Free as in Freedom (2.0): Richard Stallman and the Free Software Revolution

Sam Williams
Second edition revisions by Richard M. Stallman

This is *Free as in Freedom 2.0: Richard Stallman and the Free Software Revolution*, a revision of *Free as in Freedom: Richard Stallman's Crusade for Free Software*.

Published by the Free Software Foundation
51 Franklin St., Fifth Floor
Boston, MA 02110-1335
USA
ISBN: 9780983159216

The cover photograph of Richard Stallman is by Peter Hinely. The PDP-10 photograph in Chapter 7 is by Rodney Brooks. The photograph of St. IGNUcius in Chapter 8 is by Stian Eikeland.

Free as in Freedom (2.0): Richard Stallman and the Free Software Revolution

Sam Williams
Second edition revisions by Richard M. Stallman

This is *Free as in Freedom 2.0: Richard Stallman and the Free Software Revolution*, a revision of *Fr as in Freedom: Richard Stallman's Crusade for Free Software*.

Published by the Free Software Foundation
51 Franklin St., Fifth Floor
Boston, MA 02110-1335
USA
ISBN: 9780983159216

The cover photograph of Richard Stallman is by Peter Hinely. The PDP-10 photograph in Chapter 7 by Rodney Brooks. The photograph of St. IGNUcius in Chapter 8 is by Stian Eikeland.

Contents

Foreword
by Richard M. Stallman

I have aimed to make this edition combine the advantages of my knowledge and Williams' interview and outside viewpoint. The reader can judge to what extent I have achieved this.

I read the published text of the English edition for the first time in 2009 when I was asked assist in making a French translation of *Free as in Freedom*. It called for more than small changes.

Many facts needed correction, but deeper changes were also needed. Williams, a non programmer, blurred fundamental technical and legal distinctions, such as that between modifying existing program's code, on the one hand, and implementing some of its ideas in a new program, the other. Thus, the first edition said that both Gosmacs and GNU Emacs were developed modifying the original PDP-10 Emacs, which in fact neither one was. Likewise, it mistaken described Linux as a "version of Minix." SCO later made the same false claim in its infamous lawsu against IBM, and both Torvalds and Tanenbaum rebutted it.

The first edition overdramatized many events by projecting spurious emotions into them. F instance, it said that I "all but shunned" Linux in 1992, and then made a "a dramatic about-face" deciding in 1993 to sponsor Debian GNU/Linux. Both my interest in 1993 and my lack of interest 1992 were pragmatic means to pursue the same end: to complete the GNU system. The launch of t GNU Hurd kernel in 1990 was also a pragmatic move directed at that same end.

For all these reasons, many statements in the original edition were mistaken or incoherent. It w necessary to correct them, but not straightforward to do so with integrity short of a total rewrit which was undesirable for other reasons. Using explicit notes for the corrections was suggested, but most chapters the amount of change made explicit notes prohibitive. Some errors were too pervasiv or too ingrained to be corrected by notes. Inline or footnotes for the rest would have overwhelmed t text in some places and made the text hard to read; footnotes would have been skipped by readers tir of looking down for them. I have therefore made corrections directly in the text.

However, I have not tried to check all the facts and quotations that are outside my knowledg most of those I have simply carried forward on Williams' authority.

Williams' version contained many quotations that are critical of me. I have preserved all thes adding rebuttals when appropriate. I have not deleted any quotation, except in chapter 11 where I ha deleted some that were about open source and did not pertain to my life or work. Likewise I ha preserved (and sometimes commented on) most of Williams' own interpretations that criticized m when they did not represent misunderstanding of facts or technology, but I have freely correct inaccurate assertions about my work and my thoughts and feelings. I have preserved his person impressions when presented as such, and "I" in the text of this edition always refers to Willia except in notes labeled "RMS:".

In this edition, the complete system that combines GNU and Linux is always "GNU/Linux," a "Linux" by itself always refers to Torvalds' kernel, except in quotations where the other usage marked with "[*sic*]". See http://www.gnu.org/gnu/gnu-linux-faq.html for more explanation

why it is erroneous and unfair to call the whole system “Linux.”

