The Last Hacker:

Richard Stallman & the Free Software Foundation

Simson L. Garfinkel

A look at one man's personal crusade to make software "free."

Richard M. Stallman wakes up in his office at 10:00 a.m., his eyes bleary from a long night of hacking that ended just six hours before.

Stallman usually sleeps until noon, but this day he's busy rounding up a group of fellow computer hackers for a day-long political protest. Their target: "Science and Engineering Day" at the Massachusetts Institute of Technology, sponsored by Apple Computer Corporation. The League for Programming Freedom is urging people to boycott Apple, Lotus, and Ashton-Tate. "Apple and the others are trying to use 'look and feel' lawsuits to lock their competitors out of the market," says Stallman, the League's president, to a person passing in the MIT lobby. If Apple's lawsuit is successful, Stallman believes, it could mean an end to open competition in the software industry. Many people pass, uninterested by what Stallman sees as one of the biggest problems facing the future of business in America.

"You can tell from their faces the people who are just interested in what's going to profit them today," says Stallman, commenting on the passing students and professors. "Their intentness, their grimness, their utter lack of humor."

Stallman returns from the protest a few hours later to work on "GCC," a compiler for the C programming language. (A compiler is a program that translates the high-level statements that a programmer writes into the actual machine-level codes that a computer executes.) Stallman has been working on GCC for nearly four years; many believe the compiler is one of the best ever written.

By many accounts, Stallman is one of the best programmers in the United States. His going rate for consulting is $260 an hour. This year he received an unsolicited "genius" grant from the MacArthur Fellows Program, in the amount of $240,000 over five years.

But these days, Stallman spends nearly all his time in his crowded, 130-square-foot office in MIT's Technology Square, writing "free" software. As both president and master programmer of the Free Software Foundation, Stallman's life's work is overseeing the development of an operating system for workstations and scientific computers—including a word processor, a spreadsheet, compilers, and an array of powerful programmer's tools—and then seeing the software given away.

For Stallman, politics and computer programming go hand in hand. His crusade is for personal freedom—in particular, the freedom of programmers to write whatever programs they want, to share programs and information without being branded "pirates," and to learn from programs that other programmers have written.

"Free software is a matter of liberty, not price," says Stallman. "It's the freedom to give copies out when you want to and to change how the program works to suit yourself."

Computers are general-purpose machines for manipulating ideas. Simply by changing a program, the same computer can manage literature, music, or financial projections with equal ease. The power of computers stems from the fact that they can copy information—including the information that makes up a program—without changing it, and at virtually no cost.

Stallman sees a future in which programmers are hired to make a program exist—but whose programs then become the property of society as a whole. A world in which programs are kept in libraries, free for anyone to make a copy and share it with friends. A world where software is "free."

It's also a world in which computer users aren't trapped by what MIT Professor Louis Menand calls "the Tyranny of the Software."

In today's world, says Menand, most computer users can't make improvements or fix problems in programs they purchase. If the company that wrote the program doesn't want to make the change, the user has no recourse except to find another program.

This is because most software publishers sell just their program's machine code—the actual set of instructions that the computer's central processing unit executes in order to perform the tasks originally outlined by the programmer. Programmers don't write in machine code; they write in English-like "source code," which is translated into machine code by programs called compilers—programs like Stallman's "GCC." Even most so-called "public-domain" software that's distributed without cost comes in machine-code-only form. Getting the source code is often impossible.

Having just the machine code to a program is like having a car with the hood welded shut: You can run it, but it's very difficult to open it up and see how the program works, or to fix it if it has a bug.

With the source code, even a mediocre programmer can copy a procedure out of one program and use it as a building block in another. Having just the machine code is like having an erector set with all the pieces glued together.

Most computer companies rely upon the indecipherability of machine code to protect their trade secrets and other intellectual property. A company that spends 10 man-years developing a $500 financial modeling package doesn't want its competitors to easily extract the guts, change a few menus so that the program looks different, and resell it for $49.95.

