

**FREE
DVD**

Complete Linux Voice Archive!
ALL 32 ISSUES on a Searchable DVD

**THE
STORY
OF THE**

GPL

**LINUX
MAGAZINE**



LINUX

MAGAZINE

JULY 2017

THE STORY OF THE GPL

5 LOG MONITORING TOOLS

An insider's look at the license that changed the world

Stay Safe

2-Factor authentication with Google Authenticator

uMap

Build your own detailed maps

Create a Boot Repair Disk



We Don't Want Money Just Compliance

Former FSF counsel Eben Moglen on life in the front lines

LINUXVOICE

Start Your Own FOSS Project
Managing Systemd Service
LibreOffice Tricks

FOSS Picks
• KDevelop 5.1
• Curses Synthesizer
• Android Emulator

**FREE
DVD!**

**ALL 32 ISSUES OF
LINUXVOICE!**



Issue 200
July 2017
US\$ 15.99
CAN\$ 17.99



WWW.LINUXPROMAGAZINE.COM

RYZE 'N' SHINE

More parallelization.
More power.



Dedicated Root Server AX60-SSD

AMD Ryzen 7 1700X
Octa-Core "Summit Ridge" (Zen)
64 GB DDR4 RAM
2 x 500 GB SATA 6 Gb/s SSD
100 GB Backup Space
30 TB traffic inclusive*
No minimum contract
Setup Fee \$131

monthly \$ **65**

The ideal solution for highly-parallelizable processes.

Our new Dedicated Root Server AX60-SSD houses the latest generation AMD processor, the octa-core AMD Ryzen 7 1700X, which is based on Zen architecture. Its performance is truly impressive, especially when used for purposes such as virtualization, encryption, and data compression, which require several processor cores.

www.hetzner.de/us

* There are no charges for overage. We will permanently restrict the connection speed if more than 30 TB/month are used. Optionally, the limit can be permanently cancelled by committing to pay \$1.30 per additional TB used.

All prices exclude VAT and are subject to the terms and conditions of Hetzner Online GmbH. Prices are subject to change. All rights reserved by the respective manufacturers.

THE LAST FUD COLUMN

Dear Reader,

The news has gotten really weird recently. I suppose politicians have always blurred the line between fact and opinion, but lately things have gotten a little unglued. False facts are accepted without proof. Proven facts are ignored or denied just because they don't happen to line up with someone's preferred narrative.

I turned on my TV the other day and listened to a whole parade of seemingly well-heeled people in suits and professional-looking dresses not just stretching the truth around the edges, but slinging absolute moonshine. The moonshine was raining down – with press secretaries, paid political spokesmen, and unofficial surrogates all serving up the same distilled nonsense in a kind of pre-orchestrated assault on reality.

It all seemed familiar to me – I felt something similar to that eerie effect they call *déjà vu*. Where have I seen this before? I thought. And then I remembered: This is a lot like Microsoft, back in the FUD era.

In those days, when there didn't seem to be any actual technical benefit to using the buggy and insecure Windows system, Microsoft employed its near-monopoly position to spread Fear, Uncertainty, and Doubt, which came to be known by the acronym FUD. "Linux is unsafe." "Linux is sketchy." "Windows is stable and safe for people who don't want to take risks." This company line wasn't just spouted by the CEO – it trickled out through channels: sales managers and marketing execs, consultants, instructors teaching the Microsoft certification curriculum. A whole cozy little false reality was offered up to the masses, helped along by a well-tuned chorus of talking-point talkers who were "just doing their job" by keeping to the company line.

I've never been much into using this space for Microsoft diatribes – not that they don't deserve it, but it is just too easy, and it is kind of predictable for a guy writing in a

Linux magazine to say he isn't a fan of Windows. Why am I bringing this up now?

Just last week, the WannaCry ransomware attack tore through the world, locking up 230,000 Windows computers in 150 countries. The most vulnerable targets: Windows XP systems. All those Windows XP systems out in the world today are the final relics of the FUD era. The people running these systems really bought into the myth that Windows XP was safe (it never was) and that they were doing something sensible and practical by doing business with Microsoft (Microsoft stopped supporting XP three years ago and did not provide a viable upgrade path for older systems).

Microsoft is on to new things now – they support Linux and have even joined the Linux Foundation, but the legacy of the FUD remains. Plans change, companies revisit strategies, but FUD is a time bomb planted in the past. Shouldn't we all agree that we're not going to ignore facts and we won't invent false facts just to win arguments and get more money? Lets keep the future FUD-free.



Joe Casad,
Editor in Chief





WHAT'S INSIDE

This month we celebrate our **200th issue** with a special look at the great GNU Public License.

Other Highlights:

- **Log Monitoring Tools** – We look at LOGalyze, Logcheck, Logwatch, MultiTail, and Swatchdog (page 34).
- **Two-Factor Authentication** – A single password isn't enough if you really want to keep your data safe (page 54).

Over at Linux Voice, you'll pick up some LibreOffice tricks and learn to scrape the web with Common Crawl (page 69).

SERVICE

- 3 Comment
- 6 DVD
- 96 Featured Events
- 97 Call for Papers
- 98 Preview

NEWS

08 News

- Linux Kernel 4.11 Fearless Coyote out
- Docker appoints Steve Singh CEO
- New Docker projects: Linux Kit and Moby Project
- Ubuntu returns to Gnome as its mobile plans shatter
- Intel's serious remote management problem
- The Linux Foundation and Microsoft offer IoT platforms

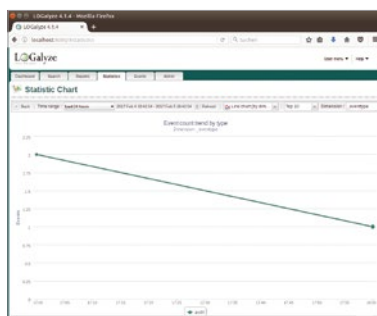
12 FOSDEM

We bring you the view from the street at Europe's largest FOSS community event in Brussels, Belgium.

REVIEWS

34 Five Log Monitoring Tools

Lightweight monitoring tools can sniff out threats as easily as the larger alternatives, and they take much less time to set up.



COVER STORIES

14 Copyleft

The GNU General Public License was born of the simple idea that freedom matters.

20 For Want of a Printer

An excerpt from the book *Free as in Freedom 2.0: Richard Stallman and the Free Software Revolution*.



28 Meet Free Software Pioneer Eben Moglen

We asked former lead counsel for the Free Software Foundation, Eben Moglen, about the legal basis for the GPL's famous copyleft protection and the long, steady effort to tell the world about free software.



IN-DEPTH

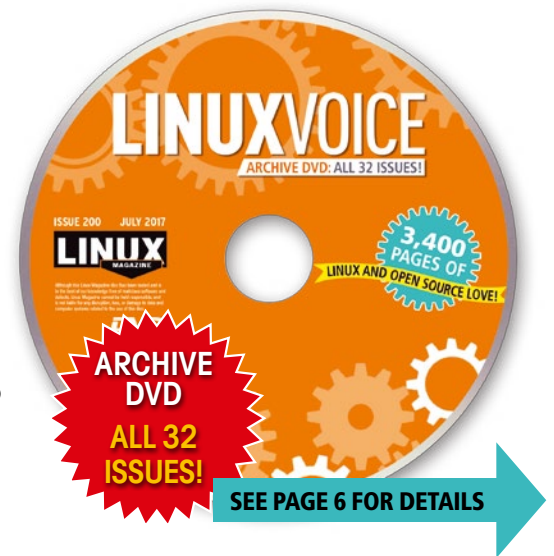
- 40 Professor Knopper's Lab – Yocto Knoppix**
The professor sets the stage for Knoppix on the Raspberry Pi.
- 44 Boot Repair Disk**
If things go wrong when installing an operating system on a hard drive or solid-state disk, a boot repair disk can get your boot configuration back on its feet.



- 48 Programming Snapshot – Pushover**
We apply the principles of home security to a login attempt on your SSH server.
- 52 Charly's Column – Pi-hole**
Charly blocks garish advertising for all computers on his network centrally with the Pi-hole tool, which is not just for Raspberry Pi devices.
- 54 Two-Factor Authentication**
Add an extra layer of protection with one-time passwords.
- 58 uMap**
Create advanced maps with tools and resources from the OpenStreetMap service.



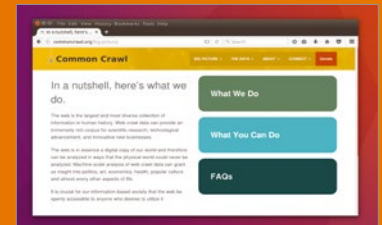
- 62 Crowdfunding**
A look at some of the latest open hardware projects.
- 64 Command Line – Pinning Sources**
Debian discourages the use of pinning to set preferences for package repositories, because the practice can have disastrous results.



SEE PAGE 6 FOR DETAILS

LINUXVOICE

- 69 Welcome**
The unstoppable march of progress.
- 70 Simon Phipps**
It's not enough to tinker with copyright rules; the whole concept needs reviewing for the digital age.
- 71 Unity Farewell**
Choice is good? We've chosen.
- 72 Start Your Own FOSS Project**
Don't just consume Free Software – contribute to it! We share the tips and tricks required to start a successful FOSS project.
- 75 Doghouse**
What are the various factors that contribute toward creating good code?
- 76 FAQ – Common Crawl**
Download the entire web to kick-start a data science empire.
- 77 Core Technologies**
Ever wondered what's happening inside a virtual machine? Join us for an exciting dive into the deep waters of virtualization.
- 78 FOSSPicks**
KDevelop 5.1, Riot.im, Cursynth, QMapShack 1.8, and more.
- 90 Tutorials – systemd**
Take control of the services running on your Linux machine.
- 92 Tutorials – LibreOffice**
Discover some hidden and lesser known features in LibreOffice to help you work faster and smarter (and gain extra geek points).



On the DVD

LINUXVOICE

ARCHIVE DVD: ALL 32 ISSUES!

ISSUE 200 JULY 2017

LINUX
MAGAZINE

3,400
PAGES OF
LINUX AND OPEN SOURCE LOVE!

LINUXVOICE

ARCHIVE DVD: ALL 32 ISSUES!

3,400 pages of Linux and Open Source love!

In the beginning, the founders of Linux Voice – Graham Morrison, Andrew Gregory, Ben Everard, and Mike Saunders – believed that “passionate communities deserve passionate magazines,” and that’s what Linux Voice has delivered to the Linux and Free Software communities. Since April 2014, Linux Voice contributors have introduced you to the very best that Free Software has to offer. Now all that knowledge is available in a single, searchable DVD archive.

The Linux Voice Archive includes EPUBs, HTMLs, and PDFs of every issue published, so you can catch up on all the news, reviews, and tutorials you missed on the newsstand. You’ll receive a deeper understanding of Linux and FOSS on the FAQs pages, share Simon Phipps’ analysis of open source news, discover Linux gaming, understand changes and trends in Linux distros, and find thoughtful software reviews.

If you love Linux and open source software and communities, you’ll love Linux Voice, and you won’t want to miss the opportunity to browse this comprehensive DVD of issues 1 through 32.



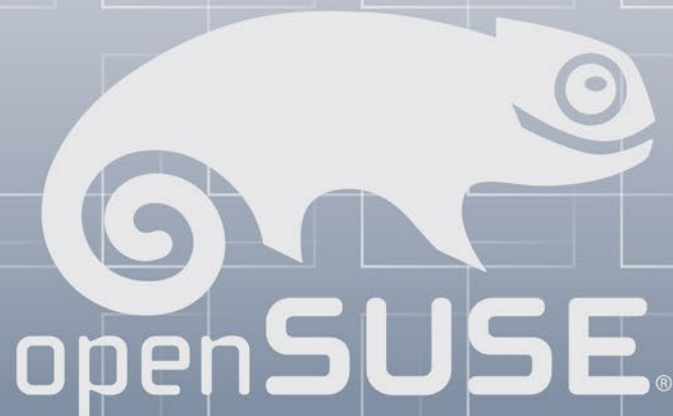
Defective discs will be replaced. Please send an email to subs@linux-magazine.com.



**open
build
service**

**A generic system to build
and distribute packages
from sources in an automatic,
consistent and reproducible way**

openbuildservice.org



NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

08 Linux Kernel Release

- Fearless Coyote, Linux Kernel 4.11 is Out
- Docker Appoints Steve Singh CEO

09 New Docker Projects

- Docker Announces Linux Kit and Moby Project
- Ubuntu Returns to Gnome as Its Mobile Plans Shatter

10 Intel AMT/ISM/SBT Flaw

- Intel Has a Serious Remote Management Problem
- The Linux Foundation and Microsoft Offer IoT Platforms

Fearless Coyote, Linux Kernel 4.11 is Out

Linus Torvalds announced the release of Linux kernel 4.11, code-named Fearless Coyote. The new release comes with performance improvements and support for additional hardware.

Announcing the release, Torvalds wrote on Linux Kernel Mailing List (LKML), "We still had various smaller fixes the last week, but nothing that made me go 'hmm..'. Shortlog appended for people who want to peruse the details, but it's a mix all over, with about half being drivers (networking dominates, but some sound fixlets, too), with the rest being some arch updates, generic networking, and filesystem (nfs [d]) fixes. But it's all really small, which is what I like to see the last week of the release cycle."

Version 4.11 brings full support for DisplayPort MST on Intel video cards, allowing audio out to the monitor connected through the DisplayPort. According to kernelnewbies.org, this release also makes the swap implementation more scalable, making it more suitable for use with modern storage devices, which is going to help cloud providers who tend to overcommit memory more aggressively and fit more VMs to a platform with a fast swap device.

This release also introduces support for the implementation of the "Shared Memory Communications-RDMA" (SMC-R), an IBM protocol that provides RDMA capabilities over RoCE transparently for applications exploiting TCP sockets.

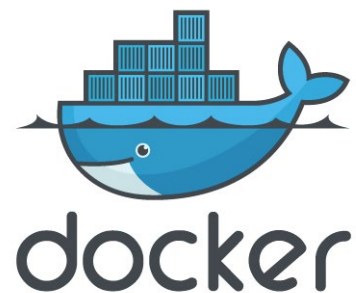
Docker Appoints Steve Singh CEO

Docker has appointed Steve Singh as the new CEO of the company, after Ben Golub stepped down from the position. Singh will join Docker's board of directors as Chairman. Singh is known for co-founding Concur Technologies, a SaaS company that was acquired by SAP for \$8.3 billion.

Singh comes to Docker at a time when Docker's container technology is enjoying phenomenal success, but Docker as a company is still trying to build a business model around open source technologies.

In an exclusive interview, Steve Singh told me that innovation will be one of the key areas for him to focus on as the new CEO of Docker.

In a press release, Singh said "I look forward to working with the entire Docker team as well as the broader Docker community to help usher in the new era of application computing. I believe that Docker, backed by its platform and fueled by innovation from its collaborative ecosystem, is on track to be the next great enterprise software company."



Docker Announces LinuxKit and Moby Project

Docker Inc., the company behind the Linux container project with the same name, announced two new projects at DockerCon called LinuxKit and Moby Project.

LinuxKit is essentially a toolkit that allows organizations to build their own containerized Linux subsystems. A Docker press release stated: "LinuxKit allows users to create very secure Linux subsystems because it is designed around containers. All of the processes, including system daemons, run in containers, enabling users to assemble a Linux subsystem with only the needed services. As a result, systems created with LinuxKit have a smaller attack surface than general purpose systems. It also provides a read-only root filesystem for an immutable infrastructure approach to deployments enabled by InfraKit. LinuxKit will have a community-first security process and will serve as an incubator for security-related innovations like WireGuard and Landlock."

Docker itself is using the kit to build different editions of Docker, such as Docker for Mac, Docker for Windows, and Docker for Cloud. Docker is taking this core component out and releasing it as an open source project. The company is considering donating it to the Linux Foundation.

The second project that the company announced is Moby Project, a framework that helps organizations assemble the new systems using components of their choice.

Solomon Hykes, Founder and CTO of Docker said, "This project will be the most important project at Docker since the launch of Docker itself as it provides the ecosystem with a way to create, share, use, and build container systems in a way that hasn't been possible with any open source project in the past."

Docker is touting Moby Project as the "Fedora" of the container world that the entire industry can collaborate to improve it.

Unlike LinuxKit, Moby Project will remain a Docker-sponsored project.

Ubuntu Returns to Gnome as Its Mobile Plans Shatter

April has been a rough month for Canonical. On April 5, 2017, Canonical founder Mark Shuttleworth announced that they are abandoning their mobile plans. Shuttleworth wrote in a blog post, "I'm writing to let you know that we will end our investment in Unity 8, the phone and convergence shell."

Ubuntu phones and tablets never really took off. I received a review unit of the device and it lacked core features that a user would expect from a mobile device. Canonical didn't manage to get big brands on board, and the fate of Ubuntu Mobile was sealed. According to some media reports, Ubuntu mobile devices will stop getting updates after June this year, although Canonical has not released a public advisory. Official Ubuntu Phone and Tablet pages are now showing a 404 error instead of relaying a message about funding the project.

With Unity out of the way, the 18.04 release of Ubuntu will return to a complete Gnome desktop, including the shell. The good news is that an official Gnome flavor of Ubuntu, called Ubuntu Gnome, already exists.

Henceforth, there will be no separate flavor of Ubuntu Gnome. Jeremy Bicha, an Ubuntu Gnome developer wrote in a blog post: "The development teams from both Ubuntu Gnome and Ubuntu Desktop will be merging resources and focusing on a single combined release that provides the best of both Gnome and Ubuntu."

The Ubuntu Gnome project is working with Canonical teams to figure out how the merging is going to work. It might be prudent for Canonical to move the base of Ubuntu 17.10 to Gnome so that they have enough time to iron out bugs by the 18.04 long-term release.

© Franz Pfluegl, Fotolia.com



MORE ONLINE

ADMIN HPC

<http://hpc.admin-magazine.com/>

How Old Is That Data? • Jeff Layton

The explosion of data is a storage burden that all system administrators bear. The agedu tool lets you discover what data is being used.

ADMIN Online

<http://www.admin-magazine.com/>

An Exclusive Interview with openSUSE Chairman Richard Brown • Swapnil Bhartiya

Over the course of three days of SUSECON, we talked to Richard Brown, the friendly chairperson of the openSUSE Board.

Designate Provides DNS as a Service in OpenStack • Martin Loschwitz

The management of DNS entries works fundamentally differently for clouds than for classic setups; OpenStack Designate is a meaningful and well-functioning DNS as a Service implementation.

Putting an Active Directory Domain Controller in the Azure Cloud • Klaus Bierschenk

Once you start putting services in Azure cloud, you might want to add an Active Directory domain controller.

Moving back to Gnome might also mean Canonical will adopt Wayland and drop its own display server Mir, but the company has not relayed a message regarding the future of Mir.

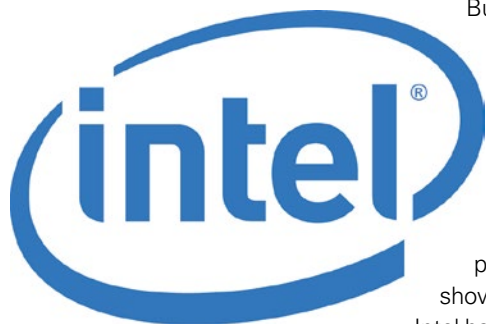
In related news, Jane Silber stepped down from the position of CEO of Canonical as Shuttleworth resumes leadership of the company. Canonical also let go of more than 80 employees, who were working on projects that Canonical is abandoning.

According to some media outlets, Canonical is seeking outside investors. The Register reported: "These investors determined that Canonical was overstaffed and some projects lacked focus."

Intel Has a Serious Remote Management Problem

Researchers have found a critical vulnerability in Intel's remote management technology that allows attackers to gain remote access to millions of systems running on Intel chips.

The vulnerability affects every laptop, PC, and server that has Intel Active Management Technology (AMT), Intel Standard Manageability (ISM), or Intel Small Business Technology (SBT) features enabled. Intel Macs do not ship with the AMT software, so they are not affected.



These remote management features are enabled on many vPro chips and allow IT admins to manage systems remotely across the organization.

Although the system is protected through username and password, Maksim Malyutin, a researcher at embedded security firm Embedi, discovered that the protection can be bypassed using simple tools. He discovered it while reverse engineering AMT, which also shows how important it is to allow reverse engineering of products.

Intel has implemented and validated a firmware update to address the problem and is working with computer makers to integrate the fix with their software.

Intel expects that computer makers will make updates available beginning the week of May 8 and continuing thereafter.

Intel has issued some tips to people and companies using business PCs and devices that incorporate AMT, ISM, or SBT to ensure the security of their system.

The Linux Foundation and Microsoft Offer IoT Platforms

IoT is gaining some serious traction, both in consumer and industrial space. On the open source side of the spectrum, the Linux Foundation has announced EdgeX Foundry project to bring together IoT vendors to collaborate on a common open framework.

The Foundry is so far the biggest industry collaboration, with more than 50 members. "The initiative is aligned around a common goal: the simplification and standardization of Industrial IoT edge computing, while still allowing the ecosystem to add significant value," said the Foundry in a press release.

On the proprietary side, Microsoft has announced Microsoft IoT Central, a new software-as-a-service (SaaS) that's built on top of Azure cloud to make it easier for customers to build IoT solutions.

"Microsoft IoT Central will be available along with our existing platform-as-a-service (PaaS) solution, Azure IoT Suite, which enables deep customization and full control. This new IoT SaaS offering has the potential to dramatically increase the speed at which manufacturers can innovate and bring new products to market, as well as lower the barriers to creating IoT solutions that generate new revenue opportunities and better experiences for customers," Microsoft wrote in a blog post.

Despite being a top tier Linux Foundation member, Microsoft is missing from the list of companies supporting the EdgeX Foundry project. The announcements came at the same time, so it will be interesting to see whether Microsoft will join forces with the Foundry on the IoT front end.

JAX[®] LONDON

Join 1000+
Software
Innovators!

The Conference for Java and Software Innovation

October 9 – 12, 2017 | London

**VERY
EARLY BIRD**
Save up to £200
by July 27

Use
LMcode10
to save an
extra 10%!



Big &
Fast Data



DevOps,
Container &
Cloud



Continuous
Delivery



Agile &
Communication



Java Core &
Languages



Microservices



Software
Architecture
& Design

www.jaxlondon.com

FOSDEM 2017

Equilibrium

Long lines and interesting discussions awaited the visitor at this year's FOSDEM event in Brussels, Belgium. *By Sebastian Grüner and Kristian Kißling*



Apparently, not even the FOSDEM team can guess which topics will attract a large audience and which will not, despite many years of experience. The result: Long lines formed before some rooms at FOSDEM'17, the largest European FOSS community event, which was held in early February 2017.

The fast campus LAN closed the breach for those waiting in line this time, however. The organizers broadcast live streams of presentations from all rooms by building their own content delivery network, proving that FOSDEM is in good shape in terms of hardware. Many of the videos are now online [1].

Buses and trains were running again, in contrast with the previous year, but visitors seem to like FOSDEM the way it is – including the beer event on the evening before the headliner. The organizers can only roughly estimate the number of visitors (about 8,000) because they do not sell tickets.

According to estimates by co-organizer Gerry Demaret, once again more people

visiting since 2005. Introducing innovations is difficult, he notes, with just 15 to 20 main organizers, with which they have achieved a kind of “equilibrium.” The team takes special care to learn from the mishaps of previous years. In general, the FOSDEM organizers adopt an evolutionary approach and do not make any long-term plans.

Speaking Time

Although some visitors remained almost entirely in individual dev rooms, an interested audience struggled with the challenge of visiting all the interesting talks. The program not only made admins and developers happy, it

also offered the rest of the community exciting and entertaining insights. For example, Curl creator and maintainer Daniel Stenberg, describing the side effects of his “fame,” notes that Curl [2] not only runs worldwide on almost any device you can imagine, but his email address also shows up everywhere in the license agreements. Every now and then, he thus receives dubious messages from people who believe he is either a hacker or the inventor of the multimedia system in their Toyota (Figure 1).

visited the event in the current year than in the previous year. Demaret has been part of the organization team, which has done without advertising

since 2005. In-

troducting innovations is difficult, he notes, with just 15 to 20 main organizers, with which they have achieved a kind of “equilibrium.” The team takes special care to learn from the mishaps of previous years. In general, the FOSDEM organizers adopt an evolutionary approach and do not make any long-term plans.

Construction Pain

Richard Brown chose a rather unusual start to his talk by presenting the Windows

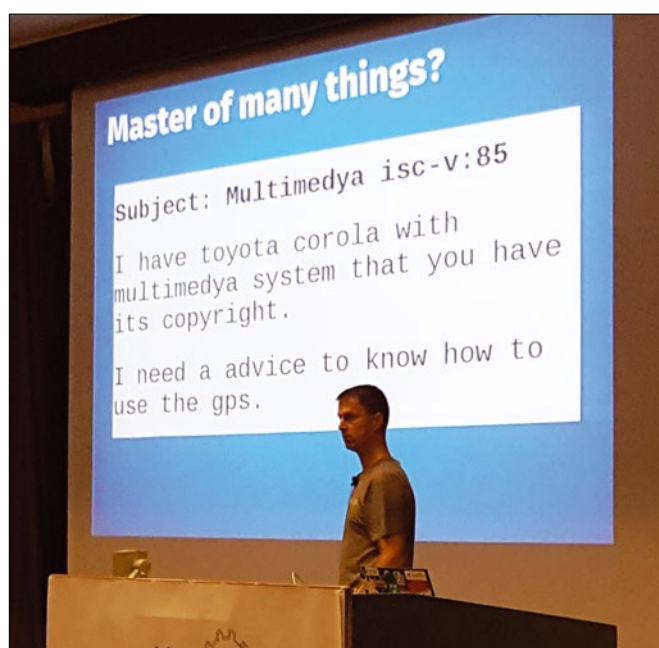


Figure 1: Among other things, a Toyota driver wanted to know from Curl inventor Stenberg how the GPS system in his car worked.

95 architecture as the opening slide. This was the sys admin and SUSE employee's method of pointing out the problems with the new package formats (e.g., AppImage, Snappy, Flatpak, etc.), which in his opinion are repeating some mistakes made by later Windows versions (and more specifically Windows 2000).

The new Linux container formats dump the packaging work on the developer, criticized Brown, and only seemingly solve the problems of compatibility and portability. His point: Not all distributions adhere to the Linux Standard Base (LSB) [3]; therefore, a containerized app might be missing dependencies – “might” being the operative word. At the end, he admitted that the makers of the container formats do not conceal this, and stated: “It works, but this is one big pile of work.” Work that the distributors usually do, reiterated Brown, who was reporting from their perspective. At times, it sounded a little like SUSE was worried about its business model, although Brown repeatedly assured everyone that the new formats would save the distributors some work.

Without dismissing all of Brown's criticisms, most app developers have a different assessment of the new formats, as posts by Linus Torvalds and

Dirk Hohndel suggest [4]. The two kernel developers are working on a diving application and like AppImage because it supports simple distribution of the app to all distros. This is difficult or impossible using the classical approach, because the application presupposes libraries that are not (yet) on-board with the distributions.

Flight Capable

Although progress is too slow for some, Linux is evolving far too quickly for others. Gerolf Ziegenhain is working on software for air traffic control. His company focuses on Linux, which has not always been easy, because the software needs greater continuity than Linux and its ecosystem can offer. They have thus adapted the software with all sorts of tricks of their own.

From the air traffic control perspective, the Linux community is much like the Gallic village community of which Asterix and Obelix [5] are a part. Ziegenhain explained how the company still managed to meet the numerous requirements for this kind of software. His conclusion: Building mission-critical systems with Linux is possible.

No RISC, No Fun?

Developer Arun Thomas received applause after his statement on RISC-V: People who rely on the instruction set architecture for CPUs no longer need to worry about license fees and complicated contracts, which attorneys negotiated for years.

The project community, originally developed at UC Berkeley, has “modest plans,” he joked: They just wanted it to become the “standard architecture for all devices.” Certainly, RISC-V is still well away from this target, but at least players in industry and research are supporting the RISC-V Foundation on a wide front. The organization now has more than 130 members including Google, HPE, IBM, Microsoft, Oracle, Nvidia, and Qualcomm.

Modularity is another benefit of RISC-V. The architecture can be used in a versatile way by anything from extremely low cost microcontrollers up to multicore CPU data centers, offering word widths of 32, 64, and even 128 bits.

However, ideas still fly ahead of practical usage; the only boards on sale feature microcontrollers by SiFive, but the odds on that soon changing are not bad given the broad industry support.

Free VR

Away from the mainstream keynotes, visitors experienced numerous meetings of small open source projects in the form of the OpenHMD project [6] (Figure 2), whose employees met in the flesh for the first time at FOSDEM. During a birds-of-a-feather (BoF) session, they sat with their VR devices in a single room, discussed the latest news, and had fun hacking.

The project is looking to develop free drivers for numerous head-mounted displays (HMDs) for use with Linux. As a charismatic leader, musician and Blender developer Joey Ferwerda ensured a good mood and distributed candy to the developers.

The current results of what are often time-consuming reverse engineering attempts are quite impressive: The project is now gradually starting to support various versions of Oculus Rift [developer kit 1 (DK1) and DK2, consumer version 1 (CV1)], the HTC Vive, Sony's PlayStation VR, and some exotic input devices and head-mounted displays. Intel may contribute its own solution to HMD, a successor model of the Alloy project. Blender will be offering official OpenHMD support in the spring.

Even if it will probably take a while to achieve full hardware support, the project can look forward to an interesting year, and as long as the developers are having fun, schedules are irrelevant, anyway. ■■■

INFO

- [1] FOSDEM: <http://fosdem.org>
- [2] Curl: <https://curl.haxx.se>
- [3] Linux Standard Base: https://en.wikipedia.org/wiki/Linux_Standard_Base
- [4] Praise for AppImage from Linus Torvalds: <https://plus.google.com/+LinusTorvalds/posts/WyrATKUnmrS>
- [5] Asterix: <https://en.wikipedia.org/wiki/Asterix>
- [6] OpenHMD: <http://openhmd.net>



Figure 2: Some visitors spent the whole of FOSDEM in a single dev room. Projects like OpenHMD used the event for hacking sessions.



The GPL and the birth of a revolution

Copyleft

The GNU General Public License was born of the simple idea that freedom matters. Yet this simple tool for protecting freedom has another important feature that makes it even more powerful, and that is the ability to build communities. *By Joe Casad*

The scene is 1983, and the high tech world is changing. Computers have been around in some form for a generation, but they are rapidly evolving from the room-sized behemoths of the past into smaller, more versatile systems. Competition is increasing and prices are dropping. And as prices drop, hardware vendors look for other sources of revenue. Corporations start to think of software as a product – independent of the hardware platform. Whole companies, like Microsoft, rise to prominence selling software alone and don't even bother with hardware.

Through this era, the companies that sold software started to become very particular about the rules for using that software, and source code started to become something like a trade secret. When you bought software, you didn't really own anything. You bought a license to use an executable binary, and you didn't even have a good way of knowing what was on that binary.

This new reality was not appealing to the community of programmers who were accustomed to tinkering with computers – in fact, the ability to shape, direct, and modify programs was the main thing they *liked* about computers. Many were paid handsomely for becoming part of the corporate computer industry, but a few held fast.

Richard Stallman didn't like the way a whole industry had developed around the strategy of withholding user rights. Another activist might have simply screamed this critique to a government committee or pinned it to the door of a cathedral, but Richard Stallman wasn't just any activist – he was also an accomplished programmer who worked at the MIT Artificial Intelligence Lab. Stallman had already written the famous Emacs editor, and he had “worked extensively on compilers, editors, debuggers, command interpreters, the Incompatible Timesharing System, and the Lisp Machine operating system.”

For Stallman the answer was clear: He and those who wanted to join him would build a complete operating system that embodied his vision of freedom for users and programmers. People would then have a choice – they wouldn't have to sign EULAs and NDAs, or accept licenses that took away their rights, just because they didn't have any other options.

Stallman announced the founding of the GNU project through an email announcement on September 27, 1983 [1]. Of course, building a movement for free software required a definition for free software. The GNU project developed a definition [2] that centered around four essential freedoms (numbered from 0 to 3 in characteristically geeky fashion). Stallman described those freedoms as follows:



(0) The freedom to run the program as you wish, for any purpose (freedom 0).

- (1) The freedom to study how the program works and change it so it does your computing as you wish. Access to the source code is a precondition for this (freedom 1).
- (2) The freedom to redistribute copies, so you can help your neighbor (freedom 2).
- (3) The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

Because software permissions are conveyed through a license, Stallman developed a very special software license to go with the GNU project: the GNU General Public License (GPL).

By now almost everyone in the FOSS space is aware of the GPL and its important central tenet: If you distribute a modified version of this software in binary form, you have to send the source code along with it, so others can also make improvements, and so everyone (at least theoretically) can see exactly what the code is doing.

This simple and elegant innovation was playfully christened the *copyleft* protection, because it was supposedly a left-wing version of the *copyright*. Volunteer programmers were reluctant to donate their time for writing code that would one day be the “property” of someone else, but the GPL gave them assurance that their work would always be available to other programmers – not just in the next release, but always, in whatever later form it would take.

Coding once again became a community activity – as it had been in the days before the proprietary license era, and the

growing power of the Internet meant these growing communities of developers could communicate from all over the world.

Using the power of the GPL, Stallman and other free software programmers began to build the system of components that would make up the GNU system. According to the GNU web-site [3], “By 1990 we had either found or written all the major components except one – the kernel. Then Linux, a Unix-like kernel, was developed by Linus Torvalds in 1991 and made free software in 1992. Combining Linux with the almost-complete GNU system resulted in a complete operating system: the GNU/Linux system.”

In his autobiography *Just for Fun* [4], Linus recalls hearing Stallman give a speech in Helsinki in 1991, remarking that “In Richard I saw, for the first time in my life, the stereotypical longhaired, bearded hacker type. We don’t have them in Finland.” Torvalds goes on to state, “I guess something from his speech must have sunk in, because a year later, Linux was licensed under the GPL.”

Linus had never wanted to make money on Linux. The first license he put on Linux stated that Linux could circulate freely but could not be sold. The license also had its own version of the copyleft, mandating that changes to the code should circulate with source code form. But Linus soon ran into trouble, “By February [1992], it was not uncommon for folks to attend Unix users meetings armed with floppies containing Linux. People started asking if they could charge, say, \$5 dollars just to cover the cost of the disk and their time. The trouble was, that was a violation of my copyright.”

The GPL offered a much more subtle solution. According to the terms of the GPL, “You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.” The nature of that fee is not explained, however, because the source code is always available, the fee tends to reflect the actual cost of providing the service (with some reasonable markup for convenience). No one can corner the market or charge a monopoly price, because if anyone tries to charge too much, someone else could just take the code and offer the same service.

Linux adopted the GPL in 1992. Linus adds that part of his reason for adopting the GPL was that he had “...stood on the shoulders of giants,” in using free tools from the GNU project while building Linux. In particular, he highlights the importance of the GCC compiler in his description of why he went with the GPL.

With a complete operating environment, a kernel, and a license that preserved innovation for the whole community, the GNU system took off. Soon startup companies, and even large corporations, were investing in it. And somewhere in the explosion, the buzz around the Linux kernel started to overwhelm the careful framework that had been put in place by the GNU project, and the whole system came to be known as Linux. To this day, the Free Software Foundation (FSF) formally objects to the name “Linux” for the whole system, preferring the name GNU/Linux to indicate the Linux kernel with GNU userland, libraries, and compiler. The popular belief is that the FSF wants to keep “GNU” in the name to receive credit



for their work, which is no doubt part of it (and a legitimate concern), but they also believe strongly that the emphasis should remain on the freedoms that the GNU project represents [5]. Linux is just software; GNU is a vision.

Versions

The first version of the GPL was released on February 25, 1989. The “General” in “General Public License” reveals the original intent of the license, which was to provide a general solution for previous copyleft licenses associated with other GNU software, such as the GNU C compiler, glibc, and the Emacs editor. GPLv2 [6] appeared a couple years later in 1991. The most important change with version 2 was Section 7, which is also sometimes called the “Liberty or Death” clause. The clause states:

“If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all...”

One of the issues with version 1 was that, in the case where the license conflicted with other licenses, patents, or contractual obligations, users would choose to simply ignore the conflicting

clause but still continue to use the software. Section 7 stated unambiguously that the user is forced to obey all parts of the license, and if other obligations conflict with the terms of the GPL, then the user should not distribute the code at all.

GPL 2 was the GNU project’s go-to license for many years, and it is still the official license for many free software projects, including the Linux kernel. But the computer industry was changing, and the FSF could see a need to change with it. In 2005, the foundation announced that work would begin on a new version of the GPL. GPLv3 appeared in 2007 after extensive committee deliberations and input from the free software community, as well as companies that used and supported free software.

The FSF describes GPLv3 as follows, “The most recent version protects users from three recent threats:

Tivoization: Some companies have created various different kinds of devices that run GPLed software, and then rigged the hardware so that they can change the software that’s running, but you cannot. If a device can run arbitrary software, it’s a general-purpose computer, and its owner should control what it does. When a device thwarts you from doing that, we call that tivoization. Laws prohibiting free software: Legislation like the Digital Millennium Copyright Act and the European Union Copyright Directive make it a crime to write or share software that can break DRM (Digital Restrictions Management). These laws should not interfere with the rights the GPL grants you.

Discriminatory patent deals: Microsoft [was, at the time] telling people that they will not sue free software users for patent infringement – as long as you get the software from a vendor that’s paying Microsoft for the privilege. Ultimately, Microsoft is trying to collect royalties for the use of free software, which interferes with users’ freedom. No company should be able to do this.” [7] GPLv3 also addressed some license compatibility issues with the Apache license, the Affero license, and others.

GPLv3 embodied the FSF’s vision for the new millennium, however, some developers believed it went a step too far. The Linux kernel community, and some other development efforts, opted to remain with GPLv2, thus creating the world of two GPLs we know today. Linus Torvalds would later remark, “In some ways, Linux was the project that really made the split clear between what the FSF is pushing, which is very different from what open source and Linux has always been about, which is more of a technical superiority instead of a religious belief in freedom...So, the GPL Version 3 reflects the FSF’s goals and the GPL Version 2 pretty closely matches what I think a license should do and so right now; Version 2 is where the kernel is.” [8]

TABLE 1: Top 6 Open Source Licenses per Black Duck

License	Share	Copyleft/Permissive?
MIT License	32.2%	permissive
GPLv2	17.75%	copyleft
Apache 2.0	14.26%	permissive
GPLv3	7.47	copyleft
BSD 2.0 (3-clause, new or revised)	5.5	permissive
ISC	5.5	permissive

TABLE 2: GNU/Linux Distros Endorsed by GNU

Name	Description	URL
Blag	Fedora-based free distro	http://www.blagblagblag.org/
Dragora	Independent GNU/Linux distribution based on concepts of simplicity	http://dragora.org/repo.fsl/doc/trunk/www/index.md
Dynebolic	GNU/Linux distro with special emphasis on audio and video editing	https://www.dyne.org/software/dynebolic/
gNewSense	Debian-based distro with sponsorship from the FSF	www.gnewsense.org/
GuixSD	All-free distro built around the Guix package manager	https://www.gnu.org/software/guix/
Musix	Knoppix-based system with the emphasis on audio production	https://musixdistro.wordpress.com/
Parabola	Arch-based system that prioritizes simple package and system management	https://www.parabola.nu/
Trisquel	Ubuntu-based distro oriented toward “small enterprises, domestic users, and educational centers”	https://trisquel.info/
Ututo XS	Gentoo-based distro that was the first fully free GNU/Linux recognized by the GNU project	https://trisquel.info/



Today, both GPLv2 and GPLv3 are supported and defended by the FSF, and they both have the same secret ingredient that keeps the code free: the copyleft.

The GPL also supports other licenses that form parts of the complete the GPL ecosystem, including:

- Lesser GPL (LGPL): A license intended for use with software libraries, the LGPL still offers a strong copyleft for modification and distribution, but it is more relaxed than the standard GPL for allowing linking with software with a wider range of license types.
- GNU Documentation License: A license intended for documentation projects, to ensure that the license for the instructions is never more restrictive than the license for the software. Richard Stallman's biography *Free as in Freedom* is covered by the GNU Documentation license. See the excerpt on page 20. The license is reprinted on page 25.
- GNU Affero Public License: A GPL-like license that adds an additional term requiring that users who interact with a

program over a network have access to the source code.

The Affero license was developed because the modern concept of a web service does not depend upon the “distribution” of the program, which is the trigger for requiring the availability of source code in the original GPL.

See the extensive information and commentary at the GNU website [9] for more on licenses supported and endorsed by the FSF.

Other Free Licenses

The GPL was hugely important in the evolution of Free Software Licenses, but it isn't the only approach. Although the FSF has long championed the copyleft protection, which ensures that future versions of the software will be distributed under the same free license, the copyleft isn't mentioned in the FSF's four essential freedoms (described previously in this article). It is possible to create a license that embodies all the four essential freedoms without placing restrictions on how the code will

PERMISSIVE LICENSES: A Closer Look

If GPL is so powerful, why don't all free software projects use it? It turns out that many FOSS developers, many of whom would claim as much commitment to the principles of free software as their GPL counterparts, prefer a permissive license such as the Apache, MIT, or BSD license to the copyleft protections offered by the GPL. These permissive licenses all qualify as free software under the guidelines of the GNU project and the FSF, but they represent a wholly different philosophy.

The *BSD Advantages* page of the FreeBSD website [11] cites an Apache project document to describe the benefits of a permissive license. “This type of license is ideal for promoting the use of a reference body of code that implements a protocol for common service...many of us wanted to see HTTP survive and become a true multiparty standard, and we would not have minded in the slightest if Microsoft or Netscape chose to incorporate our HTTP engine or any other component of our code into their products, if it helped further the goal of keeping HTTP common...All this means that, strategically speaking, the project needs to maintain sufficient momentum, and that participants realize greater value by contributing their code to the project, even code that would have had value if kept proprietary.”

The GPL adds some legal complications that make it more complicated to integrate with other software. According to the FreeBSD project, “Developers tend to find the BSD license attractive as it keeps legal issues out of the way and lets them do whatever they want with the code. In contrast, those who expect others to evolve the code, or who do not expect to make a living from their work associated with the system (such as government employees), find the GPL attractive, because it forces code developed by others to be given to them and keeps their employer from retaining copyright and thus potentially ‘burying’ or orphaning the software. If you want to force your competitors to help you, the GPL is attractive.”

Through the years, code from the permissive BSD projects has made its way into many proprietary systems. Mac OS and Solaris are both originally based on BSD code. Microsoft reportedly integrated BSD's TCP/IP implementation into Windows. The copyleft viewpoint would regard these code appropriations as a loss for the community. Permissive proponents see it differently: by making it easy to adapt and integrate these components with

other systems, they are spreading the benefits of free-software-based community development to a wider audience. Apple thus became invested in Unix, and Microsoft became a proponent of standards-based TCP/IP networking, rather than having to force the world to use its outdated proprietary protocols such as NetBEUI and its in-house, reverse-engineered version of the Novell NetWare protocols.

According to the FreeBSD community, “The question why should we help our competitors or let them steal our work?” comes up often in relation to a BSD license. Under a BSD license, if one company came to dominate a product niche that others considered strategic, the other companies can, with minimal effort, form a mini-consortium aimed at reestablishing parity by contributing to a competitive BSD variant that increases market competition and fairness. This permits each company to believe that it will be able to profit from some advantage it can provide, while also contributing to economic flexibility and efficiency.”

The GPL lends itself to large projects that keep the community working together on a single code base. Permissive licenses are better suited for smaller, collaborative projects that serve as a core or incubator for a larger ecosystem that might include proprietary implementations.

The copyleft protection of the GPL allowed Linux to become bigger and more popular than any of the permissively licensed BSD variants. However, BSD, with its permissive license and easy integration, played a role in spreading the gospel of Unix and standards-based programming to build the world in which Linux could flourish.

Another factor that has entered into the conversation over the past few years is the privacy and security concerns associated with code you can't read or see. As Eben Moglen of the Software Freedom Law Center says (see the interview elsewhere in this issue), if you can't see the code, you don't really know if your phone is spying on you or not. Although the BSDs are known as extremely secure systems (and verifiably so, since they are distributed under a free software license), the fact that the code can be removed from open source distribution means you can never really be sure where the code will end up and whether proprietary derivatives will be used in a way that respects user privacy.



be used later. Non-copyleft free licenses, which the FSF (and others) call *permissive* licenses, are another important feature of the free software landscape.

Black Duck Software, which tracks open source license usage, ranks the frequency of licenses used in open source projects [10]. The top six licenses in the widely quoted Black Duck study are shown in Table 1. According to Black Duck, the information in Table 2 comes from the analysis of “over 2,000,000 open source projects from over 9,000 forges and repositories.”

Overall, the GPL-based licenses hold approximately 33% of the market, which is down from over 50% a few years ago. (It should be noted that the Black Duck figures are not universally accepted – some observers have challenged the Black Duck methodology.) As a means of studying the trend, however, it appears that the total number of GPL-licensed projects is up, but the percentage of projects opting for a GPL-style license is down over the last few years.

Many factors undoubtedly contribute to the decline in the percentage of GPL-based licenses, and any definitive diagnosis would be purely speculation, but a couple potential influences are worth mentioning. First is that, although GPL programs have been inhabiting free software repositories for years, GPL-licensed programs do not fit easily into the contemporary concept of an iTunes-style for-profit App Store. Also, the complexity of understanding the interaction of the GPL with other license types could be intimidating for many

WHAT IS FREE?

