FREE SOFTWARE FOUNDATION

Bulletin

Issue 29
Fall/Winter 2016

Contents

Free software takes root in the White House gardens

By John Sullivan
Executive Director

A very exciting thing happened this year: the FSF had a positive impact on US government policy at the highest level. We did not get anything close to a total victory, but we did help get free software blooming where it has not bloomed before.

On August 8, the White House Office of Management and Budget (OMB) published a Federal Source Code Policy at `sourcecode.cio.gov`. The FSF influenced this policy in two ways. First, we were consulted earlier in the year to answer questions about how free software licenses work and what kind of policy we thought would be best. Second, when a draft version of that policy was published for public comment, we participated, and rallied others to do so.[1]

STOP EME
STOP EME
NO DRM IN HTML
W3C: reject Encrypted Media Extensions and safeguard a Web that puts users first.
NO DRM IN HTML
NO DRM IN HTML!
SAY NO TO <DRM> IN HTML5
FISH & CHIPS
No DRM in HTML!
Even dogs understand that this would be a bad thing.

Selfies against DRM in Web Standards: Visit u.fsf.org/1yp.

After the public comment period, the OMB updated and adopted the policy. The policy now:

> ...establishes a pilot program that requires agencies, when commissioning

[1] u.fsf.org/1z2

> new custom software, to release at least 20 percent of new custom-developed code as Open Source Software (OSS) for three years, and collect additional data concerning new custom software to inform metrics to gauge the performance of this pilot.

It focuses on values like reducing cost, avoiding lock-in, and reducing inefficiency.

In our comment, we urged the OMB to require *all* covered software be released as free software, we emphasized the importance of also requiring associated documentation to be freely licensed, and we advocated for inclusion of the Free Software Definition in order to foreground ethical values.[2] It is certainly worthwhile to reduce inefficiency in government, but the reasons governments should use free software are much bigger than that: preserving their own autonomy and protecting the freedom of their citizens.

Unfortunately, the policy took some steps backward after the public comment period. Where the draft version required that all code written by federal agencies be released to the public, and 20% of the code written by contractors, the final version lowered the agency employee requirement to be the same as contractors. While the policy does require documentation to be provided along with code, it does not require that this documentation be provided under a free license.

Despite these setbacks, the published policy is a sizable step forward, and is heartening to see. And, they did adopt one critical aspect of our comment: the Free Software Definition. This gives us something to build on, and we should continue to push for substantial improvements.

We are happy that the Free Software Definition was added as a reference, but it should be much more prominent than that. It is ethical values that should be the basis for the policy, rather than the secondary benefit of efficiencies in software sharing between agencies.

The policy should also not rely so much on Github. We do commend the White House for accepting comments on the draft policy via email, and not requiring the proprietary JavaScript used on `regulations.gov`, but this same philosophy needs to extend to implementation of the policy, so that citizens are not required or strongly steered to use a single company's site to participate in government code projects, especially not one with a number of problems when it comes to free software values.[3]

We will need to work not just to improve the policy, but to ensure its future. A new boss will move into the White House in January, and the policy says that the results of its three-year "pilot" program could lead to changes. The OMB could decide to raise the 20% requirement to 100% – or scrap it altogether.

Here are the three most important things you can do right now to help, no matter where you are:

1. Engage with the code that is released under this policy. Use it, file issue reports, submit patches for documentation and code, and encourage them. Opposition inside government to policies like

[2]gnu.org/philosophy/free-sw

[3]gnu.org/software/repo-criteria

this includes claims that there is no point in releasing custom government software, that it is just extra overhead.

2. Make your voice heard. You can continue to discuss this particular policy at `github.com/whitehouse/source-code-policy/issues`. You can advocate in your state and country for similar (but better!) policies. You can use the LibrePlanet wiki at `libreplanet.org` as a base for coordinating on advocacy materials.

3. Support the FSF financially. We should celebrate this policy as progress, but we wanted and want much more. If we had been equipped with more staff resources, we could have had a greater impact. The next time there is an opportunity, we want to be stronger and do better.

