.

对于理查德来说，软件专利更督促着黑客要时刻警觉，而且还凭添了几分戏剧色彩。也恰恰是软件专利，让自由软件一直强调的政治策略再次得以重申。即，用户的自由比其他更重要。理查德说，高性能，低价格等特性的确是GNU/Linux，FreeBSD等这些自由软件的优势。但是除了性能，价格这些东西以外，还有更重大的问题，那就是用户和开发者的自由。

This position is controversial within the community: open source advocates emphasize the utilitarian advantages of free software over the political advantages. Rather than stress the political significance of free software programs, open source advocates have chosen to stress the engineering integrity of the hacker development model. Citing the power of peer review,

the open source argument paints programs such as GNU/Linux or FreeBSD as better built, better inspected and, by extension, more trustworthy to the average user.

这个观点在圈子里也饱受争议：开源的支持者们喜欢强调自由软件的实用性，会避免探讨用户自由之类的问题。他们可能不会强调这些，转而强调黑客开发的工程模型。强调同行互相审校的重要性。他们认为，GNU/Linux或FreeBSD的开发模式可以创造出更好，更强大，更值得信赖的软件。

That's not to say the term "open source" doesn't have its political implications. For open source advocates, the term open source serves two purposes. First, it eliminates the confusion associated with the word "free," a word many businesses interpret as meaning "zero cost." Second, it allows companies to examine the free software phenomenon on a technological, rather than ethical, basis. Eric Raymond, cofounder of the Open Source Initiative and one of the leading hackers to endorse the term, explained his refusal to follow Stallman's political path in a 1999 essay, titled "Shut Up and Show Them the Code":

但是，"开源"一词也并非意味着没有任何政治诉求。对于开源支持者来说，开源有着两重含义。第一，它避免了在英文中使用Free一词。自由软件的英文是Free Software，因为Free在英文中既有自由的意思，也有免费的意项，所以难免会产生歧义。很多商业公司会觉得自由软件等于免费软件。使用"开源"一词的动机之一，就是希望消除这种歧义。第二，它让商业公司可以从技术角度去审视自由软件，而不必在道义过多解读。艾瑞克·雷蒙德（Eric Raymond）是开放源代码促进会（Open Source Initiative，缩写：OSI）的联合创始人之一，也是支持使用"开源"一词的著名黑客。他曾在1999年发表了一篇文章，用以表明他拒绝接受理查德的政治路线。文章标题是《少说话，秀代码》（Shut Up and Show Them the Code）：

> RMS's rhetoric is very seductive to the kind of people we are. We hackers are thinkers and idealists who readily resonate with appeals to "principle" and "freedom" and "rights." Even when we disagree with bits of his program, we want RMS's rhetorical style to work; we think it ought to work; we tend to be puzzled and disbelieving when it fails on the 95% of people who aren't wired like we are.

RMS[即，理查德·M·斯托曼的缩写]的用词对于我们这样的人来说很有吸引力。我们黑客都是思考者，都是理想主义者。我们都时刻准备着响应支持"自

由”，“原则”和“权利”的号召。哪怕我们有时候会与理查德的一些意见不同，但我们依旧希望他的做法能有效。我们觉得他的言辞应该有效。可是，我们看到世界上另外95%的人和我们不一样，理查德的言论对他们无效。面对这些，我们开始迷惑，甚至怀疑。

Included among that 95%, Raymond writes, are the bulk of business managers, investors, and nonhacker computer users who, through sheer weight of numbers, tend to decide the overall direction of the commercial software marketplace. Without a way to win these people over, Raymond argues, programmers are doomed to pursue their ideology on the periphery of society:

雷蒙德写道，那另外95%的人中，包括很多公司经理，投资人，还有普通的计算机用户，他们不是黑客，他们会倾向于跟从专有商业软件的大市场。雷蒙德认为，如果不能赢得这些人的支持，黑客们将只能游离于主流之外，去追随自己的理想。

> When RMS insists that we talk about “computer users’ rights,” he’s issuing a dangerously attractive invitation to us to repeat old failures. It’s one we should reject – not because his principles are wrong, but because that kind of language, applied to software, simply does not persuade anybody but us. In fact, it confuses and repels most people outside our culture.
>
> 理查德向我们谈论起“计算机用户权利”的时候，他向我们发出了一份吸引人却危险的邀请函，引导着我们重犯旧错。我们应该拒绝它——不是因为它的原则是错的，而是因为那样的词汇用在软件上，只能说服我们自己，而不能说服别人。事实是，它把大多数圈外人都拒之千里。

Stallman, however, rejects Raymond’s premises:

但是，理查德却反对雷蒙德的说法：

> Raymond’s attempt to explain our failure is misleading because we have not failed. Our goal is large, and we have a long way to go, but we have also come a long way.
>
> Raymond’s pessimistic assertion about the values of non-hackers is an exaggeration. Many non-hackers are more concerned with the political issues we focus on than with the technical advantages that open source

> emphasizes. This often includes political leaders too, though not in all countries.
>
> It was the ethical ideals of free software, not “better software,” which persuaded the presidents of Ecuador and Brazil to move government agencies to free software. They are not geeks, but they understand freedom.
>
> 雷蒙德对我们失败的解释非常有误导性。因为我们压根都没有失败呢。我们为自己设定的目标很远，我们还有多长的路要走。可是要知道，我们已经走过了很多路，克服了很多困难。
>
> 雷蒙德关于大众的悲观判断是夸大其辞。很多人不是黑客，但是却同样关注我们强调的政治问题，他们也认为自由的问题比开源所强调的技术优势要重要。这些人之中，甚至会有一些政治领导人。
>
> 厄瓜多尔和巴西的政府恰恰就是因为考虑到道义问题，而不是技术问题，才让政府计算机使用自由软件。他们可不是黑客，但他们也理解其中的自由。

But the principal flaw in the open source argument, according to Stallman, is that it leads to weaker conclusions. It convinces many users to run some programs which are free, but does not offer them any reason to migrate entirely to free software. This partially gives them freedom, but does not teach them to recognize it and value it as such, so they remain likely to let it drop and lose it. For instance, what happens when the improvement of free software is blocked by a patent?

对于理查德来说，开源的重大问题，在于得出了过于宽松的结论。它可以说服用户去使用自由软件，但却无法提供更多的理由，让用户完全放弃专有软件。这只能给用户部分自由，却不能教会用户去珍视自由。最终，用户还是很可能继续掉入专有软件的陷阱，本来的那些自由也从此丢失。举个例子，如果因为专利问题，让某个自由软件无法添加某项功能，用户该做何选择?

Most open source advocates are equally, if not more, vociferous as Stallman when it comes to opposing software patents. So too are most proprietary software developers, since patents threaten their projects too. However, pointing to software patents’ tendency to put areas of software functionality off limits, Stallman contrasts what the free software idea and the open source

idea imply about such cases.

大多数开源支持者们也会和理查德一样，极力反对软件专利。甚至很多专有软件的开发者也一样反对软件专利，因为软件专利也同样威胁着他们的项目。然而，理查德指出，软件专利会阻止人们实现某些功能。这里，开源和自由软件在理念上的分歧就显现出来了：

“It’s not because we don’t have the talent to make better software,” says Stallman. “It’s because we don’t have the right. Somebody has prohibited us from serving the public. So what’s going to happen when users encounter these gaps in free software? Well, if they have been persuaded by the open source movement that these freedoms are good because they lead to more-powerful reliable software, they’re likely to say, ‘You didn’t deliver what you promised. This software’s not more powerful. It’s missing this feature. You lied to me.’ But if they have come to agree with the free software movement, that the freedom is important in itself, then they will say, ‘How dare those people stop me from having this feature and my freedom too.’ And with that kind of response, we may survive the hits that we’re going to take as these patents explode.”

“这个时候，我们如果做不出更好的软件，并不是能力不足。而是因为有人利用专利，故意阻止我们为公众服务。倘若一位用户使用的自由软件中，由于某项专利，而缺少一个功能，用户会作何反应呢？如果当初这些用户是被开源软件的理念说服而使用这个自由软件，那么他们当初学到的是：给予用户自由是为了创造更强大可靠的软件。这时候，遇到软件缺少功能，他们会说，‘你们没能给我一个强大可靠的软件，它少了这个功能，你们骗了我。’相反，如果这个用户是因为认同自由软件的价值观，而使用自由软件的话，那么他们会说：‘那些专利竟然阻止了这个功能的实现，剥夺了我的自由。’只有用户认同自由软件的价值观，那么面对大量的软件专利，我们才可能幸存。”

Watching Stallman deliver his political message in person, it is hard to see anything confusing or repellent. Stallman’s appearance may seem off-putting, but his message is logical. When an audience member asks if, in shunning proprietary software, free software proponents lose the ability to keep up with the latest technological advancements, Stallman answers the question in terms of his own personal beliefs. “I think that freedom is more important than mere technical advance,” he says. “I would always choose a less advanced free program rather than a more advanced nonfree program, because I won’t

give up my freedom for something like that [advance]. My rule is, if I can't share it with you, I won't take it."

看着理查德阐述自己的政治诉求，并不会让人迷惑或厌恶。他衣着朴素，说起话来有理有据，逻辑清晰。听众中一位提问：如果完全避免使用专有软件，可能无法用到最新的技术，该怎么办？理查德引述自己的信念，回答了这个问题："我觉得自由本身比任何新近技术都重要。如果面前有一个先进的专有软件，和一个技术落后的自由软件，那么我宁愿选择后者。因为无论如何，我都不会靠出卖自由，而换取更新的技术。我的原则是，如果我不能和你分享这个软件，我就不会使用它。"

In the minds of those who assume ethics means religion, such answers reinforce the quasi-religious nature of the Stallman message. However, unlike a Jew keeping kosher or a Mormon refusing to drink alcohol, Stallman is not obeying a commandment, but simply refusing to cede his freedom. His speech explains the practical requisites for doing so: a proprietary program takes away your freedom, so if you want freedom, you need to reject the program.

有些人可能会把道义问题和宗教信仰联系起来。理查德的这个回答则更可能给人一种宗教信仰的感觉。不过，理查德的行为不同于犹太人只吃洁食的行为，也不同于摩门教徒不喝酒。理查德并不是要守着清规戒律，他只是不想牺牲自己的自由。他的演讲解释了他的思路：专有软件会剥夺你的自由，如果你想要自由，你就要拒绝使用专有软件。

Stallman paints his decision to use free software in place of proprietary in the color of a personal belief he hopes others will come to share. As software evangelists go, Stallman avoids forcing those beliefs down listeners' throats. Then again, a listener rarely leaves a Stallman speech not knowing where the true path to software righteousness lies.

理查德仅仅使用自由软件，拒绝任何专有软件。他把这描述为个人信仰，并且希望把这份信仰与大家一同分享。但他不会强迫别人认同自己。大多数听众，在听过理查德的演讲之后，都会明白理查德所说的正确的软件之路。

As if to drive home this message, Stallman punctuates his speech with an unusual ritual. Pulling a black robe out of a plastic grocery bag, Stallman puts it on. Then he pulls out a reflective brown computer disk and places it on his head. The crowd lets out a startled laugh.

演讲最后，像个总结一样，理查德开始了一个不太寻常的仪式。他从一个百货商店的塑料购物袋里，掏出了一条黑色长袍。他把长袍打开，套在了身上。接着，又从袋子里拿出来了一个圆形的老式计算机硬盘碟片，把它戴在头上。这碟片一戴上，就像极了基督教圣徒头顶的光环。看到这身扮相，人群中传来了一阵笑声。

“I am St. IGNUcius of the Church of Emacs,” says Stallman, raising his right hand in mock-blessing. “I bless your computer, my child.”

“我是圣·IGNUcius，来自Emacs教堂。”理查德举起右手，好似在为众人祈福：“我保佑你的计算机，孩子们。”

The laughter turns into full-blown applause after a few seconds. As audience members clap, the computer disk on Stallman’s head catches the glare of an overhead light, eliciting a perfect halo effect. In the blink of an eye, Stallman resembles a Russian religious icon.

大家的笑声在几秒钟之内变成了一阵掌声。大家在鼓掌的时候，理查德头上的那个硬盘碟片反射来一道强光，一下看起来，好像理查德头上真的顶了个光环。他眨了眨眼睛，又让我觉得他好似宗教领袖。

“Emacs was initially a text editor,” says Stallman, explaining the getup. “Eventually it became a way of life for many and a religion for some. We call this religion the Church of Emacs.”

理查德接着解释道：“Emacs最开始是一个编辑器，之后，很多人把它当作一种生活方式，还有一部分人把它作为一种信仰。我们把他们称作Emacs信徒。”

The skit is a lighthearted moment of self-parody, a humorous return-jab at the many people who might see Stallman’s form of software asceticism as religious fanaticism in disguise. It is also the sound of the other shoe dropping – loudly. It’s as if, in donning his robe and halo, Stallman is finally letting listeners off the hook, saying, “It’s OK to laugh. I know I’m weird.” [RMS: To laugh at someone for being weird is boorish, and it is not my intention to excuse that. But I hope people will laugh at my St. IGNUcius comedy routine.]

很多人以为理查德对于自由软件一直都是不苟言笑，好像一个禁欲主义者一样。眼下这个玩笑倒是很好地否认了这种认识。在黑袍和光环之下，理查德好

似在说：“你们尽情笑吧，我知道我这样很搞怪。”[理查德注：嘲笑别人行为怪异是不礼貌的，我并不鼓励这种做法。不过我是希望大家看到我圣·IGNUcius的扮相，能够笑起来]

Discussing the St. IGNUcius persona afterward, Stallman says he first came up with it in 1996, long after the creation of Emacs but well before the emergence of the “open source” term and the struggle for hacker-community leadership that precipitated it. At the time, Stallman says, he wanted a way to “poke fun at himself,” to remind listeners that, though stubborn, Stallman was not the fanatic some made him out to be. It was only later, Stallman adds, that others seized the persona as a convenient way to play up his reputation as software ideologue, as Eric Raymond did in an 1999 interview with the Linux.com web site:

之后和理查德聊起圣·IGNUcius这个舞台形象，他说他最早是在1996年开始在演讲中有这么个安排的。那会他早就开发了Emacs编辑器，那时也还没有“开源”这个词。理查德说，他当时希望能有个方式，能够“自娱自乐”一下；同时，也能告诉听众，理查德虽然固执，但还不至于疯狂到不苟言笑。不过这之后，理查德说，一些人借着这个舞台形象，把理查德形容为一个软件界的空想家。艾瑞克·雷蒙德就在1999年的一次与Linux.com的采访中，说道：

> When I say RMS calibrates what he does, I’m not belittling or accusing him of insincerity. I’m saying that like all good communicators he’s got a theatrical streak. Sometimes it’s conscious – have you ever seen him in his St. IGNUcius drag, blessing software with a disk platter on his head? Mostly it’s unconscious; he’s just learned the degree of irritating stimulus that works, that holds attention without (usually) freaking people out.
>
> 我说过，RMS（理查德·M·斯托曼名字的缩写）很会算计。我并不是说他不诚实，也不是说他虚情假意。我的意思是，他和很多布道者一样，非常有舞台感。有时候，他的这种行为是有意识的——你看过他扮成圣·IGNUcius的样子吗？他身穿黑袍，把一个碟片放在头顶。还有很多时候，那是无意识的行为。他很会掌握分寸，既能让你想去反驳，又不至于把人们吓跑。

Stallman takes issue with the Raymond analysis. “It’s simply my way of making fun of myself,” he says. “The fact that others see it as anything more than that is a reflection of their agenda, not mine.”

理查德显然不同意艾瑞克·雷蒙德的说法。他说：“这不过是我自娱自乐罢了。谁要是从中还看到了别的东西，那只能说他们想多了。那都是他们脑子里的想法，不是我的。”

That said, Stallman does admit to being a ham. “Are you kidding?” he says at one point. “I love being the center of attention.” To facilitate that process, Stallman says he once enrolled in Toastmasters, an organization that helps members bolster their public-speaking skills and one Stallman recommends highly to others. He possesses a stage presence that would be the envy of most theatrical performers and feels a link to vaudevillians of years past. A few days after the Maui High Performance Computing Center speech, I allude to the 1999 LinuxWorld performance and ask Stallman if he has a Groucho Marx complex – i.e., the unwillingness to belong to any club that would have him as a member. Stallman’s response is immediate: “No, but I admire Groucho Marx in a lot of ways and certainly have been in some things I say inspired by him. But then I’ve also been inspired in some ways by Harpo.”

理查德也承认，他的这个表演有做作的成分。“就是开个玩笑。我很享受被万众瞩目。”他如是辩解。为了让自己的演讲技能可以更加娴熟，他甚至还加入过Toastermasters。这个社团旨在帮助人们提高演讲技能。理查德也曾向别人建议过Toastermasters。他加入了这个社团，逐渐掌握了一些舞台技巧。在这次茂宜高性能计算中心的演讲之后几天，我和理查德聊天的时候，问起他是否有种所谓的“格鲁桥·马克思情结”——就是像著名喜剧演员格鲁桥·马克思（Groucho Marx）所说的那样，拒绝加入任何希望收留他的俱乐部。理查德立即回复道：“不，但是我在很多方面都很敬仰格鲁桥·马克思，我说过的很多话都曾受到过他的启发。不过我也受到过汉普·马克思（Harpo Marx）的启发。”

The Groucho Marx influence is certainly evident in Stallman’s lifelong fondness for punning. Then again, punning and wordplay are common hacker traits. Perhaps the most Groucho-like aspect of Stallman’s personality, however, is the deadpan manner in which the puns are delivered. Most come so stealthily – without even the hint of a raised eyebrow or upturned smile – you almost have to wonder if Stallman’s laughing at his audience more than the audience is laughing at him.

理查德一直以来对俏皮话，双关语的喜爱，恐怕的确是受了格鲁桥·马克思的影响。不过话说回来，双关语和文字游戏也从来都是黑客们的最爱。要说理查德真是有哪点会像格鲁桥，恐怕是他开玩笑之后，那副依旧淡定的表情。他的那

些玩笑总是在不经意之间蹦出来，他自己则一脸严肃，眉毛都不抬。在你笑过之后，甚至反而会觉得理查德正在内心中笑话你。

Watching members of the Maui High Performance Computer Center laugh at the St. IGNUcius parody, such concerns evaporate. While not exactly a standup act, Stallman certainly possesses the chops to keep a roomful of engineers in stitches. “To be a saint in the Church of Emacs does not require celibacy, but it does require making a commitment to living a life of moral purity,” he tells the Maui audience. “You must exorcise the evil proprietary operating systems from all your computers, and then install a wholly [holy] free operating system. And then you must install only free software on top of that. If you make this commitment and live by it, then you too will be a saint in the Church of Emacs, and you too may have a halo.”

不过你别担心，理查德可没笑话你。看到他在茂宜高性能计算中心演讲时候的圣·IGNUcius的扮相，你的顾虑就打消了。他这表演虽然不比单口相声，但理查德总还是有能耐把一屋子的工程师聚在一起。“Emacs教会之中，不必戒色禁欲。但也需你心存底线，道清德厚，有所顾忌。”理查德继续对听众说道：“你要在自己的电脑之上，趋邪扶正。专有软件，卸载删除。辟得净土，以敬用户。系统上下，软件自由。茫茫众生，守此信条，即入本会，得道称圣。对了，到时候你没准也能拿到这样一个光环。”

The St. IGNUcius skit ends with a brief inside joke. On most Unix systems and Unix-related offshoots, the primary competitor program to Emacs is vi, pronounced vee-eye, a text-editing program developed by former UC Berkeley student and current Sun Microsystems chief scientist, Bill Joy. Before doffing his “halo,” Stallman pokes fun at the rival program. “People sometimes ask me if it is a sin in the Church of Emacs to use vi,” he says. “Using a free version of vi is not a sin; it is a penance. So happy hacking.”

圣·IGNUcius的演出以一个圈内冷笑话做结尾。在大多数UNIX或者类UNIX系统上，和Emacs齐名的另外一款编辑器名为vi。vi编辑器最早是由比尔·乔伊（Bill Joy）开发。他当年是加州大学伯克利分校的学生，如今是Sun微系统公司的首席科学家。理查德在摘掉自己头顶的“光环”之前，给vi这个竞争对手开了个小玩笑。“人们会问，在Emacs教会中使用vi是否算作原罪。倘若使用的是自由版的vi，那不能称之为罪。可以算是自罚赎罪。那么，Happy Hacking!”

After a brief question-and-answer session, audience members gather around Stallman. A few ask for autographs. “I’ll sign this,” says Stallman, holding up

one woman's print out of the GNU General Public License, "but only if you promise me to use the term GNU/Linux instead of Linux" (when referring to the system), "and tell all your friends to do likewise."

一个简短的提问环节过后，听众们围在理查德周围，有些人向他索要签名。理查德拿着一位女士打印出来的GNU通用许可证，说："我可以答应你在上面签名，不过你要答应我，在指代那整个操作系统的时候，使用GNU/Linux这个词。并且要告诉你的朋友也这么做。"

The comment merely confirms a private observation. Unlike other stage performers and political figures, Stallman has no "off" mode. Aside from the St. IGNUcius character, the ideologue you see onstage is the ideologue you meet backstage. Later that evening, during a dinner conversation in which a programmer mentions his affinity for "open source" programs, Stallman, between bites, upbraids his tablemate: "You mean free software. That's the proper way to refer to it."

他的这个举动倒是证实了我的想法。和演员，政客不同，理查德的行为会始终如一，不会切换到"台下模式"。在圣·IGNUcius这个角色中看到的那份理想主义，会一直体现在理查德的每个言行之上。在演讲之后的晚餐上，一位程序员向理查德提起了自己参与"开源"软件的经历，理查德还没顾得上咽下嘴里的食物，说道："你是说自由软件吧。用自由软件这个词恐怕更恰当。"

During the question-and-answer session, Stallman admits to playing the pedagogue at times. "There are many people who say, 'Well, first let's invite people to join the community, and then let's teach them about freedom.' And that could be a reasonable strategy, but what we have is almost everybody's inviting people to join the community, and hardly anybody's teaching them about freedom once they come in."

在演讲结束后的提问环节，理查德承认自己很多时候有些咬文嚼字。"很多人会劝我：'咱们先把人邀请进我们的圈子，然后再教会他们什么是自由。'这也许是个好策略，但是我们当下的情况是，基本所有人都在邀请朋友们进到这个圈子里，可却没有几个人会告诉朋友自由是什么。"

The result, Stallman says, is something akin to a third-world city. "You have millions of people moving in and building shantytowns, but nobody's working on step two: getting them out of those shantytowns. If you think talking about software freedom is a good strategy, please join in doing step two. There are

plenty working on step one. We need more people working on step two."

最后的结果，用理查德的话说，就是好比很多第三世界国家建立的城市。“你把百万千万的人扔到城市中，结果是在城市里建起了成千上万的城中村，贫民窟。但是没人开始着手做下一步的工作：把城中村，贫民窟里的人解救出来。如果你觉得讨论软件用户的自由是有意义的，那么请加入我们，开始做第二步工作。很多人都在做第一步，但这第二步却无人问津。”

Working on "step two" means driving home the issue that freedom, not acceptance, is the root issue of the free software movement. Those who hope to reform the proprietary software industry from the inside are on a fool's errand. "Change from the inside is risky," Stallman stays. "Unless you're working at the level of a Gorbachev, you're going to be neutralized."

这“第二步”工作直指自由软件运动的核心：自由软件的重点并非是装机量用户量，而是赋予用户自由。有些人希望能从内部瓦解整个专有软件工业，这实在是痴人梦话。理查德说：“从内部瓦解实在不现实，除非你能像当年戈尔巴乔夫那样攀到高位，否则你多半会被同化。”

Hands pop up. Stallman points to a member of the golf shirt-wearing contingent. "Without patents, how would you suggest dealing with commercial espionage?"

又有很多人举手提问，理查德随机选择了一位身穿高尔夫衬衫的听众，这位听众问道：“如果没有专利，你会建议如何防止商业间谍？”

"Well, those two questions have nothing to do with each other, really," says Stallman.

理查德答：“嗯……这其实是两个互不相关的问题。”

"But I mean if someone wants to steal another company's piece of software."

提问者补充道：“我是说，如果有人想要窃取另一个公司的软件。”

Stallman's recoils as if hit by a poisonous spray. "Wait a second," Stallman says. "Steal? I'm sorry, there's so much prejudice in that statement that the only thing I can say is that I reject that prejudice." Then he turns to the substance of the question. "Companies that develop nonfree software and other things keep lots and lots of trade secrets, and so that's not really likely to

change. In the old days – even in the 1980s – for the most part programmers were not aware that there were even software patents and were paying no attention to them. What happened was that people published the interesting ideas, and if they were not in the free software movement, they kept secret the little details. And now they patent those broad ideas and keep secret the little details. So as far as what you're describing, patents really make no difference to it one way or another."

理查德立即反应道："等等，你说'窃取'？不好意思，你的这番陈述里包含太多偏见了。我只能说，我很难认同这些偏见。"接着，理查德转向问题本身："各个公司，它们可能会开发专有软件，或者开发任何别的东西。无论开发什么，多少都会有些商业机密，它们无论如何都会保守这些机密。这个事实恐怕很难改变。在早年间——哪怕到了八十年代——绝大部分程序员都没想到过把软件去申请专利。人们会发表文章记录自己有趣的想法，如果他们不认同自由软件的理念，他们可能会在文章中甚少提及各种细节。如今，他们同样写篇文章，笼统地记录下那些想法，依然保守着各种细节。如今不同的是，他们把这文章拿去专利局，就可以申请下专利。软件专利并没能把这个事实改变多少。"

"But if it doesn't affect their publication," a new audience member jumps in, his voice trailing off almost as soon as he starts speaking.

"但如果不会影响他们公开发表的文章……"另一位听众起立发言，但还没说完，就被打断了。

"But it does," Stallman says. "Their publication is telling you that this is an idea that's off limits to the rest of the community for 20 years. And what the hell good is that? Besides, they've written it in such a hard way to read, both to obfuscate the idea and to make the patent as broad as possible, that it's basically useless looking at the published information [in the patent] to learn anything anyway. The only reason to look at patents is to see the bad news of what you can't do."

"但它的确影响了。"理查德说："在版权局能查到的公开文章就是告诉你，从现在开始往后20年，这个想法任何人都不能随便用。这能算是什么好处吗？还请注意，他们在文章中，会把想法描述得像天书一般难懂，而且会尽量把话说得模棱两可，这样就可以让这版权覆盖尽可能多的地方。没有几个人会从那些文章里去学习哪个版权所保护的技术。你能从那些文章中了解到的信息，只能告诉你什么技术你不能用。"

The audience falls silent. The speech, which began at 3:15, is now nearing the 5:00 whistle, and most listeners are already squirming in their seats, antsy to get a jump start on the weekend. Sensing the fatigue, Stallman glances around the room and hastily shuts things down. “So it looks like we’re done,” he says, following the observation with an auctioneer’s “going, going, gone” to flush out any last-minute questioners. When nobody throws their hand up, Stallman signs off with a traditional exit line.

听众一时安静下来。这个演讲从下午3:15开始，现在已经将近五点。下面很多听众已经有些坐不住了。理查德也察觉到了听众的倦意，他目光扫视了全场，说道：“那么，今天就到这儿吧。”他又停顿了一下，等到确认没有人有问题之后，理查德又重复了一遍那句口头语：

“Happy hacking,” he says.

