FREE SOFTWARE FOUNDATION

# Bulletin

Issue 11
December 2007

## Contents

## Clearing Landmines

***by Peter Brown***
***Executive Director***

For the past twenty-two years the Free Software Foundation has had the mission to promote, preserve, and protect the freedoms to use, study, copy, modify, and redistribute modified computer software, and to defend the rights of all free software users.

When the FSF was founded in 1985 there did not yet exist an all free software operating system (OS) so our efforts focused on sponsoring the GNU system. The free software community was small at first, but it quickly grew in numbers and in its geographic reach.

In addition to software development in the 1980s, the FSF began to focus on the licensing of free software and the idea of protecting free software from efforts to make it proprietary. Developing and refining the idea of a copyleft, FSF founder and president Richard Stallman wrote the GNU General Public License and later the GNU Free Documentation License—which is used for documentation, manuals, and countless other works, including all Wikipedia articles.

Throughout the 1990s, the GNU system along with the kernel Linux was able to run as a standalone OS. Development happened so rapidly that by the end of the decade, our focus was switching from pure software development to spreading the philosophy and tackling the growing threats to free software that came as result of its success.

Today you can, rather easily, install an all free OS that has been produced from the contributions of hundreds of thousands of people from around the world. However, what took over twenty years of community development to build and spread is under threat: threats from government legislation, threats from Digital Restrictions Management (DRM), threats from software patents. The FSF has been campaigning these past few years to counter these threats.

Our anti-DRM campaign

`DefectiveByDesign.org` has had tremendous success and DRM is in decline in the music industry. After eighteen months of community review, we released GPLv3 on June 29—a license that neutralizes anti-free software laws like the Digital Millenium Copyright Act (DMCA) and protects free software from tivoization. Now, we are turning our attention to clearing the landmines represented by software patents.

In 2004 Richard Stallman said,

> Software patents are the software project equivalent of land mines: each design decision carries a risk of stepping on a patent, which can destroy your project. However, fighting patents one by one will never eliminate the danger of software patents, any more than swatting mosquitoes will eliminate malaria.

Software patents obstruct the progress of software development and they pose a threat to every software developer looking to contribute to the development of the GNU system and any free software project. Indeed, Microsoft is using the threat of software patents to extract tolls from corporations for using GNU/Linux and is threatening free software distributors.

There are many arguments against the patentability of software that one can logically make, but it is a commonly held perception that software is patentable within the USA, and the best that can be done is to limit the effect of software patents by engineering around them or looking for prior art to challenge the validity of a particular claim. But this perception is wrong. The US Supreme Court has ruled three times that software cannot be patented, and has worked hard to draw a line to ensure patentable manufacturing processes could utilize steps in software as long as there was a physically inventive step.

Unfortunately, the lower Court of Appeals for the Federal Circuit has made a mess of the work of the Supreme Court, and it is that mess we aim to clear up—and with it, all the software patent landmines.

The Free Software Foundation is launching a coalition effort to End Software Patents (ESP), `endsoftpatents.org`. Inside this bulletin, the director of ESP Ben Klemens helps clarify the quicksand upon which software patents rest.

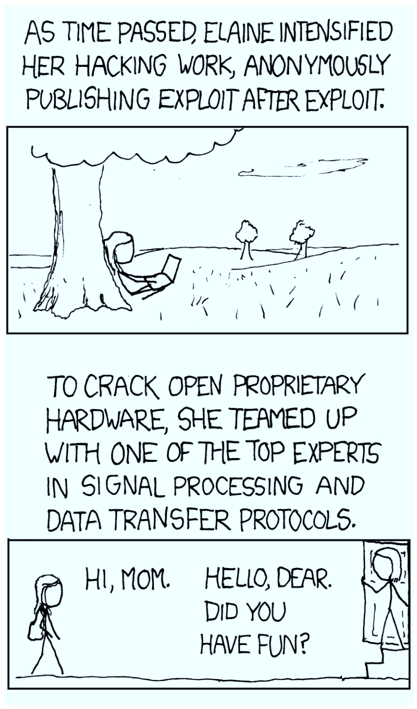
Dear Netflix,

I am unable to use your streaming video service. The Digital Restrictions Management (DRM) you impose would force me to give up control of my computer, which is an unacceptable compromise of my liberty. I have said no to DRM on music, and now I'm saying no to DRM on video.

