CHAPTER I

Free Software

*Richard Stallman**

A. INTRODUCTION

"Free software" means the software that respects the user's freedom, respects your freedom. Software that's not free is proprietary software; that is, software that tramples the user's freedom. Proprietary software keeps the users divided and helpless: divided because everyone is forbidden to share with anybody else, and helpless because the users don't have the source code, so they can't change anything. They can't even tell independently what the program is really doing to them.

But free software respects the user's freedom. What does that mean? There are four essential freedoms that the users of software should have. They are:

Freedom 0: The freedom to run the program as you wish;

Freedom 1: The freedom to study the source code of the program and change it to do what you wish;

Freedom 2: The freedom to help your neighbour. That's the freedom to redistribute copies of the program to others when you wish; and

* This is a revised transcript of the presentation given at the Knowledge Policy for the 21st Century Conference. Richard Stallman launched the development of the GNU operating system (see online: www.gnu.org) in 1984. GNU is free software: everyone has the freedom to copy it and redistribute it, as well as to make changes, either large or small. The GNU/Linux system (basically the GNU operating system with Linux added) is used on tens of millions of computers today.

> Freedom 3: The freedom to contribute to your community. That's the freedom to distribute copies of your modified versions when you wish

If you have all four of these freedoms, then the program is free software. That means that the social system of the program's distribution and use is an ethical system. If one of these freedoms is substantially missing, then the program is proprietary software which means that the social system of its distribution and use is unethical. That software's distribution is not a contribution to society; it's an attack on society.

Now this is not a question of what the program does; it's not a technical question. It's a social and ethical question. Any program could be distributed as proprietary software and then it's unethical. The same program could be distributed as free software and then, at least in regards to its distribution, it's ethical. Of course, there may be other ethical issues in the use of some particular program. All sorts of things. But they are different issues, so we have to think about them separately in the specific case that raises them.

B. FREEDOM TWO

But why define free software this way? What makes these four freedoms essential? Freedom number two—the freedom to help your neighbour, the freedom to distribute copies to others when you wish—is essential on basic moral grounds so that you can live an ethical upright life as a good member of your community. If you use a program that doesn't give you freedom two, then you're in danger of falling into a moral dilemma at any moment, whenever your friend says, "That program's nice, can I have a copy?"

At that moment you will have to choose between two evils. One evil is to give your friend a copy and violate the licence of the program. The other is to deny your friend a copy and comply with the licence of the program. Once you're in the dilemma, you ought to choose the lesser evil, which is to give your friend a copy and violate the licence of the program.

What makes this evil the lesser evil? Well, if you can't help doing wrong to somebody, it's better to do wrong to somebody who has behaved badly rather than to somebody who has behaved well. We can assume your friend is a good friend, a good member of your community. The reason we can assume this is: the other case is also possible, but it's easy. If a person who is nasty and not helpful asks you for your cooperation, you can simply say, "Why should I help you?" So that means the hard case is where it is a good person, who normally deserves your cooperation.

By contrast, the developer of the proprietary program has deliberately attacked the social solidarity of your community, so he is culpable. And therefore, if you've got to wrong one or the other, pick him. However, to be the lesser evil does not mean it is good. It is never good to make an agreement and break it. Some agreements are inherently evil, and it is better to break them than to keep them. But breaking them is still not good. Additionally, if you do give your friend a copy, what will your friend have? Your friend will have an unauthorized copy of a proprietary program, and that's a pretty bad thing. It's almost as bad as an authorized copy.

So, what you should really do, once you've understood this dilemma, is to make sure you are never in it. There are two ways to do that. One is, don't have any friends. That's the method recommended by the proprietary software developers.

The other method is, don't use proprietary software. If you don't use a program that doesn't give you freedom two then you'll never be in this dilemma. That's the solution I have chosen. If somebody offers me a program on the condition that I promise not to share it with you, I will reject it. No matter how attractive and useful that program will be for me, I will say, "My conscience does not allow me to accept that program, so take it away."