“Happy Hacking。”理查德说。

The GNU General Public License
GNU通用公共许可证

By the spring of 1985, Richard Stallman had produced the GNU Project’s first useful result – a Lisp-based version of Emacs for Unix-like operating systems. To make it available to others as free software, he had to develop the way to release it – in effect, the follow-on for the Emacs Commune.

1985年新春伊始，理查德·斯托曼完成了GNU工程的第一个完整的软件：适用于Unix系统的Emacs编辑器。这个编辑器用Lisp语言实现。为了能让别的用户可以自由使用这个软件，他需要基于早期的“Emacs公社”探索一种全新的发布方式。

The tension between the freedom to modify and authorial privilege had been building before Gosmacs. The Copyright Act of 1976 had overhauled U.S. copyright law, extending the legal coverage of copyright to software programs. According to Section 102(b) of the Act, individuals and companies could copyright the “expression” of a software program but not the “actual processes or methods embodied in the program.”

在Gosmacs之前，自由修改软件与软件作者的特权之间就已经形成了一种对立的关系。在1976的版权运动中，修订后的美国版权法案把适用范围扩展到了软件作品上。根据该法案的第102(b)节的规定，个人或公司可以对软件程序的“实现方式”声明版权，但不能对“程序中的实际处理过程或方法”声明版权。

Translated, this treated a program much like an algebra textbook: its author can claim copyright on the text but not on the mathematical ideas of algebra or the pedagogical technique employed to explain it. Thus, regardless of what Stallman said about using the code of the original Emacs, other programmers were legally entitled to write their own implementations of the ideas and commands of Emacs, and they did. Gosmacs was one of 30-odd imitations of the original Emacs developed for various computer systems.

换句话说，如果以几何教科书为例：作者可以对书籍的文本声明版权，但不能对几何中的数学原理或用于解释这些原理的教学描述声明版权。所以，不管理查德对Emacs的使用做如何的规定，其它的程序员都可以合法的为Emacs开发一些新的功能或命令，而且，他们确实也这样做了。Gosmacs就是为不同系统开发的30多种基于原始Emacs的衍生版本之一。

The Emacs Commune applied only to the code of the original Emacs program written by Stallman himself. Even if it had been legally enforced, it would not have applied to separately developed imitations such as Gosmacs. Making Gosmacs nonfree was unethical according to the ethical ideas of the free software movement, because (as proprietary software) it did not respect its users’ freedom, but this issue had nothing to do with where the ideas in Gosmacs came from.

“Emacs公社”只适用于理查德本人所写的原始Emacs版本。即使通过法律的手段来实施这种协议，它也不适用于类似于Gosmacs这种独立开发的衍生版本。依据自由软件运动的哲学，把Gosmacs变成非自由的软件是一件不道德的事情，因为作为一个私有软件，它将无法让它的用户享有自由的权力。TODO但这个问题与Gosmacs内在的思想的最初来源没有任何的关系。

Under copyright, programmers who wanted to copy code from an existing program (even with changes) had to obtain permission from the original developer. The new law applied copyright even in the absence of copyright notices – though hackers generally did not know this – and the copyright notices too began appearing.

在版权的保护下，如果有人想从一个程序中直接复用代码（包括修改后复用），都需要得到程序原始作者的许可。新的版权法案适用于所有软件作品，即使这个软件没有明确的声明版权。早期的黑客们并不清楚这一点，但为软件作品添加版权声明的做法却从那个时候开始慢慢得以普及。

Stallman saw these notices as the flags of an invading, occupying army. Rare was the program that didn't borrow source code from past programs, and yet, with a single stroke of the president's pen, the U.S. government had given programmers and companies the legal power to forbid such reuse. Copyright also injected a dose of formality into what had otherwise been an informal system. Simply put, disputes that had once been settled hacker-to-hacker were now to be settled lawyer-to-lawyer. In such a system, companies, not hackers, held the automatic advantage. Some saw placing one's name in a copyright notice as taking responsibility for the quality of the code, but the copyright notice usually has a company's name, and there are other ways for individuals to say what code they wrote.

理查得把这些版权声明的出现看成是一个不好的兆头。那时候，大部分程序都会从以前的程序中借鉴一些代码，然而，总统的大笔一挥，美国政府就剥夺了程序员和软件公司重用其它软件代码的权力。版权法案向原本不太正式的软件版权领域注入了一剂强心针。简单的说，这项法案使得原先黑客之间的较量变成了律师之间的较量。在这样的法律框架下，软件公司相对于黑客来说占据了天然的有利地位。有些人把在版权声明加入自己的名字看成是对代码质量的一种担保，不过大多数情况下，版权声明中都只会写公司的名字，对于个人来说，只能使用其它的方式来标记他们写过的代码。

However, Stallman also noticed, in the years leading up to the GNU Project, that copyright allowed an author to grant permission for certain activities covered by copyright, and place conditions on them too. "I had seen email messages with copyright notices plus simple 'verbatim copying permitted' licenses," he recalls. "Those definitely were [an] inspiration." These licenses carried the condition not to remove the license. Stallman's idea was to take this a few steps further. For example, a permission notice could allow users to redistribute even modified versions, with the condition that these versions carry the same permission.

然而，在领导GNU工程的这些年中，斯托曼注意到，版权法案使得软件的作者可以在其保护的范围中允许他人进行一些相关的活动，也可以对这些活动规定

一些条件。他回忆说：“我见过一些附有版权声明的电子邮件，并且标明‘允许原样复制’。这给我了一些灵感。”这些许可证中通常带有一个条款，那就是在任何情况下需要保留这个许可证文本。斯托曼决定对此发扬光大。比如，可以允许用户任何分发软件，甚至分发修改后的版本，但是这些版本都必须具备相同的许可证，允许再次分发。

Thus Stallman concluded that use of copyright was not necessarily unethical. What was bad about software copyright was the way it was typically used, and designed to be used: to deny the user essential freedoms. Most authors imagined no other way to use it. But copyright could be used in a different way: to make a program free and assure its continued freedom.

所以，斯托曼认为，用版权来保护软件作品并不是不道德的。只不过是人们使用版权的方式不太好：以版权保护的名义剥夺用户的自由。版权的出现似乎就是为了这个目的，大部分的软件作者也想不到别的使用版权的方式。但是，其实版权完全可以用其它的方式来使用：用版权来保证软件的自由，并保证它一直是自由的。

By GNU Emacs 16, in early 1985, Stallman drafted a copyright-based license that gave users the right to make and distribute copies. It also gave users the right to make and distribute modified versions, but only under the same license. They could not exercise the unlimited power of copyright over those modified versions, so they could not make their versions proprietary as Gosmacs was. And they had to make the source code available. Those conditions closed the legal gap that would otherwise allow restricted, nonfree versions of GNU Emacs to emerge.

1985年，GNU Emacs 16发布，斯托曼起草了一份基于版权的许可证，允许用户随意复制并分发该软件。他还允许软件的用户分发修改后软件，但是要求分发后的软件与原来的软件使用相同的许可证。其它人不能夺取修改后的软件的版本，也就 不能像Gosmacs一样把软件变为私有软件。并且，许可证还要求软件的源代码必须是可以获取的。这些限制条件就杜绝了非自由的GNU Emacs版本的出现。

Although helpful in codifying the social contract of the Emacs Commune, the early GNU Emacs license remained too “informal” for its purpose, Stallman says. Soon after forming the Free Software Foundation he began working on a more airtight version, consulting with the other directors and with the attorneys who had helped to set it up.

斯托曼认为，Emacs 16的版权声明虽然为Emacs公社树立了一个很好的样板，但它的内容还“不够正式”。不久以后，自由软件基金会开始起草一份严密的许可证，基金会的一行管理人员和帮助创建基金会的律师也都参与其中。

Mark Fischer, a Boston copyright attorney who initially provided Stallman's legal advice, recalls discussing the license with Stallman during this period. "Richard had very strong views about how it should work," Fischer says, "He had two principles. The first was to make the software absolutely as open as possible." (By the time he said this, Fischer seems to have been influenced by open source supporters; Stallman never sought to make software "open.") "The second was to encourage others to adopt the same licensing practices." The requirements in the license were designed for the second goal.

马克·费雪是波士顿一名专注于知识产权法的律师，他回忆起当年与斯托曼讨论许可证问题的情景：“对于许可证的用处，斯托曼有非常清醒的认识。他有两条基本原则，第一条是要让软件变得尽可能的开放（费雪说这番话时，也许是受到了开源运动中一些表述的影响，斯找曼本人从来没有打算让软件变得“开放”），第二条是尽可能鼓励其它人接受和使用这种对软件不多加限制的软件许可证。”许可证中很多的条款都是为了符合第二条原则而设立的。

The revolutionary nature of this final condition would take a while to sink in. At the time, Fischer says, he simply viewed the GNU Emacs license as a simple trade. It put a price tag on GNU Emacs' use. Instead of money, Stallman was charging users access to their own later modifications. That said, Fischer does remember the license terms as unique.

这最后一个条款是一个革命性的创造，人们需要一段时间才能真正理解。费雪回忆说，在那个时候，他只是把这个条款看成是GNU Emcas与用户之间达成的一成交易，也是相当于是给GNU Emacs打上了价格标签。斯托曼并不向软件的最终用户收取费用，但用户需要把自己对软件修改贡献出来作为他们所付出的代价。费雪认为，这条合同条款非常特别——

“I think asking other people to accept the price was, if not unique, highly unusual at that time," he says.

“在那个时候，让软件用户接受这样的一种软件‘价格’即便不是独一无二，也是数一数二的。”费雪说。

In fashioning the GNU Emacs license, Stallman made one major change to

the informal tenets of the old Emacs Commune. Where he had once demanded that Commune members send him all the changes they wrote, Stallman now demanded only that they pass along source code and freedom whenever they chose to redistribute the program. In other words, programmers who simply modified Emacs for private use no longer needed to send the source-code changes back to Stallman. In a rare alteration of free software doctrine, Stallman slashed the “price tag” for free software. Users could innovate without Stallman looking over their shoulders, and distribute their versions only when they wished, just so long as all copies came with permission for their possessors to develop and redistribute them further.

.....

Stallman says this change was fueled by his own dissatisfaction with the Big Brother aspect of the original Emacs Commune social contract. As much as he had found it useful for everyone to send him their changes, he came to feel that requiring this was unjust.

斯托曼说，作出这样的改动是因为他本人对于原来Emacs公社条款中自己的“老大哥”角色感到不满。仅管以前有很多用户会向他发来很多有用的改动，他仍然觉得强调要求用户这么做是不公平的。

“It was wrong to require people to publish all changes,” says Stallman. “It was wrong to require them to be sent to one privileged developer. That kind of centralization and privilege for one was not consistent with a society in which all had equal rights.”

“要求用户公开所有的改动是不对的，更何况还要求他们把这么多的改动都发给某一个拥有特权的开发者。这种中央集权的方式，与整个社区人人平等的氛围格格不入。”

The GNU Emacs General Public License made its debut on a version of GNU Emacs in 1985. Following the release, Stallman welcomed input from the general hacker community on how to improve the license’s language. One hacker to take up the offer was future software activist John Gilmore, then working as a consultant to Sun Microsystems. As part of his consulting work, Gilmore had ported Emacs over to SunOS, the company’s in-house version of Unix. In the process of doing so, Gilmore had published the changed version under the GNU Emacs license. Instead of viewing the license as a liability,

Gilmore saw it as clear and concise expression of the hacker ethos. “Up until then, most licenses were very informal,” Gilmore recalls.

1985年，斯托曼发布GNU Emacs的新版本时，GNU Emacs通用公共许可证第一次与公众见面。随后，斯托曼接受了很多来自黑客社区的对改进许可证用语的修改意见。约翰·吉尔摩是最早接受这种软件许可证的黑客之一，他后来成为了一名软件活动家并为Sun公司提供咨询服务。作为他工作的一部分，吉尔摩把Emacs移植到了SunOS上，SunOS是Sun公司内部的一种Unix发行版。在移植的过程中，吉尔摩按照GNU Emacs许可证的要求把他所作的修改都进行了公开。吉尔摩并不认为遵守GNU Emacs许可证是对他工作的一种限制，相反的，他认为这种许可证正好体现了黑客的精神气质。“不过，在那个时候，很多的许可证的条款都还不是很正式。”吉尔摩回忆道。

As an example of this informality, Gilmore cites the mid-1980s copyright license of trn, a news reader program written by Larry Wall, a hacker who could go onto later fame as the creator of both the Unix “patch” utility and the Perl scripting language. In the hope of striking a balance between common hacker courtesy and an author’s right to dictate the means of commercial publication, Wall used the program’s accompanying copyright notice as an editorial sounding board.

吉尔摩引用了一个Unix工具软件trn的版权声明，从中也可以看出那时的许可证有多么的不正式。trn是拉里·华尔的一个作品，他后来还开发了Perl语言。有了Patch工具，Unix程序可以很容易地把自己的代码以补丁的型式插入到一个大型软件中。华尔很清楚这一点，所以他把下面的版权声明写入了软件的README文档：

> Copyright (c) 1985, Larry Wall\ You may copy the trn kit in whole or in part as long as you don’t try to make money off it, or pretend that you wrote it.
>
> 版权所有 (C) 1985，拉里·华尔\ 你可以任意的复制trn软件包或它的一部分，只要你保证不以它来牟利，并且不得假装你是这个软件的作者。

Such statements, while reflective of the hacker ethic, also reflected the difficulty of translating the loose, informal nature of that ethic into the rigid, legal language of copyright. In writing the GNU Emacs license, Stallman had done more than close up the escape hatch that permitted proprietary offshoots. He had expressed the hacker ethic in a manner understandable to

both lawyer and hacker alike.

如此的声明，映射着黑客的精神和气质，也映射出要把这种松散、非正式的精神转换为严谨的法律版权语言的困难之处。在撰写GNU Emacs许可证时，斯托曼做了很多的工作，保证许可证的表述中不存在任何可能被别人利用，把这个软件变成私有软件的漏洞。他以一种律师和黑客都可以理解的方式，体现出了黑客的精神。

It wasn't long, Gilmore says, before other hackers began discussing ways to "port" the GNU Emacs license over to their own programs. Prompted by a conversation on Usenet, Gilmore sent an email to Stallman in November, 1986, suggesting modification:

没过多久，就有一些黑客们开始讨论如何把GNU Emacs许可证移植到他们自己的其它软件作品上。1986年11月的一天，吉尔摩受到Usenet上一个贴子的启发，写了一封电子邮件给斯托曼，建议对GNU Emacs许可证做一些修改：

> You should probably remove "EMACS" from the license and replace it with "SOFTWARE" or something. Soon, we hope, Emacs will not be the biggest part of the GNU system, and the license applies to all of it.
>
> 或许你可以考虑把许可证中的"EMACS"字眼替换为"软件"或其它类似的名词。我们希望在不久的将来，Emacs就不再是GNU系统中最大的一部分，但这种许可证可以适用于整个系统。

Gilmore wasn't the only person suggesting a more general approach. By the end of 1986, Stallman himself was at work with GNU Project's next major milestone, the source-code debugger GDB. To release this, he had to modify the GNU Emacs license so it applied to GDB instead of GNU Emacs. It was not a big job, but it was an opening for possible errors. In 1989, Stallman figured out how to remove the specific references to Emacs, and express the connection between the program code and the license solely in the program's source files. This way, any developer could apply the license to his program without changing the license. The GNU General Public License, GNU GPL for short, was born. The GNU Project soon made it the official license of all existing GNU programs.

吉尔摩并不是唯一一个建议用这种更通用做法的人。1986年年底，斯托曼正着手开发GNU工程中的下一个里程碑——源代码调试器，与此同时，他也同样在

考虑如何修改Emacs许可证试它可以适用于Emacs和这个新的调试器。斯托曼的解决方案是：移除许可证中所有的Emcas字眼并把它转变为一个保护所有GNU软件版权的许可证。GNU 通用公共许可证，缩写为GPL，就这样诞生了。

In publishing the GPL, Stallman followed the software convention of using decimal numbers to indicate versions with minor changes and whole numbers to indicate versions with major changes. The first version, in 1989, was labeled Version 1.0. The license contained a preamble spelling out its political intentions:

在打造GPL时，斯托曼使用了软件版本号的表示方式，用小数表示原型版本的许可证，用整数表示成熟的版本。1989年，斯托曼打入Unix阵营的两个重要作品，也就是GNU调试器发布的一年后，他发布了GPL的1.0版本（斯托曼从1985年就开始了GPL这个项目）。这个许可证包含了一个表明其政治态度的序言：

> The General Public License is designed to make sure that you have the freedom to give away or sell copies of free software, that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.
>
> To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.
>
> 通用公共许可证被设计成确保你拥有分发或出售自由软件的权力，确保你可以获取软件的源代码，确保你可以修改软件或在别的自由软件中使用这个软件，并确保你了解你拥有以上的权力。
>
> 为了保证你的权力，我们禁止任何人剥夺你的以上权力或要求你放弃这些权力。这些限制条件也作为你分发或修改这个软件时所必须承担的责任。

As hacks go, the GPL stands as one of Stallman’s best. It created a system of communal ownership within the normally proprietary confines of copyright law. More importantly, it demonstrated the intellectual similarity between legal code and software code. Implicit within the GPL’s preamble was a profound message: instead of viewing copyright law with suspicion, hackers should view

it as a dangerous system that could be hacked.

在打造GPL时，斯托曼也不得不对原来Emacs Commune的非正式协议进行修改。在原来Emacs Commune的协议中，社区的成员需要公开任何对Emacs所进行的改动，而现在，程序员只需跟斯托曼一样公开那些与相同协议公开的软件的衍生作品的代码即可。斯托曼粉碎了自由软件价格标签，免得它们破坏了自由软件的精神教义。用户可以不受斯托曼的限制对软件进行创新工作，并保证了整个社区都可以获得相同的软件版本。.....

“The GPL developed much like any piece of free software with a large community discussing its structure, its respect or the opposite in their observation, needs for tweaking and even to compromise it mildly for greater acceptance,” says Jerry Cohen, another attorney who advised Stallman after Fischer departed. “The process worked very well and GPL in its several versions has gone from widespread skeptical and at times hostile response to widespread acceptance.”

“GPL的设计过程与其它自由软件开发的过程非常相似，由一个大的社区共同来讨论它的结构、它的条款。社成员的意见可能会相左，所以还需要对它的条款进行调整甚至进行妥协，这样才能更容易为大众所接受。”另一位曾帮助斯托曼一起设计GPL的律师杰里·科恩说，“这样设计方式非常有效，GPL的各个版本从被人们怀疑逐渐变为广为接受。”

In a 1986 interview with *BYTE* magazine, Stallman summed up the GPL in colorful terms. In addition to proclaiming hacker values, Stallman said, readers should also “see it as a form of intellectual jujitsu, using the legal system that software hoarders have set up against them.” Years later, Stallman would describe the GPL’s creation in less hostile terms. “I was thinking about issues that were in a sense ethical and in a sense political and in a sense legal,” he says. “I had to try to do what could be sustained by the legal system that we’re in. In spirit the job was that of legislating the basis for a new society, but since I wasn’t a government, I couldn’t actually change any laws. I had to try to do this by building on top of the existing legal system, which had not been designed for anything like this.”

.....

About the time Stallman was pondering the ethical, political, and legal issues associated with free software, a California hacker named Don Hopkins mailed

him a manual for the 68000 microprocessor. Hopkins, a Unix hacker and fellow science-fiction buff, had borrowed the manual from Stallman a while earlier. As a display of gratitude, Hopkins decorated the return envelope with a number of stickers obtained at a local science-fiction convention. One sticker in particular caught Stallman's eye. It read, "Copyleft (L), All Rights Reversed." Stallman, inspired by the sticker, nicknamed the legal technique employed in the GNU Emacs license (and later in the GNU GPL) "Copyleft," jocularly symbolized by a backwards "C" in a circle. Over time, the nickname would become general Free Software Foundation terminology for any copyright license "making a program free software and requiring all modified and extended versions of the program to be free software as well."

正当斯托曼在思考自由软件有关伦理、政治和法律上的问题时，一位名叫Don Hopkins的加州黑客给他寄了一份68000微处理器的手册。Hopkins是一名Unix黑客，同时也是一名科幻小说爱好者，他早些时候向斯托曼借阅了这本书。为了表达对斯托曼的感激之情，他在信封上贴了一些从当地科幻小说爱好者社团获得的一些贴纸。其中，有一张贴纸吸引了斯托曼的眼球。"Copyleft (L)，保留所有权利。"在GPL的第一个版本发布以后，斯托曼称赞这张贴纸给了他灵感，并把Copyleft作为自由软件许可证的昵称。以后，Copyleft这个名字以及它的标识——一个倒着写的C，成为了自由软件基金官方对GPL的同义词。

The German sociologist Max Weber once proposed that all great religions are built upon the "routinization" or "institutionalization" of charisma. Every successful religion, Weber argued, converts the charisma or message of the original religious leader into a social, political, and ethical apparatus more easily translatable across cultures and time.

德国著名社会学家Max Web曾经说过，所有伟大的宗教都是建立在对神的感召力的常规化或制度化的基础上的。他认为，每一种成功的宗教，都是把神的感召力或这个宗教最初创始人的想法转换成一种社会化、政治化和伦理化的东西，这样才可能长时间在不同的文化之间进行传递。

While not religious per se, the GNU GPL certainly qualifies as an interesting example of this "routinization" process at work in the modern, decentralized world of software development. Since its unveiling, programmers and companies who have otherwise expressed little loyalty or allegiance to Stallman have willingly accepted the GPL bargain at face value. Thousands have also accepted the GPL as a preemptive protective mechanism for their

own software programs. Even those who reject the GPL conditions as too limiting still credit it as influential.

如果不考虑其宗教本质，GNU GPL可以看成是现代的去中心化软件开发模式中一种典型的常规化的例子。程序员和软件公司即使不信奉斯托曼，也会很乐意去接受GPL表面上所呈现出来的公开的价值。一部分人把GPL看成是对自己软件作品的先发制人的保护机制而接受它，另一部分人虽然由于GPL的条款过于苛刻而反对它，但依然把它看成是一种很有影响力的担保方式。

One hacker falling into this latter group was Keith Bostic, a University of California employee at the time of the GPL 1.0 release. Bostic's department, the Computer Systems Research Group (SRG), had been involved in Unix development since the late 1970s and was responsible for many key parts of Unix, including the TCP/IP networking protocol, the cornerstone of modern Internet communications. By the late 1980s, AT&T, the original owner of the Unix software, began to focus on commercializing Unix and began looking to the Berkeley Software Distribution, or BSD, the academic version of Unix developed by Bostic and his Berkeley peers, as a key source of commercial technology.

Keith Bostic就是属于后一个群体中的一名黑客，GPL 1.0发布的时候，他是University of California的一名雇员。Bostic所就职的Computer System Research Group(SRG)系，从二十世纪七十年代开始就从事Unix系统的开发，并且负责Unix中很多重要组件的开发。比如，作为现代Internet通信基石的TCP/IP网络协议。到了八十年代，Unix商标所有者AT&T开始专注于Unix的商业化，他们选择了由Bostic和他在伯克利的同事们一起开发的Berkeley Software Distribution(BSD)这个Unix的学术分枝作为商用Unix的主要技术来源。

The code written by Bostic and friends was off limits to nearly everyone, because it was intermixed with proprietary AT&T code. Berkeley distributions were therefore available only to institutions that already had a Unix source license from AT&T. As AT&T raised its license fees, this arrangement, which had at first seemed innocuous (to those who thought only of academia) became increasingly burdensome even there. To use Berkeley's code in GNU, Stallman would have to convince Berkeley to separate it from AT&T's code and release it as free software. In 1984 or 1985 he met with the leaders of the BSD effort, pointing out that AT&T was not a charity and that for a university to donate its work (in effect) to AT&T was not proper. He asked them to separate

out their code and release it as free software.

虽然伯克利BSD的源代码在研究人员和商业软件程序员群体中广为分享，把它商业化仍然是一个大问题。因为伯克利的代码中也混入了很多AT&T的私有代码。所以伯克利版本只能在那些已经向AT&T购买了Unix源代码许可的机构使用。当AT&T提高了它的软件许可费用后，这种看上去无辜的授权协议，越来越成为一种负担。

Hired in 1986, Bostic had taken on the personal project of porting the latest version of BSD to the PDP-11 computer. It was during this period, Bostic says, that he came into close interaction with Stallman during Stallman's occasional forays out to the west coast. "I remember vividly arguing copyright with Stallman while he sat at borrowed workstations at CSRG," says Bostic. "We'd go to dinner afterward and continue arguing about copyright over dinner."

1986年，Bostic受到雇佣，完成一个把BSD移植到DEC的PDP-11计算机上的个人项目。Bostic回忆说，在这段时间里，斯托曼有时会突然来到西海岸，所以他有机会与斯托曼进行一些密切的接触。"我直到现在还生动的记得当时斯托曼坐在CSRG一台借来的工作站上与我争论有关软件版权的问题。争论完了，我们就一起去吃晚餐，然后在晚餐过程中继续争论有关版权的问题。"博斯蒂克不无得意地说。

The arguments eventually took hold, although not in the way Stallman would have preferred. In June, 1989, Berkeley had separated its networking code from the rest of the AT&T-owned operating system and began distributing it under a copyright-based free license. The license terms were liberal. All a licensee had to do was give credit to the university in advertisements touting derivative programs. In contrast to the GPL, this license permitted proprietary offshoots. One problem limited the use of the BSD Networking release: it wasn't a complete operating system, just the network-related parts of one. While the code would be a major contribution to any free operating system, it could only be run at that time in conjunction with other, proprietary-licensed code.

这样的争论最终停止了，但并不是以一种斯托曼所喜欢的方式。1989年6月，伯克利把Unix中网格相关的代码从AT&T拥有版权的部分代码剥离，并以加州大学许可证的形式发布。许可证的条款非常的开明，代码的的使用方只需在衍生代码中注明原始代码出自于加州大学即可。与GPL不同，对代码的进行商业化的利用是被允许的。唯一制约这种许可证广为应用的原因是：BSD的网络部分并

不是一个完整的操作系统。人家可以学习这份代码，但它只能与其它私有代码放到一起才能真正运行起来。

Over the next few years, Bostic and other University of California employees worked to replace the missing components and turn BSD into a complete, freely redistributable operating system. Although delayed by a legal challenge from Unix Systems Laboratories – the AT&T spin-off that retained ownership of the Unix code – the effort would finally bear fruit in the early 1990s. Even before then, however, many of the Berkeley network utilities would make their way into Stallman's GNU system.

在接下来的几年中，Bostic和其它加州大学的同事们一起，重新编写BSD中所缺少的部分，把BSD转换一个完整的可以自由分发的操作系统。虽然这项工作由于AT&T对于Unix这个品牌的所有权而陷入了一些法律上的困境，但是到了九十年代初，这个项目几乎已经完成了。甚至在这之前，很多伯克利开发的工具软件已经开始进入到斯托曼的GNU工程中。

"I think it's highly unlikely that we ever would have gone as strongly as we did without the GNU influence," says Bostic, looking back. "It was clearly something where they were pushing hard and we liked the idea."

