FREE SOFTWARE FOUNDATION
Bulletin
Issue 39
Fall 2021

Contents

Your support helps determine the future of the FSF

By Geoff Knauth
FSF President

The most important work of the Free Software Foundation (FSF) is to promote and defend computer freedom for users. Since the beginning of the Foundation in 1985, the goal was not just a GNU system of software that maximizes user freedom, but also robust advocacy for that freedom.

As the years have passed, many organizations have chosen to advocate for aspects of free software that are easier for businesses and users to adopt than true user freedom, and while that has led to widespread incorporation of free software into the infrastructure of most computing now, that adoption rests largely on convenience. There is always a danger that the rights that users must have could be watered down or ignored altogether. It is the FSF's obligation to make sure that does not happen, that we never forget there are forces that would corrupt the intent of the free software movement, by removing the movement's idealism and replacing it with mere opportunism. Free software could then be mistaken for product without cost because of donated labor, or product without soul because it was not made with the intent of freedom to share and develop over time.

Some developers may indeed just

want their software to be used, but the best developers want their creations to be inspiring and part of a learning process. They want their software to be extended and improved by others; like good parents they want their children to thrive and have independence and better lives. When I think of programs like GNU Emacs, or TeX or LaTeX, these are packages that started with singular creative genius, but were then enhanced by thousands of others with brains and hearts just as strong. There are many other examples of course. The important thing is that flourishing of creative freedom must never be throttled, it must always be defended and encouraged, and people must not lose heart when it comes to the FSF's fundamental purpose of fighting for this freedom.

The FSF needs to rise to meet these challenges, and that includes doing work on itself. The Board recognizes there have been flaws in our governance. We listen to and discuss at great length all that is written or said, both helpful suggestions as well as harsh criticism. We take all of it to heart. This year in particular, in 2021, the Board has met very often with the aim of addressing these issues. For many years, we met twice a year for a day or two. Since March, we have been meeting two to four times a week, sometimes for two hours, or one hour, or six hours. It has been both a struggle and a team effort. We engaged outside consultants to offer suggestions and evaluate ideas for good governance, and for improved coordination in communication.

The intermediate results being crafted are a Board Member Service Agreement, a Code of Ethics, and a transparent process for bringing new people onto the Board, people with free software spirit who can help build the future of the FSF. This process will involve associate members of the FSF through including them in the nominations process and discussions in the selection process. This work is still ongoing, but progress is substantial and I think we are nearing a point where the free software community can evaluate it. You, members of this community, are welcome and encouraged to participate in this process by joining the FSF. As we move forward in renewal, I mostly ask that we remember why we are here: free software and protecting the freedom of computing for users. We must preserve those core values, as we strive while being different to be as decent and considerate of each other as we possibly can.

Keeping track of freedom while managing packages

By Michael McMahon
Web Developer

A programming-language-specific package manager is a package manager built to aid or extend the functionality of a given programming language by aggregating programs and modules that are specifically written for a programming language. Nearly every modern, popular programming language has at least one package manager and repository available. Unfortunately, it can be difficult to track the mix of free and nonfree licenses for dependencies when using these package managers.

Why do we have all of these package managers when GNU/Linux systems usually include a package manager already? Many programming language packages are maintained by GNU/Linux distributions and found in the operating system's repository, but the number of packages that are found in the repository is a small subset of the total number for each programming language.

Ideally all software would be free. We should be able to easily identify any nonfree packages that are widely used and organize efforts to either get them freed, or replace them with equivalent free packages. When someone installs software using a package manager, it does not currently verify whether its license is free or not. Manually checking the license of one package is not very difficult, but some can have 100+ dependencies. Completing a due diligence license check becomes exponentially difficult and time consuming without automation. Most repositories keep track of the licenses of the software stored within them, but the quality of repository license data varies.

Fully free GNU/Linux distributions that follow the Free System Distribution Guidelines (`u.fsf.org/i7`) handle this issue by using their own system repositories that only contain free software and by removing most additional programming-language-specific package managers from their repositories until a solution is worked out. When additional programming-language-specific packages are required, the user will be faced with a broken workflow that will need to be manually resolved by either installing it outside of their main package manager or by using a different operating system that does not remove packages.

Stable distributions such as

Trisquel, Debian, or Ubuntu lock their package versions at the time of release and maintain updates to those packages only with security patches. When using a programming language package manager with a potentially older (up to five years old) version with new security updates of the programming language, users should expect to run into unmet dependency requirement problems. The common solution for this is to use programming language version managers; these can install different versions of a programming language environment concurrently.