So that's the reason for freedom number two, the freedom to help your neighbour, the freedom to distribute copies to others when you wish. I should emphasize that each of these four freedoms is the freedom to do something if you wish, as you wish. It's not an obligation. You're never required to run the program in any particular way. You're never required to study and change the source code. You're never required to redistribute copies. And, if you've made modified versions, you're never required to publish them. But the point is that you should be free to do that if you wish. And so, I've explained the reason for freedom two.

C. FREEDOM ZERO AND FREEDOM ONE

Freedom zero is essential for a different reason: so you can have control of your computer. There are proprietary programs that actually restrict how people can use authorized copies, and that's obviously not having control of your own computing. So freedom zero is essential.

But it's not enough, because that's just the freedom to either do or not do whatever the developer has already decided and programmed into it. The developer decides what you can do, not through a licence, if you have freedom zero, but rather through deciding what's in the code. So you're still under the developer's power. Therefore, we need freedom one: the freedom to study the source code and then change it to do what you want. With this freedom you decide, instead of having the developer decide for you.

If you don't have this freedom, then you can't even check what the program is doing.

Many non-free programs contain malicious features designed to spy on the user, restrict the user, and even attack the user. For instance, spyware is common. One proprietary program that you may have heard of that spies on the user is called Microsoft Windows.[1] When the user of Windows—and I won't say "you," because I'm sure you wouldn't use a program like this—uses the feature to search throughout his files for a word, Windows sends a message saying what was searched for.

That's one spy feature, but there's another: when Windows XP asks for an upgrade, it sends a list of all programs installed on the machine. That's another spy feature. But Microsoft didn't announce these spy features. People had to figure them out, and it wasn't easy. For instance, the list of installed programs is sent encrypted. So it was hard work to figure out what information is actually in that message. There may be other spy features that we don't know about yet. It's always possible for there to be more.

But spying is not limited to Windows. Windows Media Player also spies on the user. It reports everything that the user looks at: total surveillance. But please don't think that this is because Microsoft is somehow uniquely evil. Spyware is common. For instance, Real Player spies on the user the same way Windows Media Player does.

Spyware is nasty enough, but it gets worse. There is also the functionality of refusing to function; where a program says, "I don't want to let you look at this file. I don't want to let you copy part of this file. I don't want to print this file for you—because I don't like you."

I'm not talking about "bugs" here. These are deliberately implemented features, designed to stop you from doing the natural things you might want to do with the files on your computer. They're also known as DRM, or Digital Restrictions Management. They are deliberately implemented functionality of refusing to function.

Now, one of the reasons why Apple switched to the Intel computer architecture is so that it could implement more powerful Digital Restrictions Management.[2] And the main advance in Windows Vista is not an

1 Michael Horowitz, "Windows is Spyware" (13 September 2007), online: CNET News, Defensive Computing, news.cnet.com/8301-13554_3-9778389-33.html.

2 JD Lasica, "Apple's Move to Intel: A DRM Gambit?" (8 June 2005), online: Darknet www.darknet.com/2005/06/apples_move_to_.html.

advance in serving the user, it's an advance in restricting the user. Take a look at badvista.org for more information about why people should avoid migrating to Vista even if they are Windows users. And to take a look more generally at our campaign against Digital Restrictions Management, look at defectivebydesign.org.

But malicious features can get worse than that. There's also the malicious features designed to attack the user: back doors. One proprietary program that you may have heard of that contains a back door is called Microsoft Windows.

You see, when Windows asks for an upgrade, Microsoft more or less knows the user's identity and therefore Microsoft can deliver to that user an upgrade designed specifically for him. In other words, Microsoft can take total control of his computer.

Well, this is the back door we can deduce from known facts. But it's not the only one that has been attempted. A few years ago in India, I was told they had arrested several programmers working on developing Microsoft Windows and accused them of working simultaneously for Al-Qaeda trying to install a back door that Microsoft wasn't supposed to know about. This attempt apparently failed. We don't know if another group tried the same thing and succeeded. But we do know that in 1999, Microsoft was caught having installed a back door for the United States government. Specifically, for the use of the National Security Agency.[3]

So, what this shows us is that you can't trust a proprietary program. You can divide proprietary programs into two big categories: those in which we have discovered malicious features, and those in which we have not discovered any. Those may not have any, or they may have some that we haven't discovered. There is no way we can be sure that any program does not have malicious features. The only way we can ever be sure of the answer is when we find some. The result is that you can never trust a non-free program, and yet every non-free program demands blind faith. They're all "just trust me" programs, demanding of a faith that they do not — by virtue of their own attacks on your freedom — that they *cannot* ever justify.