Bostic回忆说："如果没有GNU的影响，我想我们不可能如此坚定的走得那么远。GNU中有一种力量在推动我们的前进，那就是它让我们很欣赏的理念。"

By the end of the 1980s, the GPL was beginning to exert a gravitational effect on the free software community. A program didn't have to carry the GPL to qualify as free software – witness the case of the BSD network utilities – but putting a program under the GPL sent a definite message. "I think the very existence of the GPL inspired people to think through whether they were making free software, and how they would license it," says Bruce Perens, creator of Electric Fence, a popular Unix utility, and future leader of the Debian GNU/Linux development team. A few years after the release of the GPL, Perens says he decided to discard Electric Fence's homegrown license in favor of Stallman's lawyer-vetted copyright. "It was actually pretty easy to do," Perens recalls.

到了八十年代末，GPL开始在自由软件社区中产生了如地心引力般的效果。一个程序要成为自由软件并不一定要以GPL发布，比如那些BSD的工具软件，但是如果以GPL来发布，就会更直接的表明自己作为自由软件身份。Bruce

Perens说：“我觉得GPL的存在，激发了人们去思考自己的软件作品是否可以成为自由软件，以及应该以什么样的许可证来发布它。” Bruce是一个知名的Unix工具软件Electric Fence的开发者，也是Debian GNU/Linux开发团队的领导。在GPL发布后的几年，Perens说他决定废弃Electric Fencen所使用的自己土法炮制的软件许可证而改用斯托曼所创造的由律师审查过的许可证。他回忆说：“事实上这是非常容易做到的。”

Rich Morin, the programmer who had viewed Stallman’s initial GNU announcement with a degree of skepticism, recalls being impressed by the software that began to gather under the GPL umbrella. As the leader of a SunOS user group, one of Morin’s primary duties during the 1980s had been to send out distribution tapes containing the best freeware or free software utilities. The job often mandated calling up original program authors to verify whether their programs were copyrighted or whether they had been consigned to the public domain. Around 1989, Morin says, he began to notice that the best software programs typically fell under the GPL license. “As a software distributor, as soon as I saw the word GPL, I knew I was home free,” recalls Morin.

Rich Morin是一名程序员，他对斯托曼最初的GNU宣言在一程度上是抱有怀疑的，但是他很惊讶的发现各种软件开始慢慢的聚集到了GPL的大伞下。作为一名 SunOS用户组的领头人，Morin在80年代的一项重要工作就是分发一些载有最好的免费软件和自由软件工具的磁带。这项工作需要去联系软件的原始作者，询问他们的作品是否有版权保护并且他们是否愿意把这个软件放入公有领域。Morin说，到了1989年左右，很多最好的软件程序都开始使用GPL许可证了。”作为一名软件分发人员，我一看到GPL这个词，我就知道我不担心版权的问题了。”Morin回忆说。

To compensate for the prior hassles that went into compiling distribution tapes to the Sun User Group, Morin had charged recipients a convenience fee. Now, with programs moving over to the GPL, Morin was suddenly getting his tapes put together in half the time, turning a tidy profit in the process. Sensing a commercial opportunity, Morin rechristened his hobby as a business: Prime Time Freeware.

为了补偿制作磁带在Sun用户组中分发的成本，Morin需要向订阅者收取一部分的费用。现在，这些程序都改用GPL了，Morin收集软件制作磁带只需要花以前一半的时间，所以从此可以开始有了一点小小的赢利。Morin有着灵敏的商业嗅

觉，自此，他把自己的爱好变成了一项事业，创办了Prime Time Freeware公司。

Such commercial exploitation was completely consistent with the free software agenda. “When we speak of free software, we are referring to freedom, not price,” advised Stallman in the GPL’s preamble. By the late 1980s, Stallman had refined it to a more simple mnemonic: “Don’t think free as in free beer; think free as in free speech.”

这样的商业探索完全是在自由软件推广计划的日程表中的。斯托曼在GPL的序言中说“当我们说自由软件这个词的时候，我们在说的是自由，而不是免费”。到了80年代后期，斯托曼用一句更容易记忆的话来表述这个观点“不要把free想象成是免费啤酒中的免费，它是自由演讲中的自由”。

For the most part, businesses ignored Stallman’s entreaties. Still, for a few entrepreneurs, the freedom associated with free software was the same freedom associated with free markets. Take software ownership out of the commercial equation, and you had a situation where even the smallest software company was free to compete against the IBMs and DECs of the world.

但是从总体上来看，商业界还是无视了斯托曼的诉求。对于少数企业家来说，自由软件所关注的自由依然与自由市场中的自由是一样的。在商业市场的方程式中拿出软件的所有者，你就算是一个最小的软件公司，也可以自由的与IBM和DEC这样的大公司自由竞争。

One of the first entrepreneurs to grasp this concept was Michael Tiemann, a software programmer and graduate student at Stanford University. During the 1980s, Tiemann had followed the GNU Project like an aspiring jazz musician following a favorite artist. It wasn’t until the release of the GNU C Compiler, or GCC, in 1987, however, that he began to grasp the full potential of free software. Dubbing GCC a “bombshell,” Tiemann says the program’s own existence underlined Stallman’s determination as a programmer.

Michale Tiemann是最初几个抓住了这个概念的企业家之一，他是一名程序员，同时也是斯坦福大学的一名研究生。在80年代，Tiemann曾经一度追随GNU工程，就像一名爵士乐的发烧友追随一名他所喜爱的艺术家一样。然后，到了1987年，GNU C编译器的发布，让他开始认识到自由软件的巨大潜力。GCC可以看成是一个“炸弹”，Tiemann说软件就存在于斯托曼作为程序员的身后。

“Just as every writer dreams of writing the great American novel, every programmer back in the 1980s talked about writing the great American compiler,” Tiemman recalls. “Suddenly Stallman had done it. It was very humbling.”

“就像所有的作家都梦想要写出一部伟大的美国小说一样，80年代时，每个程序员都在想写一个伟大的美国编译器。” Tiemman回忆说，“突然间斯托曼完成了这个工作，这实在是很让人感到羞辱的。”

“You talk about single points of failure, GCC was it,” echoes Bostic. “Nobody had a compiler back then, until GCC came along.”

“你在说的其实是一个单点故障，GCC就是一个”，Bostic回应说，“那时候没人有编译器，而GCC就是唯一的一个。”

Rather than compete with Stallman, Tiemann decided to build on top of his work. The original version of GCC weighed in at 110,000 lines of code, but Tiemann recalls the program as surprisingly easy to understand. So easy in fact that Tiemann says it took less than five days to master and another week to port the software to a new hardware platform, National Semiconductor’s 32032 microchip. Over the next year, Tiemann began playing around with the source code, creating the first “native” or direct compiler for the C++ programming language, by extending GCC to handle C++ as well as C. (The existing, proprietary implementation of the C++ language worked by converting the code to the C language, then feeding the result to a C compiler.) One day, while delivering a lecture on the program at Bell Labs, Tiemann ran into some AT&T developers struggling to pull off the same thing.

与其与斯托曼竞争，Tiemann决定在他的作品基础来进行创造。GCC最早的版本大概有110000行代码，但是在Tiemann看来，这些代码都很容易理解。他只用了不到五天的时间就掌握了这份代码，并且再花了一周时间就把它移植到了另一个新的硬件平台上——国家半导体的32032微处理器上。到了下一年，Tiemann就开始修改这些源代码，并开发了一个原生的C++语言的编译器。有一天，他在贝尔实验室教授一门编程课程时，Tiemann发现有些AT&T的开发者还在努力做相同的事情。

“There were about 40 or 50 people in the room, and I asked how many people were working on the native code compiler,” Tiemann recalls. “My host said the information was confidential but added that if I took a look around the room I

might get a good general idea."

"教室里大概有四五十人，我问他们有多少人在参与开发原生的编译器"Tiemann回忆说,"主持人说具体的开发人员的人数是机密，但是如果看看这个教室里的人，就可以得出一个大概的结论。"

It wasn't long after, Tiemann says, that the light bulb went off in his head. "I had been working on that project for six months," Tiemann says. I just thought to myself, whether it's me or the code, this is a level of efficiency that the free market should be ready to reward."

Tiemann继续回忆说，不久以后，他就灵机一动有了个好想法。"我已经在个项目上投入了6个月的时间。我开始思考，倒底是我自己还是这份代码来带来了这样的高效率，谁更应该在自由市场中获得奖励？"

Tiemann found added inspiration in the *GNU Manifesto*: while excoriating the greed of proprietary software vendors, it also encourages companies, as long as they respect users freedom, to use and redistribute free software in their commercial activities. By removing the power of monopoly from the commercial software question, the GPL makes it possible for even small companies to compete on the basis of service, which extends from simple tech support to training to extending free programs for specific clients' needs.

Tiemann从GNU宣言中获得了更多的灵感，这些内容虽然会伤害到一些软件开发商，但是它鼓励其它软件开发商可以从消费者的角度出发，更好的利用自由软件。虽然GPL让商业软件无法垄断市场，但是它让一些更有眼光的开发商可以从提供服务和咨询的角度去开拓业务、开展竞争，而这些领域其实是软件市场中利润最高的两个地方。

In a 1999 essay, Tiemann recalls the impact of Stallman's *Manifesto*. "It read like a socialist polemic, but I saw something different. I saw a business plan in disguise."

在1999年的一篇文章中，Tiemann回忆起斯托曼的宣言带来的影响。"它听上去像是一个社会主义者的争辩，但是它们还是大不相同的。我在里面发现了一个隐藏着的商业计划。"

This business plan was not new; Stallman supported himself in the late 80s by doing this on a small scale. But Tiemann intended to take it to a new level.

.....

Teaming up with John Gilmore and David Vinayak Wallace, Tiemann launched a software consulting service dedicated to customizing GNU programs. Dubbed Cygnus Support (informally, “Cygnus” was a recursive acronym for “Cygnus, Your GNU Support”), the company signed its first development contract in February, 1990. By the end of the year, the company had \$725,000 worth of support and development contracts.

..... John Gilmore是另一位GNU工程的粉丝，Tiemann和他一起成立了一个软件咨询服务，专注于提供对GNU程序的定制化服务。这就是后来的Cygnus公司，它在1990年2月时与客户签订了第一份开发合同。到1990年年底，他已经签订了价值超过72.5万美元的软件支持和开发合同。

The complete GNU operating system Stallman envisioned required more than software development tools. In the 1990s, GNU also developed a command line interpreter or “shell,” which was an extended replacement for the Bourne Shell (written by FSF employee Brian Fox, and christened by Stallman the Bourne Again Shell, or BASH), as well as the PostScript interpreter Ghostscript, the documentation browser platform Texinfo, the C Library which C programs need in order to run and talk to the system’s kernel, the spreadsheet Oleo (“better for you than the more expensive spreadsheet”), and even a fairly good chess game. However, programmers were typically most interested in the GNU programming tools.

.....

GNU Emacs, GDB, and GCC were the “big three” of developer-oriented tools, but they weren’t the only ones developed by the GNU Project in the 80s. By 1990, GNU had also generated GNU versions of the build-controller Make, the parser-generator YACC (rechristened Bison), and awk (rechristened gawk); as well as dozens more. Like GCC, GNU programs were usually designed to run on multiple systems, not just a single vendor’s platform. In the process of making programs more flexible, Stallman and his collaborators often made them more useful as well.

GNU Emacs, GDB和GCC是三个最为重要的开发工具，不过它们并不是斯托曼在GNU工程最初五年中开发的唯一软件。截止1990年，斯托曼还开发了GNU版本的Bourne Shell（改名为Bourne Again Shell, BASH），YACC（改名为

Bison）和awk（改名为gawk）。与GCC一样，这些GNU程序都被设计为可以在各种系统上运行，而不是局限于某一个开发商的平台。斯托曼和他的同事们不但把程序做得更具弹性，也把它们做得越来越有用。

Recalling the GNU universalist approach, Prime Time Freeware's Morin points to a useless but vitally important software package called GNU Hello, which serves as an example to show programmers how to properly package a program for GNU. "It's the hello world program which is five lines of C, packaged up as if it were a GNU distribution," Morin says. "And so it's got the Texinfo stuff and the configure stuff. It's got all the other software engineering goo that the GNU Project has come up with to allow packages to port to all these different environments smoothly. That's tremendously important work, and it affects not only all of [Stallman's] software, but also all of the other GNU Project software."

Prime Time Freeware公司的Morin指出了一个很简单但很重要的自由软件包"hello"，这个软件包体现了GNU软件大一统的设计理念。Morin说，"hello软件包其实里面就是一个五行代码的hello worldC语言程序，不过按GNU发行包的标准进行了打包。它还包含了Texinfo的文档和configuration脚本。这个软件包为其它的软件开发者提供了一个参考，展示了很多GNU工程是如何让软件包可以平滑的移植到不同的环境中。这件事非常重要，这不但影响到所有斯托曼的软件作品，也影响到其它所有的GNU工程的软件。"

According to Stallman, improving technically on the components of Unix was secondary to replacing them with free software. "With each piece I may or may not find a way to improve it," said Stallman to *BYTE*. "To some extent I am getting the benefit of reimplementation, which makes many systems much better. To some extent it's because I have been in the field a long time and worked on many other systems. I therefore have many ideas [which I learned from them] to bring to bear."

在斯托曼看来，开发新的软件是第一要务，而改进软件则是位于第二位的。斯托曼在接受Byte杂志采访时说："对于每一个软件作品，我也许可以去改进它。但有时重新实现整个软件也许是更好的做法，因为这样会让整个系统变得更好。在一定程度上，我在这个领域已经工作了很多年并且接触过很多不同的系统，因此，我有很多想法可以对其造成影响。"

Nevertheless, as GNU tools made their mark in the late 1980s, Stallman's AI Lab-honed reputation for design fastidiousness soon became legendary

throughout the entire software-development community.

不管怎么说，GNU的工具在80年代后期发挥出了它的优势，斯托曼的人工智能实验也以能设计出严密的程序而在整个软件开发社区中出名。

Jeremy Allison, a Sun user during the late 1980s and programmer destined to run his own free software project, Samba, in the 1990s, recalls that reputation with a laugh. During the late 1980s, Allison began using Emacs. Inspired by the program's community-development model, Allison says he sent in a snippet of source code only to have it rejected by Stallman.

Jeremy Allison，20世纪80年代一名Sun的用户，同时也是一名程序员，是Samba这个自由软件项目的维护者。他在90年代时曾经嘲笑过GNU的工具。在80年代后期，Allison开始使用Emacs，并对它的社区开发模式产生了兴趣。Allison故意给斯托曼发去了一段很可能会被他拒绝的代码。

"It was like the *Onion* headline," Allison says. ``

'Child's prayers to God answered: No.'

"

Allison说："这就像是the Onion的头条新闻，孩子们在祈祷，但是上帝说'不'"。

As the GNU Project moved from success to success in creation of user-level programs and libraries, it postponed development of the kernel, the central "traffic cop" program that controls other programs' access to the processor and all machine resources.

.....

As with several other major system components, Stallman sought a head-start on kernel development by looking for an existing program to adapt. A review of GNU Project "GNUsletters" of the late 1980s reveals that this approach, like the initial attempt to build GCC out of Pastel, had its problems. A January, 1987 GNUsletter reported the GNU Project's intention to overhaul TRIX, a kernel developed at MIT. However, Stallman never actually tried to do this, since he was working on GCC at the time; later he concluded that TRIX would require too much change to be a good starting point. By February of 1988, according to a newsletter published that month, the GNU Project had shifted

its kernel plans to Mach, a lightweight “micro-kernel” developed at Carnegie Mellon. Mach was not then free software, but its developers privately said they would liberate it; when this occurred, in 1990, GNU Project kernel development could really commence.

斯托曼作为一名程序员的名气越来越大，但他也在努力尝试转型为一名项目经理。尽管GNU工程在开发各种系统工具上捷报频传，它还是没有能够开发出一个真正可用的内核。所谓“内核”，是指在所有的Unix系统中起到交通警察作用的核心程序，它决定了各种设备和程序应该如何去使用处理器和各种资源。到了80年代末，这个情况引发了越来多的抱怨。跟其它GNU工程的软件一样，斯托曼先是试图寻找一个现有的程序，通过修改它来开展内核的开发。根据1987年1月的“GNusletter”所描述的，斯托曼已经开始尝试修改TRIX，TRIX是一个麻省理工学院开发的Unix内核。

The delays in kernel development were just one of many concerns weighing on Stallman during this period. In 1989, Lotus Development Corporation filed suit against rival software companies, Paperback Software International and Borland, for copying menu commands from Lotus’ popular 1-2-3 Spreadsheet program. Lotus’ suit, coupled with the Apple-Microsoft “look and feel” battle, endangered the future of the GNU system. Although neither suit directly attacked the GNU Project, both threatened the right to develop software compatible with existing programs, as many GNU programs were. These lawsuits could impose a chilling effect on the entire culture of software development. Determined to do something, Stallman and a few professors put an ad in *The Tech* (the MIT student newspaper) blasting the lawsuits and calling for a boycott of both Lotus and Apple. He then followed up the ad by helping to organize a group to protest the corporations filing the suit. Calling itself the League for Programming Freedom, the group held protests outside the offices of Lotus, Inc.

这个时期，内核开发的延迟只是困扰斯托曼的众多问题中的一个。1989年Lotus Development Corporation起诉了一家竞争对手，Paperback Software International，控告他抄袭了Lotus的1-2-3电子表格程序中的菜单设计。Lotus的官司，加上苹果和微软的“视觉体验”的斗争，给GNU工程带来了颇多的麻烦。虽然这两场官司跟GNU工程都没有直接的关系，因为他们都主要是有关个人电脑上的操作系统和应用软件而不是类Unix的硬件系统，但是他们对整个软件开发的文化产生了深远的影响。斯托曼觉得自己不能袖手旁观，他召集了几个程序员朋友在一本杂志上刊登了一则广告。然后他基于这则广告，成立了一个抗

议小组，反对那些参与诉讼的公司。他把自己的这个组织命名为“自由编程联盟”，他们在Lotus公司和波士顿审理Lotus这起案子的法庭外发起了抗议。

The protests were notable.

这次抗议非常有效果。

They document the evolving nature of the software industry. Applications had quietly replaced operating systems as the primary corporate battleground. In its unfinished quest to build a free software operating system, the GNU Project seemed hopelessly behind the times to those whose primary values were fashion and success. Indeed, the very fact that Stallman had felt it necessary to put together an entirely new group dedicated to battling the “look and feel” lawsuits led some observers to think that the FSF was obsolete.

他们在软件工业发展史上留下了一笔。应用软件正悄悄的取代操作系统成为软件公司主要的战场。然而，GNU工程想要建立一个自由操作系统的愿景都尚未实现，这使它看起来有点跟不上时代了。不过也正是这个原因，让斯托曼觉得有必要组织一股新的力量来对付这次对“视觉体验”诉讼，这在一些观察家眼中则更显得GNU工程正在变得过时。

However, Stallman had a strategic reason to start a separate organization to fight the imposition of new monopolies on software development: so that proprietary software developers would join it too. Extending copyright to cover interfaces would threaten many proprietary software developers as well as many free software developers. These proprietary developers were unlikely to endorse the Free Software Foundation, but there was, intentionally, nothing in the League for Programming Freedom to drive them away. For the same reason, Stallman handed over leadership of LPF to others as soon as it was feasible.

.....

In 1990, the John D. and Catherine T. MacArthur Foundation certified Stallman’s genius status when it granted Stallman a MacArthur fellowship, the so-called “genius grant,” amounting in this case to \$240,000 over 5 years. Although the Foundation does not state a reason for its grants, this one was seen as an award for launching the GNU Project and giving voice to the free software philosophy. The grant relieved a number of short-term concerns for

Stallman. For instance, it enabled him to cease the consulting work through which he had obtained his income in the 80s and devote more time to the free software cause.

1990年，John D. 和Catherine T. MacArthur基金会授予了斯托曼MacArthur院士的称号，嘉奖他的天才能力，并授予他“天才奖金”。这份24万美元的奖金用于资助GNU工程并声援自由软件哲学，这帮助GNU工程缓解了些短期的困难。最重要的一点就是，它让斯托曼这个在FSF不拿薪水而靠做咨询来养活自己的员工，可以有更多的时间专注于编写GNU的代码。

The award also made it possible for Stallman to register normally to vote. In 1985 a fire in the house where Stallman lived left him without an official domicile. It also covered most of his books with ash, and cleaning these “dirty books” did not yield satisfying results. From that time he lived as a “squatter” at 545 Technology Square, and had to vote as a “homeless person.” “[The Cambridge Election Commission] didn’t want to accept that as my address,” Stallman would later recall. “A newspaper article about the MacArthur grant said that, and then they let me register.”

略带讽刺意味的是，这个奖励也赋予了斯托曼选举权。在获奖前不久，斯托曼的公寓的一场大火几乎让他损失了一切财产。在获奖的时候，斯托曼还只是在545科技广场存身的一个流浪汉。“选举中心不同意我把那里作为我的住址，”斯托曼回忆说，“不过报纸上那篇MacArthur奖新闻也么形容我，于是他们就让我注册了。”

Most importantly, the MacArthur fellowship gave Stallman press attention and speaking invitations, which he used to spread the word about GNU, free software, and dangers such as “look and feel” lawsuits and software patents.

最关键的是，MacArthur捐助的资金让斯托曼有了更多的自由，他可以全身心的投入软件自由的事业了。有了这些资助，他决定更多的去四处宣讲GNU工程的使命。

Interestingly, the GNU system’s completion would stem from one of these trips. In April 1991, Stallman paid a visit to the Polytechnic University in Helsinki, Finland. Among the audience members was 21-year-old Linus Torvalds, who was just beginning to develop the Linux kernel – the free software kernel destined to fill the GNU system’s main remaining gap.

很有意思的是GNU工程和自由软件运动最后的胜利差不多就是源于这些旅行。1990年，斯托曼去芬兰赫尔新基的Polytechnic大学访问。在听众中，有一名叫Linux Torvalds的学生，那时他只有21岁，后来他就是Linux内核的发明者。Linux内核完美的填补了GNU工程中所空缺的那一部分。

A student at the nearby University of Helsinki at the time, Torvalds regarded Stallman with bemusement. “I saw, for the first time in my life, the stereotypical long-haired, bearded hacker type,” recalls Torvalds in his 2001 autobiography *Just for Fun*. “We don’t have much of them in Helsinki.”

托瓦兹那时在附近的赫尔新基大学上学，斯托曼给他留下的第一印象是颇为奇怪的。“我这辈子头一次看到这样一个长头发、长胡子的古怪黑客。”托瓦兹在2001年自传《Just for Fun》中回忆道，“在赫尔新基很少能见到这样的人。”

While not exactly attuned to the “sociopolitical” side of the Stallman agenda, Torvalds nevertheless appreciated one aspect of the agenda’s underlying logic: no programmer writes error-free code. Even when users have no wish to adapt a program to their specific preferences, any program can use improvement. By sharing software, hackers put a program’s improvement ahead of individual motivations such as greed or ego protection.

托瓦兹并不了解斯托曼的宏大计划中的政治性的一面，但他仍然很欣赏这个计划深层次的逻辑：没有人可以写出没有错误的代码。通过共享软件，黑客们把不断改进程序的目标放到了个人野心之上。

Like many programmers of his generation, Torvalds had cut his teeth not on mainframe computers like the IBM 7094, but on a motley assortment of home-built computer systems. As a university student, Torvalds had made the step up from PC programming to Unix, using the university’s MicroVAX. This ladder-like progression had given Torvalds a different perspective on the barriers to machine access. For Stallman, the chief barriers were bureaucracy and privilege. For Torvalds, the chief barriers were geography and the harsh Helsinki winter. Forced to trek across the University of Helsinki just to log in to his Unix account, Torvalds quickly began looking for a way to log in from the warm confines of his off-campus apartment.

与他同辈的很多程序员一样，托瓦兹并不关注类似于IBM 7094这样的大型机，而是那些五花八门配置的家用电脑。作为一个大学生，托瓦兹是通过在学校的MicroVAX上学习C语言，一步步进入Unix的世界的。这种阶梯式的学习曲线使

托瓦兹对于访问主机的障碍有着一种不同的视角。对于斯托曼来说，主要的障碍是官僚作风和特权控制。对于托瓦兹来说，则是地理位置和赫尔辛基寒冷的冬天。为了避免穿过整个赫尔辛基大学的校园去登陆他的Unix系统，托瓦兹很快开始设法找到一种可以在他温暖的校外公寓中就访问Unix的方法。

Torvalds was using Minix, a lightweight nonfree system developed as an instructional example by Dutch university professor Andrew Tanenbaum. It included the non-free Free University Compiler Kit, plus utilities of the sort that Tanenbaum had contemptuously invited Stallman in 1983 to write.

这项研究把托瓦兹引向了Minix操作系统，Minix操作系统是Dutch大学教授Andrew Tanenbaum所开发的一个用于教学用途的轻量级Unix系统。这个系统可以在386电脑上正常运行，而这已经是托瓦兹能够负担的最强大的机器，但是，这个系统仍然缺乏一些必要的功能。最明显的一点是他没有终端模拟的功能，有了这项功能让能使托瓦兹可以在家里模拟出一个学校电脑的终端并登陆到学校的MicroVAX计算机上。

Minix fit within the memory confines of Torvalds' 386 PC, but it was intended more to be studied than used. The Minix system also lacked the facility of terminal emulation, which would mimic a typical display terminal and thus enable Torvalds to log in to the MicroVAX from home.

.....

Early in 1991, Torvalds began writing a terminal emulation program. He used Minix to develop his emulator, but the emulator didn't run on Minix; it was a stand-alone program. Then he gave it features to access disk files in Minix's file system. Around then, Torvalds referred to his evolving work as the "GNU/Emacs of terminal emulation programs."

1991年的夏天，托瓦兹把Minix重头重写了一遍，并加了上了一些他所开发的特性。到夏天结束的时候，他把这个作品称为"GNU/Emacs of terminal eulation programs"。 .....

Since Minix lacked many important features. Torvalds began extending his terminal emulator into a kernel comparable to that of Minix, except that it was monolithic. Feeling ambitious, he solicited a Minix newsgroup for copies of the POSIX standards, the specifications for a Unix-compatible kernel. A few weeks later, having put his kernel together with some GNU programs and

adapted them to work with it, Torvalds was posting a message reminiscent of Stallman's original 1983 GNU posting:

..... 他对这个作品很有信心，并在一个Minix新闻组上找到了一份POSIX标准，POSIX标准定义了一个程序是否是与Unix兼容的。几周后，托瓦兹发布了一则让人可以联想到斯托曼在1983年启动GNU工程时的所发的贴子的消息：

> Hello everybody out there using minix-
>
> I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386 (486) AT clones. This has been brewing since April, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things). I've currently ported bash (1.08) and gcc (1.40)…
>
> 使用Minix操作系统的各位大家好。
>
> 我正在开发一个（自由的）操作系统（只是出于兴趣，这个系统不会像386 (486) AT上的gnu系统那么庞大和专业）。这个系统从4月份开始开发，现在已经逐渐可用了。我期待听到各位喜欢或不喜欢Minix的人们的反馈，因为这个系统参考了它的不少设计（比如出于实用的需要，它们的文件系统采用了相同的物理布局，除此之外还有一些别的相同之处）

The posting drew a smattering of responses and within a month, Torvalds had posted a 0.01 version of his kernel – i.e., the earliest possible version fit for outside review – on an Internet FTP site. In the course of doing so, Torvalds had to come up with a name for the new kernel. On his own PC hard drive, Torvalds had saved the program as Linux, a name that paid its respects to the software convention of giving each Unix variant a name that ended with the letter X. Deeming the name too "egotistical," Torvalds changed it to Freax, only to have the FTP site manager change it back.