Netflix should lead the way in providing a streaming video service that is DRM-free.

Sincerely,

DRM Elimination Crew Member
`www.DefectiveByDesign.org`


AS TIME PASSED, ELAINE INTENSIFIED HER HACKING WORK, ANONYMOUSLY PUBLISHING EXPLOIT AFTER EXPLOIT.
TO CRACK OPEN PROPRIETARY HARDWARE, SHE TEAMED UP WITH ONE OF THE TOP EXPERTS IN SIGNAL PROCESSING AND DATA TRANSFER PROTOCOLS.
HI, MOM.
HELLO, DEAR. DID YOU HAVE FUN?

## End Software Patents Coalition

***by Ben Klemens***
***Director, End Software Patents***

Our primary goal in this campaign is to reverse the U.S. Court of Appeals for the Federal Circuit (CAFC) decision of *In re Alappat.* Here, I will explain the history of what that ruling meant, and why that same history has shown us that it should be the focus of our campaign.

From whence does the law of what can be claimed in a patent derive?

**Legislative:** There is basically nothing. To date, Congress has handed off this question to the other branches.

**Executive:** There is the USPTO's *Manual of Patent Examination and Procedure* (MPEP), which is their attempt to interpret the existing judicial rulings in a coherent manner.

**Judicial:** Because the other branches are currently leaving it up to the courts, the existing law by which a claim is judged is entirely from the courts.

The Supreme Court made three rulings regarding whether software is patentable: *Gottshalk v. Benson*, *Parker v. Flook*, and *Diamond v. Diehr*. These can easily be read as the Court dealing with the question of what is a general-purpose computer and how it differs from a patentable device. The first two rulings stated, hands-down, that software should not be patentable, even if the mathematical formula embodied in the software is "useful" for human affairs, and even if there is a physical last step, which the *Flook* ruling called "post-solution applications of such a formula." The third ruling, *Diamond v. Diehr*, allowed a patent on complex machinery to stand. The conclusion of the ruling is worth reading in its entirety, because it is the axis around which the "what is a general-purpose computer" definition revolves:

> We have before us today only the question of whether respondents' claims [are] patentable subject matter. We view respondents' claims as nothing more than a process for molding rubber products and not as an attempt to patent a mathematical formula. We recognize, of course, that when a claim recites a mathematical formula (or scientific principle or pheno-

> menon of nature), an inquiry must be made into whether the claim is seeking patent protection for that formula in the abstract. A mathematical formula as such is not accorded the protection of our patent laws, *Gottschalk v. Benson*, and this principle cannot be circumvented by attempting to limit the use of the formula to a particular technological environment. *Parker v. Flook.* Similarly, insignificant postsolution activity will not transform an unpatentable principle into a patentable process. To hold otherwise would allow a competent draftsman to evade the recognized limitations on the type of subject matter eligible for patent protection. [*]On the other hand, when a claim containing a mathematical formula implements or applies that formula in a structure or process which, when considered as a whole, is performing a function which the patent laws were designed to protect (e. g., transforming or reducing an article to a different state or thing), then the claim satisfies the requirements of 101. Because we do not view respondents' claims as an attempt to patent a mathematical formula, but rather to be drawn to an industrial process for the molding of rubber products, we affirm the [validity of the patent].

The axis of the *Diehr* ruling is that "On the other hand" phrase on the starred line ("[*]"). The lines before stated software loaded onto a general-purpose computer is not what patents are about, and the line after the star states that when a device is transforming matter and curing rubber, then that's a real live machine.

So far, it's all wine and roses: we have a specific discussion of what can be claimed, and it excludes pure software on the one hand but allows a machine with task-specific code to be patented as a machine.

Omitting a ruling or three, the Court of Appeals for the Federal Circuit then took over. I have nothing nice to say about the CAFC, so I won't say anything. But these are the guys who ruled on *In re Alappat*, which I mention as the single thorn in our campaign's side.

*In re Alappat* was yet another case over a software device, and the court used only the second half of the above *Diamond v. Diehr* ruling—"patent laws were designed to protect . . . transforming or reducing an article to a different state or thing . . . " and pointed out that programming a stock computer with software creates a "new machine." After all, the memory on the computer is transformed to a different state. That's why we computer geeks call it a state machine.