However, let's consider those non-free programs which do not have malicious features. We can be sure that they exist even though we can't be sure which ones they are. Their developers did not decide to abuse their power in this particular way. But they are human. They make mistakes. Their code is full of errors. And a user of a program without freedom number one is

3 "Crypto Expert: Microsoft Products Leave Door Open to NSA" (3 September 1999), online: CNN www.cnn.com/TECH/computing/9909/03/windows.nsa/.

just as helpless facing an accidental error as she is facing a deliberate malicious feature. If you use a program without freedom number one, you're a prisoner of your software.

We free software developers are also human, so we also make mistakes and our code is also full of errors (although there are some arguments that on the average it has fewer of them). But if you come across an error in our program that bothers you, you can fix it, or pay someone else to fix it. And that's the difference. We can't make ourselves perfect but we can respect your freedom. We, the free-software developers, are the ones who have chosen not to keep you helpless, to respect your freedom instead.

D. FREEDOM THREE

But freedom number one is not enough. Because that's the freedom to [personally] study and change the source code. That's not enough because there are millions of users who don't know how to program and they are not in a position to, they don't know how to exercise freedom number one. But even for programmers like me it's not enough. And the reason is, we're all busy and there's just too much software. There's too much free software for any one person to study all of the software that she uses and master it all and make all the changes that she might want. This is more than one person could do.

So, in order to fully take control of our computing, we need to do it working together, cooperating. And for that we need freedom three: the freedom to distribute copies of the modified version, the freedom to contribute to your community. With this freedom we can work together. Because it only takes one person to make any given change and publish the modified version, for everyone else to get to use it without having to write.

So if there are a million users of a free program who want some change, we can expect that there will be a few thousand who know how to program. And one day a few of those people will do it — make that change, publish their modified version — and then all of those users can switch to it. So they all get what they wanted and almost all of them didn't have to do the work. And that includes the ones who don't know how to program. So the result is that all the users get the benefit of the four freedoms.

E. THE FOUR FREEDOMS AND DEMOCRACY

Every user can exercise freedom zero and two, the freedom to run the program as you wish, and to distribute copies to others when you wish. These don't require programming. But any given user can only exercise freedoms one and three — the freedom to study and change the source code, and the freedom to distribute the modified versions — to the extent that he knows how to program.

Now this is not a zero-or-one question. Many people who don't learn enough to be professional programmers learn some programming, and they can make small changes which can be useful. Just as many people who don't know how to be professional car mechanics learn to do some maintenance on their cars and that's useful also. But in any case, any given person can exercise these two freedoms to their extents. But when programmers do exercise them and publish their modified versions, everyone else gets to use their results, so we all get the benefit. And together these four freedoms give us democracy. Because a free program develops democratically under the control of its users.

Every user, individually, can control what his copy is going to do if he wants to take the trouble to do it himself or hire someone else to do it. And together, collectively, the users make the decisions about what the program will do. The users are in control, so the users generally get what they want, because nobody is in a position to impose on the users something they don't want: malicious features such as spying on the user, restricting the user, or attacking the user. Or even mere incompatibility, which some proprietary software developers impose on the user because they think the user is captive, and that by making their software incompatible they can make sure the users never switch to anything else.

Now, one interesting thing about free software is that it brings with it a free market for all kinds of support and service. By contrast, the support of proprietary programs is strictly a monopoly. You see, only the developer has the source code, so only the developer can make a change. If the user wants a change, the user has to beg and pray, "Oh almighty developer, please make this change for me." Sometimes a developer says, "Pay us and we'll listen to your problem." If the user does, the developer says, "In six months there will be an upgrade. Buy the upgrade and you'll see if we've fixed your problem and you'll see what new problems we have in store for you." But with free software, anyone who has a copy can study the source code, master it, and begin offering support. So, free software support is a competitive market.