But the preponderance of machine, or binary code, is a distinct disadvantage for users. "If you don't have the source code to a product, you're at the mercy of the vendor, because you cannot fix bugs that are in the binary," says Jeffrey I. Schiller, manager of systems and operations for MIT Network Services. Bugs that allow unauthorized access to a computer have "severe security implications," says Schiller, because until the vendor fixes and redistributes the program, the system is susceptible to attack.

"I think it's dangerous for society for people to rely on programs where we cannot see the details," says Harold Abelson, a professor of computer science at MIT and a member of the Free Software Foundation's board of directors. "I don't object that the Coca-Cola formula is secret, because I don't think it's very important." But computers play an increasingly important role in everybody's life, says Abelson. "Why should I be in the position of having to rely very seriously on something I'm not permitted to understand and not permitted to fix?"

Turning around the computer software industry at this late date might be impossible, but if anyone can do it, Richard M. Stallman and the Free Software Foundation stand the best chance.

Stallman's life's work is overseeing the development of an operating system for workstations and scientific computers—including a word processor, a spreadsheet, compilers, and an array of powerful programmer's tools—and then seeing the software given away.

The summer after he graduated from high school, Stallman was hired as a programmer at the IBM New York Scientific Center. He took algorithms from researchers and coded them into FORTRAN, and he wrote a text editor in an APL.

He entered Harvard University in the fall of 1970. The following spring he started a project of traveling to different computer centers and reading their system documentation. "I was curious about different computer systems. I wanted to learn about as many different kinds of computers as possible," he recalls. One day, he went to the MIT Artificial Intelligence Laboratory because he'd been told that they had a PDP-10 computer with 256,000 words of memory—an unheard-of amount in 1971.

"When I got there, people asked me, 'Are you looking for a summer job?' I hadn't thought of asking for a summer job—I was just looking for manuals. There weren't very many manuals, so I talked to people instead." He was hired on the spot.

Stallman graduated *"magna cum laude* with highest honors" from Harvard in 1974. ("This says one thing in Latin and something contradictory in English. Don't blame me!" he quips.) He stayed at the AI Lab. A year later, he finished a revolutionary programmer's editor called Emacs ("editing macros"), which let a person edit many files at the same time, peruse electronic mail, compile programs, and even customize the editor to suit his or her own personal preferences.

"He was among the most important of the laboratory's contributors," remembers Patrick H. Winston, director of the AI Laboratory. That was no small compliment.

In the 1970s, the AI Lab was a veritable rain forest of computer program evolution. Many of the computers at the Lab ran operating systems that had been either developed or at least substantially modified there. The Lab even designed its own computer, called a Lisp Machine. Since everyone in the lab had access to the system source code, whenever someone discovered a bug, they fixed it. If somebody wanted to add a feature to a program—make it do something useful that it hadn't done before—they went ahead and added it.

Stallman remembers the way the Lab's programmers had modified the software for the Xerox Graphics Printer, one of the very first laser printers, that Xerox had given the lab: "It would send you a message when your document had actually been printed; it would send you a message if you had anything queued and there was a paper jam."

In 1978, Xerox replaced the XGP with a new laser printer called a Dover. "We wanted to put those features into the Dover program, but we couldn't—we didn't have the source code." Xerox wouldn't put the features into the program either, Stallman says. "So we had to suffer with paper jams that nobody knew about."

A year later, Stallman met a programmer at Carnegie-Mellon University who had the source code to the Dover, "but he refused to let me have it because he had signed a non-disclosure agreement. I learned what it was like to be the victim."

Something like that never would have happened in computing's early days. When Stallman was an undergraduate at Harvard, the computer center refused to install any program if they couldn't show the source code to students. "They said they had the policy because it was an educational institution," remembers Stallman.

But as the business of computing exploded in the 1970s and '80s, companies stopped seeing software as a tool for selling hardware and started viewing it as a moneymaker in its own right. And those companies naturally wanted to restrict access to their source code for the same reason that Stallman wanted access for everybody: Having the source code makes it easy to incorporate the ideas from one program into others.

Keeping source code proprietary may hamper the competition, but it also slows down the rate of progress for the entire computer industry. "It's a betrayal of society for personal advantage," says Stallman.