A GNU/Linux distribution might contain hundreds (or even thousands) of different programs or software components. Most distros are designed to start up easily on a wide range of hardware systems and open a wide range of files without complication. For easier “out-of-the-box” functionality, these distros might contain firmware or drivers that are not free software. Some distributions also contain non-free multimedia codecs that allow the system to play music and video encoded in proprietary formats.

Other Linux systems make an effort to keep their contents free of binary blobs and other non-free software. If you really want these non-free components, you could always install them yourself later, but by creating an environment where proprietary software is excluded by default, the users of these all-free systems live out their values, projecting a vision of a world in which proprietary licenses are unnecessary and out of place. They also provide an incentive for hardware vendors step up to the challenges of providing free drivers. If everyone used an all-free system, vendors would not be able to continue to circulate non-free drivers in binary-only form.

Still, freedom can be in the eye of the beholder. Even the quintessential all-free GNU/Linux distro Debian is not endorsed by the GNU project because it maintains a separate repository of non-free components and thus, according to the GNU project, blurs the line between what is and isn't included with the distribution.

The list of distributions fully endorsed by the GNU project for PC-grade computers [12] is actually quite small and doesn't really contain any of the popular distros currently considered “mainstream” (see Table 2).

smaller projects. (See the box entitled “Permissive Licenses: A Closer Look.”)

Another factor could be that proprietary software vendors have found ways to encourage the “free” (as in beer) circulation of their programs without fully granting the rights embodied in the FSF's four essential freedoms. (See the box entitled “What is Free?”)

Ultimately, however, the number of projects is not necessarily the best indicator of a license's importance. The list of iconic free software projects licensed under the GPL is quite impressive by any standards. In addition to the Linux kernel, glibc, and the GCC, other projects include Samba, Drupal, Gimp, Bash, Gnome, and countless other inhabitants of a standard GNU/Linux system.

Conclusion

It is clear that the Linux kernel has gone places that Linus Torvalds never expected when he started to work on it. On the other hand, the free software movement is exactly what Richard Stallman would have expected. From the beginning, his goal was to change the world, and it is easy to argue that he has succeeded in creating a new vision of freedom that stands in opposition to the corporatization of the software industry.

However, no one at the FSF believes the task is complete. Too many systems still use proprietary software, and new threats to privacy and security make it more important than ever to keep the source code for systems running on phones, computers, electronic gadgets, and voting machines free and available for study. As the FOSS community rallies to meet a new generation of challenges, they will bring along an important tool: the GNU GPL and the powerful copyleft protection. ■■■

INFO

- [1] GNU project announcement: <https://www.gnu.org/gnu/initial-announcement.html>
- [2] Free Software Definition: <https://www.gnu.org/philosophy/free-sw.html>
- [3] Overview of the GNU system: <https://www.gnu.org/gnu/gnu-history.html>
- [4] Torvalds, Linus, and David Diamond. *Just for Fun: The Story of an Accidental Revolutionary*. HarperBusiness, 2001
- [5] “What's in a Name?” by Richard Stallman: <https://www.gnu.org/gnu/why-gnu-linux.html>
- [6] GPLv2: <https://www.gnu.org/licenses/gpl-2.0.html>
- [7] A Quick Guide to GPLv3: <https://www.gnu.org/licenses/quick-guide-gplv3.html>
- [8] “Torvalds Still Keen on GPLv2”:
<http://www.internetnews.com/dev-news/article.php/3720371/Torvalds+Still+Keen+On+GPLv2.htm>
- [9] Various licenses and comments about them: <https://www.gnu.org/licenses/license-list.html#SoftwareLicenses>
- [10] Black Duck's top open source licenses: <https://www.blackducksoftware.com/top-open-source-licenses>
- [11] BSD advantages: <https://www.freebsd.org/doc/en/articles/bsd-gpl/bsd-advantages.html>
- [12] GNU project's list of free GNU/Linux distributions: <https://www.gnu.org/distros/free-distros.html>

COMPLETE YOUR LIBRARY

Order a digital archive bundle and save at least **50% off** the digisub rate!



ORDER YOURS TODAY!
shop.linuxnewmedia.com



You get an **entire year of your favorite magazines** in PDF format that you can access at any time from any device!



Free as In Freedom: Richard Stallman and the Free Software Revolution

For Want of a Printer

This is an excerpt from the book *Free as in Freedom 2.0: Richard Stallman and the Free Software Revolution* [1], a revision of *Free as in Freedom: Richard Stallman's Crusade for Free Software* [2]. Copyright © 2002, 2010 Sam Williams. Copyright © 2010 Richard M. Stallman. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included at the end of this article in the box entitled "GNU Free Documentation License."

By Sam Williams, Second edition revisions by Richard M. Stallman

I fear the Greeks. Even when they bring gifts.

– Virgil, *The Aeneid*

The new printer was jammed, again.

Richard M. Stallman, a staff software programmer at the Massachusetts Institute of Technology's Artificial Intelligence Laboratory (AI Lab), discovered the malfunction the hard way. An hour after sending off a 50-page file to the office laser printer, Stallman, 27, broke off a productive work session to retrieve his documents. Upon arrival, he found only four pages in the printer's tray. To make matters even more frustrating, the four pages belonged to another user, meaning that Stallman's print job and the unfinished portion of somebody else's print job were still trapped somewhere within the electrical plumbing of the lab's computer network.

Waiting for machines is an occupational hazard when you're a software programmer, so Stallman took his frustration with a grain of salt. Still, the difference between waiting for a machine and waiting on a machine is a sizable one. It wasn't the first time he'd been forced to stand over the printer, watching pages print out one by one. As a person who spent the bulk of his days and nights improving the efficiency of machines and the software programs that controlled them, Stallman felt a natural urge to open up the machine, look at the guts, and seek out the root of the problem.

Unfortunately, Stallman's skills as a computer programmer did not extend to the mechanical-engineering realm. As freshly printed documents poured out of the machine, Stallman had a chance to reflect on other ways to circumvent the printing jam problem.

How long ago had it been that the staff members at the AI Lab had welcomed the new printer with open arms? Stallman wondered. The machine had been a donation from the Xerox Corporation. A cutting edge prototype, it was a modified version of a fast Xerox photocopier. Only instead of making copies, it relied on software data piped in over a computer network to turn that data into professional-looking documents. Created by engineers at the world-famous Xerox Palo Alto Research Facility, it was, quite simply, an early taste of the desktop-printing revolution that would seize the rest of the computing industry by the end of the decade.

Driven by an instinctual urge to play with the best new equipment, programmers at the AI Lab promptly integrated the new machine into the lab's sophisticated computing infrastructure. The results had been immediately pleasing. Unlike the lab's old



printer, the new Xerox machine was fast. Pages came flying out at a rate of one per second, turning a 20-minute print job into a 2-minute print job. The new machine was also more precise. Circles came out looking like circles, not ovals. Straight lines came out looking like straight lines, not low-amplitude sine waves.

It was, for all intents and purposes, a gift too good to refuse.

Once the machine was in use, its flaws began to surface. Chief among the drawbacks was the machine's susceptibility to paper jams. Engineering-minded programmers quickly understood the reason behind the flaw. As a photocopier, the machine generally required the direct oversight of a human operator. Figuring that these human operators would always be on hand to fix a paper jam, if it occurred, Xerox engineers had devoted their time and energies to eliminating other pesky problems. In engineering terms, user diligence was built into the system.

In modifying the machine for printer use, Xerox engineers had changed the user-machine relationship in a subtle but profound way. Instead of making the machine subservient to an individual human operator, they made it subservient to an entire networked population of human operators. Instead of standing directly over the machine, a human user on one end of the network sent his print command through an extended bucket brigade of machines, expecting the desired content to arrive at the targeted destination and in proper form. It wasn't until he finally went to check up on the final output that he realized how little of it had really been printed.

Stallman was hardly the only AI Lab denizen to notice the problem, but he also thought of a remedy. Years before, for the lab's previous printer, Stallman had solved a similar problem by modifying the software program that regulated the printer, on a small PDP-11 machine, as well as the Incompatible Time-sharing System that ran on the main PDP-10 computer. Stallman couldn't eliminate paper jams, but he could insert software code that made the PDP-11 check the printer periodically, and report jams back to the PDP-10. Stallman also inserted code on the PDP-10 to notify every user with a waiting print job that the printer was jammed. The notice was simple, something along the lines of "The printer is jammed, please fix it," and because it went out to the people with the most pressing need to fix the problem, chances were that one of them would fix it forthwith.

As fixes go, Stallman's was oblique but elegant. It didn't fix the mechanical side of the problem, but it did the next best

thing by closing the information loop between user and machine. Thanks to a few additional lines of software code, AI Lab employees could eliminate the 10 or 15 minutes wasted each week in running back and forth to check on the printer. In programming terms, Stallman's fix took advantage of the amplified intelligence of the overall network.

If you got that message, you couldn't assume somebody else would fix it," says Stallman, recalling the logic. "You had to go to the printer. A minute or two after the printer got in trouble, the two or three people who got messages arrive to fix the machine. Of those two or three people, one of them, at least, would usually know how to fix the problem."

Such clever fixes were a trademark of the AI Lab and its indigenous population of programmers. Indeed, the best programmers at the AI Lab disdained the term programmer, preferring the more slangy occupational title of hacker instead. The job title covered a host of activities – everything from creative mirth making to the improvement of existing software and computer systems. Implicit within the title, however, was the old-fashioned notion of Yankee ingenuity. For a hacker, writing a software program that worked was only the beginning. A hacker would try to display his cleverness (and impress other hackers) by tackling an additional challenge: to make the program particularly fast, small, powerful, elegant, or somehow impressive in a clever way.

Companies like Xerox made it a policy to donate their products (and software) to places where hackers typically congregated. If hackers used these products, they might go to work for the company later on. In the 60s and early 70s, they also sometimes developed programs that were useful for the manufacturer to distribute to other customers.

When Stallman noticed the jamming tendency in the Xerox laser printer, he thought of applying the old fix or "hack" to this printer. In the course of looking up the Xerox laser-printer software, however, Stallman made a troubling discovery. The printer didn't have any software, at least nothing Stallman or a fellow programmer could read. Until then, most companies had made it a form of courtesy to publish source-code files – readable text files that documented the individual software commands that told a machine what to do. Xerox, in this instance, had provided software files only in compiled, or binary, form. If programmers looked at the files, all they would see was an endless stream of ones and zeroes – gibberish.

There are programs, called "disassemblers," to convert the ones and zeroes into low-level machine instructions, but figur-





ing out what those instructions actually “do” is a long and hard task, known as “reverse engineering.” To reverse engineer this program could have taken more time than five years’ worth of jammed printouts. Stallman wasn’t desperate enough for that, so he put the problem aside.

Xerox’s unfriendly policy contrasted blatantly with the usual practices of the hacker community. For instance, to develop the program for the PDP-11 that ran the old printer, and the program for another PDP-11 that handled display terminals, the AI Lab needed a cross-assembler program to build PDP-11 programs on the PDP-10 main computer. The lab’s hackers could have written one, but Stallman, a Harvard student, found such a program at Harvard’s computer lab. That program was written to run on the same kind of computer, the PDP-10, albeit with a different operating system. Stallman never knew who had written the program, since the source code did not say. But he brought a copy back to the AI Lab. He then altered the source code to make it run on the AI Lab’s Incompatible Timesharing System (ITS). With no muss and little fuss, the AI Lab got the program it needed for its software infrastructure. Stallman even added a few features not found in the original version, making the program more powerful. “We wound up using it for several years,” Stallman says.

From the perspective of a 1970s-era programmer, the transaction was the software equivalent of a neighbor stopping by to borrow a power tool or a cup of sugar from a neighbor. The only difference was that in borrowing a copy of the software for the AI Lab, Stallman had done nothing to deprive anyone else of the use of the program. If anything, other hackers gained in the process, because Stallman had introduced additional features that other hackers were welcome to borrow back. For instance, Stallman recalls a programmer at the private engineering firm, Bolt, Beranek & Newman, borrowing the program. He made it run on Twenex and added a few additional features, which Stallman eventually reintegrated into the AI Lab’s own source-code archive. The two programmers decided to maintain a common version together, which had the code to run either on ITS or on Twenex at the user’s choice.

“A program would develop the way a city develops,” says Stallman, recalling the software infrastructure of the AI Lab. “Parts would get replaced and rebuilt. New things would get added on. But you could always look at a certain part and say, ‘Hmm, by the style, I see this part was written back in the early 60s and this part was written in the mid-1970s.’”

Through this simple system of intellectual accretion, hackers at the AI Lab and other places built up robust creations. Not

every programmer participating in this culture described himself as a hacker, but most shared the sentiments of Richard M. Stallman. If a program or software fix was good enough to solve your problems, it was good enough to solve somebody else’s problems. Why not share it out of a simple desire for good karma?

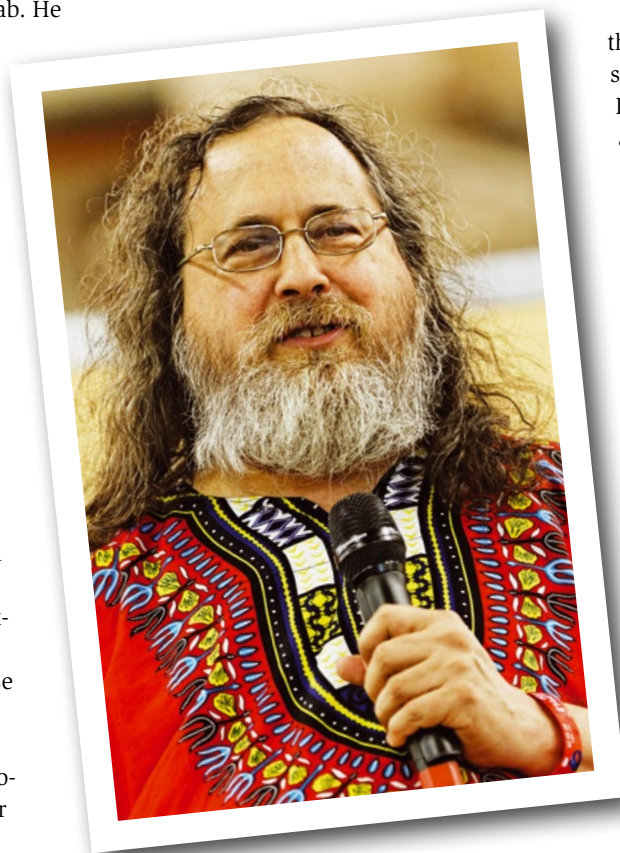
This system of cooperation was being undermined by commercial secrecy and greed, leading to peculiar combinations of secrecy and cooperation. For instance, computer scientists at UC Berkeley had built up a powerful operating system called BSD, based on the Unix system they had obtained from AT&T. Berkeley made BSD available for the cost of copying a tape, but would only give these tapes to schools that could present a \$50,000 source license obtained from AT&T. The Berkeley hackers continued to share as much as AT&T let them, but they had not perceived a conflict between the two practices.

Likewise, Stallman was annoyed that Xerox had not provided the source-code files, but not yet angry. He never thought of asking Xerox for a copy. “They had already given us the laser printer,” Stallman says. “I could not say they owed us something more. Besides, I took for granted that the absence of source code reflected an intentional decision, and that asking them to change it would be futile.”

Good news eventually arrived: word had it that a scientist at the computer-science department at Carnegie Mellon University had a copy of the laser printer source code.

The association with Carnegie Mellon did not augur well. In 1979, Brian Reid, a doctoral student there, had shocked the community by refusing to share his text-formatting program, dubbed Scribe. This text formatter was the first to have mark-

up commands oriented towards the desired semantics (such as “emphasize this word” or “this paragraph is a quotation”) rather than low-level formatting details (“put this word in italics” or “narrow the margins for this paragraph”). Instead Reid sold Scribe to a Pittsburgh-area software company called Unilogic. His graduate-student career ending, Reid says he simply was looking for a way to unload the program on a set of developers that would take pains to keep it from slipping into the public domain. (Why one would consider such an outcome particularly undesirable is not clear.) To sweeten the deal, Reid also agreed to insert a set of time-dependent functions – “time bombs” in software-programmer parlance – that deactivated freely copied versions of the program after a 90-day expiration date. To avoid deactivation, users paid the software company, which then issued a code that defused the internal time-bomb anti-feature.



CC BY-SA 3.0



For Stallman, this was a betrayal of the programmer ethos, pure and simple. Instead of honoring the notion of share-and-share alike, Reid had inserted a way for companies to compel programmers to pay for information access. But he didn't think deeply about the question, since he didn't use Scribe much.

Unilogic gave the AI Lab a gratis copy to use, but did not remove or mention the time bomb. It worked, for a while; then one day a user reported that Scribe had stopped working. System hacker Howard Cannon spent hours debugging the binary until he found the time-bomb and patched it out. Cannon was incensed, and wasn't shy about telling the other hackers how mad he was that Unilogic had wasted his time with an intentional bug.

Stallman had a Lab-related reason, a few months later, to visit the Carnegie Mellon campus. During that visit, he made a point of looking for the person reported to have the printer software source code. By good fortune, the man was in his office.

In true engineer-to-engineer fashion, the conversation was cordial but blunt. After briefly introducing himself as a visitor from MIT, Stallman requested a copy of the laser-printer source code that he wanted to modify. To his chagrin, the researcher refused.

"He told me that he had promised not to give me a copy," Stallman says.

Memory is a funny thing. Twenty years after the fact, Stallman's mental history tape is blank in places. Not only does he not remember the motivating reason for the trip or even the time of year during which he took it, he also has no recollection of who was on the other end of the conversation. According to Reid, the person most likely to have fielded Stallman's request is Robert Sproull, a former Xerox PARC researcher and current director of Sun Laboratories, a research division of the computer-technology conglomerate Sun Microsystems. During the 1970s, Sproull had been the primary developer of the laser-printer software in question while at Xerox PARC. Around 1980, Sproull took a faculty research position at Carnegie Mellon where he continued his laser-printer work amid other projects.

When asked directly about the request, however, Sproull draws a blank. "I can't make a factual comment," writes Sproull via email. "I have absolutely no recollection of the incident."

"The code that Stallman was asking for was leading-edge, state-of-the-art code that Sproull had written in the year or so before going to Carnegie Mellon," recalls Reid. If so, that might indicate a misunderstanding that occurred, since Stallman wanted the source for the program that MIT had used for quite some time, not some newer version. But the question of which version never arose in the brief conversation.

In talking to audiences, Stallman has made repeated reference to the incident, noting that the man's unwillingness to hand over the source code stemmed from a nondisclosure agreement, a contractual agreement between him and the Xerox Corporation giving the signatory access to the software source code in exchange for a promise of secrecy. Now a standard item of business in the software industry, the nondisclosure agreement, or NDA, was a novel development at the time, a reflection of both the commercial value of the laser printer to Xerox and the information needed to run it. "Xerox was at the time trying to make a commercial product out of the laser printer," recalls Reid. "They would have been insane to give away the source code."

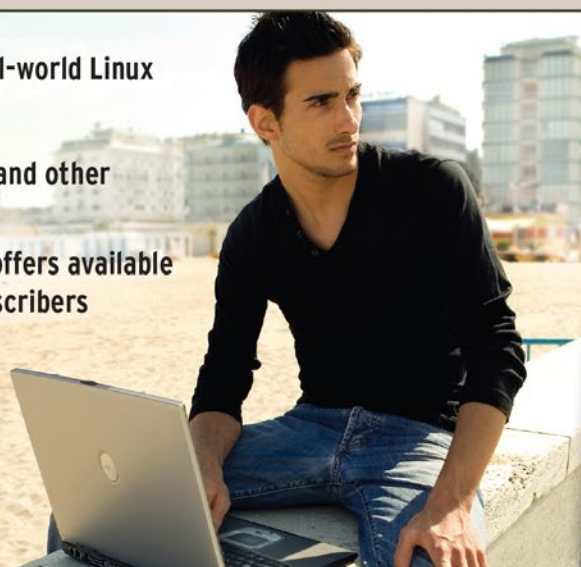
For Stallman, however, the NDA was something else entirely. It was a refusal on the part of some CMU researcher to participate in a society that, until then, had encouraged software programmers to regard programs as communal resources. Like a peasant whose centuries-old irrigation ditch had grown suddenly dry, Stallman had followed the ditch to its source only to find a brand-spanking-new hydroelectric dam bearing the Xerox logo.

For Stallman, the realization that Xerox had compelled a fellow programmer to participate in this newfangled system of compelled secrecy took a while to sink in. In the first moment, he could only see the refusal in a personal context. "I was so angry I couldn't think of a way to express it. So I just turned away and walked out without another word," Stallman recalls. "I might have slammed

LINUX UPDATE

Need more Linux? Our free Linux Update newsletter delivers insightful articles and tech tips to your mailbox twice a month.

- Original articles on real-world Linux
- Linux news
- Tips on Bash scripting and other advanced techniques
- Discounts and special offers available only to newsletter subscribers



Ft Photography, Fotolia

www.linuxpromagazine.com/mc/subscribe



the door. Who knows? All I remember is wanting to get out of there. I went to his office expecting him to cooperate, so I had not thought about how I would respond if he refused. When he did, I was stunned speechless as well as disappointed and angry.”

Twenty years after the fact, the anger still lingers, and Stallman presents the event as one that made him confront an ethical issue, though not the only such event on his path. Within the next few months, a series of events would befall both Stallman and the AI Lab hacker community that would make 30 seconds worth of tension in a remote Carnegie Mellon office seem trivial by comparison. Nevertheless, when it comes time to sort out the events that would transform Stallman from a lone hacker, instinctively suspicious of centralized authority, to a crusading activist applying traditional notions of liberty, equality, and fraternity to the world of software development, Stallman singles out the Carnegie Mellon encounter for special attention.

“It was my first encounter with a nondisclosure agreement, and it immediately taught me that nondisclosure agreements have victims,” says Stallman, firmly. “In this case I was the victim. [My lab and I] were victims.”

Stallman later explained, “If he had refused me his cooperation for personal reasons, it would not have raised any larger issue. I might have considered him a jerk, but no more. The fact that his refusal was impersonal, that he had promised in advance to be uncooperative, not just to me but to anyone whatsoever, made this a larger issue.”

Although previous events had raised Stallman’s ire, he says it wasn’t until his Carnegie Mellon encounter that he realized the events were beginning to intrude on a culture he had long considered sacrosanct. He said, “I already had an idea that software should be shared, but I wasn’t sure how to think about that. My thoughts weren’t clear and organized to the point where I could express them in a concise fashion to the rest of the world. After this experience, I started to recognize what the issue was, and how big it was.”

As an elite programmer at one of the world’s elite institutions, Stallman had been perfectly willing to ignore the compromises and bargains of his fellow programmers just so long as they didn’t interfere with his own work. Until the arrival of the Xerox laser printer, Stallman had been content to look down on the machines and programs other computer users grimly tolerated.

Now that the laser printer had insinuated itself within the AI Lab’s network, however, something had changed. The machine worked fine, barring the paper jams, but the ability to modify software according to personal taste or community need had been taken away. From the viewpoint of the software industry, the printer software represented a change in business tactics. Software had become such a valuable asset that companies no longer accepted the need to publicize source code, especially when publication meant giving potential competitors a chance to duplicate something cheaply. From Stallman’s viewpoint, the printer was a Trojan Horse. After a decade of failure, software that users could not change and redistribute – future hackers would use the term “proprietary” software – had gained a foothold inside the AI Lab through the sneakiest of methods. It had come disguised as a gift.

That Xerox had offered some programmers access to additional gifts in exchange for secrecy was also galling, but Stallman takes pains to note that, if presented with such a quid pro quo bargain

at a younger age, he just might have taken the Xerox Corporation up on its offer. The anger of the Carnegie Mellon encounter, however, had a firming effect on Stallman’s own moral lassitude. Not only did it give him the necessary anger to view such future offers with suspicion, it also forced him to turn the situation around: what if a fellow hacker dropped into Stallman’s office someday and it suddenly became Stallman’s job to refuse the hacker’s request for source code?

“When somebody invited me to betray all my colleagues in that way, I remembered how angry I was when somebody else had done that to me and my whole lab,” Stallman says. “So I said, ‘Thank you very much for offering me this nice software package, but I can’t accept it on the conditions that you’re asking for, so I’m going to do without it.’”

It was a lesson Stallman would carry with him through the tumultuous years of the 1980s, a decade during which many of his MIT colleagues would depart the AI Lab and sign nondisclosure agreements of their own. They may have told themselves that this was a necessary evil so they could work on the best projects. For Stallman, however, the NDA called the moral legitimacy of the project into question. What good is a technically exciting project if it is meant to be withheld from the community?

As Stallman would quickly learn, refusing such offers involved more than personal sacrifice. It involved segregating himself from fellow hackers who, though sharing a similar distaste for secrecy, tended to express that distaste in a more morally flexible fashion. Refusing another’s request for source code, Stallman decided, was not only a betrayal of the scientific mission that had nurtured software development since the end of World War II, it was a violation of the Golden Rule, the baseline moral dictate to do unto others as you would have them do unto you.

Hence the importance of the laser printer and the encounter that resulted from it. Without it, Stallman says, his life might have followed a more ordinary path, one balancing the material comforts of a commercial programmer with the ultimate frustration of a life spent writing invisible software code. There would have been no sense of clarity, no urgency to address a problem others weren’t addressing. Most importantly, there would have been no righteous anger, an emotion that, as we soon shall see, has propelled Stallman’s career as surely as any political ideology or ethical belief.

“From that day forward, I decided this was something I could never participate in,” says Stallman, alluding to the practice of trading personal liberty for the sake of convenience – Stallman’s description of the NDA bargain – as well as the overall culture that encouraged such ethically suspect deal-making in the first place. “I decided never to make other people victims as I had been a victim.” ■■■

INFO

- [1] Full text of the revised version *Free as in Freedom 2.0: Richard Stallman and the Free Software Revolution*, by Sam Williams with 2nd edition revisions by Richard Stallman, published by the Free Software Foundation: https://en.wikisource.org/wiki/Free_as_in_Freedom_2.0
- [2] Full text of the original book *Free as in Freedom: Richard Stallman’s Crusade for Free Software*, by Sam Williams, published by O’Reilly Media: <http://www.oreilly.com/openbook/freedom/>

**GNU Free Documentation License
Version 1.3, 3 November 2008**

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.
< <http://fsf.org/> >

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you mod-

ify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work



that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaim-

ers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (C) YEAR YOUR NAME.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

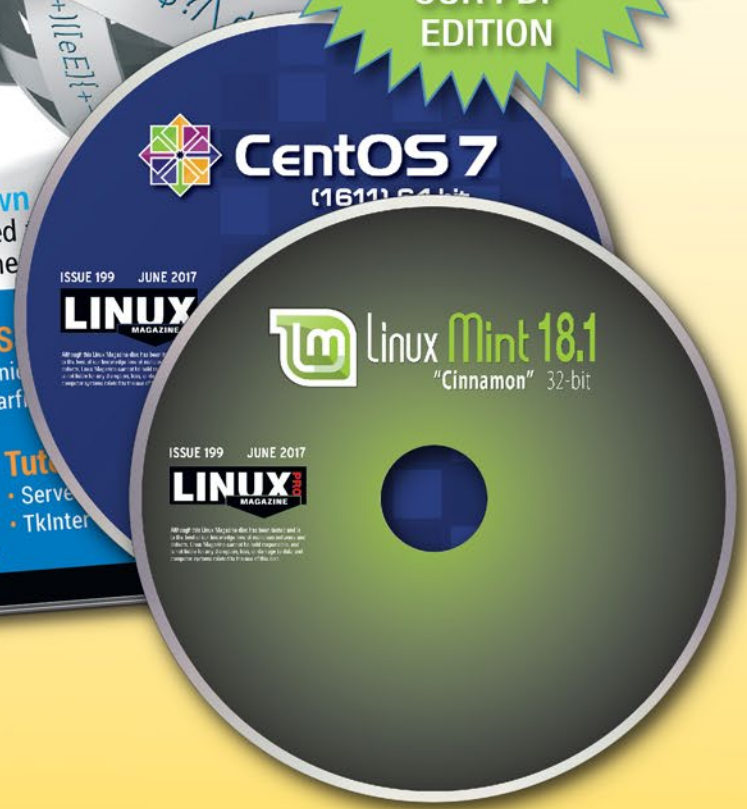
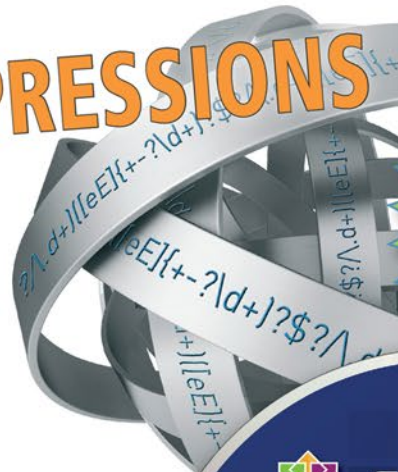
If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Subscribe now!



Don't miss a single issue of the magazine that delivers the in-depth technical solutions you'll use everyday!

GET IT NOW!
SAVE TIME ON DELIVERY WITH OUR PDF EDITION



shop.linuxnewmedia.com/subs



An interview with Eben Moglen

Ice in the Wintertime

Few have had a closer view of the Free Software revolution than Eben Moglen, former lead counsel for the Free Software Foundation and founder of the Software Freedom Law Center. We asked Moglen about the legal basis for the GPL's famous copyleft protection and the long, steady effort to tell the world about the benefits of free software. *By Joe Casad*

Linux Magazine: You started out with the Free Software Foundation (FSF) a little after the first GPL came out, right? So you weren't around at the founding.

Eben Moglen: GPLv2 was released in July of 1991. Around that time, I began working with a fellow called Philip Zimmermann, who had created a program called Pretty Good Privacy (PGP), which the United States government considered to be mutinous because it was crypto software. And there was a criminal investigation going on, claiming Mr. Zimmermann had violated the Arms Export Control Act by making PGP.

I and some other lawyers were working for Mr. Zimmermann. I had a conversation with John Markoff of the *New York Times* about the situation, and I said that the right to speak PGP is like the right to speak Navajo, which is a Native American language the United States government used as a form of radio encryption during World War II. So, John published that statement in an article in the *Times*, and when Richard Stallman read the article and saw my quote, he thought I might be able to help him with a personal legal problem he had at that time, so he got in touch.

I told him I used Emacs everyday, so it would be a long time before he exhausted his entitlement for free legal help from me, so I helped him. By this time, I was a law professor, so I had stopped making software for a living, which I did from the time I was 14 until I was getting out of law school. I had worked for IBM for a good long while.

When I started clerking in federal courts after law school, I had to drop my employment relations with IBM. I clerked for Edward Weinfeld in the southern district of New York and then for Thurgood Marshall at the U.S. Supreme Court. When I took a teaching job at Columbia, one of the things I wanted to do with my time was to work on problems of technology and freedom, so I started looking around for work to do, and that is when I found PGP and Philip Zimmerman, and I thought, if I want to understand the agenda for technology and freedom in the 21st century (this was 1993), if there is one email address known to everyone on Earth who has a problem of computers and freedom, it's `rms@gnu.org`," and I had a feeling Richard would be able to help me figure out the big picture. So, I started to help Richard, and after a couple of years, I did conclude that, indeed, I saw what the big picture was, and the big picture was right under my nose: It was free software.





By that time, I was teaching at Harvard as a visitor in '94 and '95, and Richard and I could actually spend some time together. I said, look, you need a General Counsel. And so I went to work on all the various legal business of the Free Software Foundation. Richard told me that the most important task was bringing industry behind the Free Software idea, showing that copyleft was not only beneficial to users whose rights were secured and to programmers interested in "Freedom" like Richard and me, but it would be good for business also. So, I went to work trying to convince businesses of the value of Free Software. By 1999, IBM had come to see the value, and then Hewlett-Packard, and then we could really begin to get something done to change the way the world worked.

LM: Convincing businesses of the value of Free Software seems like less of a legal counsel thing and more of a spokesman or evangelist role.

EM: Of course, part of the question is, how do these arrangements really work and why should we trust them? The real task is to show people why the interest that they have and the interests that your people have coincide. Why it is that working together will work? The real task is in creating trust.

We have a trust economy that lies underneath and alongside what economists call the "real economy." The trust economy is actually primary, as everybody knows who talks about the effects of corruption and other forms of untrustworthy conduct in affecting economic and social globe. In the world we live in, although we believe very strongly in the rule of law, we don't sue about everything, and we don't have to. We don't actually have to sue people to get them to clean up their dog's mess on the streets of New York City. When I was a child, people let their dogs go wherever they wanted to, and we had to walk around it. After a generation of training people to pick up after their dogs, they do! Sure, there are some tickets out there given to people for not cleaning up after their dogs, but the number is infinitesimal.

Similarly with respect to Free Software; sure, underneath there's copyright, and that creates potential actions for copyright infringement for people who don't play by the rules of copyleft. But, fundamentally, what we're trying to do is create trust and reliability.

The work was demonstrating to people – not merely describing but *demonstrating* – how it was that a non-profit structure intended to create social benefit could turn out to be a totally reliable partner over which to work on matters that ultimately cost these businesses billions of dollars of investments and represents fundamental parts of their business strategy.

LM: Along with this work of reaching out to

companies, were you also involved with legal actions? There were people back then who didn't think GPL was enforceable and just tried to ignore it, right? To what extent did you have to depend on litigation?

EM: Well, litigation was the last stop, and it took an awfully long time to get there. I started working for Stallman in 1993, and we didn't have to bring a lawsuit on behalf of anybody until 2006. FSF didn't have to sue anybody until around 2009. I never thought that litigation was the way that trust in the license was created.

When I went to work for Stallman in '93, Richard said "Never allow a settlement for compliance to be held up by a demand for damages," and I took that instruction to mean that, at the beginning of every call asking people to help us by complying with the rules, I could say the magic words "we don't want money." In fact, what I said to people for years and years was first, we don't want money; second, we don't want publicity; and third, we *do* want compliance. We don't need more, and we won't settle for less.

In other words, the question became, what is it going to cost you to comply? And we always tried to make clear to people that what it would cost them was basically ice in the winter-time. They were receiving valuable material, out of which they were making products or providing software to people, and we wanted them to observe the requirements associated with the materials. In situations in which we thought that there was a possibility of repetition of whatever the problem was we were working on, we asked the organization to appoint a compliance officer and to make it possible for us to communicate directly with that compliance officer so that we could talk to people directly if any recurring trouble came up.

So, basically, we settled everything on the basis that people would play right and distribute their bits in an appropriate way. The truth is it was a persuasion process. Almost never were we dealing with people who were really determined in bad faith to misuse our rights.

Sometimes in the early days we ran into problems. I worked on a problem for people who were just becoming my clients at the Samba project in 1995, in which we did have a bad faith infringer – somebody who was significantly engaged in making deliberate misappropriation of Samba – and we got those people to stop. We did that without going to court and without other kinds of expensive legal maneuvering, but we did what we needed to do, and sometimes I found myself in





a situation where I needed to apply some pressure. But anyone will tell you that litigation is wildly wasteful and enormously cumbersome, and nobody wants to use it if they don't have to.

LM: So, if somebody misused the GPL and there was a lawsuit over it, would they be able to point to a legal precedent that affirms the validity of the GPL and the copyleft protection?

EM: See, it never really worked that way. And the reason it never really worked that way is that it's not the GPL doing the job; it's copyright law. The reason it is possible to have a very strong principle of copyleft without many resources behind it is that the copyleft itself is a permission. The GPL points out that it's not a license you have to accept unless you want to avail yourself of the freedoms that copyright law otherwise accords the author.

So, here's the situation: I write a program; copyright law says I have exclusive power to copy, modify, and redistribute this work. I say to other people, I will give you permission to do all the things that I can do and only I can do, but you have to agree to exercise that permission in the following way. I don't need to go to court and test whether or not I'm allowed to give permission. I don't need to go to court to test whether or not the permission I have just given is a permission that I have a right to give with the conditions on it. Because the copyright law gives me all the power; if I want to give less than all of it away, nobody is going to doubt that I have a right to do that. And when you break the rules, you're not just breaking the rules of the license; you are infringing copyright.

So, what you have on the other side is that copyright law has rules of injunctive relief and damages, as well as extra damages for intentional infringement. What really happens is you go to people and say, look, you know it's my client's program, nobody disputes that my client's program has a copyright on it, and if you want to do whatever you are doing you need permission. The only permission you have is GPL, so you better be within its terms. If you're not within its terms, you don't have any permission at all. That's why parties didn't want to go and litigate that; what are they going to say, "Judge I'm going to use this copyright program; he gave me permission. I'm not following the permission, but that's OK; I can do it anyway"?

The real point is the copyright law stands behind with a stick, and GPL comes forward with the carrot: you can do this, as long as dot dot dot.

That's why it was comparatively easy to get people to trust us, but there's one more piece of this. Under US copyright law, there is no copyright in an infringing work. So, suppose somebody takes GPL code and puts it in a wonderful proprietary program and goes out to sell it. I go to him and I say, "You know you're violating my client's rights, because you've got our copyright work in there and it's under GPL, and you're not following the rules." Now what I'm really telling him is, "You have no copyright on your product." So, if he goes to war with me and loses, then there is capital punishment for his copyright, because there is no copyright for infringing works, and if I prove it's an infringing work, then he has no copyright.

LM: He loses his own copyright?

EM: Yes! That's the important point. So, you see that what really is happening here is that the phenomena of copyright law are being used to allow us to give a permission to people, and they can't screw around with it too much, because it comes at the ultimate expense of the very thing they are trying

to maintain, which is the copyright of their own work. This is why you get very good cooperation when you show up.

Now, most of the time, what people were distributing was software. The world of embedded products is a lot more complicated for a manufacturer and a lot more difficult, because you've got units in the field, and you've got big investments in those units, and getting to them and knowing where they are, and dealing with them may be hard. It was a lot easier to begin in a world of almost entirely pure software distribution.

LM: Internationally, the legal system could be very different from what we see in the United States?

EM: Well, the most the important part here is that you want to make use of a set of legal principles that is as close to general as you can get. And, as you say, because of the international agreement that is now more than 120 years old called the Berne Convention, we do have a lot of copyright law around the world, which is more or less harmonized. Still, GPLv2 was a pretty Americanized license; it depended upon some distinctly American features of the copyright laws. And, what we did in GPLv3 was to try and make a license that would be more dependent only on generic, internationally available principles of copyright law. Or, it took whatever it was that the local law provided and tried to deal with it exactly the way the local law did.

LM: So, GPLv3 made the license a little more international?

EM: What GPLv3 did actually was to say, in order to figure out whether something invokes a copyright of the license, there are only two questions you need to be able to answer: First, under the local copyright law for where you are, does the thing you are doing require permission? So, you don't need an expert on GPL and you don't need an expert on software law; you can go to any lawyer who knows the local copyright and say, hey, I'm doing this, do I need a license? If the answer is yes, then that is called propagation under GPLv3: I am doing something for which a license is required.

The second question is a purely factual question that an engineer could answer. Is someone other than me receiving a copy as a result of whatever I am doing? If what I am doing a) requires a license under local copyright law, which your local lawyer should be able to tell you, and b) someone else is receiving a copy, which is an engineering fact you can establish for yourself, then you have obligations under the license. So, what we tried to do was rely upon one piece of local copyright law (do I need a license for what I'm doing?) and one fact (is someone else receiving a copy?). If so, then you've got to follow the rules. That's how we did it in GPLv3.

LM: That brings up the question of web services, which has been another active topic in recent years. Do I understand correctly you've also put in some work on the Affero license?

EM: Yes, the Affero GPL is another example of a license (actually, there are two different versions that work two different ways). The Affero GPL is designed to be like the GPL copyleft except there is a separate trigger. Under the GPL family of licenses, the trigger of copyleft is that someone else is getting a copy – a binary, if language is distinguished between source code and object code. The Affero GPL adds another trigger; it says, if you are delivering services over a network to someone other than you, you have to respect those users' rights in a similar (though not exactly the same) way that you respect people's rights when you deliver them a copy. The first version of the Affero GPL was a

IT Highlights at a Glance



GPL-incompatible modification; and then in GPLv3, we took the LGPL and the AGPL, and we harmonized them with the new license in particular ways. The result was a GPLv3 that is a version of the GPL that also triggers the copyleft when services are provided using the code to other parties over the network. In that case, what the license says is you must have a facility in the program that will allow users over the network to request the source code.

LM: I remember that era when the GPLv3 was in development as being sort of contentious in a lot of ways, and you were in the middle of it. What are your memories of that era? What were the forces at play that you were trying to mediate?

EM: Well, look, you know, Richard and I started working on GPLv3 pretty much when we met. So, what did we spend, 13 years thinking about how to do this job? We worked at it very carefully. We began to talk in public about how we were going to do it a year before we started to do it. And, we explained that we were going to release a discussion draft, and then we were going to have a discussion period, and then we were going to release another discussion draft and have another discussion period, and then we were going to release a candidate license, and then we were going to promulgate the license. And, we were going to have a whole series of public committees, which were going to help us to give us those drafts. So, we spent the late 2004 and beginning 2005 getting all that together.

In the fall of 2005, Richard Stallman moved to New York City, and he and I spent months going through every single word of the first discussion draft and writing a great big long justification of everything, explaining every change from GPLv2 and, in many cases, why we were or weren't changing. We had a conference at MIT in February of 2006, and we announced the one year process of committee work that considered this license. We built a new web application called STET, which allowed us to regulate a web-based comment system for the draft, so that everybody could see everybody else's comments.

From that MIT conference, we carried away four committees of people who wanted to be directly involved in making the license. The committee I chaired had some of the world's largest IT patent holding companies: IBM and Hewlett-Packard, as well as Red Hat, Apple, and Intel. We spent a lot of time talking about the patent provisions. But all the committees were busy.

I ran that process out of the new Software Freedom Law Center with my then legal assistant, Richard Fontana, who is now a fairly senior Red Hat lawyer, and Richard [Stallman], and the FSF old GPLv3 team managing all the conference calls, mailing lists, and all the rest of it. A pretty extensive legislative history of the GPLv3 license is still available at gplv3.fsf.org. Sometimes, the discussion was contentious, and sometimes it was a complex negotiation among multiple companies and parties with many different views of how the world should operate. But, mostly, I think of it as a great big seminar of these issues – complex, with lots of teaching and learning going on. I don't want to sign up to do it again, but those 16 months of my life that I spent with Richard on GPLv3 were a very great learning experience indeed.

LM: Was it a disappointment when the Linux kernel developers decided not to adopt GPLv3?

EM: Well, I guess it's a disappointment in the sense that I had some optimism that we might actually be able to produce



Too busy to wade through press releases and chatty tech news sites? Let us deliver the most relevant news, technical articles, and tool tips – straight to your inbox.

ADMIN Update • ADMIN HPC
Linux Update • Raspberry Pi

Keep your finger on the pulse of the IT industry.

Admin and HPC:
www.admin-magazine.com/newsletter

Linux Update:
www.linuxpromagazine.com/mc/subscribe

Raspberry Pi:
www.raspberry-pi-geek.com/mc/subscribe



a license that they would want to use. That it was possible that the kernel would remain under GPLv2 seemed pretty likely all the way along. I would say that probably from roughly September of 2006, I assumed that the kernel would remain using GPLv2, at least for the short term. I did not think at that time that Java was going to be under GPLv2. I thought that Java would move to GPLv3.

Now [Java and MySQL maintainer] Oracle is the world's largest GPLv2 licensor. The kernel is one part the GPLv2 user community, but it isn't the only part. I like GPLv3 very much, as I do GPLv2, and I think GPLv3 is a better license, but I do not think that either one of them is the only license to use. I think copyleft has enormous social importance, and I think both GPL and GPLv2 and GPLv3, along with their associated families of licenses, are good copyleft licenses. I'm not disappointed when anybody uses any one of them.

LM: Describe a little more about what you're doing now, with the Software Freedom Law Center (SFLC).

EM: Yeah, I'm not the legal director, my law partner Mishu Choudhary is. My titles around here are President and Executive Director, which basically means I figure out how we get the money. SFLC is a 501c3 charity, which is chartered also as an educational foundation by the state of New York. It's a teacher practice. I teach people here how to be working lawyers for FOSS. That means that we have a lot of FOSS clients. Mishu handles the law practice; she makes sure that everything gets done and that clients are communicated with and that their instructions are taken and executed.

What we do is provide free legal representation to nonprofits that make and distribute free and open source software. Our clients include all the people you would expect: Samba, OpenSSL, and the Apache Software Foundation, as well as all sorts of people in the blockchain world. In pretty much every direction, we have some relationship to one or more projects providing free legal help.

LM: Is this a network of volunteers, who are working with you, or do you just do it all yourself?

EM: This is a training process. The volunteers we have are law students learning. The lawyers we have are lawyers who work for us in New York and New Delhi. We are supported by the donations of companies that know that a stronger ecosystem with better lawyers for the nonprofits is good for them, and we are very grateful for the generous support of companies like IBM, HP, Oracle, Red Hat, Qualcomm, and so on.

With some [for-profit] companies and with individual developers who need help with their employment or other arrangements – for which we are not given money by our donors to provide for free – we provide commercial legal assistance. We charge as reasonably as we can, because we're experts we don't need to spend a lot of extra time on things. All the work we do is in support of the general mission of helping people use clean open source software correctly. But when business or other solvent parties can afford to pay us for our help, they do! And, we give advice to companies and other entities in the world trying to understand the Free Software communities, how do they work, how do we make them better, and so on.

LM: So, if I knew somebody who was starting a free software project, and they had some questions about licensing, I could just tell them to call Software Freedom Law Center?

EM: Yes, they would write to the email address help at *SoftwareFreedom.org*, and we would go through a little back and forth to say we're not your lawyers yet, and this isn't secret, but we'll help you if we can using public resources. But, if it looks to us like the problem is one that can't be resolved by pointing somebody at a publication, we would say "look, we'll send you a retainer agreement, which says we're going to work for you for nothing, and if you sign off, we'll give you the help you need."