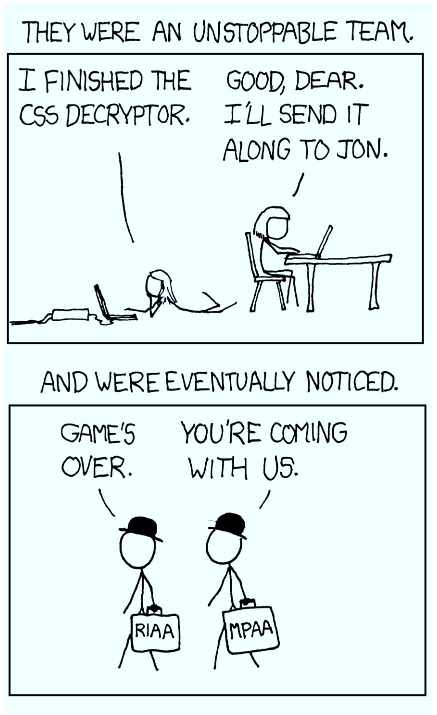
In other words, they threw out the entire debate about distinguishing between a general-purpose computer or a specialized device, and just said it's all patentable.

So once *In re Alappat* was accepted,

business method patents and tax loophole patents followed naturally. By eliminating *Diehr's* balance between "insignificant post solution activity" and bona fide machines, everything became patentable.

The important part of this narrative is that *In re Alappat* threw out the question of what is a general-purpose or special-purpose computer as just irrelevant and not worth debating.

And that, dear reader, is where software patents come from. And that is why we must end software patents.


THEY WERE AN UNSTOPPABLE TEAM.
I FINISHED THE CSS DECRYPTOR.
GOOD, DEAR. I'LL SEND IT ALONG TO JON.
AND WERE EVENTUALLY NOTICED.
GAME'S OVER.
YOU'RE COMING WITH US.
RIAA
MPAA

## AntiFeatures

***by Mako Hill***
***Board of Directors***

In 1996, a furor erupted over Microsoft Windows NT. At the time, Microsoft was selling two versions of its popular operating system: NT Workstation (NTW) and NT Server (NTS). NTS cost roughly $800 more than the NTW. While NTS included a series of server applications not bundled with NTW, Microsoft maintained that the NTW and NTS operating systems themselves were, "two very different products intended for two very different functions." NTS, Microsoft claimed, was suited and tailored for use as an Internet server while NTW was grossly inadequate. Aiming to enforce this difference, both the NTW code and the license agreement restricted users to no more than ten concurrent TCP/IP (i.e., Internet) connections; NTS remained unlimited.

Many users noticed that NTW and NTS were very similar. Digging further, an analysis published by O'Reilly revealed the kernel, and in fact every binary file included in NTW, was *identical* to those shipped in NTS. The sole difference between the two products' cores lay in the operating systems' installation information. The server version contained several options or flags that marked it as either NTW or NTS. If the machine was flagged as NT Workstation, it would disable certain functionality and limit the number of network connections. Billed as a simple web server useful for personal use, Microsoft's Personal Web Server (PWS) included on NTW was, in fact, identical to the company's enterprise-focused IIS shipped with NTS; when IIS started on a machine flagged as

NTW, it would proceed to eliminate and limit functionality. On NTS where flags were set differently, new functionality would appear, and limitations—including the ten-connection ceiling—disappeared.