Despite the changes in the outside world, within the tightly knit community of the AI Lab, software sharing was still the rule. One of the best examples was the operating system for the Lab's Lisp Machine. In the beginning of the 1980s, MIT researchers started two rival companies, Symbolics and Lisp Machine Incorporated, to commercialize the computer. Hackers began leaving the lab to work for one of the two companies, yet the sharing continued: Improvements to the Lisp Machine operating system, whether they were made at Symbolics, LMI, or the AI Lab, were freely shared among the other two partners.

The battlelines are drawn. Then, in 1982, Symbolics' programmers developed a new feature, "fast-flavors," that the company refused to share with LMI. MIT could have the source code, as long as it didn't become part of the shared source code. To many hackers at the AI Lab, such a move was blasphemy.

"RMS and I went into a crash mode," recalls Richard Greenblatt, the inventor of the Lisp machine and founder of LMI. "We hacked around the clock for two solid weeks, at the end of which we had put a comparable fast-flavors feature into the MIT sources."

Under Symbolic's new rules, people at the AI Lab could share with Symbolics or share with LMI, says Stallman, but not with both. "I decided that since it was Symbolics that was forcing me to choose a side, I would choose the side against them."

For the following two years, Stallman took every improvement that Symbolics' team of programmers made to their Lisp machine operating system and rewrote the improvements for the operating system used by MIT and LMI. Steven Levy tells the story at the end of his book, *Hackers.*

By 1984, Stallman had had enough. "I decided that it was time to start a counterattack. Instead of continuing to punish those who had destroyed the old software sharing community, I wanted to start a new one," he says.

He quit his job at the AI Lab so that MIT could claim no ownership to future programs that he wrote—after all, they had taken his code once and licensed it to a company that had prevented people from sharing it—and started on the task of building a new operating system from scratch.

An open world. While Stallman fought his battle with Symbolics, a revolution of sorts in the world of computers was taking place.

Since the birth of computing, computers had always been sold with operating systems designed by their manufacturer, operating systems that could run only on a certain kind of computing hardware. This trend continues to this day: Digital Equipment Corporation's popular VMS operating system runs only on DEC VAX computers; likewise, Hewlett-Packard's MPE operating system runs only on HP minicomputers. Even the MIT AI Lab's Lisp machine operating system ran only on, well, Lisp machines.

Ken Thompson and Dennis Ritchie at Bell Laboratories set a different movement in motion in 1969 when they took an unused PDP-7 computer and wrote a new operating system called "Unix." From the beginning, Unix was designed to be a portable operating system. Only a tiny amount of the system was written in assembly language—the fundamental instruction set that is unique to every different kind of computer system. The rest of Unix was written in a language Ritchie invented called "C." The language was compiled by the Portable C Compiler, "pcc."

Bell Laboratories made the source code for Unix available to educational institutions for a nominal charge, and Unix slowly grew in popularity. In 1977, the Computer Systems Research Group at the University of Berkeley started distributing additions to Unix called the Berkeley Standard Distribution, or BSD. In 1979, Berkeley released BSD 3, a version of Unix for Digital's VAX.

Unix is based on the idea that an operating system should consist of many small programs, called tools, each of which does one thing well. By stringing together a number of small tools, Unix programmers can quickly accomplish relatively complicated tasks. Because of its simplicity and portability, Unix increasingly became the operating system of choice for universities and research laboratories. A community of Unix enthusiasts grew up around the world, establishing its own electronic network with a system called "uucp" that came standard with the operating system. By 1985, that network consisted of more than an estimated 7,000 computers serving over 200,000 users, according to an article published in a 1986 issue of *Communications of the ACM*. Today, Unix runs on more than 1.2 million computers across the world, from the lowly IBM PC XT to Cray supercomputers.

But Bell Laboratory's parent company, AT&T, never relinquished ownership to Unix. Every copy of Unix had to be licensed. While the operating system's source code was readily available to universities, its price was prohibitive for many businesses that were increasingly looking toward Unix as a way of developing application programs that would not be intractably bound to the hardware of a particular computer manufacturer.