在不到一个月的时间内，这个贴子收到了大量的回复。托瓦兹在Internet上的一个FTP服务器上发布了这个操作系统的0.01版，这是最早向公众开放征求意见的版本。为了实现这个目标，托瓦兹需要给这个新的系统起一个名字。在他自己的电脑硬盘上，这个程序被命名为Linux，这个名字体现了它是一个Unix的变形版本，所以名字以字母X结尾。托瓦兹认为这个名字太过于自恋，所以在FTP

上把它的名字改成了Freax，但FTP管理员又把名字改了回来。

Torvalds said he was writing a free operating system, and his comparing it with GNU shows he meant a complete system. However, what he actually wrote was a kernel, pure and simple. Torvalds had no need to write more than the kernel because, as he knew, the other needed components were already available, thanks to the developers of GNU and other free software projects. Since the GNU Project wanted to use them all in the GNU system, it had perforce made them work together. While Torvalds continued developing the kernel, he (and later his collaborators) made those programs work with it too.

尽管托瓦兹准备开发一个完整的操作系统，但他和其它开发人员都意识到大部分重要的工作软件都已经有现成的版本，这些软件是由GNU、BSD和无数自由软件开发者共同完成的。对于Linux的开发团队来说，GNU C编译器是这些工具软件中最重要的一个，有了它才使用C语言进行开发成为可能。

Initially, Linux was not free software: the license it carried did not qualify as free, because it did not allow commercial distribution. Torvalds was worried that some company would swoop in and take Linux away from him. However, as the growing GNU/Linux combination gained popularity, Torvalds saw that sale of copies would be useful for the community, and began to feel less worried about a possible takeover. This led him to reconsider the licensing of Linux.

.....

Neither compiling Linux with GCC nor running GCC with Linux required him legally to release Linux under the GNU GPL, but Torvalds' use of GCC implied for him a certain obligation to let other users borrow back. As Torvalds would later put it: "I had hoisted myself up on the shoulders of giants." Not surprisingly, he began to think about what would happen when other people looked to him for similar support. A decade after the decision, Torvalds echoes the Free Software Foundation's Robert Chassell when he sums up his thoughts at the time:

.....

> You put six months of your life into this thing and you want to make it available and you want to get something out of it, but you don't want

> people to take advantage of it. I wanted people to be able to see [Linux], and to make changes and improvements to their hearts' content. But I also wanted to make sure that what I got out of it was to see what they were doing. I wanted to always have access to the sources so that if they made improvements, I could make those improvements myself.
>
> 你花了你生命中宝贵的半年时间去做一个东西，并且希望它能广为流传并从中获利，但你不希望别人用它来牟利。我希望人们都能了解Linux，并且能对它进行修改，让它更符合每个人的期望。但我同样希望能知道别人都拿它来做了些什么，我希望永远可以看到那些别人所改动的源代码，这样的话，我也能有机会对它继续进行改进。

When it was time to release the 0.12 version of Linux, the first to operate fully with GCC, Torvalds decided to throw his lot in with the free software movement. He discarded the old license of Linux and replaced it with the GPL. Within three years, Linux developers were offering release 1.0 of Linux, the kernel; it worked smoothly with the almost complete GNU system, composed of programs from the GNU Project and elsewhere. In effect, they had completed the GNU operating system by adding Linux to it. The resulting system was basically GNU plus Linux. Torvalds and friends, however, referred to it confusingly as "Linux."

Linux的0.12版本中第一次完成的集成了GCC，那时托瓦兹决定在这次发布时宣布他与自由软件运动的联盟。他废除了Linux内核原来的许可证，并替换为GPL。这个决定带来了一场移植软件的狂欢，托瓦兹和其它Linux爱好者寻找了各种GNU程序并把它们移植到了Linux中。在不到3年的时间后，Linux开发者发布了第一个可以用于生产环境的Linux版本，1.0。里面包含了经过大面积修改的GCC、GDB和一系列BSD工具。

By 1994, the amalgamated system had earned enough respect in the hacker world to make some observers from the business world wonder if Torvalds hadn't given away the farm by switching to the GPL in the project's initial months. In the first issue of *Linux Journal*, publisher Robert Young sat down with Torvalds for an interview. When Young asked the Finnish programmer if he felt regret at giving up private ownership of the Linux source code, Torvalds said no. "Even with 20/20 hindsight," Torvalds said, he considered the GPL "one of the very best design decisions" made during the early stages of the Linux project.

到了1994，这个混杂了各种程序的操作系统在黑客圈子里赢得了极大的关注，很多观察者因此认为托瓦兹在切换到GPL后的最初几个月中，并没有把程序真正的交给社区。在第一期《Linux通迅》杂志中，出版商Robert Young对托瓦兹进行了一次面对面的采访。当Young向这名芬兰程序员提问是否后悔放弃了对Linux源代码的私有权时，托瓦兹的回答是否定的。托瓦兹说：“即使是做一次事后诸葛亮，我仍觉得选择GPL是Linux项目初期一个最佳的设计决策。”

That the decision had been made with zero appeal or deference to Stallman and the Free Software Foundation speaks to the GPL's growing portability. Although it would take a couple of years to be recognized by Stallman, the explosiveness of Linux development conjured flashbacks of Emacs. This time around, however, the innovation triggering the explosion wasn't a software hack like Control-R but the novelty of running a Unix-like system on the PC architecture. The motives may have been different, but the end result certainly fit the ethical specifications: a fully functional operating system composed entirely of free software.

这个决策完全与斯托曼和自由软件基金会所宣称的GPL的普适性完全没有冲突。虽然斯托曼一开始并没有注意到这点，但是Linux的爆发式的成长反过来让Emacs受到了更多的关注。这一次的创新，决非是一个像Control-R这样的小小的软件技巧，而是让一个类Unix系统可以在PC上运行的新奇事物。虽然动机不同，但是结果是一致的：创造一个只由自由软件构成的完整的操作系统。

As his initial email message to the comp.os.minix newsgroup indicates, it would take a few months before Torvalds saw Linux as anything more than a holdover until the GNU developers delivered on the Hurd kernel. As far as Torvalds was concerned, he was simply the latest in a long line of kids taking apart and reassembling things just for fun. Nevertheless, when summing up the runaway success of a project that could have just as easily spent the rest of its days on an abandoned computer hard drive, Torvalds credits his younger self for having the wisdom to give up control and accept the GPL bargain.

就像托瓦兹在最初发表到comp.os.minix新闻组的电子邮件所写的样，他最初确实只是把Linux作为是GNU开发者发布HURD内核前的一个替代品。这个初始的想法对于自由软件基金会而言是一个打击。以托瓦兹所设想的，他只是喜欢把玩具拆开再组装起来的广大儿童中的一员。更不要说只要在一块被废弃的硬盘上用闲暇时光就可以创造出来的一个成功的项目。托瓦兹用放弃对Linux的控制并接受GPL的施舍表现出了他年轻而杰出的智慧。

“I may not have seen the light,” writes Torvalds, reflecting on Stallman’s 1991 Polytechnic University speech and his subsequent decision to switch to the GPL. “But I guess something from his speech sunk in.”

托瓦兹回忆起1991年斯托曼的演讲以及其后他决定对Linux使用GPL授权的往事，他写道：”我也许没有看到指路的明灯，但我从他的演讲中领悟到了一些东西。“

GNU/Linux GNU/Linux

By 1993, the free software movement was at a crossroads. To the optimistically inclined, all signs pointed toward success for the hacker culture. *Wired* magazine, a funky, new publication offering stories on data encryption, Usenet, and software freedom, was flying off magazine racks. The Internet, once a slang term used only by hackers and research scientists, had found its way into mainstream lexicon. Even President Clinton was using it. The personal computer, once a hobbyist’s toy, had grown to full-scale respectability, giving a whole new generation of computer users access to hacker-built software. And while the GNU Project had not yet reached its goal of a fully intact, free GNU operating system, users could already run the GNU/Linux variant.

1993年，自由软件运动走到了一个十字路口。从乐观的方面讲，黑客文化正在变得越来越成功。《*连线*》杂志作为一本新兴的时尚杂志，用有关数据加密、新闻组、自由软件等热门话题树立了自己的地位。而Internet这个原本之在黑客和科学家群体中流行的词汇，也渐渐的融入了主流社会，连时任美国总统的克林顿也不忘提到它。昔日作为数码爱好者玩具的个人电脑，也得到了广泛的普及，黑客们开发的软件也进入了新一代的计算机用户的视野。然而，GNU工程却还没完成它创建一个完整的自由操作系统的目标，用户们通常都使用Linux系统。

Any way you sliced it, the news was good, or so it seemed. After a decade of struggle, hackers and hacker values were finally gaining acceptance in mainstream society. People were getting it.

不管你怎么想，但这听起来是个好消息，或者至少看起来是。经过了十年的挣扎，黑客和黑客的价值观终于开始被主流社会所接受。大众开始认识它们。

Or were they? To the pessimistically inclined, each sign of acceptance carried its own troubling countersign. Sure, being a hacker was suddenly cool, but was cool good for a community that thrived on alienation? Sure, the White House was saying nice things about the Internet, even going so far as to register its own domain name, whitehouse.gov, but it was also meeting with the companies, censorship advocates, and law-enforcement officials looking to tame the Internet's Wild West culture. Sure, PCs were more powerful, but in commoditizing the PC marketplace with its chips, Intel had created a situation in which proprietary software vendors now held the power. For every new user won over to the free software cause via GNU/Linux, hundreds, perhaps thousands, were booting up Microsoft Windows for the first time. GNU/Linux had only rudimentary graphical interfaces, so it was hardly user-friendly. In 1993, only an expert could use it. The GNU Project's first attempt to develop a graphical desktop had been abortive.

事实确实如此吗？从悲观的角度来说，任何被接受的信号也同时间意味着麻烦正在滋长。当然，成为一个黑客突然变成了一件很酷的事情，但是对于一个社区而言被人疏远的繁荣真得算是一件好事吗？当然，白官表明态度说Internet是一个很好的趋势，甚至已经在上面注册了自己的域名，whitehouse.gov，但是它也在与各个公司、支持审查机关和执法机构一同开会，设法管理西部荒原般的Internet文化。是的，个人电脑正在变得越来越强大，但是Intel做为PC市场上的芯片供应商，已经形成了一种由私有软件开发商垄断的局面。虽然有很多人开始使用通过Linux接触到自由软件，但是与此同时有成百上千的人成为了微软Windows的用户。.....

Then there was the political situation. Copyrighting of user interfaces was still a real threat – the courts had not yet rejected the idea. Meanwhile, patents on software algorithms and features were a growing danger that threatened to spread to other countries.

.....

Finally, there was the curious nature of GNU/Linux itself. Unrestricted by legal disputes (such as BSD faced), GNU/Linux's high-speed evolution had been so unplanned, its success so accidental, that programmers closest to the software code itself didn't know what to make of it. More compilation album than unified project, it was comprised of a hacker medley of greatest hits: everything from GCC, GDB, and glibc (the GNU Project's newly developed C

Library) to X (a Unix-based graphic user interface developed by MIT's Laboratory for Computer Science) to BSD-developed tools such as BIND (the Berkeley Internet Naming Daemon, which lets users substitute easy-to-remember Internet domain names for numeric IP addresses) and TCP/IP. In addition, it contained the Linux kernel – itself designed as a replacement for Minix. Rather than developing a new operating system, Torvalds and his rapidly expanding Linux development team had plugged their work into this matrix. As Torvalds himself would later translate it when describing the secret of his success: "I'm basically a very lazy person who likes to take credit for things other people actually do."

到头来，这是Linux奇妙的天性。没有像GNU那样的设计错误，也有BSD那样的法律争论，Linux的高速发展完全是出乎意料的，它的成功也是偶然的，甚至于编写了它的代码的程序员们也没法想象事情为什么会是这样的。Linux比一个完整的操作系统更为复杂，它完全就是一个黑客们把各种顶级工具混杂在一起的产物：从GCC、GDB和glibc （GNU工程新开发的C库）到X（MIT的计算机科学实验室开发的基于Unix架构的图形化用户界面），还包括BSD开发的那些类似于BIND（the Berkeley Internet Naming Daemon，用于提供把IP地址与域名进行映射的网络服务）和TCP/IP协议。整个体系的顶石，当然就是Linux内核本身，它是一个在Minix基础上重新开发的全新的内核。托瓦兹和他高速成长的Linux开发团队并没有把所有东西都从头造一遍，而是遵循了古老的毕加索的格言："能工摹其形，巧匠摄其魂。"后来托瓦兹本人也以类似的说法评价他自己成功的秘诀："我是一个非常懒惰的人，并且喜欢把别人的成果占为已有。"

Such laziness, while admirable from an efficiency perspective, was troubling from a political perspective. For one thing, it underlined the lack of an ideological agenda on Torvalds' part. Unlike the GNU developers, Torvalds hadn't built his kernel out of a desire to give his fellow hackers freedom; he'd built it to have something he himself could play with. So what exactly was the combined system, and which philosophy would people associate it with? Was it a manifestation of the free software philosophy first articulated by Stallman in the *GNU Manifesto*? Or was it simply an amalgamation of nifty software tools that any user, similarly motivated, could assemble on his own home system?

这种惰性，从提高效率的角度来看，还是非常值得景仰的，但是从政治的角度来看就变得很棘手。一方面，它表现了从托瓦兹这一方面来说所缺乏的意识形态议程。与GNU开发者不同，托瓦兹从一开始就没有想过要让他的团队一起来

开发一个完整的操作系统，他只是想做一个自己可以玩的东西。就像汤姆·索亚用石灰水刷白篱笆一样，托瓦兹的天才能力并不表现在他的大局观上，而是在于他可以用最快的速度召集到一批黑客来完成开发任务。托瓦兹和他找来一同工作的黑客们成功了，而其他人则产生了一些疑问：那么，准确来说，Linux究竟是个什么东西呢？它是不是对斯托曼在《*GNU宣言*》中所表术的自由软件哲学的一种表现形式呢？还是只是一个人们可以在家用电脑上使用的简单的工具集合呢?

By late 1993, a growing number of GNU/Linux users had begun to lean toward the latter definition and began brewing private variations on the theme. They began to develop various “distributions” of GNU/Linux and distribute them, sometimes gratis, sometimes for a price. The results were spotty at best.

到了1993年下半年，越来越多的Linux用户开始倾向于后一种定义并开始基于Linux开发各种变种。他们甚至开始把这些衍生版本（或者说“发行版”）包装起来并向Unix爱好者出售，结果充其量就是些小打小闹。

“This was back before Red Hat and the other commercial distributions,” remembers Ian Murdock, then a computer science student at Purdue University. “You’d flip through Unix magazines and find all these business card-sized ads proclaiming ‘Linux.’ Most of the companies were fly-by-night operations that saw nothing wrong with slipping a little of their own [proprietary] source code into the mix.”

”这就是Red Hat或其它商业发行版出现前的情况。“ 伊恩·默多克回忆道，那时他还是普渡大学计算机系的一名学生。"如果你翻阅Unix的杂志，你会看到很多名片大小的小广告，推广‘Linux’。很多这样的公司都是采用短视的方式来运作，他们并不认为把自己的代码加到Linux中有什么不对的地方。”

Murdock, a Unix programmer, remembers being “swept away” by GNU/Linux when he first downloaded and installed it on his home PC system. “It was just a lot of fun,” he says. “It made me want to get involved.” The explosion of poorly built distributions began to dampen his early enthusiasm, however. Deciding that the best way to get involved was to build a version free of additives, Murdock set about putting a list of the best free software tools available with the intention of folding them into his own distribution. “I wanted something that would live up to the Linux name,” Murdock says.

默多克，作为一名Unix程序员，还记得他第一次在家里的PC电脑上下载安装

Linux时被这个系统“扫地出门”的情景。“这是一件充满乐趣的事情”，他说，”这让我感觉到自己参与到了这个项目中。“然而，各种低质发行版的泛滥渐渐消耗了他的早期的热情。默多克认为，最好的参与到这个项目中的方式是开发一个不添加任何额外组件的Linux发行版，由于他开始收集各种最好的自由软件工具，打算把它们整合到他自己的发行版中。默多克说，我希望能创造出一个对得起Linux这个名字的系统。

In a bid to “stir up some interest,” Murdock posted his intentions on the Internet, including Usenet’s comp.os.linux newsgroup. One of the first responding email messages was from rms@ai.mit.edu. As a hacker, Murdock instantly recognized the address. It was Richard M. Stallman, founder of the GNU Project and a man Murdock knew even back then as “the hacker of hackers.” Seeing the address in his mail queue, Murdock was puzzled. Why on Earth would Stallman, a person leading his own operating-system project, care about Murdock’s gripes over “Linux” distributions?

为了“激起兴趣”，默多克在网上发布了他的想法，包括在comp.os.linux这个Usenet的新闻组中。最早回复他的电子邮件中有一封来自rms@ai.mit.edu。作为一个黑客，默多克立刻就认出了这个邮件地址。这就是理查德·斯托曼，GNU工程的创始人，一个在默多克心目中被认为是”黑客中的黑客“的人物。在邮件地址列表中看到他的邮件地址，默多克感到有点疑惑。为什么斯托曼这个正在引领自己的操作系统工程的人物，会关心默多克对于Linux的所发的牢骚?

Murdock opened the message.

默多克打开了这封邮件。

“He said the Free Software Foundation was starting to look closely at Linux and that the FSF was interested in possibly doing a Linux [*sic*] system, too. Basically, it looked to Stallman like our goals were in line with their philosophy.”

“他说自由软件基金会正在开始进一步关注Linux，并很有兴趣基于Linux [*sic*] 来完成一个操作系统。总体上来看，好像斯托曼对我们的目标很有兴趣，是他们的哲学中的一部分。”

Not to overdramatize, the message represented a change in strategy on Stallman’s part. Until 1993, Stallman had been content to keep his nose out of Linux affairs. After first hearing of the new kernel, Stallman asked a friend to

check its suitability. Recalls Stallman, “He reported back that the software was modeled after System V, which was the inferior version of Unix. He also told me it wasn’t portable.”

这封信展现了斯托曼立场的360度大转弯。在1993年之前，斯托曼一直都不干涉Linux社区的事务。事实上，当Linux在1991看开始出现在Unix编程的范畴中开始，他都总是回避这个操作系统。斯托曼说，当他得知有一个可以在PC平台上运行的类Unix系统时，他找了一个朋友帮忙去了解这个系统的相关情况。“他汇报说这个系统是以System V的作为原型的，这是一个比较低劣的Unix版本。并且，他告诉我，这个系统是不可移植的。”

The friend’s report was correct. Built to run on 386-based machines, Linux was firmly rooted to its low-cost hardware platform. What the friend failed to report, however, was the sizable advantage Linux enjoyed as the only free kernel in the marketplace. In other words, while Stallman spent the next year and a half listening to progress reports from the Hurd developer, reporting rather slow progress, Torvalds was winning over the programmers who would later uproot and replant Linux and GNU onto new platforms.

那个朋友的报告并不正确。Linux虽然是为基于386的机器设计，Linux同样是生根于低成本的平台。他的朋友没有发现Linux一个重大的优势，那就是它是当时市面上唯一个一个可以自由修改的操作系统。换句话说，在接下来的三年时间中，当斯托曼还在听取Hurd团队的Bug报告时，托瓦兹正在赢得广大程序员的支持，他们后来成为把Linux移植到各种新的硬件平台的中坚力量。

By 1993, the GNU Project’s failure to deliver a working kernel was leading to problems both within the GNU Project and in the free software movement at large. A March, 1993, *Wired* magazine article by Simson Garfinkel described the GNU Project as “bogged down” despite the success of the project’s many tools. Those within the project and its nonprofit adjunct, the Free Software Foundation, remember the mood as being even worse than Garfinkel’s article let on. “It was very clear, at least to me at the time, that there was a window of opportunity to introduce a new operating system,” says Chassell. “And once that window was closed, people would become less interested. Which is in fact exactly what happened.”

直到1993年，GNU工程还是没能发布一个可以使用的操作系统内核，对于GNU工程甚至于自由软件运动本身都产生了不小的影响。1993年3月，辛姆森·加芬克尔在《*连线*》杂志上发表了一篇文章，把GNU工程描述为一个“深陷泥沼”的

项目，全然无视这个项目中所创造出来的那些成功的工具。 这个项目和作为这个项目非营性的赞助商的自由软件基金会的成员回忆说，当时的心情比加芬克尔的文章本身造成的伤害还要不好，“情况很明显，至少对于那时的我来说是这样，那就是引入一个全新的操作系统需要一个时间窗口”，查瑟尔说，“一旦错过了个时间窗口，人们对新的系统就不会有太大的兴趣，这也正是现在正在发生的情况。"

Much has been made about the GNU Project's struggles during the 1990-1993 period. While some place the blame on Stallman for those struggles, Eric Raymond, an old friend of Stallman's who supported the GNU Project lukewarmly, says the problem was largely institutional. "The FSF got arrogant," Raymond says. "They moved away from the goal of doing a production-ready operating system to doing operating-system research." Even worse, "They thought nothing outside the FSF could affect them."

1990-1993年工作期间，GNU工程一直处在挣扎中。有些人为此批评斯托曼，但GNU Emacs早期团队中的一员并后来成为斯托曼批评家的Eric Raymond认为，问题的主要原因是体制上的。“自由软件基金会太骄傲了”，雷蒙德说。“他们把自己的目标从一个成熟的操作系统上转移到了进行操作系统的研究工具”，更糟糕的事，“他们认为除了自由软件基金会以外没人可以影响他们。”

Murdock adopts a more charitable view. "I think part of the problem is they were a little too ambitious and they threw good money after bad," he says. "Micro-kernels in the late 80s and early 90s were a hot topic. Unfortunately, that was about the time that the GNU Project started to design their kernel. They ended up with a lot of baggage and it would have taken a lot of backpedaling to lose it."

默多克，作为一名并不参与GNU工程内部运作的人，更具有包容的观点。"我认为问题的一方面是他们有点过于好高骛远，把钱投入了不正确的地方。”他说。“微内核在80年代和90年代初期是一个热门的话题。很不幸的是那正是GNU工程开始设计他们的内核的时间，结果就是带来了很多的包袱，这些包袱很难被轻易甩掉。”

Stallman responds, "Although the emotions Raymond cites come from his imagination, he's right about one cause of the Hurd's delay: the Hurd developer several times redesigned and rewrote large parts of the code based on what he had learned, rather than trying to make the Hurd run as soon as possible. It was good design practice, but it wasn't the right practice for our

goal: to get something working ASAP."

.....

Stallman cites other issues that also caused delay. The Lotus and Apple lawsuits claimed much of his attention; this, coupled with hand problems that prevented him from typing for three years, mostly excluded Stallman from programming. Stallman also cites poor communication between various portions of the GNU Project. "We had to do a lot of work to get the debugging environment to work," he recalls. "And the people maintaining GDB at the time were not that cooperative." They had given priority to supporting the existing platforms of GDB's current users, rather than to the overall goal of a complete GNU system.

斯托曼引用了一些客观因素来解释延期的原因。莲花公司和苹果公司的官司带来了政治上的分心,这使得斯托曼不能全心投入到Hurd的团队中去。斯托曼还说，GNU工程中各个部分之间缺乏有效的沟通。"我们做了很多的工作才使调试环境可以正常工作起来。"他回忆道，"而那时维护GDB的团队又并不是很合作。"

Most fundamentally, however, Stallman says he and the Hurd developers underestimated the difficulty of developing the Unix kernel facilities on top of the Mach microkernel. "I figured, OK, the [Mach] part that has to talk to the machine has already been debugged," Stallman says, recalling the Hurd team's troubles in a 2000 speech. "With that head start, we should be able to get it done faster. But instead, it turned out that debugging these asynchronous multithreaded programs was really hard. There were timing bugs that would clobber the files, and that's no fun. The end result was that it took many, many years to produce a test version."

但是，斯托曼也说，他和其它GNU工程的成员低估了把Mach微内核扩展成为一个完整的Unix内核的难度。"我以为Marh内核中与硬件通信的部分已经调试无误了。"斯托曼在2000年一次演讲中回忆Hurd团遇到的问题时说。"如果有这个为基础，我们可以进展的更为顺利一些"。但是，事实上，调试这些异步的多线程程序非常困难。一些时序处理上的问题会导致文件损坏，这一点也不好玩。最终的结果就是，我们花了很多很多年的时间，还是只能发布出一个测试的版本。"

Over time, the growing success of GNU together with Linux made it clear that

the GNU Project should get on the train that was leaving and not wait for the Hurd. Besides, there were weaknesses in the community surrounding GNU/Linux. Sure, Linux had been licensed under the GPL, but as Murdock himself had noted, the desire to treat GNU/Linux as a purely free software operating system was far from unanimous. By late 1993, the total GNU/Linux user population had grown from a dozen or so enthusiasts to somewhere between 20,000 and 100,000. What had once been a hobby was now a marketplace ripe for exploitation, and some developers had no objection to exploiting it with nonfree software. Like Winston Churchill watching Soviet troops sweep into Berlin, Stallman felt an understandable set of mixed emotions when it came time to celebrate the GNU/Linux "victory."

.....

Although late to the party, Stallman still had clout. As soon as the FSF announced that it would lend its money and moral support to Murdock's software project, other offers of support began rolling in. Murdock dubbed the new project Debian – a compression of his and his wife, Deborah's, names – and within a few weeks was rolling out the first distribution. "[Richard's support] catapulted Debian almost overnight from this interesting little project to something people within the community had to pay attention to," Murdock says.

虽说是赶了个晚集，斯托曼还是有着巨大的影响力。当自由软件基金会宣布它会向默多克的软件项目提供资金和精神支持时，其它各方面的支持也开始源源不断的涌入。默多克启动了一个名为Debian的新项目，这个项目的名字是用他和他夫人Deborah的名字构成的。在短短的几个星期后，这个项目就发布了第一个版本。"[理查德的帮助]使得这个出于个人兴趣开发的小小的项目在一夜间变成了社区中每一个人都很关注的明星项目。" 默多克说。

In January of 1994, Murdock issued the *Debian Manifesto*. Written in the spirit of Stallman's *GNU Manifesto* from a decade before, it explained the importance of working closely with the Free Software Foundation. Murdock wrote:

1994年1月，默多克发布了"*Debian宣言*"， 这个宣言与斯托曼十年前发布的"*GNU宣言*"在精神上非常一致，它解释了与自由软件基金会紧密合作的重要性。默多克写道：

> The Free Software Foundation plays an extremely important role in the future of Debian. By the simple fact that they will be distributing it, a message is sent to the world that Linux [*sic*] is not a commercial product and that it never should be, but that this does not mean that Linux will never be able to compete commercially. For those of you who disagree, I challenge you to rationalize the success of GNU Emacs and GCC, which are not commercial software but which have had quite an impact on the commercial market regardless of that fact.
>
> 自由软件基金会对于Debian的未来具有非常重要的意义。一方面他们可以帮助分发Debian，向全世界发出信号，Linux不是也不应该是一个商业的产品，但这并不意味着Linux就不如商业产品。如果你们有人不同意这个观点，我可以以GNU Emacs和GCC的例子来证明，它们都不是商业软件，但它们都对商业市场产生了重大的影响。
>
> The time has come to concentrate on the future of Linux [*sic*] rather than on the destructive goal of enriching oneself at the expense of the entire Linux community and its future. The development and distribution of Debian may not be the answer to the problems that I have outlined in the *Manifesto*, but I hope that it will at least attract enough attention to these problems to allow them to be solved.