I would like to thank John Sullivan for his many useful criticisms and suggestions.

Preface by Sam Williams

This summer marks the 10th anniversary of the email exchange that set in motion the writing of *Fr as in Freedom: Richard Stallman's Crusade for Free Software* and, by extension, the work preface here, *Richard Stallman and the Free Software Revolution*.

Needless to say, a lot has changed over the intervening decade.

Originally conceived in an era of American triumphalism, the book's main storyline – about on man's Jeremiah-like efforts to enlighten fellow software developers as to the ethical, if not economi shortsightedness of a commercial system bent on turning the free range intellectual culture that gav birth to computer science into a rude agglomeration of proprietary gated communities – seems almo nostalgic, a return to the days when the techno-capitalist system seemed to be working just fin barring the criticism of a few outlying skeptics.

Now that doubting the system has become almost a common virtue, it helps to look at wh narrative threads, if any, remained consistent over the last ten years.

While I don't follow the software industry as closely as I once did, one thing that leaps out nov even more than it did then, is the ease with which ordinary consumers have proven willing to cede va swaths of private information and personal user liberty in exchange for riding atop the coole technology "platform" or the latest networking trend.

A few years ago, I might have dubbed this the "iPod Effect," a shorthand salute to Apple co founder Steve Jobs' unrivaled success in getting both the music industry and digital music listeners t put aside years of doubt and mutual animosity to rally around a single, sexy device – the Apple iPod and its restrictive licensing regime, iTunes. Were I pitching the story to a magazine or newspape nowadays, I d probably have to call it the "iPad Effect" or maybe the "Kindle Effect" both in a attempt to keep up with the evolving brand names and to acknowledge parallel, tectonic shifts in th realm of daily journalism and electronic book publishing.

Lest I appear to be gratuitously plugging the above-mentioned brand names, RMS suggests that offer equal time to a pair of web sites that can spell out their many disadvantages, especially in th realm of software liberty. I have agreed to this suggestion in the spirit of equal time. The web sites h recommends are http://DefectiveByDesign.org and http://BadVista.org.

Regardless of title, the notion of corporate brand as sole guarantor of software quality in a swift changing world remains a hard one to dislodge, even at a time when most corporate brands are tradin at or near historic lows.

Ten years ago, it wasn't hard to find yourself at a technology conference listening in on conversation (or subjected to direct tutelage) in which some old-timer, Richard Stallman include offered a compelling vision of an alternate possibility. It was the job of these old-timers, I ultimatel realized, to make sure we newbies in the journalism game recognized that the tools we pride ourselves in finally knowing how to use – Microsoft Word, PowerPoint, Internet Explorer, just t name a few popular offerings from a single oft-cited vendor – were but a pale shadow of towerin edifice the original architects of the personal computer set out to build.

Nowadays, it's almost as if the opposite situation is at hand. The edifice is now a sprawlin ecosystem, a jungle teeming with ideas but offering only a few stable niches for sustainable growt While one can still find plenty of hackers willing to grumble about, say, Vista's ongoing structura flaws, Apple's dictatorial oversight of the iPhone App Store or Google's shifting definition of th word "evil" – each year brings with it a fresh crop of "digital native" consumers willing to tru corporate guidance in this Hobbesian realm. Maybe that's because many of the problems that onc made using your desktop computer such a teeth-grinding experience have largely been paved ov with the help of free software.

Whatever. As consumer software reliability has improved, the race to stay one step ahead c consumer taste has put application developers in an even tighter embrace with moneyed interests. I' not saying that the hacker ethos no longer exists or that it has even weakened in any noticeable wa I'm just saying that I doubt the programmer who generated the Facebook algorithm that rewrites th "info" pages so that each keyword points to a sponsored page, with an 80-percent semantic error ra to boot, spends much time in his new Porsche grousing about what the program really could hav achieved if only the "suits" hadn't gotten in the way.