By granting companies the right to use the copyright system to outlaw the duplication of their programs, the government has created monopolies.

For all these reasons, when Stallman decided to write a new operating system from scratch, he decided to model it on Unix. It would even run the same programs, only better, he resolved. He named his project "GNU," a recursive acronym meaning "GNU's Not Unix."

Working tirelessly that spring in an empty room at the MIT AI Lab, Stallman developed a version of his Emacs text editor for the Unix operating system. His recorder always nearby, sometimes he played music while waiting for his program to compile. "It charms the compiler so it won't get errors," he joked.

Stallman chose to start with Emacs for many reasons. Since he had invented Emacs in 1975, it had been widely used and imitated. By providing the world with a free version of the now-popular program, he would gain thousands of instant converts to the religion of free software.

A second, more practical reason for writing Emacs first was that, as a programmer, Stallman uses Emacs more than any other program. Emacs gave Stallman a powerful tool for developing the rest of GNU.

Today, GNU Emacs is available on practically every computer that runs Unix—and many that don't. Like all GNU software, Emacs is written to be as portable as possible, and programmers around the world—some volunteers, but most of whom are paid by companies or universities—have taken up the task of the ports. Nearly all of the changes, as well as bug fixes and new features, have been returned to Stallman, tested, and integrated into his master copy. The program is now shipped as a standard part of the Unix operating system provided by over a dozen manufacturers.

"I would say about 50 percent of the engineers use GNU Emacs," says Jon Hall, one of Digital's product managers for ULTRIX Workstation Software.

Finding out how many computers are running GNU Emacs is difficult, says Len Tower, Jr., a senior systems programmer at Boston University and a member of the GNU Project, "because of the way we do our distribution. We encourage people to pass it on."

Indeed, says the AI Lab's Winston, the "world-wide acceptance of Emacs" has earned Stallman an international reputation for being "a fantastic hacker."

But far more important than the program itself, says Stallman, is the license agreement that protects it: the Copyleft.

Even though anybody may make a copy of GNU Emacs and give it away, the program is not in the public domain. Instead, Emacs—and all of GNU software—is protected by an iron-clad agreement that assures that anybody who obtains a copy of GNU Emacs will be able to distribute their own copies. Companies can sell copies of Emacs, says Stallman, but they can't keep their customers from giving away or selling copies of Emacs that they buy. Source code *must* be provided upon request. Any changes or enhancements to Emacs are free; likewise, if part of Emacs is used in another program, that entire program becomes Free Software.

Despite the success of Emacs, it created "a tension between GNU and the Unix world," says Len Tower. It violated the central UNIX philosophy.

For dyed-in-the-wool Unix hackers, who believe that the best operating system is one made up of many small tools, each one serving a single function, Emacs was an abomination. For years, the standard Unix editor was called "ed," a tiny program that let a person change the contents of a text file a line at a time. Emacs, by contrast, was 30 times the size of "ed" and came with functions for compiling programs, reading electronic mail, pursuing online documentation, and even compiling Lisp programs.

Then there was the problem with the project's name. "GNU is not Unix. We seem to have the arrogance to say we're going to reinvent Unix better than anyone else," says Tower.

The Free Software Foundation arises. In 1986, Stallman started working on the next part of GNU: the GNU C Compiler, "GCC." That year, he also became president of the newly incorporated Free Software Foundation.

Today, the foundation has a paid staff of 14 employees, including six full-time programmers and two full-time technical writers. In 1989, the foundation took in $330,377 from the sale of manuals and computer programs, and $267,782 in donations, according to Robert J. Chassell, the foundation's treasurer. The foundation has over a dozen workstations, all donated.

Foundation programmers get paid a mere $25,000 a year, although they could easily command salaries of $30,000 to $60,000 on the open market. "We have more problems hiring people than anybody else, because we merely pay the best activist wage in Boston," chuckles Len Tower.

Indeed, Stallman's programmers are committed to the cause: "I love to program," says Mike Rowan. "I hate corporate Mickey Mouse. I want to do what's right. I want to hack, and this is one of the few hacking jobs left in the country."