已经到了集中精力关注Linux [*sic*]的未来的时候了，不应该以Linux社区毁灭和它的未来的消亡作为代价来实现个人富裕的目标。虽然开发和发布Debian也许并不能回答我在宣言中提出的问题，但我希望它至少可以帮心大家更多关注这些问题并帮助解决它们。

Shortly after the *Manifesto's* release, the Free Software Foundation made its first major request. Stallman wanted Murdock to call its distribution "GNU/Linux." At first, Stallman proposed the term "Lignux" – combining the names Linux and GNU – but the initial reaction was very negative, and this convinced Stallman to go with the longer but less criticized GNU/Linux.

*宣言*发布后不久，自由软件基金会提出一个要求。斯托曼希望默多克把他的发行版叫作"GNU/Linux"。默多克说，刚开始的时候，斯托曼曾希望他们使用"Lignux"这个名字，表示GNU是Linux系统的心脏。但是这个名字在Usenet和一些即席的黑客团体中调查中得到的嘘声迫使斯托曼退而求其次，改用GNU/Linux这个相对自然一点的名字。

Some dismissed Stallman's attempt to add the "GNU" prefix as a belated quest for credit, never mind whether it was due, but Murdock saw it differently. Looking back, Murdock saw it as an attempt to counteract the growing tension between the GNU Project's developers and those who adapted GNU programs to use with the Linux kernel. "There was a split emerging," Murdock recalls. "Richard was concerned."

.....

By 1990, each GNU program had a designated maintainer-in-charge. Some GNU programs could run on many different systems, and users often contributed changes to port them to another system. Often these users knew only that one system, and did not consider how to keep the code clean for other systems. To add support for the new system while keeping the code comprehensible, so it could be maintained reliably for all systems, then required rewriting much of the changes. The maintainer-in-charge had the responsibility to critique the changes and tell their user-authors how to redo parts of the port. Generally they were eager to do this so that their changes would be integrated into the standard version. Then the maintainer-in-charge would edit in the reworked changes, and take care of them in future maintenance. For some GNU programs, this had happened dozens of times for dozens of different systems.

.....

The programmers who adapted various GNU programs to work with the kernel Linux followed this common path: they considered only their own platform. But when the maintainers-in-charge asked them to help clean up their changes for future maintenance, several of them were not interested. They did not care about doing the correct thing, or about facilitating future maintenance of the GNU packages they had adapted. They cared only about their own versions and were inclined to maintain them as forks.

.....

In the hacker world, forks are an interesting issue. Although the hacker ethic permits a programmer to do anything he wants with a given program's source code, it is considered correct behavior to offer to work with the original developer to maintain a joint version. Hackers usually find it useful, as well as

proper, to pour their improvements into the program's principal version. A free software license gives every hacker the right to fork a program, and sometimes it is necessary, but doing so without need or cause is considered somewhat rude.

在黑客的世界中，分支是一个很有意思的现象。虽然黑客伦理中允许一个程序员对程序的代码做任何的改进，大部分黑客都希望自己的改动能进入程序源代码的主干中，这样才能保证与其它的人程序有最大程度上的兼容。在Linux开发的初期就对glibc创建分支就意味着可能会失去成百上千的Linux开发者。这也会造成Linux与斯托曼和GNU团队期望开发出的GNU系统之间的不兼容性。

As leader of the GNU Project, Stallman had already experienced the negative effects of a software fork in 1991. Says Stallman, "Lucid hired several people to write improvements to GNU Emacs, meant to be contributions to it; but the developers did not inform me about the project. Instead they designed several new features on their own. As you might expect, I agreed with some of their decisions and disagreed with others. They asked me to incorporate all their code, but when I said I wanted to use about half of it, they declined to help me adapt that half to work on its own. I had to do it on my own." The fork had given birth to a parallel version, Lucid Emacs, and hard feelings all around.

作为GNU工程的领导者，斯托曼早在1991年时就体会到了创建软件分支的不良后果了。那时候，有一批就职于Lucid公司的Emacs开发人员与斯托曼闹翻了，因为斯托曼不原意把他们对Emacs代码修改合入Emacs代码主干，于是他们不得不创建了一个名为Lucid Emacs的分支版本。

Now programmers had forked several of the principal GNU packages at once. At first, Stallman says he considered the forks to be a product of impatience. In contrast to the fast and informal dynamics of the Linux team, GNU source-code maintainers tended to be slower and more circumspect in making changes that might affect a program's long-term viability. They also were unafraid of harshly critiquing other people's code. Over time, however, Stallman began to sense that there was an underlying lack of awareness of the GNU Project and its objectives when reading Linux developers' emails.

.....

"We discovered that the people who considered themselves 'Linux users' didn't care about the GNU Project," Stallman says. "They said, 'Why should I

bother doing these things? I don't care about the GNU Project. It [the program]'s working for me. It's working for us Linux users, and nothing else matters to us.' And that was quite surprising, given that people were essentially using a variant of the GNU system, and they cared so little. They cared less than anybody else about GNU." Fooled by their own practice of calling the combination "Linux," they did not realize that their system was more GNU than Linux.

.....

For the sake of unity, Stallman asked the maintainers-in-charge to do the work which normally the change authors should have done. In most cases this was feasible, but not in glibc. Short for GNU C Library, glibc is the package that all programs use to make "system calls" directed at the kernel, in this case Linux. User programs on a Unix-like system communicate with the kernel only through the C library.

The changes to make glibc work as a communication channel between Linux and all the other programs in the system were major and ad-hoc, written without attention to their effect on other platforms. For the glibc maintainer-in-charge, the task of cleaning them up was daunting. Instead the Free Software Foundation paid him to spend most of a year reimplementing these changes from scratch, to make glibc version 6 work "straight out of the box" in GNU/Linux.

.....

Murdock says this was the precipitating cause that motivated Stallman to insist on adding the GNU prefix when Debian rolled out its software distribution. "The fork has since converged. Still, at the time, there was a concern that if the Linux community saw itself as a different thing as the GNU community, it might be a force for disunity."

.....

While some viewed it as politically grasping to describe the combination of GNU and Linux as a "variant" of GNU, Murdock, already sympathetic to the free software cause, saw Stallman's request to call Debian's version GNU/Linux as reasonable. "It was more for unity than for credit," he says.

虽然有不少人觉得把Linux看成是GNU工程的一个变种在政治上未免显得有点贪心，默多克却对自由软件情有独钟，他觉得斯托曼要求把Debian称为GNU/Linux是合理的。“团结开发者比争抢名号更为重要”，他说。

Requests of a more technical nature quickly followed. Although Murdock had been accommodating on political issues, he struck a firmer pose when it came to the design and development model of the actual software. What had begun as a show of solidarity soon became a running disagreement.

很快，默多克又收到了一些更为技术化的要求。尽管默多克已经习惯于这些政治化的问题，在实际的软件设计和开发模式上，他仍然坚持他自己的模式。这起先被看成是Debian开发社区一种团结的象征，但是慢慢的这成为Debian社区与GNU工程之间的一些矛盾。

“I can tell you that I’ve had my share of disagreements with him,” says Murdock with a laugh. “In all honesty Richard can be a fairly difficult person to work with.” The principal disagreement was over debugging. Stallman wanted to include debugging information in all executable programs, to enable users to immediately investigate any bugs they might encounter. Murdock thought this would make the system files too big and interfere with distribution. Neither was willing to change his mind.

默多克笑着说：“我可以告诉你，在很多方面我与斯托曼的看法都不一致。实事求是的说，理查德是一个很与之合作的人。”。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。

In 1996, Murdock, following his graduation from Purdue, decided to hand over the reins of the growing Debian project. He had already been ceding management duties to Bruce Perens, the hacker best known for his work on Electric Fence, a Unix utility released under the GPL. Perens, like Murdock, was a Unix programmer who had become enamored of GNU/Linux as soon as the operating system’s Unix-like abilities became manifest. Like Murdock, Perens sympathized with the political agenda of Stallman and the Free Software Foundation, albeit from afar.

1996年，默多克从普渡大学毕业后，打算找人接手正在蓬勃发展的Debian项目。那时候，他已经把主要的管理职责都转交给了布鲁斯·佩伦斯B，布鲁斯·佩伦斯是以GPL发布的Unix下一个知名的软件Electric Fence的作者。佩伦斯与默多克一样，当GNU/Linux的与Unix的巨大相似性一经展现就喜欢上了它。他也

和默多克一样，对斯托曼和自由软件基金会的政治目标抱有极大的兴趣，即使还有没有真正与他们有过实际的接触。

“I remember after Stallman had already come out with the *GNU Manifesto*, GNU Emacs, and GCC, I read an article that said he was working as a consultant for Intel,” says Perens, recalling his first brush with Stallman in the late 1980s. “I wrote him asking how he could be advocating free software on the one hand and working for Intel on the other. He wrote back saying, ‘I work as a consultant to produce free software.’ He was perfectly polite about it, and I thought his answer made perfect sense.”

“我记得斯托曼设计出了《*GNU宣言*》，GNU Emcas和GCC后，有篇文章说他正在为Intel提供咨询服务。”佩伦斯说，回忆起他在80年代末期第一次与斯托曼接触的情景。“我写信给他，问他是怎么做到一方面宣扬自由软件又一方面为Intel工作的。他回信说：我是作为Intel开发自由软件的咨询师身份为他工作的。他的回答很有礼貌，而且也很有道理。“

As a prominent Debian developer, however, Perens regarded Murdock’s design battles with Stallman with dismay. Upon assuming leadership of the development team, Perens says he made the command decision to distance Debian from the Free Software Foundation. “I decided we did not want Richard’s style of micro-management,” he says.

然而，作为一个杰出的Debian开发者，佩伦斯对于默多克和斯托曼之间的设计斗争感到失望。作为开发团队的领导者，佩伦斯说他决定让Debian与自由软件基金会保持距离。他说，“我决定不采用理查德风格的微管理模式。”

According to Perens, Stallman was taken aback by the decision but had the wisdom to roll with it. “He gave it some time to cool off and sent a message that we really needed a relationship. He requested that we call it GNU/Linux and left it at that. I decided that was fine. I made the decision unilaterally. Everybody breathed a sigh of relief.”

佩伦斯表示，斯托曼对这样的决定大吃一惊，但仍然有能力去驾驭它。“他给了我们一些时间冷静下来，然后发了一个邮件说我们需要建立一种合作关系。他要求我们把这个系统叫作GNU/Linux，并一直这样称呼它。我觉得这样是可以的。我单方面做出了这样的决定，每个人都舒了一口气。

Over time, Debian would develop a reputation as the hacker’s version of

GNU/Linux, alongside Slackware, another popular distribution founded during the same 1993-1994 period. However, Slackware contained some nonfree programs, and Debian after its separation from GNU began distributing non-free programs too. Despite labeling them as “non-free” and saying that they were “not officially part of Debian,” proposing these programs to the user implied a kind of endorsement for them. As the GNU Project became aware of these policies, it came to recognize that neither Slackware nor Debian was a GNU/Linux distro it could recommend to the public.

在很长的一段时间里，Debian都被看成是最适合黑客们的Linux发行版之一，在1993至1994年这段时间，Slackware也是一个非常流行的发行版。.....

Outside the realm of hacker-oriented systems, however, GNU/Linux was picking up steam in the commercial Unix marketplace. In North Carolina, a Unix company billing itself as Red Hat was revamping its business to focus on GNU/Linux. The chief executive officer was Robert Young, the former *Linux Journal* editor who in 1994 had put the question to Linus Torvalds, asking whether he had any regrets about putting the kernel under the GPL. To Young, Torvalds’ response had a “profound” impact on his own view toward GNU/Linux. Instead of looking for a way to corner the GNU/Linux market via traditional software tactics, Young began to consider what might happen if a company adopted the same approach as Debian – i.e., building an operating system completely out of free software parts. Cygnus Solutions, the company founded by Michael Tiemann and John Gilmore in 1990, was already demonstrating the ability to sell free software based on quality and customizability. What if Red Hat took the same approach with GNU/Linux?

虽然Linux不是一个真正意义上面向黑客的系统，但是Linux在商业Unix的市场中也找到了自己的位置。在北卡罗莱纳州，一个名叫Red Hat的Unix公司，把他们的注意力慢慢转向了Linux。Red Hat的首席执行官是罗伯特·杨，他是前*Linux杂志*的编辑，并在1994年时采访过林纳斯·托瓦兹，问他对于把内核以GPL来发布是否后悔。对于杨来说，托瓦兹的回答让他对Linux有了进一步深层次的认识，他不再寻求通过传统软件销售策略的方式把GNU/Linux边缘化，而是开始考虑如果一家公司像Debian那样思考问题，会出现什么情况。比如，发布一个完全由自由软件构成的操作系统。1990年，迈克尔·蒂曼和约翰·吉尔摩成立的Cygnus Solutions的公司，早已经向世人证明通过销售自由软件相关的定制化服务是可以盈利的。如果Red Hat对于GNU/Linux采用类似的策略会如何?

“In the western scientific tradition we stand on the shoulders of giants,” says Young, echoing both Torvalds and Sir Isaac Newton before him. “In business, this translates to not having to reinvent wheels as we go along. The beauty of [the GPL] model is you put your code into the public domain. If you’re an independent software vendor and you’re trying to build some application and you need a modem-dialer, well, why reinvent modem dialers? You can just steal PPP off of Red Hat [GNU/]Linux and use that as the core of your modem-dialing tool. If you need a graphic tool set, you don’t have to write your own graphic library. Just download GTK. Suddenly you have the ability to reuse the best of what went before. And suddenly your focus as an application vendor is less on software management and more on writing the applications specific to your customer’s needs.” However, Young was no free software activist, and readily included nonfree programs in Red Hat’s GNU/Linux system.

“在西方的科学传统上，我们需要站在巨人的肩膀上。”杨说，指的是托瓦兹和艾萨克·牛顿爵士。“在商业上，这就告诫我们，应该在前进的途中避免重新造轮子。GPL模式的美在于把你的代码放入公有领域。如果你是一个独立软件提供商并且想开发一些软件，你需要一个调制解调器拨号程序，但是你没有必须重新开发一个拨号程序。你可以直接从Red Hat Linux中“盗取”PPP的实现，并把它作为你的拨号程序的核心。如果你需要一个图形工具集，你不需要重新去实现一个你自己的图形库。你只需要下载GTK，就可以立刻应用前人的最佳成果。于是，你就可以专心做你的软件提供商，少花一点时间在软件管理上，而多花一些时间在你客户所想要的功能上。”

Young wasn’t the only software executive intrigued by the business efficiencies of free software. By late 1996, most Unix companies were starting to wake up and smell the brewing source code. The GNU/Linux sector was still a good year or two away from full commercial breakout mode, but those close enough to the hacker community could feel it: something big was happening. The Intel 386 chip, the Internet, and the World Wide Web had hit the marketplace like a set of monster waves; free software seemed like the largest wave yet.

杨并不是唯一一个从自由软件的高效性中受到启发的CEO。1996年末，大部分Unix公司开始清醒过来，并嗅到了源代码的香味。那时候，Linux距离成为一个完整的商用操作系统还有一两年的路要走，但是黑客社区还是感觉到这一天已经离得很近了：一场巨大的变革即将到来。Intel 386芯片，Internet和万维网像

一波怪物到来一样深入的影响了整个市场，而Linux这个可以自由使用其源代码并以宽松的许可证发布的程序包，则是这一波怪物中最大的一个。

For Ian Murdock, the wave seemed both a fitting tribute and a fitting punishment for the man who had spent so much time giving the free software movement an identity. Like many Linux aficionados, Murdock had seen the original postings. He'd seen Torvalds' original admonition that Linux was "just a hobby." He'd also seen Torvalds' admission to Minix creator Andrew Tanenbaum: "If the GNU kernel had been ready last spring, I'd not have bothered to even start my project." Like many, Murdock knew that some opportunities had been missed. He also knew the excitement of watching new opportunities come seeping out of the very fabric of the Internet.

对于试图讨好斯托曼但后来又厌恶斯托曼的徽管理风格的伊恩·默多克来说，他在自由软件运动中花费了很多的精力，这波变革看上去既是对他的赞美又是对他的惩罚。与很多Linux发烧友一样，默多克看到过托瓦兹最初的那个帖子，他看到过托瓦兹最初把Linux看作是"兴趣爱好"的想法。他也看到过托瓦兹对Minix创造者安德鲁·塔嫩鲍姆的敬意："如果GNU内核在去年春天时就已经完成，我就不会再启动我自己的这个项目" 很多人一样，默多克知道，这种好机会已经错过了。他也看到了Internet即将带来的巨大的机遇。

"Being involved with Linux in those early days was fun," recalls Murdock. "At the same time, it was something to do, something to pass the time. If you go back and read those old [comp.os.minix] exchanges, you'll see the sentiment: this is something we can play with until the Hurd is ready. People were anxious. It's funny, but in a lot of ways, I suspect that Linux would never have happened if the Hurd had come along more quickly."

"在早期参与到Linux项目中充满着乐趣"，默多克回忆到。"在同一段时间，有些东西在向前进步，有些东西则成为过眼烟云。如果你回头去看看[comp.os.minix]上那些老贴子，你会看到这样的观点：这是在Hurd开发完成前我们可以先玩着的东西。人们总是性急的。它很好玩，但是如果HURD来得更早一些的话，我想Linux可能就根本不会出现。"

By the end of 1996, however, such "what if" questions were already moot, because Torvalds' kernel had gained a critical mass of users. The 36-month window had closed, meaning that even if the GNU Project had rolled out its Hurd kernel, chances were slim anybody outside the hard-core hacker community would have noticed. Linux, by filling the GNU system's last gap,

had achieved the GNU Project's goal of producing a Unix-like free software operating system. However, most of the users did not recognize what had happened: they thought the whole system was Linux, and that Torvalds had done it all. Most of them installed distributions that came with nonfree software; with Torvalds as their ethical guide, they saw no principled reason to reject it. Still, a precarious freedom was available for those that appreciated it.

1996年底，这些"万一"问题已经盖棺定论。称呼它为Linux或是GNU/Linux，用户们已经用行动证明了。36个月的时间窗口已经关闭，意味着即使GNU工程可以开发出HURD内核，除了GNU的核心黑客们也不会有更多人会注意它。第一个类Unix的自由软件操作系统就在这里，并且充满活力。黑客们所要做的就是安静的坐下来，等待下一波新的想法出现在他们的头脑中。即使是长着蓬乱的头发的理查德.斯托曼本人也不例外。.....

Open Source 开源 {#chapter:open source}

In November, 1995, Peter Salus, a member of the Free Software Foundation and author of the 1994 book, *A Quarter Century of Unix*, issued a call for papers to members of the GNU Project's "system-discuss" mailing list. Salus, the conference's scheduled chairman, wanted to tip off fellow hackers about the upcoming Conference on Freely Redistributable Software in Cambridge, Massachusetts. Slated for February, 1996, and sponsored by the Free Software Foundation, the event promised to be the first engineering conference solely dedicated to free software and, in a show of unity with other free software programmers, welcomed papers on "any aspect of GNU, Linux, NetBSD, 386BSD, FreeBSD, Perl, Tcl/tk, and other tools for which the code is accessible and redistributable." Salus wrote:

1995年11月，自由软件基金会成员彼得·萨卢斯（他于1994年出版了《*Unix的四分之一世纪*》一书），在GNU工程的system-discuss邮件列表中发布了一个通知，为即将在马萨诸塞州召开的"在剑桥自由发布软件"大会征集黑客的论文。这场由自由软件基金会赞助的大会将于1996年2月召开，这是第一个纯由自由软件开发工程师的参加的大会，为了表现出自由软件程序员的团结一致，大会欢迎各种有关GNU、Linux、NetBS、386BSD、FreeBSD、Perl、Tcl/tk和任何可以自由获取和分发的工具软件相关的文章来大会交流。萨卢斯在通知中写道：

> Over the past 15 years, free and low-cost software has become ubiquitous. This conference will bring together implementers of several different types of freely redistributable software and publishers of such software (on various media). There will be tutorials and refereed papers, as well as keynotes by Linus Torvalds and Richard Stallman. 在过去的15年中，自由和低成本的软件变得越来越重要。本次大会将召集各种可以自由分发的软件的开发者和通过各种渠道发布这类软件的发布者。大会上会有各种教程和参考论文，也会有林纳斯·托瓦兹和理查德·斯托曼的主题演讲。

Among the recipients of Salus' email was conference committee member Eric S. Raymond. Although not the leader of a project or company like the various other members of the list, Raymond had built a tidy reputation within the hacker community for some software projects and as editor of *The New Hacker's Dictionary*, a greatly enlarged version of *The Hacker's Dictionary* published a decade earlier by Guy Steele.

大会组委会成员埃里克·斯蒂芬·雷蒙德是最早收到萨卢斯的电子邮件的人之一，虽然他不像邮件列表中其它人那样是某个项目的开发负责人或是代表某个公司，但他在GNU Emacs的中的贡献以及作为《*新黑客词典*》（一本收录黑客社会各种黑话的书）的编辑的身份，使他在黑客社区中享有一定的声誉。

For Raymond, the 1996 conference was a welcome event. Although he did not thoroughly support the free software movement's ideas, he had contributed to some GNU programs, in particular to GNU Emacs. Those contributions stopped in 1992, when Raymond demanded authority to make changes in the official GNU version of GNU Emacs without discussing them with Stallman, who was directly in charge of Emacs development. Stallman rejected the demand, and Raymond accused Stallman of "micro-management." "Richard kicked up a fuss about my making unauthorized modifications when I was cleaning up the Emacs LISP libraries," Raymond recalls. "It frustrated me so much that I decided I didn't want to work with him anymore."

对于雷蒙德来说，1996年的这次大会是一个重要的契机。他在二十世纪80年代在GNU工程中非常活跃，但到了1992以后他就已经不再直接参与这个工程。与许多与他有一样情况的人那样，他对斯托曼的"微管理"风格颇有微词。"理查德对于我在清理Emacs LISP库中的代码时进行的修改非常不满。"雷蒙德回忆到，"这使得我感觉到很沮丧，我再也不想与他一起合作了。"

Despite the falling out, Raymond remained active in the free software community. So much so that when Salus suggested a conference pairing Stallman and Torvalds as keynote speakers, Raymond eagerly seconded the idea. With Stallman representing the older, wiser contingent of ITS/Unix hackers and Torvalds representing the younger, more energetic crop of Linux hackers, the pairing indicated a symbolic show of unity that could only be beneficial, especially to ambitious younger (i.e., below 40) hackers such as Raymond. “I sort of had a foot in both camps,” Raymond says.

除了不再直接参与GNU工程，雷蒙德在自由软件社区中仍然非常活跃。 当萨卢斯建议在大会上由斯托曼和托瓦兹做主题演讲时，雷蒙德非常强烈的建议选用别的方案。斯托曼是老一辈ITS/Unix黑客的传承人而托瓦兹是新一代的更有活力的Linux黑客的代表，这样的搭配是一种联盟的象征，但只会对一部分像雷蒙德这样的年经有抱负的黑客有好处。"我像是那个脚踏两只船的人。” 雷蒙德说。

By the time of the conference, the tension between those two camps had become palpable. Both groups had one thing in common, though: the conference was their first chance to meet the Finnish *wunderkind* in the flesh. Surprisingly, Torvalds proved himself to be a charming, affable speaker. Possessing only a slight Swedish accent, Torvalds surprised audience members with his quick, self-effacing wit.

会议召开的日子一天天临近，两个团体之间的紧张关系也变得更加明显。两个团体在一个问题上的观点是一致的：这次会议是他们第一次有机会当面见到芬兰神童托瓦兹。托瓦兹出人意料的证明了自己是一个出色和平易进人的演讲者。除了一点点轻徽的瑞典口音，托瓦兹敏捷的思路和谦虚的才智让到场的听众感到惊讶。

Even more surprising, says Raymond, was Torvalds’ equal willingness to take potshots at other prominent hackers, including the most prominent hacker of all, Richard Stallman. By the end of the conference, Torvalds’ half-hacker, half-slacker manner was winning over older and younger conference-goers alike.

更让人吃惊的是，雷蒙德说，托瓦兹对其它杰出黑客们的无情抨击，包括所有黑客中最杰出的那位理查德·斯托曼。在大会的最后，托瓦兹的半黑客半懒鬼的作风赢得了新老两代与会人员的青睐。

“It was a pivotal moment,” recalls Raymond. “Before 1996, Richard was the

only credible claimant to being the ideological leader of the entire culture. People who dissented didn't do so in public. The person who broke that taboo was Torvalds."

"这是一个关键的转折点，"雷蒙德回忆道，"在1996年之前，理查德是整个自由软件文化的唯一一个权威的精神领袖。持有不同意见的人都并不在公众面前直接表达。托瓦兹是第一个打破这个禁忌的人。"

The ultimate breach of taboo would come near the end of the show. During a discussion on the growing market dominance of Microsoft Windows or some similar topic, Torvalds admitted to being a fan of Microsoft's PowerPoint slideshow software program. From the perspective of old-line software purists, it was like bragging about one's slaves at an abolitionist conference. From the perspective of Torvalds and his growing band of followers, it was simply common sense. Why shun convenient proprietary software programs just to make a point? They didn't agree with the point anyway. When freedom requires a sacrifice, those who don't care about freedom see the sacrifice as self-denial, rather than as a way to obtain something important. Being a hacker wasn't about self-denial, it was about getting the job done, and "the job," for them, was defined in practical terms.

打破禁忌的最后一击出现在大会的最后。在一个有关微软Windows在市场中占有垄断地位的讨论中，托瓦兹承认他自己是微软的PowerPoint这个幻灯片制做软件的发烧友。从老一辈自由软件纯化论者的眼光中，这就像是一个摩门教徒在教堂中自夸自己对威士忌酒的热爱。 从托瓦兹和他越来越多的追随者的言论中，这样的观点并不奇怪。为什么为了表明自己观点就得刻意去回避私有软件呢？做为黑客，不是为了去忍受，而是为了把工作完成。

"That was a pretty shocking thing to say," Raymond remembers. "Then again, he was able to do that, because by 1995 and 1996, he was rapidly acquiring clout."

"这是一个振聋发聩的观点，"雷蒙德回忆说，"然而，他可以做到这一点，因为1995年和1996年，他正不断的接受打击。"

Stallman, for his part, doesn't remember any tension at the 1996 conference; he probably wasn't present when Torvalds made that statement. But he does remember later feeling the sting of Torvalds' celebrated "cheekiness." "There was a thing in the Linux documentation which says print out the GNU coding

standards and then tear them up," says Stallman, recalling one example. "When you look closely, what he disagreed with was the least important part of it, the recommendation for how to indent C code."