This is interesting because many people who are in favour of proprietary software, and who like to mislead the public, say that we're Communist. And yet they are the ones who operate a command-based, monopoly-based economy, where they just give the orders and all the users have to do what they say or escape.

And they don't respect private property, either. You see, if you have a copy of some proprietary program, typically it comes with an end user licence which says that that copy's not your property. All you get to do is use it under licence if you obey whatever orders they want to give you. So, they are the ones who should be compared to a Stalinist system, whereas with free software, your copy really is yours. It's your property and you can do all the things with it that contribute to society.

Now, the competitive market for free software [support] means . . . all businesses and agencies think that competition is desirable when they buy things. They want to have competition and they value good support for the software they use. They ought to be, if they are rational, running as fast as they can towards free software to make sure that they can buy their support through a competitive market and therefore that they can expect to get good support for their money.

And it's important to know that merely having a choice between proprietary software products does not give the same result. Usually, when there is a choice between two products to do a job, we say there is no monopoly. But when it's a choice between proprietary software products, that's still monopoly. Because if the user chooses this proprietary program then he falls backward into this monopoly for support. But if he uses this other proprietary program then he falls into this other proprietary monopoly for support. So it's a choice between monopolies, but no way to escape from monopolies. The only way to escape from monopoly is to escape from proprietary software.

And that is what you should do if you value your freedom. If you use a non-free program, then it has taken away your freedom. It has put a chain on you. If you want to live in freedom and use computers, it is necessary to reject non-free software and use only free software. And that's the goal of the free software movement: to help you and all users escape from non-free software. Escape to the free world.

And as you can see, this reference to the cold war is entirely intentional.

We have built a new world in cyberspace, because it is impossible to be free in the old world of cyberspace where every computer had its lord who dominates the users and pushes them around. So if you want to be free, you have to escape, you have to come to the new world, which is a new continent that we have built in cyberspace. Because it is a virtual continent, it has room for everyone. And because there are no indigenous people in cyberspace—and the closest thing to an indigenous people there ever was, was the community of free software that I belonged to in the seventies—we didn't displace any indigenous peoples, and everybody is entitled, everyone is welcome to be in the new world; everyone is welcome to escape and join us. And that's what you have got to do. And that's what all computer users have got to do.

To have a choice between proprietary programs is being able to choose your master. Freedom means not having a master. And when you use software, freedom means not using proprietary programs.

F. THE HISTORICAL CONTEXT

I reached these ethical conclusions more or less—of course not in their current form—in the year 1983. I had spent the 1970s as part of a community of programmers who shared software. I used entirely free software, with occasional exceptions which showed me how evil and ugly proprietary software was. And so when my community died, I was in a position to make the moral contrast and recognize that the proprietary software way of life was an evil way of life. And I decided I wanted to use computers in freedom and make it possible for others to do so. But how?

I didn't think I'd be able to convince governments to change their laws, or companies to change their practices, because I was just one man with no particular political experience or talents and no particular fame except as the developer of the editor Emacs. So I didn't try that road. But I came up with another method, a way I could solve this social problem through technical work. I just had to develop an operating system and make it free. And that would give everybody a way to use computers in freedom.

You see, the computer won't run without an operating system, and all the operating systems in 1983 were proprietary. So it was impossible to use computers in freedom. But if I developed a free operating system, that would make it possible.

So I was aware of a social problem that was important in my field and that was growing because the use of computers was growing. And most people didn't recognize it as a problem. I had the skills necessary to try to solve this problem and chances are, nobody was going to solve it if I did not. That meant that I had been elected by circumstances to solve this problem.

Just as if you see somebody drowning, and you know how to swim, and no one else is around, then you have a moral duty to save that person.

So, I decided to make the system Unix compatible, so that it would be portable and so that it would be easy for Unix users to migrate. And I gave it the name GNU which is a recursive acronym for GNU's Not Unix. It's a humorous way of giving credit. And then in January 1984, I quit my job at MIT to launch the development of this system.