As computing exploded in the '70s and '80s, companies stopped seeing software as a tool for selling hardware and started viewing it as a moneymaker in its own right.

Beyond the programmers and the dollars is the fact that the foundation is spearheading a popular movement among computer programmers that software must be free. "Things are moving very fast now because other people are doing the work, and I'm integrating it," explains Stallman. "At this point, so many people are contributing."

To pay his bills, Stallman spends one month a year consulting, at the rate of $260 an hour. His one condition: Any program he writes must be free. Last year, Intel hired him to port GCC to its i860 microprocessor; other companies are standing in line for his time.

Any friction that GNU Emacs created with the Unix community was wiped away by the success of GCC.

"Here was this compiler that was outdoing pcc!" exclaims Len Tower, who sits on the Free Software Foundation's board of directors. "It was not only portable, it was producing good code, in some cases the best code."

Stallman's compiler consists of two main parts: a "front end" that translates the C programming language into an internal representation, and a "back end" that translates that representation into the actual machine instructions used by the computer. So far, most of the work on the compiler has been aimed at engineering it to produce smaller, faster code, as well as increasing the number of different kinds of microprocessors it can generate code for. Today GCC has back ends for 11 different kinds of microprocessors, including the 68020, VAX, 80386, SPARC and MIPS.

"The fact that this group of free software people with these weird ideas had built this compiler just blew their minds away," continues Tower. "It's proved that some of what GNU was doing had relevance to the Unix community."

Keeping source code proprietary may hamper the competition, but it also slows down the rate of progress for the entire computer industry.

But the UNIX community was doing more than simply *using* Stallman's tools: They were building upon them. At MCC in Austin, Texas, Michael Tiemann developed a new front end for GCC to let the compiler process C++, an advanced object-oriented language based on C. Tiemann then convinced MCC to free his front end. The resulting program, G++, is now distributed along with GCC. Other programmers are working on front ends that will enable GCC to compile FORTRAN and Pascal. When the project is finished, Stallman will have overseen the creation of a compiler that could have been only imagined a decade ago: a universal program for efficiently translating from the most commonly used high-level languages to most machine languages, available for free to anybody for the asking.

GCC's users are enthusiastic about the program to the point of being evangelistic: "The code quality of GCC is as good or better than that of any commercial Unix compiler I've used; it comes with free sources; it has a fast turnaround time for bug fixes; it's easily ported to new machines; it has a substantial and growing user base. The typical vendor-supplied compilers simply can't compete," says Donn Seeley, a senior systems programmer at the University of Utah.

When Seeley's group was contracted by Hewlett-Packard to port the Berkeley-enhanced Unix operating system to HP's 9000-series computer, the first thing they did was replace HP's C compiler with GCC.

"The only way for commercial compilers to continue to exist in the face of GCC is to offer features that GCC does not," says Seeley. "I see this as the basic function of GCC in the marketplace—GCC is really just an old-technology compiler done well and distributed for free, so the many vendors whose compilers are not even current with old technology will lose. New compilers must be at least as good as GCC, or the market won't accept them."

It was partly because of Utah's decision to use GCC that Hewlett-Packard gave a $100,000 grant and six HP9000 workstations to the Free Software Foundation. The prime mover behind that grant was Roy Suza, who coordinated HP's grants to university research groups.

Giving the Free Software Foundation the grant made sense, says Suza, since many of the research groups that Hewlett-Packard supported were using GCC productively. Hewlett-Packard was even using GCC internally, to compile programs for some of their laboratory instruments.

NeXT, the computer company started by Apple's ousted founder Steve Jobs, has adopted GCC as its only compiler for its new computer system.

Nevertheless, says Tower, "There's a lot of vendor reluctance to GCC." Besides HP and NeXT, computer manufacturers haven't switched from their current compilers to GCC, even when GCC produces a significant speed-up in their code.

"Companies have this feeling that they really want to own and control any software that they distribute," says Professor Abelson. Manufacturers feel that they can't provide support for a program unless they're free to make changes in it, he adds, and most manufacturers are unwilling to invest any money in supporting GCC since they don't own the changes. The reason is the GNU Copyleft.