True, millions of people now run mostly free software on their computers with many running fre software exclusively. From an ordinary consumer perspective, however, terms like "software" an "computer" have become increasingly distant. Many 2010-era cell phones could give a 2000-e laptop a run for its money in the functionality department. And yet, when it comes time to make a ce phone purchase, how many users lend any thought to the computer or software operating syste making that functionality possible? The vast majority of modern phone users base their purchasin decisions almost entirely on the number of applications offered, the robustness of the network an most important of all, the monthly service plan.

Getting a consumer in this situation to view his or her software purchase through the lens personal liberty, as opposed to personal convenience, is becoming, if not more difficult, certainly more complex endeavor.

Given this form of pessimistic introduction, why should anyone want go on and read this book?

I can offer two major reasons.

The first reason is a personal one. As noted in the Epilogue of *Free as in Freedom*, Richard and parted on less than cordial terms shortly before the publication of that book. The fault, in large par was mine. Having worked with Richard to make sure that my biographical sketch didn't run afoul free software principles – an effort that, I'm proud to say, made *Free as in Freedom* one of the fir works to employ the GNU Free Documentation License (GFDL) as a copyright mechanism – abruptly ended the cooperative relationship when it came time to edit the work and incorpora Richard's lengthy list of error corrections and requests for clarification.

Though able to duck behind my own principles of authorial independence and journalist objectivity, I have since come to lament not begging the book's publisher – O'Reilly and Associates for additional time. Because O'Reilly had already granted my one major stipulation – the GFDL – an had already put up with a heavy stream of last-minute changes on my part, however, I was hesitant t push my luck.

In the years immediately following the publication of *Free as in Freedom*, I was able to justi my decision by noting that the GFDL, just like the GNU General Public License in the software realr makes it possible for any reader to modify the book and resell it as a competitive work. As Erne Hemingway once put it, "the first draft of anything is shit." If Stallman or others within the hack community saw *Free as in Freedom* as a first draft at best, well, at least I had spared them the tin and labor of generating their own first draft.

Now that Richard has indeed delivered what amounts to a significant rewrite, I can only b remain true to my younger self and endorse the effort. Indeed, I salute it. My only remaining hope that, seeing as how Richard's work doesn't show any sign of slowing, additional documentation ge added to the mix.

Before moving on to the next reason, I should note that one of the pleasant by-products of th book is a re-opening of email communication channels between Richard and myself. The resultir communication has reacquainted me with the razor-sharp Stallman writing style.

An illustrative and perhaps amusing anecdote for anyone out there who has wrangled wi Richard in text: In the course of discussing the passage in which I observe and document the proce of Richard losing his cool amid the rush hour traffic of Kihei, Maui, a passage that served as the bas for Chapter 7 ("A Brief Journey through Hacker Hell") in the original book, I acknowledged common complaint among the book's reviewers – namely, that the episode seemed out of place, fragment of magazine-style profile interrupting a book-length biography. I told Richard that he cou discard the episode for that reason alone but noted that my decision to include it was based on tv justifications. First, it offered a glimpse of the Stallman temper, something I'd been warned about b had yet to experience in a firsthand manner. Second, I felt the overall scene possessed a certa metaphorical value. Hence the chapter title.

Stallman, to my surprise, agreed on both counts. His concern lay more in the two off-key word At one point I quote him accusing the lead driver of our two-vehicle caravan with "deliberately leading us down a dead-end street, an accusation that, if true, suggested a level of malice outside th bounds of the actual situation. Without the benefit of a recorded transcript – I only had a notebook the time, I allowed that it was likely I'd mishandled Stallman's actual wording and had made it mo hurtful than originally intended.

On a separate issue, meanwhile, Stallman questioned his quoted use of the word "fucking Again, I didn't have the moment on tape, but I wrote back that I distinctly recalled an impressiv display of profanity, a reminder of Richard's New York roots, and was willing to stand by tha memory.

An email response from Richard, received the next day, restated the critique in a way that force me to go back and re-read the first message. As it turned out, Stallman wasn't so much objecting the "fuck" as the "-ing" portion of the quote.