When a version manager installs a new version of a programming language, each installed instance is expected to include a programming-language-specific package manager built for that version. Manually modifying package managers and repositories for all of these additional programming language installations would be difficult.

I will propose a few ways in which we can approach solutions for the issue, but we really need a community effort if we are to improve the situation. The best way to handle macro issues at scale is to work upstream and convince package managers and repository maintainers that this is an issue, and most importantly to offer help building and maintaining solutions.

• *Package manager configuration:* Package managers should have the configuration option to exclude packages that were deemed to be nonfree based on license data.

• *Fork:* If the upstream package managers are not interested in merging this functionality, forks could be maintained that have this feature. If a fork is the solution, I would propose the fully free GNU/Linux distributions band together to build and maintain such a tool. The version managers would also need the ability to install the fork.

• *Self-reporting licenses:* At the very least, repositories should require reporting the license of a package in order to submit a new entry. Most repositories do mandatory license self-reporting at this point which is an important first step.

• *Automated license compliance scanning:* The SPDX (`spdx.org`) project keeps an exhaustive list of license text and standard license headers that can be leveraged by license compliance software to better scan projects for license compliance and verify license information kept by repositories. The Free Software Directory (`u.fsf.org/ky`) teams use FOSSology, Licenseutils, and ScanCode Toolkit to help scan repositories for license compliance. Developing interpretative automation for these free software tools would help the licensing community.

• *Community review:* If tools are not built to aid the repositories in automatic license compliance, repository maintainers are unlikely to change systemically. A large number of volunteers could manually review repositories and submit corrections.

• *Alternative repositories:* When repositories are unwilling or unable to implement the changes, alternative repositories could be maintained by members of the free software community.

All of these layers of abstraction were built to make things simpler, but their long term effect is that in the process of making things simpler, it has made it difficult for people to know and understand the software that they use. We have a track record of working through major issues and I am optimistic that if we jump on the issue now, we can solve it together. 

Nominate someone for the Free Software Awards today!

`fsf.org/awards`

Just using free software makes you part of our collective journey to freedom, but some go above and beyond in their dedication to the free software movement. Now, it's time for us to show those community members and projects that we appreciate their vital work.

- *The Award for the Advancement of Free Software;*
- *The Award for Projects of Social Benefit; or*
- *The Award for Outstanding New Free Software Contributor.*

LibrePlanet 2022: Living Liberation

Sessions are now open at `my.fsf.org/lp-call-for-sessions`

All of us play a role in creating free software: developing it, distributing it, sharing feedback about it, and spreading both its code and underlying message. From the beginning of the movement, and every day, users have supplanted technological oppression with empowerment.

The theme is about how people make free software part of their daily lives, one decision at a time, and we want to hear from you! Share how you integrate free software in your life, your struggles and successes, or explore the theme through spheres of education, licensing, medicine, government, business, art, social movements, or improving accessibility.

About the FSF's work in 2021

FREE SOFTWARE FOUNDATION

45+ certified products available for sale from 7 companies

100% SCORE

CHARITY NAVIGATOR
Four Star Charity

300+ annual **copyright assignments**

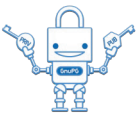
16,539 packages **Free Software Directory**

1,500 volunteers around the world

228,000 FREE SOFTWARE SUPPORTER MAILING LIST SUBSCRIBERS

2021 resources we focused on

END SOFTWARE PATENTS
Revamped campaign Web site



Email Self-Defense guide update

JShelter browser extension launch

DEFENDING THE FREEDOM OF COMPUTER USERS SINCE 1985

Thank you...

To all our associate members around the world for supporting free software.