By the early 1990s, the GNU system was almost finished, but one important component was missing. That was the kernel, which is the program that allocates the computer's resources to all of the other programs you run. And then somebody else developed a kernel named Linux. And then in 1992, he made it free software by changing the licence and adopting the GNU General Public Licence (or GNU GPL), which is the licence I had written to use on the components of the GNU system. And at that point, the combination of the almost complete GNU system plus Linux made a complete pre-operating system: the "GNU plus Linux" system. And thus for the first time, it was possible to use a PC without giving up your freedom to proprietary software developers. And the goal that we set out to achieve when I founded the free software movement in 1983 had been reached.

The development of Linux was the last step, the step that carried us across the finish line. It was able to do so because of all of the steps that we had taken to get so close. So I'd like to ask you to please give us an equal mention. Give us a share of the credit for the work that we started and that we are still the biggest contributor to. Please call the system "GNU plus Linux." You'll find many people call it Linux which means they are giving all the credit to Linus Torvalds, and worst of all they think that the system flowed from *his* vision and *his* ideals. But in fact, he doesn't have ideals that are relevant to this. He describes himself as "apolitical," and his vision is an engineer's vision. He says that he wants powerful, reliable software. That is a viewpoint that today goes by the name "open source."

G. FREE SOFTWARE v OPEN SOURCE

Open source is a way that people who want to talk about free software—but cover up the ethical issues that are the base of the free software movement—talk about the software that is free and some other software which is almost free. They think it's acceptable; their criteria are different. So that's why I don't use the term "open source," that's why I haven't mentioned it, because that's not what I'm interested in, what I'm in favour of. I'm in favour of freedom of the users. Freedom for everyone who uses computers, whereas *they* say they are aiming only for powerful, reliable software.

Open source actually refers to a development methodology. They say that if you let everybody have more or less the same four freedoms, then that will make software more powerful and reliable. Well, I think that's a nice bonus, and if you're going to develop good software it's wise to take a look at good and bad software development methodologies. But that's the secondary issue when compared with freedom and social solidarity. That is to say, when compared with ethics and politics.

So open source misses the main point, and not by accident. That term was promoted with a rhetoric designed to miss that point, and that's why if you talk about open source, you're helping other people miss that point. So I never describe my work that way.

People motivated by those values have made contributions to our community. If you develop a program and it's free, well, your actions are good regardless of what your motives might be. But nonetheless, it's important to teach the rest of society about freedom and social solidarity. And it's important to teach the users of this operating system that it was developed for the sake of those ideals. Because if we want to keep our freedom, we have to value our freedom. And most people in our community have never even heard of these freedoms.

H. HUMAN RIGHTS AND SOFTWARE

Now, in other areas of life, people have been talking about human rights for centuries, so there have been centuries for people to address the questions of which human rights people are entitled to, and centuries to spread these ideas around the world. That doesn't mean it's easy to defend these human rights. The US has abolished the right to a trial following norms of justice before someone is imprisoned: a fundamental right. The United States has abolished freedom of association by administratively declaring organizations to be "terrorist." And many other freedoms are under great pressure in the US. Freedom of speech has not been officially abolished, but people have been put under a lot of pressure for criticizing the government.

And other countries that we think of as bastions of freedom have done similar things. In the UK, it is now a crime to be suspect. Literally. The offense consists of having possessed an object which gives rise to a suspicion which is judged to be reasonable. So they have adopted, officially in their laws, the practice of imprisoning people on suspicion. And this is not even to count the imprisonment without trial which fortunately in the UK was banned by judges. So defending freedom can be hard, even when these freedoms are understood.

But computing is a new area of life and there hasn't been much time for having a debate about which human rights belong to the person who is using software. And to the extent that there has been a debate, it has been mostly dominated by the proprietary software companies. They dominated it in two ways.

First of all, they set up a status quo so they get the advantage that they're not calling for a change. They're just saying that what everybody is used to, or just about everybody is used to, is right. And secondly, they have a lot of money, and they can use that for influence in getting the support of a lot of influential people and institutions. And even in the community of users of free software, most have never heard of the ideas I've told you because they've only heard the "open source" ideas. If we are to establish a community of freedom and make it last, the most important thing to do is to teach the people who are using free software to value the freedoms *as freedoms*, so they'll be ready to defend these freedoms.