The US Federal Government has a substantial influence on the software market. Analysts routinely predict it will spend over $2 trillion on hardware, software, and related services each year.[4] If we can encourage and expand this latest policy, so that more of this money is flowing into free software development, it could make a tremendous difference in the culture of software worldwide.

Despite its shortcomings, the policy shows a lot about visibility of the free software movement and use of free software. It is consistent with the theme of our 2017 LibrePlanet conference: "the roots of freedom." The movement's roots anchor a growing structure; it may appear weak at the furthest reaches, but it can get stronger as the underlying root system expands. What we have in this policy are only the first buds of software freedom; but the fact that they made it to such heights says a lot about the strength of the roots we have been growing for 31 years. We should celebrate it as a success, but as usual, keep watering the garden.

[4] u.fsf.org/1z1

Check out our new stickers at `shop.fsf.org`!

So here's the thing: free software isn't cool

By Georgia Young
Program Manager

Now, before you say, "Who cares about being cool when it comes to freedom?" let me explain what I mean, and why cool should matter to the movement.

Let us define cool for this context. A 2014 study by S. Shyam Sundar, Daniel J. Tamul, and Mu Wu in the *International Journal of Human-Computer Studies* identified three criteria for measuring coolness in technology products: originality, attractiveness, and subcultural appeal.[5] In other words, a cool piece of tech is inventive, it looks stylish, and it helps the user assert their identity: these three criteria are my focus here.

Why should free software advocates care about the cool factor when it comes to free software projects and activism? Adoption. People need technology – but they want it to be cool.

Many of the people who drive free software development, use free software, and encourage others to ditch proprietary software in favor of freedom are involved because the Four Freedoms are more important to them than what is cool. Maybe you have been hacking on projects since childhood, or maybe you used proprietary systems for years, until something went wrong: Microsoft forced a Windows update on your home computer, or you learned that your supposedly low-emissions Volkswagen was anything but.[6] You have great reasons for going free.

But other people do not prioritize freedom or change their habits when they realize they are being treated unjustly. Maybe they fret about how they are going to afford an extra $159 for wireless headphones to go along with their new jackless smartphone, but the power of cool can be strong enough to override such concerns.[7]

[5]u.fsf.org/1yg
[6]u.fsf.org/1ye & u.fsf.org/1yf
[7]u.fsf.org/1yr

The key to cool in software and hardware is often rooted in design. We avoid Apple products because they deny us our freedoms, but others perceive their products as easy to use and beautiful, because they are designed with an eye to great user experience and a pleasing look and feel.

But we want everyone to use free software, and that means free software (which by nature promotes user freedom) must find its cool. That is best achieved through design that will be delightful and seamless for the user.

The good news is, we are already on our way. Need a simple way to build a website? Try WordPress.[8] Want a great desktop user experience for your GNU/Linux system? GNOME and KDE have embraced beautiful design throughout their desktop environments.[9] Maybe you or a child you know want to experiment with electronics. Try littleBits, useful and beautiful electronics prototyping hardware.[10] And of course, cool is often about fashion, so the FSF has you covered there with our RUN GCC t-shirt.

WordPress is a web-based publishing system, and a hugely successful free software project. Used by over 25% of the world's 10 million most-visited sites, WordPress makes it easy to assert your identity on your website, offering different themes that change the look and functionality of a website without altering its substance.[11] You might even use WordPress to create a site for your free software project.

GNOME and KDE are GNU/Linux

[8]wordpress.org
[9]gnome.org & kde.org
[10]littlebits.cc
[11]u.fsf.org/1yy

desktop environments that have emphasized user and developer experience in design, and (especially in the case of GNOME's distinctive design) that makes them stand out. GNOME offers human interface guidelines that may inspire your own efforts to integrate good interface design in your development process.[12]

littleBits makes modular electronics pieces that snap together by way of small magnets. They are meant to make prototyping and learning about electronics easy. They are freely licensed, and the best part is: they are fun! Each bit is color-coded using a neon palette that defines their function, making them easy to identify and experiment with. Magnet-based connections mean there's no soldering involved. There are so many invention possibilities, and they are so simple to use, kids and teachers alike get excited about using littleBits.[13] How can a color palette and a consistent look for common elements in your program make it more useful and cool?