- United States
- Germany
- France
- Australia
- Sweden
- Netherlands
- Norway
- Austria
- Brazil
- Belgium
- China
- Poland
- New Zealand
- Korea, Republic of
- Greece
- Hungary
- Romania
- Israel
- Turkey
- Singapore
- Luxembourg
- Thailand
- Uruguay
- Philippines
- Slovakia
- Belarus
- Guatemala
- Lithuania
- French Polynesia
- Indonesia
- Malaysia
- Sri Lanka
- Cambodia
- Croatia
- Ecuador
- Estonia
- Kazakhstan
- Macao
- Monaco
- Palestine, State of
- Qatar
- Senegal
- United Kingdom
- Canada
- Spain
- Switzerland
- Italy
- Japan
- Russian Federation
- Denmark
- Other
- Mexico
- India
- Ireland
- Czech Republic
- Taiwan
- Chile
- Portugal
- Bulgaria
- Hong Kong
- Argentina
- Ukraine
- Slovenia
- Colombia
- Viet Nam
- Serbia
- South Africa
- Dominican Republic
- Latvia
- Malta
- Iceland
- Iran
- Peru
- Brunei Darussalam
- Costa Rica
- Cyprus
- El Salvador
- Isle of Man
- Liechtenstein
- Moldova
- Nigeria
- Paraguay
- Rwanda
- Venezuela

Freedom for authors and readers: talking with Nicholas Bernhard of Nantucket E-Books

By Greg Farough
Campaigns Manager

For this issue of the *Bulletin*, and as part of the FSF's commitment to a free e-reader, Greg from the FSF campaigns team conducted an interview with Nicholas Bernhard, the founder and Chief Executive Officer of Nantucket E-Books. (See `u.fsf.org/3gp`)

Could you give a short description of Nantucket E-Books and Shanty, and what you hope to accomplish with those projects?

Nantucket E-Books is a platform that makes it easier for authors to create and share really great e-books. The first part is the e-books themselves: they can be read in the browser, so no special apps or devices are required. They are mobile-responsive, so they'll look good on phones, tablets, or laptops. They have interactive features you'd expect: dark mode, notes, text resizing, bookmarks. Audiobooks can be built-in. The platform also respects the reader's freedom: GPLv3 or later for all the software on the site, and the Web site is compliant with the GNU LibreJS browser extension.

Could you speak a little bit about why you think having free tools to read and edit e-books is important?

When I started this project, I knew I would release it as free software eventually, which happened with v1.5. The most important reason, for me, was being fair to the people reading my e-books. Readers should be able to see how the e-book is made, and what makes the interactive features work. Even if they don't exercise that option, they should know the option is there, or know they can take it to someone who can repair it or improve it. Fortunately, there's more public awareness now, at least for hardware, thanks to the Right to Repair movement.

Stallman made some good points in his "Danger of E-Books" (`u.fsf.org/3gt`) flyer. With most modern e-books, you don't really own the book. Sometimes there's Digital Restrictions Management (DRM) that prevents the reader from sharing the book.

You can't pay anonymously. Any notes you make in a Kindle e-book can be read by Amazon. They can also pull the book from your device if they feel like it. I wanted to go against that trend. Just recently, a friend of mine was listening to an Audible book for book club. Suddenly, the book stopped playing. Later that day, the audiobook on their phone had been replaced with a newer edition that tied in with an upcoming movie adaptation. I'm sure your readers are familiar with the Amazon *1984* incident.

Did you experience any kind of technical and/or social challenges during the development of Nantucket E-Books or Shanty? What were they?

I had a lot to learn about maintaining a Web project: version control, coordinating help from volunteers to help debug, publicizing updates, and making sure links work. By far the biggest challenge was Arrowhead (`u.fsf.org/3gr`), which is a browser-based text editor for previewing Shanty text as e-books. In essence, it's a graphical interface for the text parser. They say that adding a GUI to a software project increases the complexity by an order of magnitude, and I can certainly confirm that!

Are there any technical areas where you could use volunteer support from the free software community? If so, what's the best way for them to get involved?

The best way I can improve the site for authors is to get feedback from them, know what's working and what isn't. In addition, the next step for the site is a stronger back-end, where authors can create accounts and upload files through the site. If anyone would like to help with that, I would greatly appreciate it.

Nicholas can be reached at `njb@nantucketebooks.com` or on IRC at `nantucketebooks.com/6667`

Read the full transcript with extra questions on `fsf.org/bulletin`

Speaking of DRM

Join us for the International Day Against DRM (IDAD), to be held on December 10th 2021.

The fundamentals of the AGPLv3

By Craig Topham

Copyright & Licensing Associate

The GNU Affero General Public License version 3 (AGPLv3) is the most protective of computer user freedom, yet it remains the most misunderstood of the GNU family of licenses. The AGPLv3 was created to solve a very specific problem: how to protect a user's rights when the program in question is being utilized over a network. In this article we will cover where it came from, how we benefit from it, and why a developer should consider it.

