FREE SOFTWARE FOUNDATION

Bulletin

Issue 28
Spring/Summer 2016

Contents

You are what you run

By John Sullivan
Executive Director

Y*ou are what you eat* is a popular slogan in many parts of the world used by groups encouraging people to change their eating practices. It is used to promote vegetarianism, to discourage "junk food," to campaign against foods containing genetically modified ingredients, and more.

These campaigns do not all agree with each other. The slogan common between them is an attempt at motivating us to more closely examine what it is we are putting in our bodies. No food movement argues everyone needs to be, or should endeavor to become, a Michelin star chef. Everyone acknowledges that people choose to spend different amounts of time and energy learning about preparing and eating food.

But a certain very basic level of food literacy is widespread. We take many of the basics for granted. Even if you do not have advanced culinary skills, you know that hamburgers do not grow on trees (yet), that food in general is prepared by taking ingredients from different places and combining them, often with heat, and that if ingredients are not stored or handled in certain ways, some of them can make you sick.

You know that with the same or similar ingredients, you can prepare a meal at home, or you can have the meal prepared for you at a restaurant. You know that if you do not like the way it is prepared at a restaurant, you can prepare a different version at home. You know that even at a restaurant, if your food is not salty enough, you can add more salt yourself. You know that preparing a single meal can involve several people working together. You know that, while a meal is presented on a very specific plate in one restaurant, the same meal can be presented elsewhere on a different plate while still tasting the same and having the same nutritional value.

Because every healthy human has to eat, and eat every day, every human has to know *something* about food in order to pursue basic happiness. Food-focused movements start from these basics and ask people to learn more, in order to convince them to change habits and make different choices.

When it comes to advocating free software, we often find ourselves stuck on the basics. Try rereading the above "you know that" statements, substituting software for meals. Within the sizable portion of humanity now using or interacting with computers on a daily basis, how many people know those basics? How many people know that they could replace the operating system (meal) on an iPad or a Microsoft Surface tablet (plates), if those devices were not arbitrarily locked down? How many people know that their "smart" thermostat is running computer programs written by programmers (chefs) who may have failed to follow basic code security (sanitation) practices? How many people know that when they visit a Web site, their computer is given programs to execute locally behind the scenes, and that these programs may be doing all kinds of things they do not want? How many people know what source code is, or that programs are usually transformed from human-readable into machine-only formats?

More and more often at the FSF, we are finding our advocacy efforts running into challenges related to a widespread lack of fundamental computer literacy. This is not because people lack the ability to understand, or that they are doing something wrong. It is just a fact that we need to address. It is difficult, because others are very happy with the current state of affairs. There are billions of dollars from proprietary software and service operators pouring into marketing materials along the lines of, "we know you don't want to have to care about how your computer works, you just want it to work."

Ironically, when computers are delivered in new formats – cars, flat, round, handheld, goggles, whatever – they are often accompanied by advertising campaigns by these same companies portraying them as something so radically new that we should of course expect them to come with new rules. This is how people who have been installing whatever software they want on their laptops for years initially accept the idea that Apple can tell them they cannot install software on their iPad from anyone but Apple, or that if they want to switch to Android they have to buy a new phone instead of just flashing their current iPhone.

As user freedom advocates, we should get more involved in doing and encouraging basic computer literacy education. This does not mean asking people to write their own software, or to concern themselves deeply with complex software systems. As with food when we eat out, or when we buy prepared meals or produce at the grocery store, we trust others to do all or some of the work and detailed understanding for us – we just have general knowledge about the processes they are using, and that general knowledge empowers us to protect ourselves.

A little bit of knowledge goes a long way toward putting people in the position to make much better, more ethical choices. Just knowing the basics will not make people automatically support free software. But knowing a little more will make it much easier for them to give it a fair hearing. If we are successful, we could start advocacy campaigns with the slogan, "You are what you run."

In it for the long haul: A model for long-term free software campaigns

By Georgia Young
Program Manager

What is your favorite FSF campaign?[1] How long has it been around, and when was the last time you heard about a victory in that area of the FSF's work? These questions came to mind as I begin to integrate the recommendations of the High Priority Projects (HPP) committee, and it got me thinking about how our three-person campaigns team can evaluate and sustain long-term free software activism.[2]

The HPP campaign launched in 2005 to foster work on projects that are important for increasing the adoption and use of free software. For the first few years, the list was maintained by FSF staff and board members, based on our own research and on feedback occasionally sent in by interested people. In 2014, we formed a committee of free software community leaders, who have done the work of reevaluating and refining the list based on suggestions from the free software community, the changing landscape of software and hardware (and consequently, of restrictions on computer user freedom), and the achievements and failures of projects.[3] The most recent revision, which is currently underway, includes the introduction of four criteria that define what needs are important for the list.

