The monthly GNU column

BRAVE GNU WORLD

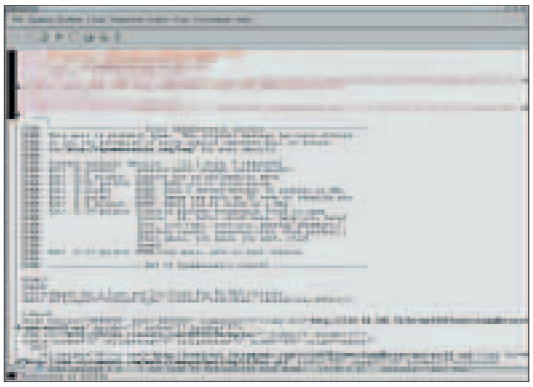
Welcome to a new issue of Brave GNU World. As promised at the end of the last issue, this month Georg CF Greve would like to introduce a project that has made his life much easier

SpamAssassin

The SpamAssassin by Justin Mason allows the "assassination" of spam in your incoming email – at least it marks the spam and so allows Procmail or the mail reader to handle spam in the least annoying way for the user.

The heart of the SpamAssassin is a Perl program distributed under the same dual GNU General Public License/Artistic License as Perl itself. This made it possible to distribute the SpamAssassin through the "Comprehensive Perl Archive Network" (CPAN) and reuse code from it without any legal problems.

Licensing issues have been a crucial part in the beginning of this project, by the way. Before he wrote the SpamAssassin, Justin used another mail filter written in Perl, which became a problem because of its static rules and unclear license situation. From this project Justin adopted the idea of working with scores, a concept very similar to the "Adaptive Scoring" employed by the GNUS News- and Mailreader.



Configuring Smap Assassin is just a text editor away

The SpamAssassin works by applying many different tests to the email it parses. There are tests for HTML-only mail, whether mail contains often-used spam-phrases, whether a mail claims not to be Spam according to certain laws and regulations, whether it contains an unusual amount of exclamation or question marks, talks about "Millions of Dollars" and so on.

For every test that is triggered, an email collects points; how many points each test scores can be specified by the user in a rather simple ASCII configuration file. If the sum of all scores passes a certain – also user-definable – threshold, the SpamAssassin judges that the mail is probably spam.

Based on this decision, the SpamAssassin inserts header flags informing about the user about the test results. If the user so wishes, the spam emails are also forced to have Content-Type "text/plain," which makes it much easier to later check the results. Also the SpamAssassin can insert a more detailed test report at the beginning of the mail, so a user can easily see why a mail has been rated as spam.

The biggest potential risk in using SpamAssassin is clearly "false-positive" results – regular, normal email falsely classified as spam. Therefore it is recommended you regularly take a look at the spam folder, which is where all detected spam should normally go, in order to rectify false-positive results.

You can also choose to lower the sensitivity of the SpamAssassin, which will increase the amount of undetected Spam. Finding the proper balance is the tricky part for the SpamAssassin administrator. To prevent spammers from finding ways to bypass the SpamAssassin tests, the project incorporates as many different tests as possible and is also easily extensible. Of course it also supports the online-blacklists. Standard DNS-blacklists referencing known sources and relays for spam are supported, as is Vipul's Razor, a database allowing identification of known spam. In order to allow easy filtering of large amounts of mail and connections to as many mail-sources as possible, Craig Hughes wrote the *spamd* daemon, which comes with the SpamAssassin package.

The biggest weakness of the SpamAssassin is that it is more or less targeted at the technically experienced user and does not (yet) have a graphical user interface. Fixing this, as well as writing more tests and creating more bindings to mail sources, is the focal point of further development. Currently available are bindings to Procmail, Qmail, Postfix, Sendmail through the Milter library and a Mail::Audit plug-in.

