Free Software Is Even More Important Now

by **Richard Stallman**

A substantially edited version of this article was published in Wired.

It is 30 years since the launch of the Free Software Movement which campaigns for software to respect the users' freedom and community. We call such software “free” and “libre” (we use that word to emphasize that we're talking about freedom, not price). Some proprietary programs, such as Photoshop, are very expensive; others, such as Flash Player, are available gratis—either way, they subject their users to someone else's power.

Much has changed since the beginning: most people, in advanced countries, now own computers (sometimes called “phones”) and connect to the Internet with them. Nonfree software still makes the users surrender control over their computing to someone else, but now there is another way to lose it: Service as a Software Substitute, or SaaSS, which means letting someone else's server do your own computing activities.

Both nonfree software and SaaSS can spy on the user, shackle the user, and even attack the user. Malware is common in services and proprietary software products because the users don't have control over them. That's the fundamental issue: while nonfree software and SaaSS are controlled by some other entity (typically a corporation or a state), free software is controlled by its users.

Why does this control matter? Because freedom means having control over your own life. If you use a program to carry out activities in your life, your freedom depends on your having control over the program. You deserve to have control over the programs you use, and all the more so when you use them for something important in your life.

Your control over the program requires four essential freedoms If any of them is missing or inadequate, the program is proprietary (or “nonfree”).

(0) The freedom to run the program as you wish, for whatever purpose.

(1) The freedom to study the program's “source code”, and change it, so the program does your computing as you wish. Programs are written by programmers in a programming language—like English combined with algebra—and that form of the program is the “source code”. Anyone who knows programming, and has the program in source code form, can read the source code, understand its functioning, and change it too. When all you get is the executable form, a series of numbers that are efficient for the computer to run but extremely hard for a human being to understand, understanding and changing the program in that form are forbiddingly hard.

(2) The freedom to make and distribute exact copies when you wish. (It is not an obligation; doing this is your choice. If the program is free, that doesn't mean someone has an obligation to offer you a copy, or that you have an obligation to offer him a copy. Distributing a program to users without freedom mistreats them; however, choosing not to distribute the program—using it privately—does not mistreat anyone.)

(3) The freedom to make and distribute copies of your modified versions, when you wish.

The first two freedoms mean each user has individual control over the program. With the other two freedoms, any group of users can together exercise *collective control* over the program. The result is

that the users control the program.

Other kinds of works are also used for practical activities, including recipes for cooking, educational works such as textbooks, reference works such as dictionaries and encyclopedias, fonts for displaying paragraphs of text, circuit diagrams for hardware for people to build, and patterns for making useful (not merely decorative) objects with a 3D printer. Since these are not software, the free software movement strictly speaking doesn't cover them; but the same reasoning applies and leads to the same conclusion: these works should carry the four freedoms.

A free program allows you to tinker with it to make it do what you want (or cease do to something you dislike). Tinkering with software may sound ridiculous if you are accustomed to proprietary software as a sealed box, but in the Free World it's a common thing to do, and a good way to learn programming. Even the traditional American pastime of tinkering with cars is obstructed because cars now contain nonfree software.

The Injustice of Proprietariness

If the users don't control the program, the program controls the users. With proprietary software, there is always some entity, the “owner” of the program, that controls the program—and through it, exercises power over its users. A nonfree program is a yoke, an instrument of unjust power.

In extreme cases (though this extreme has become widespread) proprietary programs are designed to spy on the users, restrict them, censor them, and abuse them. For instance, the operating system of Apple iThings does all of these, and so does Windows on mobile devices with ARM chips. Windows, mobile phone firmware, and Google Chrome for Windows include a universal back door that allows some company to change the program remotely without asking permission. The Amazon Kindle has a back door that can erase books.

With the goal of ending the injustice of nonfree software, the free software movement develops free programs so users can free themselves. We began in 1984 by developing the free operating system GNU. Today, millions of computers run GNU, mainly in the GNU/Linux combination.

Distributing a program to users without freedom mistreats those users; however, choosing not to distribute the program does not mistreat anyone. If you write a program and use it privately, that does no wrong to others. (You do miss an opportunity to do good, but that's not the same as doing wrong.) Thus, when we say all software must be free, we mean that every copy must come with the four freedoms, but we don't mean that someone has an obligation to offer you a copy.

Nonfree Software and SaaSS

Nonfree software was the first way for companies to take control of people's computing. Nowadays, there is another way, called Service as a Software Substitute, or SaaSS. That means letting someone else's server do your own computing tasks.