But how can we maximize the usefulness of this list – or any of our campaigns? In the case of this campaign, we have identified projects that are critical to the advancement and adoption of free software. But if nobody steps up to work on these projects, if nobody hears about them, they will never get done. The campaigns team helps by acting as a project manager for the committee, publicizing the list, recommending ways the community can help get the work done, celebrating victories, and reevaluating the work regularly – essentially, we must continue our advocacy indefinitely.

How exhausting! No wonder ac-

[1] fsf.org/campaigns

[2] See u.fsf.org/1sy for the committee's analysis, and u.fsf.org/1sz for the list.

[3] u.fsf.org/15r

tivists suffer from burnout. And we cannot stop until we have achieved a world where all software is free.

So, how do we keep up our momentum in a years-long effort like the HPP list? Here are a few thoughts:

Small wins add up over time. The aims of the HPP list are impossible to achieve in one fell swoop, but over time we can chip away at them, bit by bit. Once we have updated the list, we will be sure to celebrate progress made toward the fulfillment of any of these High Priorities. You will hear about it on `fsf.org`, in the Free Software Supporter, on social media sites, and in the press.

Revisit and challenge. The list has been revised several times – extensively, in the most recent round. This is because software has changed – and the world's needs have changed. At one point, Flash dominated the Web, so the list recommended a Flash replacement – now we are focused on free phones, decentralization, and other issues. By revisiting the project regularly – and soliciting ideas from the community when we do – we keep the work relevant.

Patience and perseverance. If we fail, what do we do? We can start by reevaluating the project. But it may be that we simply need to try again. When change is truly important, actions that energize people can benefit from their failure by observing what went wrong, and what might be needed to lead to success next time. Some people at the World Wide Web Consortium (W3C) were dismissive of our campaign calling for selfies against Digital Restrictions Management (DRM) on the Web – but it got their attention and led to conversation in the press and between the W3C and our movement – conversation that those who favor DRM would prefer to avoid.

Celebrate and share your successes. When an HPP need is met, we will announce to you and the rest of the world about it. And we will celebrate you when you help. To keep up motivation, we will share conversations with the people who are working hard to help check a priority free software project off the list. We will also be clear about how you can help – whether that is through coding, testing and bug reports, documentation, advocacy, or funding.

This is the kind of work the FSF campaigns team facilitates and champions. We've had some successes along the way: Six years ago, we recommended CiviCRM, because it met the requirements of the HPP need for a fully featured donor and contact management system for nonprofits.[4] More recently, we announced that the GNU PDF project was completed, thanks to `libpoppler`*'s* ability to support newer PDF features like annotations and forms.[5]

Wins like these add up, but we can-

[4] u.fsf.org/1t9

[5] u.fsf.org/1ta

not do it alone. Everybody in the free software community can help make the world more free by participating in our campaigns – check out all of our campaigns at `fsf.org/campaigns` for ways to help.

Mr. Snowden: or how I learned to stop worrying and love GStreamer

By Ruben Rodriguez
Systems Administrator

At the time of LibrePlanet 2016, I may have been the newest FSF employee but I was not a newcomer to the conference itself. In the previous four years, my participation involved talks on Trisquel and GNU IceCat, and were excellent experiences each time. Every year, the recordings and streaming of the talks seemed to improve, but still fall short of expectations. When I joined the tech team, it became a personal goal of mine to improve LibrePlanet's video production process. The first step was to make the most out of our equipment. We use Libreboot powered laptops and Elphel cameras, which come with freely licensed hardware specs and GPLed source code. The devices then connect to servers running fully free software. This makes for excellent freedom standards, but it comes with some technical challenges. The Elphel cameras provide high quality video at the expense of extra post-processing on the laptops – which are not that powerful – so a lot of experimenting and optimizing had to be done to get to a pipeline with low latency, good image quality and low CPU usage. This was quite the tall order, but we made great progress with it and our intern David Testé even wrote a great graphical interface![6]

We doubled down on our commitment by booking Edward Snowden to give the opening keynote as a live interview from Moscow. To make it more *fun*, we made the decision to record the interview and stream it live. We tested different options and concluded on using a WebRTC instance and broadcast it with GStreamer and IceCast. With this new toy coming together, we tripled down on our commitment by helping Snowden migrate his stack to free software. We began by researching how to replace his nonfree video compositing program (that ran on Windows), and eventually built a hack using even more GStreamer, a video compositing tool, a local Real-Time Messaging Protocol feed, a virtual webcam device, and... well, way too many things for it to sound like a good idea. But we went to MIT's Stata Center the two weekends before the conference and tested the whole thing,network and all. It worked nicely. You can read about and discuss our setup at `u.fsf.org/1tb`.