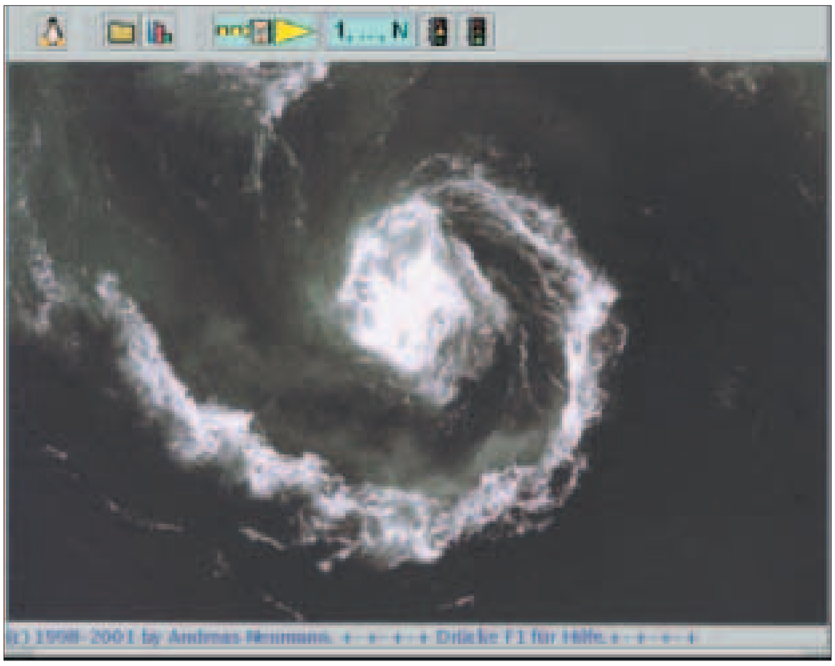
I hope to be excused for mentioning that the sendmail-milter plug-in was written by myself after an unsuccessful search for existing solutions, so I could use the SpamAssassin to filter all incoming mail. Lack of time on my side forbids me from maintaining the project properly, however, so Michael Brown, whose company employs/offers it in a commercial environment, has taken over as the maintainer in the best Free Software tradition.

This is a nice example of how Free Software can harmonise the classic "scratch your own itch" approach with commercial interests of a company for the benefit of all users.

Facing an increasing flood of spam that threatens to bury the Internet beneath it, I have to admit I hold great sympathy for projects like the SpamAssassin, which gets rid of about 60 spam emails a day for me.

Voxximate

Regular readers of the Brave GNU World should by now be pretty familiar with many of the arguments for Free Software in the scientific field. Ultimately, Free Software is the only sensible long-term choice


1, ..., N
(c) 1998-2001 by Andreas Neumann. + - + - + Drücke F1 für Hilfe + - + - +

Vortex in action

for all kinds of scientific work, because only Free Software can offer the guarantee that it will remain useful for future projects and can be included alongside scientific results i.e. publication as part of one's work.

Voxximate by Andreas Neumann is such a scientific Free Software project under the GNU General Public License. Voxximate stands for "Vortex flow simulation made at home", and it is a program for the simulation of currents/vortices in fluids. The program works based on predefined starting positions and uses concrete steps to calculate the influence of all vortices on all other vortices, tracking the development through time.

The project is probably most useful for students facing fluid dynamics at some point in their studies, who are interested in studying how vortices interact and build structures.

Voxximate was written in Java, which brings the usual Java problems, but this should not keep anyone from supporting further development. For the next steps, Andreas hopes to include a graphical editor to define starting positions and capabilities to save graphics and animations that can then be published on the Web.

Monica

Monica is a monitor calibration program by Tilo Riemer. It was written in C++ and uses the Fast Light Toolkit (FLTK) and the xgamma program of XFree86. If a monitor's gamma correction is wrongly set that it can make it impossible to distinguish between colours that lie close together, or create an unsatisfactory impression of the colouring schemes. If the computer is used for graphical work then this can be particularly problematic.