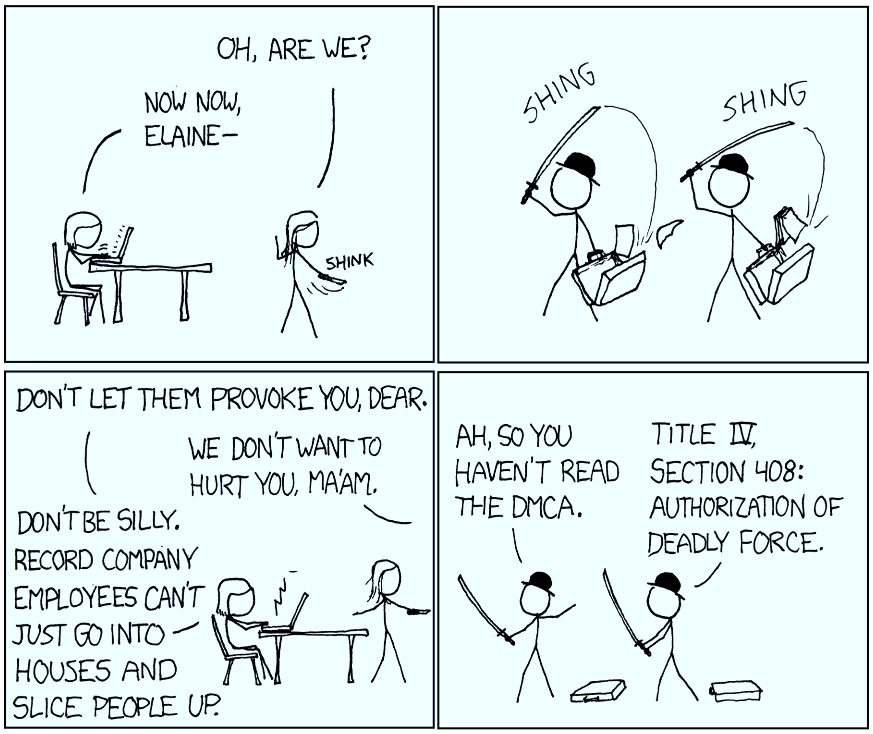
The ability to run an unlimited number of connections, as well as many of the other characteristics that differentiated NTW and NTS, are what I call *antifeatures*. An antifeature, in the way I use the term, is functionality that a technology developer will charge users to *not* include. It is more difficult for Microsoft to limit Internet connections than it is to leave them unconstrained, and the limit is not something that any user would request. DRM and Treacherous Computing systems are, in many ways, extreme examples of antifeatures. Users don't want either and they are hugely expensive and extremely difficult for developers to implement.

Region-coded DVDs, copy-protection measures, and Apple's optional DRM music store—where users initially paid more for the DRM-free tracks—are also excellent examples. It takes a large amount of work to build these systems and users rarely benefit from or request them. Like blackmail, users can sometimes pay technology providers to not include an antifeature in their technology.

But sometimes, as in the case with many DRM systems, users cannot pay to turn their antifeatures off at all! An example of such an antifeature can be seen in the fact that Mozilla and Firefox were blocking pop-ups for years before Microsoft got around to adding the feature to its competing Internet Explorer browser. Despite the fact that Firefox has become fancy about pop-up blocking recently, simply not showing pop-ups (i.e., the way the feature was originally implemented in Mozilla and celebrated by users) is easier than showing them. Microsoft held back not because it was difficult, but because others parts of Microsoft, and their partners, used and made money from pop-ups. Ultimately, Microsoft lost droves of users to the free alternative that was willing to put users first. Until 2005, another proprietary web browser, Opera (which offered pop-up blocking before Firefox did) displayed an irremovable banner advertisement unless users paid for premium version of the software. No users liked the banners, and obviously, it's more difficult to show advertisements than it is to leave them out.

Unfortunately, for the companies and individuals trying to push antifeatures, users increasingly often have alternatives in free software. Software freedom, it turns out, makes antifeatures impossible in most situations. Microsoft's predatory NT pricing is impossible for GNU/Linux, where users can program around it. A version of Firefox funded by advertisements would be too—users would simply build and share a version of the software without the antifeatures in question. Ultimately, the absence of similar antifeatures form some of the easiest victories for free software. It does not cost free software developers anything to avoid antifeatures. In many cases, doing nothing is exactly what users want and what proprietary software will not give them.[1]

[1] `web.archive.org/web/20010708214136/www.zdnet.com/eweek/news/0826/26lic.html` `ftp.ora.com/pub/examples/windows/win95.update/ntwk4.html` `www.oreilly.com/news/differences_nt.html` `tim.oreilly.com/articles/10-conn.html`


NOW NOW, ELAINE–
OH, ARE WE?
SHINK
SHING
SHING
DON'T LET THEM PROVOKE YOU, DEAR.
WE DON'T WANT TO HURT YOU, MA'AM.
DON'T BE SILLY. RECORD COMPANY EMPLOYEES CAN'T JUST GO INTO HOUSES AND SLICE PEOPLE UP.
AH, SO YOU HAVEN'T READ THE DMCA.
TITLE IV, SECTION 408: AUTHORIZATION OF DEADLY FORCE.