Then the day arrived... LibrePlanet.

We got to the venue at 7 a.m., with three hours to unpack all the hardware, set up, have a muffin, and go live with the event (well, that was the plan anyway). Daniel Kahn Gillmor, the interviewer of Snowden, arrived at 8 a.m. and we started to set things up. Quickly we started to realize the network *smelled funny.* The MIT network has a captive portal: to access the Internet you must register your device and wait a couple minutes for the system to let you through. Since we

[6] u.fsf.org/1sn

were bringing a bunch of laptops, we registered them in advance to avoid problems, but the captive portal was still asking us to register. So we did, we waited (while the room started to fill up), and the network continued to fail for every machine, and even for MIT registered staff. We urgently asked the IT department to look into it, and they quickly fixed the issue. We had Internet!

With the room now quite full and only five minutes to spare, we connected with Snowden on the videoconference. To our dismay, his screen blacked out. After some more testing we decided to try a different browser on Snowden's end. He moved to Iceweasel and things were working again. This ate so much of our setup time that not only were we already past the start time, we still had to set and test the recording and streaming... which were also failing!

With my nerves running thin, and trying not to raise my gaze to the nearly four-hundred people in the room that were looking at my frantic typing, finally managed to get it working. We were delayed enough that no testing could be done, so we just declared it ready. I sat nervously next to Daniel, and let the keynote start. It worked like a charm!

Governments pay to reinvent the wheel, or buy a proprietary wheel

By kosa
Web Developer

We all know that free and gratis are not the same thing, and sometimes free software is also about money. In the global economic south, huge sums of money, in the form of public resources, are paid to develop proprietary software to try to overcome the gap between them and the so-called developed nations.

I think it is great that governments pay for software development. I am also absolutely convinced that it should all be free, especially if it is being developed with money coming from people's taxes. Unfortunately, local governments in the emerging economies do not use free software nearly enough; therefore they pay to reinvent the wheel, or buy a proprietary wheel, instead of taking advantage of the software that has already been developed and has the freedom to continue developing without restrictions or licensing costs.

As an example twelve years ago, the Mexican government spent twenty-four billion pesos (roughly about two billion US dollars at that time) to "develop" a platform called Enciclomedia that does pretty much the same thing Wikipedia does, but is based on the infamous Microsoft's Encarta and only adds a few interactive functions.[7]

Eventually the entire project was abandoned due to its absolutely horrifying design and the prohibitive licensing fees the Mexican government was obliged to keep paying to use it, with more taxpayer funds, after paying for development.

In this case, one of the developers took the so-called "digital objects" and created a fork from it called Encicloabierta (coming from encyclopedia and *abierta*, which is the Spanish word for "open"). He kept maintaining it even after facing a trial for using "copyrighted materials," but even-

[7] u.fsf.org/1s-

tually stopped doing it due to lack of resources.

There are several other cases of governments paying for the "development" of the exact same piece of code in different counties, either for water and public services management or even public schools administration.

While free software is a widespread ideal among many developed circles, it is not in most emerging economies' governments. And that, mixed with ambitious salespeople, companies, and corrupt governments, makes a broth for "those who know" to make millions out of poor people's taxes instead of using those funds to do real development. The governments could both save money and make better use of the funds by getting it back to the community that started and supported the software development in the first place.

Spreading the existence of such free software among all people, but especially within non-governmental organizations who struggle to make government more transparent, can both reduce corruption and save millions of pesos, soles, quetzales, guaranís, bolivares, colones, lempiras and reales all across Latin American and the world's emerging economies. But furthermore, it will add some great stories to the book of a different history for humankind as a whole.

A lot of people around the free software movement are fed up with what government does, and especially about what is done with money coming from taxes, but this is a great chance to make it do the right thing. So, if you feel inspired by any of these ideas, look at "Measures Governments Can Use to Promote Free Software" and join the online LibrePlanet advocacy group focused on this issue.[8]

User freedom in the age of computer-generated software

By Chris Webber
GNU Maintainer

Many of us share a vision for the way software, free or otherwise, is developed: software is written by a programmer as "source code" and transformed through some mechanisms into "object code." As free software activists, we are used to thinking about our legal, development, and community processes and tooling in terms of this workflow. But what happens when software which used to be written manually by humans is developed generatively through other software? How does this affect software and user freedom?