"The GNU license doesn't have any loopholes in it," says Abelson. "If it were an ordinary license that said 'if you want to use it, you have to pay a million dollars,' that would be okay." The idea that any work done on the compiler is free scares them off.

At Digital Equipment Corporation, Jon Hall also speaks of support. Support, says Hall, is hampered when users have access to the system source code: "Part of the problem in handing out source code to customers is that they may not only change things in the operating system that need fixing, but may change things that do *not* need fixing. In doing so, they may introduce bugs that don't exist in the code that ships."

Since Digital charges significantly more than the foundation for its software and doesn't allow it to be freely redistributed, the company is able to offer a different caliber of support.

"Digital supports people in mass quantities," says Hall. "Thousands of customers at one time. Some of our customers aren't even computer literate, much less Unix literate."

Many of those customers demand peculiarities in their support that the Free Software Foundation can't provide. "We have some customers who can't upgrade their system at a particular time—they like to remain on a particular version of the OS, even though there are new patches available," says Hall. To accommodate such users, Digital keeps computers running with older versions of its software. When bugs are reported, engineers verify that the problems exist and give the users patches without forcing them to bring their software up to the current revision.

But although the Free Software Foundation doesn't support its software, the foundation doesn't prevent other people from supporting it, just the same. Since GNU software is distributed with its own source code, providing service for the software is open to the free market. Already, a for-profit consulting firm in San Francisco, Cygnus Software, is making a business out of providing support for GNU and writing programs that are protected by the Copyleft. And Emacs itself comes with a list of more than 50 computer consultants around the world who have offered to do the same.

If Project GNU is successful, it may merely further and strengthen a fundamental change that is already taking place in the computer industry.

In the 1980s, computer shoppers demanded "open systems"—that is, computer operating systems that don't marry users to the equipment of one hardware vendor or another. As the only multi-tasking, multi-user operating system that could run on more than one kind of hardware platform, Unix became the *de facto* definition of an "open system." Although two flavors of Unix existed—one from AT&T and one from the University of Berkeley—the differences between the two were insignificant compared with the differences between them and any other.

Barred from the computer business until the break-up of the Bell System, AT&T entered the computer business with a vengeance. The company immediately tightened the terms of its liberal Unix licensing agreements. One of the first casualties was the Computer System Research Group at the University of Berkeley: AT&T stopped selling the low-cost Unix Version 7 license on which the Berkeley operating system was based. From that point on, the only way for a school or business to get Berkeley Unix was to first purchase the much more expensive AT&T Unix System V.

As AT&T tightened the terms of its licensing agreements even more, a group of seven computer companies—Apollo Computer, Digital Equipment Corp., Hewlett-Packard, IBM, and three major European computer manufacturers—rebelled. In May 1988, these companies announced the formation of the Open Software Foundation, a consortium that would develop and maintain a version of the Unix operating system based on an earlier version that had a more cordial licensing agreement. The basic idea behind the OSF is that the same operating system should run on the hardware from any manufacturer.

If it's suddenly possible to run the same high-caliber operating system—be it OSF or GNU—on any manufacturer's hardware, then hardware considerations such as price, performance, and reliability will be the only reasons to purchase a computer from one manufacturer or another. That spells a future of intense competition for hardware manufacturers. "[You] can be the fastest for only a short period of time," says Digital's Hall.

In order to be competitive, computer companies are going to have to stop selling "hardware" and "software" and start selling total solutions, from selecting and installing the computers to customizing the software to meet a particular end-user's needs. "I believe that every computer company is going to have to get into that type of business, given the new trends of open software, even without the Free Software Foundation being there," says Hall.

Having just the machine code to a program is like having a car with the hood welded shut: You can run it, but it's very difficult to open it up and see how the program works or to fix it if it has a bug.

Finishing the dream. After more than 20 years of typing, Richard Stallman's hands have said "no more."

Stallman is afflicted with tendinitis. For more than a year, he's been unable to type without great pain. In order to finish his dream, he's been forced to hire typists to press the keys on his workstation as he dictates lines of C.