I. LEGAL PROHIBITION AND FREE PROGRAMS

Freedoms are frequently threatened. The US has two different laws that can prohibit free software, and many other countries have adopted them also under US influence and pressure. One of these two laws in the US is the *Digital Millennium Copyright Act*,[4] which has been used to censor the free software for playing a DVD. You see, the movie on a DVD is often in encrypted format. The encryption format was secret so that all the authorized DVD players restricted the user. (They imposed digital restrictions management.) Then somebody figured out the format and wrote a free program to play the movie on a DVD. And then they published a free program to do the job.

Well, that free program is censored in the US. Even making a link to a foreign site that distributes it is illegal in the US. So how are we going to serve the public when it's illegal to serve the public? This law expresses—takes to the extreme—the corruption of democracy, which replaces government of the people, by the people, for the people, with government of the people, by the cronies, for the corporations. This law says that the people exist to be the market or the prey of business. And that for people to help each other is illegal.

But this law can only prohibit programs in a particular narrow range of jobs, that is, access to encrypted works. By contrast, patent law can prohibit any kind of free program no matter what job it does. Of course it can also prohibit proprietary programs too. In fact, any software developer is in danger of getting sued, probably dozens or hundreds of times, because in

4 *Digital Millennium Copyright Act of 1998*, Pub. L. No. 105–34.

the US, any software idea, any method, technique, algorithm, feature, even some aspect of a feature, some aspect of a technique can be patented. And in a large program, there are thousands of these. So a large program is likely to be covered by hundreds of different patents at once.

One study was done, about three years ago, by a lawyer who checked one particular large component—namely Linux, which is the kernel of the GNU/Linux system—and he found 283 different US software patents, each of which prohibited some computation done somewhere in Linux. Around the same time, I read an estimate that Linux was 0.25 percent of the entire system. So by multiplication, we can estimate that there are some 100,000 software patents prohibiting something somewhere in the system (rough estimate). That shows you how bad the patent system is. If you look at it as a distribution of GNU/Linux, you could expect around 100,000 patents covering that one CD-ROM worth of software. So we now face political battles just to be free to continue serving the public.

J. GOVERNMENT AND FREE SOFTWARE

Meanwhile, one secondary question is why government agencies should use free software. Now, there is a general reason: because the government's mission is to assure freedom and a good society for its citizens. Therefore, in regards to computer use, governments should try to direct society away, out of the propriety software path, and onto the free software path. Now this takes some effort, because there is inertia which keeps society, at least most of it, going down the proprietary software path. Society is essentially stuck in a rut. And the proprietary software developers know this, so every year they make the rut a little deeper, figuring that the more society uses computers, the more they can get away with squeezing society; thus, the deeper the rut, the more money they can make.

So, governments should make the investment of pulling society out of this path and onto the path that leads to freedom. Every government agency has its own particular mission, but all government agencies should participate in carrying out the overall mission for society of defending freedom, and not let their own narrow, specific missions blind them to the overall purpose of the government.

However, there is also a more direct reason. You see, a government agency does its computing on behalf of the public and it has a responsibility to assure its control over its own computing. It is wrong, it is a dereliction, for any government agency to allow computing to fall into private hands—any private hands. However, using a non-free program is doing exactly that.

It is allowing the agency's computing to fall into the hands of the software's developer. Thus, using proprietary software is a dereliction on the part of the agency.

When agencies migrate to free software, they not only regain their sovereign control over the work they do on behalf of the public, they also encourage free software use among private entities. For instance, these agencies want support, so when they use free software they then get the benefit of the free market of support so they can decide who to hire. But the point is, they are hiring somebody, so they are contributing to the growth of the free market that free software support. And the bigger that market is, the easier it is for private companies and NGOs to buy support for free software, the more choices they have. So that makes it easier for them to migrate as well.