Now that you are thinking about how cool your free software project could be, what about your own look? Take the hip-hop inspired RUN GCC t-shirt. While most people probably do not know the GNU Compiler Collection (GCC) – a key piece of the GNU Project – Run-D.M.C. is a wildly popular American hip-hop group founded in the 1980s. Their personal style – black fedoras, Adidas tracksuits, and thick, ropelike gold chains – was as memorable as their lyrics. Their logo, RUN DMC in huge white letters on a black background, framed by horizontal red bars, is highly recognizable, and the RUN GCC logo created for the FSF mirrors that style, sparking curiosity in those unfamiliar with GNU.

It just looks cool.

For more cool stuff visit `shop.fsf.org`.

How might visual and user experience design improvements make your favorite free software project cooler, potentially attracting more users? Share design-related resources and your thoughts on LibrePlanet.[14]

Head in the clouds, files on an actual server

By Ruben Rodriguez
Systems Administrator

Servers are high-grade computers not very different from a regular desktop machine, usually having multiple processors, redundant disk systems, and high-speed network

[12] u.fsf.org/1yh
[13] u.fsf.org/1yi & u.fsf.org/1yx (you can download and watch this video without proprietary JavaScript using youtube-dl)
[14] u.fsf.org/1yt

adapters installed on a high-end motherboard. When people talk about "the cloud," this just means using servers that are under somebody else's control. Even if you do have control of your own servers, they are still a minefield of freedom issues, although there are a few good options.

Freedom advocates often make the point that the backbone of the Internet runs on free software. And while it is true that many free software applications have made their way to be the standard of the industry – be it HTTP servers, databases, code processors, virtualization systems, or management software, among many others – it is still hard to build a completely free software solution if you take into account networking devices and appliances. And in a time when corporations and governments are pushing to weaken our privacy by trying to outlaw or cripple cryptography, or by planting backdoors on common software and hardware, having servers we can trust from the ground up is a priority.

Servers are usually managed remotely by administrators who connect to them to perform setup and maintenance tasks in an efficient way. This is usually done at the application level, but modern servers also offer methods to gain control at a much lower level, in a way that is independent of the operating system or applications that the machine is running, often even if the machine is turned off. Such methods provide complete control over all the data and actions performed by the machine without the operating system being aware of it. Those capabilities could be useful for a sysadmin who has to work with many machines, but when control is in the wrong hands, this access becomes the ultimate backdoor. Most modern processors implement such features: Intel calls it Management Engine, and AMD calls it Platform Security Processor. They both include it in every processor they currently make.

The code that implements these backdoors is of course nonfree software, so we cannot be sure if it is there to serve us or somebody else. Even if we were to assume that it has been placed there with nothing but good intentions, we cannot audit the proprietary software, and we should not trust it. In a similar way, many server motherboards implement remote control functions in their BIOS, which should be avoided for the same reasons. At the FSF, the platform we selected to avoid these problems uses the last common CPU that did not implement any backdoors: AMD Opteron 62xx, released in 2011. It runs on a motherboard (ASUS KGPE D16) that is compatible with the free BIOS replacement, Libreboot.[15] It is powerful enough for a single server to run dozens of virtual machines efficiently.