The AGPLv3 traces its origins to a company called Affero, Inc. (u.fsf.org/3gu). Affero was established in 2001, and they provided a platform for interactive "Web applications" like discussion forums, mailing lists, email, and blogs. Affero wanted to be sure that users could access the source code for these applications, and that anyone who built derivatives from them would also share alike. The copyleft license of choice at the time was the GNU General Public License version two (GPLv2). However, the GPLv2 was written when the client/server paradigm was not widespread; it could not provide the copyleft assurance desired for Affero's platform. That is to say, one could obtain Affero's source code, modify it, and allow users access to the program over a network without the obligation of releasing its source code to the public. With this dilemma in mind and some help from the FSF, the Affero General Public License version one was published in March 2002 by Affero. In November of 2007, the AGPL joined the GNU family of licenses with version three, giving us a freedom-protecting copyleft license for an increasingly networked world.

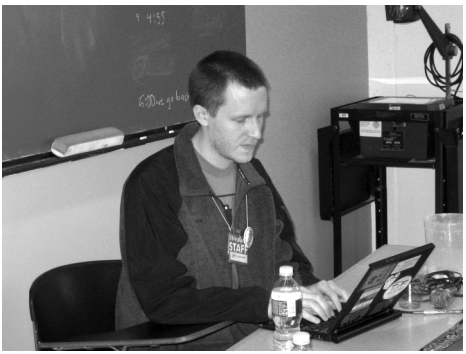
Simply put, the AGPLv3 is effectively the GPLv3, but with an additional licensing term that ensures that users who interact over a network with modified versions of the program can receive the source code for that program. In both licenses, sections four through six provide the terms that give users the right to receive the source code of a program. These terms cover the distribution of verbatim or modified source code as well as compiled executable binaries. However, they only apply when a program is distributed, or more specifically, conveyed to a recipient. Using a program over a network is not "conveying." It is important to note that this only applies to the code running on the server, and not for example to the

JavaScript programs that your browser may download and run locally — these *are* conveyed to you.

The AGPLv3 does not adjust or expand the definition of conveying. Instead, it includes an additional right that if the program is expressly designed to accept user requests and send responses over a network, the user is entitled to receive the source code of the version being used. For license compatibility reasons, written into section 13 of both the GPLv3 and the AGPLv3 is the explicit permission to link or combine any covered work under the other license. Paraphrased from the GPLv3 section thirteen; you have permission to link or combine any covered work with a work licensed under the AGPLv3 into a single combined work. The GPLv3 license will continue to apply to the part which is the covered work, but the special requirements of the AGPLv3, section thirteen, will apply to the combined work.

When confronted with a choice between the AGPLv3 and GPLv3, a developer may think that their program doesn't need the extra protection afforded by the AGPLv3, but who knows what the future may hold! For now, their program does not get used over a network, but someday it might. We encourage developers to consider carefully whether their program *could* be deployed by someone else as part of a network service. By choosing the AGPLv3 (or any later version) in these situations, the developer can future-proof their program in case someone takes the project in that direction. See `u.fsf.org/3gv` for more information when considering the AGPLv3. To learn more about the AGPL and the GNU family of licenses, visit `u.fsf.org/3gw`.

A note from the makers of this *Bulletin*



You might have noticed that this is the first *Bulletin* in a very long time that doesn't include an article from our outgoing executive director John Sullivan.

We would like to thank John for his years of contributions to this publication and his dedication to free software. The FSF staff is sad to see him go, and wish him the best of luck in his future endeavors.

Donate to the FSF with Bitcoin:
1CnyKZ26peigXyShTz
2664ReoUbMQM4RE4

Published twice yearly by the Free Software Foundation, 51 Franklin Street, 5th Floor, Boston, MA 02110-1335, (617) 542-5942
info@fsf.org

This *Bulletin* was produced using all free software, including Inkscape, Scribus, and GIMP.

IMAGE CREDITS

Page 4: Image by Michael McMahon
Page 5, 6, 7: Images by Zoe Kooyman

How to get involved:

Associate Membership:
Become an associate member of the FSF. Members will receive a bootable 16GB USB card, email forwarding, access to the FSF videoconferencing server, and an account on the FSF's Jabber/XMPP server. Plus: participate in our members forum at forum.members.fsf.org! To sign up or get more information, visit member.fsf.org or write to membership@fsf.org.

Online: Make a donation at donate.fsf.org, or contact us for more information on supporting the FSF.

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

Free Software Directory:
Help update and add to 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! And join us for the yearly LibrePlanet conference next spring.

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