Of course, by speaking of the above I am talking about *artificial intelligence* (AI), a topic and term which is both compelling and vague. At one point almost *everything* in the world of computers was considered to be "artificial intelligence," including fundamental building blocks like compilers. This has lead to what is sometimes called "the AI effect," where everything is considered "artificial intelligence" until we know how to do it. This has lead to not only push back against the term "AI" but even its pursuit; why chase a concept which ceases to exist once uncovered?

I think this is missing something important: regardless of term vagueness, original visions for AI aimed

[8] See u.fsf.org/1t0 and u.fsf.org/1t1, respectively.

for machines which could think for or program themselves. This vision permitted the idea of "generative software," where humans were not manually writing so much of the logic of the system. But much of the resources towards AI research tapered off through the "AI Winter" which settled in through the 1980s and 1990s. Since the "AI Winter," we've seen the majority of programming resources going towards other things like web development, graphical interfaces, business needs, games, and so on. Typically development has involved humans manually writing the logic underpinning the system.

Recently this has been changing. There has been much news around Google beating Atari games and Go champions, not through manually written strategies, but through neural networks which are trained to build something resembling human intuition. Likewise, many more companies are hiring for positions involving "machine learning" to reduce the amount of manual programming required.

In other words, the AI Winter has thawed. So where does this leave free software?

One question we might ask is, "do user freedom questions still apply?" Let's consider a scenario.[9] Imagine you are in a generatively programmed self-driving car, and the car unexpectedly swerves off the road into a ditch. Afterwards, you would like to ask the car, "why did you do what you did?" Via some mechanism, you could in theory "talk to" the machine's generated AI system to ask *it* why it did what it did. But will the car manufacturer permit you to do so?

Through this example, we can quickly realize that all four software freedoms still apply: the freedom to run, study, redistribute, and redistribute modified versions of an AI. It is also easy to see that not everyone might want you to have these rights; one can easily imagine a less scrupulous manufacturer saying, "I'm sorry, we can't let you talk to that AI... that's *our* AI." (Thus one can easily see that even generative software should not have owners.)

So, software freedom applies, but how does it "work?" It may be hard to apply the methodologies we are used to when humans are not manually programming the software used.[10] Still, one can imagine collaborative methodologies that do work on the basis of sharing some dataset; perhaps many users (programmers and non-programmers alike) helping train software generated via genetic programming.

And what of our legal tools? Does copyright apply? Does copyleft apply? If not, are there other ways to protect the commons of software being developed as others attempt to lock it down?

There are multiple directions of generative software to approach, from machine learning to symbolic-based expert reasoning systems to genetic programming. Some of these systems may be more appealing than others; systems which clearly express their

[9] Thanks to Gerald Sussman for inspiring this example through conversations at the FSF's 30th anniversary party.

[10] This led fellow free software activist Asheesh Laroia to observe that perhaps this demonstrates that "open source as methodology" was a distraction, and that software freedom was the real goal all along.

symbolic reasoning may be preferable (and are more "accountable"). At this time, what is most important is to get more free software activists exploring this space.

Happy spelunking!

On the road with RMS

By Jeanne Rasata
Assistant to the President

Richard Stallman (RMS) is not slowing down! Over the past six months, with his characteristic unflinching focus, he has continued getting the word out on computing ethics and raising awareness of the social issues at stake: since we last wrote you, he has attended over a dozen conferences and given forty-one speeches in thirty-seven cities across eleven countries and three continents.

Last fall, in Paris, he spoke to Supélec's researchers and doctoral students about the importance of free hardware designs. In Ghent, Stockholm, Østersund, Barcelona, Utrecht, Athens and France, he addressed diverse student audiences and the general public alike.

He attended Guadalajara's annual Feria Internacional del Libro, Latin America's most important editorial meeting and an important cultural festival, to speak on the importance of a free digital society and warned attendees of the dangers of the Trans-Pacific Partnership's restrictions on computing and the Internet.[11] He took advantage of the visit to give six other speeches at different universities throughout Mexico, including at the Universidad Tecnológica de la Mixteca, as a guest of KadaSoftware, a part of the university that is dedicated to supporting the school and the region through software development and an exemplary part of the free software movement.

In India he gave speeches in Delhi and Bhopal and spoke at two different tech festivals, in Pilani and in Roorkee. Likewise, he spoke at Fossetcon, in Orlando, FL, and then in Madrid, at Retina 2016, a conference directed at professionals responsible for the digital transformation of their companies, and to Istanbul Tech Talks, which targets software engineers. Even when his audience is experienced developers, he continues to open eyes to the social implications of software.

In Canada, he spoke both at the Université Laval, in Quebec City, and then in Montreal, at the III Colloque libre de l'ADTE, which promotes free software in colleges and universities in Quebec.