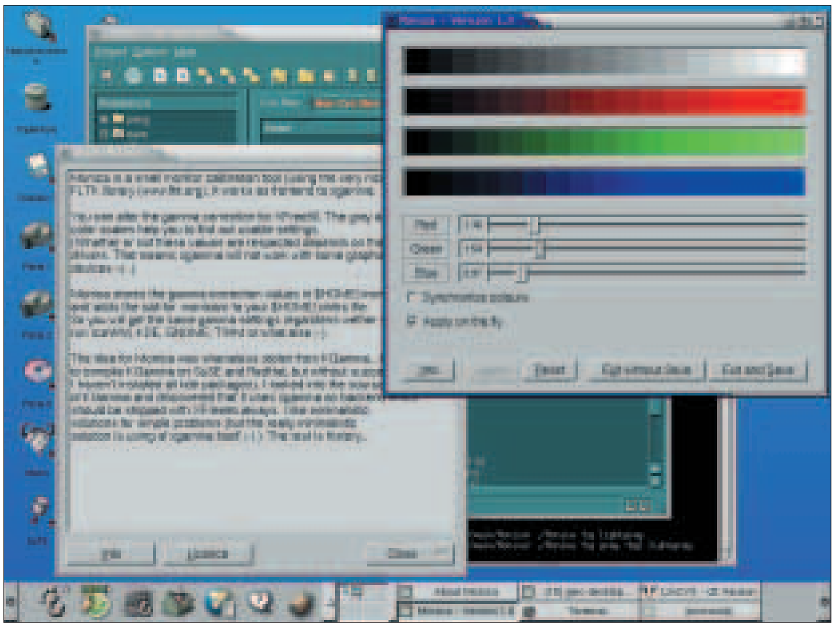
Initially, Tilo Riemer tried to use the related project KGamma, but failed to compile it because several KDE libraries were missing and Kgamma also seemed to be so deeply embedded in KDE that it needs large parts of KDE to work. So in January 2002 he began writing Monica, which has the advantage of being very small and fast. This enabled the inclusion of an "on-the-fly" mode, making dynamic feedback possible. On a 900MHz computer this needs about 10-20 per cent of the CPU time.

Further strengths of Monica are an absence of dependencies beyond the FLTK and the policy to save changes in the user's .xinitrc to make them independent of the window manager or desktop.

The recent release of version 1.0 indicates that Tilo does not plan to invest a lot more time into Monica, although he would welcome efforts to internationalise it. Originally, Tilo planned to release Monica as "Public Domain," since it seemed too small and insignificant to warrant thinking about licenses. Into the sourcecode he wrote "Copyright © Tilo Riemer" though, without further thinking about it.

The notion of Public Domain isn't totally unproblematic in continental Europe, however. In Germany, the standard legal interpretation of the term is free of authorship/copyright claims, which usually means either the author is unknown or dead for more than 70 years. Both cases clearly did not apply. Therefore Tilo decided to publish Monica under a BSD-like license, solving the immediate problems and making Monica Free Software.

This scenario is not uncommon and demonstrates that developers obviously don't like thinking about licenses very much, although it is very easy to introduce insecurities when not doing so. Therefore I will try to give an understandable introduction into the legal maintainability of Free Software.


Calibration made easy

Legal maintainability of Free Software

Most people are aware that a large part of all software requires permanent technical maintenance or it will quickly lose its usefulness. Often only software that is permanently maintained can be employed over longer periods of time.

This per se technical procedure depends on access to the source code and the right – i.e. the freedom – to perform the maintenance. Generally speaking, defining the rights and obligations of every member of society is what the political/legal system does.

Whether the legal system is working well or not isn't the important point. What one should realise is that some of the prerequisites for technical maintainability are of legal nature.

Particularly in a commercial environment, the guarantee of permanent and lasting maintainability is one of the seminal advantages for Free Software. This advantage depends strongly on the legal maintainability of Free Software.

The freedoms, rights and obligations of Free Software are granted and sometimes protected through licenses, which are "anchored" to the software through the copyright of the author. Free Software does not strictly depend on copyright law to work, but since copyright law exists, we need to deal with it.

What does legal maintainability mean in this context?

Even if this is not how most people percieve it intuitively, the legal system is not static, it is ever-changing.

Changes affecting the copyright law could, as was recently the case in Germany, potentially weaken or even outlaw Free Software. In this specific case, ifrOSS and FSF Europe were capable of suggesting a change of the proposed copyright law revision, introducing an exception for Free Software. This change made it into the law in the original form suggested by the ifrOSS and became part of the law passed on 25 January 2002 that will be enacted soon.

One of the tasks of the FSF Europe is to keep looking for such developments and influence them in a positive way for Free Software. Without cooperation with organisations like the ifrOSS, which is clearly entirely legal in nature, doing this would be much harder; which is why the FSF Europe works on establishing and strengthening cooperation throughout Europe.