LM: Free Software has certainly come a long way over the years; if there were anything you could fix, anything missing you would like to see, what would it be?

EM: I wish I thought that all we had left was our wish list. We have some significant challenges. We are not in a world where copyleft is just like all other free software; it never was and isn't now. Most of the free software licensing that we have, what we call permissive licensing, is designed to create respect for programmers' rights. Respect for programmers' rights is very important, and it's great to do that, and I'm not going to talk it down at all, but the unique part of copyleft is that it protects users' rights. And the world where we live right now, if your rights as a user of that thing [phone] you are holding in your hand aren't respected, then you don't have any political or social rights at all.

It is particularly easy to teach this message today. You go around the world and you hold up a smart-ass phone, and you say "Folks, if you don't have any rights to know what's going on in here, then you don't have any liberty." That used to be a hard lesson, but people now realize that they are dependent upon technology networks for everything in their lives, and if they don't have rights respected, they're in trouble, because their other rights depend on the technology. But we do not have uniform respect for copyleft as a way of protecting users' rights around the world. Governments don't care quite so much about users' rights; the FCC and other regulatory agencies don't care much about users' rights.

We're not in a world where we can sit back and worry about our wish list; we have to explain to people again why copyleft is so important. We have to make sure that federal research grants in the United States don't say, as an increasing number of grants from the NSF or DARPA say, "You must make all your software under open source license, but no bad viral licenses like GPL or NPL." I've got dozens of those funding solicitations now being published over the last year, and that has become a major headache for me.

I've just been given the gift of working again on the Board of Directors of SFLC with my old friend and colleague Daniel Weitzner, who was in the White House office of Science and Technology policy under President Obama. Danny and I are going to be talking to federal research entities about the importance of not discriminating against copyleft. Similar problems are beginning to crop up elsewhere in the world outside of the United States.

Our job now is selling copyleft and making sure that everybody understands why it's so important – that licenses that respect users' rights are crucial. They're not a by-the-way or an accident or an anti-business this or an anti-capitalist that. Licenses that respect users' rights are an attempt to secure technological liberty in the 21st century, without which social and political liberty does not exist. And that's not less of a job than it was 20 years ago; that's *more* of a job than it was 20 years ago. ■■■

An entire year of
**RASPBERRY PI PROJECTS
AND OPERATING TIPS**
for one low price!
Two bundles to choose from:



Hurry while supply lasts!

ORDER NOW!



<http://shop.linuxnewmedia.com/us/magazines/raspberry-pi-geek/catchup.html>

Five lean tools for monitoring logfiles

Small Supervisor

Anyone wanting to monitor logfiles could use one of the big dogs like Nagios or Icinga. However, lightweight alternatives can also sniff out threats and take much less time to set up. We put five of these little guard dogs to the test.

By Tim Schürmann



A system's logfiles not only record failed login attempts by users, but they also log program errors and information about attacks. Admins therefore should keep a continuous eye on them. Tools

such as LOGalyze [1], Logcheck [2], Logwatch [3], MultiTail [4], and SwatchDog [5] can help you here.

Unlike with large monitoring solutions like Nagios and Icinga, the minor variants focus on analyzing logfiles. They

use fewer resources and can be set up much more quickly. They are therefore ideally suited for use on weak hardware and embedded devices like the Raspberry Pi, as well as on servers with few selected services.

TABLE 1: Tools for Logfile Monitoring

Name	LOGalyze	Logcheck	Logwatch	MultiTail	SwatchDog
URL	http://www.logalyze.com	http://logcheck.اليو.ديبان.ورج	https://sourceforge.net/projects/logwatch/	https://www.vanheusden.com/multitail/	https://sourceforge.net/projects/swatch/
Tested version	4.1.4	1.3.18	7.4.3	6.4.2	3.2.4
License	GNU GPLv2	GNU GPLv2	MIT license	GNU GPLv2	GNU GPLv2
Filtering / with regular expressions	yes / no	yes / yes	yes / no	yes / yes	yes / yes
Notification by email	yes	yes	yes	no (via external program)	yes
Permanent monitoring of a log	yes	yes	yes	yes	yes
Unique processing of a complete log	no	no	yes	no	yes
Information about security problems	no	yes (limited)	yes (limited)	no	no
Summary / statistics	yes	no	yes	no	no
GUI	yes	no	no	no	no

Lead image © nishop1, 123RF.com

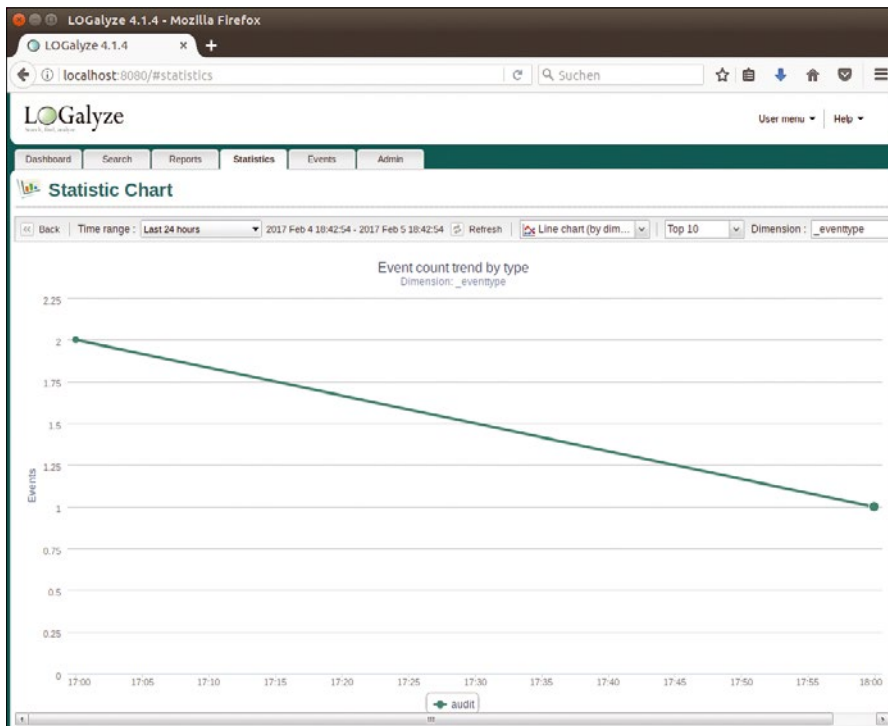


Figure 1: LOGalyze displays statistics directly in the web interface as a pie, bar, or line chart. Here it is clear that the number of events dropped in recent times.

All candidates use one or more logfiles and filter out important messages according to predefined rules. As an option, they can send the result by email to the administrator or output it on the command line. Admins can also add their own filter rules, usually in the form of regular expressions. Sometimes the developer provides a set of oft-needed rules. Powerful tools may also put together a report about the state of the system and indicate security problems. However, a comparison of the above candidates shows that these functions are not a matter of course (Table 1).

LOGalyze

LOGalyze [1] comes from the Hungarian company Zuriel Ltd. The formerly proprietary tool may now be available under GPLv2, but the developers are still keeping the source code under lock and key. The latest version 4.1.4 was released in December 2016, but it only fixes minor bugs from the almost four-year-old previous version. LOGalyze therefore still relies on Oracle's Java runtime environment in the completely outdated version 1.6.

A short and concise guide in PDF format provides a description of the installation. Administrators can configure LOGalyze using a supplied web application that

requires one of the application servers Apache Tomcat, Jetty, GlassFish, or JBoss. The hopelessly outdated Tomcat 6.0.35 from 2011 is included with the installation package; however, it can be quickly booted using a prepared script.

Initially, administrators can create one or more collectors in the user interface. These collectors retrieve the

log data via the network or from a file. Admins can switch each collector on or off individually. LOGalyze then generates statistics and reports from the imported data and summarizes all critical errors, for example, in a concise report (Figure 1).

Admins can also create their own statistics and reports by clicking the corresponding criteria in the user interface. LOGalyze then generates either a PDF or CSV file on this basis. Admins can either download these files or have them sent by email.

They can also search the logfiles for terms. LOGalyze may not allow regular expressions, but it does link several search terms using operators like AND and OR. It stores frequently required search queries to allow quick retrieval later via mouse click (Figure 2). LOGalyze provides plenty of predefined searches which, among other things, quickly list all the errors from the syslog.

Logcheck

The Debian project currently looks after Logcheck [2], which is available under GPLv2. It independently assesses logfiles for problems, security vulnerabilities, and possible intrusion attempts. After it's started, Logcheck accesses the syslog and the auth.log by default. However, you can use the tool on other logfiles.

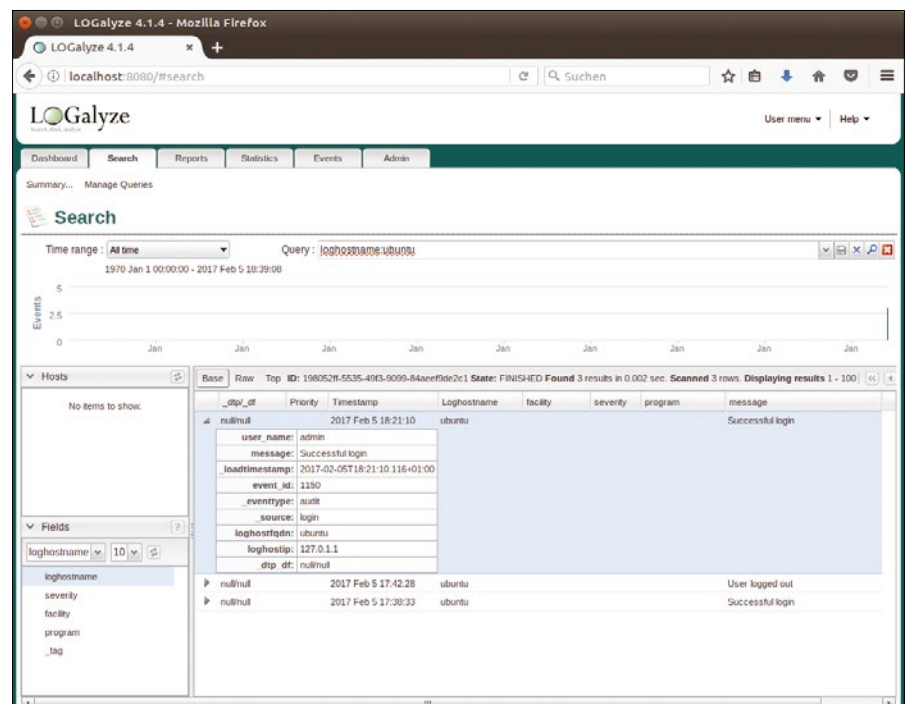


Figure 2: LOGalyze has found three events for the ubuntu computer.

```

tim@ubuntu:~$ sudo -u logcheck logcheck -o
This email is sent by logcheck. If you no longer wish to receive
such mail, you can either deinstall the logcheck package or modify
its configuration file (/etc/logcheck/logcheck.conf).

System Events
=====
Feb 6 12:45:55 ubuntu gnome-terminal-[4968]: Allocating size to GtkWidget 0x7f3af0
012ba0 without calling gtk_widget_get_preferred_width/height(). How does the cod
e know the size to allocate?
Feb 6 12:45:59 ubuntu dbus-daemon[3070]: Activating service name='org.gnome.Scr
eenshot'
Feb 6 12:45:59 ubuntu dbus-daemon[3070]: Successfully activated service 'org.gn
ome.Screenshot'
Feb 6 12:46:04 ubuntu dbus[734]: [system] Activating via systemd: service name=
'org.freedesktop.hostname1' unit='dbus-org.freedesktop.hostname1.service'
Feb 6 12:46:04 ubuntu systemd[1]: Starting Hostname Service...
Feb 6 12:46:04 ubuntu dbus[734]: [system] Successfully activated service 'org.f
reedesktop.hostname1'
Feb 6 12:46:04 ubuntu systemd[1]: Started Hostname Service.
Feb 6 12:46:05 ubuntu gvfsd[3101]: mkdir failed on directory /var/cache/samba:
Keine Berechtigung
Feb 6 12:46:07 ubuntu gvfsd[3101]: message repeated 5 times: [ mkdir failed on
directory /var/cache/samba: Keine Berechtigung]
Feb 6 12:46:07 ubuntu systemd-resolved[1057]: Using degraded feature set (UDP)
for DNS server 127.0.1.1.
Feb 6 12:46:09 ubuntu gvfsd[3101]: mkdir failed on directory /var/cache/samba:
Keine Berechtigung
    
```

Figure 3: Ubuntu users can start Logcheck with the help of the `logcheck` user. The `-o` parameter outputs the events in the terminal. The outputs can be quite confusing.

Logcheck compares all records added since the last test with a load of preset filtering rules. Depending on the result, the tool either moves directly to the next entry or classifies it as an important system event, security issue, or warning. Logcheck then sends all the events from the last three categories in an email to the administrator or writes them in the standard output (Figure 3).

Administrators can choose between three filter levels to maintain an overview: The highest, called `paranoid`, is intended for particularly safety-relevant systems with a few selected services. In this filter level, Logcheck provides an especially large number of detailed messages that it would discard in the other levels. The `Server` level is the default, and there are the least messages in the `Workstation` level. Logcheck sends warnings about security issues and attacks in each filter level. To ensure that the tool only reports each system event once, it remembers the last position in the logfiles to be assessed with the help of the Perl script `Logtail`.

All filter rules are available as regular expressions, so that admins can add their own, as desired (Figure 4). To provide a better overview, all of the expressions for a service, such as the Apache web server, are moved to a separate configuration file. When started, Logcheck

automatically imports all configuration files. The developers kindly provide a package with several of these configuration files. However, the rules contained in this Logcheck database only cover a few basic errors and particularly important attack patterns.

Most distributions have Logcheck in their repositories. On Debian systems, a cron job initiates Logcheck every hour, and the attentive tool is automatically activated at every system startup. Any suitable command-line program assumes the responsibility for sending emails – the task is assigned to `Sendmail` by default.

Logwatch

Like Logcheck, Logwatch [3] is waiting to be installed in the repositories of most major distributions. The tool is available

```

tim@ubuntu:~$ sudo more /etc/logcheck/ignore.d.server/sudo
^w{3} [ :0-9]{11} [._[:alnum:]-]+ sudo: pam [[:alnum:]]+\(sudo:session\)
: session closed for user [[:alnum:]]+
^w{3} [ :0-9]{11} [._[:alnum:]-]+ sudo: pam [[:alnum:]]+\(sudo:session\)
: session opened for user [[:alnum:]]+ by ([[:alnum:]]+)?\(\(uid=[0-9]+\)\)
^w{3} [ :0-9]{11} [._[:alnum:]-]+ sudo:[[:space:]]+[[:alnum:]]+ : TTY=
(unknown|pts/|tty|vc/)[[:digit:]]+ ; PWD=[[:alnum:]]+ ; USER=[._[:alnum:]]+ ;
COMMAND=(/usr/etc/bin/sbin|/sudoedit) .*$
^w{3} [ :0-9]{11} [._[:alnum:]-]+ sudo:[[:space:]]+[[:alnum:]]+ : \(co
mmand continued\) .*$
tim@ubuntu:~$
    
```

Figure 4: Among other things, in the `Server` filter level, Logcheck discards all the events that correspond to these regular expressions in the `/etc/logcheck/ignore.d.server/sudo` file.

under the MIT license and requires Perl 5.8. Once Logwatch starts, it accesses all the logs known to it and checks all the events from the last 24 hours in them. Admins can extend or shorten this observation period at their discretion.

Unlike with the competitor Logcheck, Logwatch generates a concise summary (Figures 5 and 6) from the read events. A separate section is devoted to each of the services running on the system, and the information displayed there is based on the respective service. For example, Logwatch lists all the packages installed in the past 24 hours for the package manager `dpkg`. If Logwatch was unable to meaningfully interpret an event, the tool simply attaches this event to the report as an attachment.

The administrator informs Logwatch about numerous configuration files, about which services are running on the system, and about in which logfiles the services usually store their information. In turn, the configuration files are distributed across several subdirectories. Logwatch provides finished configuration files for many important and well-known system services, and some distributors supplement them with additional services. Thanks to these specifications, Logwatch scours more logfiles immediately after the installation than its competitor Logcheck.

A specialized Perl script analyzes a service's logfiles. For example, the `/usr/share/logwatch/scripts/services/dpkg` script processes the logfiles of the package manager `dpkg`. If Logwatch is to analyze an individually compiled service for the administrator, it needs to be able to write an evaluation script in Perl. A detailed how-to included with Logwatch helps here.

The analysis scripts import the events from the logfiles, generate a summary,

```

tim@ubuntu:~$ logwatch
##### Logwatch 7.4.3 (04/27/16) #####
Processing Initiated: Sun Feb 5 16:39:25 2017
Date Range Processed: yesterday
                    ( 2017-Feb-04 )
                    Period is day.
Detail Level of Output: 0
Type of Output/Format: stdout / text
Logfiles for Host: ubuntu
#####

----- dpkg status changes Begin -----

Installed:
libipc-signal-perl:all 1.00-6.2
liblockfile-bin:amd64 1.09-6ubuntu1
liblockfile1:amd64 1.09-6ubuntu1
libmime-types-perl:all 2.13-1
libproc-waitstat-perl:all 1.00-4.2
lockfile-progs:amd64 0.1.17
logcheck-database:all 1.3.17
logcheck:all 1.3.17
logtail:all 1.3.17
mime-construct:all 1.11+nmu2
postfix:amd64 3.1.0-5

----- dpkg status changes End -----

----- pam_unix Begin -----

lightdm:
Unknown Entries:
  session opened for user tim by (uid=0): 1 Time(s)

lightdm-greeter:
Unknown Entries:
  session closed for user lightdm: 1 Time(s)
  session opened for user lightdm by (uid=0): 1 Time(s)

polkit-1:
Unknown Entries:
  session opened for user root by (uid=1000): 4 Time(s)
    
```

Figure 5: Logwatch can also generate the summary in text form ...

and pass it on to Logwatch. Ultimately, the tool sends the collected results in an email, writes them to a file, or delivers them via stdout. In the past, Logwatch presented all information on a simple HTML page upon request (Figure 6). The final report also contains some system information, for example, the available disk space.

Admins can also dictate the detail of Logwatch's report. There are a total of 10 detail levels available. The individual evaluation scripts determine which (additional) information each level of detail produces. The distributions generally start Logwatch via cron job once at night and send the generated report via email. The tool delegates the actual sending to Sendmail or another, freely selectable command-line program.

MultiTail

MultiTail [4] presents the ends of several text files in only one (terminal) window

(Figure 7). Additional features were added over the course of time, in particular filter and monitoring functions for logfiles.

The admins can thus filter the flood of information using regular expressions. If a regular expression applies, MultiTail launches an external program upon request. You can be sent email notifications, for example. Alternatively, MultiTail works like a visual pipe in that it writes the filtered information in files or forwards it to other processes. MultiTail can even act as a syslog server itself and accept outputs from other programs upon request – such as netstat.

The tool also highlights the row concerned and attracts attention via a beeper. An admin can highlight individual events in specific colors using regular expressions. For example, you can highlight in red all rows starting with Error. MultiTail also automatically converts inputs. It converts IP addresses into the appropriate domain name, converts signal numbers into names, and supplies each date in the local format.

However, administrators need to set up MultiTail completely on their own. Unlike Logcheck or Logwatch, the tool does not have any pre-made regular expressions. Admins also need to manually configure the forwarding and sending of emails. MultiTail is included with all major distributions and is available under GPLv2. The tool also comes with a detailed manual in HTML format.

SwatchDog

The Simple Log Watcher, Swatch for short [5], began as a small watchdog

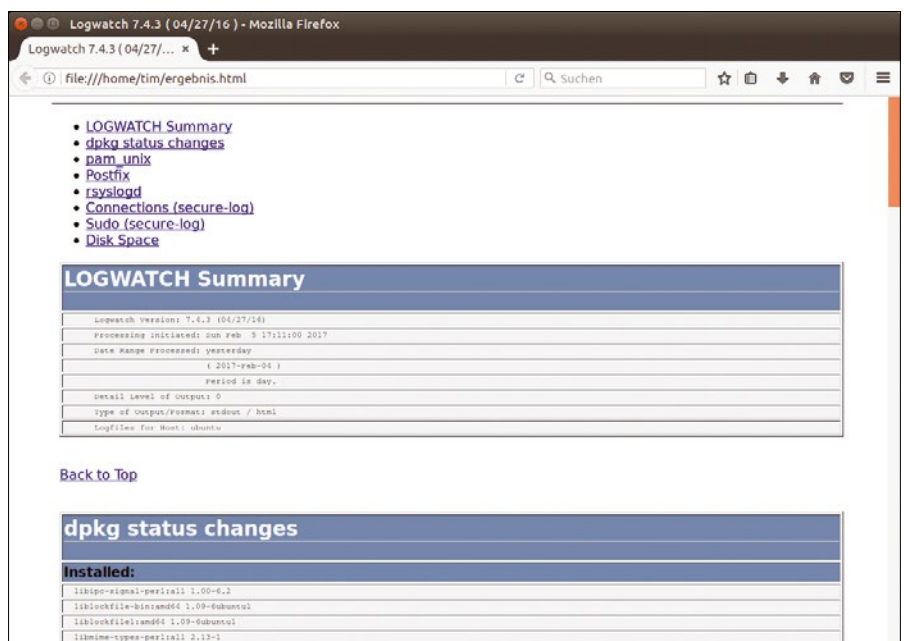


Figure 6: ... or as a HTML file with rather small font.

```

tim@ubuntu: /var/log/syslog (Sun Feb 5 18:03:18 2017) [0.170000]
Feb 5 17:55:35 ubuntu gnome-session[3575]: gnome-session-binary[3575]: Glib-GIO-CRITICAL: g_dbus_connection_call_internal: assertion 'object_path != NULL && g_variant_is_object_path(object_path)' failed
Feb 5 17:55:35 ubuntu gnome-session-binary[3575]: Glib-GIO-CRITICAL: g_dbus_connection_call_internal: assertion 'object_path != NULL && g_variant_is_object_path(object_path)' failed
Feb 5 17:55:35 ubuntu unity-panel-ser[3862]: menus destroyed: assertion 'IS_WINDOW_MENU(wm)' failed
Feb 5 17:55:43 ubuntu unity-panel-ser[3862]: menus destroyed: assertion 'IS_WINDOW_MENU(wm)' failed
Feb 5 17:58:58 ubuntu zeitgeist-datat[4135]: zeitgeist-datatub.vala:212: Error during inserting events: GDBus.Error:org.gnome.zitgeist.EngineError.InvalidArgument: Incomplete event: interpretation, manifestation and actor are required
Feb 5 18:02:01 ubuntu CRON[15066]: (logcheck) CMD ( if [ -x /usr/sbin/logcheck ]; then nice -n10 /usr/sbin/logcheck; fi)
Feb 5 18:02:04 ubuntu postfix/pickup[14286]: 1122012B05A: uid=126 from=<logcheck>
Feb 5 18:02:04 ubuntu postfix/cleanup[16023]: 1122012B05A: message-id=<20170205170204.1122012B05A@ubuntu.fritz.box>
Feb 5 18:02:04 ubuntu postfix/qmgr[1846]: 1122012B05A: from=<logcheck@ubuntu.fritz.box>, size=6491, nrcpt=1 (queue active)
Feb 5 18:02:04 ubuntu postfix/local[16028]: 1122012B05A: to=<root@ubuntu.fritz.box>, orig_to=<logcheck>, relay=local, delay=0.61, delays=0.39/0.13/0.08, dsn=2.0.0, status=sent (delivered to mailbox)
Feb 5 18:02:04 ubuntu postfix/qmgr[1846]: 1122012B05A: removed
Feb 5 18:03:03 ubuntu gnome-terminal-[5155]: Allocating size to GtkBox 0x7fd74003ba0 without calling gtk_widget_get_preferred_width/height(). How does the code know the size to allocate?
Feb 5 18:03:18 ubuntu dbus-daemon[3109]: Activating service name='org.gnome.Screenshot'
Feb 5 18:03:18 ubuntu dbus-daemon[3109]: Successfully activated service 'org.gnome.Screenshot'
[0] /var/log/syslog:18KB - 2017/02/05 18:03:18
Feb 5 16:11:22 ubuntu pkexec[11967]: tim: Executing command [USER=root] [TTY=unknown] [CWD=/home/tim] [COMMAND=/usr/lib/update-notifier/package-system-locked]
Feb 5 16:17:01 ubuntu CRON[12047]: pam_unix(cron:session): session opened for user root by (uid=0)
Feb 5 16:17:01 ubuntu CRON[12047]: pam_unix(cron:session): session closed for user root
Feb 5 17:02:01 ubuntu CRON[12578]: pam_unix(cron:session): session opened for user logcheck by (uid=0)
Feb 5 17:02:04 ubuntu CRON[12578]: pam_unix(cron:session): session closed for user logcheck
Feb 5 17:17:01 ubuntu CRON[14063]: pam_unix(cron:session): session opened for user root by (uid=0)
Feb 5 17:17:01 ubuntu CRON[14063]: pam_unix(cron:session): session closed for user root
Feb 5 17:43:15 ubuntu polkit-agent-helper-1[14521]: pam_unix(polkit-1:auth): authentication failure; logname= uid=1000 euid=0 tty= ruser=tim rhost= user=tim
Feb 5 17:43:19 ubuntu polkitd(authority=local): Operator of unix-session:c2 successfully authenticated as unix-user:tim to gain ONE-SHOT authorization for action com.ubuntu.pkexec.synthetic for unix-process:14515:621402 [/bin/sh /usr/bin/synaptic-pkexec] (owned by unix-user:tim)
Feb 5 17:43:19 ubuntu pkexec: pam_unix(polkit-1:session): session opened for user root by (uid=1000)
Feb 5 17:43:19 ubuntu pkexec[14517]: tim: Executing command [USER=root] [TTY=unknown] [CWD=/home/tim] [COMMAND=/usr/sbin/synaptic]
Feb 5 17:44:22 ubuntu pkexec: pam_unix(polkit-1:session): session opened for user root by (uid=1000)
Feb 5 17:44:22 ubuntu pkexec[14968]: tim: Executing command [USER=root] [TTY=unknown] [CWD=/home/tim] [COMMAND=/usr/lib/update-notifier/package-system-locked]
Feb 5 18:02:01 ubuntu CRON[15065]: pam_unix(cron:session): session opened for user logcheck by (uid=0)
Feb 5 18:02:04 ubuntu CRON[15065]: pam_unix(cron:session): session closed for user logcheck
[0] /var/log/auth.log:4KB - 2017/02/05 18:03:15
    
```

Figure 7: Admins can determine the order of the logfiles themselves in MultiTail. They just need to depict two logfiles one above the other.

tool that monitored a syslog for activities. The program, which is available under GPLv2, now digests any logfiles. Formally, the command-line tool is called SwatchDog to avoid any confusion with a well-known Swiss watch manufacturer. However, in most distributions, it is in the swatch package, and the man page is the only documentation.

The tool itself consists of a small Perl script that assesses the logfiles stated via parameter (Figure 8). SwatchDog either goes through all the rows contained in the files or continuously monitors the file. In the latter case, SwatchDog can be started as a daemon and thus move to the background. Upon request, the tool also accepts log data via a pipe.

In any case, you can specify in a configuration file for which events the tool needs to perform which actions. This configuration file uses SwatchDog's own syntax – Listing 1 shows a simple example. According to the instructions shown there, SwatchDog needs to search for the keywords warning and error. Administrators can use a regular expression for specifying the search pattern.

SwatchDog then performs all the actions that follow in the indented list. For example, echo outputs the corresponding line from the logfile on the console,

while mail sends the message by email with the subject line Error occurred to tim@example.com. SwatchDog also calls any programs (exec) and forwards the affected event via pipe (pipe command). Perl experts can store Perl code, which the tool executes.

Users start SwatchDog on the command line by default. Administrators need to create a suitable cron job or

LISTING 1: Configuration File .swatchrc

```

01 watchfor /warning|error/
02     echo
03     mail addresses=tim@example.com, subject=error occurred
    
```

```

tim@ubuntu: ~
tim@ubuntu:~$ swatch --examine /var/log/auth.log

*** swatch version 3.2.3 (pid:7410) started at So 5. Feb 23:14:03 CET 2017

Use of uninitialized value $attr in concatenation (.) or string at /usr/share/perl/5.22/Term/ANSIColor.pm line 482, <GEN2> line 65.
Feb 5 19:31:05 ubuntu dbus[745]: [system] Rejected send message, 1 matched rule s; type="method call", sender=":1.67" (uid=1000 pid=3697 comm="/usr/lib/x86_64-linux-gnu/indicator-sound/indicato") interface="org.freedesktop.DBus.Properties" member="GetAll" error name="(unset)" requested_reply="0" destination=":1.44" (uid=0 pid=2696 comm="/usr/lib/x86_64-linux-gnu/unity-greeter-session-br")
Use of uninitialized value $attr in concatenation (.) or string at /usr/share/perl/5.22/Term/ANSIColor.pm line 482, <GEN2> line 100.
Feb 5 22:39:00 ubuntu dbus[739]: [system] Rejected send message, 1 matched rule s; type="method call", sender=":1.70" (uid=1000 pid=3426 comm="/usr/lib/x86_64-linux-gnu/indicator-sound/indicato") interface="org.freedesktop.DBus.Properties" member="GetAll" error name="(unset)" requested_reply="0" destination=":1.46" (uid=0 pid=2750 comm="/usr/lib/x86_64-linux-gnu/unity-greeter-session-br")
tim@ubuntu:~$
    
```

Figure 8: SwatchDog checks the whole /var/logs/auth.log file once here based on the --examine parameter.

systemd units themselves. Unlike with Logwatch, SwatchDog does not provide an example configuration. Users should therefore initially plan a bit of time to write a suitable configuration file.

Learn to Love the Dog

The choice of the appropriate tool massively depends on the specific requirements and your personal programming skills. None of the five candidates can replace a full-scale monitoring system, let alone an intrusion detection system. In any case, administrators need to interpret the sent system events themselves. See the “Old Comrades” box for some other alternatives.

LOGalyze provides a GUI and can also be remotely operated via your browser. However, anyone who wants to use the tool should remember the tool's age. The supplied Tomcat version also needs to be replaced as quickly as possible. Admins also need to be able to figure out for themselves how to use LOGalyze.

Logcheck can be put into operation particularly quickly. Anyone who masters regular expressions can reduce the flood of data using quickly added and customized filtering rules. While Logcheck only sends the naked events to the admin, Logwatch provides the admin with a summary. If

admins want to monitor their own services using Logwatch, they need Perl scripts.

MultiTail is worthwhile for administrators who literally want to keep an eye on several logfiles and only want to trigger

actions in certain cases. Sending emails and forwarding filtered events may be possible; however, to do so admins need to write suitable regular expressions and manually configure MultiTail. The tool is therefore useful as a very good supplement to Logcheck and Logwatch.

Finally, SwatchDog is comparable to Logwatch: It can be set up quickly but requires knowledge of regular expressions. Additionally, it only reports individual events specified by the administrator.

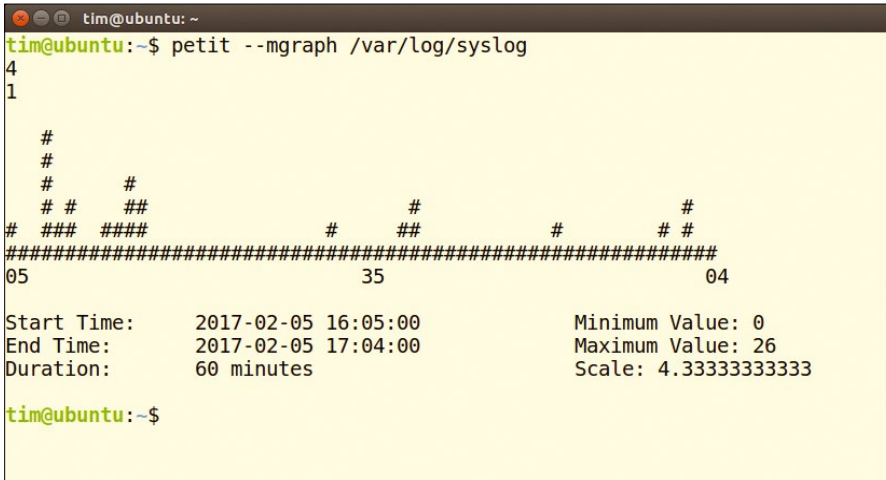


Figure 9: As this diagram of Petit shows, a lot of events were entered in the first five minutes of the log recording.

OLD COMRADES

Anyone searching the Internet for other alternatives to the featured tools will automatically stumble upon a few representatives that are now obsolete. The Logdigest [6] tool works like Logcheck, but has been on ice since 2009. LogSurfer [7] is pretty similar to SwatchDog, but can also group similar events. In addition, LogSurfer is written in C and should therefore work much more quickly. However, the most recent version of the tool was released in September 2011.

Petit [8] is about the same age, but it is still in the repositories of Ubuntu. The tool uses language analysis methods to curb the flood of data, especially in system logs. This allows administrators to, for example, list all words that occur particularly frequently in a logfile. In addition, the tool draws a graph that presents the number of messages in a given period of time (Figure 9). The hash function, which keeps track of similar messages in the log, is also interesting. It allows the viewer to immediately see which errors occur most frequently.

INFO

- [1] LOGalyze: <http://www.logalyze.com>
- [2] Logcheck: <http://logcheck.alioth.debian.org>
- [3] Logwatch: <https://sourceforge.net/projects/logwatch/>
- [4] MultiTail: <https://www.vanheusden.com/multitail/>
- [5] SwatchDog: <https://sourceforge.net/projects/swatch/>
- [6] Logdigest: <https://sourceforge.net/projects/logdigest/>
- [7] LogSurfer: <http://www.crypt.gen.nz/logsurfer/>
- [8] Petit: <http://crunchtools.com/software/petit/>



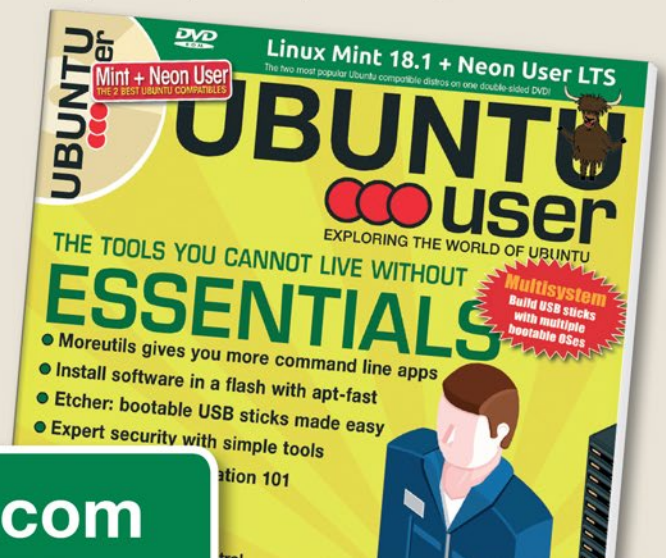
Find us on Facebook
<http://www.facebook.com/linuxpromagazine>

DON'T MISS A SINGLE ISSUE!

The first print magazine created specifically for Ubuntu users! Ease into Ubuntu with the helpful Discovery Guide, or advance your skills with in-depth technical articles, HOW-TOs, reviews, tutorials, and much, much more.

SUBSCRIBE NOW!
 4 issues per year for only
 £ 24.90 / EUR 29.90 / US\$ 39.95

- ✓ Don't miss a single issue!
- ✓ Huge savings – Save more than 35% off the cover price!
- ✓ Free DVD – Each issue includes a Free DVD!



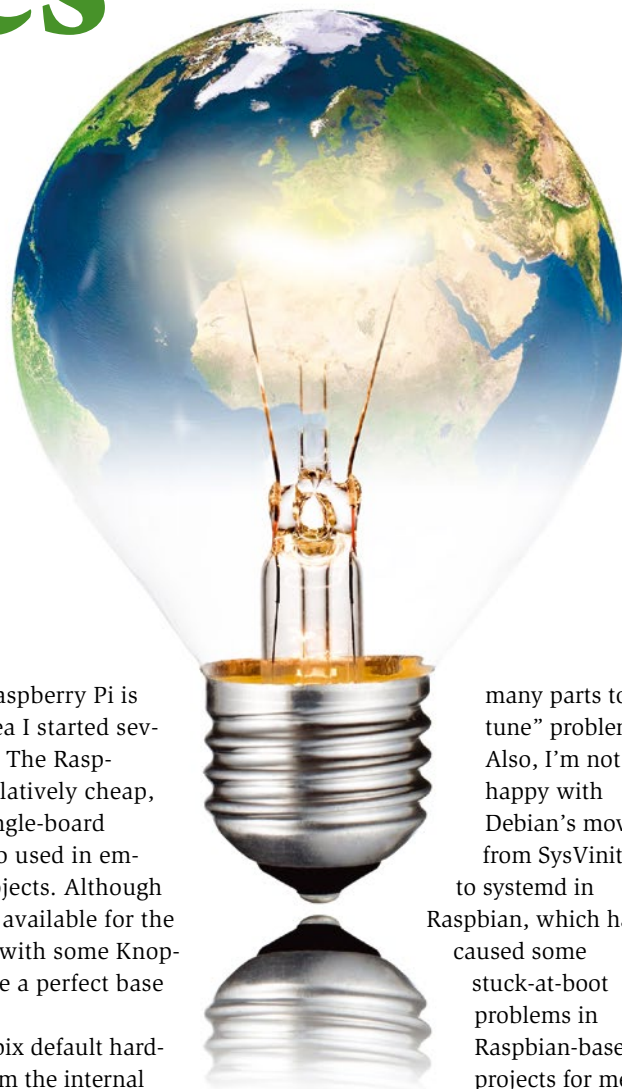
shop.linuxnewmedia.com

Building a Yocto-based Knoppix for the Raspberry Pi

Pi Recipes

The professor sets the stage for Knoppix on an SD card that runs on the Raspberry Pi.

By Klaus Knopper



Knoppix on a Raspberry Pi is an ongoing idea I started several years ago. The Raspberry Pi is a relatively cheap, yet widely available single-board computer, which is also used in embedded and mobile projects. Although many distributions are available for the Pi, creating a new one with some Knoppix technology could be a perfect base for projects.

Because of the Knoppix default hardware setting to boot from the internal SD card reader, the boot procedure already has many similarities with existing distributions. However, Raspbian (the Debian derivate) for Raspberry Pi increased complexity and became a “too

many parts to tune” problem. Also, I’m not happy with Debian’s move from SysVinit to systemd in Raspbian, which has caused some stuck-at-boot problems in Raspbian-based projects for me.

Therefore, I decided to give the “build-from-scratch” approach a try (Table 1) instead of basing Knoppix on Debian, as in previous versions.



Klaus Knopper is an engineer, creator of Knoppix, and co-founder of LinuxTag expo. He works as a regular professor at the University of Applied Sciences, Kaiserslautern, Germany.

TABLE 1: Pros and Cons of Knoppix Built from Scratch

Pros	Cons
The complete software is self-built and linked only with components included in the system	May be incompatible with large software repositories like Raspbian
No unwanted “leftover” packages	Have to create build instructions for new programs
Smallest system possible, which can be a base for incremental projects	Needs recompilation with changes or add-ons
No side effects expected from binary upstream	Updates have to get pulled in at the source level

Lead image © Jan Treger, 123RF.com

Yocto and OpenEmbedded

Yocto is a framework for creating custom Linux distributions for specialized embedded hardware, as well as for standard boards. Yocto is based on OpenEmbedded, a build framework for embedded Linux [1].

Yocto [2] provides a sample distribution called “Poky,” which is filled with templates for the so-called meta-layers. These are directories containing build *recipes* for software components, which can be thought of as stacked on top of each other, so you have a base layer for the hardware-dependent parts and features, and incremental layers for graphical environments and user software.

Optimally, all changes toward the standard meta-directories that come with the Yocto distribution would be placed into one or more self-created directories with your own recipes, add-ons, and override rules.

Striving to “guarantee 100% reproducible builds,” Yocto first downloads, compiles, and installs a completely new development environment from original sources for the host and target architectures, so you have the same version of compilers and libraries on each build system, with no errors attributable to different versions or features. Still, to run Yocto scripts, the host distribution needs to meet some requirements:

- Git 1.8.3.1 or greater
- tar 1.24 or greater
- Python 3.4.0 or greater.

For Debian as the hosting distribution, the following installation and update command is provided in the documentation [3]:

```
sudo apt-get install gawk wget \
git-core diffstat unzip texinfo \
gcc-multilib build-essential chrpath \
socat cpio python python3 python3-pip \
python3-pexpect xz-utils debianutils \
iputils-ping
```

After these packages are installed, Yocto is supposed to be able to build all software packages and images included in the Poky sample distribution. The integration hierarchy of Yocto is shown in Figure 1.

Layer Cake

The Yocto framework itself is rather small after installation; uncompressed, it is less than 100MB. During the build process, sources are downloaded via git or http, unpacked, and compiled, which fills up the `build/tmp` directory with a tremendous amount of data (Table 2).

When first using Yocto, I found it somewhat unusual that all the seemingly *important* build stuff, including compilers and supporting binaries as well as the final generated SD card image, resided in a directory named `tmp`. Even more remarkable is that when removing

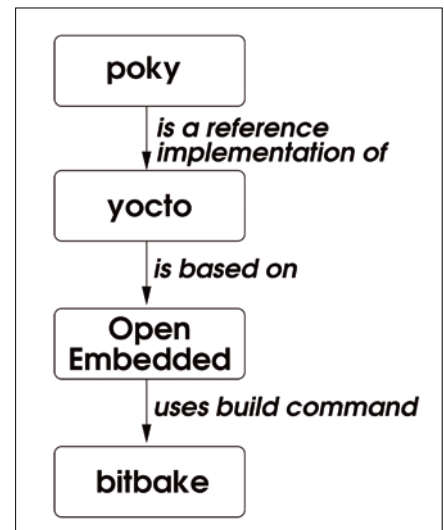


Figure 1: Yocto integration hierarchy.

the `tmp` directory and its contents completely, rebuilding the project takes less than a tenth of the original time, because most files will be restored quickly from the `sstate-cache`.

Setting Up Yocto

Before starting the build process, an initial setup needs to be done. I’m using 32-bit userspace binaries on a 64-bit kernel, which is a special setup that Yocto does not yet handle well. For this reason, I start a new shell that changes the output of `uname -m` from `x86_64` to `i686`, which ensures Yocto will download and compile all supporting tools for the correct host architecture:

```
setarch i386 /bin/bash
```

Afterward, paths and the environment need to be configured in the `poky` directory for the current shell:

```
cd yocto/poky
source oe-init-build-env
```

These commands have to be repeated whenever I restart the build procedure from a new shell.

The resulting architecture and software choices, as well as the preferred configurations, can be changed by modifying some variables in `build/conf/local.conf` (Listing 1).

BitBaking Software Packages and Images

The command that does everything in the correct order, parallelizing whenever

TABLE 2: yocto/poky Directory

Kilobytes	Directory	Contents
9,040	<code>bitbake</code>	The <code>bitbake</code> command and tools
19,688	<code>documentation</code>	Yocto documentation sources
29,908	<code>meta</code>	First meta-layer with base recipes
1,212	<code>meta-lxde*</code>	Added layer for building LXDE components
704	<code>meta-office*</code>	Added layer for building LibreOffice
55,512	<code>meta-openembedded</code>	Added dependency layer for Office and LXDE
316	<code>meta-poky</code>	Layer for the “Poky” distribution
2432	<code>meta-raspberrypi*</code>	Added layer for supporting Raspberry Pi-specific rules
28	<code>meta-raspi-knoppix*</code>	Added layer for my own (Knoppix) configuration
356	<code>meta-selftest</code>	Layer for built-in tests
168	<code>meta-skeleton</code>	Layer containing recipe samples
12	<code>meta-yocto</code>	Layer for architecture-independent recipes
232	<code>meta-yocto-bsp</code>	Layer handling board-specific rules
3,144	<code>scripts</code>	Supporting scripts
	<code>build</code>	Main working directory
28	<code>build/conf/*.conf</code>	Global configurations
7477,860	<code>build/sstate-cache</code>	Build process results as binary fragments and state information
23,371,760	<code>build/tmp</code>	Sysroot directories, build directories, images

possible and reporting errors in Yocto (and OpenEmbedded, on which Yocto is based), is BitBake, which calls a series of actions for each software recipe (files ending with `.bb`). BitBake

1. *fetches* sources and unpacks and verifies the checksum,
2. *configures* software packages,
3. *compiles* software packages in the desired architecture (some for the host,

some for the target system, as needed),

4. *installs* in a host or native sysroot directory, and
 5. creates software *packages* and installs them in the target image or directory.
- The `bitbake` command is mostly called with the desired recipe as a single argument, but it also has some options to control the build process or clean up

after a failed build, to avoid errors from leftover files. The standard command for building the Poky sample distribution would be

```
bitbake core-image-minimal
```

Of course, the real name of the target image is provided by the target distribution's build recipes.