但斯托曼表示他完全不记得1996年的大会上有什么紧张的气氛，只不过，他记得是那次被托瓦兹的厚脸皮刺激到了而已。"在Linux的文档中，有一段描述，让人们把GNU编码标准打印出来，然后再把它们撕烂。"斯托曼举例
道。.....

"OK, so he disagrees with some of our conventions. That's fine, but he picked a singularly nasty way of saying so. He could have just said, 'Here's the way I think you should indent your code.' Fine. There should be no hostility there."

"好吧，所以他对我们的一些做法持有反对意见。有不同意见是可以的，但是他选择了一种格外令人生厌的方式去表达这样的观点。他完全可以换个说法："我觉得你可以使用另一种方式来缩进你的代码。这样就不会有攻击性了。"

For Raymond, the warm reception other hackers gave to Torvalds' comments confirmed a suspicion: the dividing line separating Linux developers from GNU developers was largely generational. Many Linux hackers, like Torvalds, had grown up in a world of proprietary software. They had begun contributing to free software without perceiving any injustice in nonfree software. For most of them, nothing was at stake beyond convenience. Unless a program was technically inferior, they saw little reason to reject it on licensing issues alone. Some day hackers might develop a free software alternative to PowerPoint. Until then, why criticize PowerPoint or Microsoft; why not use it?

对于雷蒙德来说，其它黑客给于托瓦兹的热烈评价和真诚的欢迎从另一个角度上证实了他的怀疑。Linux开发者与GNU/Linux开发者之间几乎存在着一个代沟。很多像托瓦兹那样的Linux黑客是在私有软件的世界中成长起来的。除非软件的质量确实很低下，否则没什么人会对软件的许可证问题发出抱怨。在自由软件的世界中，也许隐藏着一个PowerPoint的自由软件替代品。但是，在黑客们真正开始转向使用这个软件前，为什么要嫉妒微软开发了这样一个好用的软件并对此保留版权呢?

This was an example of the growing dispute, within the free software community, between those who valued freedom as such, and those who mainly valued powerful, reliable software. Stallman referred to the two camps as political parties within the community, calling the former the "freedom party."

The supporters of the other camp did not try to name it, so Stallman disparagingly called it the “bandwagon party” or the “success party,” because many of them presented “more users” as the primary goal.

.....

In the decade since launching the GNU Project, Stallman had built up a fearsome reputation as a programmer. He had also built up a reputation for intransigence both in terms of software design and people management. This was partly true, but the reputation provided a convenient excuse that anyone could cite if Stallman did not do as he wished. The reputation has been augmented by mistaken guesses.

在GNU工程开始的前十年中，斯托曼已经在程序员中建立起了威信。他也在软件设计和人员管理中建立了一种不妥协声望。.....

For example, shortly before the 1996 conference, the Free Software Foundation experienced a full-scale staff defection. Brian Youmans, a current FSF staffer hired by Salus in the wake of the resignations, recalls the scene: “At one point, Peter [Salus] was the only staff member working in the office.” The previous staff were unhappy with the executive director; as Bryt Bradley told her friends in December, 1995:

在1996年大会开始前不久，自由软件基金会遭遇了一次大面积的离职事件，事件的起因很大程度是因为斯托曼。Brain Youmans，Salus招聘的现任的FSF职员回忆到：“在那时候，Peter [Salus]是留在办公室中的唯一职员”。.....

> [name omitted] (the Executive Director of the FSF) decided to come back from Medical/Political Leave last week. The office staff (Gena Bean, Mike Drain, and myself) decided we could not work with her as our supervisor because of the many mistakes she had made in her job tasks prior to her taking a leave. Also, there had been numerous instances where individuals were threatened with inappropriate firing and there were many instances of what we felt were verbal abuse from her to ALL members of the office staff. We requested (many times) that she not come back as our supervisor, but stated that we were willing to work with her as a co-worker. Our requests were ignored. We quit.

.....

The executive director in question then gave Stallman an ultimatum: give her total autonomy in the office or she would quit. Stallman, as president of the FSF, declined to give her total control over its activities, so she resigned, and he recruited in Peter Salus to replace her.

.....

When Raymond, an outsider, learned that these people had left the FSF, he presumed Stallman was at fault. This provided confirmation for his theory that Stallman's personality was the cause of any and all problems in the GNU Project.

.....

Raymond had another theory: recent delays such as the Hurd and recent troubles such as the Lucid-Emacs schism reflected problems normally associated with software project management, not software code development.

.....

Shortly after the Freely Redistributable Software Conference, Raymond began working on his own pet software project, a mail utility called "fetchmail." Taking a cue from Torvalds, Raymond issued his program with a tacked-on promise to update the source code as early and as often as possible. When users began sending in bug reports and feature suggestions, Raymond, at first anticipating a tangled mess, found the resulting software surprisingly sturdy. Analyzing the success of the Torvalds approach, Raymond issued a quick analysis: using the Internet as his "petri dish" and the harsh scrutiny of the hacker community as a form of natural selection, Torvalds had created an evolutionary model free of central planning.

.....

What's more, Raymond decided, Torvalds had found a way around Brooks' Law. First articulated by Fred P. Brooks, manager of IBM's OS/360 project and author of the 1975 book, *The Mythical Man-Month*, Brooks' Law held that adding developers to a project only resulted in further project delays. Believing

as most hackers that software, like soup, benefits from a limited number of cooks, Raymond sensed something revolutionary at work. In inviting more and more cooks into the kitchen, Torvalds had actually found a way to make the resulting software *better*.

另一方面，雷蒙德觉得，托瓦兹找到了一种突破Brooks法则的方式。在IBM OS/360项目经理Fred P. Brooks于1975年出版的《*人月神话*》一书中，Brooks提出了Brooks法则：向一个软件项目中增加开发者的作法只会造成项目的进一步延迟。对于黑客们来说，写软件就像做汤，增加厨师的人数对于改善汤的口味并没有太多作用。雷蒙德从中感觉到了一些革命性的变化：托瓦兹确实在聘请更多的厨师进入厨房的同时，做出了*更好*的软件。

Raymond put his observations on paper. He crafted them into a speech, which he promptly delivered before a group of friends and neighbors in Chester County, Pennsylvania. Dubbed “The Cathedral and the Bazaar,” the speech contrasted the “Bazaar” style originated by Torvalds with the “Cathedral” style generally used by everyone else.

雷蒙德把他的发现写成了论文，并把它变成若干次演讲，在宾夕法尼亚州Chester County的一些朋友和邻居们面前宣传。在一个名为《教堂与集市》的演讲中，他比较了GNU工程与托瓦兹和内核黑客位使用的管理风格进行了比较。。。。。。。。。。。。。。。。。。。。。

Raymond says the response was enthusiastic, but not nearly as enthusiastic as the one he received during the 1997 Linux Kongress, a gathering of GNU/Linux users in Germany the next spring.

雷蒙德说这些演讲的反响很热烈，不过都不如次年春天他在德国召开的1997 Linux Kongress上的那次演讲。

“At the Kongress, they gave me a standing ovation at the end of the speech,” Raymond recalls. “I took that as significant for two reasons. For one thing, it meant they were excited by what they were hearing. For another thing, it meant they were excited even after hearing the speech delivered through a language barrier.”

”在Kongress上，演讲结束后听众都起身鼓掌“，雷蒙德回忆道。”我把这次成功归纳成两个因素。首先，这说明听众们为所听到的内容感到激动。其次，这意味着即使存在一些语言交流上的障碍，听众们依然感到兴奋。“

Eventually, Raymond would convert the speech into a paper, also titled “The Cathedral and the Bazaar.” The paper drew its name from Raymond’s central analogy. Previously, programs were “cathedrals,” impressive, centrally planned monuments built to stand the test of time. Linux, on the other hand, was more like “a great babbling bazaar,” a software program developed through the loose decentralizing dynamics of the Internet.

最后，雷蒙德把这些演讲的内容写成了论文，同样起名为《教堂与集市》。这篇论文的题目来自雷蒙德的核心思想。GNU程序就是“教堂”，它们都是有计划的修建的宏伟的黑客精神的纪念碑，经得起时间的考验。而另一方面，Linux则更像是一个嘈杂的大集市，它是在Internet去中心化的松散组织结构中开发出来的。

Raymond’s paper associated the Cathedral style, which he and Stallman and many others had used, specifically with the GNU Project and Stallman, thus casting the contrast between development models as a comparison between Stallman and Torvalds. Where Stallman was his chosen example of the classic cathedral architect – i.e., a programming “wizard” who could disappear for 18 months and return with something like the GNU C Compiler – Torvalds was more like a genial dinner-party host. In letting others lead the Linux design discussion and stepping in only when the entire table needed a referee, Torvalds had created a development model very much reflective of his own laid-back personality. From Torvalds’ perspective, the most important managerial task was not imposing control but keeping the ideas flowing.

相似的类比同样也适用于斯托曼和托瓦兹。斯托曼代表着经典的教堂架构，他像一个程序魔法师一样，在消失了18月以后，带来了类似于GNU C编译器这样的神作。而托瓦兹则像是一个亲切的宴会主人。他让别人来领导Linux的设计讨论，并只在关键时候提出一些参考建议。托瓦兹所创建的这种开发模式正好反映了的懒散的个性。从托瓦兹的观点来看，最好的管理工作不是要加强对事情的控制，而是要保持思维的活跃度。

Summarized Raymond, “I think Linus’s cleverest and most consequential hack was not the construction of the Linux kernel itself, but rather his invention of the Linux development model.”

雷蒙德总结说：，“我觉得Linux最聪明的地方和最重要的hack就是创造出来Linux内核，而不是他所创造的Linux开发模式。”

If the paper's description of these two styles of development was perceptive, its association of the Cathedral model specifically with Stallman (rather than all the others who had used it, including Raymond himself) was sheer calumny. In fact, the developers of some GNU packages including the GNU Hurd had read about and adopted Torvalds' methods before Raymond tried them, though without analyzing them further and publicly championing them as Raymond's paper did. Thousands of hackers, reading Raymond's article, must have been led to a negative attitude towards GNU by this smear.

In summarizing the secrets of Torvalds' managerial success, Raymond attracted the attention of other members of the free software community for whom freedom was not a priority. They sought to interest business in the use and development of free software, and to do so, decided to cast the issue in terms of the values that appeal to business: powerful, reliable, cheap, advanced. Raymond became the best-known proponent of these ideas, and they reached the management of Netscape, whose proprietary browser was losing market share to Microsoft's equally proprietary Internet Explorer. Intrigued by a speech by Raymond, Netscape executives took the message back to corporate headquarters. A few months later, in January, 1998, the company announced its plan to publish the source code of its flagship Navigator web browser in the hopes of enlisting hacker support in future development.

在总结托瓦兹的成功管理经验方面，雷蒙德做自己也完成得的非常出色。.....跟预期的情况一样，听众并不仅仅被黑客所震惊，也有很多人对自由软件运动的快速成长表达了强烈的兴趣。Netscap就是其中的一个，。这家成立于加州山景城的创业公司，刚刚经历一了一场场持续了三年的与微软争夺Web浏览器的市场的争夺。出于对雷蒙德演讲的好奇和希望赢回失去的市场份额的期望目的，Netscape的执行官们居们把这次演讲的信息带回了公司总部。几个月以后，1998年1月，Netscape宣布计划把它的旗舰级产品Navigator Web浏览器的代码公开，希望能得到黑客们的支持，完成以后的开发。.....

When Netscape CEO Jim Barksdale cited Raymond's "Cathedral and the Bazaar" essay as a major influence upon the company's decision, the company instantly elevated Raymond to the level of hacker celebrity. He invited a few people including Larry Augustin, founder of VA Research which sold workstations with the GNU/Linux operating system preinstalled; Tim O'Reilly, founder of the publisher O'Reilly & Associates; and Christine

Peterson, president of the Foresight Institute, a Silicon Valley think tank specializing in nanotechnology, to talk. “The meeting’s agenda boiled down to one item: how to take advantage of Netscape’s decision so that other companies might follow suit?”

Netscape CEO Jim Barksdale说雷蒙德的“教堂与集市”文章对于公司做出这样的决策有着深远的影响，公司很快把雷蒙德地位抬升到了黑客界的知名人士。为了紧紧抓住这次机会，雷蒙德出差去西海岸去进行采访，向Netscape的执行官们提供建议，并且参与了Netscape Navigator源代码开放庆祝的聚会。Navigator的源代码的code name是“Mozilla”，这个名字一方面象征着这个程序的源代码有30万行的代码，同时也象征着它的血统。作为一个Mosaic浏览器的非商业化版本，Marc Adressen在Illionis大学开发这个浏览器时也参考了Mosaic的设计，这就再次证明了大部分程序员在打算开发一个新的程序时，总是会借鉴一些现有的、可以修改的程序。.....Tim O’Reilly当时也在Linux Kongress的会场聆听演讲，他是O’Reilly & Associcates出版公司的创始人，这家公司专注于出版软件使用手册和软件相关的书籍（同样也是本书的出版商）。听完了雷蒙德在Kongress上的演讲后，O’Reilly迅速地邀请雷蒙德参加下半年在加州Monterey举办的首届Perl大会，在会议上再次分享这个主题。.....

Raymond doesn’t recall the conversation that took place, but he does remember the first complaint addressed. Despite the best efforts of Stallman and other hackers to remind people that the word “free” in free software stood for freedom and not price, the message still wasn’t getting through. Most business executives, upon hearing the term for the first time, interpreted the word as synonymous with “zero cost,” tuning out any follow-up messages in short order. Until hackers found a way to get past this misunderstanding, the free software movement faced an uphill climb, even after Netscape.

雷蒙德不记得这次讨论的内容，但是他记得这次会议中的提到的另一个问题。不管斯托曼和其他黑客如何努力的去提醒人们自由软件中的“Free”一词是指“自由”而不是“免费”，但这一点仍然没有得到大众的真正认知。第一次听到这个名词时，许多商业管理者都把它当作是“零成本”的同义词，而并不注意它蕴含着的其它相关信息。除非黑客们能找到一种方法解决这个认识上的偏差，否则自由软件运动就会面临一个高耸难攀的山峰，即使有Netscape的案例供参考，也很难一举登顶。

Peterson, whose organization had taken an active interest in advancing the

free software cause, offered an alternative: “open source.”

于是，彼德森成立了一个组织，专门致力于推广自由软件， 他们提供了另一个词语：开源。

Looking back, Peterson says she came up with the “open source” term while discussing Netscape’s decision with a friend in the public relations industry. She doesn’t remember where she came upon the term or if she borrowed it from another field, but she does remember her friend disliking the term.

彼德森回忆道，她是在与一个朋友讨论Netscape的决定的时候，想到了“开源”这个词语。她不记得是她自己首创了这个词语，还是从别的什么领域借用了这个词语，但是她确实记得她的朋友并不喜欢这个词语。

At the meeting, Peterson says, the response was dramatically different. “I was hesitant about suggesting it,” Peterson recalls. “I had no standing with the group, so started using it casually, not highlighting it as a new term.” To Peterson’s surprise, the term caught on. By the end of the meeting, most of the attendees, including Raymond, seemed pleased by it.

彼德森说，但是在这次会议上，这个词语得到的反馈却是截然不同的。彼德森回忆说：“我很犹豫要不要提出这个建议，我对这个组织并不了解，所以我在不经意中去使用这个词语，而不是把它作为一个新词语去强调。”然而，这个词语出乎意料的受到了人们的关注。在这次会议结束时，大部分的与会者，包括雷蒙德，对于这个词语都非常满意。

Raymond says he didn’t publicly use the term “open source” as a substitute for “free software” until a day or two after the Mozilla launch party, when O’Reilly had scheduled a meeting to talk about free software. Calling his meeting “the Freeware Summit,” O’Reilly says he wanted to direct media and community attention to the other deserving projects that had also encouraged Netscape to release Mozilla. “All these guys had so much in common, and I was surprised they didn’t all know each other,” says O’Reilly. “I also wanted to let the world know just how great an impact the free software culture had already made. People were missing out on a large part of the free software tradition.”

雷蒙德说，在Mozilla发布会前几天O’Reilly组织的一次讨论自由软件的会议之前，他从来不公开使用“开源”这个词语来替代“自由软件”。O’Reilly把这次会议叫作“免费软件峰会”，期望把媒体和社区的注意力吸引到其它一些对于Netscape

发布Mozilla具有有价值的项目上去。“这些家伙有如此多的相似之处，但我很惊讶他们互相之间谁都不了解谁。”O’Reilly说，“我同时也想让世界知道自由软件文化已经产生了如此深远的影响。人们如果再不对自由软件引起重视，就是错过了一次很好的机会。”

In putting together the invite list, however, O’Reilly made a decision that would have long-term political consequences. He decided to limit the list to west-coast developers such as Wall, Eric Allman, creator of sendmail, and Paul Vixie, creator of BIND. There were exceptions, of course: Pennsylvania-resident Raymond, who was already in town thanks to the Mozilla launch, earned an quick invite. So did Virginia-resident Guido van Rossum, creator of Python. “Frank Willison, my editor in chief and champion of Python within the company, invited him without first checking in with me,” O’Reilly recalls. “I was happy to have him there, but when I started, it really was just a local gathering.”

邀请这些人聚集在一起，是O’Reilly一个深思熟虑的决定，并将产生深远的政治影响。他只打算邀请西海岸的开发者们，比如sendmail的作者Wall, Eric allman，BIND的作者Pal Vixie。不过也有一个人是例外，那就是长居在宾夕法尼亚的雷蒙德，因为他正好来参加Mozilla的发布会，所以也马上邀了他。同样还有弗吉尼亚的Guido van Rossum，他是Python语言的作者。“Frank Willison是我的主编，他是Python的斗士，他决定邀请Guido参加，甚至都没有跟我商量。”，O’Reilly回忆说，“我非常高兴能请到他出席，不过刚开始的时候，我确实只是想组织一个本地聚会。”

For some observers, the unwillingness to include Stallman’s name on the list qualified as a snub. “I decided not to go to the event because of it,” says Perens, remembering the summit. Raymond, who did go, says he argued for Stallman’s inclusion to no avail. The snub rumor gained additional strength from the fact that O’Reilly, the event’s host, had feuded publicly with Stallman over the issue of software-manual copyrights. Prior to the meeting, Stallman had argued that free software manuals should be as freely copyable and modifiable as free software programs. O’Reilly, meanwhile, argued that a value-added market for nonfree books increased the utility of free software by making it more accessible to a wider community. The two had also disputed the title of the event, with Stallman insisting on “Free Software” rather than “Freeware.” The latter term most often refers to programs which are available gratis, but which are not free software because their source code is not

released.

对于一些观察者来说，没有邀请斯托曼参加这次会议可以算是有点怠慢。“正是由于这个原因，我决定不去参加这次会议。” 佩伦斯回忆这次峰会时回忆道。雷蒙德参加了这次会议，并认为即使邀请斯托曼也是徒劳的。有关这次会议的组织者O’Reilly怠慢斯托曼的传言，在O’Reilly与斯托曼公开表明在软件手册的权限问题上的不同意见后，变得更能让人信服。在会议前，斯托曼曾表示，自由软件手册应该可以像自由软件本身一样自由的复制和修改。而O’Reilly那时认为非自由的图书是对自由软件的增值，它们能让自由软件被更多的社区所熟悉。O’Reilly和斯托曼还为这次会议的名称争论，斯托曼坚持要使用“自由软件”来代替不那么政治化的“免费软件”一词。

Looking back, O’Reilly doesn’t see the decision to leave Stallman’s name off the invite list as a snub. “At that time, I had never met Richard in person, but in our email interactions, he’d been inflexible and unwilling to engage in dialogue. I wanted to make sure the GNU tradition was represented at the meeting, so I invited John Gilmore and Michael Tiemann, whom I knew personally, and whom I knew were passionate about the value of the GPL but seemed more willing to engage in a frank back-and-forth about the strengths and weaknesses of the various free software projects and traditions. Given all the later brouhaha, I do wish I’d invited Richard as well, but I certainly don’t think that my failure to do so should be interpreted as a lack of respect for the GNU Project or for Richard personally.”

回头去看，O’Reilly仍然认为不邀请斯托曼的决定是怠慢了他。O’Reilly说：“在那个时候，我没有见过理查德·斯托曼本人，不过在我们的电子邮件沟通中，斯托曼看上去是个很强势的人，并且不太愿意加入对话。我希望能够保证在会议中展示出GNU的传统，所以我邀请了约翰·吉尔摩和Michael 蒂曼，这两个人我都比较熟悉，我知道他们对于GPL的价值都非常有热情，不过看上去他们更愿意参与有关自由软件项目和传统的优势和弱点的讨论。从后面发生的事情来看，我有点后悔没有邀请理查德，但是我没有这么做并不能看成是对GNU工程或理查德.斯托曼本人的不敬。”

Snub or no snub, both O’Reilly and Raymond say the term “open source” won over just enough summit-goers to qualify as a success. The attendees shared ideas and experiences and brainstormed on how to improve free software’s image. Of key concern was how to point out the successes of free software, particularly in the realm of Internet infrastructure, as opposed to playing up the

GNU/Linux challenge to Microsoft Windows. But like the earlier meeting at VA, the discussion soon turned to the problems associated with the term “free software.” O’Reilly, the summit host, remembers a comment from Torvalds, a summit attendee.

无论是不是怠慢了斯托曼，O’Reilly和雷蒙德说的“开源”一词成功的赢得了很多与会者的支持。与会者分享了各种想法和经验，并且一起进行了一场头脑风暴，讨论如何提升自由软件的形象。讨论的焦点是如何才能展示自由软件的成功之处，尤其是在实现Internet的基础架构方面，而不是讨论GNU/Linux是如何与微软Windows进行竞争的问题。不过跟很多早期在VA召开的会议一样，这些讨论的内容在不经意间就转向了讨论“自由软件”这个词自身所存在的问题。作为会议的主办者，O’Reilly记得峰会的与会者托瓦兹的一段很有见地评论。

“Linus had just moved to Silicon Valley at that point, and he explained how only recently that he had learned that the word ‘free’ had two meanings – free as in ‘libre’ and free as in ‘gratis’ – in English.”

“那时候，Linus刚刚搬到硅谷来，他解释说其实他本人也是最近才知道‘Free’这个词在英文中有两个意思，一个意思是自由，另一个意思是免费”。

Michael Tiemann, founder of Cygnus, proposed an alternative to the troublesome “free software” term: sourceware. “Nobody got too excited about it,” O’Reilly recalls. ``That’s when Eric threw out the term ‘open source.’

”

Gygnus的创始人迈克尔·蒂曼建议用sourceware一词来代替容易引起误解的”自由软件“一词。”不过大部分人都对这个词没什么兴趣。“O’Reilly回忆到，”直到Eric抛出了‘开源’一词。”

Although the term appealed to some, support for a change in official terminology was far from unanimous. At the end of the one-day conference, attendees put the three terms – free software, open source, or sourceware – to a vote. According to O’Reilly, 9 out of the 15 attendees voted for “open source.” Although some still quibbled with the term, all attendees agreed to use it in future discussions with the press. “We wanted to go out with a solidarity message,” O’Reilly says.

虽然“开源”这个词对一些人很有吸引力，但是要让它取代“自由软件”变为一个官方的用词还有很长的路要走。结束一天的会议前，与会者开始对三个词语进行

投票，“自由软件”、“开源”和“sourceware”。O’Reilly说，15名与会者中有9人把票投给了“开源”。虽然有一部分人对这个词还有些意见，不过所有的与会者都同意在以后与媒体的讨论中可以使用这个词汇。O’Reilly说，”我们希望发出一些团结的信号“。

The term didn’t take long to enter the national lexicon. Shortly after the summit, O’Reilly shepherded summit attendees to a press conference attended by reporters from the *New York Times*, the *Wall Street Journal*, and other prominent publications. Within a few months, Torvalds’ face was appearing on the cover of *Forbes* magazine, with the faces of Stallman, Perl creator Larry Wall, and Apache team leader Brian Behlendorf featured in the interior spread. Open source was open for business.

没过多久，”开源”这个词就进入了词典。这次峰会后不久，O’Reilly带着与会者们参加了一个纽约时报、华尔街日报和其它著名出版社记者参与的发布会。在短短几个月中，托瓦兹就登上了福布斯杂志的封面，正文中还出现了斯托曼、Perl的作者Larry Wall和Apache团队领导者Brain Behlendorf。开源开始向商业开放。

For summit attendees such as Tiemann, the solidarity message was the most important thing. Although his company had achieved a fair amount of success selling free software tools and services, he sensed the difficulty other programmers and entrepreneurs faced.

对于像蒂曼这样的参会者来说，发出团结的信息才是最重要的事情。仅管他的公司通过销售自由软件和相关服务已经捞到了第一桶金，他还是可以感觉到很多其它的程序员和企业在面对自由软件时还是困难重重。

“There’s no question that the use of the word free was confusing in a lot of situations,” Tiemann says. “Open source positioned itself as being business friendly and business sensible. Free software positioned itself as morally righteous. For better or worse we figured it was more advantageous to align with the open source crowd.”

蒂曼说：“毫无疑问，使用free这个词在很多场合都会造成误解。开源这个词则对于商业来说更加友好，也更容易引起商业的注意力。自由软件则更是关注于一种道德上的正义。不管好不好，我们都觉得与开源团队合作是有益无害的。”

Raymond called Stallman after the meeting to tell him about the new term

“open source” and ask if he would use it. Raymond says Stallman briefly considered adopting the term, only to discard it. “I know because I had direct personal conversations about it,” Raymond says.

斯托曼没有马上对“开源”这个新词语作出回应。雷蒙德说斯托曼简单了解了一个这个词语，其实只是为了抛弃它。“我知道这一点是因为我跟斯托曼就这个问题有过直接的对话。”

Stallman’s immediate response was, “I’ll have to think about it.” The following day he had concluded that the values of Raymond and O’Reilly would surely dominate the future discourse of “open source,” and that the best way to keep the ideas of the free software movement in public view was to stick to its traditional term.

。。。。。。。。。。。。。。。。。。。。

Later in 1998, Stallman announced his position: “open source,” while helpful in communicating the technical advantages of free software also encouraged speakers to soft-pedal the issue of software freedom. It avoided the unintended meaning of “gratis software” and the intended meaning of “freedom-respecting software” equally. As a means for conveying the latter meaning, it was therefore no use. In effect, Raymond and O’Reilly had given a name to the nonidealistic political party in the community, the one Stallman did not agree with.

到了1998年年底，斯托曼正式表明了立场：开源这个词在讨论自由软件技术优点时，是一个在沟通时便于使用的词语，但是它也诱使演讲者弱化软件自由的重要性。。。。。。。。。。。。。。。由于存在这样的不足，斯托曼坚持使用“自由软件”这个词汇。

In addition, Stallman thought that the ideas of “open source” led people to put too much emphasis on winning the support of business. While such support in itself wasn’t necessarily bad in itself, he expected that being too desperate for it would lead to harmful compromises. “Negotiation 101 would teach you that if you are desperate to get someone’s agreement, you are asking for a bad deal,” he says. “You need to be prepared to say no.” Summing up his position at the 1999 LinuxWorld Convention and Expo, an event billed by Torvalds himself as a “coming out party” for the “Linux” community, Stallman implored his fellow hackers to resist the lure of easy compromise.