SaaSS doesn't mean the programs on the server are nonfree (though they often are). Rather, using SaaSS causes the same injustices as using a nonfree program: they are two paths to the same bad place. Take the example of a SaaSS translation service: The user sends text to the server, and the server translates it (from English to Spanish, say) and sends the translation back to the user. Now the job of translating is under the control of the server operator rather than the user.

If you use SaaSS, the server operator controls your computing. It requires entrusting all the pertinent data to the server operator, which will be forced to show it to the state as well—who does that server

really serve, after all?

Primary And Secondary Injustices

When you use proprietary programs or SaaSS, first of all you do wrong to yourself, because it gives some entity unjust power over you. For your own sake, you should escape. It also wrongs others if you make a promise not to share. It is evil to keep such a promise, and a lesser evil to break it; to be truly upright, you should not make the promise at all.

There are cases where using nonfree software puts pressure directly on others to do likewise. Skype is a clear example: when one person uses the nonfree Skype client software, it requires another person to use that software too—thus surrendering their freedoms along with yours. (Google Hangouts have the same problem.) It is wrong to make such a suggestion. We should refuse to use such programs even briefly, even on someone else's computer.

Another harm of using nonfree programs and SaaSS is that it rewards the perpetrator, encouraging further development of that program or “service”, leading in turn to even more people falling under the company's thumb.

All the forms of indirect harm are magnified when the user is a public entity or a school.

Free Software and the State

Public agencies exist for the people, not for themselves. When they do computing, they do it for the people. They have a duty to maintain full control over that computing so that they can assure it is done properly for the people. (This constitutes the computational sovereignty of the state.) They must never allow control over the state's computing to fall into private hands.

To maintain control of the people's computing, public agencies must not do it with proprietary software (software under the control of an entity other than the state). And they must not entrust it to a service programmed and run by an entity other than the state, since this would be SaaSS.

Proprietary software has no security at all in one crucial case—against its developer. And the developer may help others attack. Microsoft shows Windows bugs to the US government digital spying agency, the NSA, before fixing them. (See http://arstechnica.com/security/2013/06/nsa-gets-early-access-to-zero-day-data-from-microsoft-others/). We do not know whether Apple does likewise, but it is under the same government pressure as Microsoft.

Thus, if the government of any other country uses such software, it endangers national security. Do you want the NSA to break into your government's computers? See http://www.gnu.org/philosophy/government-free-software.html for our suggested policies for governments to promote free software.

Free Software and Education

Schools (and this includes all educational activities) influence the future of society through what they teach. They should teach exclusively free software, so as to use their influence for the good. To teach a proprietary program is to implant dependence, which goes against the mission of education. By training in use of free software, schools will direct society's future towards freedom, and help talented programmers master the craft.

They will also teach students the habit of cooperating, helping other people. Each class should have

this rule: “Students, this class is a place where we share our knowledge. If you bring software to class, you may not keep it for yourself. Rather, you must share copies with the rest of the class—including the program's source code, in case someone else wants to learn. Therefore, bringing proprietary software to class is not permitted except to reverse engineer it.”

Proprietary developers would have us punish students who are good enough at heart to share software and thwart those curious enough to want to change it. This means a bad education. See http://www.gnu.org/education/ for more discussion of the use of free software in schools.

Free Software: More Than “Advantages”

I'm often asked to describe the “advantages” of free software. But the word “advantages” is too weak when it comes to freedom. Life without freedom is oppression, and that applies to computing as well as every other activity in our lives. We must refuse to give the owners of the programs or computing services control over the computing we do. This is the right thing to do, for selfish reasons; but not solely for selfish reasons.

Freedom includes the freedom to cooperate with others. Denying people that freedom means keeping them divided, which is the start of a scheme to oppress them. In the free software community, we are very much aware of the importance of the freedom to cooperate because our work consists of organized cooperation. If your friend comes to visit and sees you use a program, she might ask for a copy. A program which stops you from redistributing it, or says you're “not supposed to”, is antisocial.

In computing, cooperation includes redistributing exact copies of a program to other users. It also includes distributing your changed versions to them. Free software encourages these forms of cooperation, while proprietary software forbids them. It forbids redistribution of copies, and by denying users the source code, it blocks them from making changes. SaaSS has the same effects: if your computing is done over the web in someone else's server, by someone else's copy of a program, you can't see it or touch the software that does your computing, so you can't redistribute it or change it.

Conclusion

We deserve to have control of our own computing; how can we win this control? By rejecting nonfree software on the computers we own or regularly use, and rejecting SaaSS. By developing free software (for those of us who are programmers) By refusing to develop or promote nonfree software or SaaSS. By spreading these ideas to others.

We and thousands of users have done this since 1984, which is how we now have the free GNU/Linux operating system that anyone—programmer or not—can use. Join our cause, as a programmer or an activist. Let's make all computer users free.