LISTING 1: build/conf/local.conf

```
# Build for the Raspberry Pi 2 architecture (will also run on the Pi 3)
MACHINE ??= "raspberrypi2"

# Use the "poky" distro samples
DISTRO ?= "poky"

# Build software packages in Debian format
PACKAGE_CLASSES ?= "package_deb"

# Build 32bit host binaries, though I'm using a 64bit kernel
SDKMACHINE ?= "i686"

# Avoid "Host distribution has not been validated" warning
SANITY_TESTED_DISTROS_append = "debian"

# Allow proprietary add-ons
# LICENSE_FLAGS_WHITELIST = "commercial"

# Save LOTS of space during build by deleting temporary sources
INHERIT += "rm_work"

# Number of parallel threads (otherwise number of CPUs)
BB_NUMBER_THREADS = "8"

# want to add libreoffice. have already installed meta-office layer
DISTRO_FEATURES_append += "opengl"

# Enable UART also on Pi3
ENABLE_UART = "1"
```

LISTING 2: build/conf/bblayers.conf

```
# POKY_BBLAYERS_CONF_VERSION is increased each time build/conf/bblayers.conf
# changes incompatibly
POKY_BBLAYERS_CONF_VERSION = "2"

BBPATH = "${TOPDIR}"
BBFILES ?= ""

BBLAYERS ?= " \
    ${TOPDIR}/../meta \
    ${TOPDIR}/../meta-poky \
    ${TOPDIR}/../meta-yocto-bsp \
    ${TOPDIR}/../meta-raspberrypi \
    ${TOPDIR}/../meta-openembedded/meta-oe \
    ${TOPDIR}/../meta-openembedded/meta-python \
    ${TOPDIR}/../meta-openembedded/meta-gnome \
    ${TOPDIR}/../meta-lxde \
    ${TOPDIR}/../meta-office \
    ${TOPDIR}/../meta-raspi-knoppix \
"
```

A Simple Raspberry Pi "Hardware Up" Image

A good start for building Raspberry Pi images based on Yocto/Poky is the Board Support Package layer `meta-raspberrypi` from Yocto Project, which is available under the MIT license [4]. It features recipes to build a recent kernel, base system (SysVinit or systemd – your choice), and embedded Qt5 for building applications running directly on the framebuffer (which I will probably not use; Knoppix needs full X11 support to provide a common desktop). The simple original image is created by

```
bitbake rpi-hwup-image
```

which builds a kernel, bootloader, and root filesystem as an SD card image and needs about two hours for the first run. This SD card image successfully boots the Raspberry Pi 2 into a shell and even supports the serial port connection over GPIO pins.

More Layers

The *meta-office* layer that contains LibreOffice was fetched from GitHub [5] and *meta-lxde* from OpenEmbedded [6]. If space is tight on the disk used for building the distro, adding this line to `build/conf/local.conf` makes sure that unpacked sources and temporary files are being removed after successfully building a component:

```
INHERIT += "rm_work"
```

The current layer configuration for Knoppix for Raspberry Pi is set forth in `build/conf/bblayers.conf` shown in Listing 2.

The `meta-raspi-knoppix` directory contains all recipes for Knoppix-specific packages, custom kernel configurations and patches, and software choices. The contents of file `meta-raspi-knoppix/recipe-`

pes-core/images/rpi-knoppix-image.bb are shown in Listing 3.

The current feature list for Knoppix for Raspberry Pi includes:

- Cloop block compression. Unlike the Intel/AMD architecture DVD version of Knoppix, compression is not really needed for space reasons, because the usual Raspberry Pi SD card size of 8GB is plenty. However, if the system is compressed, physical reads are reduced to about a third of the usual volume, which may speed up read access and makes it easier to update the system in the form of compressed files containing all essential data except for the user's private files and configuration. Benchmarks for the choice of fastest decompression algorithm on the Raspberry Pi's CPU architecture need to show whether there is a substantial performance gain in this approach.
- AuFS [7] stacked filesystem. If system data is compressed in a read-only cloop container file, the user's changes need to be stored inside a writable partition mounted as overlay.
- ADRIANE audio desktop assisting blind computer users.
- Accelerated graphics support for Com-piz.
- LXDE, LibreOffice, browser(s), ...

AuFS and cloop have already been tested successfully as patches for the kernel,

and both are working fine on the architecture (although the gzip algorithm implementation seems much slower on ARM than on Intel/AMD).

The build process for the Knoppix image is started with:

```
bitbake rpi-knoppix-image
```

Because of the *libreoffice* package, the build process now takes half a day on my build machine, compared with the two hours of the initial "minimal" image.

If a recipe fails to build in the process, it might be necessary to remove the *sstate-cache* files for that component to force a "clean build" of all included binaries:

```
bitbake -c cleansstate libreoffice
```

One very useful debugging tool is the *devshell* subcommand of *bitbake*; for example:

```
bitbake -c devshell linux-raspberrypi
```

Instead of building a package, this command will change directory to the unpacked sources and then start an xterm terminal window with a shell – with all variables correctly set for the recipe – and allow a manual build, as in the traditional

```
./configure
make
...
```

to detect errors in the build process and allow manually changing files in the temporary build folders.

After all required recipes are completed in

```
bitbake rpi-knoppix-image
```

(which adds up to around 8,000 tasks that BitBake shows during the build process), an SD card image resides in `build/tmp/deploy/images/raspberrypi2/` that can be flashed to an SD card. For example, for an MMC card reader, use:

```
dd if=rpi-knoppix-image-raspberrypi2.rpi-  
sding of=/dev/mmcblk0 bs=1M
```

Booting the image on Raspberry Pis 2 and 3 worked perfectly for me, although I'm not yet starting the graphical environment and haven't changed the startup procedure to the Knoppix-specific "single shell script" boot. Of course, the Yocto-created image does not offer much of the software yet that you would get in a Raspbian installation, but it's comparably small and only contains software really needed for now.

I will continue writing about the next steps of creating Knoppix for Raspberry Pi in the next issue. ■■■

LISTING 3: Knoppix Image .bb File

```
# Base this image on the minimal "rpi-hwup-image" contained
# in the meta-raspberrypi layer
include rpi-hwup-image.bb

# Including SSH for remote access
IMAGE_FEATURES += "ssh-server-dropbear"

# Include dpkg and apt for software management
IMAGE_INSTALL_append += "dpkg apt"

# Add libreoffice
IMAGE_INSTALL_append += "libreoffice"

# Desktop & X11
IMAGE_INSTALL_append += "packagegroup-lxde-base"
IMAGE_INSTALL_append += "xserver-xorg"

# Bash is needed for many KNOPIX-specific scripts, make sure
# it gets installed
IMAGE_INSTALL_append += "bash"

# Shell-Dialog is needed for ADRIANE
IMAGE_INSTALL_append += "dialog"

# To be continued: elinks, speech-dispatcher, sbl, ...
```

INFO

- [1] OpenEmbedded: https://www.openembedded.org/wiki/Main_Page
- [2] Yocto: <https://www.yoctoproject.org/>
- [3] Host system packages: <http://www.yoctoproject.org/docs/2.3/mega-manual/mega-manual.html#required-packages-for-the-host-development-system>
- [4] meta-rpi: <http://git.yoctoproject.org/cgi.cgi/meta-raspberrypi/>
- [5] meta-office: <https://github.com/schnitzeltony/meta-office>
- [6] meta-lxde: <https://layers.openembedded.org/layerindex/branch/master/layer/meta-lxde/>
- [7] AuFS: <http://aufs.sourceforge.net/>



Repair damaged boot configurations

Start Assist

Sometimes things go wrong when you are installing an operating system on a hard disk drive or SSD. A boot repair disk gets your boot configuration back on its feet, quickly.

By Erik Bärwaldt

The boot process for computers has become massively more complicated in recent years. Unified Extensible Firmware Interface (UEFI) has largely replaced the traditional BIOS, while increasingly large storage devices require new types of partitioning.

The configuration options of bootloaders such as GRUB 2 have thus been massively extended; even minor changes to the system can cause start-up problems. In the worst case, you will be left sitting in front of a black screen with a flashing cursor without the operating system having booted.

In this situation Boot Repair Disk provides invaluable assistance: The operating system, based on the lean Ubuntu 14.04 LTS with the LXDE desktop takes care of damaged boot configurations even in heterogeneous environments, repairing them automatically at the push of a button.

Ready, Steady, Boot

Boot Repair Disk [1] is available as an ISO image of approximately 642MB for

64-bit architectures, or as 627MB 32-bit variant. Thus, you can burn both versions of the operating system on a CD, which you can use even on legacy hardware without a DVD drive. Alternatively, you can use UNetbootin [2] to transfer the image to a USB stick for use on computers without an optical drive. In our lab, we were unable to write a bootable image to a stick with the on-board tools.

After setting up the image, boot the computer from the corresponding media, and choose the bottom entry in the boot manager *Boot Repair Disk session*. Within a short time, the system starts and immediately launches the *Boot Repair* software on a very plain LXDE desktop before proceeding with a system scan. Then the program's control dialog appears (Figure 1).

Under normal circumstances, you will just want to press the large button labeled *Recommended repair* to initiate an automatic reconstruction of damaged system components such as the master boot record (MBR) and boot manager. If you first need accurate data on the mass storage media, but do not want to make

any modifications for the time being, then click instead on *Create a BootInfo summary*. In addition, the window also offers advanced configuration options, which you can access by clicking on *Advanced options*.

The window then expands to include a configuration dialog for the GRUB boot-loader (Figure 2), which groups various options in tab groups and grays any inaccessible tabs. In the first tab *Main options*, you can only configure a few basic settings for GRUB 2; you will see that the tool has already activated the option for reinstalling the bootloader. In addition, you can trigger an automated



Figure 1: The Boot Repair window is totally uncluttered.

Lead Image © Spacejunkie, photocase.com

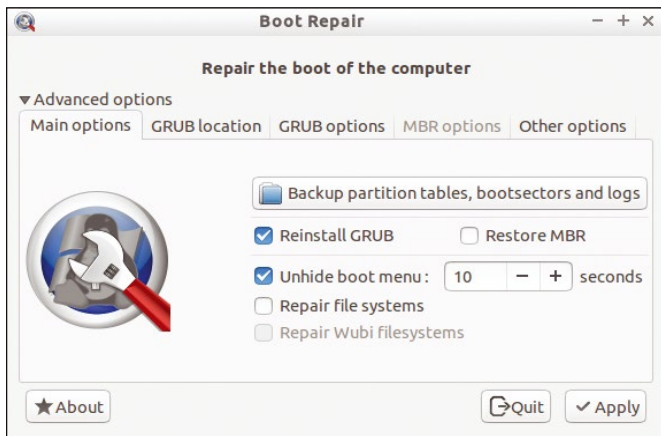


Figure 2: Numerous startup parameters can be set in the advanced options.

filesystem repair; Boot Repair gives you the option of reanimating a damaged MBR if necessary.

As an important additional option, the dialog offers the possibility to make a backup of the partition table, boot sector, and all logfiles so that you can reconstruct the old data later in case of problems. If you enable the *Restore MBR* option, Boot Repair grays the following tabs *GRUB location* and *GRUB options* and instead enables the *MBR options* dialog. In the second tab from the left, labeled *GRUB location*, you can define where GRUB 2 is installed. You can either select all mass storage media or a specific disk, which you choose in a selection box. You can also specify which operating system the bootloader should load as the default.

In the following tab, *GRUB options*, you can choose to completely delete an existing GRUB 2 installation before setting up GRUB again, or enable GRUB Legacy as the default boot manager. You can also configure various parameters that GRUB 2 needs to correctly start specific operating systems. If the configurations offered here do not meet your needs, you can press *Edit GRUB configuration file* to tune the configuration file manually to your liking.

In the *Other options* tab, you can define various options for logging the individual tasks. If you also have a Windows version on your computer, you can enable the *Repair Windows boot files* option to repair a Microsoft system that fails to launch. Then enable the respective options by clicking on the *Apply* button. If you want to repair a mass storage device's MBR, enable the *Restore*

MBR option in the *Main options* tab. Boot Repair then grays the settings dialogs for GRUB and instead enables the *MBR options* tab. You can then select which tool to use to reconstruct the MBR. If there are multiple partitions on the mass storage device, you can also define here which of

them to boot by default (Figure 3).

BootInfo

The Boot Repair Disk also comes with another program dubbed BootInfo, which helps you with problems at system startup time. It can be found in the System Tools menu of the operating system and provides a clear-cut window where you can define with a single mouse click whether the tool should store the boot log online or locally.

After another click on *Local report (text file)*, the

tool scans the computer and then opens the Leafpad text editor, which opens up with the scan log. You will not only find detailed information on the system configuration here, but – at the end of the log – also some hints on how the repair tool will approach the task. You can thus determine what modifications the tool will make on the computer (Figure 4).

The tool lists all the partition data of all mass media (including USB flash drives) connected to the computer system, as well as the GRUB configuration files. In addition to the repair program's log, you will also find the complete output from the parted -l, parted -lm, mount, df -Th, and fdisk -l commands in the text file. You are thus given a good overview of the mass storage device configuration.

OS Uninstaller

The third in-house developed tool included with Boot Repair Disk is found in

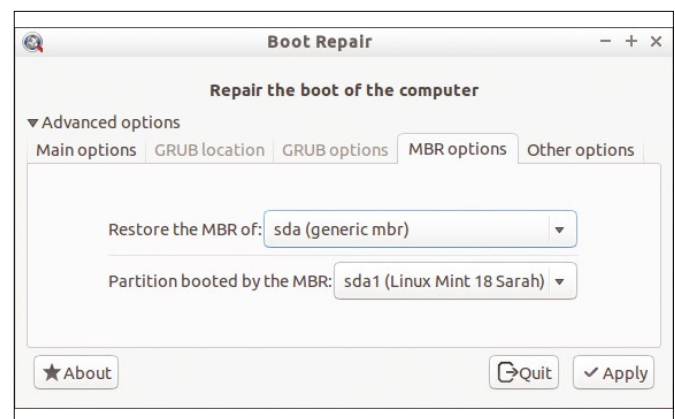


Figure 3: Boot Repair also includes repair options for the MBR.

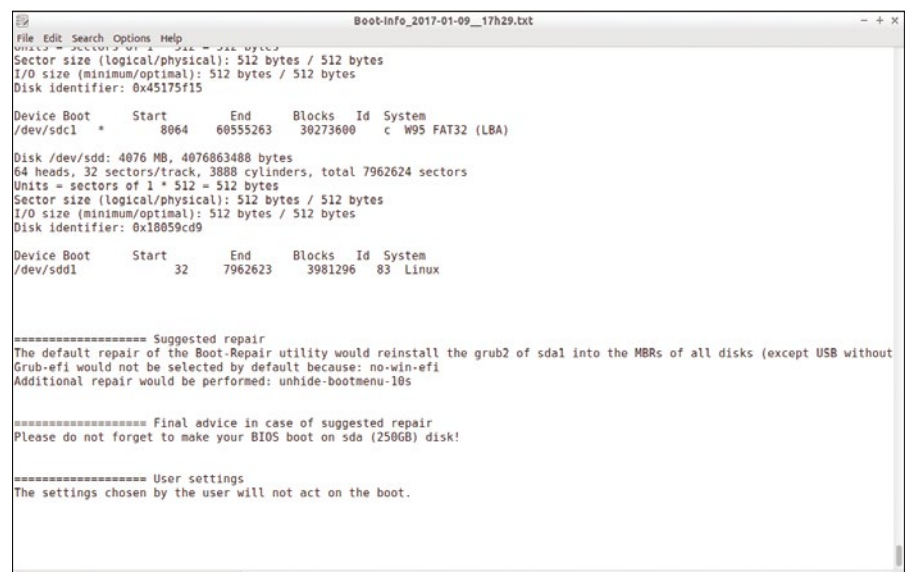


Figure 4: BootInfo impresses with extensive testing and logging options.

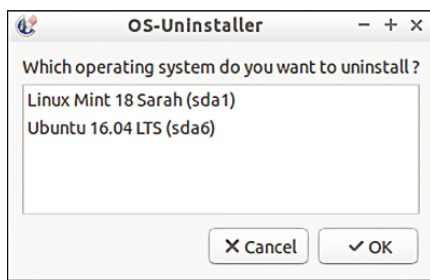


Figure 5: A mouse click selects the operating system that you want to delete.

the System Tools menu: OS Uninstaller. This helps you delete an operating system without leaving any remains on your mass storage device and without painstaking manual work.

After launching, the application first lists all the existing operating systems after a brief scan of the system (Figure 5). Select the operating system that you

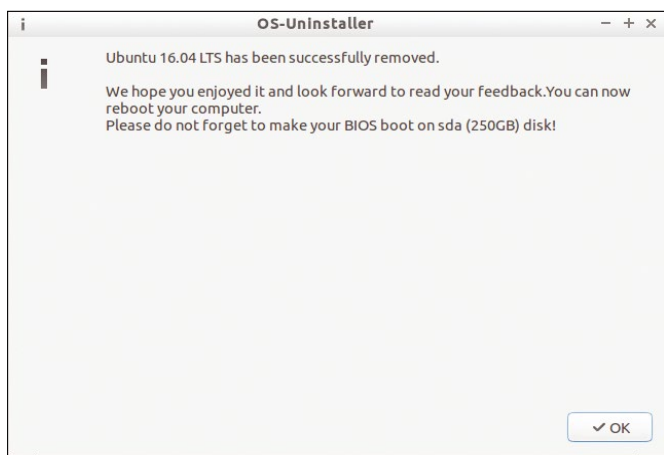


Figure 6: We successfully deleted Ubuntu in just a couple of minutes.

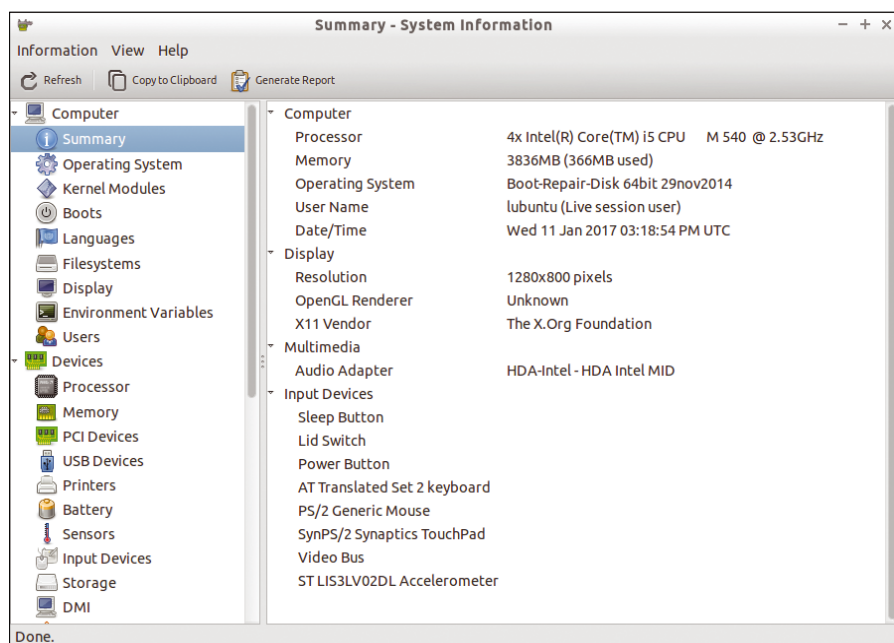


Figure 7: HardInfo displays the hardware and modules installed in the computer.

want to delete from this list and press *Apply*. After a safety prompt, the uninstaller first deletes the operating system, then reconfigures the boot manager, and finally displays the results (Figure 6).

After a reboot, you will find the remaining operating systems in the GRUB startup menu, where the OS Uninstaller adds entries for the Plopp Boot Manager and Smart Boot Manager. They do not have a function without additional configuration, so you can safely remove them from the Start menu.

More Tools

In the event that hardware problems cause difficulties when booting a computer, Boot Repair Disk comes with two graphical diagnostic tools in the form of HardInfo and GParted. HardInfo, which you will find under the System Tools

menu labeled *System Profiler and Benchmark*, clearly visualizes the hardware in a two-pane window and also performs benchmarks. GParted, on the other hand, helps to manage the storage devices. This is where you can, for example, identify problems arising from in-

correct formatting or damaged filesystems (Figure 7).

Desktop

Besides the tools for repairing the system, you will only find a few preinstalled applications. Office applications, games, multimedia, and educational applications are missing completely. The Accessories menu contains entries for the LX terminal, the Leafpad text editor, and the PCManFM file manager.

You can launch Firefox from the Internet menu, and the usual LXDE configuration dialogs are found below the Preferences menu. As a special feature, the System Tools menu offers the Synaptic package manager, which provides access to the Ubuntu repositories, if you need additional software.

Hands-On

In our practical tests, the system was totally convincing. It not only successfully restored damaged boot sectors in Linux-only installations, but also repaired a mixed system with one Windows and two Linux partitions.

Also the OS Uninstaller enormously simplifies administrative tasks: In our lab, it always reliably completed the desired tasks on multiple machines with a variety of shared storage devices, thus removing the need for time-consuming manual deletion and modification of the partition table, as well as the GRUB configuration files.

Conclusions

Boot Repair Disk is undoubtedly a very useful tool that every administrator in a heterogeneous environment should include in their toolbox. Even less experienced users will be able to quickly control the intuitive tools without any problems. The system itself is extremely stable, and the special tools for revitalizing the MBR and the GRUB 2 boot manager were impressive. Also the deletion tool for unneeded operating system installations saves a huge amount of manual configuration work. ■■■

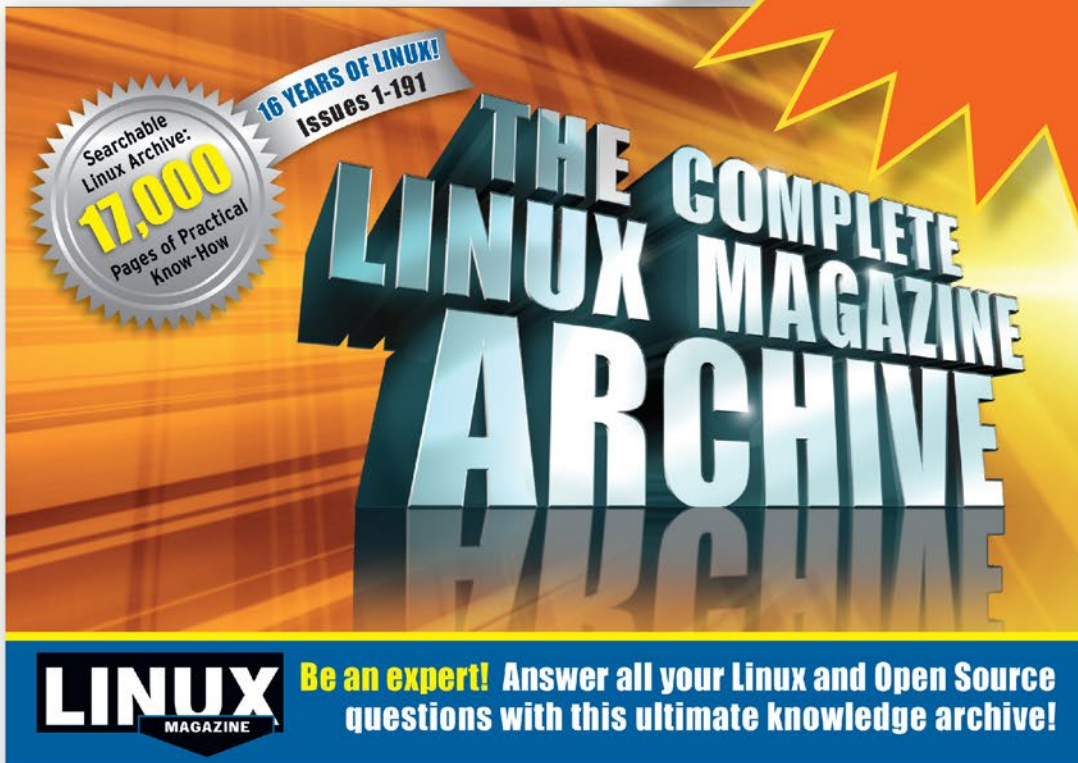
INFO

- [1] Boot Repair Disk: <https://sourceforge.net/projects/boot-repair-cd/files/>
- [2] UNetbootin: <https://unetbootin.github.io/>

Happy 25th Anniversary to Linux!

Help us celebrate
25 years of Linux
with the All-Time
Archive DVD of
Linux Magazine!

Order today and
get 191 issues of
Linux Magazine on
one handy DVD!



You get 17,000 pages of practical know-how on one searchable disc - that's 191 issues including 40+ issues that were not included in the previous release.

Order Now! Shop.linuxnewmedia.com

A Python script warns of failed login attempts

THE BOUNCER

A number of sensors and cameras send author Mike Schilli a short message if someone tampers with his apartment door. He has now applied this security principle to the SSH entrance of his Linux computer. *By Mike Schilli*

As an alternative to the Prowl solution for sending text messages described in a previous article [1], another provider in the colorful world of phone apps, Pushover, now – for a one-off payment of \$5 – lets you distribute 7,500 messages a month for the rest of your life through a web API to either iOS, Android, or desktop clients.

Rough and Ready Browser

On iOS or Android, the user logs in to the Pushover app, which then displays incoming messages as push notifications (Figure 1), even if the phone isn't being used and displays the lock screen. Additionally, Pushover offers native desktop clients for the Mac and a somewhat hacky browser solution for the Linux desktop.

To install the desktop client in Chrome or Firefox, you

go to the Pushover login page [2], enter your email address and password for your Pushover account, and then allow the browser to output notifications on your desktop. This only works while the browser is running and one tab is pointing to the Pushover website (Figure 2). Because I already do this for Gmail and Evernote on my home system, with the help of pinned tabs, an extra tab does not really matter.

Tracking

Following the latest entries in a logfile like `auth.log` in `/var/logs` is not as easy as you might think. Even implementing a Unix function like `tail -f` (Figure 3), which every admin is likely to use several times a day,

requires knowledge of the system `seek()` function, which you can use to advance the read cursor associated with a file handle to the end of a file.

If `read()` fails to return any data, you have most likely reached the end of the

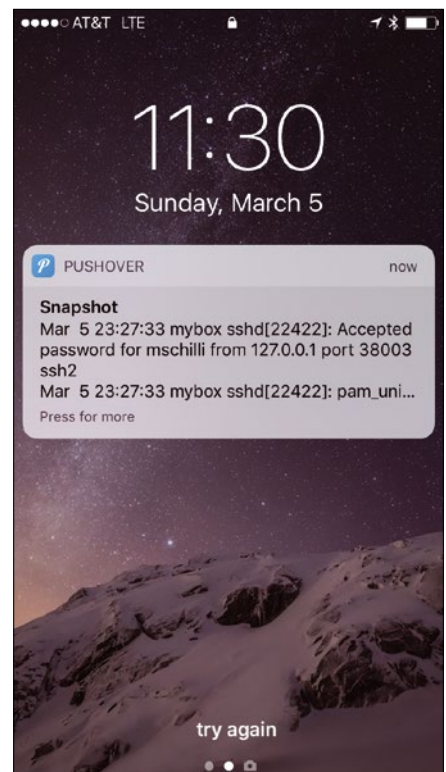


Figure 1: The cell phone with the installed Pushover app shows a login attempt on the SSH server of the monitored Linux computer.

MIKE SCHILLI

Mike Schilli works as a software engineer in the San Francisco Bay area of California. In his column, launched back in 1997, he focuses on short projects in Perl and various other languages. You can contact Mike at mschilli@perlmeister.com.



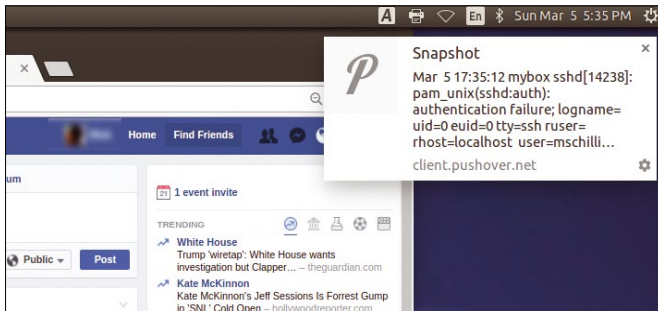


Figure 2: A browser tab for the Pushover service enables pop-ups on the Linux desktop.

file. But if additional lines do appear afterward, appended by another process in the meantime, then `tail -f` outputs the content. Even if the admin renames the file, the data-consuming process and the open file handle remain the same, and the reading program won't even notice.

But even `tail -f` can trip up if the distribution's logfile rotator steps in to shift the old file out of the way, then compresses the file and replaces it with a fresh, empty file. In this case, it would be fatal to keep tracking the open file handle with `read()`, because the fresh data would now undoubtedly be written to a completely different file.

Processes trailing logs can cater for this by periodically checking whether the file under the specified name still uses the same inode on the filesystem. If `stat()` shows that the inode has changed, the log analyzer needs to close the open file handle and open a new one on the new file with the same name.

Prefabricated

Luckily, no one actually needs to convert this logic into program code nowadays

because several open source implementations already do the job perfectly. Python has `pygtail` [3], for example, which is supposedly a Python port of the widespread `logcheck` utility. If you do not want to write your own Python program and you can do without customizing the code to your needs, you can also use `logcheck` directly to parse the system logfile as the first leg in the alarm pipeline.

To install the `Pygtail` module for Python 3.x, use:

```
pip3 install pygtail
```

Listing 1 [4] imports the `Pygtail` module, and line 9 composes a path for the flag file required by `Pygtail` in the `data` directory below the user's home directory; in the case at hand, this is `data/authwatch.auth.log.offset`. This is where `Pygtail` stores the byte count of how far it got in the file; the next call carries on reading

LISTING 1: authwatch

```
01 #!/usr/bin/python3
02 import sys
03 import os
04 import re
05 from pygtail import Pygtail
06
07 log_file = '/var/log/auth.log'
08
09 offset_file = os.path.join(os.getenv("HOME"), "data",
10 os.path.basename(sys.argv[0]) + "." +
11 os.path.basename(log_file) +
12 ".offset" )
13
14 for line in Pygtail(log_file, offset_file=offset_file):
15     if not re.search('CRON',line) and \
16         not re.search('Connection closed',line):
17         sys.stdout.write(line)
```

behind the offset. It will also output any new data, if available, or otherwise keep quiet.

Cron later calls the script at five-minute intervals and immediately says goodbye after its work is done, so it needs this persistent position marker in the offset file. The admin only has to create the `~/data` directory once manually before using the script if it does not already exist.

Listing 1 also filters out regular events, such as entries in which keywords like `CRON` or `Connection closed` occur. Lines 15 and 16 use regular expressions, courtesy of the imported standard module `re`, to search for these entries.

If you have a flavor of Linux that relies on the much maligned `systemd`, you will not find an `auth.log` file, but you can use

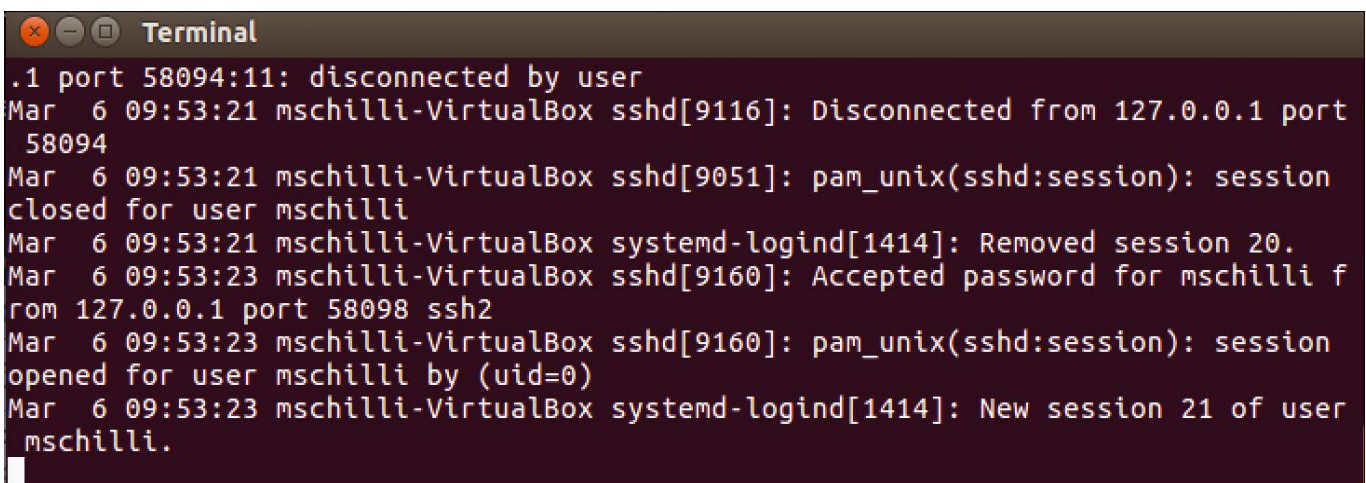


Figure 3: Running `tail -f` against the `/var/log/auth.log` file shows successful and failed login attempts against the SSH daemon.

LISTING 2: pushover

```

01 #!/usr/bin/python3
02 import requests
03 import sys
04 import re
05
06 string = sys.stdin.read()[:1024]
07
08 if re.search('\S', string):
09     r = requests.post(
10         'https://api.pushover.net/1/messages.json',
11         data = {
12             'token': '<XXXXXXXXXXXXXXXX>',
13             'user': '<YYYYYYYYYYYYYYYY>',
14             'message': string
15     })

```

journalctl at the command line or the systemd Python bindings and their journal method to find the newest entries in the system log.

Instead of an offset into a file, the script then stores the timestamp of the last query in an extra file and jumps just beyond it for a call to seek_realtime(), to avoid reporting duplicates. In this case, the script does not need to worry about rotated logfiles because systemd abstracts such implementation details.

Convenient Web Requests

The second part of the alarm pipeline is shown in Listing 2; the REST request sent to the Pushover API server requires two tokens (lines 12 and 13), which users receive when registering with the Pushover service. The trial period lasts for four weeks and allows free access; if you like what you see, you can then buy a license for \$5,

which covers a single platform: iOS, Android, or the desktop client.

Registered users will find the token assigned to the user key in the Pushover dashboard overview in Figure 4. The token labeled token, on the other hand, identifies the app (Figure 5) to the Pushover service; in this case, it is the Python script in Listing 2, which I registered with Pushover under the name Snapshot.

For a successful REST request with Python's requests library, you also need the message parameter with the message text for the post() method in line 9 to contact Pushover. In typical Python style, the library throws exceptions that abort the program in case of errors if they're not processed and outputs a stack trace that will hopefully help you solve the problem.

The Python requests library keeps its promise of being a self-proclaimed "HTTP for Humans." At least it is more carefully thought out than urllib and urllib2, which you might remember I have complained about before.

Line 6 in Listing 2 retrieves the text sent from the first part of the pipeline from standard input and truncates it with the usual syntax for array slices in Python – [:1024] – to the maximum length of 1,024 characters allowed by Pushover. The if construct in lines 8-15 then uses the regular expression \S to check whether the message contains printable characters and terminates empty runs without further ado.

Friend Cron

All you have to do now is set up a cron job that calls the pipeline about every five minutes and finds the user's home directory and the writable data directory therein for storing the flag file:

```

*/5 * * * * /path/authwatch | &
/path/pushover

```

The Pushover service distributes any messages sent to all devices registered by the user, so it may happen that a failed login attempt triggers a whole explosion of notifications if several mobile devices in the room are actively connected. At least this only counts as one message, though, in the usage constraints, of which users are allowed 7,500 per month on the all-inclusive tier. ■■■

Push a Notification
To send a notification to one or all of your devices, enter a message below. To send notifications programmatically, check out our API.

Send As: Pushover (default)
Device: All active devices
Sound: (Device default sound)
Title: optional
Message:
URL: optional
Send Notification

Your User Key
To receive notifications from a Pushover-powered application, service, or website, just supply your user key:
u: [redacted] pd
To receive Pushover notifications from e-mails, send to: [redacted]@pomain.net

Your Quiet Hours (Edit)
You do not have any enabled quiet hours.

Your Devices (Add Your Mobile Device or Desktop) (View Your Licenses)

Name	Status	Last Synced	Messages Received/Pending
chrome	Trial Period (Upgrade Now) (Login)	2 minutes ago	7 received, 0 pending
iphone	Trial Period (Upgrade Now)	2 minutes ago	14 received, 0 pending

Figure 4: Registered users can see their user tokens in the dashboard.

Pushover
Latest Pushover News: Pushing data directly to a complication on your Apple Watch posted on November 04, 2016

Snapshot (Application) [Back to Apps](#)

API Token/Key (Edit or Delete Application)
To begin using our API to send notifications, use this application's API token:
a91h [redacted] f4

Bar chart showing messages sent over time:
Free Messages Sent (blue bars), Paid Messages Sent (red bars).
X-axis: Jan 1, Jan 8, Jan 16, Jan 24, Feb 1, Feb 8, Feb 15, Feb 22, Mar 1.
Y-axis: 0 to 8.

Figure 5: Newly registered user apps receive a token.

INFO

- [1] "WiFi Connect Messages" by Mike Schilli, *Linux Pro Magazine*, issue 186, May 2016, p. 64, <http://www.linux-magazine.com/Issues/2016/186/Perl-WiFi-Connect-Messages>
- [2] Pushover login: <https://client.pushover.net>
- [3] Pygtail: <https://github.com/bgreenlee/pygtail>
- [4] Listings for this article: <ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/200>

LOST YOUR BOOKSTORE?

LET US BE YOUR BOOKSTORE



Browse our shop for single issues of *ADMIN*, *Linux Pro*, *Linux Magazine*, *Ubuntu User*, *Drupal Watchdog*, and *Raspberry Pi Geek* – delivered right to your door.

■ shop.linuxnewmedia.com/single

Better yet, subscribe, and you won't need a bookstore.

■ shop.linuxnewmedia.com/subs



shop.linuxnewmedia.com

DIGITAL AND PRINT EDITIONS AVAILABLE!



The sysadmin's daily grind: Pi-hole

The Hole Truth

A strange rule seems to dictate that the most useless products and services have the most annoying online advertising. Columnist Charly blocks the garish advertising for all computers on his network centrally with the Pi-hole tool, which is not only for Raspberry Pi devices. *By Charly Kühnast*

There are two irreconcilable camps in the discussion on the use of banners and skyscrapers on websites: One is populated by people who get annoyed by garish, flashing, fidgety advertising formats that remind them of neon signs from the 50s. An increasing number of these users simply reject advertising on the web as garbage. The opposing camp is occupied by website owners – amateur bloggers, to name just one example – for whom advertising is the only way to recoup their costs for servers and other things.

People who place ads on their websites usually source them from one of several large commercial networks and simply create placeholders on the sites, which are then later replaced with the ads. Most people do not know exactly what advertising their site is showing at any given time.

The ad networks, in turn, allow the ad creators a great amount of freedom. It is no longer only images that are used here, but also JavaScript and the like. Criminals exploit this to display manipulated advertisements that scan the visitor's browser for vulnerabilities and – if they find any –

install malicious software or animate the user to download applications of dubious repute. It can thus happen that visiting a highly reputable website actually infects your own PC with malware.

Those who are aware of this “malvertising” – a word composed from malware and advertising – or are simply annoyed by the visual overkill can turn to an ad blocker in the form of a plugin for their browser. But because I have many computers, I need a centralized, easy-to-maintain instance that solves the problem. It seems to me that Pi-hole [1] is extremely useful for this task. The tool got its name from the company that originally developed it for use on a Raspberry Pi, but it has long since been adapted for deployment on most standard Linux distributions.

Pi-hole is underpinned by the lean Dnsmasq DNS server with a special configuration. I entered Pi-hole as the DNS server on all my clients, and it now filters out the undesirable requests by the clients to ad networks and submits the remaining DNS requests to the regular DNS server.

Easy Install

The easiest way to install Pi-hole is with the following command:

```
curl -sSL https://install.pi-hole.net | bash
```

Security-conscious admins might go into meltdown at the sight of this line, but the makers of Pi-hole have a way of calming them down. Of course, anyone can download the code, inspect it at their leisure, and then proceed with the install. Corresponding links and instructions can also be found online [1]. When done, the installer displays a randomly generated password for the web interface. You can access it on `http://<IP address>/admin`.

The web interface is visually appealing and offers a wealth of statistics (see Figure 1). You also can maintain your own blacklists and whitelists there. I make good use of this option, because I do not oppose advertising on the web as a matter of principle; I thus specifically add sites that I would like to support to the white list. In return, I punish sites that are badly behaved – because they install poster-sized pop-overs, for example – with a blacklist entry that filters their ads directly into a black hole.

Incidentally, there is no advertising at all on *pi-hole.net*. The project is free, and the code is open source. The authors simply ask you to donate an amount of your choosing. It would be nice if many people complied. ■■■

CHARLY KÜHNAST

Charly Kühnast manages Unix systems in the data center in the Lower Rhine region of Germany. His responsibilities include ensuring the security and availability of firewalls and the DMZ.

INFO

[1] Pi-hole: <https://pi-hole.net>

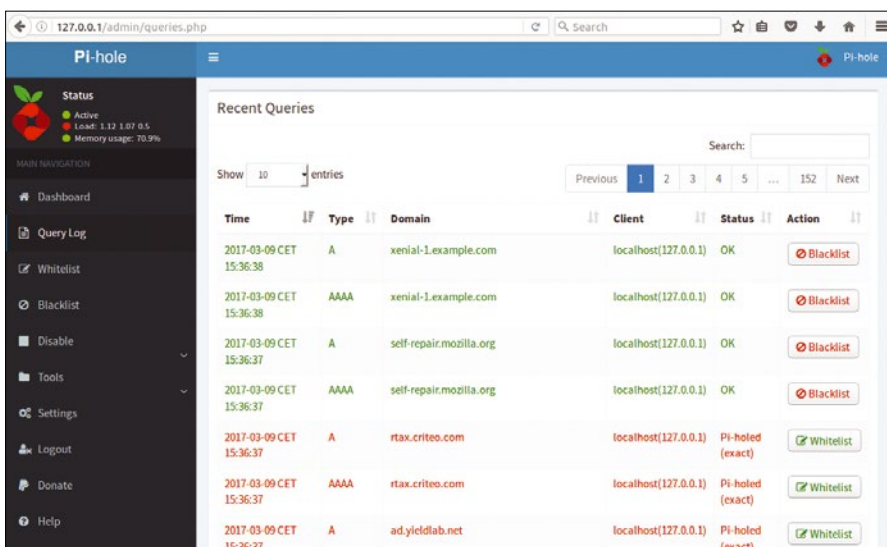
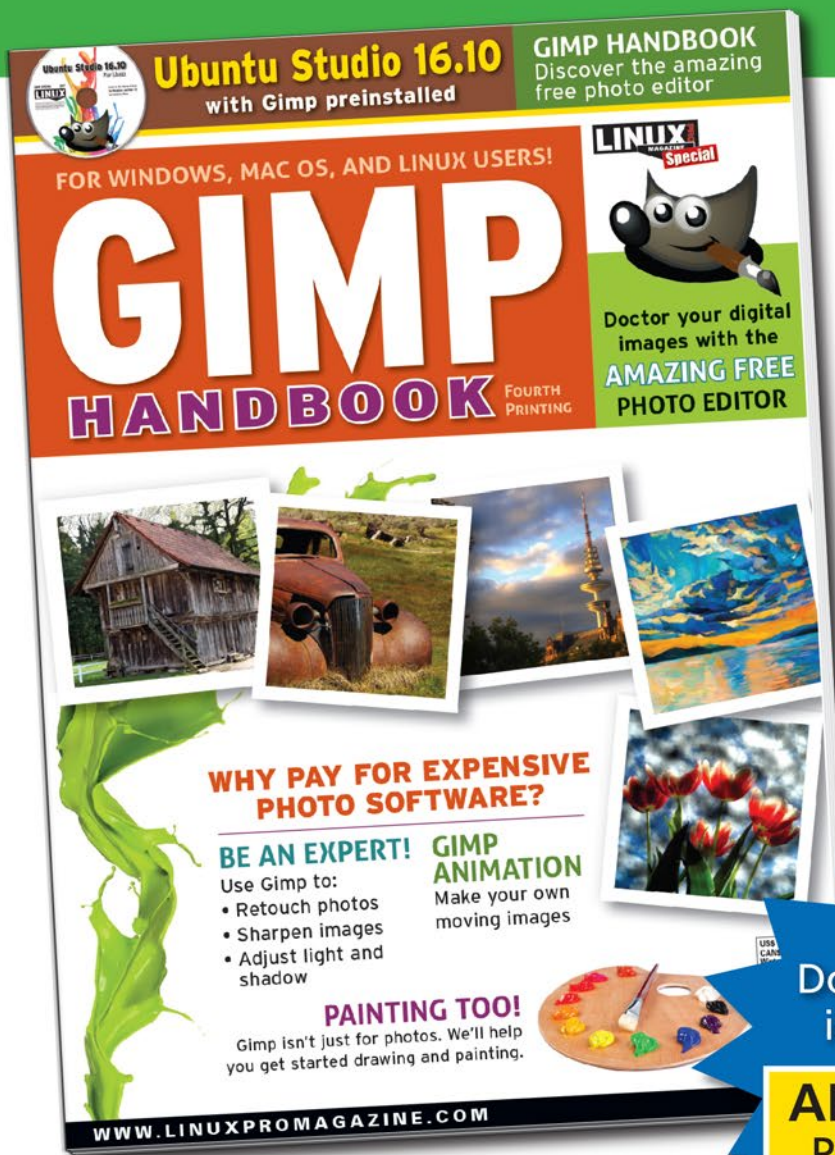
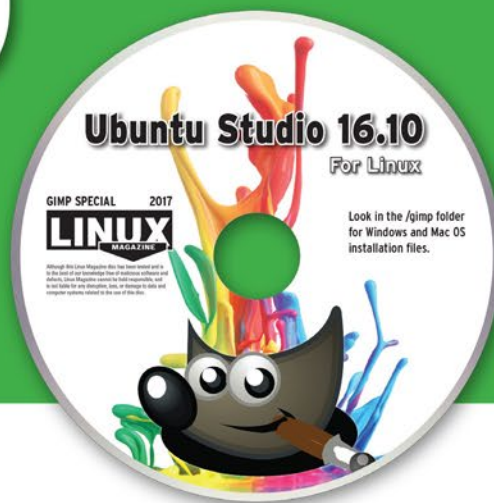


Figure 1: The Pi-hole UI, which is appealing both visually and in terms of content, presents various statistics and lists.