TODO。。。。。。。1999年LinuxWorld对话与博览会，托瓦兹把它看成是Linux社区产出的宣传会，在这次会议上，斯托曼总结了他的立场，他恳求他的追随者抵抗住诱惑。

“Because we’ve shown how much we can do, we don’t have to be desperate to work with companies or compromise our goals,” Stallman said during a panel discussion. “Let them offer and we’ll accept. We don’t have to change what we’re doing to get them to help us. You can take a single step towards a goal, then another and then more and more and you’ll actually reach your goal. Or, you can take a half measure that means you don’t ever take another step, and you’ll never get there.”

斯托曼在一次小组讨论上说：“我们已经展示了很多我们已经取得的成就，所以我们不需要去讨好公司而放弃我们的目标。如果公司能向我们提供帮助，我们可以接受。如果公司不愿意，我们也没有必要去改变自己来争取这些帮助。只要你能不断向着目标慢慢前进，最终一定能实现目标。如果你们什么事情都采用折中的方式，不真正往前前进，就永远也不能实现目标。”

Even before the LinuxWorld show, however, Stallman was showing an increased willingness to alienate open source supporters. A few months after the Freeware Summit, O’Reilly hosted its second annual Perl Conference. This time around, Stallman was in attendance. During a panel discussion lauding IBM’s decision to employ the free software Apache web server in its commercial offerings, Stallman, taking advantage of an audience microphone, made a sharp denunciation of panelist John Ousterhout, creator of the Tcl scripting language. Stallman branded Ousterhout a “parasite” on the free software community for marketing a proprietary version of Tcl via Ousterhout’s startup company, Scriptics. Ousterhout had stated that Scriptics would contribute only the barest minimum of its improvements to the free version of Tcl, meaning it would in effect use that small contribution to win community approval for much a larger amount of nonfree software development. Stallman rejected this position and denounced Scriptics’ plans. “I don’t think Scriptics is necessary for the continued existence of Tcl,” Stallman said to hisses from the fellow audience members.

事实上，在LinuWorld之前，斯托曼已经表现出他正在慢慢疏远他的同盟。Freeware峰会后几个月，O’Reilly的主办了他的第二次年度Perl大会。这一次，斯托曼参加了会议。在一次称赞IBM决定在他的商用方案中整合Apache Web服

务器这个自由软件的小组讨论中，斯托曼抢过了听众的话筒，把矛头指向了讨论人John Ousterhout，他是Tcl脚本语言的作者。斯托曼把Ousterhout称为是自由软件社区的寄生虫，因为他通过他自己的名为Scriptics的创业公司稍售一个商业版本的Tcl。“我不觉得Scriptics是Tcl继续存活下去所必须的。”斯托曼在听众们的窃窃私语中坚定地表明道。

“It was a pretty ugly scene,” recalls Prime Time Freeware’s Rich Morin. “John’s done some pretty respectable things: Tcl, Tk, Sprite. He’s a real contributor.” Despite his sympathies for Stallman and Stallman’s position, Morin felt empathy for those troubled by Stallman’s discordant words.

“这是一种很尴尬的场面。” Prime TIme Freeware的李奇·莫林回忆说，“John做出了一些很让人值得尊敬的东西——Tcl，Tk和Sprite。他是一个真正的贡献者”。莫林很理解斯托曼以及他的立场，但他也同样也理解那些被斯托曼“无理”的行为扫了兴的人们。

Stallman will not apologize. “Criticizing proprietary software isn’t ugly – proprietary software is ugly. Ousterhout had indeed made real contributions in the past, but the point is that Scriptics was going to be nearly 100% a proprietary software company. In that conference, standing up for freedom meant disagreeing with nearly everyone. Speaking from the audience, I could only say a few sentences. The only way to raise the issue so it would not be immediately forgotten was to put it in strong terms.”

TODO.....

“If people rebuke me for ‘making a scene’ when I state a serious criticism of someone’s conduct, while calling Torvalds ‘cheeky’ for saying nastier things about trivial matters, that seems like a double standard to me.”

TODO.....

Stallman’s controversial criticism of Ousterhout momentarily alienated a potential sympathizer, Bruce Perens. In 1998, Eric Raymond proposed launching the Open Source Initiative, or OSI, an organization that would police the use of the term “open source” and provide a definition for companies interested in making their own programs. Raymond recruited Perens to draft the definition.

斯托曼在Perl大会上的爆发，影响了另一位潜在的追随者——布鲁斯·佩伦斯。

1998年，埃里克·雷蒙德打算发行开源倡议组织，OSI，一个监管“开源”这个词语使用并向感兴趣的公司提供标准的组织。雷蒙德请佩伦斯来企起划这个标准。

Perens would later resign from the OSI, expressing regret that the organization had set itself up in opposition to Stallman and the FSF. Still, looking back on the need for a free software definition outside the Free Software Foundation's auspices, Perens understands why other hackers might still feel the need for distance. “I really like and admire Richard,” says Perens. “I do think Richard would do his job better if Richard had more balance. That includes going away from free software for a couple of months.”

佩伦斯不久就从OSI的高管位子上辞职，他对这个组织与斯托曼和FSF对立的立场表示遗憾。同样的，回头去看，在FSF以外重新定义自由软件的需求，佩伦斯理解了为什么其它的黑客们仍然觉得需要与之保持距离。“我非常敬仰理查德”,佩伦斯说，“我觉得如果他能更为平衡的处理各方关系的话，他会把事情完成得更好。这包括花几个月时间远离自由软件。”

Stallman's energies would do little to counteract the public-relations momentum of open source proponents. In August of 1998, when chip-maker Intel purchased a stake in GNU/Linux vendor Red Hat, an accompanying *New York Times* article described the company as the product of a movement “known alternatively as free software and open source.” Six months later, a John Markoff article on Apple Computer was proclaiming the company's adoption of the “open source” Apache server in the article headline.

斯托曼的偏执不太会抵消与开源斗士动力之间的公共关系。1998年8月，芯片厂商Intel购买了GNU/Linux供应商Red Hat的股份，纽约时报在随后的文章中评价Red Hat为“自由软件和开源”运动的产物。 半年后，John Markoff在Apple Computer杂志上发表文章，标题中称该公司使用“开源”的Apache服务器。.....

Such momentum would coincide with the growing momentum of companies that actively embraced the “open source” term. By August of 1999, Red Hat, a company that now eagerly billed itself as “open source,” was selling shares on Nasdaq. In December, VA Linux – formerly VA Research – was floating its own IPO to historic effect. Opening at \$30 per share, the company's stock price exploded past the \$300 mark in initial trading only to settle back down to the \$239 level. Shareholders lucky enough to get in at the bottom and stay until

the end experienced a 698% increase in paper wealth, a Nasdaq record. Eric Raymond, as a board member, owned shares worth \$36 million. However, these high prices were temporary; they tumbled when the dot-com boom ended.

这样的势头与很多公司拥抱“开源”一词的动力非常一致。到了1999年8月，Red Hat这家乐意把自己标榜为“开源”的公司在纳斯达克上市。同年12月，VA Linux公司（曾经的VA Research）正式IPO。这家公司的股票从开盘的30美元一股一路彪升到300美元，并最后回稳在239美元左右。股东们很幸运的抄了底,并且享受到了698%的纸面收益，打破了斯达克的历史记录。.....

The open source proponents' message was simple: all you need, to sell the free software concept, is to make it business-friendly. They saw Stallman and the free software movement as fighting the market; they sought instead to leverage it. Instead of playing the role of high-school outcasts, they had played the game of celebrity, magnifying their power in the process.

.....

These methods won great success for open source, but not for the ideals of free software. What they had done to “spread the message” was to omit the most important part of it: the idea of freedom as an ethical issue. The effects of this omission are visible today: as of 2009, nearly all GNU/Linux distributions include proprietary programs, Torvalds' version of Linux contains proprietary firmware programs, and the company formerly called VA Linux bases its business on proprietary software. Over half of all the world's web servers run some version of Apache, and the usual version of Apache is free software, but many of those sites run a proprietary modified version distributed by IBM.

不管这些评论是否减轻了雷蒙德和其它开源拥护者是为金钱为目的嫌疑，但是他们向开源社区传达了一个最基本的信息：想要宣传自由软件的概念，只需要一张友善的面孔和合乎情理的信号。不用像斯托曼那样正面与商业市场斗争，雷蒙德、托瓦兹和其它黑客社区的新领导者们需要一种更宽松的方式：在某些领域忽略市场，通过别的方式去影响它。不能扮演被学校开除的那个角色，而应该成为社会的名流、在这个过程中不断扩大自己的影响力。

“On his worst days Richard believes that Linus Torvalds and I conspired to hijack his revolution,” Raymond says. “Richard's rejection of the term open source and his deliberate creation of an ideological fissure in my view comes

from an odd mix of idealism and territoriality. There are people out there who think it's all Richard's personal ego. I don't believe that. It's more that he so personally associates himself with the free software idea that he sees any threat to that as a threat to himself."

雷蒙德说："在理查德最不如意的日子里，他认为我和Linus托瓦兹搞阴谋绑架他的革命。理查德拒绝开源一词的使用和他故意对我的观点做出的意识形态上的分裂是由于他自己对理想主义和地域性奇怪混合的结果。从外人的眼光看来，可能会以为这是理查德的个人野心膨胀的结果。我不这样认为。我觉得这不仅仅是野心，这更是因为理查德把自己和自由软件的理念紧紧的绑在了一起，在他看来，任何对自由软件的威胁就是对他本人的威胁。"

Stallman responds, "Raymond misrepresents my views: I don't think Torvalds 'conspired' with anyone, since being sneaky is not his way. However, Raymond's nasty conduct is visible in those statements themselves. Rather than respond to my views (even as he claims they are) on their merits, he proposes psychological interpretations for them. He attributes the harshest interpretation to unnamed others, then 'defends' me by proposing a slightly less derogatory one. He has often 'defended' me this way."

.....

Ironically, the success of open source and open source advocates such as Raymond would not diminish Stallman's role as a leader – but it would lead many to misunderstand what he is a leader of. Since the free software movement lacks the corporate and media recognition of open source, most users of GNU/Linux do not hear that it exists, let alone what its views are. They have heard the ideas and values of open source, and they never imagine that Stallman might have different ones. Thus he receives messages thanking him for his advocacy of "open source," and explains in response that he has never been a supporter of that, using the occasion to inform the sender about free software.

让人觉得讽刺的是，开源和像雷蒙德这样的开源拥护者的胜利并不能撼动斯托曼作为领导者的角色。斯托曼的一举一动都有机会给他带来新的追随者。而雷蒙德的领地的主管则没有这样的能力。斯托曼时常手握武器，不是因为出于原则，而更多是出于习惯：比如，他一开始反对Linux内核，现在则不愿意成为一个为了软件问题理想冒险的政治形象。".....

Some writers recognize the term “free software” by using the term “FLOSS,” which stands for “Free/Libre and Open Source Software.” However, they often say there is a single “FLOSS” movement, which is like saying that the U.S. has a “Liberal/Conservative” movement, and the views they usually associate with this supposed single movement are the open source views they have heard.

TODO.....

Despite all these obstacles, the free software movement does make its ideas heard sometimes, and continues to grow in absolute terms. By sticking to its guns, and presenting its ideas in contrast to those of open source, it gains ground. “One of Stallman’s primary character traits is the fact he doesn’t budge,” says Ian Murdock. “He’ll wait up to a decade for people to come around to his point of view if that’s what it takes.”

除此之外，近期关于开源的争论也表明：当斯托曼手握武器时，通常是他找到了一种可以让他的事业有所进展的方式的时候。伊恩·默多克说：”斯托曼最重的一个性格特征就是他不会动摇自己的立场。如果需要，为了让别人接受他的观点，他可以等上十年。”

Murdock, for one, finds that unbudgeable nature both refreshing and valuable. Stallman may no longer be the solitary leader of the free software movement, but he is still the polestar of the free software community. “You always know that he’s going to be consistent in his views,” Murdock says. “Most people aren’t like that. Whether you agree with him or not, you really have to respect that.”

默多克所发现的斯托曼这个不动摇的特质既新鲜又宝贵。就算斯托曼不再是自由软件运动的唯一领袖，他仍然是自由软件社区的北极星。默多克说：“你可以确定他永远会坚持他自己的观点。大部分人都不是这样的。不管你是否同意他的观点，你都必须尊重他的观点。” 因为事实常常证明：他站得更高，看得更远。

A Brief Journey through Hacker Hell
开往黑客地狱的短暂旅途

Richard Stallman stares, unblinking, through the windshield of a rental car, waiting for the light to change as we make our way through downtown Kihei.

理查德·斯托曼双目凝视着车外，眼睛一眨不眨，我们正坐在这辆租来汽车里途经津汇城区，等待着信号灯变绿。

The two of us are headed to the nearby town of Pa'ia, where we are scheduled to meet up with some software programmers and their wives for dinner in about an hour or so.

我们正在前往附近一个名为芭雅的小镇，去会见一些软件开发者和他们的妻子，然后一同去参加一小时以后另一地点开场的晚宴。

It's about two hours after Stallman's speech at the Maui High Performance Center, and Kihei, a town that seemed so inviting before the speech, now seems profoundly uncooperative. Like most beach cities, Kihei is a one-dimensional exercise in suburban sprawl. Driving down its main drag, with its endless succession of burger stands, realty agencies, and bikini shops, it's hard not to feel like a steel-coated morsel passing through the alimentary canal of a giant commercial tapeworm. The feeling is exacerbated by the lack of side roads. With nowhere to go but forward, traffic moves in spring-like lurches. 200 yards ahead, a light turns green. By the time we are moving, the light is yellow again.

斯托曼刚刚在茂宜高性能计算中心完成一次演讲，晚宴的开始时间距离演讲结束只有两小时时间。演讲开始前，津汇对我们的到来显得如此的热情，而现在，则让人感觉到它处处在为难我们。跟很多海边城市一样，津汇只有一条主干道贯穿全城。行驶在津汇的主干道上，看着路边的汉堡店、房产中介和比基尼商店，让人感觉到像是一条巨大的绦虫正在吞下一片穿着钢盔铁甲的食物。除了一直向前开，没有别的选择，车流就像一条蜿蜒的溪流。200码开外，信号灯变绿了。当我们开始往前挪动车子时，灯又变黄了。

For Stallman, a lifetime resident of the east coast, the prospect of spending the better part of a sunny Hawaiian afternoon trapped in slow traffic is enough to trigger an embolism. [RMS: Since I was driving, I was also losing time to answer my email, and that's a real pain since I can barely keep up anyway.] Even worse is the knowledge that, with just a few quick right turns a quarter mile back, this whole situation easily could have been avoided. Unfortunately, we are at the mercy of the driver ahead of us, a programmer from the lab who

knows the way and who has decided to take us to Pa'ia via the scenic route instead of via the nearby Pilani Highway.

对于斯托曼这样长期在东部生活的人来说，把夏威夷午后明媚的阳光浪费在拥堵的马路上足以诱发他的脑血栓。[RMS：因为如果我在开车，我就无法回复电子邮件，这对于我来说实在是太痛苦了。] 更糟糕的是，其实在四分之一英里前，有一些可以右转的道路，如果车辆趁早转弯，就可以避免这可怕的交通拥堵。不幸的是，我们需要跟着前面车辆，那辆车的司机是一名实验室的程序员，他认识去往目的地的路，并且是他决定带我们从芭雅的观光道路通行，而不是绕行附近的彼拉尼高速公路。

"This is terrible," says Stallman between frustrated sighs. "Why didn't we take the other route?"

斯托曼叹着气失望的说："这实在是太可怕了，我们为什么没走另一条路呢？"

Again, the light a quarter mile ahead of us turns green. Again, we creep forward a few more car lengths. This process continues for another 10 minutes, until we finally reach a major crossroad promising access to the adjacent highway.

终于，我们前方1/4英里处的信号灯变绿了。但我们仍然只能前进很短的一段距。10分钟内这样的过程重复了好几次，直到我们最终慢慢的挪到了相临一条公路的十字路口后。

The driver ahead of us ignores it and continues through the intersection.

我们前面的车没有理会这个十字路口，继续向前行驶。

"Why isn't he turning?" moans Stallman, throwing up his hands in frustration. "Can you believe this?"

斯托曼沮丧的挥动着双手，抱怨道："他为什么不拐弯呢？你们不觉得奇怪吗？"

I decide not to answer either. I find the fact that I am sitting in a car with Stallman in the driver seat, in Maui no less, unbelievable enough. Until two hours ago, I didn't even know Stallman knew how to drive. Now, listening to Yo-Yo Ma's cello playing the mournful bass notes of "Appalachian Journey" on the car stereo and watching the sunset pass by on our left, I do my best to fade into the upholstery.

我决定不必要回答这两个问题。我知道我正与斯托曼在毛伊岛同乘一辆车，他在开车，这让人很难以至信。事实上，两小时前我还不知道斯托曼会不会开车。而现在，我们在车上欣赏着马友友的“阿巴拉契旅行”专辑中大提琴所发出的让人感到悲伤的低音音符，看着太阳渐渐从我们的左边落下，我尽可能的让自己沉浸在这样的氛围中。

When the next opportunity to turn finally comes up, Stallman hits his right turn signal in an attempt to cue the driver ahead of us. No such luck. Once again, we creep slowly through the intersection, coming to a stop a good 200 yards before the next light. By now, Stallman is livid.

当下一个转弯的机会出现的时候，斯托曼打开了右转向灯，试图提醒前车的司机。不过仍然没有好运气。我们又一次慢慢的驶过了十字路口，在距离下一个信号灯至少200码的地方停了下来。斯托曼终于忍不住大发雷霆。

“It’s like he’s deliberately ignoring us,” he says, gesturing and pantomiming like an air craft carrier landing-signals officer in a futile attempt to catch our guide’s eye. The guide appears unfazed, and for the next five minutes all we see is a small portion of his head in the rearview mirror.

“他简直就是故意无视我们的存在。”他愤愤地说。一面又像机场的信号官指挥飞机降落一样打着手势，尝试着吸引向导的注意力。向导像是完全没有注意到这一点，在接下来的5分钟里，我们只能在他的后视镜里看到他一小部分的脑袋。

I look out Stallman’s window. Nearby Kahoolawe and Lanai Islands provide an ideal frame for the setting sun. It’s a breathtaking view, the kind that makes moments like this a bit more bearable if you’re a Hawaiian native, I suppose. I try to direct Stallman’s attention to it, but Stallman, by now obsessed by the inattentiveness of the driver ahead of us, blows me off.

我从斯托曼的车窗向外看，附近的卡胡拉威岛和拉拿夷岛与落日一起构成了一副美丽的画面。这是一副美得让人窒息的画面，我想，如果你一个夏威夷本地人，一定会因为这样的美景而忘记了堵车的烦恼。我试图把斯托曼的注意力引向这里，但是他的注意力仍然一心集中在前面那个无视我们的司机身上，完全不搭理我。

When the driver passes through another green light, completely ignoring a “Pilani Highway Next Right,” I grit my teeth. I remember an early warning relayed to me by BSD programmer Keith Bostic. “Stallman does not suffer

fools gladly," Bostic warned me. ``If somebody says or does something stupid, he'll look them in the eye and say, 'That's stupid.'

"

司机开过另一个亮着绿灯的路口，完全无视边上"下一个路口向右转驶入彼拉尼高速"的标识。我咂了咂嘴。我记起以前一位BSD程序员基思·博斯蒂克警告过我："斯托曼无法容忍傻瓜，如果有人说了或者做了一些什么蠢事，他会看着他的眼睛说：'这样做太愚蠢了。'"

Looking at the oblivious driver ahead of us, I realize that it's the stupidity, not the inconvenience, that's killing Stallman right now.

看着前方心不在焉的司机，我觉得他做的就是所谓的蠢事，而不只是一些不太聪明的事，这些蠢事正在让斯托曼备受煎熬。

"It's as if he picked this route with absolutely no thought on how to get there efficiently," Stallman says.

"他好像完全没有想过应该如何有效的到达目的地，所以才选择了这么一条路。"斯托曼说。

The word "efficiently" hangs in the air like a bad odor. Few things irritate the hacker mind more than inefficiency. It was the inefficiency of checking the Xerox laser printer two or three times a day that triggered Stallman's initial inquiry into the printer source code. It was the inefficiency of rewriting software tools hijacked by commercial software vendors that led Stallman to battle Symbolics and to launch the GNU Project. If, as Jean Paul Sartre once opined, hell is other people, hacker hell is duplicating other people's stupid mistakes, and it's no exaggeration to say that Stallman's entire life has been an attempt to save mankind from these fiery depths.

"有效"这个词像是一股坏气味停留在了空中。很少有事情能比"低效"更刺激到一个黑客的神经了。当年，正是因为施乐的打印机一天要检修两三次所来带的低效才激发了斯托曼想要获取打印机源代码的想法。也正是因为要重头编写被商业公司所绑架的商用软件的低效，才引发了斯托曼与Symbolics之间的斗争，从而诞生了GNU工程。让-保罗·萨特曾经说，如果别人是地狱，那么黑客的地狱就是重复其它人愚蠢的错误，毫不夸张的说，斯托曼的一生，就是在尝试把人类从地狱的火焰中拯救出来。

This hell metaphor becomes all the more apparent as we take in the slowly passing scenery. With its multitude of shops, parking lots, and poorly timed street lights, Kihei seems less like a city and more like a poorly designed software program writ large. Instead of rerouting traffic and distributing vehicles through side streets and expressways, city planners have elected to run everything through a single main drag. From a hacker perspective, sitting in a car amidst all this mess is like listening to a CD rendition of nails on a chalkboard at full volume.

这种有关地狱的隐喻在我们缓慢通过这美景时变得更为明显。四处都是商场、停车场和缺乏设计的信号灯，看上去不像是个城市，倒更像是一个设计得很糟糕的软件。城市的规划者把城市设计成所有的车辆都要通过主干道，而不是把车流分散到支路和高速公路上。从黑客的角度来说，坐在车里并被围困在这一团糟的交通中，就像是用最大的音量听一张录有在木板上钉钉子声音的CD。

“Imperfect systems infuriate hackers,” observes Steven Levy, another warning I should have listened to before climbing into the car with Stallman. “This is one reason why hackers generally hate driving cars – the system of randomly programmed red lights and oddly laid out one-way streets causes delays which are so goddamn *unnecessary* [Levy’s emphasis] that the impulse is to rearrange signs, open up traffic-light control boxes . . . redesign the entire system.”

“不完美的系统会激怒黑客。”史蒂芬·李维说过这样的话，这是我决定与斯托曼同坐一辆车前应该听取的另一个忠告，“这是黑客们通常不喜欢开车的原因之一：这是一个充满不确定性的程序，交通信号灯总是随机的变化，还有横七竖八的单行道，导致交通经常堵塞。这实在是太*不必要*了（李维强调说），只要让黑客们重新安排一下信号灯，打开交通灯控制盒，重新设计整个系统。”

More frustrating, however, is the duplicity of our trusted guide. Instead of searching out a clever shortcut – as any true hacker would do on instinct – the driver ahead of us has instead chosen to play along with the city planners’ game. Like Virgil in Dante’s *Inferno*, our guide is determined to give us the full guided tour of this hacker hell whether we want it or not.

更让人感到沮丧的事，是我们的向导的愚笨，他没有像一个黑客那样本能的做出最聪明的选择，选择一条更为聪明的捷径，而是坚持陪着城市设计者玩他们愚蠢的游戏。像但丁《*神曲*》中的维吉尔一样，不管我们是否希望，他都打定主意要让我们完整的体验这个黑客的地狱。

Before I can make this observation to Stallman, the driver finally hits his right turn signal. Stallman's hunched shoulders relax slightly, and for a moment the air of tension within the car dissipates. The tension comes back, however, as the driver in front of us slows down. "Construction Ahead" signs line both sides of the street, and even though the Pilani Highway lies less than a quarter mile off in the distance, the two-lane road between us and the highway is blocked by a dormant bulldozer and two large mounds of dirt.

在我还没有来得及告诉斯托曼我的发现以前，前方的司机终于在路口右转了。斯托曼耸起的肩膀总于放松了一些，车里紧张的空气稍稍消散了一些。然而，当前面的司机把车慢慢停下来时，紧张的气氛又回来了。道路两侧的放着"前方施工"的标识，虽然彼拉尼高速公路就在前方不到四分之一英里的距离，我们的车与高速公路间的一条两车道的公路被一辆停着的推土车和两大堆土方完全堵住了。

It takes Stallman a few seconds to register what's going on as our guide begins executing a clumsy five-point U-turn in front of us. When he catches a glimpse of the bulldozer and the "No Through Access" signs just beyond, Stallman finally boils over.

我们的向导在我们的面前忽然把车笨拙的调了一个头，斯托曼一时都没有反应过来。当他看到了面前的推土机和"禁止通行"标识后，他终于再一次爆发了。

"Why, why, why?" he whines, throwing his head back. "You should have known the road was blocked. You should have known this way wouldn't work. You did this deliberately." [RMS: I meant that he chose the slow road deliberately. As explained below, I think these quotes are not exact.]

"为什么，为什么，为什么？"他抱怨道，把头一仰，"你早该知道前面的路被封住了，你早该知道不能走这条路。你是故意这么做的。" [RMS：我原来的意思是说他故意选择了那条拥堵的路，跟下面所解释的一样，我觉得这里所引用的我说的话并不准确。]

The driver finishes the turn and passes us on the way back toward the main drag. As he does so, he shakes his head and gives us an apologetic shrug. Coupled with a toothy grin, the driver's gesture reveals a touch of mainlander frustration but is tempered with a protective dose of islander fatalism. Coming through the sealed windows of our rental car, it spells out a succinct message: "Hey, it's Maui; what are you gonna do?"

向导的车已经调好了头，从我们身边开过，开回那条拥塞的主干道。在他经过我们身边时，他摇了摇了头，给了我们一个抱歉的耸肩。他露出牙齿，司机的姿势显示出了一个外地人的失望，但是这已经被岛国人的宿命所中和了。从我们租来车关闭的窗户望出去，我们似乎看到了这么一句话：“嘿，这就是毛依岛，你想做什么？”

Stallman can take it no longer.

斯托曼再也无法忍受了。

“Don’t you fucking smile!” he shouts, fogging up the glass as he does so. “It’s your fucking fault. This all could have been so much easier if we had just done it my way.” [RMS: These quotes appear to be inaccurate, because I don’t use “fucking” as an adverb. This was not an interview, so Williams would not have had a tape recorder running. I’m sure things happened overall as described, but these quotations probably reflect his understanding rather than my words.]

“你可不可以不要再笑了！”他咆哮道，雾气蒙上了他的眼镜片，“这全他妈是你的错。如果听我的走另一条路就不会这样了。”[RMS：这些话记录的似乎并不准确，因为我从来不把“他妈的”当成副词来使用。由于我说这些话里并不是在接受采访，所以威廉姆斯并没有录下我说的话。我很确定这里的文字确实还愿了当时的情景，但是这些话也许是反应了他对我所说的话的个人理解，并不是我的原话。]

Stallman accents the words “my way” by gripping the steering wheel and pulling himself towards it twice. The image of Stallman’s lurching frame is like that of a child throwing a temper tantrum in a car seat, an image further underlined by the tone of Stallman’s voice. Halfway between anger and anguish, Stallman seems to be on the verge of tears.