"Part of the reason I doubt [the words] is that they involve using fucking as an adverb," Stallma wrote. "I have never spoken that way. So I am sure the words are somewhat altered."

Touché.

The second reason a person should feel compelled to read this book cycles back to the openi theme of this preface – how different a future we face in 2010 compared to the one we were st squinting our eyes to see back in 2000. I ll be honest: Like many Americans (and non-Americans), m worldview was altered by the events of September 11, 2001, so much so that it wasn't much long after the publication of *Free as in Freedom* that my attention drifted sharply away from the fr software movement and Stallman's efforts to keep it on course. While I have managed to follow th broad trends and major issues, the day-to-day drama surrounding software standards, softwa copyrights and software patents has become something I largely skip over – the Internet nev equivalent of the Water Board notes in the local daily newspaper, in other words.

[RMS: The September 2001 attacks, not mentioned later in the book, deserve brief comme here. Far from "changing everything," as many proclaim, the attacks have, in fact, changed very litt in the U.S.: There are still scoundrels in power who hate our freedoms. The only major difference that they can now cite "terrorists" as an excuse for laws to take them away. See the political notes o http://stallman.org for more about this.]

This is a lamentable development in large part because, ten years in, I finally see the maturi 21st century in what I believe to be a clear light. Again, if this were a pitch letter to some editor, I call it "The Process Century."

By that I mean I we stand at a rare point in history where, all cynicism aside, the power to chan the world really does delegate down to the ordinary citizen's level. The catch, of course, is that th same power that belongs to you also belongs to everyone else. Where in past eras one might hav secured change simply by winning the sympathies of a few well-placed insiders, today's reform must bring into alignment an entire vector field of competitive ideas and interests. In short, being a effective reformer nowadays requires more than just titanic stamina and a willingness to cry out in th wilderness for a decade or more, it requires knowing how to articulate durable, scalable ideas, how beat the system at its own game.

On all counts, I would argue that Richard M. Stallman, while maybe not the archetype, is at th very least an ur-type of the successful reformer just described.

While some might lament a future in which every problem seems to take a few decades committee meetings and sub-committee hearings just to reach the correction stage, I, for one, see th alternative – a future so responsive to individual or small group action that some self-appointed act finally decides to put that responsiveness to the test – as too chilling to contemplate.

In short, if you are the type of person who, like me, hopes to see the 21st century follow a le bloody course than the 20th century, the Water Board – in its many frustrating guises – is where tha battle is currently being fought. As hinted by the Virgil-inspired epigraph introducing the book's fir chapter, I've always held out hope that this book might in some way become a sort of epic poem fo the Internet Age. Built around a heroic but flawed central figure, its authorial stamp should be allowe to blur with age.

On that note, I would like to end this preface the same way I always end this preface – with request for changes and contributions from any reader wishing to improve the text. Appendix B GNU Free Documentation License offers a guide on your rights as a reader to submit changes, mak corrections, or even create your own spin-off version of the book. If you prefer to simply run th

changes through Richard or myself, you can find the pertinent contact information on the Fr
Software Foundation web site.

In the meantime, good luck and enjoy the book!

Sam Williams
Staten Island, USA

Chapter 1
For Want of a Printer

I fear the Greeks. Even when they bring gift
– Virgil, *The Aene*

The new printer was jammed, again.

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

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

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

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

Driven by an instinctual urge to play with the best new equipment, programmers at the AI La promptly integrated the new machine into the lab's sophisticated computing infrastructure. The resul had been immediately pleasing. Unlike the lab's old printer, the new Xerox machine was fast. Page came flying out at a rate of one per second, turning a 20-minute print job into a 2-minute print jo The new machine was also more precise. Circles came out looking like circles, not ovals. Straig lines came out looking like straight lines, not low-amplitude sine waves.

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

Once the machine was in use, its flaws began to surface. Chief among the drawbacks was th machine's susceptibility to paper jams. Engineering-minded programmers quickly understood th

reason behind the flaw. As a photocopier, the machine generally required the direct oversight of human operator. Figuring that these human operators would always be on hand to fix a paper jam, if occurred, Xerox engineers had devoted their time and energies to eliminating other pesky problems. engineering terms, user diligence was built into the system.