Back in the US, he spoke at MIT and at Oakland University, in Rochester, MI, in addition to speaking at LibrePlanet, the FSF's annual conference, and handing out this year's Free Software Awards.

All throughout his travels, RMS

[11] u.fsf.org/1t6

highlighted the injustices of proprietary software through user subjection to developers' power, exemplified by surveillance, DRM, and back doors.

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`, speeches to publish in our audio-video archive, or to extend a speaking invitation to RMS.[12] See `u.fsf.org/zi` for a list of his confirmed engagements.

We need to fight for strong encryption. And stop DRM in Web standards.

By Zak Rogoff
Campaigns Manager

Encryption is the backbone of privacy and anonymity technologies, and these technologies are an important driving force for democracy in the 21st century. Their adoption is partially equalizing the balance of power between people and governments, enabling transparency, accountability, and freedom. Intimidated, the law-enforcement arms of many governments are attempting to slow this process by banning strong encryption.

The FBI took a big swing at the right to encrypt this spring, when it attempted to force Apple to break its own encryption on an iPhone. They did not quite succeed, but it is important that we stay vigilant – they will try again to strong-arm tech companies into weakening widely-used encryption systems. We must stop them, both to protect the new political power of encryption, but also because weakening it would cause huge collateral damage to millions of innocent people using the Internet and the global financial system.

The mainstream media has covered the FBI versus Apple fight, but so far the free software movement has not been able to effectively leverage it as an opportunity to teach the public the deeper truth about our computing rights: encryption is important, but no real popular control of computers is possible without free software at the center. Programs like GnuPG, Tor, and OpenSSL are the gold standard in encryption because their free licenses grant users the transparency necessary to verify that they are secure, and the freedom to fix insecurities. If we celebrate Apple's stand for strong encryption uncritically, we miss an opportunity to point out that Apple's proprietary encryption (while it is a step up from proprietary software *without* encryption) still represents an evolutionary dead-end for our society.

Just as the FBI hopes to set a precedent by making Apple crack its own encryption, the DRM lobby is currently pushing for a major political victory to legitimize its restrictive technology and make it easier and cheaper to implement. DRM is software that runs on your devices and polices your behavior. It is what stops you from copying streaming videos and songs onto your hard drive, prevents you from using some programs without an Internet connection, and stops you from moving books between e-readers.

Its owners claim DRM is necessary to "protect creators" by stop-

[12] audio-video.gnu.org

ping unauthorized copying. While this sounds very virtuous, it is rarely, if ever, true. The precise motivations vary, but the goal of DRM is usually either removing functionality and selling it back piecemeal, or preventing competitors from making interoperable products.

Recently, Netflix, Apple, Google, and Microsoft have crafted a new universal DRM system for the Web, called Encrypted Media Extensions (EME). They are trying to get it ratified by the W3C, which sets official Web standards. For many of the same reasons that we need to protect strong encryption, we also need to stop this power grab by those that profit from DRM.

Weakened encryption loosens our control on our computers. DRM does this as well, by encumbering our devices with proprietary code that treats us as adversaries. DRM is impossible to implement effectively with free software, so any system that requires it also locks out users that are committed to protecting their own freedom. Perhaps worst of all, the continued legal and political acceptance of DRM marginalizes our general claim to control over our computers, and legitimizes the idea that media distributors' business models should trump user freedom.

Encryption is an essential pillar in computer security, which is one of the reasons that such diverse groups are united against government attempts to weaken it. Like weakened encryption, DRM is a nightmare for security.

Because it is a black box that users are compelled to install and that is designed to be hard to remove, DRM becomes a tempting home for every kind of abuse and attack that a software author can perpetrate on a user. Even if a DRM's owner does not actually command it to attack or spy on users, others often slip through the hole it has punched in users' security.

Fearful of public scrutiny, the DRM lobby has passed laws (the Digital Millennium Copyright Act in the US, followed by similar laws and treaties in many countries) to effectively gag security researchers seeking to expose and fix vulnerabilities in systems that include DRM. This means that the best system we have for protecting users from insecure programs – independent expert review – is outlawed.

To protect user control and digital security, we need to make DRM politically expensive. Currently, we are fighting this struggle in the arena of Web standards. The free software community plays a leadership role in the fight against this backwards step for the Web, through our Defective by Design campaign. We call on anyone concerned with strong encryption to join us by signing our petition and by adding a protest selfie to our growing gallery.[13]

There is a blooming global consciousness of the need for secure and user-controlled technology, and DRM is not a part of that picture. Resist DRM with us, and demand a Web that puts users first.

[13] defectivebydesign.org/action

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`.