"Dee, dee, en, en, one, ar, control-ex, eye, notes . . ." says Stallman, standing behind Evan Ridel, an MIT undergraduate who is his hands today. Stallman dances and plays with his hair as he speaks, unable to control the nervous tension pent up in his head.

Typing for Stallman is a terrible job—the typist could just as well be typing in German or Italian. Anybody who can understand the C code on Stallman's screen can find a more interesting job than being a human robot. Stallman has a hard time finding typists for the 60 hours a week he wants to work. Ridel, one of Stallman's best typists ever, quit after a few months on the job.

Two companies in the Boston area have recently developed computer programs that reliably translate spoken words to text. "For those who can afford the $10,000 to $15,000 for a system, it basically changes their lives," says Christopher R. Seelbach, an analyst at Probe Research who follows the voice-processing industry. The big cost of such systems is software. Even though he can afford the dollars, Stallman refuses to use them. That's the problem, says Stallman. As computers get faster and cheaper, expensive software will keep them nevertheless from making a *real* difference in many people's lives. For that reason Stallman will only use a program that's not Free to write that program's replacement, and he doesn't have the time to write a speech-recognition system just yet.

Robert Chassell, the Free Software Foundation's treasurer, says that the high price of voice-recognition software is a warning of what may be in store for the future of all computer-based systems. "As computers become more powerful, restrictions on the software that runs them become more significant," he says. "It destroys the promise of the computer revolution."

By granting companies the right to use the copyright system to outlaw the duplication of their programs, the government has created monopolies. "It's very clear that a fair market price for software is quite low. You can discover this by finding out how much people charge for public-domain software or GNU Emacs: It turns out that the free-market cost of software is a very little bit above the incremental cost of reproducing the software. It's basically how much it costs to put the software onto the tape."

When asked about the investment of developing new programs, Chassell responds: "Even though the cost of developing a program *looks* impressive, it's actually very little compared to the costs of physical distribution and teaching people how to use the program. Society benefits when these are opened to the free market."

But the monopolies provided by copyrights on computer programs are a fact of American law. Stallman calls those who make use of them "software hoarders." He looks to a future where all computer companies follow the Free Software Foundation's model: low overhead; no advertising; paying their programmers a fair, living wage; and having all their software protected by the Copyleft. Far fewer programmers would be needed in such a world, says Stallman, because there would be far less duplication of effort.

"The problem with that is that it doesn't make any economic sense," says Tom Lemberg, vice president and chief counsel for Lotus Development Corporation in Cambridge. "It suggests a lack of belief in the market."

Lemberg sits on the sixth floor of a

plush office building with a view that looks over the Charles River. This building—and several others in the neighborhood—is an opulent display of just how much money one company has made through "software hoarding."

"It's nice to say that we should just sell support and give away the software, but why?" asks Lemberg rhetorically. "The way our economic system works is that people who create value are able to get value by selling it."

Stallman is sickened by such words: "I'm disappointed with people who strive to be personally successful. I think they should strive for something more important than that. It's okay to want to make your life comfortable, but you should want to make other people's lives comfortable, too."

Lemberg believes that even if Stallman and his cohorts may be successful in the short term, there's no long-term viability for Project GNU: "Is it possible that over time an industry will be sustained by people who do not want to be compensated for their efforts?" he asks rhetorically. "Probably not."

Writing software takes a tremendous amount of time and is an inherently risky proposition for most people, says Lemberg, because most programs don't sell well. Furthermore, he argues that the computer industry has undergone such explosive growth over the past ten years because companies have been innovative and have not tried simply to write copies of each other's programs.

"If you look at the people who have succeeded in this industry, they're people who have been radically different. The number-two-selling spreadsheet is not a clone of 1-2-3, it's [Microsoft] *Excel*. It has a radically different interface."

By many accounts, Stallman is one of the best programmers in the United States.

Lemberg even disagrees with Stallman's fundamental belief about computer programs—that it's easier to write a new computer program if you have the source code to a similar one.

"I've talked with developers," says the Lotus vice president. "If you want to copy somebody's program, you don't necessarily want to copy the code. You might be able to do it better yourself."