Selecting all the other components that a server stack usually requires is tricky. Fiber optics network cards have embedded firmware that can potentially host backdoors at a network level, and so do switches. We opted for 10-gigabit Ethernet controllers (Intel X540) that work with the GNU Linux-Libre kernel and unmanaged switches. We also chose a Linux-Libre compatible disk controller card with no RAID support, to avoid nonfree firmware blobs. And of course, these servers

[15] u.fsf.org/1yj

run on fully free GNU/Linux distributions.[16]

The resulting server stack allows for large amounts of fast storage, which is replicated through the network using Ceph. This data pool is then accessed by servers running virtualization, and every component is fully redundant and load-balanced. With this we achieve the most powerful, freedom and privacy respecting servers available today. But there are still things to improve: hard drives have nonfree embedded firmware, and processors contain microcode. These are big black boxes that still need to be set free through reverse engineering.[17]

On the road with RMS

By Jeanne Rasata
Assistant to the President

FSF founder and president Richard Stallman (RMS) is still not slowing down! He continues to champion free software and, since mid-May, has been to nineteen cities across eight countries on three continents to spread the free software movement's message.

He went to Valencia and Alicante, Spain, to raise awareness among political leaders and technical managers of the benefits of free software. As the guest of the Department of Transparency, Social Responsibility, Participation and Cooperation of the Valencian Government, he gave his speech "Free Software in Governments," at the Technical University of Valencia and at the University of Alicante Polytechnic School, respectively.

At the invitation of the school of applied sciences (ENSA), he then headed to Morocco, where, at the colloquium, "Free Software in the Economic South," in Meknès, he spoke about the free software movement and, in Tangiers, about free software, digital development, and the relationship between cybersecurity and free software.

He then went to Villepinte, to be part of Viva Technology Paris and to the Pas Sage En Seine Hacker Space Festival, in Choisy-le-Roi, where he discussed, "policies that have been proposed for freedom in computing, specifically to promote free software in the State and in education, and to limit systematic surveillance of the public, either by the State or by private entities."

At the Eleventh HOPE conference, in New York City, he explained the importance of having software that schools or the government make us run – to get an education, or to avail ourselves of services we have a claim to, or to exercise our rights, or to be heard – be free software.

In August, RMS spoke at the World Social Forum, in Montreal, and at Abstractions, in Pittsburgh, Pennsylvania, where he reached out to an audience of software developers, and also, in September, at Symbiosis Gathering, in Oakdale, California, where he both gave a speech and was on the Technological Society Panel, and at Libre Learn Lab, in Cambridge, Massachusetts, to speak about free software in schools.

RMS also gave stand-alone speeches, throughout the summer and fall, in Monza, Italy; Frankfurt, Germany; Amsterdam, Netherlands; Fresno, California; and Grenoble, France.

[16] u.fsf.org/1yk
[17] u.fsf.org/1yl

In June, RMS was honored by the Association for Computing Machinery (ACM), when it awarded him the prestigious ACM Software System Award, "for the development and leadership of GCC (GNU Compiler Collection), which has enabled extensive software and hardware innovation, and has been a linchpin of the free software movement." This comes twenty-five years after they awarded him the Grace Murray Hopper Award, "for pioneering work in the development of the extensible editor Emacs (Editing Macros)."[18]

RMS literally on the road.

Later, in October, he was honored again, this time by the Pierre and Marie Curie University and the Paris-Sorbonne University, which, in a joint ceremony, in anticipation of their upcoming merger, recognized RMS's entire life's work by awarding him his sixteenth honorary doctorate.

Please write to `rms-assist@gnu.org` with any photographs you would like us to share on RMS's blog, at `fsf.org/blogs/rms`, with recordings of his speeches for our audio-video archive `audio-video.gnu.org`, or to extend a speaking invitation to RMS. See `u.fsf.org/zi` for a list of his confirmed engagements.

[18]Read the award committee's full announcement at `u.fsf.org/1yu`

Free software at the wheel

By Zak Rogoff
Campaigns Manager

Developers are constantly at work extending the opportunity for full computer user freedom on multiple fronts, from smartphones to 3D printers, and we have written enough free software to use PCs with no proprietary programs. There is no question that we will be technically capable of building a functional autonomous car running only free software soon; efforts to do this already exist. The question is whether governments will allow these cars on the road. If we do not engage in a productive dialogue with policymakers and prototype new enforcement and accountability mechanisms for developers, we are likely to get only halfway to free-as-in-freedom autonomous cars.