Shop the Shop

shop.linuxnewmedia.com

GIMP HANDBOOK



**SURE YOU
KNOW LINUX...**
but do you know **Gimp?**

- Fix your digital photos
- Create animations
- Build posters, signs, and logos

Order now and become an expert in one of the most important and practical open source tools!

Gimp
Doctor your digital images with the
AMAZING FREE PHOTO EDITOR!

Order online:
shop.linuxnewmedia.com/specials



FOR WINDOWS, MAC OS, AND LINUX USERS!



Secure your logins with two-factor authentication

LOCKDOWN

Add an extra layer of protection with one-time passwords.

By Mayank Sharma

There really is no such thing as too much security. Thanks to the availability of easy-to-use password cracking tools and multicore processors, it is only a matter of time before a determined hacker breaks your

password and gains access to your computer. If you're really concerned about unauthorized access to your computer, you should definitely add an additional layer of authentication (see the "What Is Two-Factor Authentication?" box).

One of the easiest mechanisms for implementing a two-step verification is the Google Authenticator service. You can use the service to issue a time-based authentication token to supplement the existing password challenge. Google Authenticator

uses the IETF-approved Open Authentication Time-Based One-Time Password (OATH-TOTP) protocol that generates a password (usually a six-digit number) that's meant for a single use before it expires and is replaced by a new number.

The good thing about the TOTP protocol [3] is that it doesn't require an active connection to the Internet. The protocol uses the current timestamp along with a pre-shared key that's available on both the machine you're trying to log in to, as well as on your portable app used to create the six digit code. When you type in the code at the login prompt in the computer, it performs the same computation

WHAT IS TWO-FACTOR AUTHENTICATION?

Broadly speaking, there are three different types of authentication factors. The most common is the use of a password or sometimes a security question to validate a user. Two-factor authentication is the use of another factor to verify the authenticity of the user who wants to gain access. Think of it as an extra layer of security. In addition to something the user *knows*, like the password, the user is prompted for something they *have*, like a security token. The use of passwords together with security tokens is known as two-factor authentication. Installations that can't afford security breaches use a third factor for authentication: Biometrics authentication is something you *are*, like your voice or fingerprint.

The most commonly used two-factor authentication mechanism is the use of passwords along with security token. The authentication tokens can be in the form of physical devices, such as smart cards, or can be doled out by software, such as mobile apps that generate PIN codes for authentication. Google Authenticator [1] (Figure 1) is an example of the latter. Note that while previous versions of the app were open source, the current releases now are all proprietary. For an open source option, check out FreeOTP [2] (Figure 2).

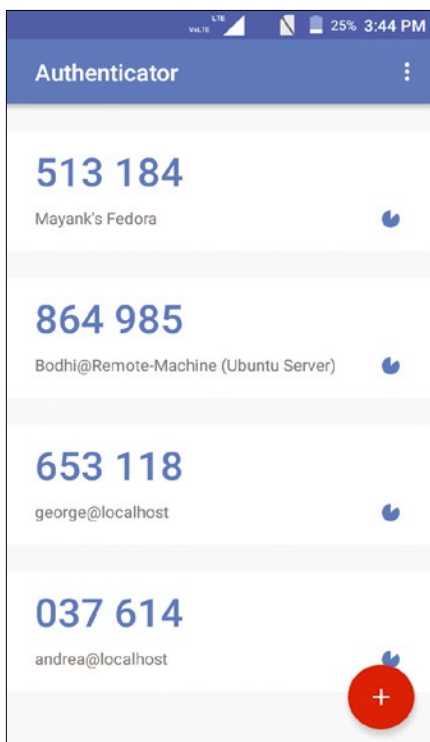


Figure 1: For a security app, it's rather amusing that Google Authenticator doesn't have PIN protection to lock the app.

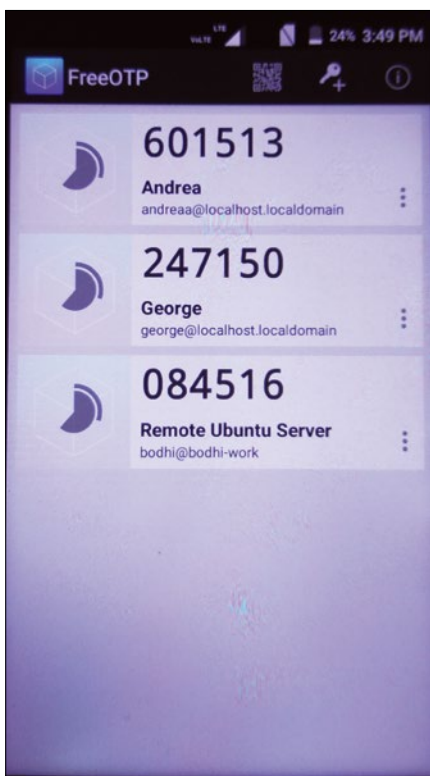


Figure 2: If you're weary of proprietary apps, you can use the open sourced FreeOTP app.

Lead Image © Kran Kanthawong, 123RF.com

with the same pre-shared key and arrives at the same six-digit code, thus allowing you to log in.

The TOTP-based authentication mechanism integrates into the Linux Pluggable Authentication Module (PAM) system. PAM is the most popular authentication infrastructure used on Linux systems to authenticate a user. Once you've integrated the service with your distribution's login manager, in addition to your user password, you'll be prompted for the tokens before being allowed to log in. The tokens are only valid for a limited duration, which effectively means you can't reuse them. So even if an attacker manages to shoulder-surf a token, it'll expire soon and can't be used to gain access after a few seconds. The Google Authenticator app (Figure 1) will generate these OTPs on your Android device once it's been configured to do so for every user on your Linux machine.

Configure Google Authenticator

To implement this multifactor authentication with Google Authenticator, you'll first need the Google Authenticator PAM module. The PAM mechanism allows for plugging different forms of authentication into a Linux computer.

The Google Authenticator PAM module is available in the official software repositories of major distributions like Ubuntu and Fedora. To install the package on Ubuntu, head to the terminal and type:

```
sudo apt-get install \
libpam-google-authenticator
```

If you are Fedora user, switch to the root user and type

```
dnf install google-authenticator
```

to install the PAM module for Google Authenticator.

Once the package has been installed, make sure you're logged in as the user you want to protect with the two-factor authentication. Now irrespective of the distribution, in the terminal window, type `google-authenticator`. This will initiate the process of creating a secret key for the user by asking you a bunch of questions. While it's safe to answer yes to all of them, it's a good idea to understand each question before making your

final choice as these choices help balance security with ease-of-use.

The first question is a pretty safe one, and you should allow the command to update your Google Authenticator file by answering yes. As soon as you answer in the affirmative, you'll be shown a secret key and several emergency scratch codes. Make sure you note down these emergency scratch codes somewhere safe. They'll help you log in if you misplace the Android phone that generates the OTPs. Also remember that each scratch code can only be used once. The `google-authenticator` command will also generate a QR code that you can scan with the Google Authenticator app in your Android phone. Because we haven't installed the app yet, for the time being just note down the 16-digit code, which we'll use to manually add the account on the app.

You'll then be asked if you'd like to disallow multiple uses of the same authentication token. Although it might seem in-

convenient at first, it is a good idea to restrict the use of a token, which forces you to wait 30 seconds between logins. You should agree to this limitation for maximum protection. The next question asks for permission to increase the time window for which tokens can be used from the default 1.5 minutes to four minutes. Although you can answer yes to this question to avoid any issues, type no for maximum security. To give you an idea, if you answer yes here, Google Authenticator will generate up to eight valid codes in a four minute window. By declining to increase the time window, you limit this to only three valid codes in a 1.5 minute window. In any case, if you notice any issues later on, refer to the "What's in the Hidden File?" box to increase the expiration time.

The fourth and the last question asks you to limit the number of attempts for entering the authentication code. You should definitely enable this option as it helps prevent brute-force login attacks.

WHAT'S IN THE HIDDEN FILE?

When you run the `google-authenticator` helper script (Figure 3), it generates a hidden file under the user's home directory (`~/.google-authenticator`). This file contains all the configuration information you need to replicate and modify Google Authenticator's behavior. You can open the file in any text editor, and it contains the information in the following order:

```
<secret key>
<options>
<recovery codes>
```

Because it contains the 16-digit key and the scratch keys, it's a good idea to back up this file to a trusted location. To change your Google Authenticator settings, you can either rerun the `google-authenticator` helper script or simply edit the `~/.google-authenticator` file. Options that are set in this file have a line in the options section. If you answered "no" to any particular option during

the initial setup, the corresponding line is excluded from the file.

For example, to extend the code expiration window to four minutes, add the line `WINDOW_SIZE 17`. To change the threshold of rate limiting, find the line `RATE_LIMIT 3 30` and adjust the numbers. The `3` indicates the number of attempts over a period of time, and the `30` indicates the period of time in seconds. Although it isn't prudent, you can also disable the use of recovery codes by removing the five eight-digit scratch codes at bottom of the file.

```
bodhi@localhost:~$ google-authenticator
Your new secret key is: 847X055F5Q5UH0H6
Your verification code is 801877
Your emergency scratch codes are:
79435201
18456743
61636058
78877311
42566006
Do you want me to update your "/home/bodhi/.google-authenticator" file (y/n) y
Do you want to disallow multiple uses of the same authentication
token? This restricts you to one login about every 30s, but it increases
your chances to notice or even prevent man-in-the-middle attacks (y/n) y
By default, tokens are good for 30 seconds and in order to compensate for
possible time-skew between the client and the server, we allow an extra
token before and after the current time. If you experience problems with poor
time synchronization, you can increase the window from its default
size of 1:30min to about 4min. Do you want to do so (y/n) n
If the computer that you are logging into isn't hardened against brute-force
login attempts, you can enable rate-limiting for the authentication module.
By default, this limits attackers to no more than 3 login attempts every 30s.
Do you want to enable rate-limiting (y/n) y
bodhi@localhost ~$
```

Figure 3: The `google-authenticator` helper script has a brief configuration process with verbose steps that are easy to comprehend.

Once enabled, the Google Authenticator PAM module will force the attacker to wait for 30 seconds after he's entered a wrong code thrice.

Now repeat this process for each user account that uses your computer. Ask everyone you share the computer with to log into their account and then run `google-authenticator` and make a note of their respective emergency scratch codes and the 16-digit code.

Plug In the Module

After you've generated the authentication code for all users, it's time to configure the login process to work with Google Authenticator. All you need to do is edit one file to add two-step authentication for all login attempts. The file you need to edit varies from one distribution to another. For example, if you are using Fedora, to prompt users for an authentication token before they are allowed to login into their desktop, fire up the terminal and type:

```
$ su -
# nano /etc/pam.d/gdm-password
```

The file will have several lines with the top few beginning with the word `auth` (Figure 4). In a new line below the last `auth` line, enter the following:

```
# To use OTPs generated by Google
Authenticator
auth required
pam_google_authenticator.so nullok
```

The `nullok` bit at the end of the line asks the distribution to let a user login even if they haven't run the `google-authenticat-`

Figure 4: If you add the call to the `pam_google_authenticator` module to the top of the relevant `/etc/pam.d/gdm-password` file, the distribution will ask for the OTP before prompting for your account password.

tor command to set up two-factor authentication. Say you have two users, andrea and george, but have set up Google Authentication only for andrea. Thanks to `nullok`, while andrea will have to enter the OTP, george will be able to log in with just his password. Note however that while this is a useful flexibility to have while you're testing Google Authenticator, once everything works smoothly and you have no issues logging in with the two-factor authentication, it's advisable to force all users to log in through Google Authenticator only by removing `nullok` from the end of this command.

In Ubuntu, however, you can add this line to the `common-auth` file, which is included in various other PAM configuration files. Open the file in a text editor with `sudo nano /etc/pam.d/common-auth`. Now scroll to the end of the file and add the above given line along with the comment. By adding the Google Authenticator PAM module to the `common-auth` file, we've asked Ubuntu to use the one-time passwords for all login attempts, including calls to `sudo`.

Set Up the App

Your Linux distribution is now all set up for two-factor authentication. To receive the OTPs, you'll now have to install the Google Authenticator app on your Android mobile phone from Google Play. After installing the app, you'll have to add an account for all the users you've run the `google-authenticator` command for on your Linux installation.

As soon as you launch the app on your mobile device, it'll take you through a small tour advertising its use. When

you've run through the tour (Figure 5), you'll be presented with a screen to add an account for a user you've already configured to use Google Authenticator in your Linux distribution. The app lets you add an account either by scanning a QR code or by manually adding the unique 16-digit key.

While scanning the QR code is the more convenient of the two options, we'll manually enter the code by tapping on the *Enter a provided key* option. In the following screen, first specify the name of the user for which this is meant, such as Mayank and then enter the 16-digit key you had made a note of earlier.

If you've keyed it in properly, the phone will vibrate to confirm, and the app will display a six digit code. Below this code will be the name of user that this code will help authenticate along with an animated timer that counts down to 30 seconds after which the app displays a new code.

If you've followed our advice and run the `google-authenticator` helper script for more than one user, you'll have to bring them to the attention of the Android app to generate OTPs for these other accounts as well. For this, first tap the + icon followed by the *Enter a provided key* and then repeat the process described above to first add a name to identify the user followed by their 16-digit secret key.

That's it, you've now successfully set up two-factor authentication for your Linux installation. The Android app will keep generating a new six-digit code

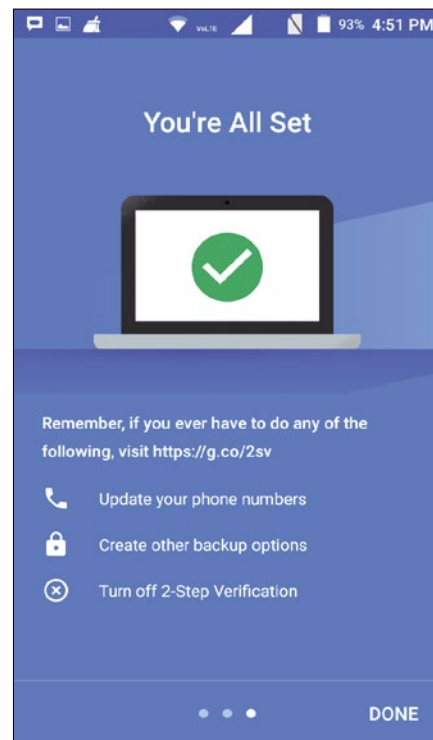
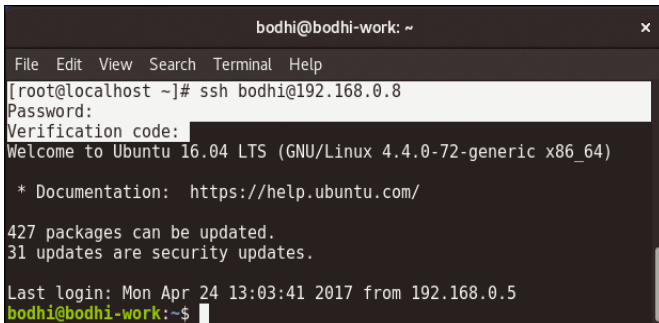


Figure 5: You can also enable two-factor authentication for your online Google accounts.

TROUBLESHOOT THE AUTHENTICATOR

If you can't access the the Android device that generates OTPs, you can use one of the five scratch codes to log into the machine. Note that these recovery codes are for one-time use and cannot be reused. If you've changed phones, install the Google Authenticator app on the new mobile and simply use the secret key in the `~/.google-authenticator` file to generate OTPs.

If, however, you've misplaced the Android device that doles out the OTPs, you should immediately generate a new secret key. To do this, once you've logged in using the scratch codes, first delete the existing `~/.google-authenticator` file and then rerun the `google-authenticator` helper script to generate a new `~/.google-authenticator` file. Now install the Google Authenticator app on a new mobile and add the newly generated secret key.



```

bodhi@bodhi-work: ~
File Edit View Search Terminal Help
[root@localhost ~]# ssh bodhi@192.168.0.8
Password:
Verification code:
Welcome to Ubuntu 16.04 LTS (GNU/Linux 4.4.0-72-generic x86_64)

* Documentation:  https://help.ubuntu.com/

427 packages can be updated.
31 updates are security updates.

Last login: Mon Apr 24 13:03:41 2017 from 192.168.0.5
bodhi@bodhi-work:~$

```

Figure 6: Even if you don't protect your local logins, it's still a good idea to lock access to a remote server with OTPs.

every 30 seconds. When you log into your account, your distribution will prompt for your password before asking you to enter the authentication code for the account. At this point, enter the digits currently on display in the Android app.

Once you've logged in successfully, make sure you edit the `/etc/pam.d/gdm-password` or the `/etc/pam.d/common-auth` file and remove the `nullok` option to force logins through Google Authenticator. Also remember to create an account in the Android app for all the users on your installation. If you've lost your Android device that generates the OTPs, refer to the "Troubleshoot the Authenticator" box to wriggle out of a potentially sticky situation.

Going through the additional security prompt might seem like a hassle at first. However, if you're using the computer in a public place, you'll quickly learn to appreciate the benefits of the two-factor authentication mechanism.

Secure Remote Access

There's no dearth of tricks to harden an SSH connection. For example, you can change its default port and restrict access by IP addresses. The most commonly suggested mechanism is to switch to key-based authentication instead of passwords, but this still presents a single point of failure. To make it more difficult

for an attacker to compromise an SSH connection, you can also enable two-factor authentication for remote logins (Figure 6).

The procedure for adding OTP-based logins to an SSH connection is similar to the one for local logins. The only difference is that you need to install and setup the Google Authenticator PAM module on the remote SSH machine instead of the local machine. Begin by SSHing into the remote server as the user you wish to protect such as `ssh bodhi@remote-machine`.

Now open another SSH connection to the machine and use this second connection to make changes to SSH's configuration. When you restart the SSH service on the remote machine, it won't close open connections. So the first connection acts as a fail-safe and ensures you won't lock yourself out in case of any accidental misconfiguration.

After you've logged into the remote machine, follow the steps described earlier to first install the Google Authenticator PAM module and then use the helper script to generate the 16 digit key for the remote user. Then add the details of this remote SSH user in the Google Authenticator mobile app.

Next, you can edit SSH's PAM configuration file on this remote machine with `nano /etc/pam.d/ssh`. Scroll down to the bottom and add the following lines to the file:

```

# Secure SSH with OTPs
auth required pam_google_authenticator.so nullok

```

Just like earlier, `nullok` tells PAM that this authentication method is optional. This allows users without a Google Authenticator key to still log in using their SSH password. Remember that this is just a fail safe to prevent you from being locked out in case something goes wrong with the setup process. However, once you've tested it successfully, generate a key for all SSH users and delete `nullok` from the end of this line to make logins via OTP mandatory.

After editing SSH's PAM file, it's time to configure SSH to support this kind of authentication. Open the SSH configuration file for editing with:

```
nano /etc/ssh/sshd_config
```

Look for the line that reads `ChallengeResponseAuthentication` and change its value from `no` to `yes`. If the line doesn't exist, make sure you add it manually. Save and close the file, and then restart SSH to reload the configuration files with:

```
systemctl restart sshd.service
```

When you now re-establish the SSH connection, in addition to the remote user's password, you'll also be prompted for the Google Authenticator code.

Two-factor authentication is relatively straightforward to roll out but it takes some getting used to. However, you can rest easy knowing that you've increased the security of your computer by making it virtually impossible for crackers to brute force their way into your home directory. ■■■

AUTHOR

Mayank Sharma is a technology writer. You can read his scribbles in various geeky magazines on both sides of the pond.



INFO

- [1] Google Authenticator: https://en.wikipedia.org/wiki/Google_Authenticator
- [2] FreeOTP app: <https://freeotp.github.io/>
- [3] The TOTP protocol: https://en.wikipedia.org/wiki/Time-based_One-time_Password_Algorithm



Maps with uMap

DIY Maps

uMap provides an easy way to create advanced maps based on the OpenStreetMap service. This article explains how to put uMap's features and functionality to practical use.

By Dmitri Popov

Thanks to its open nature, OpenStreetMap spurred a large number of genuinely useful applications that piggy-back on this excellent map project. uMap [1] is a case in point: This web-based application allows you to create multi-layered maps complete with markers, lines, and polygons (Figure 1).

uMap software is released under an open source license, and the project's website provides instructions on how to deploy a uMap instance on your own server. This is not a trivial task, though. So, it probably makes sense to start with a hosted service. You can choose from several hosted uMap installations,

including the one maintained by uMap developers at umap.openstreetmap.fr. Although you can use the service anonymously, it's worth creating an account, because this allows you to save and manage your maps. uMap supports several popular sign-in providers, including GitHub, Bitbucket, Twitter, and, of course, OpenStreetMap. Once you've signed in, click the *Create a map* button to generate a map.

All tools in the map editor are grouped into two panels. You'll find the navigation and sharing functions in the left panel, while the right panel holds all editing tools. The first order of business is to give the map a descriptive name and

save the map. To do this, click the *Edit map settings* button (it looks like a cog) in the right panel and enter the desired name and short description in the appropriate fields. While you are at it, you might want to adjust a few other settings. The *User interface options* section, for example, lets you customize the navigation panel by toggling zoom, search, measure, and other buttons.

Although you can leave most of the settings in other sections at their default values, there are at least two options you might want to adjust. In the *Default properties* section, you can specify the default zoom level, whereas the *Limit bounds* section lets you limit the map to a specific area (Figure 2). You can specify the desired limits manually or let uMap do the heavy lifting. Locate the desired area and click the *Use current bounds* button to populate the fields with the actual geographical coordinates. If you want to delete, clone, or reset the current map, you can do this with the appropriate buttons in the *Advanced options* section. Once you are satisfied with the settings, click *Save* to save the map.

Next step is to configure permissions. By default, only the owner (you) has editing rights, and the map is available for public viewing. To change this, press the *Update*

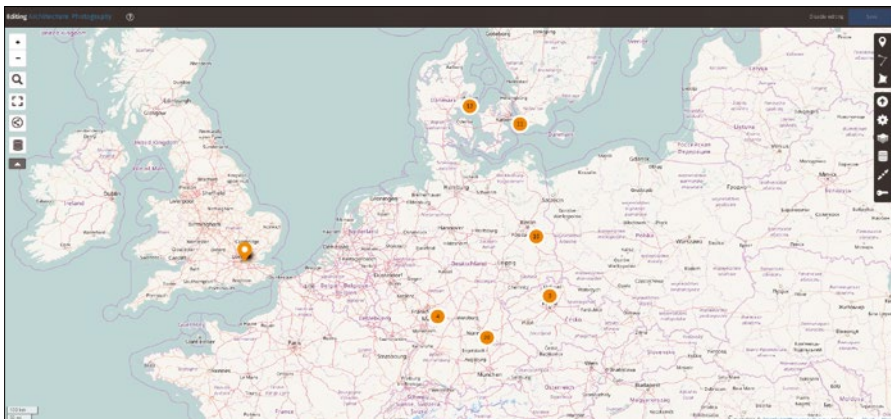


Figure 1: uMap allows you to build advanced dynamic maps in no time.

Lead image © Matttilda, fotolia.com

permission and editors button (it looks like a key) in the right panel, and then configure the permissions to your liking.

Look at the map, and you'll immediately notice that it's in French. Fortunately, you

can easily switch to an English-based OpenStreetMap. In fact, uMap supports different map styles (or tile layers), including OSM monochrome, OSM Lyrk, OSM OpenCycleMap, and many others

(Figure 3). To switch to a different map style, click the *Change tilelayers* button in the right panel and choose the desired layer. Remember to save the changes.

Once the map is ready, you can start

populating it with markers (Figure 4). To add a marker, click the *Draw a marker* button in the right panel, and then click on the desired location. This places a marker and opens the marker properties' right sidebar.

Here, you can give the marker a name and a short description. To change the marker's default shape and color, expand the *shape properties* section, and then use the *Color* and *Icon symbol* options to specify the desired color and marker shape. You can also assign a symbol to the marker using the *Icon symbol* option. Adding lines and polygons is equally easy. Use the appropriate buttons in the right panel to pick the desired tool, draw a line or a shape, and then use the properties sidebar to modify the available settings.

uMap allows you to add multiple layers to a map (Figure 5). This functionality can come in handy in many situations. For example, if you plan a trip that goes to several countries, you can create a separate layer for each destination. You can then show and hide various layers as needed. To work with layers, click the *Manage layers* button in the right panel. This opens a sidebar with the list of existing layers. You can use buttons next to

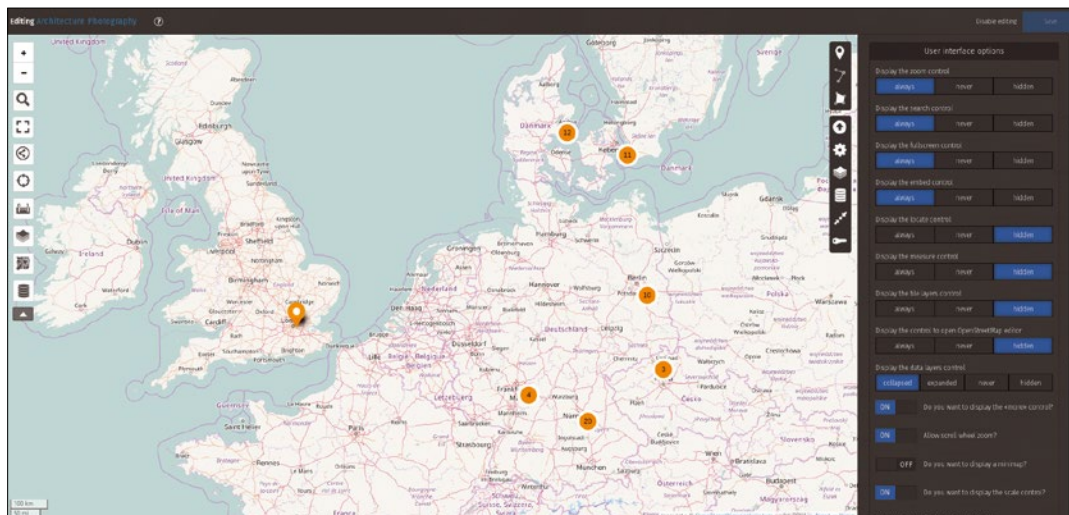


Figure 2: Modifying map properties.

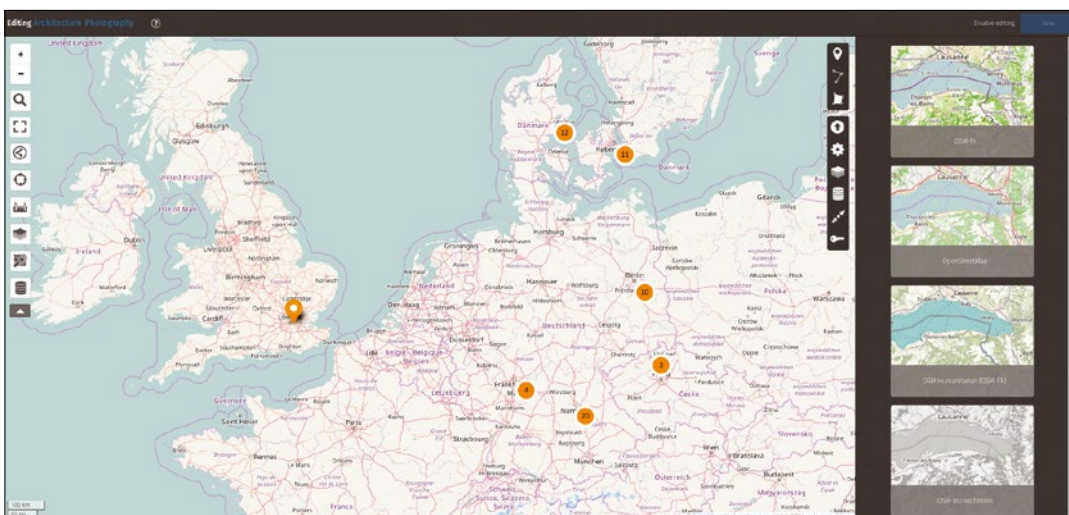


Figure 3: uMap supports various map styles.

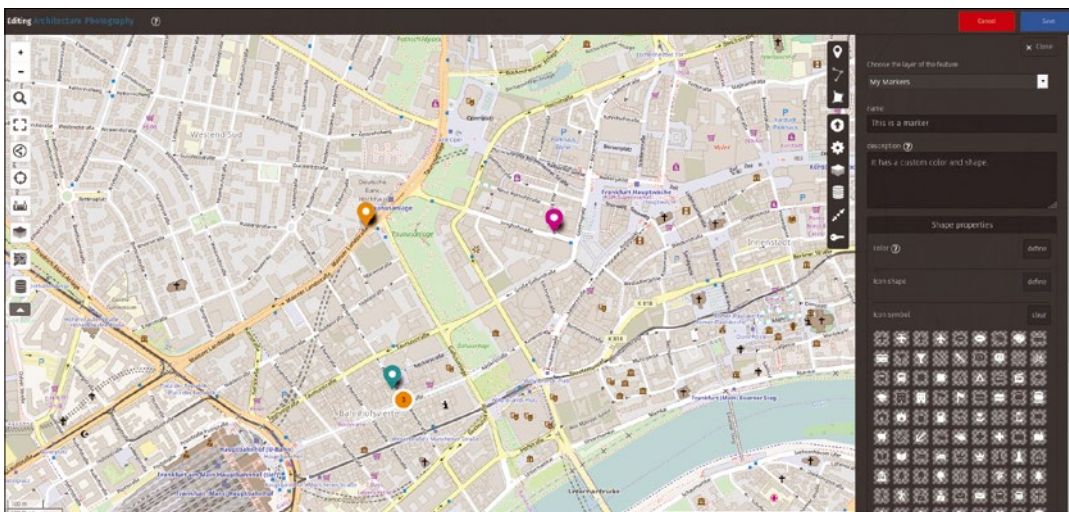


Figure 4: Adding a marker to the map.

each layer to toggle its visibility, edit properties, and view data. To add a layer, click *Add a layer* and give the new layer a name and description. If desired, change the layer type by selecting the

appropriate entry from the *Type of layer* drop-down list.

Besides the default type that shows individual markers on the map, uMap supports the Clustered and Heatmap types.

The former displays map markers as expandable circles, and counters inside them indicate the number of elements in each cluster. And, as the name suggests, the Heatmap type displays data of the

map as a heat map. To distinguish between different layers, you might want to specify the marker color and shape for the entire layer in the *Shape properties* section.

Exporting Layer Data

After you've spent hours pouring data into your map, you'd probably want to back up the final result. Although the application doesn't provide a straightforward way to create a backup file, you can use a relatively simple workaround. Take the URL of your map and remove the name of the maps from it, leaving the map's key. For example, if the map's full URL is http://umap.openstreetmap.fr/en/map/my-map_133151, the sanitized URL is <http://umap.openstreetmap.fr/en/map/133151>. Now, append */geojson* to the URL: <http://umap.openstreetmap.fr/en/map/133151/geojson>. Point your browser to this URL to get map data in the GeoJSON format. Next, use the browser's search feature to find the name of the layer you want to back up. Note the ID number of the layer and point the browser to <http://umap.openstreetmap.fr/en/datalayer/ID/> (replace *ID* with the actual id number of the layer).

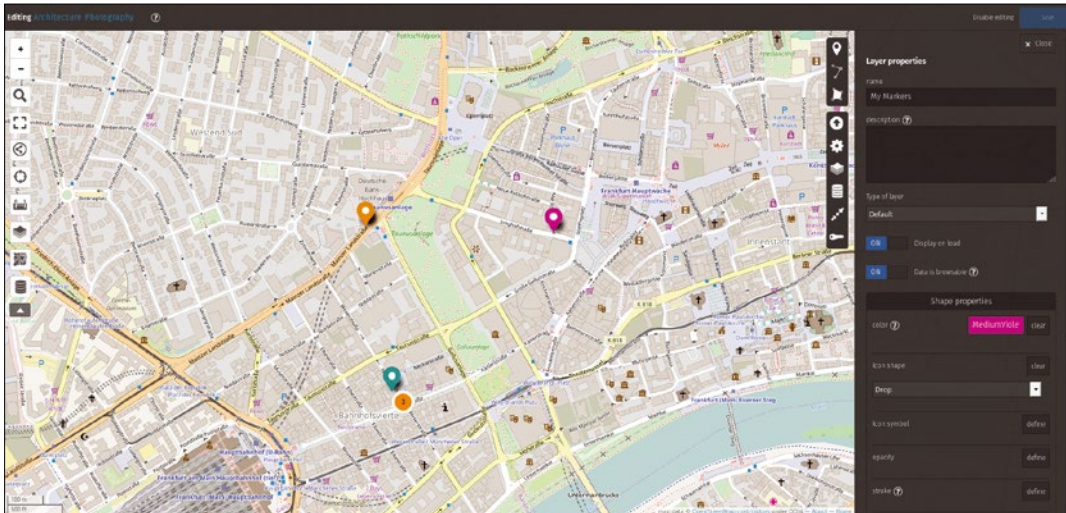


Figure 5: uMap allows you to add multiple layers to a map.

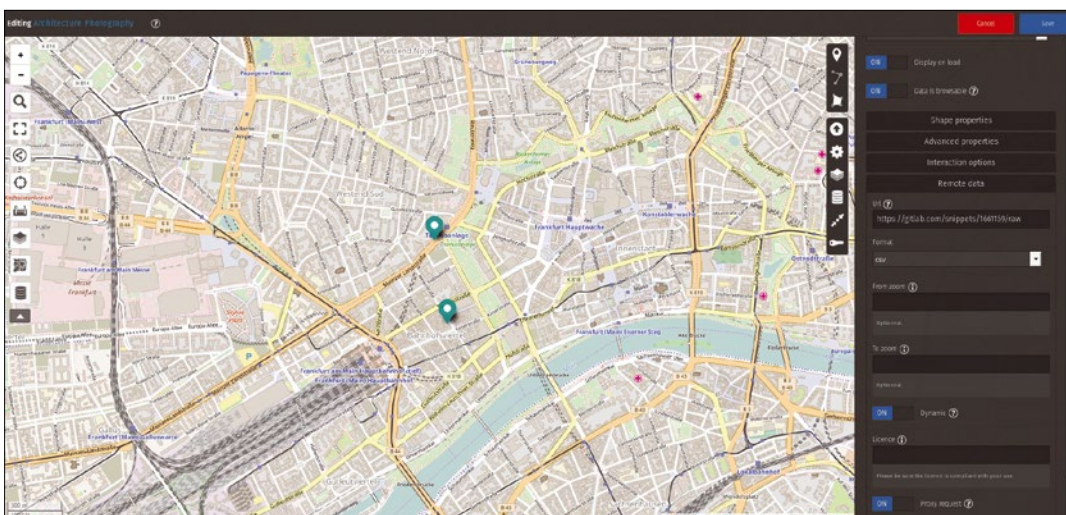


Figure 6: Linking a map layer to a remote data source.

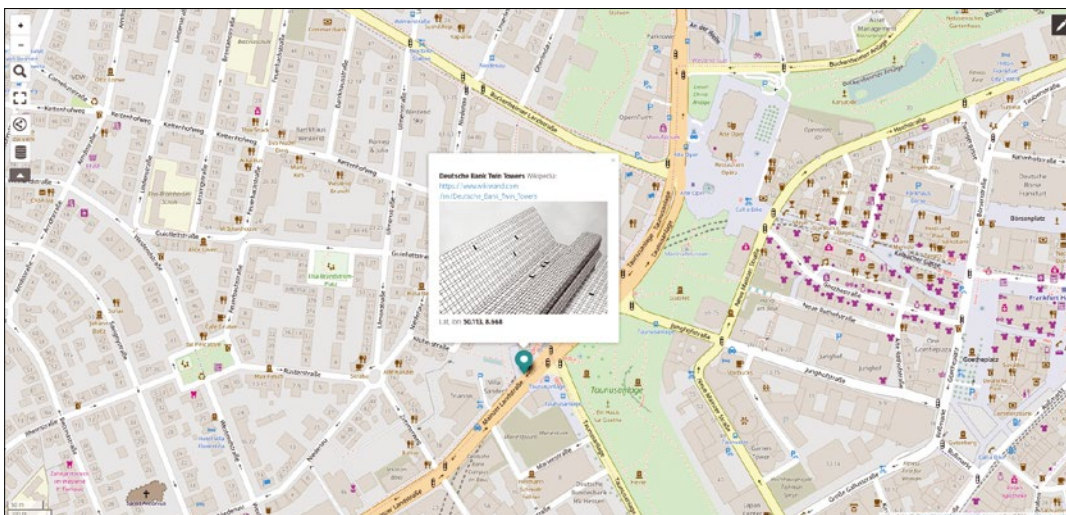


Figure 7: You can add descriptions, links, and images to a pop-up.

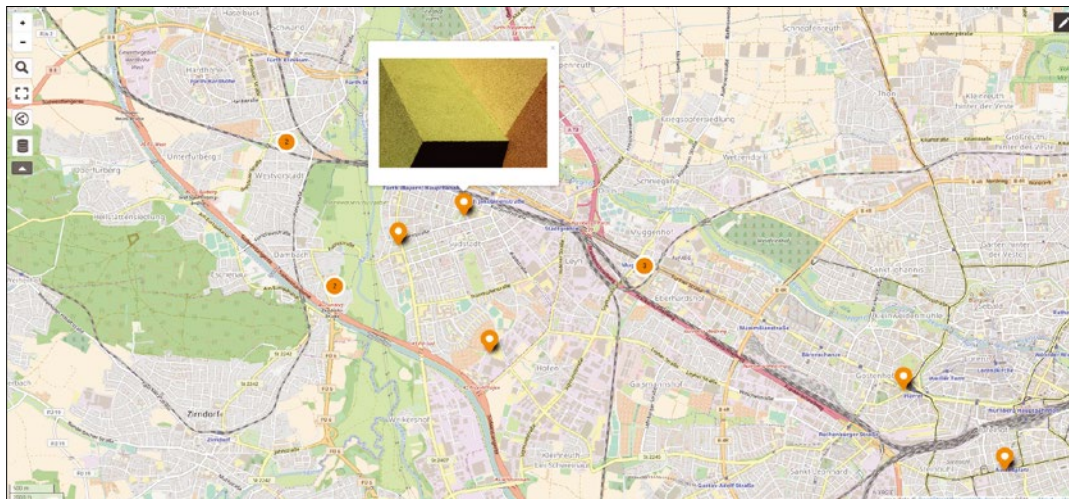


Figure 8: Map that pulls data and thumbnails directly from Mejiro.

This returns the layer data in the GeoJSON format. Then, copy the data, paste it into a text file, and save it.

Building Dynamic Maps

Although uMap makes it easy to populate a map with markers, doing this manually can still be a rather laborious process – especially if you already have basic geographical data in a spreadsheet or a text file. Fortunately, the application also offers advanced features that let you create maps that pull data from a remote source (Figure 6).

Suppose you want to build a map of all interesting locations you want to photograph during your next trip. Instead of adding them directly to the map, you can create a comma-separated text file

FROM MEJIRO TO UMAP

The ability to pull data from external sources opens a whole new world of possibilities. I use my own Mejiro PHP-based web application [2] to publish photos. It can be used to show the location of each geotagged photo on OpenStreetMap, but I wanted to display all photos in Mejiro on uMap. So I wrote a quick-and-dirty PHP script that generates a comma-separated output containing latitude and longitude values along with links to thumbnails. I put the script in the Mejiro directory on the server. Then, I created a dedicated layer in a uMap map and linked the layer to the script. You can see the result in Figure 8. The script is available on GitLab [3], and it can be easily adapted to work with other photo sharing applications that store photos and thumbnails in dedicated directories.

containing location data and link it to the map. First, you need to create a text file and publish it on the web. There are several ways to do that, but the easiest one is probably to create a public snippet on GitLab or a gist on GitHub. The snippet should contain the header in a list of entries as follows:

```
lat,lon,description
50.109,8.669,Short description goes here.
```

Once you've added the desired entries, save the snippet. To get the snippet's raw content, append `/raw` to the snippet's URL, for example: `https://gitlab.com/snippets/1771159/raw`. Copy this direct URL and switch to uMap. Here, you need to configure a layer to pull data from the snippet. To do this, click the *Manage layers* button, click *Edit* next to the desired layer, and expand the *Remote data* section. Paste the direct URL of the snippet into the *Url* field and select *csv* from the *Format* drop-down list. Enable the *Dynamic* and *Proxy request* options, and then save and reload the map. You should see map markers for each entry in the snippet. If you click on a marker, you'll see an empty pop-up. If you would like the pop-up to display something useful, uMap's got you covered.

Suppose you want the makers in a specific layer to display the content of the *description* field in the snippet. For this, you need to modify the default pop-up template. In the *Layer properties* sidebar, expand the *Interactive options* section. In the *Popup content template* field, add the `{description}` placeholder. Note that the name of the placeholder

must match the name of the field in the snippet. Save and reload the map, click on a marker, and you should see the content of the description field in the pop-up.

uMap has a few other clever tricks up its sleeve. The application supports basic Markdown options, so you can add text formatting to the descriptions directly in the snippet. It's also supremely easy to add

photos to pop-ups (See the "From Mejiro to uMap" box for more information.) Add the *url* column to the snippet and links to photos for each entry:

```
lat,lon,description,url
50.109,8.669,Short description goes here.,http://url/foo.jpg
```

In the *Popup content template* field, add the `{{url}}` placeholder. The extra curly brackets instruct uMap to render the URL as an image. Save and reload the map, click on a marker, and you should see the specified photo in the pop-up (Figure 7).

Conclusion

When it comes to creating maps with a minimum of effort, uMap is hard to beat. Not only does it allow you to create maps in a matter of minutes, the application also provides advanced functionality that can be used to create multiple layers, link them to remote data sources, and configure marker pop-ups to show all sorts of useful information. You can also publish and share finished maps with others, as well as embed maps in web pages. In short, if you are looking for a powerful yet easy-to-use mapping tool, uMap is just the ticket. ■■■

INFO

- [1] uMap: <http://wiki.openstreetmap.org/wiki/UMap>
- [2] Mejiro: <http://dmpop.github.io/mejiro/>
- [3] PHP script to generate CSV output: <https://gitlab.com/snippets/1661335>

The arrival of open hardware

Up and Coming

We look at some of the new open hardware projects underway. *By Bruce Byfield*

Work like you are living in the early days of a better nation. – Oysterband

For years, free software advocates have dreamed of hardware whose specifications and firmware are free-licensed. Yet, largely because of the costs of production, open hardware has never succeeded. Today, though, we seem to be in the first stages of an open hardware culture – and already the diversity is intoxicating.

Like free software 20 years ago, open hardware has little appeal to commercial technology companies. Small companies like bq and Slimbook have experimented with preloading free software, but even their efforts have used proprietary firmware and hardware. However, unlike free software, open hardware has previously been too expensive to be produced by individuals.

So what has changed? The question can be answered in a single word: crowdfunding.

With crowdfunding, those who dream about open hardware have a shot at making their dreams a reality. A newer company like SiFive can develop a free-licensed chip that can eventually lead to further experiments in open hardware. Companies like Design Shift can dabble with open hardware that traditional sources of funding would be unlikely to support. Even more significantly, dozens of individuals have a chance to launch small businesses that give them a chance to make a living doing what they love. If none of these ventures are likely to be the next Apple or Microsoft, few care as long as they can make enough profit to survive and develop their next products.

These developments have enough potential that Crowd Supply [1], a relatively

small site compared to Kickstarter or Indiegogo, has even started specializing in this intersection of open hardware and crowdfunding, tutoring the wannabes in the reality of business and selling the resulting products online – thereby offering a partial solution to the problems of distributing new products. It even has a section of its site labeled Open Hardware.

According to the Crowd Supply blog [2], the combination of crowdfunding and open hardware has distinct advantages, including:

- A source of income for engineers turned entrepreneur that includes not only crowdfunding, but also volunteers.
- Increased quality and rapid improvement thanks to the availability of specifications and software.
- The ability to patch, port, branch, and revive existing products quickly.
- An existing market in the open source and maker communities that has a personal interest in a product's success.
- Credibility from peer reviews.
- Market validation of concepts that might be expanded into other products.
- A low entry cost and (at least compared to traditionally marketed products) a low cost of product failure.

If some of these benefits sound familiar, they should. Open hardware shares many of these benefits with free software.

Thanks to crowdfunding, projects that once seemed impossible now have a chance.

New Pioneers

To give you an idea of what is happening, here are some of the open hardware projects now underway.

- **Keyboardio:** Currently, in the middle of production, Keyboardio has designed what is probably the ultimate keyboard (Figure 1). Mounted on hardwood, Keyboardio's Model 01 is an ergonomic keyboard with four different positions, and mechanical keys, almost two-thirds of which are individually sculpted. Both keys and backlights are programmable, and the whole is regulated by an Arduino microcontroller. The company is committed to quality, which has caused delays in production, but should be worth the wait [3]. Keyboardio is also noteworthy for having benefited by the release its firmware. With the Model 01 still in



Figure 1: Keyboardio's Model 01 is intended as the ultimate keyboard and is entirely open.

Lead Image © artqu, 123RF.com



Figure 2: Laptops that are not only free but also recyclable.

production, its firmware was recently massively improved by a Hungarian enthusiast who saw the specifications on GitHub.