## The GNU AGPL

***by Brett Smith***
***Licensing Compliance Engineer***

We recently published the GNU Affero General Public License, version 3. You can find it at `fsf.org/licensing/licenses/agpl-3.0.html`. It has the same conditions as GPLv3, with an additional term that requires you to make sure that, when you modify the program, it offers its source to users who interact with it over a network.

This is a spiritual successor to the original Affero General Public License, which was based on GPLv2 and also had an additional provision so that people using the software over a network could get its source. However, the new license also offers several improvements over its predecessor. First, all the benefits of GPLv3 exist in AGPLv3 as well: internationalization, protection against tivoization, strong defenses against patent aggression, and so on.

Second, AGPLv3 has a broader scope. The original Affero GPL was designed primarily for web applications; it stated that users had to be able to "request immediate transmission by HTTP of the complete source code," making it hard to adapt for programs using other protocols. AGPLv3 instead simply requires that the program "prominently offer all users . . . an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server. . . " This is a flexible requirement that any networkcapable software should be able to handle with relative ease.

Third, AGPLv3's new requirement degrades gracefully. If you use AGPLv3-covered code in a program that doesn't interact with users over a network, there are no additional obligations for you to meet. You can share and modify the program under the same conditions that apply to GPLv3-covered software.

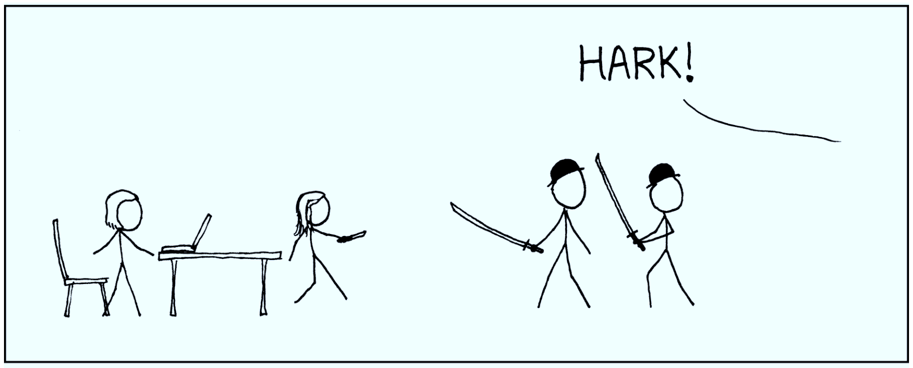
Finally, you can combine modules released under AGPLv3 and GPLv3. Programmers who want to use AGPLv3 for their own work can take advantage of the many libraries and other source files available under GPLv3. And developers working on GPLv3-covered projects will often be able to use the AGPLv3-licensed modules with minimal hassle as well, since AGPLv3's additional term has no requirements for software that doesn't interact with users over a network.

This isn't a complete solution for bringing freedom to web services. Unfortunately, the current architecture that we use to build network applications is inherently unfriendly to users' freedom. Generally, all the code and data lives on someone else's server, under their control and out of yours. You can never be completely free in such an environment.

However, AGPLv3 advances a vision of how freedom works in the computing environment of today and tomorrow. Before this, GPLed web applications provided a lot of good for the community despite their limitations. Even if you can't duplicate a whole site, being able to duplicate its functionality can be helpful. There's only one Wikipedia, but the free software MediaWiki code also runs Citizendium, Conservapedia, and the GPLv3 Wiki. After SourceForge released their code as free software, people started creating alternatives to the site, some of them more featureful than the original.

AGPLv3 takes that one step further by putting developers on a more level playing field. One of the reasons the GPL has been such a successful free software license is because it creates an environment where people feel comfortable contributing—they know that everybody else will play by the same rules. By making sure that developers at least have access to source, AGPLv3 will similarly enable people to create communities around network-centric applications that are focused on freedom.

A big conversation about freedom on the network is beginning. We think that AGPLv3, and the code that uses it, will add a lot to the discussion, but that's not going to be the end of it. We're looking forward to new technology that can provide the benefits of web services but keep control in users' hands. We also want to see what community norms take shape around this kind of software—what you all think deployers ought to do, and what it would be nice of them to do. There's a lot of exciting work ahead, and we hope you'll join us in it. Together, we can build software that gives its users both convenience and freedom.