斯托曼加重了“听我的”一词，紧紧的抓住方向盘并两次把它拉向自己。斯托曼的样子就像是一个在汽车座位里发脾气的小孩子，他的声音进一步加强了这样的形象。斯托曼又生气又郁闷，眼泪几乎就要掉下来。

Fortunately, the tears do not arrive. Like a summer cloudburst, the tantrum ends almost as soon as it begins. After a few whiny gasps, Stallman shifts the car into reverse and begins executing his own U-turn. By the time we are back on the main drag, his face is as impassive as it was when we left the hotel 30 minutes earlier.

幸运的是，眼泪最终没有到来。就像夏天的暴雨，斯托曼的愤怒转瞬即逝。他轻轻叹了口气，把车挂到倒档，并开始调头。当我们回到城市主干道上时，他的表情让人过目不忘，就跟我们提前半小时离开酒店时的表情一样。

It takes less than five minutes to reach the next cross-street. This one offers easy highway access, and within seconds, we are soon speeding off toward Pa'ia at a relaxing rate of speed. The sun that once loomed bright and yellow over Stallman's left shoulder is now burning a cool orange-red in our rearview mirror. It lends its color to the gauntlet wili wili trees flying past us on both sides of the highway.

我们花了不到五分钟时间，到达了下一个十字路口。这里可以容易的驶入高速公路，很快，我们就加大马力向着芭雅驶去。刚才斯托曼左肩上若隐若现的黄色太阳现在变成了橙红色出现在我们的后视镜中。金色的阳光撒向高速公路两边的树木。

For the next 20 minutes, the only sound in our vehicle, aside from the ambient hum of the car's engine and tires, is the sound of a cello and a violin trio playing the mournful strains of an Appalachian folk tune.

接下来的20分钟，只剩下了汽车的声音，包含着引擎和轮胎的轰鸣声。像是大提琴与小提琴的三重奏，演绎着阿巴拉契的民歌旋律。

Appendix A – Hack, Hackers, and Hacking 附录 A – 黑客的三层含义 {#Appendix A}

To understand the full meaning of the word "hacker," it helps to examine the word's etymology over the years.

要理解"黑客"一词内在的含义，需要追朔这个词的词源和这么多年来的演化过程。

The New Hacker Dictionary, an online compendium of software-programmer jargon, officially lists nine different connotations of the word "hack" and a similar number for "hacker." Then again, the same publication also includes an

accompanying essay that quotes Phil Agre, an MIT hacker who warns readers not to be fooled by the word's perceived flexibility. "Hack has only one meaning," argues Agre. "An extremely subtle and profound one which defies articulation." Richard Stallman tries to articulate it with the phrase, "Playful cleverness."

在*新黑客词典*这本在线的软件程序员专业术语词典中，对"Hack"一词列出了9种不同的含义，"Hacker"一词也有相对应的多种含义。同时，上面也引用了一段麻省理工学院的黑客费尔·阿格雷所撰写的短文，这篇文篇提醒读者不要被这个词给人的第一印象所迷惑。"Hack这个词只有一种意思，"阿格雷争辩道，"这个意思非常微妙和有深度，难以用语言来描述。"理查德·斯托曼把这个词表述为一个短语："顽皮的小聪明"。

Regardless of the width or narrowness of the definition, most modern hackers trace the word back to MIT, where the term bubbled up as popular item of student jargon in the early 1950s. In 1990 the MIT Museum put together a journal documenting the hacking phenomenon. According to the journal, students who attended the institute during the fifties used the word "hack" the way a modern student might use the word "goof." Hanging a jalopy out a dormitory window was a "hack," but anything harsh or malicious – e.g., egging a rival dorm's windows or defacing a campus statue – fell outside the bounds. Implicit within the definition of "hack" was a spirit of harmless, creative fun.

不管是讨论这个词的狭义定义还是广义定义，大部分现代的黑客都会把这个词追朔到麻省理工学院，从20世纪50年代起，这个词就成为学生们互相交流时的使用的一句黑话。1990年，麻省理工学院博物馆还发表了一篇有关黑客现象的论文。根据这篇论文的描述，50年代时，这个学校的学生们使用"Hack"这个词的方式与现在的学生使用"Goof（混混）"一词的含义类似。在宿舍的窗户上挂上一个破飞机的行为就可以算是Hack。但是，任何残忍的或恶意的行为就不能算是Hack，比如怂恿另人去打破宿舍窗户或损坏学院的雕像。"Hack"一词内在含义是在不损害他人的前提下寻找创新点和乐趣的精神。

This spirit would inspire the word's gerund form: "hacking." A 1950s student who spent the better part of the afternoon talking on the phone or dismantling a radio might describe the activity as "hacking." Again, a modern speaker would substitute the verb form of "goof" – "goofing" or "goofing off" – to describe the same activity.

这种精神赋于了Hack这个词的动名词形式"Hacking"更多活力。在20世纪50年

代，如果一个学生花上大半天的时间打电话或拆装收音机，这样行为就可以被称作是“Hacking”。而在现代，人们通常会用“Goof”一词的动词形式“Goofing”或“Goofing Off”来描述这类行为。

As the 1950s progressed, the word “hack” acquired a sharper, more rebellious edge. The MIT of the 1950s was overly competitive, and hacking emerged as both a reaction to and extension of that competitive culture. Goofs and pranks suddenly became a way to blow off steam, thumb one’s nose at campus administration, and indulge creative thinking and behavior stifled by the Institute’s rigorous undergraduate curriculum. With its myriad hallways and underground steam tunnels, the Institute offered plenty of exploration opportunities for the student undaunted by locked doors and “No Trespassing” signs. Students began to refer to their off-limits explorations as “tunnel hacking.” Above ground, the campus phone system offered similar opportunities. Through casual experimentation and due diligence, students learned how to perform humorous tricks. Drawing inspiration from the more traditional pursuit of tunnel hacking, students quickly dubbed this new activity “phone hacking.”

经过整个50年代的演化，“Hack”这个词的含义变得更加尖锐和反叛。50年代的麻省理工学院校园里充满了竞争，而且黑客的出现，正是这种竞争文化的产物和延续。Goof和Pranks（喜欢恶作剧的人）突然成为学生们发泄多余精力的方式，他们甚至会跑去学校的教务处去做鬼脸，沉溺于各种稀奇古怪的想法和被繁重的研究生课程所遏制的行为。学院众多走廊和地下蒸气管道提供了无畏于紧闭的大们和“禁止进入”标识的学生们足够多探索的机会。学生们开始把这种不被规章制度允许的探索称为“地道黑客”。在地面上，校园的电话系统也提供了类似的探索机会。通过业余的实验和一些专业的研究，学生找到一些恶作剧的办法。学生们从相对古老的“地道黑客”活动中获得了灵感，很快就参与到了这种新型的“电话黑客”活动中。

The combined emphasis on creative play and restriction-free exploration would serve as the basis for the future mutations of the hacking term. The first self-described computer hackers of the 1960s MIT campus originated from a late 1950s student group called the Tech Model Railroad Club. A tight clique within the club was the Signals and Power (S&P) Committee – the group behind the railroad club’s electrical circuitry system. The system was a sophisticated assortment of relays and switches similar to the kind that controlled the local campus phone system. To control it, a member of the

group simply dialed in commands via a connected phone and watched the trains do his bidding.

强调创新游戏和无限制的探索活动，成为后来黑客活动的文化基础。20世纪60年代出现了一些最早自称为是计算机黑客的人，他们来自麻省理工学院里一个起源于50年代末期、名为铁路科技模型俱乐部（Tech Model Railroad Clus）的学生组织。这个俱乐部中有一个紧密的小团队叫做“信号与电源（S&P）委员会”，是这个俱乐部的电子电路系统的主要负责团队。这个系统是一个由继电器和开关组成的复杂系统，有点像当时学校电话的控制系统。俱乐部的成员只须通过电话拨打一些特定的号码就可以控制火车的运行。

The nascent electrical engineers responsible for building and maintaining this system saw their activity as similar in spirit to phone hacking. Adopting the hacking term, they began refining it even further. From the S&P hacker point of view, using one less relay to operate a particular stretch of track meant having one more relay for future play. Hacking subtly shifted from a synonym for idle play to a synonym for idle play that improved the overall performance or efficiency of the club’s railroad system at the same time. Soon S&P committee members proudly referred to the entire activity of improving and reshaping the track’s underlying circuitry as “hacking” and to the people who did it as “hackers.”

负责建造和维护这套系统的电子工程爱好者们把这样的活动认为是与电话黑客活动一脉相成。他们不但接受了黑客这个概念，而且还把它变得更为完善。从S&P黑客的观点来看，如果某一段火车铁轨的控制电路中可以少用一个继电器，就意味着可以节省一个继电器供以后使用。黑客这个词的含义渐渐从“闲玩”的同义词演化成了在闲玩的同时去提高俱乐部的铁路系统的整体性能和效率。不久以后，S&P委员会的成员就自豪的宣布，把改进和重构铁路的底层电路的活动称为“黑客活动”，而参与这样活动的人就是“黑客”。

Given their affinity for sophisticated electronics – not to mention the traditional MIT-student disregard for closed doors and “No Trespassing” signs – it didn’t take long before the hackers caught wind of a new machine on campus. Dubbed the TX-0, the machine was one of the first commercially marketed computers. By the end of the 1950s, the entire S&P clique had migrated en masse over to the TX-0 control room, bringing the spirit of creative play with them. The wide-open realm of computer programming would encourage yet another mutation in etymology. “To hack” no longer meant soldering unusual

looking circuits, but cobbling together software programs with little regard to “official” methods or software-writing procedures. It also meant improving the efficiency and speed of already-existing programs that tended to hog up machine resources. True to the word’s roots, it also meant writing programs that served no other purpose than to amuse or entertain.

复杂的电子电路对黑客们有着天然的亲和力，抛开那些无视紧锁的大门和“禁止进入”标识的传统的麻省理工学院的学生，新一代的黑客们很快就在校园中找到了一种新的玩具。这种被称为TX-0的机器，是最早进入商用市场的计算机之一。到了20世纪50年代末，整个S&P的团队都已经把兴趣转移到TX-0的控制室里，用他们的创新精神来折腾这个机器。从那时起，计算机程序设计领域开始进入“黑客”一词的词源。“黑客行为”不再是指焊接一些看起来很奇怪的电路板，而是更多的是与软件程序开发联系起来，还包括少许“正式”的软件开发方法或过程。它同样意味着提高效率和改进现有程序让它们运行的更快，因为只有这样才能节省更多的系统资源。从词源上来讲，这个词也指仅仅为了好玩或娱乐而编写一些没有实质性用途的程序。

A classic example of this expanded hacking definition is the game Spacewar, the first computer-based video game. Developed by MIT hackers in the early 1960s, Spacewar had all the traditional hacking definitions: it was goofy and random, serving little useful purpose other than providing a nightly distraction for the dozen or so hackers who delighted in playing it. From a software perspective, however, it was a monumental testament to innovation of programming skill. It was also completely free. Because hackers had built it for fun, they saw no reason to guard their creation, sharing it extensively with other programmers. By the end of the 1960s, Spacewar had become a diversion for programmers around the world, if they had the (then rather rare) graphical displays.

“空间大战（Spacewar）”游戏是体现这种黑客定义的经典例子，这是世界上第一个在电脑上运行的交互式的视频游戏。早在20世纪60年代，麻省理工学院的黑客们就开发出了这个游戏，“空间大战”游戏诠释了所有传统黑客的定义：它既无聊又随机，除了可以让一群黑客们通宵达旦的娱乐外，没有什么实质性的用途。从软件的观点来说，它倒也算是程序开发技能上的一次伟大的创新。同时，它也是一个完全自由的软件。由于黑客们只是为了好玩而设计了它，所以他们觉得没有什么理由要去刻意保护这样的创作成果。因此，这个游戏可以在程序员之间广为分享。到了20世纪60年代末，“空间大战”已经成为全世界大型机程序员们最爱的一种消遣方式。

This notion of collective innovation and communal software ownership distanced the act of computer hacking in the 1960s from the tunnel hacking and phone hacking of the 1950s. The latter pursuits tended to be solo or small-group activities. Tunnel and phone hackers relied heavily on campus lore, but the off-limits nature of their activity discouraged the open circulation of new discoveries. Computer hackers, on the other hand, did their work amid a scientific field biased toward collaboration and the rewarding of innovation. Hackers and “official” computer scientists weren’t always the best of allies, but in the rapid evolution of the field, the two species of computer programmer evolved a cooperative – some might say symbiotic – relationship.

60年代的计算机黑客与50年代的地道黑客和电话黑客的最大区别，就在于这种合作创新与公有的软件所有权。传统的黑客更崇尚个人或小团体的活动。地道黑客和电话黑客活动与校园里的情况紧密相关，他们这种不被规章制度所允许的活动天然的不鼓励通过公开的交流来分享新的发现。计算机黑客则不同，他们在科学领域中工作，这是一个完全基于协作和鼓励创新的氛围。黑客们与“官方”的计算机科学家并不一定是最好的搭档，但是由于计算机技术的高速发展，这两种计算机程序员之间很快就形成了一种或多或少的合作共生关系。

Hackers had little respect for bureaucrats’ rules. They regarded computer security systems that obstructed access to the machine as just another bug, to be worked around or fixed if possible. Thus, breaking security (but not for malicious purposes) was a recognized aspect of hacking in 1970, useful for practical jokes (the victim might say, “I think someone’s hacking me”) as well as for gaining access to the computer. But it was not central to the idea of hacking. Where there was a security obstacle, hackers were proud to display their wits in surmounting it; however, given the choice, as at the MIT AI Lab, they chose to have no obstacle and do other kinds of hacking. Where there is no security, nobody needs to break it.

黑客们大多都很讨厌官僚的制度。他们把阻碍自由访问计算机的安全系统看作是系统中的一个Bug，只要有机会，就要想办法绕过它或把它修复。因此，在1970年时，绕过安全系统（但不是出于恶意的目的）被看成是一种典型的黑客行为，这样做不但获取了计算机的访问权限，还增加了饭后的谈资（受害者可能会说：“我觉得我被黑了。”）。但这并不是黑客活动的核心理念。黑客们在遇到一些安全系统阻碍时，会发挥自己的才智绕过它们。但是，如果可以选择的话，麻省理工学院人工智能实验室的黑客们更倾向于从一开始就不要构建这样阻碍，从而可以把自己的才智用到更有意义的地方。如果没有所谓的安全系

统，也就没有人想着要去破坏它。

It is a testament to the original computer hackers' prodigious skill that later programmers, including Richard M. Stallman, aspired to wear the same hacker mantle. By the mid to late 1970s, the term "hacker" had acquired elite connotations. In a general sense, a computer hacker was any person who wrote software code for the sake of writing software code. In the particular sense, however, it was a testament to programming skill. Like the term "artist," the meaning carried tribal overtones. To describe a fellow programmer as a hacker was a sign of respect. To describe oneself as a hacker was a sign of immense personal confidence. Either way, the original looseness of the computer-hacker appellation diminished as computers became more common.

这是早期计算机黑客们惊人的技能的一种证明，很多后来的程序员，包括理查德·马修·斯托曼，都渴望能够做到这一点。到了70年代中后期，“黑客”这个词又有了更精确的内涵。在一般人看来，计算机黑客是指那些可以为了写代码而写代码的人。他们觉得这是对于编程技能的一种证明。“黑客”一词跟“艺术家”这个词很像，它的涵义超越了它字面上的意思。把一名程序员称为是黑客，是对他的一种尊称，而把自己称为是黑客则表明了极强的自信。不管是哪一种情况，随着计算机的普及，“计算机黑客”这种早年很少见的称谓也变得越来越常见。

As the definition tightened, "computer" hacking acquired additional semantic overtones. The hackers at the MIT AI Lab shared many other characteristics, including love of Chinese food, disgust for tobacco smoke, and avoidance of alcohol, tobacco and other addictive drugs. These characteristics became part of some people's understanding of what it meant to be a hacker, and the community exerted an influence on newcomers even though it did not demand conformity. However, these cultural associations disappeared with the AI Lab hacker community. Today, most hackers resemble the surrounding society on these points.

对“黑客”的定义越来越精确，“计算机”黑客则还有一层超越了字面的涵义。麻省理工学院人工智能实验室的黑客们有一些共同的特质，包括，喜欢吃中国菜、讨厌抽烟、不沾烟、酒和其它一些可能会上瘾的药物。这些特质也渐渐成为一些人心目中对黑客群体的印象，这种印象对于新加入这个群体的人也产生了潜移默化的影响。然而，这些黑客文化在人工智能实验室黑客社区中渐渐的消失了。如今，大部分黑客在这些点上表现通常都与他周围的环境保持一致。

As the hackers at elite institutions such as MIT, Stanford, and Carnegie Mellon

conversed about hacks they admired, they also considered the ethics of their activity, and began to speak openly of a “hacker ethic”: the yet-unwritten rules that governed a hacker’s day-to-day behavior. In the 1984 book *Hackers*, author Steven Levy, after much research and consultation, codified the hacker ethic as five core hacker tenets.

麻省理工学院、斯坦福和卡耐基梅隆大学这样的精英大学中的黑客谈论他们所敬仰的黑客特质时，会谈到有关黑客活动的伦理，并开始公开宣传“黑客伦理”：这些不成文的规定影响着黑客们的日常行为。在1984年出版的《*黑客*》一书中，作者史蒂文·利维在进行了很多研究和咨询以后，把黑客伦理定义为五种核心的黑客信条。

In the 1980s, computer use expanded greatly, and so did security breaking. Mostly it was done by insiders with criminal intent, who were generally not hackers at all. However, occasionally the police and administrators, who defined disobedience as evil, traced a computer “intrusion” back to a hacker whose idea of ethics was “Don’t hurt people.” Journalists published articles in which “hacking” meant breaking security, and usually endorsed the administrators’ view of the matter. Although books like *Hackers* did much to document the original spirit of exploration that gave rise to the hacking culture, for most newspaper reporters and readers the term “computer hacker” became a synonym for “electronic burglar.”

20世纪80年代，计算机开始普及，破坏安全系统的行为也变多了。大部分破坏安全系统的行为都具有某些犯罪动机，实施这些破坏的内鬼其实根本就不能被称作是黑客。然而，警察和计算机管理员都会把破坏规则的行为看成是一种恶意的行为，他们把这种“入侵”计算机的行为追溯到了那些持有“不能伤害别人”理念的黑客身上。一些杂志的报道中，常常会从计算机管理员的视角出发，把“黑客行为”当成是破坏安全系统的同义词。尽管像《*黑客*》这样书用很多的篇幅去介绍黑客精神中的探索本质，并且弘扬黑客文化，但是对于大部分报纸记者和读者仍然把“计算机黑客”当成是“电子盗贼”的同义词。

By the late 1980s, many U.S. teenagers had access to computers. Some were alienated from society; inspired by journalists’ distorted picture of “hacking,” they expressed their resentment by breaking computer security much as other alienated teens might have done it by breaking windows. They began to call themselves “hackers,” but they never learned the MIT hackers’ principle against malicious behavior. As younger programmers began employing their

computer skills to harmful ends – creating and disseminating computer viruses, breaking into computer systems for mischief, deliberately causing computers to crash – the term “hacker” acquired a punk, nihilistic edge which attracted more people with similar attitudes.

到了80年代末，很多美国的青少年都开始有机会接触计算机。他们中的一部分有些叛逆的性格，受到杂志上歪曲了的“黑客”形象的误导，开始通过破坏计算机安全系统来发泄他们的对社会的不满，就像一些人通过砸玻璃来发泄一样。他们自称为“黑客”，但是他们从来没了解过麻省理工学院的黑客们反对恶意行为的信条。一些年轻的程序员开始在一些错误的方向利用他们的计算机才能：制作和传播计算机病毒、恶作剧入侵计算机系统、故意让计算机宕机。“黑客”这个词开始渐渐成为一个反叛、无政府主义的象征，这样的形象吸引了更多不了解黑客本质的人开始走向错误的方向。

Hackers have railed against this perceived misusage of their self-designator for nearly two decades. Stallman, not one to take things lying down, coined the term “cracking” for “security breaking” so that people could more easily avoid calling it “hacking.” But the distinction between hacking and cracking is often misunderstood. These two descriptive terms are not meant to be exclusive. It’s not that “Hacking is here, and cracking is there, and never the twain shall meet.” Hacking and cracking are different attributes of activities, just as “young” and “tall” are different attributes of persons.

黑客们为了改变这种错误的认识已经自发的进行了近二十年的斗争。斯托曼为了解决这个问题，创造了一个新词来表达“破坏计算机安全系统的人”的意思：“骇客”，避免人们错误的使用“黑客”一词。而且，黑客和骇客这两个词的还是常常引起人们的误解。这两个词表达的含义还是有一些重叠的地方，并不是“黑客在这里，骇客在那里，我们水火不容”。黑客行为和骇客行为只是同一种行为的不同层面上的属性，就像“年轻”和“高”是一个人的两个不同层面的属性。

Most hacking does not involve security, so it is not cracking. Most cracking is done for profit or malice and not in a playful spirit, so it is not hacking. Once in a while a single act may qualify as cracking and as hacking, but that is not the usual case. The hacker spirit includes irreverence for rules, but most hacks do not break rules. Cracking is by definition disobedience, but it is not necessarily malicious or harmful. The computer security field distinguishes between “black hat” and “white hat” crackers – i.e., crackers who turn toward destructive, malicious ends versus those who probe security in order to fix it.

其实，大部的黑客活动都与安全无关，所以不能称之为骇客活动。大部分骇客活动都是为了牟取利益或出于恶意，不会是出于“玩”的心态，所以不能称之为黑客活动。在少数情况下，可能会有个别的行为可以同时看成是黑客行为与骇客行为，但大部分情况下都不会是这样。黑客精神中包含对规则的不敬，但大部分黑客并不会去破坏规则。骇客活动从定义上来说就是不守规则，但是也并不一定就是恶意或有害的。在计算机安全领域中还区分“黑帽”骇客和“白帽”骇客，如果一个骇客的目的是搞破坏、恶意的，他就是一个“黑帽”；那些在安全系统中寻找漏洞是为了修复它的骇客，被称作为”白帽”。

The hacker’s central principle not to be malicious remains the primary cultural link between the notion of hacking in the early 21st century and hacking in the 1950s. It is important to note that, as the idea of computer hacking has evolved over the last four decades, the original notion of hacking – i.e., performing pranks or exploring underground tunnels – remains intact. In the fall of 2000, the MIT Museum paid tribute to the Institute’s age-old hacking tradition with a dedicated exhibit, the Hall of Hacks. The exhibit includes a number of photographs dating back to the 1920s, including one involving a mock police cruiser. In 1993, students paid homage to the original MIT notion of hacking by placing the same police cruiser, lights flashing, atop the Institute’s main dome. The cruiser’s vanity license plate read IHTFP, a popular MIT acronym with many meanings. The most noteworthy version, itself dating back to the pressure-filled world of MIT student life in the 1950s, is “I hate this fucking place.” In 1990, however, the Museum used the acronym as a basis for a journal on the history of hacks. Titled *The Journal of the Institute for Hacks, Tomfoolery, and Pranks*, it offers an adept summary of the hacking.

这种忌讳恶意行为的核心原则成为联系21世纪初的黑客与20世纪50年代的黑客的文化纽带。值得注意的是，在过去的40年中，计算机黑客概念的演化了很多。最初这个词的意思是指有点恶作剧的行为，比如发现一些隐蔽的地道，这层含义现在依然存在。2000年秋天，麻省理工学院博物馆举行了一个专门向古老的黑客传统致敬的展览，名为黑客殿堂。这个展览上展出了很多20世纪20年代的老照片，其中有一张是有人愚弄警察的巡逻车的照片。1993年，有学生向这种行为致敬，把一模一样的巡逻车、警灯放到了学院主楼的圆顶上。巡逻车的仿制车牌上写着IHTFP，这是一个在麻省理工学院很流行的缩写词，包含着很多的含义。其中最引人注意的意思是始创于20世纪50年代麻省理工学院压力最大的时候的流行语，“我憎恨这个鬼地方（I hate this fucing place）”。然而，在1990年，博物馆基于这个缩写词创作了一本期刊：《*黑客、蠢举与恶作剧学报（The Journal of the Institute for Hacks, Tomfoolery, and Pranks）*》，这本

期刊的主要内容有是关黑客的历史，它精妙描述了黑客活动的本质。

“In the culture of hacking, an elegant, simple creation is as highly valued as it is in pure science,” writes *Boston Globe* reporter Randolph Ryan in a 1993 article attached to the police car exhibit. “A Hack differs from the ordinary college prank in that the event usually requires careful planning, engineering and finesse, and has an underlying wit and inventiveness,” Ryan writes. “The unwritten rule holds that a hack should be good-natured, non-destructive and safe. In fact, hackers sometimes assist in dismantling their own handiwork.”

“在黑客的文化中，优雅、简洁的创新是被高度评价的，就像在纯科学的领域，”《*波士顿环球报*》的记者伦道夫·赖安在1993年的一篇警车展览相关的文章中写道，“黑客与一般校园中的那些喜欢恶作剧的人在一些需要细致计划、工程化和使用手段的事件中的表现不太一样，他们本质上更为聪明和具有创造力。”Ryan写道，“黑客的本质应该是好的，不能去搞破坏，应该是安全的，这是一条不成文的规则。事实上，黑客们有时也参与破坏他们自己的作品”。

The urge to confine the culture of computer hacking within the same ethical boundaries is well-meaning but impossible. Although most software hacks aspire to the same spirit of elegance and simplicity, the software medium offers less chance for reversibility. Dismantling a police cruiser is easy compared with dismantling an idea, especially an idea whose time has come.

主张把计算机黑客的文化限定在特定的伦理边界中是一个不错的想法，但却是不太可能的。尽管大部分软件黑客们渴望优雅和简单的精神气质，但是软件媒体很少提供回头的机会。摆脱警察的巡逻车要比破碎一个想法更为容易，尤其是一个出现的正是时候的想法。

Once a vague item of obscure student jargon, the word “hacker” has become a linguistic billiard ball, subject to political spin and ethical nuances. Perhaps this is why so many hackers and journalists enjoy using it. We cannot predict how people will use the word in the future. We can, however, decide how we will use it ourselves. Using the term “cracking” rather than “hacking,” when you mean “security breaking,” shows respect for Stallman and all the hackers mentioned in this book, and helps preserve something which all computer users have benefited from: the hacker spirit.

“黑客”这个词曾经只是学生们一个含糊其辞的黑话，现在却成为了一个语言学上的一个台球，服从于政治的束缚和伦理上的差异。这也许就是为什么很多黑客

和记者们喜欢使用这个词语的原因。没有人可以预料这个词以后会如何被使用，但是，我们可以决定我们自己如何去使用它。当你在谈论“破坏安全系统”时，请使用“骇客”这个词，而不要使用“黑客”，这样不但可以表现出你对斯托曼和本书中提到的其他黑客们的尊敬，也可以让黑客精神可以不断传承下去，造福更多计算机用户。