- **EOMA68 laptops:** Remember the laptops with bamboo frames in William Gibson's cyberpunk stories? EOMA68 is making them for real, adding recycling and self-repairs to the famous four software freedoms and providing the 3D printer schematics for the base as well (Figure 2). The laptops also include operating systems housed in old PCMCIA cases, making them easy to swap in and out. The company expects to ship by August 2017, with the Free Software Foundation's Respects Your Freedom Certification for at least some of its offerings, meaning that it will be a completely free device [4].
- **ORWL:** ORWL is described as a physically secure computer. Developed by Design Shift, it includes an encrypted hard drive and an option to use Qubes OS, a distribution that brings an unprecedented level of security to the desktop (Figure 3). The computer requires a security key and closes down if the key is moved too far away. A metal mesh surrounds the CPU, and if the

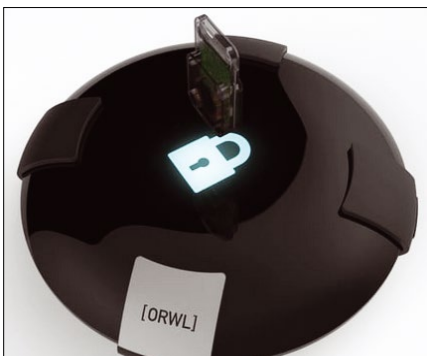


Figure 3: ORWL is a computer that specializes in physical security.

mesh is broken, the computer's memory may be wiped in some cases. ORWL uses coreboot, and its firmware is free-licensed but has been criticized for using a proprietary microcontroller – a circumstance that Design Shift hopes eventually to change [5].

- **SiFive:** A major obstacle to open hardware is the difficulty in finding CPU chips with free-licensed firmware. Even when such chips are available, suppliers have a habit of switching them for cheaper proprietary chips. Founded by the inventors of the relatively new RISC-V chips, SiFive designs custom free-licensed chips in partnership with Taiwan Semiconductor Manufacturing Company, Limited (TSMC) [6]. Open-V [7] also has been raising funds for RISC-V chips, while HiFive1 [8] has completed an Arduino-like board using RISC-V chips – both, presumably, with the assistance of SiFive. Already, RISC-V chips seem to be increasing the number of free software products.

These are just a few of the open hardware projects available. Others range from tools for hobbyists to devices suitable for individual desktops to small businesses. They include personal cloud devices, radios, a \$50 smartphone, USB password managers, a bench power supply, a mini-laptop, an open source 3D printer, and many more, with others coming into existence every week.

Not all these open hardware projects will meet their campaign goals. Others will fail to come to market or falter after the release of their first products. However, these outcomes are typical not only of open hardware but of any new products. The point is that, win or lose, the creativity and diversity being displayed shows just how much attention is being paid to open hardware today – much of it escaping the notice of the mainstream technology media.

The Challenges of Production

Fundraising, of course, is only part of the challenge for a new hardware vendor. Production, too, can be a major

challenge, with unexpected delay following closely on delay. New vendors often have to face near-monopolies from large rivals. Parts may be unavailable because they are reserved for large corporations. Many manufacturers are uninterested in orders of under several hundred thousand, and others give small orders low priority. Some manufacturers have even been known to delay or cancel orders because larger ones have been received. Such obstacles mean that any new product, free-licensed or not, are likely to be delayed or fail through no fault of their own.

Additionally, more than one open hardware vendor has found that frequent trips to Asia, or long stays, are necessary to ensure quality and to respond quickly to emergencies. Both the Keyboardio [9] and EOMA68 [10] blogs detail some of the tribulations of modern production for small vendors, and they can make for harrowing reading. The fact that people persist in their efforts to bring open hardware to market despite the difficulties shows just how dedicated they are.

Despite these difficulties, open hardware vendors soldier on. In coming months, I will look at some of these vendors and the wonders they produce. Meanwhile, take a look at the crowdfunding sites and the wonders on display. For anyone with an interest in open hardware, the experience is like looking at a *SkyMall* catalog for techies. Finally, after all these years, open hardware is happening at last. ■■■

INFO

- [1] Crowd Supply: <https://www.crowdsupply.com>
- [2] Crowd Supply Blog: <https://www.crowdsupply.com>
- [3] Keyboardio: <https://shop.keyboard.io/>
- [4] EOMA68: <https://www.crowdsupply.com/eoma68/micro-desktop>
- [5] ORWL: <https://www.crowdsupply.com/design-shift/orwl>
- [6] SiFive: <https://www.sifive.com/>
- [7] Open-V: <https://www.crowdsupply.com/onchip/open-v>
- [8] HiFive1: <https://www.crowdsupply.com/sifive/hifive1>
- [9] Keyboardio blog: <http://blog.keyboard.io/>
- [10] EOMA68 blog: <https://www.crowdsupply.com/eoma68/micro-desktop/updates>

Pinning sources in Debian

PROCEED WITH CAUTION

CAUTION

CAUTION

CAUTION

CAUTION

CAUTION

Debian discourages the use of pinning to set preferences for package repositories, because the practice can have disastrous results. We take a closer look. *By Bruce Byfield*

Pinning is the black art of Debian and its derivative distributions. Using pinning, you can set your preferences for which package repository to use, either for all installations or upgrades, or for a specific set of packages. Officially, however, Debian discourages pinning, because it can prevent package upgrades or even corrupt an entire system if used carelessly.

Debian, as you may know, uses three main repositories [1]. These are Stable, Testing, and Unstable, also known by their release names, which are currently Jessie, Stretch, and Sid – all characters from the *Toy Story* movies. A new package enters Debian in Unstable, and, when it meets certain requirements, moves to Testing. When a general release is made, the package moves to Stable. Between releases, StableUpdates and Backports are used to help keep Stable up to date with borrowings from Testing. Debian derivatives like Ubuntu organize repositories by other criteria but are still likely to have some repositories that are more stable or otherwise preferred.

However, both Debian's and Debian derivatives' main repositories are divided into sections based on licensing. The main section contains free-licensed packages, including core system components. By contrast, contrib contains free-licensed

packages that depend on proprietary applications, while non-free contains proprietary packages. Debian installs with only main enabled, although contrib and non-free can be enabled by editing `/etc/apt/sources.list` and then running `apt-get update`.

The reason Debian discourages pinning is that it disrupts the repository system's careful arrangement of repositories by package quality – which can be disastrous at times. For example, when systemd was in Unstable, installing the packages on an otherwise Stable system slowed performance and limited the available display resolutions. These problems persisted for months, with no solution except a complete reinstall.

Such problems are especially likely to occur in the months before a general release, when developers are concentrating on perfecting the packages in Testing to prepare for their move into the new Stable repository and often neglecting Unstable. They are also far more likely with core components, such as Linux kernels or desktop environments that have numerous essential dependencies.

Using third-party repositories can sometimes cause similar problems. Similarly, pinning can add unfree packages to an otherwise free-licensed system.

All the same, pinning remains popular. Debian can take a long time between releases, and some users will always want the most up-to-date versions

```
# Stable (Jessie)
deb http://ftp.us.debian.org/debian/ jessie main contrib non-free
deb-src http://ftp.us.debian.org/debian/ jessie main contrib non-free

deb http://security.debian.org/ jessie/updates main contrib non-free
deb-src http://security.debian.org/ jessie/updates main contrib non-free

# jessie-updates, previously known as 'volatile'
deb http://ftp.us.debian.org/debian/ jessie-updates main contrib non-free
deb-src http://ftp.us.debian.org/debian/ jessie-updates main contrib non-free

#Jessie backports
deb http://ftp.debian.org/debian jessie-backports main contrib non-free

# Testing (Stretch)
#deb http://ftp.us.debian.org/debian/ stretch main contrib non-free
#deb-src http://ftp.us.debian.org/debian/ stretch main contrib non-free

# Unstable (Sid)
#deb http://ftp.us.debian.org/debian/ sid main contrib non-free
#deb-src http://ftp.us.debian.org/debian/ sid main contrib non-free
```

Figure 1: Pinning prioritizes the package sources listed in `/etc/apt` that are not commented out.


```

bb@nanday:~$ apt-cache policy
Package files:
100 /var/lib/dpkg/status
   release a=now
500 http://ftp.debian.org/debian/ jessie-backports/non-free Translation-en
500 http://ftp.debian.org/debian/ jessie-backports/main Translation-en
500 http://ftp.debian.org/debian/ jessie-backports/contrib Translation-en
100 http://ftp.debian.org/debian/ jessie-backports/non-free amd64 Packages
   release o=Debian Backports,a=jessie-backports,n=jessie-backports,l=Debian Backports,c=non-free
   origin ftp.debian.org
100 http://ftp.debian.org/debian/ jessie-backports/contrib amd64 Packages
   release o=Debian Backports,a=jessie-backports,n=jessie-backports,l=Debian Backports,c=contrib
   origin ftp.debian.org
100 http://ftp.debian.org/debian/ jessie-backports/main amd64 Packages
   release o=Debian Backports,a=jessie-backports,n=jessie-backports,l=Debian Backports,c=main
   origin ftp.debian.org
500 http://ftp.us.debian.org/debian/ jessie-updates/non-free Translation-en
500 http://ftp.us.debian.org/debian/ jessie-updates/main Translation-en
500 http://ftp.us.debian.org/debian/ jessie-updates/contrib Translation-en
500 http://ftp.us.debian.org/debian/ jessie-updates/non-free amd64 Packages
   release o=Debian,a=stable-updates,n=jessie-updates,l=Debian,c=non-free
   origin ftp.us.debian.org

```

Figure 2: By default, Debian assigns all the main repositories an equal priority. Lesser repositories like Backports and Updates are given a lesser priority.

of applications. Moreover, once set up, pinning is more convenient than alternatives such as constantly commenting and uncommenting lines in `/etc/apt/sources.list`. Fortunately, by keeping informed about Debian development and taking a few precautions, pinning can be used with minimum risk – although some risk is likely to remain.

Getting Priorities Straight

Pinning sets priorities for packages in the repositories listed in files in `/etc/apt`, such as `sources.list`, or else those at a particular URL (Figure 1). If you want to borrow from a repository, you must first add it to the sources for the system and make sure that the entry is not commented out and then run `apt-get update`. Otherwise, pinning has no effect.

By default, Debian assigns all repositories and gives each of the main repositories section an equal weight of 500 (Figure 2), preferring none of them (see below). You can check the current priorities on a system by running `apt-cache policy`. To read the priority for a particular package, run `apt-cache policy PACKAGE`.

Changing priorities means creating a preference file. In the past, all priorities were listed in `/etc/apt/preferences`, with each priority setting in a stanza separated from the others by a line above and below. In the last few years, though, Debian and its derivatives have used individual files in the `/etc/apt/preferences.d` directory (Figure 3). These individual

files often use a `.pref` extension, although any file in `preferences.d` is read in setting priorities.

The structure for defining a preference can begin with an Explanation line for adding comments, although in practice this line is rarely used. More often, the structure consists of three lines. The first line, `Package`, lists the package affected. The first line accepts regular expressions, so an asterisk (*) sets the priority for an entire repository – something that can be dangerous to do – while `/WORD/` applies to any package that contains the enclosed word.

For safety, you are better off specifying an entire package or set of packages. For example, if you want the latest packages for the solitaire game `pysolfc`, you might complete the first line with `*pysolfc*`. For a package with numerous dependencies, such as a desktop environment, you might have to choose the regular expressions carefully, although probably dependencies would drag in any other files not covered by them – something you definitely will want to check. The most common first line is:

```
Package: *
```

Pin, the second line, usually defines the package source by defining the archive (`a=`), version (`v=`), release name (`n=`), and/or section component (`c=`). A private repository can be defined with label (`l=`). These building blocks can be combined in a comma-separated list, so that the line

```
Pin: release a=unstable,c=main,v=8.6
```

would apply to the main section of Unstable in Debian's 8.6 release. Less commonly, origin URL refers to a mirror site, or `origin ""` to the local system. Refer to the `sources.list` file to check that you have identified a source properly.

The third line is Pin-Priority. Its working is simple: The package source with the highest number is used first. If it is

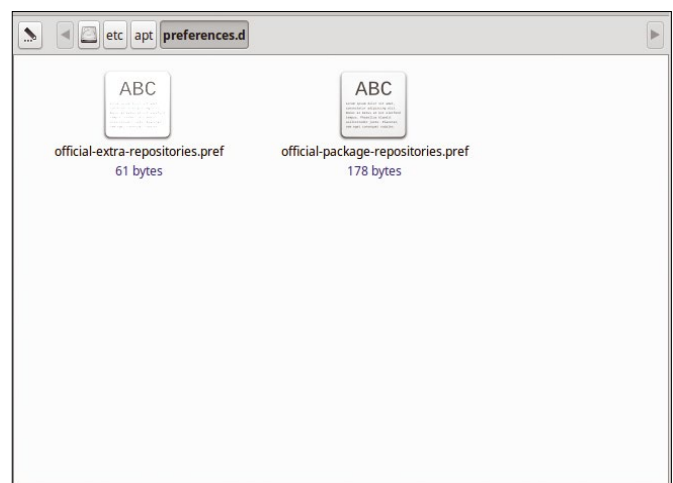


Figure 3: The default content of `/etc/apt/preferences.d` in Linux Mint.

```

Explanation: Uninstall or do not install any Debian-originated package versions
Explanation: other than those in the distribution codenamed with stretch or sid
Package: *
Pin: release n=stretch
Pin-Priority: 900

Explanation: Debian unstable is always codenamed with sid
Package: *
Pin: release n=sid
Pin-Priority: 800

Package: *
Pin: release o=Debian
Pin-Priority: -10

```

Figure 4: A preference file that gives high priority to all code-named repositories and a low priority to all other sources. By setting priorities in this way, the file favors high-quality sources.

not available for some reason, the package will be installed – if possible – from the source with the next highest possibility. Table 1 lists the effect of each priority. A sample third line would be:

```
Pin-Priority: 500
```

Priorities can be used in a number of ways beside the obvious. For example, problems with dependencies can sometimes be fixed by setting priorities to a number over a thousand, so that you can downgrade packages to what they once were. Similarly, a cautious administrator could set priorities so that the Stable has a priority of 1000, and everything else has a negative priority, so that even if a user with the right privileges adds another source, those sources will never be used in preference to Stable (Figure 4).

However you set source priorities, you always need to be cautious, because dependencies can lead to unexpected results. You should also be aware that, if a source is unavailable for some line, the source with the next highest priority will be used, which can sometimes lead to a broken package. In my experience, the safest way to use pinning is to assign a priority to the fewest number of packages

possible. If that requires more thought and more .pref files, it should also cause fewer problems.

Using Pinning

When you have set priorities, run `apt-get update` to enable them. Before adding or upgrading packages, you should check the descriptions of the packages involved. Take special note of their dependencies: Generally, the more dependencies a package has, or the more core components are among its dependencies, the more borrowing from Testing, Unstable, or third parties is likely to cause problems.

Another precaution to use with pinning is to run `apt-get` first with the `-s` or `--simulate` option. This option tells you what the result of your action will be but makes no changes to your system. The more complicated an action, the more sensible using this option becomes.

Pinning does not work automatically – only when you specify. It can be used with one of two command structures. The first example is:

```
apt-get install PACKAGE/unstable
```

Replace `unstable` with the repository that you wish to use.

fix the broken packages before you can install or upgrade anything else.

The second command structure example is:

```
apt-get -t unstable install PACKAGE
```

Again, replace `Unstable` with another repository if necessary.

With this structure, both the package and its dependencies are taken from `Unstable`. But whether this tactic works depends on how much effort the maintainer has put into the package. If they have only made sure that the package works for a specific set of circumstances, you are just as likely to have broken packages to fix as with the first tactic. No matter how you use pinning, some uncertainty always remains possible.

A Final Caution

If I sound overly cautious, the reason is that, like many Debian users, I have sometimes been tempted by one upgrade too many, condemning a system to dependency hell and hours of recovery.

Pinning can be a handy tool, which is why it remains popular. However, it is not a tool for beginners, nor one to use recklessly. After all, at this stage in Linux's development, having the latest upgrade matters less than it once did when the gaps in functionality were larger and more frequent.

Still, if you are like me, you will probably be tempted to pin at some point. If so, then use it carefully, so that it helps rather handicaps you. ■■■

INFO

- [1] Debian repositories: <http://www.datamation.com/open-source/when-to-use-which-debian-linux-repository.html>

TABLE 1: Priority Settings

Setting	Effect
More than 1000	Version is installed even if it means downgrading the package version. Can be useful for fixing broken packages.
990-1000	Version is installed even if not from the target release, unless the installed version is more recent.
500-989	Version is installed unless the target release has a version or the installed version is most recent. This is the default priority for Debian.
100-499	Version is installed unless another version exists in another repository or the installed version is more recent.
0-99	Version is only installed if no other version is already installed.
Less than 0	Prevents the version from being installed. Useful when you suspect a version may cause problems.

MORE UBUNTU!



Can't get enough Ubuntu? We've got a whole lot more!

Ubuntu User is your roadmap to the Ubuntu community. In the pages of **Ubuntu User**, you'll learn about the latest tools, best tricks, and newest developments in the Ubuntu story.

Ubuntu User helps you explore the treasures of open source software within Ubuntu's expansive repositories. We'll bring you exclusive interviews with Ubuntu leaders, keep you current on the exciting Ubuntu community, and answer your most perplexing Ubuntu questions. Learn how to choose a video editor, find the perfect tool to customize your desktop, and configure and manage Ubuntu systems using the best admin tools.

DON'T MISS ANOTHER ISSUE!



HUGE SAVINGS OFF THE NEWSSTAND PRICE!

SUBSCRIBE NOW: SHOP.LINUXNEWMEDIA.COM



What?!
Archives
come with my
digital subscription?



Archives + Current!

Sign up for a digital subscription to get the latest issues of Linux Magazine, PLUS access to archive articles.

shop.linuxnewmedia.com/digisub

That's a lot of articles!



Ben Everard

The unstoppable march of progress

We often talk about modern technology as advanced, and think of it as the pinnacle of electronics mastery. It's not. Future historians will look back on the devices and computers we use now and think of them as the first, fledgling attempts by a people still getting to grips with the ideas and concepts of the area. Don't forget that we're still well within the first hundred years of programmable computers, which may seem like a long time, but in terms of human progress, it barely counts as a moment. I don't have a magic looking glass that

tells me which way technology will go, but I am sure that the golden age of computing is ahead of us, and not behind.

If you think of it in terms of another icon of modern technology, the car, the Ford Model T didn't come out until 139 years after the first full-sized, steam-powered car. Today's so-called advanced technology will come to be viewed as we view cars of the 1800s. It's clunky, creaky and primitive, but it's also exciting, because now is the time when things are changing fast and advancements come quickly.

One thing about advancement is that when things have truly advanced, they become mundane. When there are no more giant leaps of progress, things quickly become ordinary. So, maybe our computers are slow and prone to failure, but they're exciting in way future generations will never be able to understand.

This month, we're focusing on the emerging technologies of the future, not the past. Simon Phipps takes a look at how copyright could change to support digital progress, Valentine Sinitsyn shows you how to work with virtual machines, Mike Saunders looks at the future of LibreOffice and I embrace systemd.

The future is coming; let's reach out and grab it.

– Ben Everard



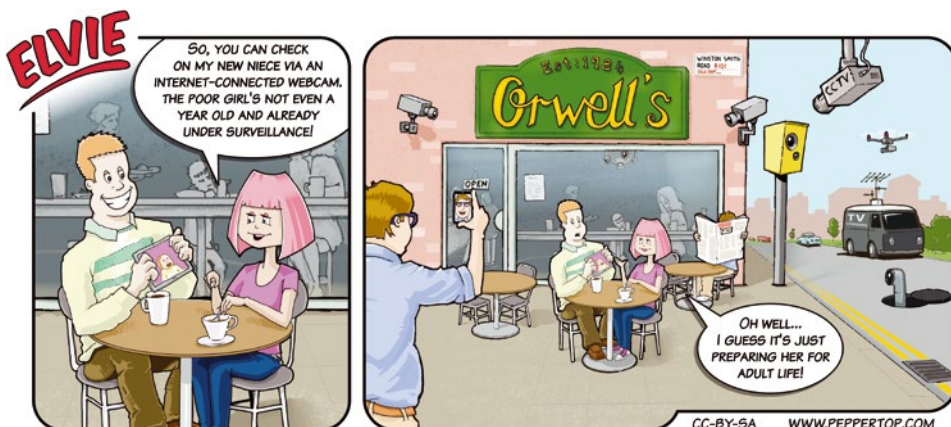
Andrew Gregory



Graham Morrison



Mike Saunders



LINUXVOICE

News 70

Simon Phipps
It's not enough to tinker with copyright rules; the whole concept needs reviewing for the digital age.

Unity Farewell 71

Andrew Gregory
Choice is good? And we've chosen.

Start Your Own FOSS Project 72

Mike Saunders
Don't just consume free software – contribute to it! We share the tips and tricks required to start a successful FOSS project.

Doghhouse – Professional Programmers 75

Jon 'maddog' Hall
"maddog" takes a look at various factors that go into creating good code.

FAQ – Common Crawl 76

Ben Everard
Download the entire web to kickstart a data science empire.

Core Tech 78

Valentine Sinitsyn
Ever wondered what's happening inside a virtual machine? Join us for an exciting tour into virtualization deep waters.

FOSSPicks 84

Graham Morrison
We explore KDevelop 5.1, Riot.im, Cursynth, QMapShack 1.8, KWipe 2.1.3, Rapid Photo Downloader 0.9, Kakoune, VPG 0.2.8, Anbox (alpha), Terasology Alpha 7, and Mudlet 3.0.

Tutorials – Systemd 90

Ben Everard
Take control of the services running on your Linux machine.

Tutorials – LibreOffice 92

Mike Saunders
Discover some hidden and lesser-known features in LibreOffice, to help you work faster and smarter (and gain extra geek points).

NEWS ANALYSIS

The Linux Voice view on what's going on in the world of Free Software.

Opinion

Copyright Needs Radical Reform

It's not enough to tinker with copyright rules; the whole concept needs reviewing for the digital age. **BY SIMON PHIPPS**



Simon Phipps is a board member of the Open Source Initiative, the Open Rights Group, and The Document Foundation (makers of LibreOffice).

Copyright is back in the news in Europe. In the UK, the Digital Economy Bill proposes to increase the maximum prison sentence for online copyright infringement to 10 years. Meanwhile, an extensive modernization of copyright for the EU is also in progress, with a goal of making the treatment of copyright the same across Europe, especially in relation to digital media.

None of the proposals I have seen address the most significant issue we face today: Copyright was never meant to apply to things you and I routinely do. It was a law made in the context of the end of general censorship and the rise of the printing press. It was intended to protect the weak from the powerful and the powerful from each other. It never applied to people who read printed works – only to those who printed them. That's why the penalties associated with infringement are so disproportionate; they are meant to influence magnates, not minnows.

We've seen the immense harm that's resulted from the semantic sleight of hand that justifies the violation of our rights because the phrases "war on drugs" and "war on terror" includes the word "war." A similar,

more cunning sleight of mind observes that every enjoyment of a work in the digital age requires a "copy." Use of that word is taken to mean copyright law applies, and thus a license is required by the consumer to waive the monopoly that copyright grants.

In turn, those licenses are used as a control point at which to impose extensive contractual terms, in which anything goes. Those terms can prevent sharing; limit which devices can be used to handle the work; restrict how often or how long access is allowed; extend control beyond the duration for which a license is actually required; and so on.

Applied to printer cartridges or coffee capsules, those contracts – keyed, remember, on a need for a copyright license – restrict who we can buy supplies from, who can mend our stuff, and who we can

manipulating digital bits means "making a copy" has become an excuse for the imposition of abusive licenses by middle men on the individual. Whereas originally copyright was about how works were created and distributed, today people believe copyright is about the way works are enjoyed, and that belief has led to a self-perpetuating spiral into the abyss of control. Radical reform is overdue.

Radical means "from the roots upwards." My call is for copyright to be reinterpreted for the connected era. That's not a call to eliminate copyright, which I think is too extreme a remedy. But use of copyright today far exceeds the ways the framers of copyright law ever imagined it would be used. The social contract upon which it is based – the exchange of temporary monopoly among distributors for protection of both the creator and ultimate enjoyer of the

work – remains valid. But it needs recasting for an age where every citizen is a peer, rather than in an age of controlling hubs and passive spokes. One great start would be to stop treating the instantiation of data in device

"Originally copyright was about how works were created and distributed; today people believe copyright is about the way works are enjoyed."

pass it on to. That could seriously impact the dynamics of the emerging, connected economy by, for example, chilling open source development, criminalizing competitors to established businesses, and robbing citizens of the tools to maintain both transparency and privacy.

That's a disaster for the digital age. Whereas book publishers' control of what you did with their books under the original intent of copyright ended at the point of purchase, the convention that

memory as a "copy" for example.

To be clear, I am not an advocate of a world where no one respects anyone else's rights. For example, I prefer to simply avoid music and movies that I am only able to access illegally. But the unvarnished reptilian selfishness of the music industry (and others) is pushing the copyright agenda too far away from the interests of society. It's time for radical reform, not just readjustment of the broken legacy. ■■■

Unity Is Strength

BY ANDREW GREGORY

Choice is good? And we've chosen.

So farewell then, Unity. Ubuntu's new, shiny, convergence-enabling desktop, which I rather liked once I got used to it, will go the way of all cast-off projects as of version 18.04, when the most important Linux distro around will switch back to Gnome as its standard desktop. Unity will continue on as Yunit, a community-run fork, and the world will go on turning. This is all good.

Since the day Canonical chose Unity over Gnome as the user-friendly desktop, their destinies have been locked together. This town (in the sense that Linux is a town, which of course it isn't) ain't big enough for the both of them. So given that only one was going to survive, we should be glad that it's Gnome that

wins, rather than the less popular, less open competitor.

We should be glad that, thanks to the GPL, Unity will live on for as long as there are developers willing to work on it. And we should be glad that desktop Linux has a benefactor who, when he realizes that one of his favorite projects isn't working, isn't afraid to pull the plug to stop wasting time and effort. I speak, of course, of Mark Shuttleworth, Ubuntu's End Of Level Boss, who tried to convince us that convergence was the future back in the day. It turns out that people don't want a single operating system that works across all their devices and are quite happy to use different interfaces on different devices.

This isn't the end of innovation from Canonical, but it is the end of this particular innovation. The next one might be more successful or less successful, but we'll only find out once it exists.

Unity was so central to Ubuntu and Canonical for such a long time (it was unveiled with great fanfare back in 2010) that the decision to go back to Gnome can't have been easy. So much mental energy has gone into it that it must have felt like a defeat to let it go, but it's far healthier to think of its demise as paving the way for the next win. With this in mind, I take my hat off to Canonical for having the guts to admit when it's got things wrong: That's the first step to getting things right. ■■■

A Webzine for High-Performance Computing Specialists

ADMIN
Network & Security

If you work with high-performance clusters, or if you're ready to expand your skill set with how-to articles, news, and technical reports on HPC technology.

<http://hpc.admin-magazine.com>

Get Involved: How to Start Your Own FOSS Project

Don't just consume Free Software – contribute to it! We share the tips and tricks required to start a successful FOSS project.

BY MIKE SAUNDERS

There are many ways you can give something back to the free and open source software (FOSS) community. You can help new users come to grips with Linux or write documentation for your favorite app. If you have coding or graphic design skills, you can help in those areas as well. But, what if you're really itching to contribute to a FOSS project and can't find one that really appeals to you? The answer, of course, is to start your own!

Now, obviously, you'll need some prerequisite skills for this. You'll need to be pretty adept at the programming language you're going to use. But don't let that put you off – if all goes well, you'll spend more time implementing patches from other users rather than writing new code by yourself. I can attest to this: When I started the MikeOS project [1], I wasn't especially knowledgeable about x86 assembly language. But I learned the basics – enough to judge the quality of the patches that then started trickling in.

Creating a whole new FOSS project may seem like a daunting task, but if you break it down into steps, it becomes a lot easier. We've watched many FOSS projects come and go over the years – and have our own experiences starting and contributing to them – so this month, we'll share what

we've learned to help you successfully kick-start your project.

Step 1: Set the Goal (and Name)

What sort of software are you going to create? Whether it's a desktop app, a programming tool, or a game – or anything else – make sure it has a reason to exist. For example, if you want to make an email client just because

you're curious about how they work, that's fine – good luck! But it's not really worth creating yet another SourceForge or GitHub page for it. If you define a clear goal for your app, however, you'll stand a much better chance of attracting new contributors.

This goal is up to you: Maybe you want to make the world's fastest email client or a text editor geared toward a specific programming language. Perhaps you want to create the world's most awesome Tetris variant or a DHCP server that only needs 2K or RAM to operate (e.g., for embedded devices). Whatever the case, have something you can shout about on your web page – something that makes potential contributors think: "Cool, I've not seen something like that before!"

Then you have to come up with a name. This is one of the trickiest aspects of launching a new FOSS project, because it's difficult to change it further down the road. You can modify your goals, your branding, and even the structure of your community, but you won't be able to change your project's name without major headaches.

Take Gimp as an example – great piece of software but absolutely ridiculous name (Figure 1). If you've been using the application for a long time, you might not think about how bad it is – but have you ever tried explaining or advocating Gimp to a Windows or Mac OS user? Just think of the images that people get in their heads when they hear the word gimp.

Over the years, there have been many discussions about changing Gimp's name. But it's such a mature and well-established piece of software, that it'd be an enormous job. There's so much Gimp-related information and material on the Internet that, even if the team decided on a name change, all that other content would refer to Gimp for years to come.

Some people argue that your project's name should say exactly what it does, but I don't think that's important. Firefox doesn't set fire to foxes, but it's short, snappy, quirky, and makes for a great logo. Think of Google, Flickr, Twitter, and many other strong web brands – the names have little

Figure 1: Gimp has a pretty good mascot (Wilber), but the application's name has all manner of problems.



relation to what the services provide, but everyone knows them.

So keep the name short (two to four syllables), make sure it's easy to pronounce (think of all the different ways people can say Mageia), and check that it won't offend anyone in the world's major languages. Also consider how people can search for information about it. Gnome's web browser is called Web, for example, so if a new user needs some help, he or she will search the web for, er, Web. That makes it much harder to get targeted results.

Step 2: Build a Brand

Once you have a name and some goals for your project, it's time to start coding. You should really do this before even creating a website or asking for potential contributors. The web is laden with of 0.0.1 proof-of-concept FOSS projects that never went anywhere – projects where the original creators had grand ambitions, put loads of work into fancy websites and logos, but didn't even provide a working version of the software for people to try.

So, get cracking with the code. Even if you only get to version 0.1, missing some key features, if you have something that people can actually compile and use (or play in the case of a game), it'll be much easier to bring new contributors on board. Not only will you have something to build on, but you also prove to your fledgling community that you are capable of writing code and doing the grunt work to keep things moving.

Then think about branding. A logo or mascot isn't essential, but it adds some personality to your project. It's a good idea to start with something very simple, because over time you can then make variants on the original logo with more detail. Consider the LibreOffice document logo (Figure 2): It's basically two shapes, in black and white, but the individual modules of the suite (Writer, Calc, Impress, etc.) all have their own variants of the logo with more details inside the document shape.

Logos can be very simple yet extremely memorable. Think of IBM's logo – just those three letters, but drawn like a grid. Or Nintendo, which is just that word in a particular font with an oval around it. For my project, MikeOS, a contributor sent in a logo design (Figure 3), which we still use. The green triangle going through the letter "O" doesn't mean anything, but it's simple, unique, and easy to reproduce in different sizes.

Next up, you'll want some kind of web host for your project. There are many of these – e.g., GitHub and SourceForge – all with their own features and restrictions, so you should investigate a few and find one that's best for you.

When creating a web page for your app, there are some things you absolutely should do. First,

somewhere near the top of the page, state what your project does. This simple description is often buried under news updates or other content – or isn't even there. When compiling the FOSS Picks section of this magazine, we come across many projects for which there's no actual short and obvious description of what they do. So we just skip over them. They may be great, but the websites aren't doing their job.

It's a good idea to have some news updates on the front page as well – for new major/minor releases or new contributors to the project. This gives visitors an immediate indication of how active (or not) your project is.

Step 3: Develop Your Community

So you've got a working alpha version of your app or game. You've given it a snappy name, come up with a clear and striking logo, and your website has all the essentials, neatly structured to welcome potential new contributors. Where do you go from here?

Well, you need to get the word out. Reddit is a great site for this – go to the open source [2] or Linux [3] subreddits and write a post explaining who you are, what you're doing, and how people can get involved. Remember that SourceForge and GitHub are littered with abandoned 0.0.0.1 FOSS projects, so you really need to sound active and positive. Make it really easy for potential contributors to try your app (e.g., with prebuilt RPMs, .debs, Snaps, Flatpaks, Docker images, etc.) Maybe create a few easy hacks – coding or design jobs that don't require much time but will help people join your community. Set up a mailing list and encourage people to voice their opinions.

Another site where you can post your announcement is Hacker News [4]. Note that it's a very busy site, so unless you're very lucky, your post will fall off the front page within 20 minutes. But it's still worth a try. Also try talking to Linux journalists – like us – so that we can learn about your project and consider featuring it in the magazine.

In the early days, you may have new contributors trying to one-up each other to establish control in the project. There's nothing wrong with making clear from the start that you're the "benevolent dictator" – after all, someone has to make decisions, and endless arguing at the start typically kills projects in their infancy. Don't be afraid to be the main decision maker, but also encourage your contributors to set up their own subprojects (e.g., documentation, design) and teams, so that



Figure 2: Notice how the main LibreOffice logo at the top is very simple, and variants on it are used for the components of the suite.

Figure 3: The MikeOS logo was created by someone who just happened to randomly come across the project, but it has survived the test of time.



The Art of Bikeshedding

Parkinson's law of triviality, commonly known in FOSS circles as "bikeshedding," states that "members of an organization give disproportionate weight to trivial issues." In other words, people spend huge amounts of time bickering about petty stuff – like the color of a bike shed before it's even built. It's a common problem in FOSS projects: You may want some important feedback about a major new feature you're adding, but everyone starts arguing about comparatively trivial things like the design of an icon, or whether it's better to use tabs or spaces in code.

Still, bikeshedding can actually be useful in some cases. If your project has been rather quiet for a while, or a particular mailing list or forum appears to be dead, you can inject some life into it by posing a trivial question – but one for which everyone has an opinion. For example, you could say that you're going to rearrange some of the menus in your app, or tweak some colors, or modify some command-line options.

Even if you have no serious intention to do any of these, you'll still get plenty of opinions on the forum or mailing list. Ideally, this will generate other discussions and activity in your project – and, of course, it makes bystanders and lurkers in the project feel more involved. For more on bikeshedding, check out a fascinating insight from the FreeBSD camp [5].

they also feel like they have an element of control. (See the "Art of Bikeshedding" box for more details.)

And from there onwards, it's up to you. Most FOSS projects take years to reach maturity and develop bustling communities, but with the approach we've outlined here, you have a decent chance of success. There will be problems down the line (see the "Dealing with Trolls" box), and your project may even get forked a few times. But that's often healthy when developers simply have fundamental disagreements and can even bring competition back to stagnant projects.

Above all, make sure you're having fun. If you're not being paid to work on FOSS, you should make sure

burned out; put a deputy project leader in place if you need time off, and above all, enjoy.

Real abuse (threats, harassment, posting personal info, etc.) is much more serious than trolling, and while rare, it can happen. Consider setting a Code of Conduct [6] for your project to establish the behavior you expect from the start. And, if someone violates the guidelines, take the discussion off-list immediately. Talk to the offender personally and try to understand why he/she is acting in that way before taking further steps. Chances are you'll have to ban them from the project – but make sure you hear all sides of an argument first. ■■■

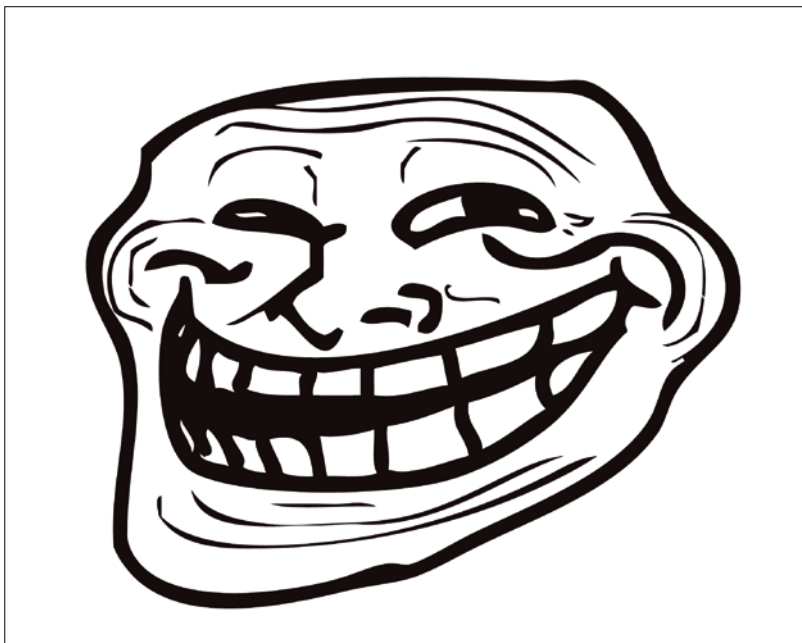
Dealing with Trolls

"Hurr durr, your app sucks and you suck, and you are obviously paid for by Microsoft AND Apple and you are literally Hitler and I'm going to tell everyone not to use your software because it's rubbish." See that? It's a typical comment on the Internet. If you've ever spent any time on YouTube, you'll know that 99.98% of all comments are dribbling nonsense in that vein (but usually much, much worse). Trolls are everywhere – they're part and parcel of the Internet – and in many cases, they're just causing irritation and chaos for their own entertainment.

Anyway, your open source project isn't likely to be swamped by trolls, but a few may rear their heads, especially if your software starts to gain popularity or is perceived as a challenger to an established app. The best thing to do is simply ignore them. Delete their posts, ban their accounts, but don't give them any attention whatsoever. Don't respond at all. Most trolls don't mean any serious harm and when they realize they can't get you in a fluster, they head elsewhere for their "lulz" (Figure 4).

Figure 4: Don't let trolls get to you – they thrive off angry responses. Don't even let them think they're having any effect.

you're getting something valuable out of it for yourself. We often like to think we contribute to FOSS for altruistic reasons – but it's also a way to gain valuable experience (in terms of coding and project management) as well. Don't get



Info

- [1] MikeOS project: <http://mikeos.sourceforge.net>
- [2] Open source subreddit: www.reddit.com/r/opensource/
- [3] Linux subreddit: www.reddit.com/r/opensource/
- [4] Hacker News: <https://news.ycombinator.com>
- [5] Bikeshedding: <http://phk.freebsd.dk/sagas/bikeshed.html>
- [6] Code of Conduct: <http://contributor-covenant.org>

MADDOG'S DOGHOUSE

“maddog” takes a look at various factors that go into creating good code. BY JON “MADDOG” HALL



Jon “maddog” Hall is an author, educator, computer scientist, and free software pioneer who has been a passionate advocate for Linux since 1994 when he first met Linus Torvalds and facilitated the port of Linux to a 64-bit system. He serves as president of Linux International®.

Paid vs. Unpaid

Recently, I was talking with a member of the Free Software community who made a statement that showed disdain for programmers that were paid versus those that volunteered their time. When I questioned this person more closely, he expressed the view that volunteer programmers take more time and are more careful in writing code than paid programmers. “You can see it in their code,” he offered.

I have been in the computer industry since 1969. I have known good programmers, not-so-good programmers, and downright poor programmers, and I really have not seen any correlation between volunteering and doing a very good job in programming. There are more important factors that determine “good code” from “bad code.”

One factor is the programmer’s experience level. Are they just starting and, therefore, do not have all of the necessary skills for writing well-structured code? Perhaps they have not had a lot of practice making their code follow the coding style of the project they are working on – making it look like only one person has written the code, rather than many people.

Do they have a good understanding of the underlying principles of the project, exactly what it is trying to do, and what the parameters of the project are, or are they just trying to put in a patch and not seeing the overall picture?

I will grant that sometimes “paid programmers” are under a managerial time constraint in getting their code submitted to the code pool, and sometimes quality assurance people do not give the testing the coverage it needs due to schedules driven by staffing and time constraints.

Volunteer projects and programmers can also have monetary, staffing, and time constraints. Volunteers have to eat, too, so they typically work at some other “day job,” so their time for programming and testing of their volunteer job can be limited.

The biggest difference, I think, is that volunteer programmers usually work on projects that really interest them, and which they might use in their daily lives. Some paid programmers (not

all) may take a job simply for the money and not even use the software that they are developing. However, this is not a solid indicator of programming quality.

Looking at the history of Free and Open Source Software, however, many people currently paid to work on projects started as volunteers. The core kernel developers are a good example of this. Many started as volunteers, but various companies and agencies rationalized that if they paid the kernel programmers to develop the code, the programmers would be able to work on Free Software many more hours each week and therefore bring desired features out faster. And even when paying these programmers a good salary, it was cheaper overall for the different system vendors (IBM, HP, Dell, etc.) or chip vendors (Intel, AMD, ARM, Motorola, IBM) to have these skilled programmers as paid employees (with health plans) than to develop their own complete operating system groups.

This “paid volunteer development” is not limited to operating systems. Other projects are also sponsored or funded by groups who are making money off Free Software.

Many times, volunteers are also paid because they make their own living off Free Software. They use the software in their day jobs, and by improving the projects they are working on, they make their own day jobs more efficient.

Over the years, there have been many attempts both in the amateur and professional space to improve the code that programmers develop – project reviews, testing grids, language sensitive editors, IDEs, and more – but none of these are real differentiators between the unpaid and paid programmers. I believe the difference lies in each programmer’s skill and work levels, paid or unpaid.

When I started programming in 1969, most programs were distributed in source code, and most programs were written by the people who would later use them. There were few people who programmed for someone else exclusively: the professional programmer.

Right before I graduated from college, a professor said to me, “Jon, you will never be able to earn a living as a professional programmer.” I am still trying to determine if he was right. ■■■

FAQ

Common Crawl

Download the entire web to kick-start a data science empire.

BY BEN EVERARD

Q Is this some new swimming stroke that's all the rage?

A Is that really the best guess you can come up with? The Common Crawl project [1] scrapes the web, sucking up as much information as possible, and makes this data available for anyone who wants to use it. Data is released approximately every month and goes back to 2007.

Q They scrape the web for pages that are accessible to the public and make this data available to the public? What exactly is this meant to achieve?

A Loads of information is available on the web, but there's little structure to it. There's no centralized index – no single place you can go to find out what is on this vast area of cyberspace we call the web. All the average person can do is either click through links and slowly traverse the network of information, or rely on a commercial search engine such as

Google or Bing to help make sense of what's out there.

The Common Crawl project puts the the web into a machine-readable format that allows anyone – whether they're an individual, a startup, or a multinational company – to analyze the information on the web for whatever purpose they want (Figure 1).

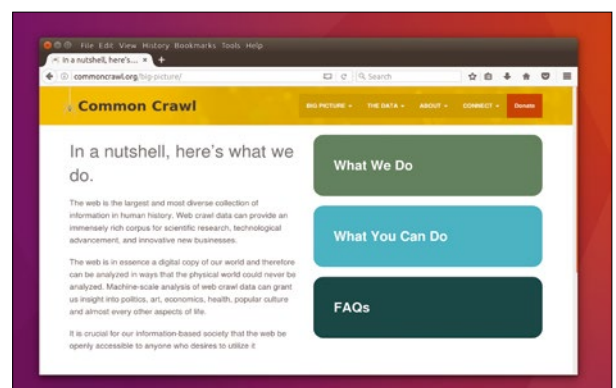
Q So, Common Crawl makes it easy to set up my own competitor to Google?

A Not really. Real-time web searching is hard, and gathering the data needed for the search is one of the simplest parts of the challenge. Common Crawl lets you analyze the information on the web in ways that simply aren't possible with existing commercial tools. Suppose, for example, Graham Morrison, Mike Saunders, Andrew Gregory, and I are in a pub and decide that the person with the most mentions on the web should buy the next round of drinks. How should we go about that? We can search for our names on Google (adding the word Linux to try and weed

out other people of the same name) and the results page tells us that Mike Saunders has "About 15,500 results." That's more than the rest of us, but it doesn't sound like a very scientific answer. Surely Google knows the actual number and doesn't have to hedge. What's more, that only counts the number of pages that includes the result for Mike, not the actual number of mentions of his name. This information simply isn't available from Google.

To persuade Mike that he has the most mentions and therefore has to buy the next round (Mike

Figure 1: Head to the Common Crawl website to find out how to get started with the dataset.



doesn't buy a round without overwhelming evidence), we need to do our own scanning of the web. To do that, we need a dump of the web in a machine-readable format. In other words, we need Common Crawl. With a simple bit of code, we can scan through the data and analyze it in anyway we want, including counting the number of times our names are mentioned.

Obviously, this is an esoteric example, but it's easy to see situations where the information could be more useful. For example, you could use a sentiment analyzer to see how the writing on the web views a particular topic (even using the monthly dumps to see how this has changed over time), or analyze how different websites view different topics, or ... well, you get the idea. It gives you the ability to analyze the information on the web without setting up your own crawling infrastructure.

Q You've made a lot of sweeping statements there about what you can do. I assume you need some technical skills, however. How hard is it to run an analysis on the Common Crawl dataset?

A Surprisingly easy. The data is all stored in WARC format, which was developed for the Internet Archive, but it's a plain-text format that's quite easy to process. There's a library for reading them in Python but if

you prefer another language, you shouldn't have too much trouble getting the data in. Beyond that, it depends on what sort of processing you intend to do. Searching for particular terms is easy; building neural networks to identify complex features of text is harder. The point is, though, that with this data you can focus on the processing and not worry about getting hold of the data in the first place.