The debate is likely to come down to reprogrammability. A lot of regulations will be written about the code that manufacturers load in autonomous cars and other robots, likely requiring them to drive safely, to drive in an energy-efficient way, and to pull over when signaled by a police officer (or autonomous police robot). Governments will be uncomfortable with the prospect of individuals overwriting legally compliant car software with something else, and they will be interested in creating meta-policy that makes it harder for owners to bring their cars out of compliance with regulations.

It will probably be impossible to stop governments from creating these meta-policies, but we will have a chance to influence the form they take. The most freedom-maintaining option is to reinforce the existing system of human accountability around cars – for example, extending liability for a car's autonomous behavior to the person that programmed it. The other end of the spectrum is a proprietary software mandate, policy that requires manufacturers to make cars resist users' attempts to reprogram them in the first place, to minimize the possibility of faulty or malicious reprogramming.

As free software advocates and users, we hope to have a system of human accountability that preserves our same rights over car computers that we have when loading software on traditional computers. However, the promise of ex post facto accountability, or even mandatory code inspections, may not be enough to reassure those concerned with the very real possibility of a maliciously reprogrammed autonomous car. Such an atrocity could take many lives before its programmer could be brought to justice and its code taken out of use.

Under restrictions that prevent owner reprogramming, even companies that want to make free software cars would only be able to get halfway there. The most-free cars would echo the TiVo TV-recorder of the early 2000s, which ran programs compiled from free source code that users could copy, study, and modify, but were also hampered with hardware restrictions that prevented users from loading modified software onto the TiVo. Even though the source code for the TiVo's software was free, the version running on the devices was not, because the owner could not exercise Freedom 1 and run a modified copy in its place.

A TiVoized car *may* be better than a car whose software was entirely opaque to the owner, because it would at least be possible to study the code and look for vulnerabilities or other bugs. But it would not empower car owners to fully control the behavior of their vehicles. It would not allow them to fix the vulnerabilities they found if manufacturers were uninterested in addressing them. It would not allow them to prevent their cars from listening to the manufacturer's instructions before theirs, or sharing information about them. It would stop the healthy competition that comes from allowing third-party companies to service cars without manufacturer approval. There is even some risk that it could make matters worse, since malicious attackers could devise exploits based on reading the source code, and users would be unable to update the software to defend themselves. TiVoized cars would be missing some of the most important benefits that we get from free software.

If we want to protect these benefits in cars as with our other devices, we will have to be creative with novel mechanisms of human accountability that demonstrate that it is safe to modify our cars without prior approval – just as a mechanic would. We can start to brainstorm now, drawing on the rich experience of the existing communities that reprogram consumer devices. In fact, if you have any ideas or want to connect with others working on free software cars, we encourage you to share them on the FSF's libreplanet-discuss email list.[19] There is a real chance that we will be able to come up with something in time – the free software community is known for legal as well as technical innovation. Fasten your seatbelts, it's going to be bumpy ride.

Get 10% off!

Support the Free Software Foundation by purchasing GNU Press merchandise.

Visit our `shop.fsf.org` with new stuff coming soon! Enter discount code FALL2016.

The role of lawsuits in GPL compliance

By Donald Robertson, III
Copyright and Licensing Associate

GNU General Public License (GPL) compliance is a perennial topic of interest in the community. As authors of the GPL, and the first organization to release software under that license, the FSF has the longest organizational history in GPL compliance activity. Today, the GNU GPL is widely used by many projects with no FSF affiliation, so interest and discussion about GPL compliance has become varied and widespread.

Nevertheless, the FSF remains a leader in the enforcement of the GPL, and in considerations and discussions about appropriate behavior in the GPL compliance process. When questions arise, part of our role is to clarify the fundamental tenets of copyleft – the tool we invented to advance and defend software freedom for all users. This year, the FSF co-published The Principles of Community-Oriented GPL Enforcement (Principles) with the Software Freedom Conservancy (Conservancy), which explain and formalize the principles that charities like ours follow when employing the GPL to advance software freedom.[20]

That tool is undeniably one with legal backing. Understandably, projects that adopt the GPL regularly discuss its legal aspects, including how and when it is enforced. Many in the free software community read with interest one such discussion a few months ago surrounding Linux.[21] That discussion aroused great interest, since the kernel Linux is widely used as part of the GNU/Linux system, but it also exposed some misunderstandings about how organizations like the FSF handle compliance work.