K. GOVERNMENT SCHOOLS AND FREE SOFTWARE

Now, I've been talking about government agencies. One kind of government agency which you might not think of is actually the most important of all, and that is the school, the government-supported school, which, like all schools, should teach exclusively with free-software. There are four reasons for this.

The first reason is to save money. Schools in all countries are limited by their budgets. There are things they want to do but can't do, so they should not be wasting money paying for permission to run proprietary software. That reason is obvious and superficial, and some proprietary software companies get rid of it by donating *gratis* copies of their non-free software to the schools. And why do they do that? Is it because they are idealistic and want to promote education?

I don't think so. Rather, they want to use the school as an instrument to impose dependency on their software onto society. The idea is that the students in the school will learn to use that software and become dependent on it just as if they had been given an addictive drug. And then after they graduate, you can be sure that the same company is not going to donate *gratis* copies to them nor to the companies that they work for. And the result is that the schools push all of society into the path of dependency and helplessness and division which is the path of using proprietary software.

This is counter to the schools' mission in society which is to prepare people to be citizens of a free society in which people treat each other well. Therefore schools should refuse these poisonous gifts; actually you might say addictive gifts. For the school to teach the students dependency on the software is like teaching the students to be addicts to an addictive drug. And of course, the companies that make the drug would be glad to donate the first dose. The first dose is *gratis*. Afterwards, when you're hooked, you've got to pay.

Schools should refuse to participate in this. Schools should refuse to teach non-free software. They should teach exclusively free software. That's the second reason.

But there is an even deeper reason for the sake of computing, of computer science education. You see, at the age of thirteen, some people who are natural born programmers want to learn everything about the software, the computer and the system. So if they are using a program, they want to know how it works. But when the student says, "Teacher, how does this work?", if it's proprietary, the teacher can only say, "I'm sorry, it's a secret." So education cannot begin. But if the program is free, the teacher can explain what he knows and then say, "Here's the source code. Read it and you'll understand everything." And that student will read it because she wants to understand everything, she's fascinated by software. And this is the way these born programmers can fully realize their skills.

You see, they're supposed to teach them to program, because to them programming is obvious. But learning to program "well" is something else. You learn to program well by reading lots of code and writing lots of code. You can only do that with free software. First you read the program, and if there is anything you don't understand, you ask the teacher or an older student, "What does this do?" and from that experience you learn something very important: that is the wrong way to write it because it's not clear enough. You have to see lots of ways of doing things wrong to learn not to do them. Then, you can try something else that's useful for learning, which is, to write a small change in the big program you just read. This is before you are at the stage where you could do a good job of writing a large, useful program yourself, but you could try writing a small change in an existing large program. That's a small job.

So you could write one change, and another, and then change another big program. And after writing thousands of changes in large programs, you get very good at writing changes in large programs. This is the big job of the software field: writing changes into existing large programs. And now and then, of course, we write a new large program. But once you've written lots of changes, you could probably also write a new program because you've come to understand the right way to write each piece.

In 1971, I had to go to the artificial intelligence lab (at MIT) to have an opportunity to learn this way and realize my skill. Today, any school can provide this opportunity, but only with free software, not with proprietary software.

There's even a deeper reason: for the sake of moral education. You see, schools are supposed to teach not just facts and not just skills and techniques, but above all, how to be a good citizen: the spirit of goodwill, the habit of helping your neighbour. So, every class should have a rule: "students, if you bring a program to class, you can't keep it for yourself. You must share it with the rest of the class. If you won't share it, you can't bring it." However, the school has to follow its own rule to set a good example. So the school must only bring free software to class. This applies to all levels from elementary school to the university. The university you are in must use exclusively free software. And if that's not the case already, you ought to be working to make it so.

L. CONCLUSION

I'd like to urge you to look at the website for gnu.org for more information about the GNU system and free software movement, and please look at fsf.org for information about the Free Software Foundation and to join. We need your support.

Governments above all should call for free software because protecting the public's freedom is the mission of the government. If we have governments that don't want to talk about protecting the freedom of their citizens, we're in deep trouble.

As well, the universities ought to listen to you and start teaching students how [to work on free software]. And the best way is to set up programs where students who are interested in programming, work through the whole period of time that they are in the university on free software projects.