Q So there's nothing hard about processing the data at all?

A Well, there is one thing...

Q Go on.

A The full data dump is 250TB.

Q 250 terabytes! How on earth am I supposed to download that, let alone process it.

A Well, that's the uncompressed size, so downloading it is a little easier. It's designed for processing in chunks so you can download a bit, process that, then move on to the next part. On a single computer, this could take a while, but you can speed up matters by spreading the load across machines using something like Hadoop's MapReduce.

Q A minute ago you were telling me this was easy, and now you're telling me that I need to use Hadoop! This doesn't sound easy at all.

A Well, you don't need to use Hadoop, but it can help. It also doesn't need to be hard. The Common Crawl team has put together a tool for launching MapReduce jobs that automatically links in the data. You just have to fill in the details and provide the Python code to process the data. You can see the examples online [2].

Q Well, yes, but you've neatly stepped over the part where you have to set up the Hadoop cluster first.

A Yes, it does need a Hadoop cluster, but this doesn't have to be a pain to set up either. If you don't have machines to run it on, you can spin up machines in the cloud and run it there. The Common Crawl data is hosted in Amazon's S3 storage, and you can link it to their cloud machines without paying for bandwidth. Using Elastic MapReduce and the cc-mrjob, you can automatically spin up a Hadoop cluster using cheap spot instances and process the data using just a single command.

Q Ok, that doesn't sound too bad, but is there anything I can do with this data without creating clusters of machines to download terabytes of data?

A As it happens, yes. The Web Data Commons project [3] analyzes the common crawl dataset and pulls out some useful information (Figure 2). The result is smaller datasets that are more manageable but don't have as much data. For example, there's a dataset of all locations linked to web pages that's 700MB, all calendar events on the web (2GB), all reviews (3GB), and more. Head to the Web Data Commons website to download the files. ■■■

Info

- [1] Common Crawl: www.commoncrawl.org
- [2] MapReduce examples: <https://github.com/commoncrawl/cc-mrjob>
- [3] Web Data Commons: <http://webdatacommons.org/>

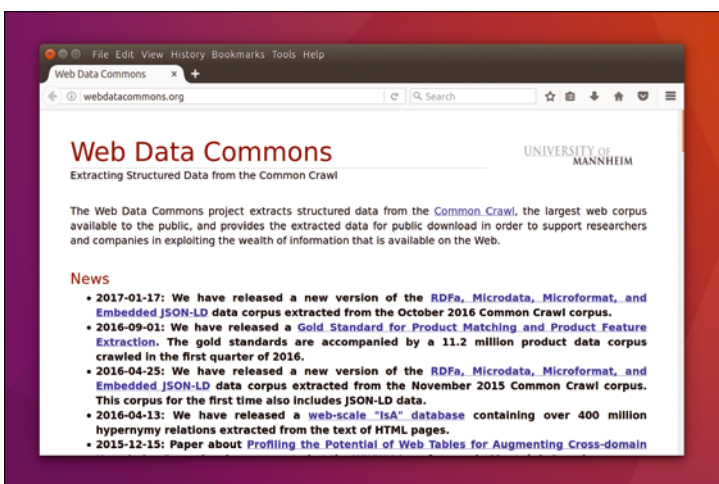


Figure 2: Interested in the data but don't want to deal with the full 250TB of Common Crawl? The Web Data Commons project picks out some highlights.



Valentine Sinitsyn works in a cloud infrastructure team and teaches students completely unrelated subjects. He also has a KDE Developer account that he's never really used.

CORE TECHNOLOGY

Ever wondered what's happening inside a virtual machine? Join us for an exciting tour into virtualization deep waters.

BY VALENTINE SINITSYN

Virtualization Sneak Peek

Today most of us enjoy the benefits of virtualization. With hardware support that finally came to x86 a decade ago and powerful open source hypervisors (also called virtual machine managers), such as VirtualBox or KVM, it's pretty straightforward to run Windows alongside Linux or share a single physical server between a dozen of tenants. In early 2000, to give a new Linux distro a try, you'd install it on a separate hard disk or perhaps run it directly from the CD. Now, you can just boot a virtual machine (VM) from the downloaded ISO image and have fun while reading your friends' tweets.

Yet virtualization internals is still a hairy topic that you may not have a complete picture of. In Linux, Qemu/KVM is the de facto standard tool – or tools (Figure 1). These two have quite a different history but are now seen together more often than not. What's the reason? How does Qemu interact with KVM (and vice versa) to keep your VMs running? In this Core Tech, we'll build a big picture of how an x86 hypervisor operates internally. Qemu, however, is a large and complex project, so we'll dissect a lighter alternative instead: kvmtool. It is simpler yet still real-world and functional, so you may find it useful in your virtualization scenarios.

A Need for Qemu (or Something)

KVM stands for Kernel-based Virtual Machine, so why is there a userspace part in the first place? To answer this question, we need to learn a little bit about how virtualization works in x86.

To virtualize a CPU, you need a mechanism to intercept so-called "control-sensitive instructions," which may affect other VMs running on the same host. Suppose you don't want the guest to access arbitrary I/O ports (this way it can reboot the host) or read arbitrary memory pages, for obvious reasons. Historically, there was no easy way to do this

in x86, but things changed around 2006. At that time, both Intel and AMD announced virtualization extensions to their instruction sets, known as VMX (marketed as VT-x) and SVM (AMD-V), respectively. Although technically incompatible (even though KVM supports both), they are very similar in spirit.

Before this change, x86 CPUs had four privilege rings. Operating systems such as Linux or Windows use only two: Ring 0 to run the kernel and Ring 3 for userspace code. Hardware-assisted virtualization adds another dimension: host (sometimes called "root") mode and guest (non-root) mode. Any instruction that may affect other guests (even if it is not privileged) causes an exit from guest mode to host mode, often called a "VM exit" or "the world switch." This way, a hypervisor can always evaluate the instruction and execute it or inject a fault into the guest. VM exits are expensive in terms of performance, so good hypervisors try to keep their number at a minimum.

The hypervisor's main loop is as follows. First, the hypervisor sets up control structures that tell the CPU which events to trap. They also store the current guest state such as CPU registers. Then, the hypervisor executes a special machine instruction to switch into guest mode. This mode lasts until some event, such as an interrupt, switches the control back to the hypervisor. The hypervisor next analyzes the exit reason, modifies control structures to reflect changes to the guest state, and resumes the guest. That's basically what the KVM kernel module does.

However, Linux never executes guest code: It runs processes. Moreover, you need a way to launch new guests, specifying where their disk images are, how much memory they have, and so on. There is also device emulation: When a guest touches an I/O port that belongs to, say, a PS/2 controller, something should read the register and act accordingly.

The Qemu userspace process handles these tasks. It's an entity that holds the guest code at OS level. When Linux chooses to execute the Qemu process, whatever guest you launched (maybe Windows or Mac OS X) really runs. This means KVM reuses the Linux scheduler, thus confirming its "Kernel-based Virtual Machine" name. When the guest touches an I/O port, the KVM kernel module forwards the request to the Qemu process to emulate. This works on top of the `ioctl(2)` interface, and that's what we are going to examine in a moment (Figure 1).

Meet kvmtool

Qemu is a natural choice for a Linux-based hypervisor userspace component. It can already run unmodified guest OSs, so all the hairy stuff the hypervisor needs to emulate is already here. Put simply, you just want to hook into where Qemu is going to execute a CPU instruction and call KVM for that.

The reality is of course much more complex, and Qemu is a complex piece of software, too. But if you agree not to run anything beyond Linux kernels preconfigured for virtual environments, much of this complexity goes away. Most importantly, you want guest kernels to use virtualized I/O devices instead of real hard disks or network cards. Emulating peripherals is hairy and slow; on the contrary, virtio [1] devices are somewhat like thin wrappers for the ring buffers. This makes virtio devices faster to run and much simpler to implement.

Kvmtool [2] is such a lightweight Linux native KVM tool. It supports Linux guests only, and they must be compiled for the same architecture as the host (so no ARM on x86-64 this time). It emulates a bare minimum of legacy devices (including a real-time clock, a serial port, and a keyboard controller), and that's it. Surprisingly enough, that's the configuration the majority of us run Qemu/KVM anyway.

Born as a hobby tool and an experiment, kvmtool

is now being (slowly) adopted in production.

For example, rkt, a CoreOS application container engine, includes the experimental kvmtool-based stage1 as an alternative to the traditional cgroups/namespaces-based approach [3]. Qemu-less KVM is in fact not exotic: Google also uses a home-grown tool (albeit not kvmtool) in the Google Compute Cloud (Figure 2) for security reasons [4].

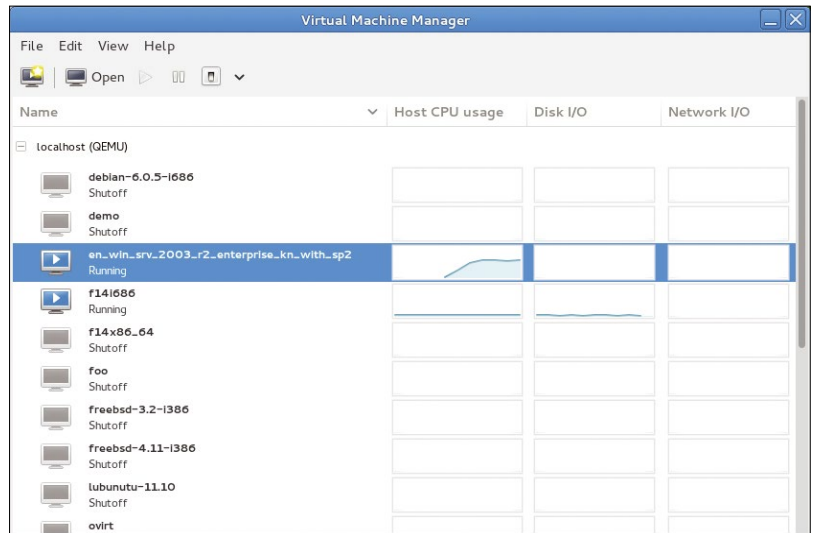


Figure 1: Many of us don't interface with KVM and Qemu directly. Instead, we use libvirt and a graphical front end, such as virt-man-ager. (© Daniel P. Berrangé 2009-2013, licensed under the GNU GPL v3+)

The simplest way to get kvmtool is to clone the official Git repository:

```
git clone git://git.kernel.org/pub/
scm/linux/kernel/git/will/kvmtool.git
```

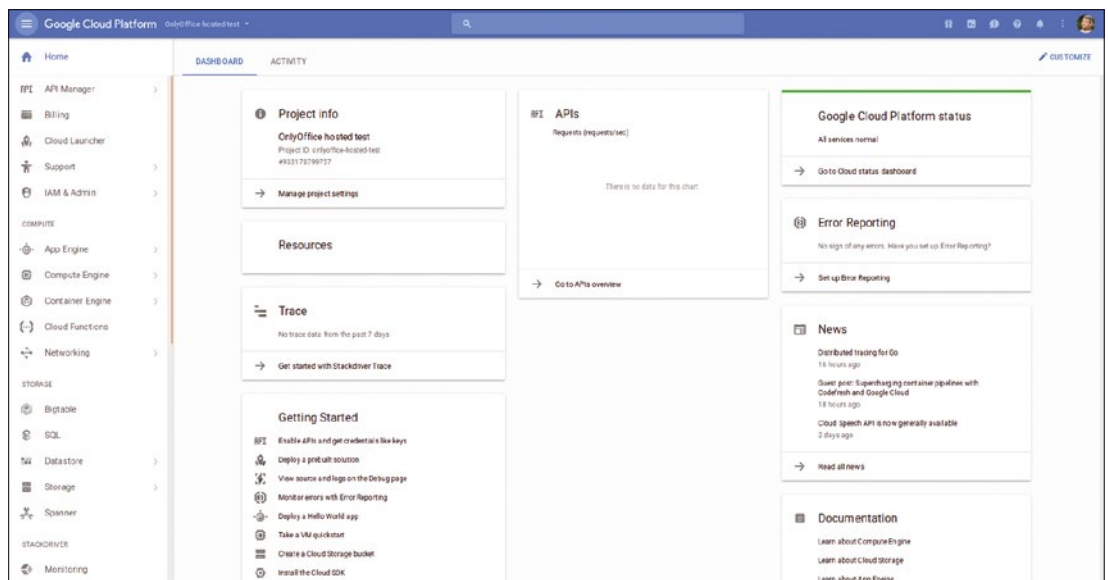
Note that although it lives at *kernel.org*, it's not part of the official Linux kernel (and Linus Torvalds is repeatedly rejecting the idea).

Today's kvmtool is somewhat larger than the initial prototype, which spanned around 5K lines of C code. It's still clean, well structured, and easy to understand, which supports its role as a learning tool. Let's follow the code path that triggers when you run a VM and see what exactly the userspace part does and how it communicates to the kernel KVM module.

Under the Hood

The bottom-up approach seems a natural choice for this task. To run the VM, you issue a run command (see below). Kvmtool implements it in `bu1-`

Figure 2: Google Cloud Platform also runs on KVM but not on Qemu. Sources cite security as the driving force for the latter decision.



Listing 1: kvm__init() Function Snippet

```

01 kvm->sys_fd = open(kvm->cfg.dev, O_RDWR);
02 if (kvm->sys_fd < 0) {
03     /* Error handling is omitted for brevity */
04     ret = -errno;
05     goto err_free;
06 }
07
08 ret = ioctl(kvm->sys_fd, KVM_GET_API_VERSION, 0);
09 if (ret != KVM_API_VERSION) {
10     pr_err("KVM_API_VERSION ioctl");
11     ret = -errno;
12     goto err_sys_fd;
13 }
14
15 kvm->vm_fd = ioctl(kvm->sys_fd, KVM_CREATE_VM, KVM_VM_TYPE);
16 if (kvm->vm_fd < 0) {
17     pr_err("KVM_CREATE_VM ioctl");
18     ret = kvm->vm_fd;
19     goto err_sys_fd;
20 }

```

tin-run.c. A great deal of this file parses command-line options and prepares VM configuration such as the guest RAM size; see the `kvm_cmd_run_init()` function. As a part of this initialization, the `kvm__init()` function is called.

It begins with the opening of the `/dev/kvm` device file. This serves as a gateway between the user-space tool (be it `kvmtool`, `Qemu`, or anything else)

Listing 2: KVM Main Loop (Edited)

```

01 while (cpu->is_running) {
02     if (cpu->paused) {
03         kvm__notify_paused();
04         cpu->paused = 0;
05     }
06     /* Some other checks */
07
08     kvm_cpu__run(cpu);
09
10     switch (cpu->kvm_run->exit_reason) {
11     case KVM_EXIT_UNKNOWN:
12         break;
13     case KVM_EXIT_DEBUG:
14         /* Handle debugging */
15     case KVM_EXIT_IO:
16         /* Handle I/O access */
17     case KVM_EXIT_MMIO:
18         /* Handle memory-mapped I/O */
19     case KVM_EXIT_INTR:
20         /* Handle a signal */
21     case KVM_EXIT_SHUTDOWN:
22         /* Exit the loop */
23     case KVM_EXIT_SYSTEM_EVENT:
24         /* Complain and reboot */
25     }
26 }

```

and the KVM kernel part. `ioctls` are used as the communication mechanism, and you see a couple of them right away in Listing 1.

There, they check if the KVM API the kernel speaks is supported and create a VM for us. The `kvm__init()` function triggers some architecture-specific initialization and creates the guest's RAM. Finally, it loads the kernel image into the guest memory. On bare metal, a bootloader such as GRUB does this, and you can see `kvmtool` emulating the boot protocol for `bzImage` kernels in the `load_bzimage()` function in `x86/kvm.c`. Note that `load_bzimage()` adjusts the instruction pointer (CS:IP) to point just where the real-mode initialization code is in the Linux kernel:

```

kvm->arch.boot_selector = BOOT_LOADER_SELECTOR;
kvm->arch.boot_ip = BOOT_LOADER_IP + 0x200;

```

The next initialization function we encounter is `kvm_cpu__init()`. It creates and initializes all virtual CPUs (vCPUs) the guest runs on. You can pass the exact number as the command-line parameter (see below); otherwise, KVM will supply a sane default. The function issues `KVM_CREATE_VCPU` to allocate and initialize KVM vCPU kernel structures. Then, it maps a read-write memory block backed with the `/dev/kvm` file descriptor. The size of this block is determined with the `KVM_GET_VCPU_MMAP_SIZE` `ioctl`, and the `KVM_RUN` `ioctl` uses it later to exchange data with userspace.

Now, back to `builtin-run.c`. There, `kvm_cmd_run_work()` comes into play. It creates a thread per vCPU with `kvm_cpu_thread()` as the thread function. The latter is a thin wrapper on top of `kvm_cpu_start()`, which implements the KVM "main loop" we discussed.

The first thing `kvm_cpu__start()` does is, unsurprisingly, reset the vCPU. On x86, the majority of registers get all-zeros default values. The instruction pointer and the stack pointer are notable exceptions: They get their "boot values" from `kvm->arch.boot_*`. Additionally, `kvm_cpu__start()` sets Unix signal handlers for the vCPU thread. `kvmtool` employs real-time Unix signals for VM life-cycle management and `SIGUSR1` for debugging, as shown below. The KVM "main loop" (heavily trimmed) is shown in Listing 2.

First, the loop checks for `cpu->paused` and a few other flags. Those are set in the signal handler in response to life-cycle management events. Next, it calls `kvm_cpu__run()`, which translates to the `KVM_RUN` `ioctl`. This is where the magic happens: In the kernel, KVM switches to the guest mode and continues executing as a guest until the next VM exit. When it happens, KVM fills relevant parts of the mapped memory block and returns from the `ioctl`.

The large `switch` that follows handles VM exits. This could be due to a debug-related event such as

breakpoint or because of the I/O. In the latter case, `kvmtool` analyzes the memory block contents to determine which port or memory address was accessed, and whether it was read or write. Then it calls into various parts of the device emulation code (see `hw/` in the sources). `KVM_EXIT_INTR` indicates that there was a signal pending; `KVM_EXIT_SHUTDOWN` means that the guest is shutting down and we want to break the main loop. As for system events, `kvmtool` implements reboots, so the last case is a virtual equivalent of a reset button.

Now you understand what KVM does to run your guest; it's time to see it in action.

Putting It All Together

I assume you have cloned the `kvmtool` Git repository already, so let's go straight to the build process. Alternatively, you can install `kvmtool` from the distribution's repositories with `sudo apt-get install kvmtool` or something similar. `kvmtool` has minimum dependencies. If you need a GUI, as I do in this example, you'd want `SDL` or `GTK+ 3.x` development headers. I opted for the former:

```
sudo apt-get install libsdl-dev
```

Building `kvmtool` is as simple as typing `make`. There are no configuration steps as, again, build dependencies are minimal. Compilation takes a few seconds, and you get a command named `1kvm` as the result. Supposedly, "1" stands for lightweight. The initial name was just `kvm`, but Qemu had the same, so `kvmtool` was renamed. Alternatively, you can run `vm`, which is just an alias.

For usage summary, call `./1kvm --help`. You see it implements various sub-commands, one per `builtin-*.c` file we just dissected. To start a VM, you need `./1kvm run`. This is already a working command that opens a simple shell, with some caveats. It runs the guest on the same kernel as the host, so the image (`/boot/vmlinuz-$version`) must

```
KVM tool
yboard as /devices/platform/i8042/serio1/input/input1
done.
Begin: Will now check root file system ... fsck from util-linux 2.27.1
fsck: error 2 (No such file or directory) while executing fsck.ext2 for /dev/vda
fsck exited with status code 8
done.
Warning: File system check failed but did not detect errors
[ 10.985515] EXT4-fs (vda): mounting ext2 file system using the ext4 subsystem
[ 10.986475] EXT4-fs (vda): warning: mounting unchecked fs, running e2fsck is
recommended
[ 10.999018] EXT4-fs (vda): mounted filesystem without journal. Opts: (null)
done.
Begin: Running /scripts/local-bottom ... done.
Begin: Running /scripts/init-bottom ... done.
mount: mounting /run on /root/run failed: No such file or directory
mount: mounting /sys on /root/sys failed: No such file or directory
[ 11.056876] EXT4-fs (vda): re-mounted. Opts:
Linux version 4.8.0-46-generic (build@1cy01-15) (gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.4) ) #49~16.04.1-Ubuntu SMP Fri Mar 31 14:51:03 UTC 2017
QEMU Linux test distribution (based on Redhat 9)
Type 'exit' to halt the system
SIOCSIFADDR: No such device
eth0: unknown interface: No such device
SIOCADDRT: Network is unreachable
sh-2.05b#
```

Figure 3: `kvmtool` running a guest (here, the same kernel as the host) in an `SDL` window. `GTK+ 3` and `VNC` are also supported.

be readable. On my Ubuntu 16.04, this requires root privileges (hence, you need `sudo`). Moreover, `kvmtool` needs the guest kernel to support certain features (notably, `virtio`), which are often shipped as modules. So, you'd need to tell `1kvm` where the initial RAM drive (`initrd`) is.

Last but not least, `1kvm` needs a disk image to run. The default is a small rootfs, which `1kvm` builds in the `~/1kvm/default` directory in the host and exports as a 9P filesystem (see the "What on Earth Is 9P?" box).

Another option is to use one of the testing images Qemu provides [5]. I'd go this route and download `linux-0.2.img`. With all these bits in place, a complete command to run a VM could be as follows:

```
./1kvm run --kernel 2
/boot/vmlinuz-4.8.0-46-generic --initrd 2
/boot/initrd.img-4.8.0-46-generic 2
--disk /boot/linux-0.2.img --cpus 2 --sdl
```

What on Earth Is 9P?

`kvmtool` understands two types of disk images. It could be a file storing a bitwise copy of a real hard drive or SSD. Or, it could be a directory in the host filesystem exported via 9P.

9P, also known as Plan 9 Filesystem Protocol is a remote resource access protocol. As the name suggests, it originates from the Plan 9 operating system. The latter is sometimes loosely described as "Unix done right." In Unix, everything is a file. Plan 9 takes this concept to a whole new level: Kernel interfaces are files. Windows are files. And files are files, too. No wonder Plan 9 needs a good

protocol to access files over the network. 9P is just that.

9P itself is network-agnostic. The only thing it needs is a reliable, in-order transport. This means messages shouldn't disappear or arrive out of the order in which they were sent. TCP/IP is okay for 9P, as well as shared memory or `virtio` channels. Linux has implemented 9P for a while, so it seems a natural choice for remote file access in `virtio`-enabled VMs.

If you ever used Shared Folders in `VirtualBox`, 9P serves similar purposes, yet it comes with some pedigree.

Note that `lkvm` itself doesn't need superuser privileges, but you'd want to prefix this command with `sudo` if your kernel image or `initrd` requires privileges to read, as explained previously. Here, `--cpus 2` sets the number of vCPUs your guests will have, and `--sd1` tells `kvmtool` to open it in an SDL window (Figure 3).

The command will also print a line on a terminal where you started it, like this:

```
# lkvm run -k /boot/vmlinuz-4.8.0-46-generic -m 320 -c 2 --name guest-4361
```

Take note of `--name`: You'll need it to manage the VM. For example, `./lkvm stop -n guest-4361` will shut down the guest gracefully. Here, the name was autogenerated, but as in this example,

```
lkvm run --name linux-0.2 <other arguments follow>
```

you can also assign a VM something more descriptive. ■■■

Info

- [1] Virtio v1.0: <http://docs.oasis-open.org/virtio/virtio/v1.0/virtio-v1.0.html>
- [2] Kvmtool homepage: <https://git.kernel.org/pub/scm/linux/kernel/git/will/kvmtool.git/about/>
- [3] Running rkt with KVM stage1: <https://coreos.com/rkt/docs/latest/running-kvm-stage1.html>
- [4] 7 ways we harden our KVM hypervisor at Google Cloud: <https://cloudplatform.googleblog.com/2017/01/7-ways-we-harden-our-KVM-hypervisor-at-Google-Cloud-security-in-plaintext.html>
- [5] Testing/System Images at Qemu wiki: http://wiki.qemu-project.org/Testing/System_Images

Command of the Month: lkvm debug

Although `kvmtool` is great for learning, I still tend to run Qemu/KVM in production. Reports are though that `kvmtool` addresses one specific use-case particularly well. I'm talking now about early boot-time debugging.

You can already specify some debugging switches to `lkvm run`. Say, you want to enable single-step mode. Just add `--debug-single-step`, and `kvmtool` will dump the system state (Figure 4) after every machine code instruction. Naturally, that would make a guest really slow.

Then there is a dedicated debugging sub-command named (you guessed it) `debug`. Typically, you supply it a guest name (either the one you assigned with `--name` or autogenerated) and an action to do, like this:

```
./lkvm debug --name guest-4849 --dump
```

This makes `kvmtool` dump the guest system state (again, see Figure 4). Internally, this command sends a `SIGUSR1` signal to the vCPU thread, which causes it to request the guest state (such as registers) from the kernel-side KVM via an `ioctl`.

It is also possible to signal a non-maskable interrupt (NMI) to the guest:

```
./lkvm debug --name guest-4849 --nmi 0
```

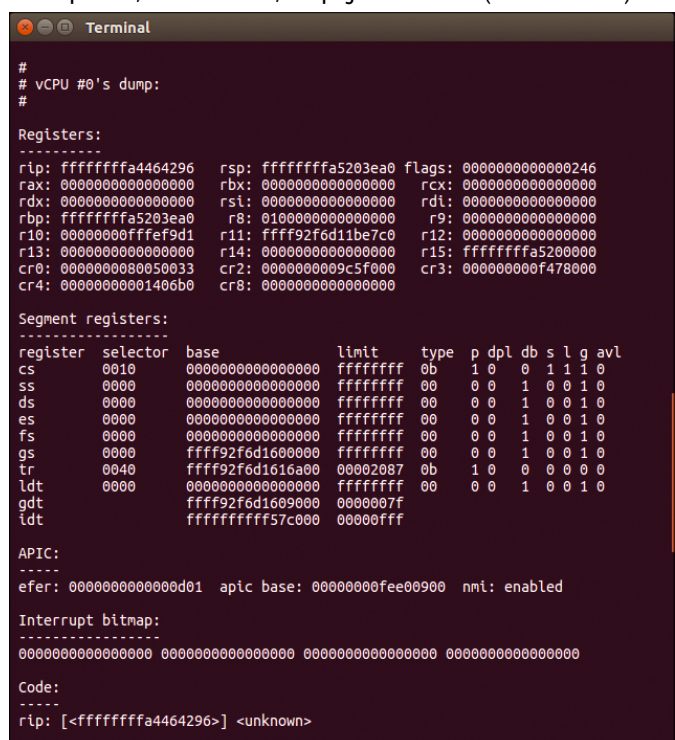
The integer argument is a vCPU number to send an NMI to. Non-maskable interrupts are a serious weapon, and Linux would complain if it came unexpectedly:

```
[ 45.364335] Uhuh. NMI received for unknown reason 20 on CPU 0.
```

```
[ 45.364968] Do you have a strange power saving mode enabled?
[ 45.365467] Dazed and confused, but trying to continue
```

Otherwise, they can be an only option to solicit a feedback from an otherwise irresponsible kernel. ■■■

Figure 4: Kvmtool dumping the guest's state. You get the register values, interrupts info, stack contents, and page table entries (not shown here).



REAL SOLUTIONS FOR REAL NETWORKS

FREE DVD

openSUSE Leap 42.2

Parrot OS 3.5 Full Edition (32-bit)

.NET on Linux

ADMIN

Network & Security

.NET on Linux

Will Microsoft's flagship framework sail without Windows?

ANGULAR 2
Web development framework

TROUBLESHOOTING NETWORKS
Solving problems with DNS, AD, and Group Policy

MICROSOFT OPERATIONS MANAGEMENT SUITE
Monitor Windows systems and Linux servers

Package Security
How Linux distros secure their...

Microsegmentation
Enhanced security with segmented data centers

Optimize PowerShell scripts with loops

INTERVIEW
openSUSE Chairman
Richard Brown

WINDOWS SERVER 2016

- Highly available Hyper-V
- Software-defined networking

FREE CD or DVD in Every Issue!

ADMIN Issue 38 £7.99 38 9 772045 070003

E.COM

Each issue delivers technical solutions to the real-world problems you face every day.

Learn the latest techniques for better:

- network security
- system management
- troubleshooting
- performance tuning
- virtualization
- cloud computing

on Windows, Linux, Solaris, and popular varieties of Unix.

6 issues per year!

ORDER ONLINE AT: shop.linuxnewmedia.com

FOSSPicks

Sparkling gems and new releases from the world of Free and Open Source Software



Graham tears himself away from updating Arch Linux to search for the best new free software. **BY GRAHAM MORRISON**

Development Environment

KDevelop 5.1

We should have covered KDevelop earlier. The major milestone of version 5.0 was passed in 2016, representing the successful completion of a journey for this long-standing programmer's development environment. KDevelop had an auspicious beginning back in 1999. It was the only graphical development environment for Linux that could make sense of the mess of makefile dependencies, and many developers found themselves using it. It was even popular outside of the KDE/Qt community because it used the best parts of the Kate

text editor with Vim-like keybindings, syntax highlighting, and latterly, code completion.

But like KDE itself, KDevelop seemed to be completely rewritten just as one version was becoming stable. The poor reception of KDE 4.0 and the release of the excellent Qt Creator seemed to be double blows, and KDevelop development ground to a halt. Consequently, KDevelop 5.0 is the result of considerable effort over two years to bring the project back, moving the code from KDE 4 to KDE 5 and away from CMake to a new QMake project manager back end. Other

languages and frameworks are also featured, with better support for QML, Python, and PHP, alongside essential updated support for C++ 11 and KDE 5.

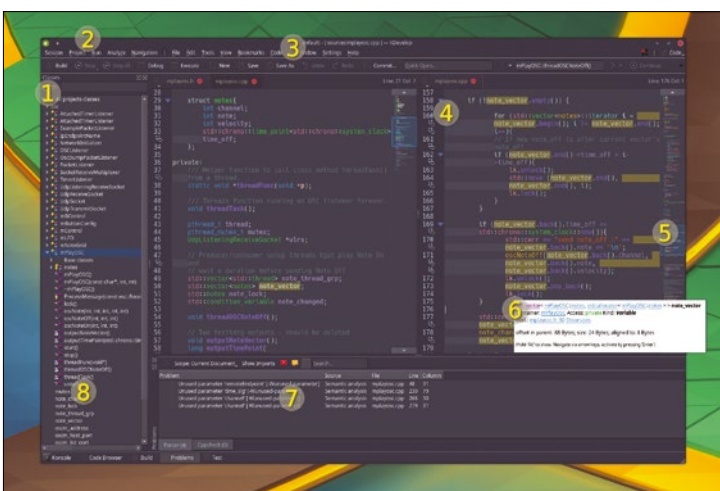
That KDE 5.1 comes only six months after 5.0 is the best possible indication that development is back on track. The move from internal code analysis to Clang in version 5.0 has started to pay off, with the addition of Cppcheck for static analysis of C/C++ code, and the general quality of the syntax highlighting and code parsing is absolutely fantastic. If you're a Python programmer, this new update may also make KDevelop a compelling alternative, with new support for Python 3.6 syntax and semantics. KDevelop is a great option for beginners, too, as obvious mistakes are highlighted and even solutions suggested. It does take up more CPU than Vim, but if you're programming all day, you sometimes need all the help you can get.

This release also adds OpenCL language support. This is the Open Computing Language that's typically used to write code executed on your graphics hardware (GPUs). As GPUs are vastly superior at parallel code execution, OpenCL and its Nvidia counterpart, CUDA, are already having a huge impact when working with Big Data. KDevelop's CUDA support is reportedly incoming, which firmly puts this IDE at the cutting edge of development technologies.

Finally, you can switch color themes on the go, much like you can in applications like digiKam. Appearances are often superficial, but many developers rely on schemes with varying contrast for different times and lighting conditions. The latest version of Qt Creator does have a dark theme, but you still need to edit its CSS if you want to take customization further. KDevelop offers KDE's usual plethora of color options, which are many. This feature may not teach you more about your code, but it may save your eyesight.

Project Website

<https://www.kdevelop.org>



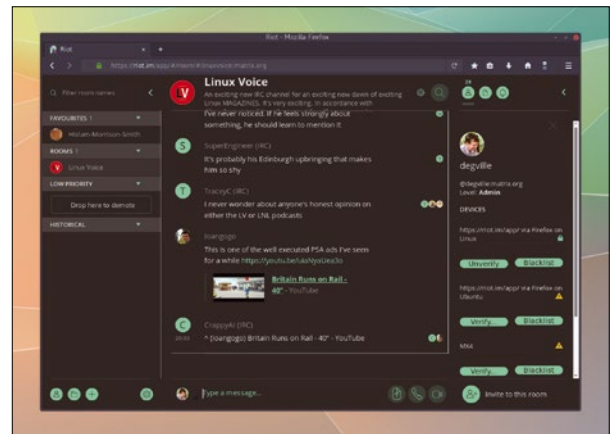
- 1. New project wizard:** Templates for common KDE and Qt apps are included.
- 2. qmake support:** Import qmake-based projects, such as those generated by Qt Creator.
- 3. Layouts:** Split views, add tabs, and change the colors.
- 4. Syntax highlighting:** Even separate variables have their own color.
- 5. Kate preview:** See the entire file with a zoomable thumbnail.
- 6. Object overview:** Hover over your own objects for details on their use.
- 7. Analysis:** Check your code with Cppcheck.
- 8. Project overview:** There are many different ways of viewing your project.

Federated Chat

Riot.im

The latest wave of government attacks on privacy have made the issue of security more pressing, even for people not used to worrying about how their data is stored. As great as the promises are that WhatsApp and Telegram are peer-to-peer encrypted (although not by default), they are not enough. This may be why open source social networking is suddenly gaining momentum. The excellent Twitter-like, Mastodon, has been inundated with new accounts, with the main server even closing to new requests, and the open protocol, Matrix, finally seems to have come of age. Matrix is both a protocol and a non-profit initiative to build both a federated and a persistent communication platform. Because the

data is federated across many open source servers and you can easily run your own, there's no single point of failure and authority with complete control. This openness has led to multiple server implementations, as well as services bridges that link other networks such as IRC, Google Hangouts, SMS, Twitter, Telegram, and even Minecraft, into the Matrix ecosystem. This diversity is equally apparent in the clients that are available, and the most comprehensive is the Riot messenger. Through a browser, Riot feels much like Facebook or the defunct Diaspora. You can join public groups, create private groups (with their own encryption keys), chat with people and paste media such as photos and YouTube links. Riot handles all of this



With Matrix and Riot, you can even bridge to other messaging platforms such as IRC (thanks, ioangogo!).

quickly and feels just like a modern communication platform should. Fundamentally, there are equally mature iOS and Android apps, the latter installable from F-Droid, and these feel equally modern. This is important if we're going to convince our friends and family to switch. As with Linux versus proprietary operating systems in the early 1990s, open source social networking may have found its calling.

Project Website

<https://riot.im/>

Curses Synthesizer

Cursynth

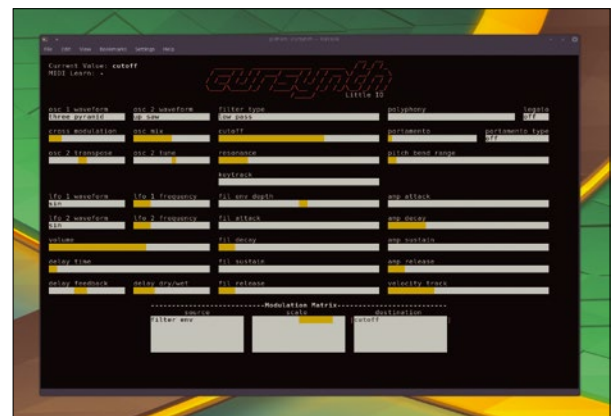
This is another late discovery, and one that hasn't been updated for a couple of years. But it's so impressive, it's worth the coverage in the hope it inspires renewed development from someone. Cursynth is a synthesizer that you run and operate from the command line. But unlike some of the other sound generators you can run from the command line, Cursynth sounds and looks absolutely fantastic, and its choice of UI doesn't inhibit you from accessing some complex editing functions. In fact, thanks to clever use of Curses, the UI could almost be a retro-inspired skin pulled across a VST or Audio Unit instrument. This is perhaps not surprising when you consider its developer – Matt Tytel – is also responsible for my

favorite open source software synthesizer, Helm. Cursynth is a precursor to Helm in many ways and features a similar analog-style sound. It has a simple classic design that belies the quality and breadth of its output. There are two oscillators for sound generation, a characterful low pass or high pass filter, mono or polyphonic options, and two envelopes. There's even a delay effect. But, most impressively, there's a modulation matrix that allows you to modulate a destination from a source with a varying amount of scale. This is an advanced feature that Helm also includes, and even though you're navigating the UI with the cursor keys, it's still easy to set up a source and destination, and because everything can be played

from your QWERTY keyboard, as well as MIDI, it's immediate and responsive. From MIDI, you can also remap controllers to the various on-screen controls, so you don't have to cursor around. That does slightly diminish the fun of playing a synth on the command line, however.

Project Website

<https://github.com/mtytel/cursynth>



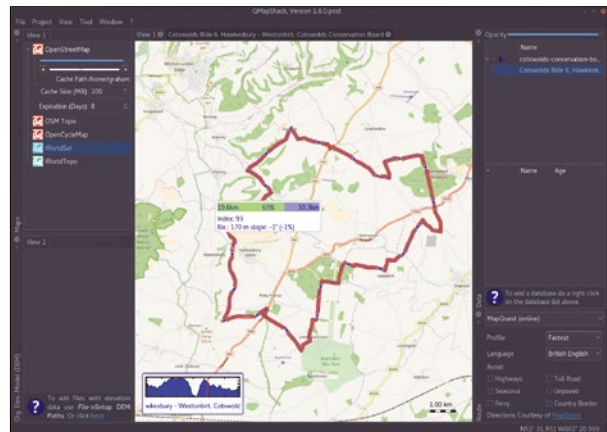
You can now play a remote synthesizer through an SSH terminal.

Route Planning

QMapShack 1.8

OpenStreetMap is a wonderful exception in an area that's blighted by proprietary software and content. In the UK, even the tax payer funded "Ordnance Survey" locks down its content behind paywalls and restrictive use clauses, which is a real lost opportunity for open source-like innovation. OpenStreetMap has helped fill such gaps in the UK and all over the world, helping people in crisis, such as the Humanitarian OpenStreetMap team in Haiti, as well as people lost in the mountains. The data is always growing in depth, accuracy, and influence. Open source apps, such as Osmand and QMapShack, are a brilliant way of making use of all this data, allowing you to follow routes or find where you are.

Unlike Osmand on Android, QMapShack is built for the desktop and is perfect for editing routes and diving into the details of what a map offers. It was never built specifically for OpenStreetMap data, being a sequel of sorts to QLandkarte, which was built primarily to access Garmin maps, but it supports OSM natively, along with various other map formats. When first launched, QMapShack guides you through getting access to these maps as soon as possible, caching maps locally and easily from OpenStreetMap. You can even search Google and see any resultant location in the main view. QMapShack's great strength, however, is in editing and creating routes. You can import a GPX file from Osmand, for



Thanks to TwistedLucidity on IRC for giving us the heads-up on this excellent application.

instance, and study the height contours or edit the way-points in a fast, responsive user interface that's much easier to use than a web portal. It's exactly the kind of third-party tool OpenStreetMap is missing on the Linux desktop and is the best way I've found to prepare a route for a bike ride or a walk in the hills – exactly the kind of application you'd be expected to pay for on other platforms.

Project Website

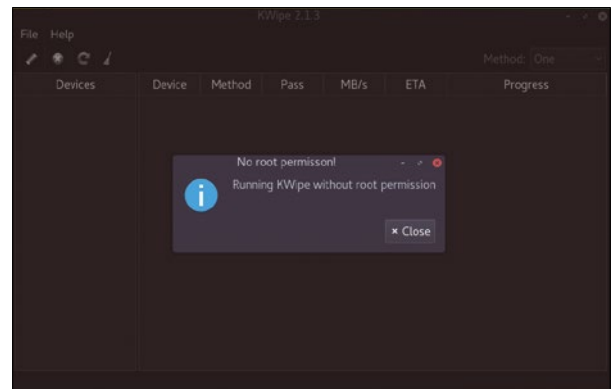
<https://bitbucket.org/maproom/qmapshack>

Secure Erase

KWipe 2.1.3

A couple of caveats are needed before looking closer at this otherwise excellent tool. The first is that this is an application that requires root and wants to help you securely erase data. This is a noble and worthwhile endeavor. But, and this is important, there are no obviously official download sites, no package signing, no checksums for the download, and no third-party security audits. By running KWipe, you're running a Python script with root access and then asking it to clean your partitions of recoverable data. On the positive side, the code is Python and easy to read, so you can check to make sure it's doing what you hope it's doing, and it certainly offered no surprises when I used it.

KWipe is different from deleting your files in the usual way because it implements several different methods for scrubbing your drive/SSD of any last vestiges of a file. These methods have names like One, Zero, Gutman, Bruce Schneier Algorithm, British HMG Standard 5, Russian GOST p50739-95, and NSA 130-2. The way these work is well known and studied, and KWipe couldn't be easier to use. A list of devices and partitions appear on the left, and you select your target and a method before selecting the *Erase* button. It's about as simple an application as you could make, but the background process is multithreaded, which helps improve the speed, and you don't really need many more options. If you want to check the



Used with caution, KWipe is an excellent tool for making sure your deleted data stays deleted.

veracity of the erasure, you need to use another tool, such as Testdisk. KWipe performs an important job, but it should only be used with caution. I'd recommend only when donating/selling drives, and from a Live CD on a machine with nothing else. If this is something you do all day, KWipe is going to make your life easier.

Project Website

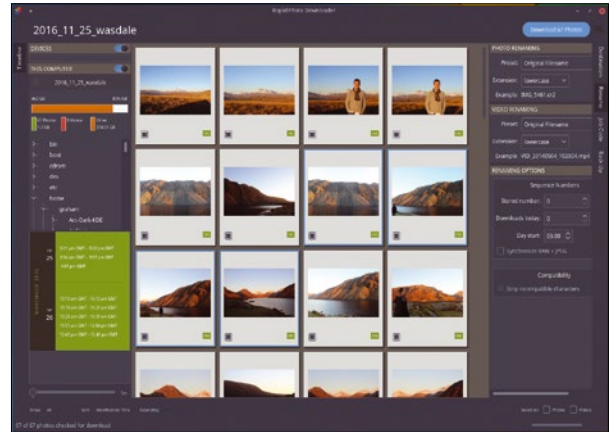
<http://www.2blabla.ch/>

Photo Importer

Rapid Photo Downloader 0.9

The RAW photo processor, Darktable, has become an integral part of many photographer workflows. It's brilliant at improving an image or recovering an image from unexposed/overexposed disaster. But, it's not great at photo management or at importing photos from various locations. This has obviously not been an emphasis for Darktable, and with good reason – Linux needed an Adobe Lightroom replacement much more than it did another photo album generator. This also means that many users resort to other applications, such as digiKam, to handle their photo import and management duties, keeping them from spending more time in Darktable. Rapid Photo Downloader is perhaps the best possible solution to this, allowing you to avoid a second or third photo application

while not having to rely on Darktable for import and management. The tool's name tells you everything you need to know. With Rapid Photo Downloader installed, you can quickly import hundreds of photos off your camera or memory card and into organized folders on your local drive. This is in complete contrast to many other applications that often leave you enough time to make lunch and brew a cup of tea during the import process. The main window shows good-sized thumbnails of even large RAW files quickly and makes their formats easily discernible. There's no editing or processing or larger previews, which is perfect for this simple process. The clever part is that each import job can be customized by tagging photos with "job codes," which can then be used to define subfolder and file



Importing photos from external devices can be automated by enabling the option in the preferences panel.

names during a renaming process. It's quick to set up and does exactly what many users need when juggling hundreds of photos on an SD card – organize new photos into folders by date and a descriptive tag or two – essential if you're doing any serious photography with Linux.

Project Website

<http://www.damonlynch.net/rapid/>

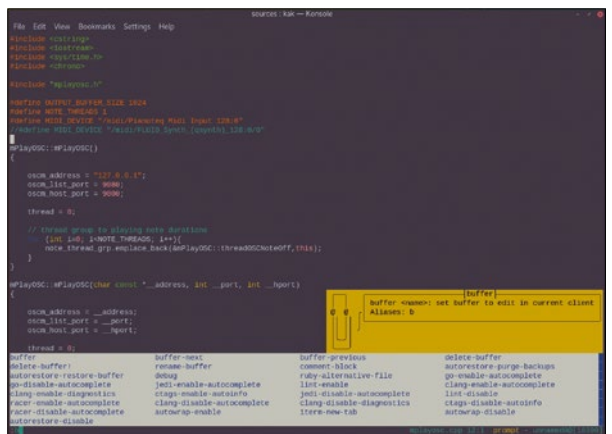
Code Editor

Kakoune

Vim, the text editor, is seriously powerful. It's the tool many of us prefer to use on a day-to-day basis for both text work and code. The lexicon of its keyboard shortcuts has become a universal language, and its philosophy has spread far and wide. The same can also be said of Emacs, of course, and like Emacs, there are Vim plugins that can do almost anything. But as soon as you start installing and customizing these plugins for your own use, you tread further away from the common path.