The zone of agreement in our community is actually much wider than these discussions suggest. We all agree that jumping into lawsuits will not bring violators into the community. Carefully executed compliance activity, fitting with the Principles, welcomes potential collaborators. Jumping into litigation dashes any hope for

[19] u.fsf.org/1ho

[20] u.fsf.org/1yq & u.fsf.org/1yz

[21] u.fsf.org/1y-

that ideal outcome. As stated in the Principles, a lawsuit remains a last resort.

That is how the FSF and Conservancy have always handled compliance. The FSF has done compliance work for the GNU Project for decades, and in all that time, we have only been forced to file a lawsuit once. The suit came about after years of working with the violator trying to correct their compliance. Even in that instance, where the FSF eventually did have to sue, the violator later went on to become a contributor to the GNU Project, and continued other free software activities as well. Conservancy has a similar track record of avoiding lawsuits; they are currently funding Christoph Hellwig's lawsuit against VMware in Germany, which marks the first time Conservancy has ever been involved with a lawsuit regarding Linux, and their FAQ explains the lawsuit came after four years of friendly efforts by many parties asking VMware to follow the GPL's requirements.[22]

The vast majority of our compliance work happens behind the scenes, for good reason: it allows the majority of violators to quietly ameliorate compliance problems and join the free software community. Generally people will only hear about a compliance case if it ends up in court, where by necessity it becomes public. This leaves some with the erroneous impression that GPL compliance involves frequent litigation, and has caused some organizations to take an alarmist stance in opposition to all GPL enforcement. But this perception and policy is based on a confusion. Compliance is almost always an educational matter; most violators are unaware of their obligations under the license and simply need additional help to come into compliance. Almost all GPL compliance cases end quietly with the violator correcting their mistakes, with only a minimal notification of past recipients of the then-violating distribution that anything has happened.

While lawsuits are a last resort, they must unfortunately remain an option. The threat of litigation provides leverage that we need with the rare violators whose GPL compliance problems are not merely mistakes, but are intentional attempts to limit their users' freedom. While compliance work is primarily educational, we need a tool that can work with the rare few who are already educated but chose to violate anyway. Copyleft was designed from the start to serve as that tool.

After our decades of work in GPL compliance, we at the FSF welcome discussion and community feedback. We hope in future discussions that more developers will step forward to share their views, as this issue impacts everyone in the software freedom community. We also hope you will continue to support our compliance work, the work of Conservancy, and any nonprofit enforcing explicitly in line with the Principles, with your memberships and donations. Anyone who has fears about how GPL enforcement could be done in negative ways ought to support organizations who commit to doing it right.

[22] u.fsf.org/1z0

Donate to the FSF with Bitcoin.

Published twice yearly by the Free Software Foundation, 51 Franklin Street, 5th Floor, Boston, MA 02110-1301, (617) 542-5942.
`info@fsf.org`, `fsf.org`

How to Contribute

Associate Membership: Become an associate member of the FSF. Members will receive a bootable USB card, e-mail forwarding, and an account on the FSF's Jabber/XMPP server. To sign up or get more information, visit `member.fsf.org` or write to `membership@fsf.org`.

Online: Use your credit card or PayPal account to make a donation at `donate.fsf.org` or contact `donate@fsf.org` for more information on supporting the FSF.

Jobs: List your job offers on our jobs page: `fsf.org/jobs`.

Free Software Directory: Browse and download from thousands of different free software projects: `directory.fsf.org`.

Volunteer: To learn more, visit `fsf.org/volunteer`.

LibrePlanet: Find local groups in your area or start your own at `libreplanet.org`!

Free Software Supporter: Receive our monthly email newsletter: `fsf.org/fss`.