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

Stallman was hardly the only AI Lab denizen to notice the problem, but he also thought of remedy. Years before, for the lab's previous printer, Stallman had solved a similar problem b modifying the software program that regulated the printer, on a small PDP-11 machine, as well as th Incompatible Timesharing System that ran on the main PDP-10 computer. Stallman couldn eliminate paper jams, but he could insert software code that made the PDP-11 check the print periodically, and report jams back to the PDP-10. Stallman also inserted code on the PDP-10 to noti every user with a waiting print job that the printer was jammed. The notice was simple, somethi along the lines of "The printer is jammed, please fix it," and because it went out to the people with th most pressing need to fix the problem, chances were that one of them would fix it forthwith.

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

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

Such clever fixes were a trademark of the AI Lab and its indigenous population of programmer Indeed, the best programmers at the AI Lab disdained the term programmer, preferring the mor slangy occupational title of hacker instead. The job title covered a host of activities – everything fro creative mirth making to the improvement of existing software and computer systems. Implicit with the title, however, was the old-fashioned notion of Yankee ingenuity. For a hacker, writing a softwar program that worked was only the beginning. A hacker would try to display his cleverness (ar impress other hackers) by tackling an additional challenge: to make the program particularly fas small, powerful, elegant, or somehow impressive in a clever way. [1]

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

When Stallman noticed the jamming tendency in the Xerox laser printer, he thought of applyin the old fix or "hack" to this printer. In the course of looking up the Xerox laser-printer softwar however, Stallman made a troubling discovery. The printer didn't have any software, at least nothin Stallman or a fellow programmer could read. Until then, most companies had made it a form courtesy to publish source-code files–readable text files that documented the individual softwa commands that told a machine what to do. Xerox, in this instance, had provided software files only compiled, or binary, form. If programmers looked at the files, all they would see was an endle stream of ones and zeroes – gibberish.

There are programs, called "disassemblers," to convert the ones and zeroes into low-lev machine instructions, but figuring out what those instructions actually "do" is a long and hard tas known as "reverse engineering." To reverse engineer this program could have taken more time tha five years' worth of jammed printouts. Stallman wasn't desperate enough for that, so he put th problem aside.

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

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

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

Through this simple system of intellectual accretion, hackers at the AI Lab and other places bui up robust creations. Not every programmer participating in this culture described himself as a hacke but most shared the sentiments of Richard M. Stallman. If a program or software fix was good enoug to solve your problems, it was good enough to solve somebody else's problems. Why not share it o

of a simple desire for good karma?

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

Likewise, Stallman was annoyed that Xerox had not provided the source-code files, but not y 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 t absence of source code reflected an intentional decision, and that asking them to change it would l futile.”

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

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

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

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

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

In true engineer-to-engineer fashion, the conversation was cordial but blunt. After brief

introducing himself as a visitor from MIT, Stallman requested a copy of the laser-printer source co
that he wanted to modify. To his chagrin, the researcher refused.

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

Memory is a funny thing. Twenty years after the fact, Stallman's mental history tape is blank
places. Not only does he not remember the motivating reason for the trip or even the time of ye
during which he took it, he also has no recollection of who was on the other end of the conversatio
According to Reid, the person most likely to have fielded Stallman's request is Robert Sproull,
former Xerox PARC researcher and current director of Sun Laboratories, a research division of th
computer-technology conglomerate Sun Microsystems. During the 1970s, Sproull had been th
primary developer of the laser-printer software in question while at Xerox PARC. Around 198
Sproull took a faculty research position at Carnegie Mellon where he continued his laser-printer wo
amid other projects.

When asked directly about the request, however, Sproull draws a blank. "I can't make a factu
comment," writes Sproull via email. "I have absolutely no recollection of the incident."