## Interview: Rob Myers

***Interviewed by Matt Lee***
***Campaigns Manager***

Rob Myers is an artist, hacker and free culture activist who has been remixing images and coding up art for more than fifteen years. He held the first solo all-copyleft-licensed art show, has been involved with community projects such as Free Culture UK and Remix Reading, and has advised on corporate projects such as "Where Are The Joneses?" and "4Laughs."

**Matt**: Rob, you are known to many for your commentary on free culture, both on and offline. I imagine you interact with this community a lot, what is that like?

**Rob**: It's always good to find out how free culture touches on different people's interests, not just activists'. One of the best discussions of music copyright I've had was with the driver of a bus I was on who turned out to be a Beatles fan and knew about the George Harrison "My Sweet Lord" case. Another time I spoke to an artist at her private view who'd almost lost her funding for that show because the funding body wanted her to track down every single copyright holder and get permission from them for all the found video footage she'd used.

People have different views on how free culture should be approached, and discussing my own views with others is a way of learning about the issues and keeping honest. Finding out how things are different around the world helps to keep a sense of perspective as well. When I talked about Creative Commons licenses in Serbia, people just laughed and told me that copyright didn't apply there. In the UK the activity around the Gowers Review Of "Intellectual Property" was very positive.

**Matt**: As an author, both online as well as in occasional printed form, what are your thoughts on eBooks and eBook readers like the Amazon Kindle?

**Rob**: As someone who has been through art school and designed books and other materials for print, I really appreciate the different qualities of paper and ink. I love finding an old Art & Language monograph in a book stall. But I also enjoy being able to get classic texts from Project Gutenberg and read them or search them or feed them into a Markov chain. And having reference works in electronic format is very convenient. I think eBooks complement and extend what is possible with physical books rather than replacing them entirely.

What breaks the commercialization and wider adoption of eBooks, the same as it did for digital music formats, is DRM. I've owned physical books with anti-photocopying printing or with restrictive licenses printed in them and that is bad but it is nothing compared to DRM.

Kindle, as well as being a horrible piece of industrial design with a disempowering user experience, is encumbered with DRM. I just hope that Amazon follow the same path, away from DRM, that Apple seems to be following. But that will take competition—and with a few notable exceptions, the other eBook vendors all seem to be using DRM as well.

I want DVD-style special editions of print books as eBooks, I want the potential of hypertext to be realized, and I want a good platform for generative literature all on something like a Kindle but with industrial design by IDEO and software by GNU. I don't want my old ASCII files uploaded to Amazon, made useless by the addition of DRM, then downloaded to something that I'd be embarrassed to be seen on the tube with.

**Matt**: What do you think of tivoization? What does it mean for free culture?

**Rob**: I don't understand people, usually developers, who want to privilege Tivo's ability to modify software that wasn't theirs to start with over the freedom of the actual users of that software to do the same. If those developers had been in the position of these users when they wanted to develop the software that they are happy to see tivoized, they would not have been able to. I'm very glad that GPLv3 tackles tivoization.

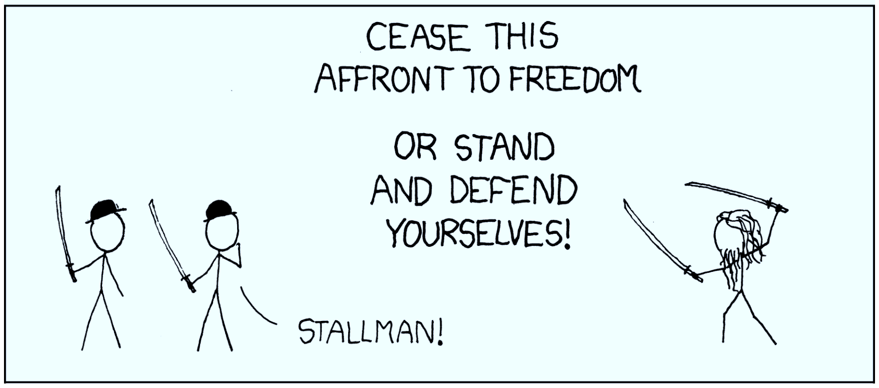
Tivoization can lock you out of access to cultural work and personal memorabilia that you own or have created. Its legal or technological form is unimportant when considering its practical effects. This is true of DRM as well. Those within the free culture movement can and should tackle these issues alongside those within the free software movement.