Kakoune is a code editor that takes plenty of inspiration from Vim but is forging its own popular path. It has the same move-

ment, the same Normal and Insert modes, and a very similar appearance. It also adds some excellent well thought-out features, and some you can't easily replicate in Vim. At the top of this list is the idea of a *selection*. To Kakoune, a selection is "an inclusive, directed range of characters with two ends, the anchor and the cursor." The reason this is important is because you can have more than one selection at a time and then process those selections easily, whether that's aligning columns or splitting words. It's a little like a regex search in Vim, except it feels much more natural. The auto-completion and hinting are well implemented, showing the pos-



Importing photos from external devices can be automated by enabling the option in the preferences panel.

sible options for a command when you start typing, and it makes you realize just how little feedback or help Vim gives you. If you're a beginner, Kakoune is easier to learn than Vim, but it's Vim users who will most appreciate the excellent selection options and feedback

Project Website

<http://kakoune.org/>

VST Preset Randomize

VPG 0.2.8

If you're interested in audio and sound, the most significant news of recent months was that Steinberg, the German proprietary software company behind Cubase, has released its VST3 SDK for Linux with a GPLv3 license. Cubase changed the music industry, and its original VST API gave Steinberg and third-party developers equal footing when it came to developing audio plugins. These plugins have since taken over the world of audio production, much like the idea behind Photoshop filters, from the recreations of classic and costly synthesizers to the invention of Auto-tune. But VST support for Linux has only ever been semi-official. Earlier APIs could be included but were incomplete, missing the GUI, and certainly without the blessing

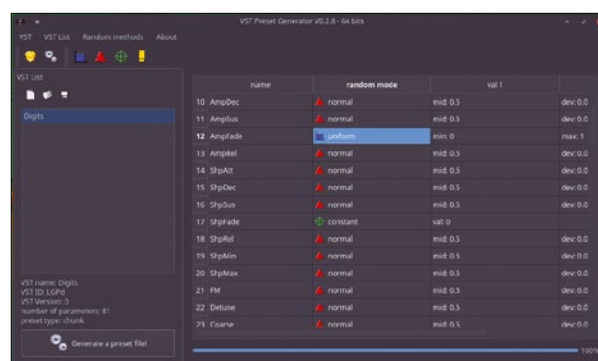
of a Libre license. This meant Linux VST adoption has been slow, with only very few plugin developers offering Linux versions, and fewer using Wine to host the Windows versions.

With the new API, VPG (the VST Preset Generator) should have a very healthy future. It's a simple application, but it's especially useful on Linux where there's a lack of anything designed to work with VST plugins specifically. All VPG does is load the plugin file for the instrument or effect and then parses this file for the parameters it uses to process sounds. You can then randomly generate new presets for the plugin and either save this as a VST formatted bank or individual preset. It's great for experimentation, especially with complex synthesizers like the Digits

emulation of the Casio CZ where you can set different randomization modes for different parameters. Considering that Linux doesn't currently have much of a preset library it can call its own yet, this is a great way of getting started, and a great way of adding an experimental edge to some of the effects and instruments without tinkering with any values.

Project Website

<http://vst-preset-generator.org/>



Add random elements to your sounds with a random patch generator.

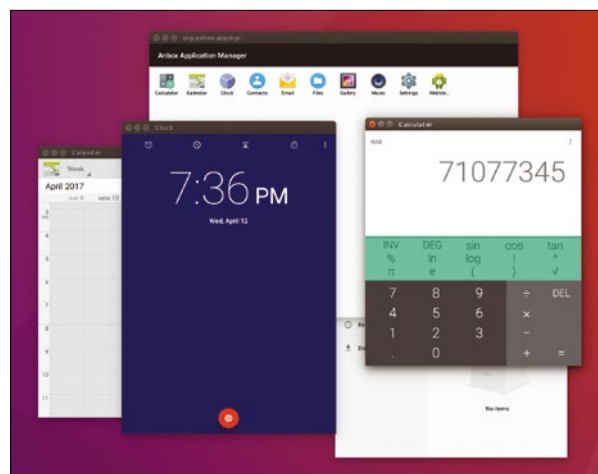
Android Emulator

Anbox (alpha)

One reason why it's easy to forget that Android is Linux is that there's so little convergence between the Linux desktop and the Android mobile experience. Even Chrome OS running on many Chromebooks is unable to access Android applications without some official blessing from Google. This is a shame, because there are many apps on Android that work well with a mouse and keyboard and that don't exist on the Linux desktop. Anbox – Android in a box – is a new and potentially brilliant solution. It's still in the early alpha stages of development, but it's showing real promise already.

The only requirement for installing Anbox is that your distribution supports the snap

package format. Fortunately, snaps are now supported by the vast majority of distributions, including Ubuntu, Arch, Fedora, Debian, and SUSE. The snap requirement isn't just because it's a rather neat way of installing new applications, but because container isolation with LXC is an integral part of its function, and this is exactly how Anbox isolates Android from the remainder of your system. With Anbox installed, Android can be booted into its own window, and any apps from there are booted into further containerized windows. The version of Android that's used is purely open source, which means you'll need to find some way of adding Google Play or Google's Apps if you want to take the experiment further. This is no different from



Like many unofficial Android bundles, Anbox can't include Google Play, but it can be installed manually.

other un-Google distributions of Android, such as AOSP on your Android phone, and shouldn't be too difficult if you're happy using a few Android debug commands to copy a few important package files over. It's definitely going to be worth the effort, especially if there are proprietary Android apps that don't have Linux equivalents.

Project Website

<http://anbox.io/>

Voxel World

Terasology Alpha 7

Notch, the creator of Minecraft, once promised to release his incredible game under an open source license. This promise, quite understandably, has been lost among the billions of dollars and the Microsoft buyout. But, remarkably, it doesn't matter too much. This is thanks to several significant open source projects that implement voxel graphics and similar crafting and survival mechanics to Minecraft. Terasology is one of the best. This is alpha release 7, and the result of a lot of work from a Google code-in over winter and a "Summer of Code." And, despite its alpha tag, Terasology is already well established and comprehensively furnished with assets, levels, and

modes. It's also great fun to play, especially if you already enjoy and understand how Minecraft works. Its development plan is also ambitious by extending the original idea with caretaker management styles inspired by Dwarf Fortress and Dungeon Keeper.

The only slight hitch, and one Minecraft also suffered from, is that Terasology needs Java to run. Thankfully, this isn't much of a problem if you already have Java installed, and I had no problem playing the game with OpenJDK 1.8.0. The first thing you notice is that the game looks staggeringly beautiful. If your graphics hardware is up to the task, there's depth of field and bloom effects that really bring the voxels to life. This is



Terasology is a rare exception in open source gaming where the art assets are also open source (CC BY 4.0).

thanks to the huge number of creators and maintainers who have contributed already. There are also dozens of add-on modules, too, adding everything from dynamic cities to different soil types – none of which you expect when you read this is an alpha. If you know someone who loves Minecraft, try switching it one night with Terasology and see if they thank you.

Project Website

<https://github.com/MovingBlocks/Terasology>

MUD Client

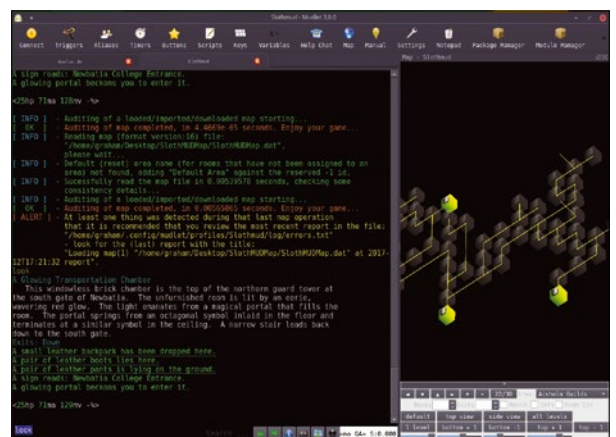
Mudlet 3.0

In the early 1990s, I spent some time at a college in the United States, and there were two brilliant things about being there at that time. The first was that the computer lab was open 24 hours a day, every day. The second was that local phone calls were free. This meant I could use a borrowed Amiga 1000 with a modem and the old "Term" application to dial into the lab computers and stay connected for as long as needed, all from my dorm room. I barely needed to get out of bed. This was, of course, a good way to work, but there was something far more compelling than my Modula-2 assignments, and that was playing MUD games.

The MUD genre, an acronym for Multi User Dungeon, is a combination of text-based interactive

fiction with very early multi-user game mechanics, similar to the kind you now find in online role-playing games. They were popular at a time when text was all you could really transfer across a network in close to real time, and when text was all screens were good at displaying. Entering a MUD was a little like playing Dungeons & Dragons, where the dungeon master was the parser, describing the location, local items, and objects while allowing you to do things that might affect gameplay. The only difference is that you're not alone, and that specific instance of your universe is usually persistent. The lamp stays smashed until the server is reset.

Mudlet is the result of four years work and is a simple yet powerful portal into this MUD



SlothMUD is 22 years old and is probably the most active MUD still running, maybe thanks to its 27,500 rooms and 9,300 creatures.

world. A simple click will give you immediate access to many of the still-popular MUDs people play, and serious players can augment their typing with mapping scripts and 3D maps, automation, notes, and many other advanced features. It's brilliant, it's easy to use, and if you've not tried playing a MUD for a while, it's also the perfect excuse to revisit a wonderful game type.

Project Website

<http://www.mudlet.org/>

Systemd Services 101

Take control of the services running on your Linux machine

BY BEN EVERARD

Systemd is the init system of almost all modern Linux distributions. It's the first process to start when Linux boots up, and it controls everything else that runs on your computer. Much of this happens automatically, and you should never have to think about it, but there are some bits that you may wish to poke around from time to time.

Before we get too far in, the first thing to do is check if you're using systemd. You can do so with the command:

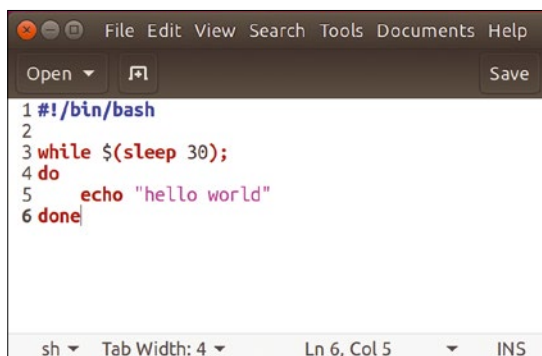
```
pidof systemd
```

If this returns a number, then you're running on a systemd distro, if there's no output, then you're using some other init system. To follow along with this tutorial, you'll need systemd, so you may want to download a live distro such as Ubuntu and run it in a virtual environment.

I mentioned that systemd controls everything that runs on your computer. It does this through services, which are normal bits of software that systemd starts automatically at a predetermined point. For example, if you're using a desktop environment, systemd will launch your login manager once the graphical system has started. This login manager starts your desktop and your desktop starts your graphical software. You can see how everything branches out of systemd with the command:

```
pstree -p
```

Figure 1: System services don't have to be complex, and ours is only six lines long.



```
1 #!/bin/bash
2
3 while $(sleep 30);
4 do
5   echo "hello world"
6 done
```

Systemd decides what to launch by looking at the service files. If you're familiar with the old style SysVinit, these service files take the place of the init.d scripts, but they're a very different syntax.

Let's look at this by creating our own service. In general, services are bits of software that run indefinitely, and ours will be no different. It's a hello world service that just outputs "hello world"

every 30 seconds (Figure 1). Create a text file called `hws.sh` in your home directory with the following:

```
#!/bin/bash

while $(sleep 30);
do
    echo "hello world"
done
```

Make this file executable with:

```
chmod a+x hws.sh
```

You can now run this from the command line with `./hws.sh`, but that's not running it as a service. Now let's look at how to get systemd to start this automatically every time you start your computer.

The service files are typically split into two sections, Unit and Service. The Unit section has the basic information about what this file is for and when to run it. The Service section contains details about what actually we want to run. A basic service file for our Hello World Service is:

```
[Unit]
Description=Hello world service
After=systemd-user-sessions.service

[Service]
Type=simple
ExecStart=/home/ben/hws.sh
```

As you can see, this contains a description that can be anything you want. The After line tells systemd when to start this service by telling it what needs to be started before this one can. We're just waiting for the user sessions to start, although really, this could be started earlier if desired.

Setting `Type` to `simple` tells systemd that this is a command that will continue to run in the session it was started. The alternative here is `Type=forking`, which is for commands that handle their own daemonization. You can think of this in terms of running something from the command line. If, when you run a command, it continues to send output to the terminal, it's a simple type, whereas if the commands

kick of some background processes and return the command prompt, then it's a forking type. The final line here is just the command to run.

Save this in a file called `hws.service` (the filename is important in this case), and then copy this file to `/etc/systemd/system` (you may need to use `sudo`). Once this is in place, you can start the service with:

```
systemctl start hws.service
```

Once that's finished, your hello world service is running in the background greeting the rest of the planet. All the output from processes started like this is gobbled by another part of systemd: `journald`. You can view this with the command:

```
systemctl status hws
```

This will show you the last few log lines as well as information about the processes running. You can also view the entire log with:

```
journalctl -u hws -e
```

The `-u` flag tells `journalctl` which unit (that is, service file) to show the output of, and the `-e` flag means start from the end of the file and work backwards. By default, this will show the log in a scrollable environment, but if you pipe the command into something else (e.g., `grep`), then it will send the text to `stdout`.

Most distributions also send the output from systemd services to the `syslog`. This is an option in systemd rather than a fixed feature, and we may find that over time, this happens in fewer and fewer distros as people become used to working with `journalctl` rather than the logfiles.

Let's take a look at these `journald` lines. You should have seen something like this:

```
Apr 24 20:02:52 ben-desktop
hws.sh[3790]: hello world
Apr 24 20:03:22 ben-desktop
hws.sh[3790]: hello world
```

Obviously, the first section is the date. The next item in the line is the machine that this happens on. Although it might initially seem obvious which machine it's running on, this isn't always the case as logs are often amalgamated into a centralized logging system. The next bit is the name of the software that's running and the process identifier. The final bit is the actual output.

Most of this output looks quite good, but the `hws.sh` part isn't very clear. We could rename the file that runs, but we don't need to. Instead, we can just tell systemd to use something else in the logging. This is the `syslog` identifier. If you include the following line in the `Service` section of the `hws.service` file, it will replace `hws.sh` with something more meaningful:

Systemd Timers

In this tutorial, we've looked at services that start and then continue to run. However, there's another type of background program: scheduled tasks. These we want to run periodically rather than constantly. You can use `cron`, but systemd provides tools for this. First, you need to create a service file as we have done in the main tutorial; then you need a timer file, which should have the same name as the service file but with the `.timer` ending. This is saved in the same directory as the service file. A simple Timer file that would run our

hello world service (even if it stopped running or didn't have a loop to continue to run) is:

```
[Unit]
Description=Run HWS 30 mins
after booting then every week
```

```
[Timer]
OnBootSec=30min
OnUnitActiveSec=1w
```

```
[Install]
WantedBy=timers.target
```

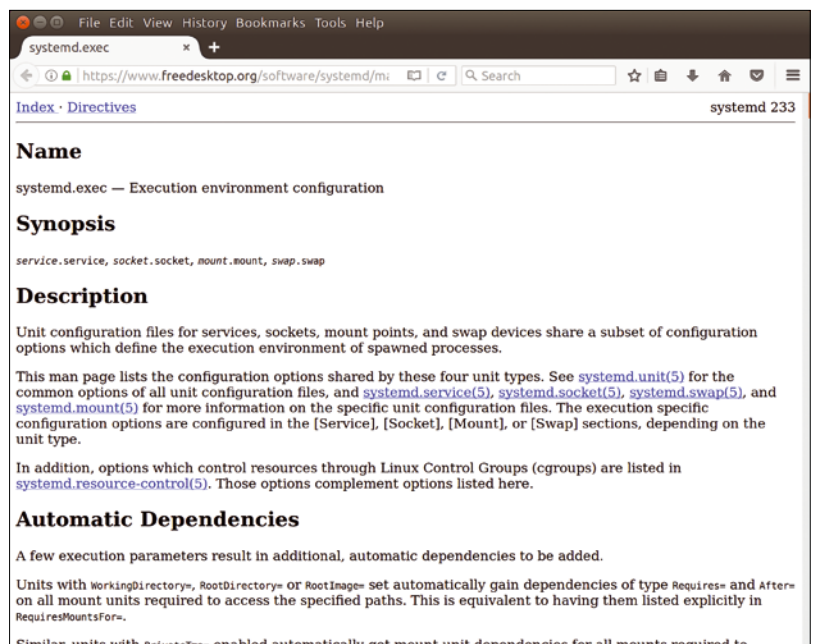
As you can see, you can time relative to booting as well as on pre-defined periods.

```
SyslogIdentifier=HelloWorldService
```

We've only looked at the most basic options here, and there are many more advanced features. Most of them are as simple as setting options in the service file, and the process is well documented in the `systemd.exec` man page (Figure 2). See the "Systemd Timers" box for more information.

Systemd has made it much easier to create services, especially for people not skilled in the dark art of Bash scripting. As you've seen, a simple service file isn't scary; it's just a place to tell systemd what you want to do. You should be able to take the skills we've covered in this tutorial and apply them to running almost any software as a service. Don't fear the daemon. ■■■

Figure 2: You can find out how to extend your service files to take advantage of the full range of options on the `systemd.exec` man page.



Dive into LibreOffice: Hidden Gems

Discover some hidden and lesser known features in LibreOffice, to help you work faster and smarter (and gain extra geek points).

BY MIKE SAUNDERS

If you've been following the development of LibreOffice since it was forked from OpenOffice.org in 2010, you'll know that a great deal of effort has been put into cleaning up the source code. This has made the suite easier to compile, and, more importantly, it's easier for new contributors to get involved. Along the way, new features have been added and Microsoft Office compatibility has been improved, but by and large the software doesn't look or feel drastically different in 2017.

That's a good thing, of course – office suite users tend to value consistency and stability rather than shiny new widgets with every release. But there are some changes and new features that could really take LibreOffice up a level, and although they aren't available out of the box, with a bit of tweaking you can try them out. In this article, we'll show you some tips and tricks to get more out of LibreOffice, using features you may not have come across before.

Try a Tasty MUFFIN

In recent years, some LibreOffice users have been asking for a more “modern” user interface alternative to the regular toolbar layout. However, implementing such a change is no easy task. The LibreOffice design community [1] has to constantly juggle the demands of various types of users, some of whom like things exactly as they are and resist any change, and others who want the software to look and feel more like Microsoft Office. Whatever the design team does, someone will be unhappy.

That means lots of things need to be considered when making large-scale changes to the GUI, potentially breaking many workflows. In December 2016, The Document Foundation (the non-profit entity behind LibreOffice) announced MUFFIN, the “My User Friendly and Flexible Interface” [2]. This concept will be gradually introduced in forthcoming versions of the suite, providing users with multiple GUI layout options:

- The standard layout with two toolbars at the top
- A single toolbar mode to save screen space

- A single toolbar with a pop-out sidebar to provide extra functionality

- And, the new NotebookBar

You may be thinking at this point: why so many options? Couldn't the design team just focus on one layout and make it perfect? Well, that would be one approach, but it risks alienating a big chunk of users. The LibreOffice design community has to keep long-time users (who want to stick with regular toolbar layouts) satisfied, while also enticing new users who want a more “modern” design. The Document Foundation is financed largely by donations from end users, so it makes sense to keep as many as possible happy, even if the end result is a bunch of interface options.

One of those options, as mentioned above, is the NotebookBar. This made its first appearance in LibreOffice 5.3, but even if you're running that version of the suite, you may not have tried it yet (more on that in a moment). The NotebookBar was developed by the design community as a way to “evolve past the restrictions of toolbars” [3]. Essentially, it's a large toolbar where designers have the freedom to do anything they want with the space. To quote the design team:

“With a blank canvas, a designer can place any UI widget on it, including the usual buttons with or without a label, a section label to identify the group of controls, or more advanced widgets like tabs. They also have the ability to define any dimension for buttons, so they serve as a visual attractor, and all together have a larger catalog of controls to choose from tha[t] couldn't be found in classic toolbars. Furthermore, they can define that the main menu can be hidden, or whether it should have a particular icon theme, for instance.”

What's especially interesting here is the use of contextual groups. This means that some controls and sections of the NotebookBar are disabled when you're doing certain tasks. For example, you probably always want access to buttons for loading and saving files, and performing undo and redo operations, but you don't need immediate access

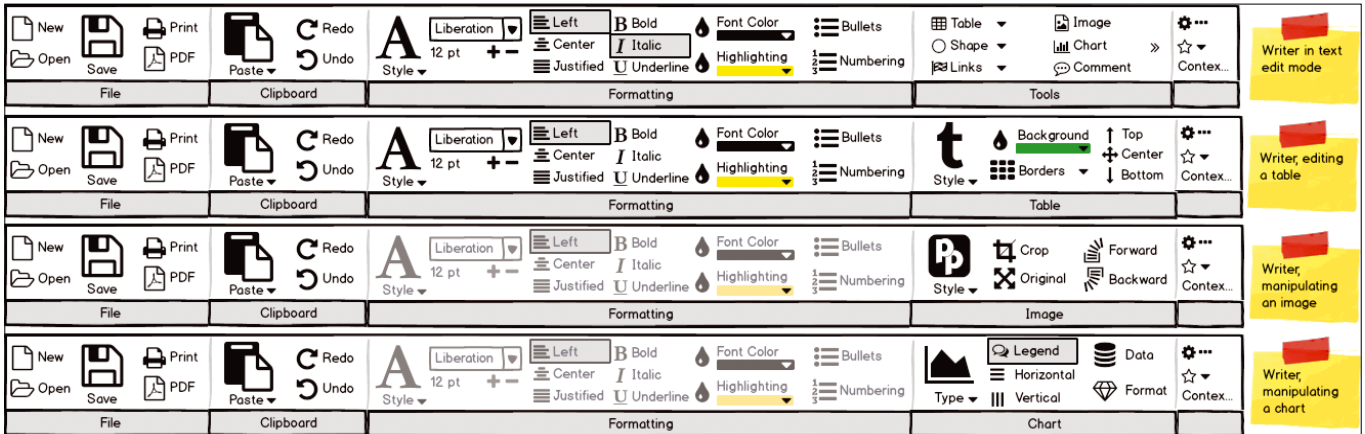


Figure 1: The LibreOffice NotebookBar came to life following many discussions and mock-ups from the design community.

to cropping and rotation operations unless you've clicked on an image. To see how this works, see the design team's mock-up in Figure 1.

Give It a Go

That's enough background – here's how to try it. The NotebookBar is included in LibreOffice 5.3, but because it's very much an experimental feature and not recommended for production use, you won't find it in the menus or options dialog. To make it available, go to *Tools | Options* in the menu, and then *Advanced* on the left-hand side. Check the *Enable experimental features (may be unstable)* box and click OK – you'll be prompted to restart LibreOffice.

After that, go to *View | Toolbar Layout* in the menu and choose *NotebookBar*. Et voilà, you'll see the shiny new interface like shown in Figure 2. Immediately, you'll notice that the NotebookBar contains several tabs along the top to switch between various categories. Click a tab and the buttons underneath change accordingly – and these buttons are grouped into categories as well. It might look a little odd compared to the traditional toolbar approach, and you may notice some glitches in places (e.g., missing icons), but it is very much an experimental feature as mentioned.

Now, there are some alternative layouts for the NotebookBar as well. Click on the document icon in the top left and choose *MenuBar* to enable the menu. Then go to *View | NotebookBar* and try one of the other designs – *Contextual groups* and *Contextual single*. If you choose the former, try inserting an image; when you then click on the image, you'll see that the NotebookBar's design changes to offer functionality relating to images. Type some text, though, and the NotebookBar adapts accordingly. (See Figure 3 for an example.)

You can now spend some time exploring the NotebookBar and its various layouts, but if you want to return to the normal GUI, click *View | Toolbar Layout | Default*. As we speak, the LibreOffice design team is working on updates and refinements to the

NotebookBar, and it may even become a standard option (i.e., not just experimental) in the next version of the suite, 5.4, which is due in late July or early August 2017. If you're a big fan of the traditional GUI and the NotebookBar doesn't float your boat, worry not – there are no plans to remove the standard toolbar layout. For more information on

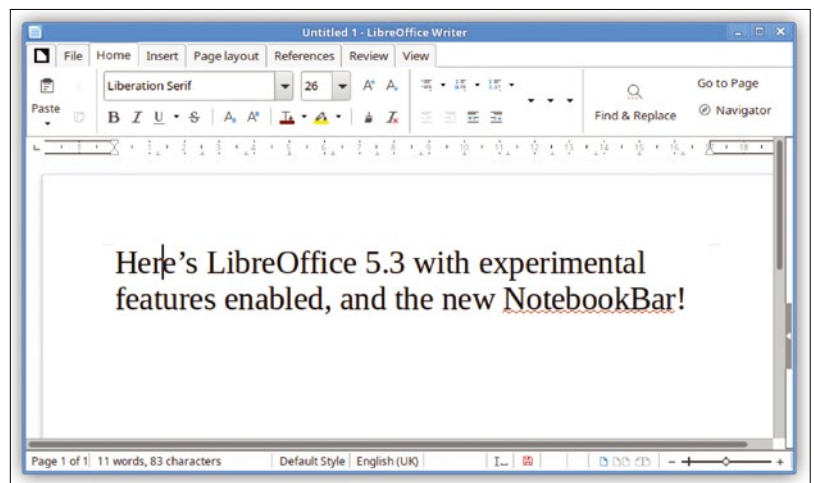


Figure 2: This shows the NotebookBar in its tabbed view – click the document icon in the top-left to access the menu.

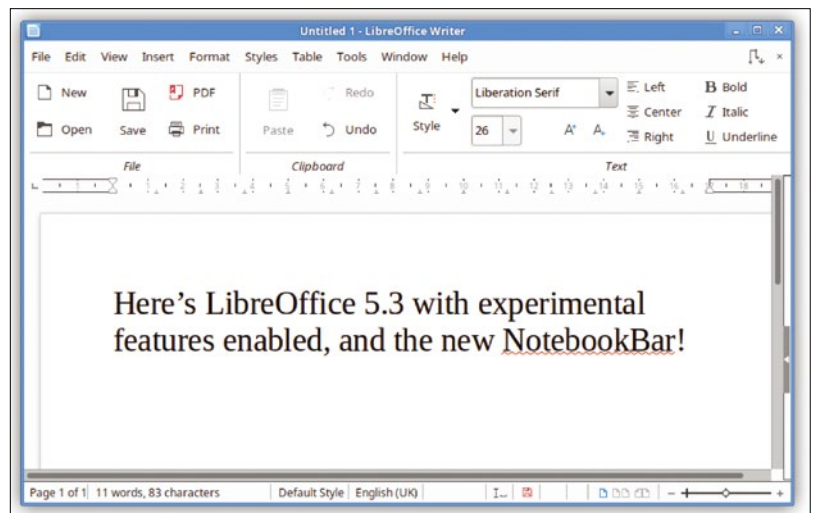


Figure 3: Another NotebookBar layout is "contextual groups," where buttons change depending on what you've clicked.

the NotebookBar, including a roadmap for upcoming releases, see the TDF Wiki [4].

Command-Line Tricks

You probably start LibreOffice via your desktop environment or window manager, clicking a button to start the suite as a whole or one of its modules (e.g., Writer or Calc). But you can run it from the command line as well with some useful extra options. First of all, the exact command you use will vary depending on your distro and how you installed LibreOffice. If you're using your distro's own packages, you can usually start it with `soffice` (referring to StarOffice), `loffice`, or `libreoffice`. However, if you've installed using the packages on LibreOffice's own website, you'll need to add the version number, such as `libreoffice5.3`.

To start the suite with Writer being displayed immediately (rather than the Start Center), use:

```
soffice --writer
```

For Calc use `--calc`, and for Impress use `--impress`; these parameters are especially useful if you're creating new launcher buttons in your desktop or window manager. For Impress, it's possible to launch the suite and go straight into a presentation, without initially displaying the user interface, like so:

```
soffice --show filename.odp
```

This will switch to full-screen mode immediately, and when the presentation ends, LibreOffice is closed. This feature is handy if you give a lot of presentations, in that you can link them to launchers on your desktop or window, and have them shown seamlessly in just a couple of clicks (without having to fiddle around in the GUI for a few seconds). Another related parameter is `--view`,

which shows a document in read-only mode – so no editing is allowed.

If you've been unlucky and had LibreOffice crash on you, the next time it's started, it will offer to (try to) recover lost data. You can keep this from happening by passing the `--norestore` option. Similarly, if you're having major problems even starting LibreOffice (e.g., due to an issue with extensions, hardware acceleration, or your user profile) then you can activate the suite's safe mode using `--safe-mode`. This pops up a dialog box – like in Figure 4 – from which you can disable certain options or even reset to “factory settings” (i.e., like a completely fresh installation).

If you want to print a bunch of files without tediously having to open them all up and click Print for each one, do this:

```
soffice -p doc1.odt doc2.odt
```

Note the single dash before `p` here. Supply as many documents as you want; they will all be sent to the default printer. If you have multiple printers and want to specify one, use the `--pt` option followed by the name of the printer, and then the document(s) you want to print.

One of the biggest time-savers when using LibreOffice from the command line is batch conversion of documents. Consider this, for example:

```
soffice --convert-to pdf *.odt
```

That generates PDF versions of all Open Document Format Text (ODT) files in the current directory. If you have a bunch of Microsoft Word documents and want to convert them to LibreOffice's native format, use this:

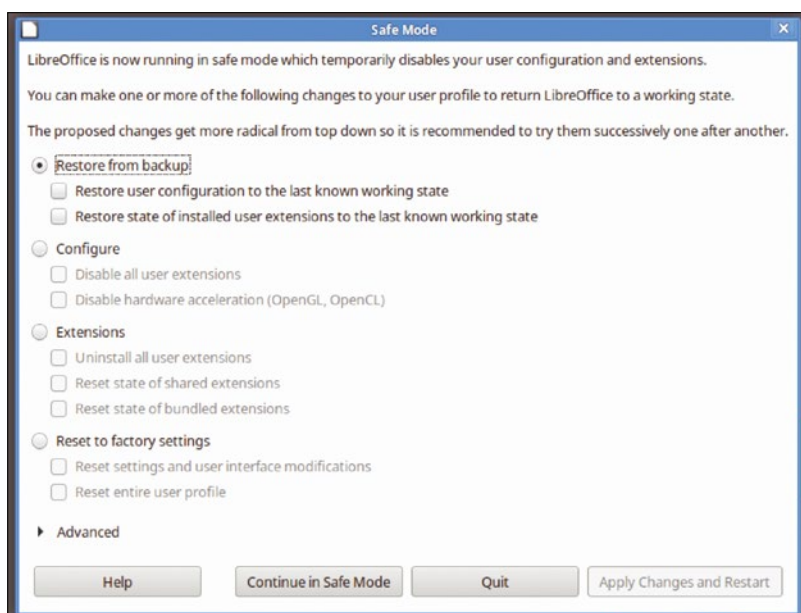
```
soffice --convert-to odt *.docx
```

Now, LibreOffice includes import and export filters for a huge range of file formats, and you can use many of these in your batch conversion jobs. Unfortunately, however, the documentation listing these formats is lacking – so you have to look in the right places. Suppose we want to convert an ODF document created by LibreOffice Writer into Microsoft Word 97 format. The command to use is this:

```
soffice --convert-to doc:"MS Word 97" file.odt
```

But how do we know to use this exact command? The trick is to look at the LibreOffice source code tree, and specifically the list of file format filters [5]. On that web page, you'll see a file called `MS_Word_97.xcu` – so you know a filter exists for Word 97 files. If you click it, you can

Figure 4: LibreOffice 5.3 includes a new Safe Mode, which temporarily disables settings and extensions, to help fix problems.



see this line toward the top of the file, following the first block of comments:

```
<node oor:name="MS Word 97" oor:op="replace">
```

This tells us that “MS Word 97” is the name we should use when applying the filter. (It’s also the same as the filename, but without underscores – but it’s always worth checking the content of the filter file as well.)

With this trick, supplying the document extension followed by the filter name in quotes, you have access to a wide range of file formats. If you’re working with large collections of legacy documents and need to shift them from one format to another, this command can be a godsend. Yes, it’s a bit fiddly having to poke into the LibreOffice source code to explore this, but it works. And, if you want to contribute back to the suite, an effort to document this would be very welcome by the dev team!

Coming Up in LibreOffice 5.4

LibreOffice follows a time-based release schedule, so updates with new features (as opposed to bug

LibreOffice Online

Something else worth keeping an eye on is LibreOffice Online (see Figure 5). As the name suggests, this is a version of the suite that can run inside a web browser, with the content being served over the network. LibreOffice Online has been in development for a few years, but has recently made major strides forward thanks to Collabora, which specializes in long-term supported versions of the suite (e.g., for government and enterprise usage).

Currently, LibreOffice Online looks and feels like a trimmed-down version of the suite; it offers most essential formatting features, but it lacks every single menu item, dialog, and checkbox that you’d find in the regular desktop edition. Still, because it uses the same underlying document layout engine as desktop LibreOffice, Open Document Format (ODF) files should be displayed identically when you’re switching between the browser and native app.

LibreOffice Online can be integrated with Nextcloud for a fully FOSS document sharing and collaboration system. It still needs some work, especially in terms of performance, but it’s coming on well – you can get access to a demo on Collabora’s website [6]. Note that The Document Foundation has no plans to provide hosted versions of LibreOffice Online, because it would require a vast amount of resources if uptake was quick. Instead, the software is available for third parties to install and run on servers.

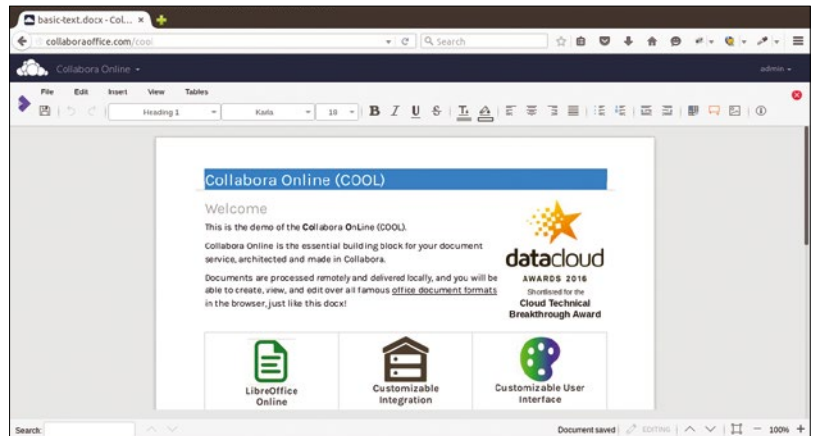


Figure 5: LibreOffice Online is making good progress, although it needs some work on performance.

fix revisions) are released every six months. Version 5.4 of the suite is due in late July or early August 2017 and looks to be a fairly conservative version compared to 5.3, which was jam-packed with new goodies.

At the time of writing, there were still a few months of development ahead, but already some useful changes have been implemented: new context menu items for editing sections, headers and footers in Writer; stored settings for CVS export in Calc (so you don’t have to correct them every time you export a file); and many improvements to EMF+ vector images import (used by Microsoft Office file formats).

Additionally, there are some welcome performance improvements to LibreOffice Online (see the “LibreOffice Online” box), along with the usual bevy of bug fixes. If you want to try a development snapshot of what will become LibreOffice 5.4, you can grab the latest RPM or .deb packages (or compile from source if you’re particularly daring) via the dev-builds section of the LibreOffice website [7]. In general, snapshots can be installed alongside stable releases, so you can happily use LibreOffice 5.3 for real work while testing out the newer version. ■■■

Info

- [1] LibreOffice design community: <http://www.libreoffice.org/community/design/>
- [2] MUFFIN: <https://blog.documentfoundation.org/blog/2016/12/21/the-document-foundation-announces-the-muffin-a-new-tasty-user-interface-concept-for-libreoffice/>
- [3] The restrictions of toolbars: <https://design.blog.documentfoundation.org/2016/12/21/evolving-past-the-restrictions-of-toolbars/>
- [4] TDF Wiki: <https://wiki.documentfoundation.org/Development/NotebookBar>
- [5] File format filters: <https://cgit.freedesktop.org/libreoffice/core/tree/filter/source/config/fragments/filters>
- [6] Collabora demo: https://www.collaboraoffice.com/collabora_online_demo/
- [7] LibreOffice dev-builds: <http://dev-builds.libreoffice.org/pre-releases/>

FEATURED EVENTS

Users, developers, and vendors meet at Linux events around the world. We at *Linux Magazine* are proud to sponsor the Featured Events shown here.

For other events near you, check our extensive events calendar online at <http://linux-magazine.com/events>.

If you know of another Linux event you would like us to add to our calendar, please send a message with all the details to events@linux-magazine.com.



ISC High Performance

Date: June 18–22, 2017

Location: Frankfurt, Germany

Website: <http://isc-hpc.com/>

The 32nd conference in the ISC High Performance series focuses on HPC technological development, its application in scientific fields, and its adoption in commercial environments. The conference offers 12 HPC topics grouped under three categories: systems, applications, and emerging topics. You can also look forward to ISC tutorials, workshops, and the exhibition.

SUSECON 2017

Date: September 25–29, 2017

Location: Prague, Czech Republic

Website: <http://www.susecon.com>

SUSECON 2017 features exceptional technical content presented by SUSE employees, customers, partners, and community enthusiasts. SUSECON focuses on helping you find ways to build and define your future to provide a flexible and efficient infrastructure. The dynamic and entertaining keynote speeches inspire, inform, and invigorate.

Linux Kernel Summit

Date: October 24–27, 2017

Location: Prague, Czech Republic

Website: <http://events.linuxfoundation.org/events/linux-kernel-summit>

The annual Linux Kernel Summit brings together core kernel developers to discuss the state of the existing kernel and plan the next development cycle. New in 2017 are four days of sessions and workshops opened to a larger group of developers, along with the half-day, invitation-only Maintainer Summit.

EVENTS

ISC High Performance (ISC 2017)	June 18–22	Frankfurt, Germany	http://www.isc-hpc.com/
Golem-Konferenz – Die Quanten kommen	June 23	Berlin, Germany	https://www.golem.de/
TÜBIX	June 24	Tübingen, Germany	http://www.tuebix.org/
Deutsche Openstack Tage	June 26–28	Munich, Germany	https://openstack-tage.de/
AnDevCon	July 17–19	Washington, DC	http://www.andevcon.com/
Debconf 17	August 6–12	Montreal, Canada	https://debconf17.debconf.org/
InterDrone	September 6–8	Las Vegas, Nevada	http://www.interdrone.com/
15. Kieler Open-Source und Linux-Tage	September 15–16	Kiel, Germany	http://kilux.de/
WiSTEM 2017	September 11–12	San Francisco, CA	http://www.womeninstemconference.com/
Storage Developer Conference (SDC)	September 11–14	Santa Clara, California	http://www.snia.org/events/storage-developer
14th Annual 2017 HPC for Wall Street, Cloud, AI & Data Centers	September 18	New York, NY	http://www.flagmgmt.com/hpc/
SUSECON 2017	September 25–29	Prague, Czech Republic	http://www.susecon.com/
JAX London	October 9–12	London, UK	https://jaxlondon.com/
Open Source Summit Europe	October 23–25	Prague, Czech Republic	http://events.linuxfoundation.org/events/open-source-summit-europe
Linux Kernel Summit	October 24–27	Prague, Czech Republic	http://events.linuxfoundation.org/events/linux-kernel-summit
Mesoscon Europe	October 25–27	Prague, Czech Republic	http://events.linuxfoundation.org/events/mesoscon-europe
SC17	November 12–17	Denver, Colorado	http://sc17.supercomputing.org/

CALL FOR PAPERS

We are always looking for good articles on Linux and the tools of the Linux environment. Although we will consider any topic, the following themes are of special interest:

- System administration
- Useful tips and tools
- Security, both news and techniques
- Product reviews, especially from real-world experience
- Community news and projects

If you have an idea, send a proposal with an outline, an estimate of the length, a description of your background, and contact information to edit@linux-magazine.com.



AUTHORS

Erik Bärwaldt	44
Swapnil Bhartiya	8
Bruce Byfield	62, 64
Joe Casad	3, 14, 28
Mark Crutch	69
Ben Everard	69, 76, 90
Andrew Gregory	71
Sebastian Grüner	12
Jon "maddog" Hall	75
Kristian Kißling	12
Klaus Knopper	40
Charly Kühnast	52
Vincent Mealing	69
Graham Morrison	84
Simon Phipps	70
Dmitri Popov	58
Mike Saunders	72, 92
Mike Schilli	48
Tim Schürmann	34
Mayank Sharma	54
Valentine Sinitsyn	78
Richard M. Stallman	20
Sam Williams	20

The technical level of the article should be consistent with what you normally read in *Linux Magazine*. Remember that *Linux Magazine* is read in many countries, and your article may be translated into one of our sister publications. Therefore, it is best to avoid using slang and idioms that might not be understood by all readers.

Be careful when referring to dates or events in the future. Many weeks could pass between your manuscript submission and the final copy reaching the reader's hands. When submitting proposals or manuscripts, please use a subject line in your email message that helps us identify your message as an article proposal. Screenshots and other supporting materials are always welcome.

Additional information is available at:

http://www.linux-magazine.com/contact/write_for_us.

CONTACT INFO

Editor in Chief

Joe Casad, jcasad@linux-magazine.com

Managing Editor

Rita L Sooby, rsooby@linux-magazine.com

Localization & Translation

Ian Travis

News Editor

Swapnil Bhartiya

Copy Editors

Amber Ankerholz, Amy Pettle

Layout

Dena Friesen, Lori White

Cover Design

Dena Friesen, Lori White

Cover Image

© Valerii Matviienko, 123RF.com

Advertising – North America

Ann Jesse, ajesse@linuxnewmedia.com
phone +1 785 841 8834

Advertising – Europe

Brian Osborn, bosborn@linuxnewmedia.com
phone +49 89 99 34 11 48

Publisher

Brian Osborn, bosborn@linuxnewmedia.com

Marketing Communications

Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
616 Kentucky St.
Lawrence, KS 66044 USA

Customer Service / Subscription

For USA and Canada:
Email: cs@linuxpromagazine.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)
Fax: 1-785-856-3084

For all other countries:
Email: subs@linux-magazine.com
Phone: +49 89 99 34 11 67

www.linuxpromagazine.com – North America
www.linux-magazine.com – Worldwide

While every care has been taken in the content of the magazine, the publishers cannot be held responsible for the accuracy of the information contained within it or any consequences arising from the use of it. The use of the disc provided with the magazine or any material provided on it is at your own risk.

Copyright and Trademarks © 2017 Linux New Media USA, LLC, except "For Want of a Printer" is an excerpt from *Free as in Freedom 2.0: Richard Stallman and the Free Software Revolution*, Copyright © 2002, 2010 Sam Williams; Copyright © 2010 Richard M. Stallman.

No material may be reproduced in any form whatsoever in whole or in part without the written permission of the publishers. It is assumed that all correspondence sent, for example, letters, email, faxes, photographs, articles, drawings, are supplied for publication or license to third parties on a non-exclusive worldwide basis by Linux New Media USA, LLC, unless otherwise stated in writing.

Linux is a trademark of Linus Torvalds.

All brand or product names are trademarks of their respective owners. Contact us if we haven't credited your copyright; we will always correct any oversight.

Printed in Germany

Distributed by COMAG Specialist, Tavistock Road, West Drayton, Middlesex, UB7 7QE, United Kingdom

LINUX PRO MAGAZINE (ISSN 1752-9050) is published monthly by Linux New Media USA, LLC, 616 Kentucky St., Lawrence, KS, 66044, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 616 Kentucky St., Lawrence, KS 66044, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Ziebländstr. 1, 80799 Munich, Germany.

Issue 201 / August 2017

Network Security Toolkit

Approximate

UK / Europe
USA / Canada
Australia

JUL 08
AUG 04
SEP 04

On Sale Date

Who's on your network? As attackers grow more numerous, and attacks grow more sophisticated, you'd better get familiar with the tools you'll need for keeping watch.

Preview Newsletter

The Linux Magazine Preview is a monthly email newsletter that gives you a sneak peek at the next issue, including links to articles posted online.

Sign up at: www.linux-magazine.com/newsletter

Celebrating 25 Years of Linux!

ORDER NOW!

Get 7 years of *Ubuntu User*

ON ONE DVD!



THE COMPLETE

UBUNTU
 **user**
ARCHIVE



Over
3,000
PAGES!
7 GREAT YEARS
OF UBUNTU
USER

Searchable DVD!

All Content Available in Both HTML and PDF Formats



Ubuntu User is the only
magazine for the
Ubuntu Linux Community!



Order Now! Shop.linuxnewmedia.com

SUPERMICRO®

BigTwin®



Front View

The IT Industry's Highest Performing
Twin Multi-Node System

SYS-2028BT-HNR+/HTR+



Rear View

2U Multi-Node System Supporting

- **A Full Range of Processors** Up to the highest performing 205 watt Intel® Xeon® Processor E5-2600 v4/v3 product families
- **Maximum Memory** Full 24 DIMMs of memory per node
- **All-Flash NVMe** 24 NVMe or Hybrid NVMe/SAS3 drives
- **Double the I/O Capacity** 2 PCI-E 3.0 x16 slots and 1 SIOM slot



Intel Inside®. Powerful Productivity Outside.



Learn more at www.supermicro.com/BigTwin

©Super Micro Computer, Inc. Specifications subject to change without notice. Intel, the Intel logo, Xeon, and Xeon Inside are trademarks or registered trademarks of Intel Corporation in the U.S. and/or other countries.