"The code that Stallman was asking for was leading-edge, state-of-the-art code that Sproull ha
written in the year or so before going to Carnegie Mellon," recalls Reid. If so, that might indicate
misunderstanding that occurred, since Stallman wanted the source for the program that MIT had us
for quite some time, not some newer version. But the question of which version never arose in th
brief conversation.

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

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

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

Twenty years after the fact, the anger still lingers, and Stallman presents the event as one th

made him confront an ethical issue, though not the only such event on his path. Within the next fe
months, a series of events would befall both Stallman and the AI Lab hacker community that wou
make 30 seconds worth of tension in a remote Carnegie Mellon office seem trivial by compariso
Nevertheless, when it comes time to sort out the events that would transform Stallman from a lo
hacker, instinctively suspicious of centralized authority, to a crusading activist applying tradition
notions of liberty, equality, and fraternity to the world of software development, Stallman singles o
the Carnegie Mellon encounter for special attention.

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

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

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

As an elite programmer at one of the world's elite institutions, Stallman had been perfect
willing to ignore the compromises and bargains of his fellow programmers just so long as they didn
interfere with his own work. Until the arrival of the Xerox laser printer, Stallman had been content
look down on the machines and programs other computer users grimly tolerated.

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

That Xerox had offered some programmers access to additional gifts in exchange for secrecy w
also galling, but Stallman takes pains to note that, if presented with such a quid pro quo bargain at
younger age, he just might have taken the Xerox Corporation up on its offer. The anger of th
Carnegie Mellon encounter, however, had a firming effect on Stallman's own moral lassitude. No
only did it give him the necessary anger to view such future offers with suspicion, it also forced hi
to turn the situation around: what if a fellow hacker dropped into Stallman's office someday and
suddenly became Stallman's job to refuse the hacker's request for source code?

"When somebody invited me to betray all my colleagues in that way, I remembered how angry

was when somebody else had done that to me and my whole lab," Stallman says. "So I said, 'Than you very much for offering me this nice software package, but I can't accept it on the conditions th you're asking for, so I'm going to do without it.' "

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

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

Hence the importance of the laser printer and the encounter that resulted from it. Without i Stallman says, his life might have followed a more ordinary path, one balancing the material comfor of a commercial programmer with the ultimate frustration of a life spent writing invisible softwa code. There would have been no sense of clarity, no urgency to address a problem others weren addressing. Most importantly, there would have been no righteous anger, an emotion that, as we soo shall see, has propelled Stallman's career as surely as any political ideology or ethical belief.

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

Endnotes

1 For more on the term "hacker," see Appendix A – Hack, Hackers, and Hacking.

Chapter 2
2001: A Hacker's Odyssey

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-conditionin vents create a surrounding moat of hot air, discouraging loiterers and solicitors alike. Visitors wh breach the moat encounter another formidable barrier, a security check-in counter immediately insid the building's single entryway.

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

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

Once inside the auditorium, a visitor finds the person who has forced this temporary shutdown building security procedures. The person is Richard M. Stallman, founder of the GNU Project, origin president of the Free Software Foundation, winner of the 1990 MacArthur Fellowship, winner of th Association of Computing Machinery's Grace Murray Hopper Award (also in 1990), corecipient of th 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 ow http://www.gnu.org site, Stallman is in Manhattan, his former hometown, to deliver a muc anticipated speech in rebuttal to the Microsoft Corporation's recent campaign against the GN General Public License.

The subject of Stallman's speech is the history and future of the free software movement. Th location is significant. Less than a month before, Microsoft senior vice president Craig Mund appeared at the nearby NYU Stern School of Business, delivering a speech blasting the GNU Gener Public License, or GNU GPL, a legal device originally conceived by Stallman 16 years before. Built counteract the growing wave of software secrecy overtaking the computer industry – a wave fir noticed by Stallman during his 1980 troubles with the Xerox laser printer – the GPL has evolved into central tool of the free software community. In simplest terms, the GPL establishes a form c communal ownership – what today's legal scholars now call the "digital commons" – through th legal weight of copyright. The GPL makes this irrevocable; once an author gives code to th community in this way, that code can't be made proprietary by anyone else. Derivative versions mu carry the same copyright license, if they use a substantial amount of the original source code. For th reason, critics of the GPL have taken to calling it a "viral" license, suggesting inaccurately that spreads itself to every software program it touches. [1]