**Matt**: You've written a few pieces of free software for artists—most of these are now released under version 3 of the GPL. What's the hold up on Minara?

**Rob**: I regard software freedom as a principle to be protected rather than as the product of a particular license. GPLv3 seems to be an effective and proportionate response to challenges to that principle that have emerged over the last decade or so. Arguing that tivoization or DRM are not or should not be part of the GPL's remit does not make sense: they are threats to the freedom of software that didn't exist a decade ago in much the same way that the threats to older versions of the GPL tackled didn't exist forty years ago.

Minara (Minara Is Not A Recursive Algorithm) is a programmable graphics program editor. A kind of Emacs for vector graphics. The program, its tools, and the images it edits are all written in user-editable Scheme. I used some Scheme code from a third party that seems to be GPLv2 only and I need to find the author of that code to relicense it. The moral of this is that people need to make it clear that their code is licensed GPL version x or later.

**Matt**: On the subject of the Ama-


CEASE THIS AFFRONT TO FREEDOM
OR STAND AND DEFEND YOURSELVES!
STALLMAN!

zon's Kindle eBook Reader, Mark Pilgrim pointed out that "The Right To Read" was supposed to be a warning, not a design document—what would be your message to authors unaware of the dangers of DRM books?

**Rob**: Don't mistake a protection racket for job security. Find out what authors like Cory Doctorow have to say about the negative effects that DRM has on your relationship with your readers and customers, and, look at their examples of turning unauthorized copying into free promotion. Using copyleft and network effects to drive sales is a strategy that works, Hasbro adopted it for their relaunch of the *Dungeons & Dragons* game books a few years back.

And don't forget that DRM-free music still sells well and is now being called for by music retailers as well as fans.

**Matt**: Finally, as web applications become increasingly common, what dangers do you see ahead, as the possibility of hosted graphics and design software becomes ever closer? Do you think the newly released GNU Affero GPL will help?

**Rob**: Software as service may be the next great threat to software freedom. It is tivoization for the web. Web 2.0 may just be Google's aftermarket—"economic heat noise" as Eben Moglen accurately described it. But the trend to utility computing and The Cloud means that not just executable software but data will be locked away outside of users' control.

As an artist or author, I wouldn't want to use hosted software or store my work primarily on a hosted service. If there is an earthquake or an economic slump near my data center, I don't want to lose my work.

AGPL will help with access to software, but not with access to data or cultural work, at least not directly. Common formats, web services, and user rights charters are needed as much as access to code—which you will need to pay for access to The Cloud to run anyway. Free culture can help both to make the case for this environment and to provide and protect content for participants in it.

## Netflix Mini-Letter.

If you are a Netflix customer, we encourage you to cut out the mini-letter letter on page two, sign-it, and ship it off with your next return-DVD. You can find more mini-letters at `DefectiveByDesign.org`.

**Special Thanks** to Randall Munroe, the author of *xkcd*, for allowing us to use and rearrange *1337: Part 4*. See it all in one piece at
`http://xkcd.com/344`.

*This bulletin was produced using only free software: LaTeX, Emacs, Inkscape, GIMP, Imagemagick and Ghostview, markdown, gnuhtml2latex, and Perl.*

## How to Contribute

**Associate Membership**: Become a "card-carrying" associate member of the FSF. Benefits include a complimentary FSF Privacy USB Key or a copy of *Free Software, Free Society*, plus bootable membership card, and e-mail forwarding. 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`.

**Phone**: You can also make a credit card contribution by calling us at +1-617-542-5942.

**United Way**: As a 501(c)(3) tax-exempt organization, the FSF is eligible to receive United Way funds. On the donor form, check the "Specific Requests" box and include the sentence, "Send my gift to the Free Software Foundation, 51 Franklin Street, 5th Floor, Boston, MA 02110".

**Buy GNU Gear**: Order one of our T-shirts, caps or manuals at `order.fsf.org`. You can even get a copy of *Free Software, Free Society* signed by Richard Stallman!

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

**Contact: `donate@fsf.org` for more information on supporting the FSF.**