But there is no reason, says Stallman, that every single piece of a program should have to be re-invented every time a person writes a new application. That sort of needless duplication simply retards progress.

A holding pattern. Today there's a lot of free software. In addition to GNU Emacs and GCC, there's the GNU Debugger, a fast assembler, a suite of programmers' tools, *Ghostscript*, a version of the PostScript typesetting language, and the games *Chess* and *Go*. The Free Software Foundation also distributes the popular MIT X Window System, which is also free software, although it's not protected by the GNU Copyleft.

Since the middle of 1989, the GNU project has been in a holding pattern. The last part of the operating system that Stallman needs is "the kernel" — the master program at the center of the computer that oversees all other programs.

The idea of writing a kernel from scratch is daunting, even to a programmer of Stallman's prowess. A more economical approach is to build upon another university-developed kernel. But which one?

The kernel, says Stallman, must provide a functionality to most programmers that's nearly identical to Unix, or people won't use it. It must be modular, so that people can easily make changes to it. And, of course, it must be free.

At Carnegie-Mellon University, Professor Richard Rashid is overseeing the development of a such a kernel: Mach.

Mach is more than a rewrite of Unix: It's a fundamental redesign. Since the early 1970s, the size of the Unix kernel has exploded as more and more features have been added to it. Mach is an attempt to recapture the simplicity of Unix by removing all but the most critical features of the operating system from the kernel and placing them in small, single-function programs.

Although Mach has been operational for several years, part of it is still based on AT&T's Unix source code. The Free Software Foundation has been waiting for Rashid's group to replace that code with their own. That work was supposed to be finished by the fall of 1988, but Rashid's development schedule slipped.

"It's still slipping, but supposedly it's going to come out next month, or at least that's what they said in March," said Stallman in April. By October, Rashid had delivered a copy of Mach to Stallman, but had not given Stallman permission to redistribute it.

When it's operational, the GNU operating system will run on any 32-bit byte-addressable architecture, including the Motorola 68020 (inside Apple's Macintosh II computer) and the Intel 80386 (inside many computers that are IBM-PC compatible.)

If you ask Stallman "when will GNU be finished?" he's most likely to answer, "It will never be finished." Because the source code is available, Stallman believes that people will always be improving his work. If you ask him when GNU will be operational, his answer is much simpler:

"It will be finished quicker if you help." ◘

Simson L. Garfinkel is a doctoral candidate at the MIT Media Laboratory. Copyright 1990 by Simson L. Garfinkel.

Statement of Ownership, Management, and Circulation
(required by 39 U.S.C. 3685)

1. BCS Update. A. Publication No. 1041-4770
2. October 1, 1990
3. Monthly. A. Twelve. B. $40.00
4. One Center Plaza, Boston, MA 02108
5. Same as above
6. The names and addresses of the Publisher, Editor, and Managing Editor are: Publisher, Jonathan Rotenberg, same address as above; Editor, Mary E. McCann, same address as above; Managing Editor, none.
7. The owner is The Boston Computer Society, Inc., One Center Plaza, Boston, MA 02108
8. The known bondholders, mortgagees, and other security holders owning or holding 1 percent or more of the total amount of bonds, mortgages, or other securities are: None.
9. The purpose, function, and nonprofit status of this organization and the exempt status for Federal income tax purposes has not changed during the preceding 12 months.

10.	Average No. Copies Each Issue During Preceding 12 Months	Single Issue Nearest to Filing Date
A. Total No. Copies Printed (Net Press Run)	30,583	28,300
B. Paid Circulation		
1. Sales through dealers and carriers, street vendors, and counter sales	30	6
2. Mail subscription	29,457	27,304
C. Total Paid Circulation	29,487	27,310
D. Free distribution by mail, carrier or other means, samples, complimentary, and other free copies	350	400
E. Total Distribution (Sum of C and D)	29,837	27,710
F. Copies not distributed	746	590
1. Office use, left over, unaccounted, spoiled after printing		
2. Returns from news agents	0	0
G. Total (Sum of E and F—should equal net press run shown in A)	30,583	28,300

I certify that the statements made by me above are correct and complete.

Mary E. McCann, Editor