In an information economy increasingly dependent on software and increasingly beholden

software standards, the GPL has become the proverbial “big stick.” Even companies that once deride it as “software socialism” have come around to recognize the benefits. Linux, the kernel developed b Finnish college student Linus Torvalds in 1991, is licensed under the GPL, as are most parts of th 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 b the worldwide hacker community. Instead of viewing this community as a threat, high-tech companie like IBM, Hewlett Packard, and Sun Microsystems have come to rely upon it, selling softwa applications and services built to ride atop the ever-growing free software infrastructure. [2]

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

Hence the company’s growing concern over the GPL’s rate of adoption. Hence the recent Mundi speech blasting the GPL and the “open source” approach to software development and sale (Microsoft does not even acknowledge the term “free software,” preferring to use its attacks to dire attention towards the apolitical “open source” camp described in chapter 11, and away from the fr software movement.) And hence Stallman’s decision to deliver a public rebuttal to that speech on th same campus here today.

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

Among those who value progress above freedom, the fact that the high-technology marketpla has come so far in such little time is cited both for and against the GNU GPL. Some argue in favor the GPL, pointing to the short lifespan of most computer hardware platforms. Facing the risk buying an obsolete product, consumers tend to flock to companies with the best long-term survival. A a result, the software marketplace has become a winner-take-all arena. [3] The proprietary softwa environment, they say, leads to monopoly abuse and stagnation. Strong companies suck all the oxyge out of the marketplace for rival competitors and innovative startups.

Others argue just the opposite. Selling software is just as risky, if not more risky, than buyin software, they say. Without the legal guarantees provided by proprietary software licenses, not t mention the economic prospects of a privately owned “killer app” (i.e., a breakthrough technology th

launches an entirely new market), [4] companies lose the incentive to participate. Once again, th market stagnates and innovation declines. As Mundie himself noted in his May 3rd address on th same campus, the GPL's "viral" nature "poses a threat" to any company that relies on the uniquenes of its software as a competitive asset. Added Mundie:

> It also fundamentally undermines the independent commercial software sector because effectively makes it impossible to distribute software on a basis where recipients pay for th product rather than just the cost of distribution. [5]

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

Nevertheless, the battle for momentum is an important one in the software industry. Eve powerful vendors such as Microsoft rely on the support of third-party software developers who tools, programs, and computer games make an underlying software platform such as Windows mo attractive to the mainstream consumer. Citing the rapid evolution of the technology marketplace ov the last 20 years, not to mention his own company's impressive track record during that perio Mundie advised listeners to not get too carried away by the free software movement's rece momentum:

> Two decades of experience have shown that an economic model that protects intellectu property and a business model that recoups research and development costs can create impressiv economic benefits and distribute them very broadly. [6]

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

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

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

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

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

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

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

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

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

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

"Richard is the perfect example of somebody who, by acting locally, started thinking globall
[about] problems concerning the unavailability of source code," says Schonberg. "He has developed
coherent philosophy that has forced all of us to reexamine our ideas of how software is produced,
what intellectual property means, and of what the software community actually represents." [7]

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

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

For the uninitiated, Stallman dives into a quick free software warm-up analogy. He likens
software program to a cooking recipe. Both provide useful step-by-step instructions on how
complete a desired task and can be easily modified if a user has special desires or circumstance
"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 yo
should cut down on salt – whatever."

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

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

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

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

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

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

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

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

Richard Matthew Stallman’s rise from frustrated academic to political leader over the last 2
years speaks to many things. It speaks to Stallman’s stubborn nature and prodigious will. It speaks t
the clearly articulated vision and values of the free software movement Stallman helped build.
speaks to the high-quality software programs Stallman has built, programs that have cemente
Stallman’s reputation as a programming legend. It speaks to the growing momentum of the GPL,
legal innovation that many Stallman observers see as his most momentous accomplishment.