It would also have been possible that changes in other legal parameters would have required an adaptation of the licenses. Legal changes or new technical concepts, like "Application Service Providing" (ASP), could possibly bypass the protection of freedom in some areas or even render it ineffective, effectively violating the spirit of the licenses.

Most developers publish their software under the GNU General Public License, conscious of, by doing so, having secured and protected the freedom of their software. When doing so, the most important step towards securing Free Software has already been taken. By employing the "or any later version" clause the FSF is also partially empowered to globally protect, defend and maintain the licensing under the (L)GPL.

Sometimes it may become important to explicitly change a license, however. Projects that have not established a central maintenance of legal rights can get into serious trouble in such a case, especially when the "or any later version" clause of the GPL has been removed.

In such a case all developers – assuming they can all be found – have to agree to the change. Given the rather wide spectrum of interests and opinions of the developers working on some projects, this does not seem very likely.

Additionally, in most cases only the holder of the so-called "exclusive exploitation rights" – i.e. the "Copyright" – is legally entitled to enforce the license in court. So projects can run into serious difficulties when trying to represent the interests of the project in court.

Given that many authors are working on a project, they will effectively have to team up and act together in order to protect their individual interests. This requires a lot of coordination, time and effort. Also not all authors are willing or capable of seeing a potentially protracted legal struggle through to the end.

It would be good if more projects became aware of these relationships and took adequate precautions. By appointing a trustee, authors can also get back to improving the software itself.

For the future it seems likely that projects with clear and orderly legal circumstances will have an advantage gaining popularity, since users will quite likely more often pay attention to this.

In order to secure the legal maintainability of Free Software – especially inside the core area of the GNU Project, but not limited to it – the Free Software Foundation has started early to work with the so-called "Copyright Assignments," which empower it to defend the rights of Free Software (even in court, if need be) and adapt the licensing to the changing circumstances.

Since the continental-European Authorship law has a different basis than the Anglo-American Copyright, the FSF Europe has also been working together with Axel Metzger, Carsten Schulz and Till Jaeger of the ifrOSS on a "Fiduciary Licence Agreement," which allows the FSF Europe to act as the fiduciary of the authors.

The author retains an unlimited amount of "single exploitation rights," which can be used by the author to dual-license the software under other (potentially even proprietary) licenses.

At the same time, the FSF Europe guarantees to only use the transferred rights in the interest of Free Software and will only publish the software under a Free license – otherwise all rights fall back to the author.

This agreement is currently in the final internal "review phase" and will be introduced to the public in the not too distant future.

As the president of the FSF Europe I consider the Free Software Foundation to be best-suited for this task as they will continue meeting these challenges with the reliability that the FSF has been known for in long years.

They not only possess the largest knowledge and experience with the GNU General Public License and Lesser GPL, they can act worldwide and have a justified reputation for being able to defend the interests of Free Software; also with legal means, if need be.

Enough for today

That should be enough for today. I hope to have succeeded in the last part to create some more awareness for the background and the tasks and work of the Free Software Foundation. As usual, I'd like to ask for loads of email containing ideas, comments, questions and new projects.

Info

Send ideas, comments and questions to Brave GNU World	*column@brave-gnu-world.org*
Homepage of the GNU Project	*http://www.gnu.org/*
Homepage of Georg's Brave GNU World	*http://brave-gnu-world.org*
"We run GNU" initiative	*http://www.gnu.org/brave-gnu-world/rungnu/rungnu.en.html*
SpamAssassin homepage	*http://spamassassin.org*
Comprehensive Perl Archive Network (CPAN)	*http://www.cpan.org*
GNUS homepage	*http://www.gnus.org*
Vipul's Razor homepage	*http://razor.sourceforge.net*
Sendmail-Milter Plug-in homepage	*http://savannah.gnu.org/projects/spamass-milt/*
Voxximate homepage	*http://voxximate.sourceforge.net*
Monica download	*http://lincvs.sunsite.dk/index.php?order=download,Monica&lan=en*
Fast Light Toolkit (FLTK) homepage	*http://www.fltk.org*
XFree86 homepage	*http://www.xfree86.org/*
KGamma homepage	*http://www.vonostheim.de/kgamma/index.html*
ifrOSS homepage	*http://www.ifross.de*
FSF Europe homepage	*http://fsfeurope.org*