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

Maybe that’s why, even at a time when most high-technology stars are on the wane, Stallman
star has grown. Since launching the GNU Project in 1984, [8] Stallman has been at turns ignore

satirized, vilified, and attacked–both from within and without the free software movement. Through all, the GNU Project has managed to meet its milestones, albeit with a few notorious delays, and sta relevant in a software marketplace several orders of magnitude more complex than the one it entere 18 years ago. So too has the free software ideology, an ideology meticulously groomed by Stallma himself.

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

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

Explaining how a journey that started with a laser printer would eventually lead to a sparri match with the world's richest corporation is no easy task. It requires a thoughtful examination of th forces that have made software ownership so important in today's society. It also requires a thoughtfu examination of a man who, like many political leaders before him, understands the malleability human memory. It requires an ability to interpret the myths and politically laden code words that hav built up around Stallman over time. Finally, it requires an understanding of Stallman's genius as programmer and his failures and successes in translating that genius to other pursuits.

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

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

"I had just the right skills," says Stallman, summing up his decision for launching the GN Project to the audience. "Nobody was there but me, so I felt like, 'I'm elected. I have to work on thi If not me, who?' "

Endnotes

1 Actually, the GPL's powers are not quite that potent: just putting your code in the same computer with a GPL-covered program does not put your code under the GPL.

"To compare something to a virus is very harsh," says Stallman. "A spider plant is a more accurate comparison; it goes to another place if you actively take a cutting."

For more information on the GNU General Public License, visit http://www.gnu.org/copyleft/gpl.html.

---

2 Although these applications run on GNU/Linux, it does not follow that they are themselves free software. On the contrary, most of them applications are proprietary software, and respect your freedom no more than Windows does. T
may contribute to the success of GNU/Linux, but they don't contribute to the goal of freedom for which it exists.

3 See Shubha Ghosh, "Revealing the Microsoft Windows Source Code," *Gigalaw.com* (January, 2000), http://www.gigalaw.com/.

4 Killer apps don't have to be proprietary. Still, I think the reader gets the point: the software marketplace is like the lottery. The bigger the potential payoff, the more people want to participate. For a good summary of the killer
phenomenon, see Philip Ben-David, "Whatever Happened to the 'Killer App'?", *e-Commerce News* (December 7, 2000), http://www.ecommercetimes.com/story/5893.html.

5 See Craig Mundie, "The Commercial Software Model," senior vice president, Microsoft Corp., excerpted from an online transcript of Mundie's May 3, 2001, speech to the New York University Stern School of Busi
http://www.microsoft.com/presspass/exec/craig/05-03sharedsource.asp.

6 *Ibid.*

7 If this were to be said today, Stallman would object to the term "intellectual property" as carrying bias and confusion. See http://www.gnu.org/philosophy/not-ipr.html.

8 The acronym GNU stands for "GNU's not Unix." In another portion of the May 29, 2001, NYU speech, Stallman summed up the acronym's origin:

> We hackers always look for a funny or naughty name for a program, because naming a program is half the fun of writing the program. We also had a tradition of recursive acronyms, to say that the program that you're writing is simil
> some existing program…I looked for a recursive acronym for Something Is Not UNIX. And I tried all 26 letters and discovered that none of them was a word. I decided to make it a contraction. That way I could have a three-letter acro
> for Something's Not UNIX. And I tried letters, and I came across the word "GNU." That was it.

Although a fan of puns, Stallman recommends that software users pronounce the "g" at the beginning of the acronym (i.e., "gah-new"). Not only does this avoid confusion with the word "gnu," the name of the African ante
Connochaetes gnou, it also avoids confusion with the adjective "new." "We've been working on it for 17 years now, so it is not exactly new any more," Stallman says.

Source: author notes and online transcript of "Free Software: Freedom and Cooperation," Richard Stallman's May 29, 2001, speech at New York University, http://www.gnu.org/events/rms-nyu-2001-transcript.t